Парадигма развития науки

Методологическое обеспечение

А.Е. Кононюк

КОНСАЛТОЛОГИЯ

ОБЩАЯ ТЕОРИЯ КОНСАЛТИНГА

Книга 3

Киев
Освита Украины
2011

УДК 51 (075.8)
ББК В161.я7
К65.

Рецензент: *Н.К. Печурин* - д-р техн. наук, проф. (Нацинальный авиационный университет).

Кононюк А.Е.
К65 Консалтология. Общая теория консалтинга. К. 3
К.: Освита Украины. 2011. - 520 с.
ISBN 978-966-7599-50-8

Настоящая работа посвящена вопросам развития общей теории консалтинга с использованием научных методов формирования рекомендаций для решения задач консультируемых проблем. Сформулированы основные положения построения автоматизированных консультационных процессов. Рассмотрены принципы построения систем автоматизарованного консультирования (САК). С позиций пользователя (лица, формирующего рекомендации) изложены основные положения, связанные с разработкой, исследованием и реализацией сформированных рекомендаций по решению задач консультируемых проблем различных проблемных областей.

Книга предназначена для научных работников, магистров, аспирантов, докторантов соответствующих специальностей.

ББК В161.я7

ISBN 978-966-7599-50-8

Оглавление

5. Системы автоматизированного консультирования – средства реализации консультационных процессов и формирования рекомендаций

В первой и второй книгах общей теории консалтинга мы детально рассмотрели вопроссы, связанные с разработкой и моделированием консультационных процессов, реализация которых обеспечивает формирование рекомендаций по решению задач консультируемых проблем. Во трегьей и четвертой книгах общей теории консалтинга будут рассмотрены средства, реализующие консультационные процессы, результатом которых являются сформированные рекомендации по решению задач консультируемых проблем. Этими средствами будут выступать объектные системы, представляемые как системы автоматизированного консультирования (САК).

5.1. Общая теория систем и ее связь (взаимодействие) с общей теорией консалтинга

Как мы определили ранее, целью общей теории консалтинга является создание единой методологии исследования и создания средств формирования рекомендаций для решения задач консультируемой проблемы любой проблемной области.

Такая формулировка цели общей теории консалтинга находится в полной корреляции с понятием цели общей теории систем.

Прежде чем приступить к изложению основных положений, которые могут (помогут) подтвердить тесное взаимодействие общих теорий систем и консалтинга, рассмотрим основные проблемы, которые существуют и которые необходимо решить при рассмотрении вопросов взаимосвязей теорий систем и консалтинга.

Типы систем консультирования. Фундаментальная проблема теории консалтина состоит в выяснении законов организации, поведения и развития реальных систем консультирования. Эту проблему можно решить, построив абстрактную систему консультирования, являющуюся математическим описанием системы консультирования и в достаточной мере адекватно отражающей ее организацию, поведение и развитие.

Все системы, независимо от их профильной ориентации (принадлежности), разделим на ***два тип***.

К ***первому типу систем*** отнесем системы, которые для эффективного выполнения своих функций, испытывают в отдельных

случаях потребность в рекомендациях по выполнению этих функций. Такие системы будем называть ***консультируемые проблемы*** (КП). Другими словами - это такие системы, которые нуждаются в консультировании (в рекомендациях по решению задач консультируемых проблем)

Ко ***второму типу систем*** отнесем системы, которые выполняют (реализуют) консультационный процесс формирования рекомендаций. Такие системы будем называть ***системами консультирования*** (СК). Другими словами - это такие системы, которые выполняют функции консультантов по формированию рекомендаций для решения задач консультируемой проблемы.

Системы обоих типов взаимодействуют, как правило, в рамках одной системы, в которую в ранге подсистем входят система, представлевляемая консультируемой проблемой (КП) и системой консультирования (СК). Такую систему будем называть ***комплексной системой консультирования*** (КСК), или, ***консультационным комплексом*** (КК). Структура КСК приведена на рис. 5.1.

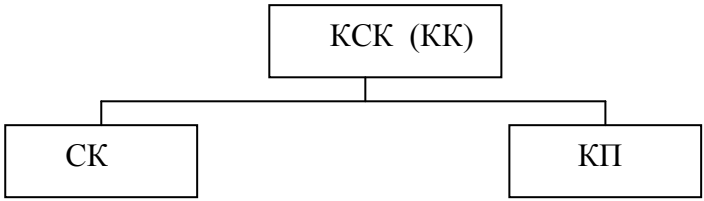


Рис. 5.1. Структурная схема консультационного комплекса

Отдельные системы консультирования могут не иметь жестких связей с консультируемыми проблемами, т.е. выполнять свои функции, будучи в сравнительно независимом (свободном) состоянии. Такие системы мы будем называть ***автономными системами консультирования*** (АвСК).

В настоящей работе речь будет идти о математических описаниях и исследованиях КП, СК, КСК и АвСК. Исходя из этого, рассматриваемая часть консалтинговой теории будет носить название «***математическая теория консалтинга***». Она строится на основе понятия системы, под которой будем понимать абстрактную систему консультирования или консультируемую проблему как некоторую аналогию или модель определенного класса, в достаточной мере, тождественную реальной системе консультирования или

консультируемой проблеме. Такую теорию можно рассматривать как ***теорию абстрактных моделей*** СК и КП. Отсюда основной задачей нашей теории является ***построение абстрактных моделей СК и КП и исследование поведения и свойств этих систем на уровне абстракции.***

Сначала рассмотрим системы, которые будем рассматривать как консультируемые проблемы. На рис. 5.2 в качестве примера приведена схема одной из множества консультируемых проблем, а именно: системы управления технологическим процессом.

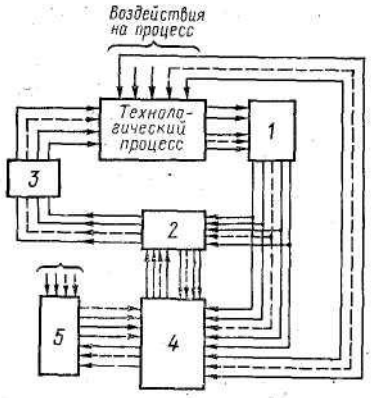


Рис. 5.2. Схема консультируемой проблемы
(на примере системы управления технологическим процессом)

На этой схеме сведения о протекании технологического процесса воспринимаются системой датчиков информации *1*; информация поступает на вход вычислительной машины регулятора *2,* где вырабатывается воздействие на исполнительные устройства *3,* изменяющие параметры технологического процесса. Таким образом, существует контур регулирования, состоящий из собственно технологического процесса, комплекса датчиков, вычислительной машины и исполнительных устройств. Реальное протекание технологического процесса, как правило, зависит от многих факторов: сырье может иметь различное качество, происходит износ элементов оборудования и т. д. Поэтому на основе сведений о протекании технологического процесса и воздействий на него вычислительная машина *4* в соответствии с выбранным критерием эффективности рассчитывает программу и параметры регулятора *2,* оптимизирующие протекание процесса и формирует рекомдации по регулированию протекания технологического процесса. Вычислительная машина *4* может не

участвовать в управлении процессом, но тогда, естественно, из-за переменности его характеристик критерий эффективности не будет удовлетворяться, хотя система останется работоспособной. Для контроля за работой оборудования и для формирования рекомендаций по результатам этого контроля принимает участие человек-оператор (лицо, формирующее рекомендации (ЛФР)) *5*. Это же ЛФР может быть одновременно и лицом, принимающим решение (ЛПР), которое принимает решения в сложной ситуации .

Анализируя данную систему, можно заметить, что она представляет собой совокупность подсистем и объектов, которая объединена информационным процессом. Эта совокупность действует таким образом, чтобы достигался определенный результат, например, обеспечивалась бы эффективность реализации технологического процесса. Задача описываемой теории в данном случае состоит в том, чтобы сформировать рекомендации по построению такой абстрактной модели системы управления технологическим процессом, с помощью которой, путем реализации консультационного процесса формирования рекомендаций, на уровне абстракции можно было бы исследовать влияние различных, рекомендуемых системой консультирования, факторов по эффективности их действия и, на основании сформированных рекомендаций, синтезировать консультационные параметры, реализация которых обеспечила бы выполнение процесса управления в заданных показателях эффективности.

Такая система консультирования фактически является комплексной системой консультирования, так как в ее составе находится подсистема консультирования, которая, выполняя (реализуя) консультационный процесс по формированию рекомендации по управлению данным технологическим процессом, помогает ЛПР принимать правильные (оптимальные) решения.

Следует также заметить, что данная система консультирования имеет иерархическое строение. На первом уровне иерархии — контур регулирования, включающий собственно процесс и элементы *1, 2, 3.* На втором уровне — вычислительная машина *4,* которая «осмысливает» деятельность первого уровня и формирует рекомендации по управлению им так, чтобы удовлетворялся критерий эффективности, охватывающий большее число показателей, чем это можно обеспечить на первом уровне.

Рассматриваемая теория носит в основном математический характер, и чаще всего исходным материалом для нее служат задачи организации человеческой деятельности. Постараемся сформулировать некоторые

общие принципы организации систем консультирования на основе структуры многоуровневой системы.

Принципы организации систем консультирования. Первый принцип — принцип избытка (недостатка) взаимодействий.

Рассмотрим двухуровневую систему консультирования, показанную на рис. 5.3.

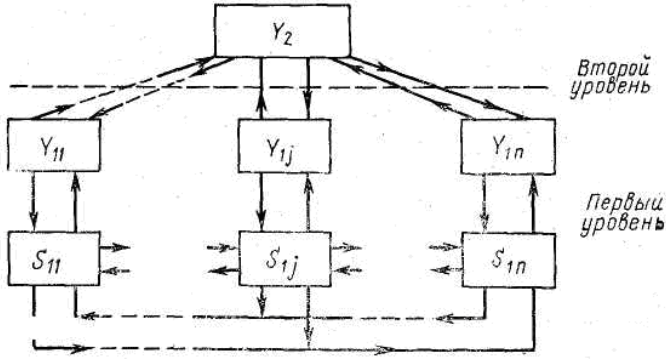


Рис. 5.3. Схема двухуровневой системы консультирования.

Эта система консультирования состоит из взаимодействующих элементов целенаправленных систем (обладающих целенаправленным действием). Первый уровень включает S_{1j} (j = l, 2, ..., n) объектов, управляемых системами консультирования Y_{1j}. На втором уровне— одна управляющая системами консультирования Y_2. Пусть нас интересует вопрос о том, как должны быть связаны между собой эти два уровня, чтобы вся системами консультирования составляла некое гармоничное целое, или более конкретно: какую информацию о поведении первого уровня должен получать второй уровень для того, чтобы осуществлять процесс формирования рекомендаций по координированию активности объектов на нижнем уровне, и как должна осуществляться эта координация?

Сделаем некоторые допущения. Предположим ради простоты, что цели, поставленные перед каждым уровнем системы консультирования, не противоречат друг другу, а скорее дополняют в том смысле, что достижение всех целей на первом уровне обеспечивает достижение цели второго уровня. В этом случае имеется основа для гармонического поведения системы консультирования в целом, и главное препятствие на пути улучшения качества формирования рекомендаций заключается в том, что системы консультирования первого уровня не имеют сведений о будущем взаимодействии между объектами этого же уровня. А именно объекты

первого уровня не могут полностью предсказать, как на них повлияют действия других объектов этого же уровня, поскольку для этого потребовалась бы непомерно большая способность объектов первого уровня к восприятию сформированных рекомендаций и на их основе к принятию решений и вычислениям. Чтобы обеспечить гармоническое поведение системами консультирования в целом, можно организовать обмен информацией между уровнями следующим образом: система консультирования второго уровня воздействует на цель каждой подсистемы первого уровня путем реализации сформированных рекомендаций на изменение выбранного надлежащим образом параметра координации.

Пусть параметры координации входят в функции качества формирования рекомендаций систем первого уровня. Решение вопроса о том, на основании каких рекомендаций второй уровень должен обеспечить соответствующие значения параметров координации, основывается на информации о взаимодействиях, поступающей от систем консультирования первого уровня. Стараясь достичь своих собственных целей, каждая из систем консультирования первого уровня рассматривает, руководствуясь сформированными рекомендациями, переменные, описывающие взаимодействия, как подлежащие изменению в дополнение к своей собственной локальной переменной управления. При этом задача достижения цели отдельного объекта первого уровня сводится к стремлению, на основании сформированных рекомендаций, либо увеличить, либо уменьшить существующие в данный момент взаимодействия. Соответствующие запросы на формирование рекомендаций передаются затем на второй уровень и используются им как основа для выбора значений параметров координации, т. е. система консультирования второго уровня так регулирует параметры координации в функциях цели первого уровня, выполняя сформированные рекомендации, что запросы на увеличение взаимодействия уравновешиваются с запросами на уменьшение взаимодействий, если не происходит дальнейших изменений в окружающей среде (консультируемой проблеме). Иначе, взаимодействия на первом уровне уравновешиваются в том смысле, что объекты осуществляют их в соответствии со сформированными рекомендациями, которые устанавливаются данными значениями параметров координации. Следовательно, для того чтобы иметь возможность координировать поведение систем консультирования первого уровня, система консультирования второго уровня, для формирования эффективных рекомендаций, нуждается в информации только двух типов:

а) запросы на формирование рекомендации по изменению во взаимодействиях;

б) зависимость параметров цели от изменений во взаимодействиях.

Это значительно меньше того количества, которое потребовалось бы для обеспечения полного функционирования только со стороны второго уровня.

Математические результаты здесь можно получить в предположении, что цели на обоих уровнях достигаются в результате поиска оптимума.

При построении моделей многоуровневых систем консультирования можно ожидать, что будут заданы менее жесткие требования, например, что координация осуществляется только тогда, когда рассогласование между запрашиваемыми рекомендациями по взаимодействию превышает некоторый определенной величины порога, и, следовательно, что цель объектов второго уровня заключается только в поддержании приближенного равновесия взаимодействий с заданной точностью. Тогда можно утверждать, что гармоническое функционирование двухуровневой системы консультирования может быть обеспечено, если в ней имеются:

а) совокупность переменных (или свойств) объектов первого уровня, указывающих величину избытка или недостатка взаимодействий, определяемую достижением локальных целей;

б) механизм, возможно распределенный по всей системе консультирования, который изменяет функции достижения цели объектов первого уровня таким образом, чтобы равновесие взаимодействий поддерживалось с заданной точностью.

Второй принцип — ***принцип оптимальной связи.*** Остановимся на проблеме связи между системами консультирования, находящимися на одном и том же уровне. Едва ли нужно подчеркивать важность такой проблемы связи для сложных система консультирования. Например, имеются данные, свидетельствующие о зависимости определенных типов элементов от нарушения связей между элементами. Межэлементные связи, по-видимому, специфичны для разных элементов (или блоков), и один из важных вопросов состоит в том, насколько активными должны быть эти связи для нормального функционирования всей совокупности элементов в составе системы консультирования. Ряд исследований этой проблемы показали, что для определенных структур многоуровневых систем консультирования существует «оптимальная» степень связи между подсистемами, находящимися на одном и том же уровне. Оптимальность понимается в том смысле, что увеличение степени связи облегчает достижение цели первого уровня, одновременно ослабляя, однако, организацию в целом,

т. е. снижает общее качество система консультирования; при этом появляется тенденция к дезинтеграции. Это позволяет сделать следующее заключение: ***чрезмерно большая степень связи так же, как и нарушение связи, может привести к дезинтеграции многоуровневой системы консультирования***.

5.2. Обобщенная структура системы консультирования

5.2.1. Структура системы консультирования

Определения абстрактной системы консультирования. Определение ***абстрактной системы консультирования*** может быть построено на основе использования различного математического аппарата, так как в зависимости от вида методов реализации консультационных процессов в системах консультирования тот или иной аппарат может дать наиболее адекватное описание. Поэтому в данном случае мы будем формулировать определения абстрактной системы консультирования, используя математическую логику, теорию множеств и теорию дифференциальных уравнений.

Приведем определение абстрактной системы консультирования, использующее лингвистическую формулировку.

Введем, прежде всего, некоторые вспомогательные понятия. Начнем с понятия *высказывания* на некотором языке *L*. Таким языком может быть любой естественный язык, например русский, некоторый машинный язык или любой формальный рекурсивный язык. *Высказыванием F* на языке *L* называется *предложение,* построенное по правилам грамматики этого языка, но такое, что истинность этого предложения не вытекает из самого его содержания. Иначе говоря, предполагается, что высказывание содержит некоторые свободные переменные и, следовательно, может оказаться истинным для некоторых значений этих переменных. Предположим теперь, что имеется некоторое множество *K* таких высказываний. Если некоторое подмножество этих высказываний принимается истинным, то оно (это подмножество) определяет *теорию T* относительно *K*. А именно, теория предполагает, что только высказывания из подмножества *M* всегда истинны, а истинность остальных остается неопределенной.

Предположим теперь, что высказывания из *M* таковы, что свободные переменные в них образуют *формальные объекты*, под которыми

понимается ***абстрактное представление объекта, отражающее некоторые его реальные свойства***. Такие высказывания будем называть *правильными*. Так как в подмножестве M свободные переменные представляют собой формальные объекты, то, следовательно, высказывания из M адекватно отражают некоторые свойства реальной системы консультирования. Тогда приведем формулировку следующего определения.

Определение 1. ***Абстрактной системой консультирования*** *будем называть множество правильных высказываний.*

Пусть теперь формальные объекты определяются явным образом, а не с помощью концептуальных классов высказываний, как это было в определении 1, поскольку приходилось формулировать посредством языка L свойства, которые определяют чисто интуитивным (содержательным) путем совокупность реальных объектов и понятий в системе консультирования.

Начнем с рассмотрения семейства множеств X_1, ... X_j, ... X_n (j = 1, 2, ..., n). Пусть каждое из этих множеств определяет некоторый формальный объект, а именно, формальный объект, соответствующий множеству X_j, может принять вид любого элемента из этого множества. Элементы множества X_j можно называть *значениями объекта* в множестве.

Образуем теперь прямое произведение X семейства множеств X_j.

$$X = X_1 \times X_2 \times \ldots \times X_j \times \ldots \times X_n, \qquad (5.1)$$

т. е. множество X упорядоченных конечных последовательностей $\{(x_1, \ldots, x_j, \ldots x_n)\}$, где $x_j \in X_j$. Поскольку формальный объект отражает свойства некоторого реального объекта, то можно предположить, что некоторые из упорядоченных конечных последовательностей адекватно отражают свойства реальной системы консультирования.

Определение 2. ***Абстрактной системой консультирования*** *будем называть некоторое собственное подмножество X_s, $X_s \subset X$.* В действительности некоторое собственное подмножество прямого произведения множеств X определяет отношение между формальными объектами X_1, ..., X_j, ..., X_n (j = 1, 2, ..., n). Тогда ***абстрактной системой консультирования*** *можно назвать некоторое отношение R, определенное на произведении X,* т. е. абстрактная система консультирования определяется заданием множества

$$X = X_1 \times X_2 \times \ldots \times X_j \times \ldots \times X_n,$$

и некоторого множества отношений

$$R = \{R_1, \ldots, R_n\}.$$

Именно множество отношений R позволяет выделить некоторое подмножество $X_s \subset X$.

Определение 3. Пусть элементами множества X_i будут некоторые функции времени, т. е. $x_i(t) \in X_i(t)$. Тогда можно предположить, что ***абстрактная система консультирования*** *отображает некоторые динамические консультационные процессы, протекающие в реальной системе консультирования*. Действительно, например, для линейной динамической системы консультирования можно записать

$$x_2(t) = \int_{-\infty}^{t} k(\tau) x_1(t-\tau) d\tau, \qquad (5.2)$$

где $x_2(t)$—выходной консультационный процесс, или реакция системы консультирования (сформиованные рекомендации), $x_1(t)$—входной консультационный процесс, или входное консультационное воздействие, $k(\tau)$—импульсная переходная функция системы консультирования. В общем случае $x_1(t)$ может быть элементом множества $X_1(t)$ и, следовательно, учитывая интегральное преобразование, будем иметь $x_2(t) \in X_2(t)$. Тогда можно записать что

$$X_2(t)\ R\ X_1(t)\ \ (t \in T). \qquad (5.3)$$

В этом выражении R — отношение, которое является некоторым множеством, поскольку параметры импульсной переходной функции $k(\tau)$ могут представлять собой наборы дискретных значений или быть функциями времени.

Нелинейную динамическую систему консультирования можно описать с помощью ряда Вольтерра

$$x_2(t) = h_0 + \int_{\eta} h_1(\tau) x_1(t-\tau) d\tau + \int_{\eta}\int_{\eta} h_2(\tau_1,\tau_2) x_1(t-\tau_1) \times x_1(t-\tau_2) d\tau_1 d\tau_2 + \ldots + \int_{\eta}\cdots\int_{\eta} h_n(\tau_1,\ldots,\tau_n) \times$$
$$\times\, x_1(t-\tau_1)\ldots x_1(t-\tau_n) d\tau_1 \ldots d\tau_n \qquad (5.4)$$

где η – область интегрирования, т.е. область, на которой определена функция $x_1(t)$, являющаяся входным сигналом системы консультирования, а h_n (n=0,1,2,…) – ядра ряда Вольтера.

В этом случае можно также записать соотношение, аналогичное (5.3).

В этих рассуждениях мы пришли к понятию ***абстрактной динамической системы консультирования***, которую теперь можно определить следующими аксиомами:

1. *Для системы консультирования определены пространство состояний* К *и множество Т моментов времени, в которых определено поведение системы консультирования. Здесь* К *— некоторое топологическое пространство, а Т — упорядоченное топологическое пространство, являющееся подпространством пространства вещественных чисел.*

2. *Для системы консультирования определено некоторое топологическое пространство* Ω *функций времени, определенных на* T *и называемых допустимыми входными сигналами системы консультирования или воздействиями на систему консультирования.*

3. *Для произвольного начального момента* t_0 *из* T*, произвольного начального состояния* x_0 *из* К *и произвольного входного сигнала «*u*» из* Ω*, определенного для* $t \geq t_0$*, все будущие состояния системы системы консультирования определяются видом функции перехода*

$$\varphi: \Omega \times T \times T \times К \rightarrow К,$$

что символически можно записать так:

$$\varphi_u(t; t_0, x_0) = x_t.$$

Эта функция определена только для $t \geq t_0$*. Более того, для любых* $t_0 \leq t \leq t_2$ *из* T*, любых* x_0 *из* К *и любых фиксированных «*u*» из* Ω*, определенных на* $[t_0, t] \cap T$*, справедливы соотношения*

$$\varphi_u(t_0; t_0, x_0) = x_0. \qquad (5.5)$$

$$\varphi_u(t_2; t_0, x_0) = \varphi_u(t_2; t_1, \varphi_u(t_1; t_0, x_0)) \qquad (5.6)$$

Помимо этого, система консультирования должна быть физически возможной, т. е. если «u*», $v \in \Omega$ и «*u*» = v на* $[t_0, t] \cap T$*, то необходимо, чтобы*

$$\varphi_u(t; t_0, x_0) \equiv \varphi_v(t; t_0, x_0). \qquad (5.7)$$

4. *Каждый выходной сигнал или реакция системы консультирования является некоторой вещественной функцией* ψ*, определенной на произведении* $T \times К$.

5. *Функции* φ *и* ψ *непрерывны относительно топологий, выбранных на* К, T и Ω.

Структура системы консультирования. Понятие структуры системы консультирования является, пожалуй, одним из самых важных в теории консалтинга. Абстрактная система консультирования, например, представляет собой некоторое подмножество прямого произведения множеств, каждое из которых представляет собой формальный объект. Можно представить, что каждый формальный объект обладает определенными свойствами, но в достаточной мере простыми по сравнению со свойствами всей системы консультирования. Если теперь из совокупности формальных объектов построить некоторую «организацию», систему консультирования, притом делать это целенаправленно, то эта «организация» может проявить более сложные свойства и закономерности, чем это наблюдалось у отдельного формального объекта.

Приведем пример системы из живой природы. Известно, что человеческий мозг содержит очень большое количество нейронов, каждый из которых имеет определенные свойства. Математическая

абстракция свойств нейрона есть формальный объект, или формальный нейрон. Фактом является то, что отдельный нейрон не «мыслит». Однако совокупность нейронов, будучи соответствующим образом связанной, приводит к возникновению мысли. И именно процесс мышления возможно использовать в системах консультирования для формирования рекомендаций по решению задач консультируемой проблемы

Это, конечно, упрощенные рассуждения. На самом, деле, процесс возникновения мысли значительно сложнее. Но несомненным является то, что ***установление связи между формальными объектами (здесь формальный объект — математическое описание свойств нейрона) приводит к проявлению новых, более сложных закономерностей***. Следовательно, ***свойства*** абстрактной системы консультирования ***зависят не только от свойств отдельных формальных объектов, но и от взаимосвязи между ними***. Поэтому *сеть взаимосвязей между формальными объектами, причем такую, которая приводит к проявлению более сложных закономерностей системы консультирования по сравнению с отдельным объектом, будем называть* ***структурой абстрактной системы консультирования.***

Обратимся к определению 2 абстрактной системы консультирования. Вообще говоря, отношение R, заданное на прямом произведении множеств и по сути дела устанавливающее взаимосвязь между формальными объектами, содержит, по крайней мере, одну постоянную составляющую, имеющую некоторое конкретное значение. Поэтому R можно рассматривать как частный случай более общего отношения, в котором составляющая, о которой мы упомянули, является свободной. Например, в формальном высказывании «Он старше Бори» роль отношения играет понятие «старше». Однако приведенное отношение можно рассматривать как одну из рекомендуемых реализаций более общего высказывания «разного возраста», или, что то же, «быть старше (моложе)».

В отношениях, используемых в задачах синтеза или анализа, свободные составляющие принимают одно значение из множества рекомендуемых значений или являются одной функцией из множества функций. Можно считать, что такое отношение R определяется заданием некоторого более общего отношения и конкретным значением свободной составляющей, которую будем называть ***конституэнтой отношения:***

$$R=\{S, \xi\}, \qquad (5.8)$$

где S — структура системы консультирования, а ξ — множество конституэнт отношения. В соответствии с этим структура системы консультирования получается в результате обобщения отношения, описывающего систему консультирования, т. е. в том случае, если положить конституэнты этого отношения свободными. По существу, это означает, ***что структура S представляет собой множество взаимосвязей, а множество конституэнт выделяет подмножество взаимосвязей, имеющих смысл (например, систему обоснованных рекомендаций).*** Это обстоятельство аналогично тому, что высказывание, сформулированное на каком-либо языке, содержит некоторые свободные переменные и может оказаться истинным для некоторых значений этих переменных (см. определение 1 абстрактной системы консультирования).

Рассмотрим, каким образом можно определить ***структуру динамической системы консультирования.*** Для простоты рассуждений возьмем случай линейной динамической системы консультирования с одним входом и одним выходом, описываемой уравнением

$$y(t)=\int_{-\infty}^{t} k(\tau)x(t-\tau)d\tau, \qquad (5.9)$$

или

$$yRx. \qquad (5.10)$$

Отношение R для этой системы может быть описано, по крайней мере, двумя способами. Во-первых, конституэнтой отношения можно считать импульсную переходную функцию по $k(\tau)$, и в этом случае отношение, характеризующее систему консультирования, имеет вид

$$R = \{C, k(\tau)\}, \qquad (5.11)$$

где C — преобразование типа свертки. В этом случае структура системы консультирования определяется преобразованием C;

$$S = C, \xi = k(\tau).$$

Во-вторых, структуру системы консультирования можно получить, полагая свободными некоторые числовые параметры, определяющие $k(\tau)$. Например, для абстрактной системы консультирования, описываемой дифференциальным уравнением первого порядка

$$\Theta\frac{dy}{dt} + ky = x(t), \qquad (5.12)$$

отношение имеет вид

$$R=\left\{\frac{1}{s},\bullet,+,\Theta,k\right\}, s=\frac{d}{dt}, \quad (5.13)$$

где 1/*s*, • и + являются соответственно операциями интегрирования, умножения и сложения, а конституэнты Θ и k — параметры системы консультирования, называемые обычно ***постоянной времени*** и ***коэффициентом усиления***. В этом случае структура системы консультирования имеет вид

$$S=\left\{\frac{1}{s},\bullet,+\right\},$$

а конституэнтами отношения являются $\xi=\{\Theta, k\}$.

Итак, определение структуры системы консультирования связано с погружением отношения, описывающего систему консультирования, в некоторое множество отношений, отличающихся друг от друга только значениями некоторых конституэнт. Если это множество отношений не задано в явном виде, то, очевидно, можно выбрать различные множества, в рамках которых могло бы рассматриваться отношение интересующей нас системы консультирования. В этом случае выбор множества *S* определяется практическими соображениями. Например, если необходимо усовершенствовать уже существующую систему консультирования, то отношение этой системы консультирования нужно погрузить в такое множество отношений, чтобы выбор приемлемого отношения был возможен в результате отыскания подходящих значений для конституэнт, считающихся свободными. Выбор подходящего элемента рассматриваемого множества может быть произведен организованным образом, например с помощью какой-либо процедуры оптимизации. Однако выбор самой консалтинговой структуры, т. е. выбор соответствующего множества отношений, пока возможен лишь эвристически.

5.2.2. Общие свойства систем консультирования

Открытые и замкнутые системы консультирования. При изучении системы консультирования существенное значение имеет возможность проведения измерений тех или иных консультационных процессов таким образом, чтобы реализация этих измерений не приводила к изменению свойств системы консультирования или на результаты измерений не сказывались бы факторы, обусловленные не собственными свойствами системы консультирования, например

внешними воздействиями. В связи с этим введем понятие ***открытой*** и ***замкнутой*** системы консультирования.

Рассмотрим абстрактную систему консультирования, определяемую явным образом, т. е. воспользуемся определением 2. Подмножество X_s есть по сути дела некоторая совокупность упорядоченных наборов из n чисел. Некоторый набор из n чисел, являющийся любым элементом множества X_s, будем называть ***экземпляром***. Элементы этих наборов состоят из всех допустимых значений соответствующих формальных объектов. Разобьем множество X_s на j (j = 1,2, ..., m) подмножеств X^1_s, X^2_s ,..., X^j_s ,... , X^m_s , таких, что $X^j_s \subset X_s$ для всех j. Предположим, что задан некоторый экземпляр $x_i \in X_s$. Пусть отношение $L(x_i, X_s)$, определенное на X_s, позволяет выделить некоторое собственное подмножество $X_{si}(L_1) \subset X_s$, содержащее заданный экземпляр $x_i \in X_{si}(L_1)$. В то же время самое отношение L_1 определяет, какое из подмножеств $X_{si}(L_1) \subset X_s$ рассматривается, поскольку предпочтительными считаются только те системы консультирования, которые содержат $X_{si}(L_1)$.

Предположим теперь, что существует конечная последовательность отношений L_1, ..., L_n, что множество $X_{si}(L_1, ..., L_n)$ состоит из единственного элемента и этим элементом является экземпляр x_i. Такую последовательность отношений назовем ***эффективным процессом идентификации.***

Теперь сформулируем следующее определение:

абстрактная система консультирования $X^j_s \subset X_s$ ***называется замкнутой тогда и только тогда, когда для каждого*** $x_i \in X^j_s$ ***существует эффективный процесс идентификации.***

Таким образом, используя предпочтительный эффективный консультационный процесс идентификации, замкнутую систему консультирования X^j_s можно отличить от любой другой системы $X^j_s \subset X_s$. Следует полагать, что существование ***эффективного консультационного процесса идентификации*** — это не есть какое-либо свойство собственно системы консультирования, а есть скорее выражение характера взаимосвязи между ЛФР и самой системой консультирования. Например, отношение L_i можно рассматривать как некоторое измерение, а эффективный консультационный процесс идентификации, т. е. применение последовательности отношений L_1, ..., L_n, — как эксперимент по формированию рекомендаций, осуществленный на некоторой реальной системе консультирования, с целью, например, выявления тех или иных закономерностей формирования рекомендаций, присущих системе консультирования.

Система консультирования, для которой перечисленные выше условия для замкнутой системы консультирования не выполняются, т. е. для которой существует по крайней мере один экземпляр $x_i \in X^{\kappa}_{s}$ ***такой, что для него не существует эффективного консультационного процесса идентификации, называется открытой.***

Открытую систему консультирования X_s^k принципиально нельзя отличить от некоторой другой системы консультирования X_s^i. Вообще говоря, система консультирования становится открытой, если в предположениях, которые можно сделать об ее свойствах и проверить экспериментально, опущены какие-либо принципиально важные составляющие, например, рассматривается меньшее число формальных консультируемых объектов, чем это возможно.

Примеры открытых систем консультирования.

1. Системы консультирования, не полностью изолированные от окружающей среды (система консультирования с внешними возмущениями).
2. Системы консультирования, реагирующие на процесс формирования рекомендаций таким образом, что это вызывает существенное изменение в их поведении (самоприспосабливающиеся и самоорганизующиеся системы консультирования).
3. Системы консультирования, с которыми ЛФР взаимодействует двусторонне, т. е. воздействуя на систему консультирования, он одновременно испытывает воздействие с ее стороны (ЛФР во время формирования рекомендаций оказывается скорее «внутри» системы, чем «снаружи»).

Большинство принципиальных трудностей в теории систем консультирования связано с тем, что рассматриваемые системы являются открытыми и управление необходимо вырабатывать в условиях неопределенности. Разработка методов изучения открытых систем консультирования имеет большое значение для практики. Важно помнить, что при решении проблемы идентификации следует отличать задачу определения структуры от задачи определения отношения. Например, задачу идентификации можно решить, используя для этого заданные множества пар входных и выходных сигналов, и определить при этом отношение системы консультирования. Решение задачи разделяется на две части:

1) выбор структуры S системы консультирования;

2) описание поведения системы консультирования отношениями, принадлежащими множеству S, для определения удовлетворительных значений для конституэнт отношения ξ.

Если вторую из этих частей можно решить систематически, применяя тот или иной метод идентификации, то при решении первой части задачи приходится пользоваться интуитивными соображениями ЛФР.

Декомпозиция системы консультирования. Как с теоретической, так и с практической точек зрения, представляет значительный интерес наличие возможности разложения, или декомпозиции, системы консультирования на несколько более простых подсистем консультирования. Если обратиться к теории автоматического регулирования непрерывных систем, то задача декомпозиции здесь сводится к возможности представления передаточной функции высокого порядка некоторым числом передаточных функций элементов, называемых ***типовыми***.

Важно установить, можно ли осуществить декомпозицию системы консультирования, оставаясь на принятом уровне общности. Определим понятие ***декомпозиции***. Предположим, что система консультирования определяется с помощью отношения *n*-го порядка

$$R[X_1, \ldots, X_m], \qquad (5.14)$$

заданного на множестве *X*.

Общий метод декомпозиции опишем с помощью умножения отношений. Отношение *R* называется *произведением отношений* R_1 и R_2, если выполняется условие

$$(xRy) \leftrightarrow [(xR_1z) \cap (zR_2y)], \qquad (5.15)$$

где *z* — новый терм отношения. Общий метод декомпозиции состоит в том, чтобы представить отношение системы консультирования *R* в виде произведения двух других отношений R_1 и R_2. После того как два таких отношения найдены, систему консультирования можно представить как совокупность двух подсистем консультирования:

$$R_1[X_1, \ldots, X_i, Z].$$
$$R_2[Z, X_{i+1}, \ldots, X_n]. \qquad (5.16)$$

Рассмотрим теперь следующую задачу: каков наименьший порядок отношения подсистем консультирования, на которое можно разложить многоместное отношение *n*-го порядка, используя условие (5.15). Решение этой задачи можно получить, доказав следующую теорему.

Теорема. *Систему консультирования n-го порядка можно разложить на:*

1) *(n*—2) *трехместные отношения;*

2) *двухместные отношения имеют место тогда и только тогда, когда любое трехместное отношение, полученное в соответствии с первым утверждением, удовлетворяет условию*

$$[X_i R_j(X_{i+1}, X_{i+2})] \leftrightarrow \{(X_i R^1_j Z_j) \cap [Z_j R^2_j(X_{i+1}, X_{i+2})]\} \qquad (5.17)$$

$$Z_i = X_{i+1} \cup X_{i+2}.$$

Доказательство. Первое утверждение теоремы легко доказать, построив соответствующее разложение. Представим отношение R системы консультирования в виде произведения двух других отношений R_1 и R_2, т. е. $R = R_1R_2$,

$$[R_1(Z_1, X_2, X_3)] \cap [R_2(X_1, Z_1, X_4, \ldots, X_n)]. \quad (5.18)$$

Поскольку вместе с новыми отношениями R_1 и R_2 введен новый терм Z_1, на выбранные отношения можно не накладывать никаких ограничений, и, следовательно, такая декомпозиция возможна.

Представим теперь R_2 как произведение отношений

$$R_2 = R_3R_4. \quad (5.19)$$

тогда

$$[R_1(Z_1, X_2, X_3)] \cap [R_3(Z_2, Z_1, X_4)] \cap [R_4(Z_2, X_1, X_5, \ldots, X_n)] \quad (5.20)$$

Продолжая этот процесс, получаем

$$R_{2(k-1)} = R_{2(k-1)+1}\, R_{2(k-1)+2} \quad (k=1,\ldots,(n-3)) \quad (5.21)$$

(например, для $n = 6$ имеем $R = R_1R_3R_5R_6$) или в более общем случае

$$[R_1(Z_1, X_2, X_3)] \cap [R_3(Z_2, Z_1, X_4)] \cap \ldots \cap [R_{2(n-4)+1}(Z_{n-3}, Z_{n-4}, X_{n-2})] \cap$$
$$\cap [R_{2(n-4)+2}(Z_{n-3}, X_1, X_n)]. \quad (5.22)$$

Как и раньше, на вводимые отношения не накладывается никаких ограничений и поэтому разложение возможно, а выражение (5.22) содержит ровно (n—2) трехместных отношений. Следовательно, первое утверждение доказано.

Теперь приведем доказательство второго утверждения. Рассмотрим одну из подсистем отношения (5.22) и введем для удобства новые обозначения для термов:

$$R_j(Y^1_j, Y^2_j, Y^3_j), \quad (5.23)$$

где Y^1_j, Y^2_j, Y^3_j — соответствующие термы.

Запишем R_j в виде следующего произведения отношений:

$$R_j = R^1_j, R^2_j,$$
$$[R^1_j(Z^1, Y^3_j)] \cap [R^2_j(Z^1, Y^1_j, Y^2_j]. \quad (5.24)$$

Предположим, что условие (5.17) теоремы, состоящее в том, что новый терм Z' совпадает с одним из термов Y^1_j или Y^2_j, выполнен. Обозначим промежуточный терм Z' через Y^2_j. Тогда

$$[R^1_j(Y^2_j, Y^3_j)] \cap [R^2_j(Y^2_j, Y^1_j, Y^2_j] = [R^1_j(Y^2_j, Y^3_j)] \cap [R^2_j(Y^1_j, Y^2_j)] \quad (5.25)$$

следовательно, подсистему консультирования R_j удалось разложить на два двухместных отношения. Поскольку такое разложение возможно для всех j, вся система консультирования в целом разлагается на 2(n—2) подсистемы консультирования двухместных отношений. Это доказывает достаточность условий теоремы.

Покажем их необходимость. Предположим, что существует трехместное отношение, для которого ни один промежуточный терм не совпадает ни с одним из трех термов исходного отношения, тогда

$$R_j(Y^1_j,Y^2_j,Y^3_j),\ R_j=R^1_j,R^2_j,\ [R_i^1\ (Y^1_j,\ Z^1)]\cap[\ R^2_j\ (\ Z^1,\ Y^2_j,Y^3_j)]. \quad (5.26)$$

Здесь Z^1 —новый терм и, следовательно, второе отношение трехместно.

Таким образом, представление исходного отношения в виде произведения двух других R_i^1 и R_i^2 не изменило максимального порядка отношения, поскольку R_i^2 — трехместно.

Состояние системы консультирования. Доказанная теорема свидетельствует о том, что система консультирования высшего порядка не может быть разложена, вообще говоря, на подсистемы консультирования с менее чем трехместными отношениями.

Следствие. *Одним из термов трехместного отношения является* ***состояние*** *системы консультирования.*

Рассмотрим систему консультирования, осуществляющую отображение 1 семейства множеств $X_2(t)$ на множество элементов X_1, т. е.

$$X_1RX_2(t) \quad (5.27)$$

Примером такой задачи может быть задача синтеза системы консультирования, имеющей некоторое множество входных сигналов (управляемых и наблюдаемых факторов) как функций времени и обладающей такими характеристиками, что достигается экстремум некоторого функционала X_1. Элемент отношения $X_2(t)$ в (5.27) является множеством функций времени и, следовательно, элементами этого множества являются функции времени, т. е. в свою очередь некоторые множества. Предположим, что множества $x_2(t)\in X_2(t)$ конечны и содержат по p элементов. Отношение (5.27) тогда имеет $p + 1$ порядок и в соответствии с доказанной теоремой не может быть разложено на отношения ниже третьего порядка.

Предположим, что элементы $x_2(t)$ упорядочены:

$$x_2\ (t) = [x_2\ (t_1),\ x_2(t_2),...,\ x_2\ (t_p)]. \quad (5.28)$$

Тогда (5.27) будет иметь вид

$$x_1\ R = [x_2\ (t_1),\ x_2(t_2),...,\ x_2\ (t_p)]. \quad (5.29)$$

Рассмотрим теперь подмножество всех элементов $x_2(t)$ с индексом, большим j,

$$x^j_2\ (t) = [x_2\ (t_{j+1}),\ ...,\ x_2\ (t_p)]. \quad (5.30)$$

Отношение (5.27) будет эквивалентно отношению

$$x_1\ R = [x^j_2\ (t),\ x^{jr}_2(t)]. \quad (5.31)$$

где $x^{jr}_2(t)$ состоит из оставшихся элементов $x_2(t)$,

$$x^{jr}_2(t) =[\ x_2\ (t_1),...,\ x_2\ (t_j)]. \quad (5.32)$$

Представим теперь R в виде произведения отношений:

$$X_1R_1[X^j{}_2(t), Z^j], Z^jR_2X^{jr}{}_2(t). \quad (5.33)$$

Терм X_1 зависит теперь от промежуточного терма Z и не зависит от элементов $X_2(t)$, у которых индекс меньше или равен j. Можно утверждать, что элементы Z описывают состояние системы консультирования. Разложение системы консультирования на два отношения (5.33) можно понимать как утверждение, что x_1 зависит лишь от состояния системы консультирования Z^j в момент $t=t_i$ и от всех будущих элементов x_2, но не зависит от всех предыдущих элементов.

Распространим приведенные рассуждения на случай бесконечных множеств; так, если множество $x_2(t)$ имеет бесконечную мощность, то его всегда можно представить как объединение двух подмножеств:

$$x_2(t) = \{x^1{}_2(t); x^2{}_2(t)\}, \quad (5.34)$$

удовлетворяющих условию

$$\forall_j \forall_i \{[x_2(t_i) \subset x^1{}_2] \cap [x_2(t_j) \subset x^2{}_2] \leftrightarrow [i < j]\} \quad (5.35)$$

где $\forall$ — квантор общности. Тогда отношение системы консультирования можно переписать в виде

$$X_1R[X^1{}_2, X^2{}_2] \quad (5.36)$$

и, вновь представляя (5.36) произведением, получим

$$X_1R_1[X^1{}_2, Z], ZR_2X^2{}_2 \quad (5.37)$$

Таким образом, можно считать, что элементы множества Z описывают состояние системы консультирования.

Заметим, что понятие состояния системы консультирования появилось здесь как следствие теоремы о разложении. ***Три терма*** в отношении (5.37) в этом случае являются ***входом, выходом и состоянием.*** Дадим пояснения понятию состояния системы консультирования. В качестве входов и выходов системы консультирования обычно выбираются такие переменные, которые ЛФР может наблюдать и измерять, природа же промежуточных переменных часто может оставаться неизвестной, а измерение их — невозможным. Значение промежуточных переменных заключается в их комбинированном действии на зависимости между входными и выходными переменными. Указанное действие будем называть ***состоянием системы консультирования***. Состояние системы консультирования в момент t_v будем обозначать через z_v. Набор всех возможных состояний системы консультирования, которые ей присущи, называется ***множеством состояний*** и обозначается через Z, $z_v \in Z$.

5.2.3. Свойства и поведение системы консультирования

Общие замечания. Изучение ***свойств*** и ***поведения*** системы консультирования можно начать после того, как задана некоторая абстрактная система консультирования, например, в виде некоторого $X_s \subset X$.

Очевидно, нерационально изучать такую систему консультирования, осуществляя перебор элементов множества X_s. Эта ситуация напоминала бы попытку изучать какую-либо реальную систему консультирования, не руководствуясь подходящей теорией. Вероятно, для того чтобы решить, к какому типу относится рассматриваемая система консультирования, необходимо найти различие между этой системой консультирования и некоторой другой, возможно весьма близкой системой консультирования, с помощью относительно небольшого числа ***высказываний***. Эти высказывания должны относиться к системе консультирования как к единому целому.

Рассмотрим теперь произведение $X=X_1 \times X_2 \times \ldots \times X_n$ и построим на нем пропозициональную функцию $L(x)$, область определения которой совпадает со всем пространством X. Если предложение $L(x_i)$ истинно всякий раз, когда $x_i \in X_s$, то будем считать, что ***L(x) определяет*** некоторое ***свойство системы консультирования***.

Свойством системы консультирования *называется некоторая* ***пропозициональная функция****, определенная на X и истинная на* X_s. Заметим, что задача выбора для исследования тех или иных свойств системы консультирования выходит за рамки формального изучения абстрактных систем. Решение этой задачи зависит от того, каким образом предстоит использовать какую-либо реальную систему консультирования. Однако и формальное изучение должно указывать на то, какого рода свойства могут быть у данной системы консультирования. Например, ***приступая к изучению любой системы консультирования, хотелось бы знать, является ли эта система консультирования открытой или замкнутой, устойчива или нет, обладает ли она свойством управляемости и т. д.***

С точки зрения использования системы консультирования могут интересовать различные ее свойства. Эта совокупность свойств вытекает из поведения системы консультирования.

Поведением абстрактной системы консультирования будем называть некоторое множество ее свойств:

$$B=\{L_1 \ldots, L_p\}.$$

Целенаправленное поведение системы консультирования. Примем два основных направления для обсуждения систем консультирования и, следовательно, два контекста, в соответствии с

которыми можно интерпретировать взятые для описания системы консультирования. Эти два направления характеризуются двумя принципиально различными отношениями ЛФР к системе консультирования. При ***первом подходе*** система консультирования изучается как бы извне, а ***ее поведение рассматривается как некоторое отображение одного подмножества термов (входных величин и состояний) в другое.*** Такой подход, такой взгляд на поведение системы консультирования будем называть ***терминальным подходом.*** Приведенные ранее определения относились в основном к формальному описанию этого подхода.

При ***втором подходе*** предполагается, что известны некоторые инвариантные аспекты поведения системы консультирования, отражающие ***цель ее действия (целенаправленный подход),*** и имеется полное представление о действии системы консультирования, обеспечивающем достижение этой цели.

Для того чтобы развивать целенаправленный подход, необходимо владеть четкими определениями понятий: ***«управление», «консультационный процесс», «формирование рекомендаций», «принятие решения», «адаптация», «самоорганизация»*** и т. д.

Теперь остановимся на некоторых общих замечаниях относительно целенаправленного поведения систем консультирования.

1. ***Каждую систему консультирования, по крайней мере в принципе, можно описать с точки зрения либо терминального подхода, либо целенаправленного***. Выбор соответствующего подхода определяется отношением ЛФР к системе консультирования и той информацией о поведении системы консультирования, которой оно располагает.

2. Поведение целенаправленных систем консультирования гораздо сложнее поведения терминальных систем консультирования.

3. Целенаправленный подход при описании поведения системы консультирования позволяет предсказывать поведение системы консультирования в условиях, отличных от наблюдавшихся в прежних экспериментах.

Третье замечание относительно целенаправленного поведения поясним на примере. Пусть имеется подопытное животное, на которое воздействует пара стимулов. Первый стимул — световой сигнал — на короткий промежуток времени опережает второй стимул — электрический ток. Подопытное животное может покинуть шоковую зону, преодолев некоторый барьер, ограждающий зону безопасности. Производя опыты многократно, можно сформировать рекомендации, реализация которых позволит построить для этого животного кривую

обучения, откладывая по оси ординат время, затрачиваемое на уход из шоковой зоны, а по оси абсцисс число проведенных опытов (рис. 5.5).

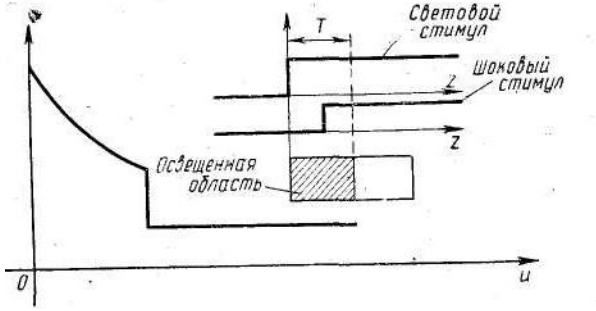


Рис. 5.5. Кривая обучения животного.

Как видно, кривая терпит разрыв непрерывности, соответствующий тому моменту, когда животное освоило взаимосвязь между световым и шоковым стимулом и начинает бежать к барьеру, как только зажглась лампочка.

Очевидно также, что для получения разрывной кривой обучения нельзя пользоваться теми же рекомендациями по преобразованию стимулов в реакцию, которые наблюдаются для того же животного при других условиях эксперимента, например, приводящих к непрерывной характеристике обучения. Поэтому если надо объяснить поведение некоторой обучающейся системы в различных ситуациях, то следует лучше разобраться в самом процессе формирования рекомендаций для обучения, происходящем внутри системы. Именно в этом случае могут оказаться эффективными модели, использующие целенаправленный подход. Если бы, например, удалось обнаружить, по какому принципу система консультирования формирует рекомендации о переходе из режима поведения с экспонентой на другой режим, то наверное удалось бы без специальных экспериментов описать поведение при самых разнообразных условиях.

5.2.4. Внешние и внутренние воздействия

Вводные замечания. Из доказательства теоремы о декомпозиции следует, что при определенных условиях абстрактная система консультирования представляется комбинацией трех термов соответствующих входу, состоянию и выходу системы консультирования. Если обратиться к реальной системе консультирования, то ее входом является совокупность воздействий (консультационных приемов, операций), обусловленных наличием среды, в которой она работает, а выход может характеризовать

качество работы системы консультирования по формированию рекомендаций, определенное ее свойствами или поведением. Поэтому при проектировании систем консультирования, при их анализе, при испытаниях и эксплуатации в условиях, где они предназначены работать, необходимо располагать методами, которые позволяли бы изучить характеристики среды, оценить состояние системы консультирования и ее качество. Здесь мы остановимся на рассмотрении этих задач.

Реальная система консультирования выполняет те или иные действия по формированию рекомендаций для решения задач консультируемой проблемы в некоторых условиях, которые определяются окружающей средой. В качестве окружающей среды, воздействующей на систему консультирования, можно рассматривать, например, атмосферу, электрические, тепловые или световые сигналы, воздействующие на аппаратуру систем управления, и т. д. Естественно, что результат действия системы консультирования во многом определяется тем, насколько точно при ее создании были учтены характеристики среды. Поэтому весьма важно при исследовании свойств и поведения систем консультирования знать эти воздействия. Определение характеристик среды не является простой задачей и притом эта задача весьма сложно решается аналитически.

В настоящее время развиваются основы теории систем консультирования, позволяющей исследовать широкий класс динамических систем консультирования, рассматриваемых как системы сбора, переработки и преобразования информации. Успех применения данного подхода, а также и других подходов, применяемых при анализе и синтезе систем консультирования, в значительной мере зависит от достоверности сведений о внешних воздействиях на систему консультирования и об изменении внутренних свойств системы консультирования, т. е. от достоверности информации, которая собирается и перерабатывается системой консультирования.

Если исследователь или ЛФР не располагает сведениями о воздействиях, то результаты исследования или сформированные рекомендации могут быть неудовлетворительными. Это обстоятельство, как правило, является причиной больших затрат времени на создание системы консультирования. Требуется рассматривать также физическую природу различного рода явлений, причины их возникновения и каким образом эти явления воздействуют на системы консультирования. Характер воздействий определяется объективными свойствами явлений и поведением собственно системы

консультирования, поэтому характеристики воздействий рассматриваются исходя из взаимодействия с природой, т. е. не остаются без внимания условия применения систем консультирования в зависимости от их назначения.

Внешние и внутренние воздействия. Практически всякая система консультирования состоит из некоторого объекта управления и совокупности средств, предназначенных для получения информации о действии консультируемого объекта и для формирования рекомендаций по решению задач консультируемой проблемы. Эта совокупность средств, состоящая из устройств получения информации, ее обработки, устройств, позволяющих формировать рекомендации, передавать их ЛПР и исполнить управляющий сигнал, будем называть ***управляющей системой системы консультирования.*** Если эти действия осуществляются без участия человека, т. е. автоматически, то такая система консультирования, состоящая из средств консультирования и системы управления, называется *автоматической,* в противном случае ее называют *автоматизированной.* Воздействия на систему консультирования, которые возникают в природе, могут быть приложены или к консультируемому объекту, или к какому-либо из устройств системы управления, или это воздействие влияет на деятельность человека (ЛФР,а), работающего в системе консультирования.

Изменение внутренних свойств системы консультирования может также происходить за счет изменений либо в объекте управления, либо в каком-нибудь из устройств системы управления. Так как система консультирования есть нечто целое, действующее во внешней среде и обладающее определенными свойствами, то поэтому всякое изменение во внешней среде или изменение внутренних свойств влияет в той или иной степени на работу системы консультирования. ***Без среды нет системы!***

В качестве внешних воздействий можно рассматривать атмосферные возмущения, помехи, сопровождающие различного рода измерения, флюктуации давления воздуха, волнение океана, потоки заявок на формирование рекомендаций в САК по отпуску продукции и т. д. В качестве изменений внутренних свойств — уходы параметров из-за дрейфа, внутренние шумы аппаратуры, вибрации, отказы и т. п. Все эти явления, как правило, в достаточной мере адекватно описываются нестационарными случайными консультационными процессами. Поэтому в развитие высказанных положений сформулируем общий принцип измерения и обработки нестационарных случайных консультационных процессов, которые часто возникают и

характеризуют ее воздействие на систему консультирования. Заметим, что изучение стационарных случайных консультационных процессов здесь можно рассматривать как частный случай. Отметим также, что ***под изучением случайных консультационных процессов мы понимаем определение их средних характеристик.***

Сущность проблемы получения средних характеристик нестационарных случайных консультационных процессов определяется в основном двумя факторами: во-первых, трудно получить необходимую совокупность реализаций консультационного процесса в эксперименте, во-вторых, не менее трудно построить в достаточной мере адекватную математическую модель реального консультационного процесса или явления для того, чтобы найти совокупность реализаций нестационарного случайного консультационного процесса путем статистического моделирования.

Многие результаты в этой области, полученные в предположении стационарности процессов, имеют большое значение для решения задач формирования рекомендаций и для выявления физической сущности консультационных процессов, однако они не решают основной проблемы получения статистических характеристик нестационарных консультационных процессов. В то же время требование высокой точности управления консультируемыми проблемами в нестационарных условиях, решение задачи формирования рекомендаций и т. п. делает весьма актуальным получение сведений именно о ***нестационарных случайных консультационных процессах.***

В развитие имеющихся в этой области работ, основанных на получении, как правило, одной реализации нестационарного консультационного процесса при дальнейшем выделении стационарной составляющей или применении гипотезы квазистационарности к исходной реализации, рассмотрим другой принцип изучения нестационарных случайных консультационных процессов, основанный на такой организации измерений, в результате проведения которых получается конечное множество реализаций $\{x_1(\lambda),...,x_n(\lambda)\}$ некоторых стационарных случайных консультационных процессов $X_1(\lambda)$, ..., $X_n(\lambda)$ соответственно. Каждая такая реализация характеризует случайную величину, соответствующую совокупности значений x_t в моменты $t_1, t_2, ..., t_j, ... t_n$ $(j=1, 2, ..., n)$, изучаемого нестационарного консультационного процесса $(X_t, t \in T)$. Реализации $\{x_1(\lambda), ..., x_n(\lambda)\}$ должны быть получены одновременно, т. е. измерения должны быть проведены с помощью нескольких измерителей одновременно.

Получение среднего по времени соответствующих произведений этих реализаций стационарных консультационных процессов позволяет определить искомые статистические характеристики нестационарного консультационного процесса. Такими характеристиками могут быть моменты случайного консультационного процесса или законы распределения.

На основе такого принципа построения измерений и обработки результатов в литературе излагается в значительной мере общий метод, позволяющий определить статистические характеристики нестационарных случайных консультационных процессов во многих консультационных задачах.

5.3. Математичекое моделирование систем консультирования

5.3.1. Математичекое моделирование систем консультирования рядом Винера

Вводные замечания. В самом общем случае система консультирования может быть описана весьма сложными соотношениями. Остановимся на описании нелинейных система консультирования с постоянными параметрами. Систематический подход к характеризации нелинейных систем при помощи явного описания зависимости между входом и выходом первым осуществил Н. Винер, используя теорию рядов Вольтерра.

Новая концепция, которая в этом случае развивается, состоит в том, что в качестве ***математического эквивалента системы консультирования рассматривается функционал***.

Известно, что функция *f(x)* каждому значению независимого переменного *x* ставит в соответствие некоторое число. ***Функционал же F[x(t)] ставит в соответствие каждой функции x(t) число и определен в некоторой области функционального пространства.***

Аналогично ***система консультирования ставит в соответствие функции (предыстории входного воздействия) некоторое число, а именно, выходную реакцию***.

Этот подход сводит ***задачу описания системы консультирования с заданным классом входных воздействий к задаче построения функционала, заданного на некотором классе функций.***

Рассмотрим метод, дающий систематический аналитический подход к задаче описания непрерывных нелинейных систем консультирования.

В основу метода можно положить описание ***аналитических функционалов с помощью ряда Вольтерра***:

$$y(t)=h_0+\int_{\eta} h_1(\tau)\,x(t-\tau)\,d\tau+\int_{\eta}\int_{\eta} h_2(\tau_1,\ \tau_2)\,x(t-\\ -\tau_1)\,x(t-\tau_2)\,d\tau_1 d\tau_2+\ldots, \tag{5.38}$$

где η — область интегрирования, т. е. область, на которой определена функция *x(t)*. Фреше доказал, что любой непрерывный функционал *y*[*x*(*t*)], определенный на множестве функций {*x*(*t*)}, областью определения которых является интервал [*a, b*], может быть представлен интегралами Вольтерра. Бриллиант доказал эту теорему для бесконечного интервала. Суть такого описания состоит в том, что ***вместо явного выражения для абстрактной системы консультирования отыскивается метод ее аппроксимации, который начинается с простых элементов, а затем при постепенном усложнении он дает возможность аппроксимировать систему консультирования с желаемой точностью.***

Для описания системы консультирования по существу необходимо знание ядер ряда

$$h_n(\tau_1,\ldots,\ \tau_n)\ (n=1,\ 2,\ldots).$$

Ортогональные функционалы Винера. Далее мы рассмотрим, каким образом можно построить математическую модель системы консультирования на основе использования при этом ряда из ортогональных функционалов Винера и когда *x(t)* является белым гауссовым процессом.

Развитие винеровских ортогональных функционалов позволило впервые записать однородные и неоднородные функционалы, например, однородный функционал степени *n* есть

$$\int_{-\infty}^{\infty}\ldots\int_{-\infty}^{\infty} h_n(\tau_1,\ldots,\tau_n)\,x(t-\tau_1)\ldots x(t-\tau_n)\,d\tau_1\ldots d\tau_n \tag{5.39}$$

и неоднородный функционал степени *n,* представляющий собой сумму функционалов,

$$\int_{-\infty}^{\infty}\underbrace{\ldots}_{n}\int_{-\infty}^{\infty}h_n(\tau_1,\ldots,\tau_n)x(t-\tau_1)\ldots x(t-\tau_n)d\tau_1,\ldots,d\tau_n+$$
$$+\int_{-\infty}^{\infty}\underbrace{\ldots}_{n-1}\int_{-\infty}^{\infty}h_{n-1}(\tau_1,\ldots,\tau_{n-1})x(t-\tau_1)\ldots x(t-$$
$$-\tau_{n-1})d\tau_1,\ldots,d\tau_{n-1}+\ldots+h_0. \tag{5.40}$$

В качестве первого шага при получении ортогональных функционалов для белого гауссова процесса $x(\tau)$ нужно брать неоднородный функционал первой степени

$$\int_{-\infty}^{\infty}k_1(\tau)x(t-\tau)d\tau+k_0 \tag{5.41}$$

и определить k_0, удовлетворяющее условию, чтобы (5.41) было ортогональным любой константе «K», что означает:

$$\lim_{T\to\infty}\frac{1}{2T}\int_{-T}^{T}\left[\int_{-\infty}^{\infty}k_1(\tau)x(t-\tau)d\tau+k_0\right]Kd\tau=0. \tag{5.42}$$

Здесь ядро первого порядка в (5.42) обозначено через k_1 и константа через k_0 с тем, чтобы избежать путаницы с h_1 и h_0 в (5.38). Будем называть h_n в (5.38) ***ядром Вольтерра*** и k_1 в (5.42) и другие k в выражениях ядер высокого порядка — ***ядрами Винера.***

Следующим шагом надо брать неоднородный функционал второй степени

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}k_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)d\tau_1d\tau_2+$$
$$+\int_{-\infty}^{\infty}k_{1(2)}(\tau_1)x(t-\tau_1)d\tau_1+k_{0(2)} \tag{5.43}$$

и определять $k_{1(2)}$ (т. е. ядро первого порядка в функционале второй степени) и $k_{0(2)}$ в зависимости для k_2, удовлетворяющими условию, чтобы (5.43) было ортогональным любой константе и любому функционалу первой степени.

Продолжая этот процесс, получаем неоднородный функционал n-й степени и $k_{n-1(n)}$, $k_{n-2(n)}$,. . ., $k_{0(n)}$ в зависимости для k_n при условии, что неоднородный функционал будет ортогональным любой константе и всем однородным функционалам степени, меньшей n.

Пусть корреляционная функция $R_x(x)$ белого гауссова процесса есть

$$R_x(\tau) = N\delta(\tau), \tag{5.44}$$

где N — const, $\delta(\tau)$—дельта-функция. Спектральная плотность такого процесса есть

$$S_x(\omega) = \frac{N}{2\pi}. \tag{5.45}$$

Процедура ортогонализации, только что описанная выше, дает для функционала (5.41), удовлетворяющего (5.42), функционал вида

$$G_1[k_1,\ x(t)] = \int_{-\infty}^{\infty} k_1(\tau)\, x(t-\tau)\, d\tau. \tag{5.46}$$

Функционал $G_i[k_1,\ x(t)]$ будем называть ***G-функционалом Винера первой степени***. А для (5.43), удовлетворяющего требованию, приведенному непосредственно после (5.43), получим ***функционал Винера второй степени:***

$$G_2[k_2,\ x(t)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\ \tau_2)\, x(t-\tau_1)\, x(t-\tau_2) d\tau_1\ d\tau_2 - N\int_{-\infty}^{\infty} k_2(\tau_2,\ \tau_2)\, d\tau_2 \tag{5.47}$$

и т. д.

В результате ортогонализации постоянный член можно записать в виде

$$G_0[k_0, x(t)] = k_0 \tag{5.48}$$

Н. Винер построил ряд из ортогональных функционалов и для случая, когда воздействие $x(t)$ представляет собой белый гауссов процесс, выразил ***реакцию нелинейной системы через воздействие с помощью этого ряда:***

$$y(t) = \sum_{n=0}^{\infty} G_n[k_n,\ x(t)], \tag{5.49}$$

где $\{k_n\}$—множество ядер, $\{G_n\}$—множество ортогональных функционалов. Ортогональное свойство этих функционалов выражается тем, что среднее по времени произведения $G_m G_n$ равно соотношению

$$\overline{G_n[k_n,\ x(t)]\, G_m[k_m, x(t)]} = 0 \text{ при } m \neq n. \tag{5.50}$$

Пример. Перечислим первые четыре *G*-функционала:

$$G_0[k_0,\ x(t)]=k_0,$$

$$G_1[k_1,\ x(t)]=\int_{-\infty}^{\infty}k_1(\tau_1)\,x(t-\tau_1)\,d\tau_1,$$

$$G_2[k_2,\ x(t)]=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}k_2(\tau_1,\ \tau_2)\,x(t-\tau_1)\,x(t-\tau_2)\,d\tau_1 d\tau_2-$$

$$-N\int_{-\infty}^{\infty}k_2(\tau_2,\ \tau_2)\,d\tau_2,$$

$$G_3[k_3,\ x(t)]=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}k_3(\tau_1,\ \tau_2,\ \tau_3)\,x(t-\tau_1)x(t-\tau_2)\times$$

$$\times x(t-\tau_3)\,d\tau_1 d\tau_2 d\tau_3-3N\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}k_3(\tau_1,\ \tau_2,\ \tau_2)x(t-\tau_1)d\tau_1 d\tau_2. \qquad (5.51)$$

Старшим членом функционала *n*-й степени G_n является интеграл

$$\underbrace{\int_{-\infty}^{\infty}\dots\int_{-\infty}^{\infty}}_{n}k_n(\tau_1,\dots,\ \tau_n)\,x(t-\tau_1)\dots x(t-\tau_n)\,d\tau_1\dots d\tau_n, \qquad (5.52)$$

который представляет собой однородный функционал *n*-го порядка. Другими же слагаемыми функционалами G_n будут однородные функционалы степени меньше *n,* ядра которых выводятся из ядра старшего члена, так что для воздействия *x(t)* функционал G_n является ортогональным относительно всех функционалов степени, меньшей *n.* Это можно видеть в формулах (5.51) для первых четырех функционалов *G.*

Винер показал, что коэффициенты функционалов G_n представляют собой коэффициенты полинома Эрмита:

$$H_n(u)=u^n-C_2^nNu^{n-2}+1\cdot3C_4^nN^2u^{n-4}-$$

$$-1\cdot3\cdot5C_6^nN^3u^{n-6}+\dots, \qquad (5.53)$$

где *u* — действительная переменная, а C^n_m — биномиальные коэффициенты.

Таким образом, ***ряд из ортогональных G-функционалов Винера может быть использован для моделирования нелинейных систем консультирования,*** и если теперь рассмотреть ряд (5.49), то можно видеть, что определение ядер Винера $\{k_n\}$ будет представлять собой основную проблему при исследовании нелинейных систем консультирования.

Винер предложил разлагать ядра $\{k_n\}$ в ряд ортогональных функций таких, как функции Лагерра. Если, например, $\{\varphi_k(\tau)\}$ представляет

собой множество функций Лагера, то ядра $\{k_n\}$ можно представить в виде

$$k_1(\tau_1) = \sum_{k=0}^{\infty} \varkappa_k \varphi_k(\tau_1),$$

$$k_2(\tau_1, \tau_2) = \sum_{k_1=0}^{\infty} \sum_{k_2=0}^{\infty} \varkappa_{k_1 k_2} \varphi_{k_1}(\tau_1) \varphi_{k_2}(\tau_2)$$

. . . . . .

$$k_n(\tau_1, \ldots, \tau_n) = \sum_{k_1=0}^{\infty} \ldots \sum_{k_n=0}^{\infty} \varkappa_{k_1 \ldots k_n} \varphi_{k_1}(\tau_1) \ldots \varphi_{k_n}(\tau_n). \quad (5.54)$$

Определение коэффициентов разложения Лагерра $\aleph_k$, приводящее к определению *G*-функционалов, осуществляется системой измерений. Существует также другой способ определения ядер Винера, основанный на использовании взаимной корреляции между воздействием и реакцией системы консультирования. В дальнейшем рассмотрим именно эти два способа и покажем возможность их практического использования.

Суть исследования систем консультирования теперь состоит в том, что, либо определив коэффициенты разложения ядер по функциям Лагерра, либо, определив ядра непосредственно, можно вычислить реакцию системы консультирования на любой входной консультационный процесс.

5.3.2. Моделирование систем консультирования сетями Петри

В связи с широким использованием параллельных и распределенных систем консультирования особую актуальность приобретают дискретные структуры, представляющие параллельные консультационные процессы. Аппаратом описания сложных систем консультирования, взаимодействующих с консультационными процессами, являются формальные системы консультирования типа ***сетей Петри***, моделирующие динамические свойства систем консультирования.

Формализм сетей Петри общего вида основан на понятии комплекта, являющемся в некотором роде обобщением понятия множества. Как и множество, ***комплект*** — *это набор элементов, но всякий элемент может входить в него более одного раза*. Иначе говоря, отношение включения, связывающее элементы и множества, заменяется на

функцию числа экземпляров элемента в комплекте, которая обозначается как *#(x, B)* (читается: «число *x* в комплекте *B*»).

Множество — частный случай комплекта.

Многие понятия теории множеств распространяются и на комплекты. Так, *пустой комплект* аналогичен пустому множеству. *Мощность комплекта* есть общее число экземпляров элементов в комплекте. Комплект *A* включен в комплект *B* (является подкомплектом), если для всякого x # $(x, A) \le$ # (x, B). С помощью функции # легко определяются *операции* над комплектами:

- для *объединения комплектов*

$$A \text{ и } B \ \#\ (x, A \cup B) = \max(\#\ (x, A), \#\ (x, B));$$

- для *пересечения комплектов*

$$A \text{ и } B \ \#\ (x, A \cap B) = \min(\#\ (x, A), \#\ (x,B));$$

- для *суммы комплектов*

$$A \text{ и } B \ \#\ (x, A + B) = \#,(x, A) + \#\ (x, B);$$

- для *разности комплектов A* и *B*

$$\#\ (x, A - B) = \#\ (x, A) - \#\ (x, A \cap B).$$

Если *M* - множество, то M^n — множество всех таких комплектов, построенных из элементов *M*, что

$$\#\ (x, B) \le n,\ B \in M^n;$$

M^∞ — множество всех комплектов, построенных из элементов *M* без ограничения на число экземпляров элемента в комплекте.

Сеть Петри — это <u>четверка</u>

$$C=(P, T, I, O),$$

где *P* — конечное множество *позиций, T*—конечное множество *переходов, I: T→P^∞* — входная функция (воздействие), отображающая переходы в комплекты позиций, *O: T→P^∞* — выходная функция (реакция), отображающая переходы в комплекты позиций. Графически сеть Петри изображают в виде мультиграфа с вершинами двух видов: ***кружки*** соответствуют позициям, ***планки*** — переходам. Функции *I* и *O* представляются дугами (рис. 5.6).

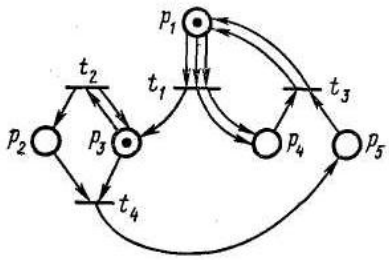


Рис. 5.6.

Позиции, дуги из которых ведут в переход t_j, называются *входными* для t_j, аналогично, позиции, в которые ведут дуги из перехода t_j, называются *выходными* для t_j. Множество входных позиций обозначают $I(t_j)$, а выходных — $O(t_j)$. В сети Петри, изображенной на рис. 5.6, $I(t_1) = \{ p_1, p_1, p_1\}$, $O(t_1) = \{p_3, p_4, p_4\}$. Функции I и O удобно обобщить и на отображение из позиций в комплекты переходов $(P \rightarrow T^{\infty})$, что позволяет обозначать множества входных и выходных переходов позиции p_i, определяемые аналогично множествам входных и выходных позиций перехода, соответственно как $I(p_i)$ и $O(p_i)$. В сети Петри, изображенной на рис. 5.6, $I(p_3) = \{t_1, t_2\}$, $O(p_3) = \{t_2, t_4\}$.

Введенные понятия относятся к статической структуре сети Петри. Динамические свойства сети Петри определяются с помощью понятия маркировки.

Маркировка μ ***сети Петри*** $C=(P, T, I, O)$ — ***это функция, отображающая множество позиций P в множество неотрицательных целых чисел N.***

Маркировка изображается с помощью помещаемых внутрь позиций ***фишек*** (точек). Так, маркировка сети Петри, приведенной на рис. 5.6, определяется как $\mu(p_1)=\mu(p_3)=1$, $\mu(p_2)=\mu(p_4)=\mu(p_5)=0$.

Удобно представлять маркировку как n-вектор $\mu = (\mu_1, \mu_2, ..., \mu_n)$ (где $n = | P |$), каждый элемент которого μ_i есть $\mu(p_i)$, а также как комплект μ, в который входят позиции сети $p_i \in P$ и $\#(p_i, \mu) = \mu(p_i)$. Сеть Петри C с определенной в ней маркировкой μ называется ***маркированной сетью Петри***.

Маркировка сети может изменяться в результате ***запуска переходов***. Переход t_j маркированной сети Петри C с маркировкой μ называется ***разрешенным***, если $I(t_j) \subseteq \mu$, т. е. в каждой входной позиции t_j находится не меньше фишек, чем из этой позиции исходит дуг в t_i. Всякий разрешенный переход может запуститься. В результате *запуска* перехода t_j маркировка сети μ изменяется на новую: $\mu' = \mu - I(t_j) + O(t_j)$, т. е. из всякой входной позиции p_i перехода t_i удаляется столько фишек, сколько дуг ведет из p_i в t_j, а в каждую выходную позицию p_k помещается столько фишек, сколько дуг ведет из t_i в p_k. Последовательность запусков переходов называется ***выполнением сети Петри.***

Рассмотрим выполнение сети Петри, изображенной на рис. 5.6. В ***начальной маркировке*** разрешен только переход t_2. При его запуске фишка удалится из p_3, а затем в позиции p_2 и p_3 добавится по фишке, т. е. в результате запуска в новой маркировке μ' появится фишка еще и в p_2. Теперь становятся разрешенными переходы t_2, t_4. Поскольку запуститься может любой разрешенный переход, предположим, что

запускается переход t_4. После его запуска из позиции p_2 и p_3 фишки удалятся, а в позиции p_5 появится одна фишка. В получившейся маркировке μ" не разрешен ни один переход. На этом выполнение сети Петри заканчивается.

Рассмотрим маркировку μ сети Петри *C = (P, T, I, O)*. Маркировка μ' называется ***непосредственно достижимой*** из μ, если найдется такой переход $t_i \in T$, разрешенный в μ, что при его запуске получается маркировка μ'; в этом случае пара (μ, μ') принадлежит ***отношению непосредственной достижимости***, определенному на P^{∞}. Транзитивное замыкание этого отношения называется ***отношением достижимости***. Маркировки μ' такие, что (μ, μ') принадлежат отношению достижимости, называются ***достижимыми*** из μ. Множество достижимых из μ маркировок сети Петри *C* называется ***множеством достижимости*** и обозначается *R (C,* μ).

Интерпретация сетей Петри основана на понятиях ***условия*** и ***события***.

Состояние системы консультирования описывается совокупностью условий.

Функционирование системы консультирования состоит в осуществлении последовательности определенных действий (консультационных операций), т. е. событий.

Для возникновения события необходимо выполнение некоторых условий, называемых ***предусловиями***. Возникновение события может привести к нарушению предусловий и к выполнению некоторых других условий, называемых ***постусловиями***. В сети Петри ***условия*** моделируются ***позициями***, ***события — переходами***. ***Предусловия события*** представляются ***входными позициями*** соответствующего перехода, ***постусловия — выходными позициями***. Возникновение события моделируется ***запуском перехода***. ***Выполнение условий*** представляется ***наличием фишек*** в соответствующих позициях, ***невыполнение — их отсутствием***.

Рассмотрим, например, простую систему консультирования, последовательно обрабатывающую консультационные операции (задания), которые поступают во входную очередь. Когда процессор свободен и во входной очереди имеется консультационное задание, оно обрабатывается процессором, затем выводится. Эту систему можно промоделировать сетью Петри, изображенной на рис. 5.7.

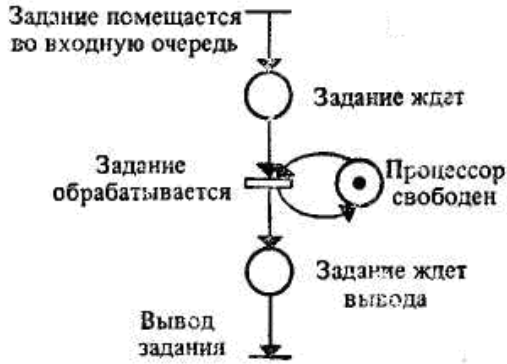


Рис. 5.7.

Установим, какие особенности систем учитывают сети Петри. Это прежде всего ***асинхронность. В сети Петри отсутствует понятие времени.*** Время возникновения событий никак не указывается, но тем не менее структура сети Петри устанавливает частичный порядок возникновения событий. Далее, поскольку возникновение событий представляется запуском переходов, предполагается, что события происходят ***мгновенно***. Если же моделируемое событие имеет отличную от нуля длительность, как, например, событие «задание обрабатывается» (рис. 5.7), и это существенно, то оно представляется в виде двух мгновенных событий типа «***начало события***», «***конец события***» и условия «***событие происходит***» (рис. 5.8).

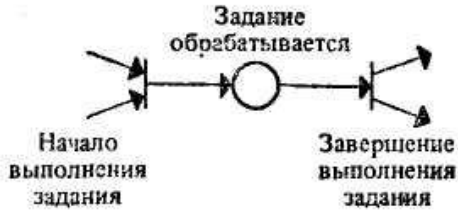


Рис. 5.8.

Кроме того, считается, что события происходят ***неодновременно*** (***мгновенные события не могут происходить одновременно***). Действительно, если допустить одновременное возникновение некоторых событий *i* и *j*, которым в сети Петри соответствуют переходы t_i и t_j, то можно ввести дополнительный переход t_{ij} с

$$I(t_{ij})=I(t_i)+I(t_j),$$
$$O(t_{ij})=O(t_i)+O(t_j),$$

интерпретирующийся как одновременное возникновение событий *i* и *j*. В этом случае переходы можно запускать последовательно.

Другим важным свойством сетей Петри, как инструмента моделирования систем консультирования, является их способность представлять ***параллелизм*** и ***конфликтные ситуации***.

Параллелизм двух событий представляется двумя разрешенными переходами, множества входных позиций которых не пересекаются (рис. 5.9), конфликт — переходами с общей входной позицией (рис. 5.10).

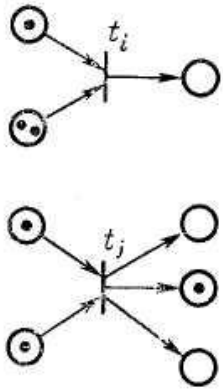


Рис. 5.9.

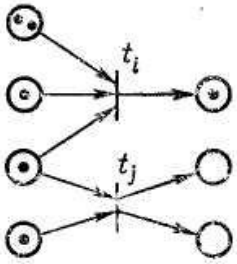


Рис. 5.10.

Сети Петри будем использовать в основном как формальный аппарат при ***моделировании систем консультирования***, которым ***присущ параллелизм в реализации консультационного процесса.***

Если рассматривать систему консультирования в целом, то возможны два принципиально различных подхода к использованию сетей Петри. В первом система консультирования моделируется сетью Петри, которая преобразуется по определенным правилам к некоторому «оптимальному» виду. Полученная сеть Петри преобразуется в проект системы консультирования. Предполагается, что он также оптимальный. Здесь сети Петри применяют непосредственно при проектировании системы консультирования. Этот подход, однако, имеет трудности, связанные с неоднозначностью обратного преобразования сети Петри в проект системы консультирования,— что подвергает сомнению оптимальность получаемого проекта системы консультирования. Во втором, более общепринятом, подходе сначала с помощью обычных средств создается проект системы

консультирования и по нему строится модель в виде сети Петри. Затем исследуются свойства полученной сети и делаются выводы о свойствах и характеристиках проекта системы консультирования. Если они неудовлетворительны, то полученные в результате исследования сети Петри данные используют для модификация проекта системы консультирования. Модифицированный проект системы консультирования снова преобразуют в сеть Петри, и цикл повторяется. ***Этот процесс заканчивается, когда сеть Петри будет обладать необходимыми свойствами.***

Рассмотрим, какие свойства сети Петри как модели системы консультирования могут интересовать ЛФР. Одно из важнейших свойств — ***эффективность.***

Позиция сети Петри называется эффективной, если число фишек в ней никогда не превышает 1.

Маркированная сеть Петри эффективна, если эффективны все ее позиции.

Это свойство очень важно при интерпретации позиций как простых условий: если в позиции есть фишка, то условие выполняется, если нет, то не выполняется. Если интерпретация фишек более сложная, например количество фишек показывает число информационных единиц, то может быть интересен вопрос, ограничено ли число фишек в данной позиции, и если да, то какова граница. Так приходим к свойству ***ограниченности***. Позиция называется *k-ограниченной,* если число фишек в ней в любой достижимой маркировке не превышает целого *k.* Маркированная сеть Петри называется *k-ограниченной,* если ее позиции являются *k* -ограниченными. В сети Петри, приведенной на рис. 5.11, позиции p_1, p_2 являются эффективными, позиция p_3 – 2-ограниченная, вся сеть - 2-ограниченная.

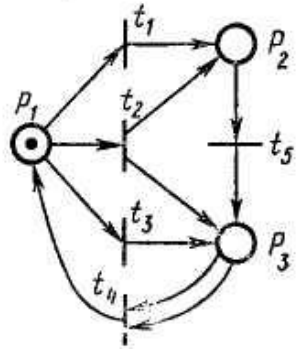


Рис. 5.11.

В случае когда фишки интерпретируются как некоторые ресурсы, они не должны ни создаваться, ни уничтожаться. Иначе говоря, в сети должен действовать закон сохранения. Маркированная сеть Петри называется ***строго сохраняющей***, если мощность маркировки (как комплекта позиций) постоянна. В общем случае фишка может интерпретироваться как некоторое ***число элементарных ресурсов***, причем это число меняется от позиции к позиции. Введем понятие ***взвешивания позиций****: вектор* $\mathbf{W} = (w_1, w_2, ..., w_n)$, где w_i — вес позиции p_i.

Сеть Петри называется ***сохраняющей по отношению к вектору взвешивания*** **W**, если скалярное произведение вектора **W** и маркировки (рассматриваемой как вектор) постоянно; сеть Петри — ***сохраняющая***, если она является сохраняющей по отношению к вектору взвешивания **W**, все элементы которого положительны.

Рассмотренные до сих пор свойства относятся как к последовательным, так и к параллельным системам консультирования. Но при переходе от последовательных систем консультирования к параллельным возникают принципиально новые трудности: возможность **тупиковых ситуаций.**

Тупиком в сети Петри называется множество переходов, которые в некоторой достижимой маркировке μ' и в последующих достижимых из μ' маркировках не разрешены.

Свойство возможности возникновения тупиков в системе консультирования моделируется свойством активности в сетях Петри. Переход t_i называется ***активным***, если он не входит ни в какой тупик. Переход называется ***пассивным***, если он не разрешен ни в какой достижимой маркировке. При детальном исследовании активности сети Петри используют также понятие ***уровней активности***.

Переход t_i обладает *активностью:*

- *уровня* 0, если он не может быть запущен (пассивный);
- *уровня* 1, если он потенциально запустим, т. е. существует достижимая маркировка, в которой он разрешен;
- *уровня* 2, если для любого целого k существует последовательность запусков переходов, в которой он присутствует не менее k раз;
- *уровня 3,* если существует бесконечная последовательность запусков, в которой он присутствует неограниченно часто;
- *уровня 4,* если он потенциально запустим из всякой достижимой маркировки (т. е. активен).

В сети Петри, изображенной на рис. 5.12, *а,* переход t_3 активен, переходы t_1, t_2, t_4, t_6 имеют уровень активности 3; переход t_6 пассивен. В сети Петри, приведенной на рис. 5,12, *б,* переход t_4 пассивен, переход

t_3 обладает активностью уровня *1*, переход t_2 — активностью уровня *2*, а t_1 — активностью уровня *3*.

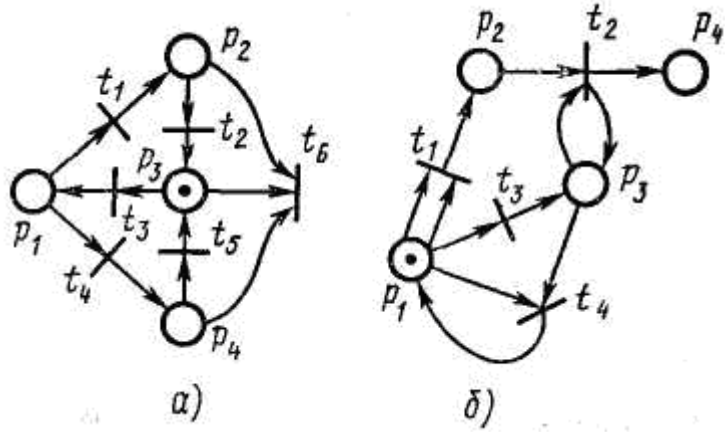


Рис. 5.12.

Одной из наиболее важных задач анализа сетей Петри является ***задача достижимости:*** достижима ли маркировка μ′ из начальной маркировки μ данной сети Петри? Важность этой задачи обусловлена тем, что ***маркировка служит интерпретацией состояния системы консультирования***. Решение задачи достижимости позволит определить, достижимо ли определенное состояние системы консультирования, будь оно «хорошим» или «плохим» для системы консультирования.

Описанные свойства и соответствующие задачи анализа сетей Петри — наиболее общие, хотя и не охватывают все множество вопросов, которые могут возникнуть при анализе сетей Петри.

Для решения задач анализа имеется ***два основных подхода***.

Первый основан на построении ***дерева достижимости***.

Дерево достижимости — это ориентированное корневое дерево, вершинам которого соответствуют возможные маркировки, дугам — переходы.

Корневой вершине соответствует начальная маркировка. Из каждой вершины исходят дуги, соответствующие разрешенным переходам. Построение дерева осуществляется последовательно начиная с корневой вершины; на каждом шаге строится очередной ***ярус дерева.*** Например, дерево достижимости для сети Петри, изображенной на рис. 5.12, *а,* после трех шагов имеет вид, приведенный на рис. 5.13, *а* (маркировки представлены векторами).

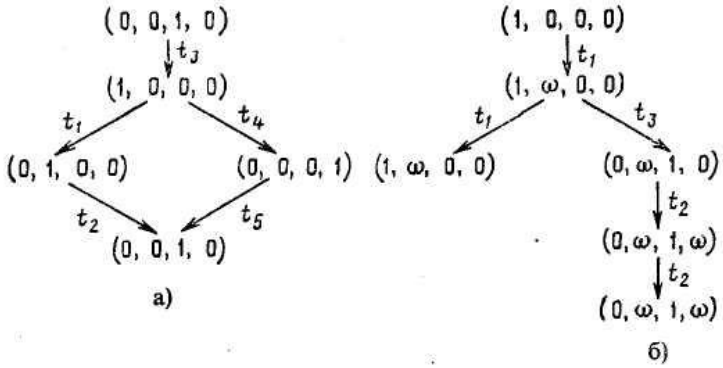


Рис. 5.13.

Очевидно, что если не использовать при построении дерева определенные соглашения, то активные (даже ограниченные) сети Петри будут иметь бесконечное дерево достижимости.

Назовем вершины (и соответствующие маркировки), построенные на очередном шаге алгоритма, ***граничными.*** Если в какой-либо граничной маркировке нет разрешенных переходов, то будем называть такую маркировку ***терминальной.*** Если какая-либо граничная вершина имеет маркировку, уже существующую в дереве, то назовем ее ***дублирующей.*** Для терминальных и дублирующих вершин не будем строить исходящих из них дуг. Это обеспечивает конечность дерева достижимости для ограниченной сети Петри (например, рис. 5.12, *а,* 5.13, *а*). Для неограниченных сетей требуется как-то обозначить неограниченное число фишек в позиции. Пусть ω обозначает такое число, причем

$$\omega + a = \omega,\ \omega - a = \omega,\ a < \omega,\ \omega \le \omega,$$

где a — произвольное целое положительное число.

Будем использовать при построении дерева достижимости следующее правило.

Пусть граничная вершина μ не является терминальной или дублирующей. Для каждого разрешенного перехода t_i в маркировке μ построим дугу, исходящую из μ. Дугу будем помечать переходом t_i. Маркировка μ' новой вершины определяется следующим образом. Если $\mu(p_i)=\omega$, то $\mu'(p_i)=\omega$. Если на пути от корневой вершины к μ существует вершина μ" такая, что в результате запуска в μ перехода t_i число фишек в каждой позиции не меньше чем в μ", а в позиции p_i строго больше, то $\mu'(p_i) = \omega$. В противном случае $\mu'(p_i)$ - число фишек в позиции p_i, получающееся после запуска t_j из μ (рис. 5.13, *б).*

Теорема. *Дерево достижимости любой сети Петри конечно.*

Доказательство этого утверждения основано на свойствах ω и на правилах введения этого символа в маркировку граничных вершин.

Метод анализа систем консультирования, основанный на дереве достижимости, позволяет определить ***свойства эффективности, ограниченности, сохранения, исследовать свойства активности и достижимости.***

Сеть Петри ограниченна тогда и только тогда, когда символ ω отсутствует в дереве достижимости. Кроме того, положение символа ω показывает, какие позиции неограниченны. Если символ ω отсутствует в дереве, то число достижимых маркировок конечно и все вопросы анализа можно решить простым перебором. В частности, для нахождения границы маркировки данной позиции p_i нужно найти наибольшее значение i-й компоненты среди всех вершин дерева. Если эта граница не превышает 1, то позиция эффективна.

Для установления того, является ли сеть Петри сохраняющей по отношению к некоторому вектору взвешивания $\mathbf{W} = (w_1, w_2, \ldots, w_n)$, необходимо решить следующую систему линейных уравнений с ограничениями :

$$\begin{cases} \sum\limits_{i=1}^{n} w_i \mu_j(p_i) = s \quad \text{при } j = (1, \ldots, k), \\ w_i \geqslant 0 \quad \text{при } i = (1, \ldots, n), \end{cases}$$

где k - число вершин дерева достижимости, которым соответствуют различные маркировки. (Очевидно, что если имеется $\mu_j(p_i) = \omega$, то $w_i = 0$.)

Возможность решения задач активности и достижимости ограничена существованием символа ω, скрывающего конкретную информацию о числе фишек. Например, если в сеть Петри, изображенную на рис. 5.13,б, ввести две дуги (t_1, p_2), (p_2, t_2), то полученная сеть Петри будет иметь то же дерево достижимости, что и первоначальная. При этом в новой сети Петри в позиции p_2 может быть только четное число фишек, тогда как в первоначальной сети — любое, т. е. множества достижимости этих сетей Петри не совпадают. Можно привести разные сети с различными свойствами активности, но имеющие одно дерево достижимости.

Тем не менее, хотя дерево достижимости и не дает полной информации о свойствах достижимости и активности, в некоторых случаях оно позволяет ответить на вопросы по достижимости и активности. Например, если в нем имеется терминальная вершина, то сеть Петри не активна. При решении задачи достижимости может оказаться, что

маркировка μ' присутствует в дереве достижимости (положительный ответ) или что маркировка μ' не покрывается никакой вершиной дерева достижимости, т. е. μ'' μ для всех вершин μ'• (отрицательный ответ).

Другой подход к анализу сетей Петри называется <u>матричным</u> и основан на их матричном представлении. Введем матрицы D^- и D^+, столбцам которых соответствуют позиции, строкам — переходы, и $D^-(j, i) = \#(p_i, I(t_j))$, $D^+(j, i) = \#(p_i, O(t_j))$. Пусть $e(j)$ - вектор-строка, компоненты которого соответствуют переходам и все равны нулю, за исключением j-й, которая равна 1. Тогда переход t_j разрешен, когда $\mu \geq e(j) \cdot D^-$ (μ рассматривается как вектор), а результатом запуска t_j из μ является

$$\mu' = \mu - e(j)D^- + e(j)D^+ = \mu + e(j) \cdot (D^+ - D^-) = \mu + e(j) \cdot D,$$

где $D = D^+ — D^-$ *— составная матрица изменений.*

Маркировка μ', получающаяся из μ в результате запуска последовательности $\sigma = t_{j_1} t_{j_2} \ldots t_{j_k}$, определяется как

$$\mu' = \mu + e(j_1) D + e(j_2) D + \ldots + e(j_k) D =$$
$$= \mu + (e(j_1) + \ldots + e(j_k)) D = \mu + f(\sigma) D,$$

где $f(\sigma) = e(j_1) + \ldots + e(j_k)$ *— вектор запуска, i-*я компонента которого равна числу запусков t_j в σ.

Если маркированная сеть Петри является сохраняющей по отношению к некоторому вектору взвешивания **W** (**W** — вектор-столбец), то $\mu W = \mu' W$ для любой μ' = $R(C, \mu)$. Так как μ' = μ + $f(\sigma)D$, то $f(\sigma)D$ W = 0. Поскольку это верно для всех $f(\sigma)$, имеем DW=0. Следовательно, сеть Петри является сохраняющей по отношению к некоторому вектору взвешивания тогда и только тогда, когда имеется такой вектор **W**, что DW=0. Это уравнение позволяет отыскать и сам вектор взвешивания **W**.

Если маркировка μ' достижима из начальной маркировки μ сети Петри, то должно существовать целое неотрицательное решение уравнения μ' = μ + xD, решением которого будет $x = f(\sigma)$.

Исследуем задачу достижимости для сети Петри, изображенной на рис. 5.12, *б,* с начальной маркировкой (1, 0, 0, 0) для маркировки μ' = (0, 2, 1, 2). Уравнение μ' = μ + xD принимает вид

$$(0,\ 2,\ 1,\ 2) = (1,\ 0,\ 0,\ 0) + x \cdot \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}.$$

и имеет решение x = (4, 2, 1, 0), соответствующее последовательности запусков переходов $t_1t_1t_1t_1t_3t_2t_2$.

Матричный подход к анализу сетей Петри, как и подход, основанный на дереве достижимости, не позволяет в общем случае решить задачу активности и достижимости. Проблемы матричного анализа заключаются в том, что, во-первых, вектор запуска, получаемый при решении уравнения, не дает информации о порядке запуска переходов и, во-вторых, может соответствовать неразрешенной последовательности запусков.

Установлено, что задачи достижимости и активности *эквивалентны,* но неизвестно, разрешимы ли они вообще, т. е. нет ни алгоритма, позволяющего решить эти задачи, ни доказательства его отсутствия.

Рассмотрим применение методов анализа сетей Петри, моделирующих практические системы консультирования.

Специализированная система консультирования для реализации итерационных параллельных консультационных процессов состоит из совокупности процессорных элементов (ПЭ) и модулей памяти (памяти данных (МПД) и памяти команд (МПК)), связанных в кольцо (рис. 5.14).

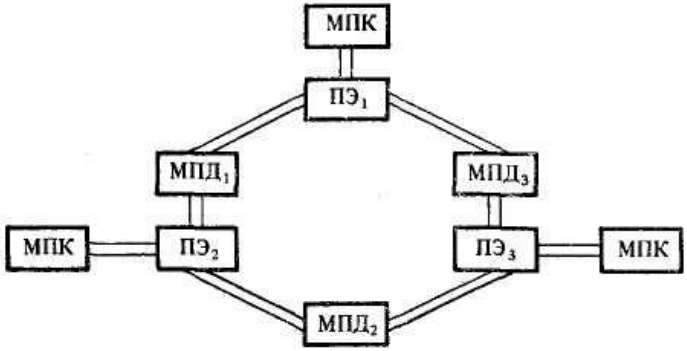


Рис. 5.14.

ПЭ работает в двух режимах. В первом случае он захватывает оба смежных МПД, используя левый для извлечения исходных данных новой итерации, правый - для выборки результатов предыдущей итерации. По завершении итерации он помещает результат в правый МПД и освобождает оба МПД. В другом режиме ПЭ работает с внутренними данными. Режим указывается командой, считываемой из МПК. Рассмотрим два варианта реализации устройств управления ПЭ. В первом варианте захват МПД при осуществлении итерации

производится последовательно. ПЭ может находиться в следующих состояниях: «обработка внутренних данных» (S_1), «захвачен левый МПД» (S_2), «захвачен правый МПД» (S_3), «захвачены оба МПД, обработка данных очередной итерации» (S_4). МПД может быть либо свободным, либо захваченным. Событиями будем считать захват и освобождение МПД. Этот вариант работы представляется сетью Петри, в которой каждому ПЭ соответствуют четыре позиции, реализующие описанные условия, а каждому МПД — позиция (фишка, в которой означает, что МПД свободен) (рис. 5.15).

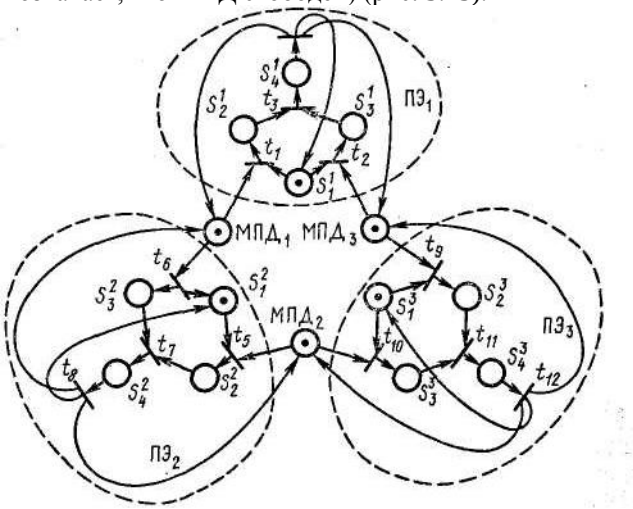


Рис. 5.15.

Можно убедиться, что дерево достижимости этой сети Петри содержит две терминальные маркировки: $\mu_1=\{S^1_2,S^2_2,S^3_2\}$ и $\mu_2=\{S^1_3,S^2_3,S^3_3\}$, представляющие ситуации, когда каждый ПЭ захватывает левый и правый МПД соответственно. Это означает, что сеть Петри не активна, т. е. в ***системе консультирования возможны тупиковые ситуации***. При другом варианте работы системы консультирования ПЭ осуществляют только одновременный захват смежных МПД (если он возможен). В этом случае каждому ПЭ соответствуют только условия S_1 и S_4 (рис. 5.16).

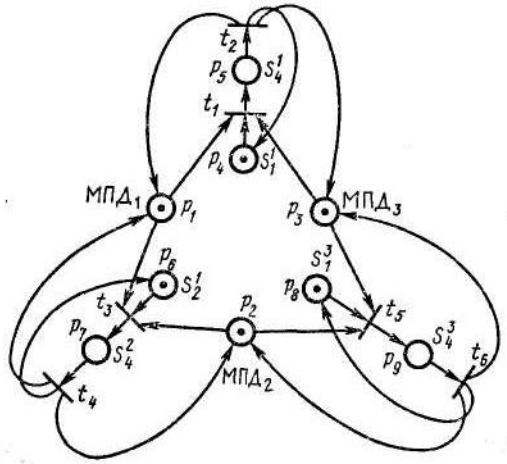


Рис. 5.16.

Дерево достижимости (рис. 5.17) не содержит терминальных маркировок, сеть Петри активна.

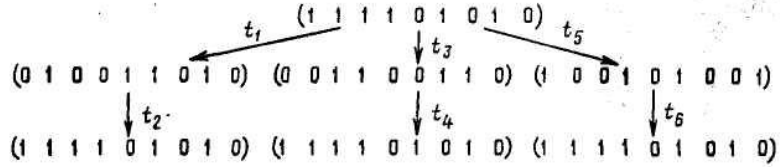


Рис. 5.17.

Кроме того, из рассмотрения дерева достижимости очевидно, что сеть Петри безопасная (позиции интерпретируются как простые условия). В системе осуществляется распределение ресурсов, которые не появляются и не исчезают, т. е. выполняется закон сохранения. Определим, является ли сеть Петри сохраняющей. Для этого решим уравнение $DW = 0$, которое принимает вид

$$\begin{bmatrix} -1 & 0 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \\ w_9 \end{bmatrix} = 0.$$

Решением его является $W=(1, 1, 1, 1, 3, 1, 3, 1, 3)$. Действительно, позиции p_5, p_7, p_9 представляют условия, связанные с тремя устройствами, остальные позиции — условия, которые связаны с

одним устройством. Таким образом, сеть Петри обладает основными необходимыми свойствами, что обеспечивает работоспособность второго варианта.

Моделирования реальных систем консультирования приводят к различным доопределениям и модификациям сетей Петри. В основном эти модификации связаны с изменением правила запуска переходов.

Мощность моделирования обычных сетей Петри ограничена невозможностью проверки позиции на нуль (т. е. того, что маркировка позиции нулевая). Одним из способов преодоления этого недостатка является введение ***сдерживающих дуг.*** По новым правилам запуска переход разрешен, если фишки есть в его обычных входных позициях (из которых ведут обычные дуги) и отсутствуют в сдерживающих входных позициях (из которых ведут сдерживающие дуги). Сдерживающая дуга изображается как обычная, только на конце имеет вместо стрелки маленький кружок (это обозначение заимствовано из теории переключательных схем, где кружок означает «не») (рис. 5.18).

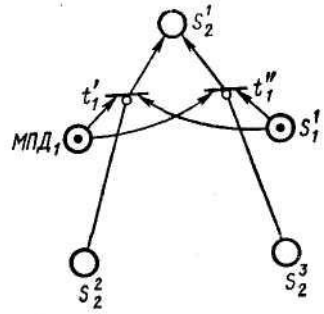


Рис. 5.18.

Если в обычных сетях Петри переход запускается по *логике И,* то в *сетях Петри со сдерживающими дугами* логика расширена до включения отрицаний. Поскольку событие может представляться несколькими переходами, можно смоделировать событие, предусловие которого записывается как объединение нескольких конъюнкций условий и отрицаний условий, соответствующих позициям сети Петри со сдерживающими дугами. Таким образом, сети Петри позволяют моделировать предусловия в виде ДНФ, т. е. условия самого общего вида.

Рассмотренная задача организации работы специализированной системы консультирования может быть решена и первым вариантом, в котором разрешен последовательный захват МПД (см. рис. 5.15), но с предотвращением тупиковых ситуаций. Очевидно, что возможны две

тупиковые ситуации, описываемые маркировками с фишками в позициях S^1_2,S^2_2,S^3_2 и S^1_3,S^2_3,S^3_3 (см. рис. 5.15). Для предотвращения первой тупиковой ситуации необходимо в маркировках $\{ S^1_2,S^2_2\}$, $\{ S^1_2,S^3_2\}$, $\{ S^3_2,S^2_2\}$ предотвратить проявление фишки в позициях S^3_2,S^2_2,S^1_2 соответственно. Следовательно, переход t_1 должен иметь в качестве предусловия конъюнкцию МПД$_1$ $\&S^1_1\&\overline{(S^1_1 \& S^2_2)}$, т. е. его нужно заменить на переходы t'_1 и t''_1 с предусловиями МПД$_1\&\overline{S^1_1}\ \&\ \overline{S^2_2}$ и МПД$_1$ $\&S^1_1\&\overline{S^2_2}$ (рис. 5.18). Аналогично следует поступить с переходами t_5, t_9 (см. рис. 5.15). Подобные действия предпринимаются и для предотвращения второй тупиковой ситуации.

Другие предложения по изменению правил запуска либо эквивалентны введению сдерживающих дуг, либо носят даже более частный характер. Например, в сетях Петри с областями ограничения имеются множества позиций (называемые областями ограничения), в которых фишки одновременно находиться не могут. Правила запуска модифицированы так, чтобы не нарушать это условие. Если в сеть Петри, изображенную на рис. 5.15, включить две области ограничения $\{S^1_2,S^2_2,S^3_2\}$ и $\{S^1_3,S^2_3,S^3_3\}$, то можно предотвратить возникновение тупиковых ситуаций.

5.4. Анализ и синтез систем консультирования

Реальная и «математическая» системы консультирования. Пусть даны две системы консультирования, одна из которых имеет известные характеристики, а другая — неизвестные. При этом неизвестная система консультирования подлежит исследованию, а известная используется для проведения исследования. На вход обеих систем консультирования поступает одно и то же воздействие. Если эти системы не идентичны, то на их выходах будут различные реакции.

Здесь возникают две задачи.

Первая — пусть даны воздействия и желаемая реакция неизвестной системы консультирования. Требуется найти такие характеристики известной системы консультирования, при которых реакция на ее выходе близка к желаемой реакции неизвестной системы консультирования. Эту задачу будем называть ***задачей синтеза***.

Вторая — пусть дана неизвестная система консультирования. Требуется по воздействию и реакции этой системы консультирования определить характеристики известной системы консультирования, позволяющие установить аналитическую зависимость между входом и

выходом неизвестной системы консультирования. Эту задачу будем называть ***задачей анализа.***

Если обратиться к соотношению (5.49), то при решении задачи синтеза требуется определить, какими должны быть ядра Винера $k_n(\tau_1, ..., \tau_n)$, чтобы обеспечить желаемую реакцию $y(t)$ при заданном воздействии $x(t)$.

При решении задачи анализа надо определить, какими характеристиками обладает система консультирования при ее действии. Именно определение ядер Винера некоторой реальной системы консультирования решает в том числе и задачу идентификации.

Таким образом, ***как при решении задачи анализа, так и синтеза необходимо <u>определение ядер</u>*** $k_n(\tau_1, ..., \tau_n)$. Можно также утверждать, что практическое решение задачи определения ядер $k_n(\tau_1, ..., \tau_n)$ можно осуществить, если параллельно некоторой реальной системе консультирования подключить «математическую» систему (математические зависимости, реализованные на устройствах вычислительной техники) и, варьируя параметрами «математической» системы, обеспечить их близость по реакциям в том или ином смысле. Сначала дадим общую идею теории. Она строится на основе использования ортогональных свойств некоторых функционалов, зависящих от $k_n(\tau_1, ..., \tau_n)$ в (5.49).

Рассмотрим схему, приведенную на рис. 5.19.

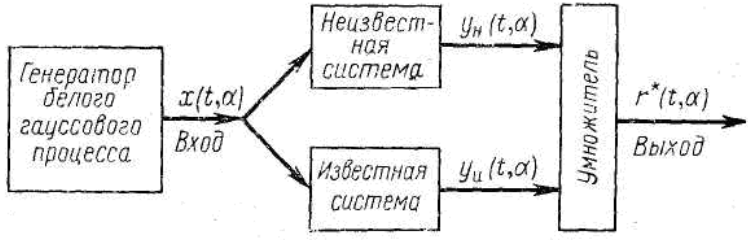


Рис. 5.19.

Белый гауссов процесс $x(t, \alpha)$, создаваемый генератором случайного процесса, поступает на входы известной и неизвестной систем консультирования. Соответственно на выходе известной системы консультирования будем иметь

$$y_ц(t, \alpha) = \sum_n G_n[H_n(t+\tau_1, \ldots, t+\tau_n), \alpha], \tag{5.55}$$

а на выходе неизвестной системы консультирования

$$y_н(t,\ \alpha) = \sum_n G_n[K_n(t+\tau_1, \ldots,\ t+\tau_n),\ \alpha]. \tag{5.56}$$

В формулах (5.55) и (5.56) G_n — ортогональные функционалы, K_n и H_n — ядра функционалов неизвестной системы консультирования и известной соответственно, α — параметр, показывающий, что случайный процесс является не только функцией времени, но и события, или совокупности реализаций. Выходные процессы перемножаются и получается процесс $r(t,\ \alpha)$. Это произведение выходных процессов запишется в виде

$$r(t,\ \alpha) = \sum_m \sum_n G_m[K_m(t+\tau_1, \ldots,\ t+\tau_m),\ \alpha]\, G_n[H_n(t+ \\ +\tau_1, \ldots,\ t+\tau_n), \alpha]. \tag{5.57}$$

Среднее по времени произведение (5.57) равно

$$\lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} r(t,\ \alpha)\, dt \tag{5.58}$$

для почти всех α.

Преобразование, которое получается, если прибавить время t к аргументу всех броуновских кривых, является сохраняющим меру преобразованием броуновских движений, и это сохраняющее меру преобразование эргодическое. Поэтому выражение (5.58) равно среднему по совокупности:

$$\int_0^1 d\alpha \sum_m \sum_n G_m[K_m(\tau_1, \ldots,\ \tau_m),\ \alpha]\, G[H_n(\tau_1, \ldots,\ \tau_n)\, \alpha]. \tag{5.59}$$

В этом содержание эргодической теоремы: *практически для всех* α *среднее по времени совпадает со средним по состояниям или по совокупности реализаций случайного процесса.*

Уже утверждалось, что G_n-функционалы обладают ортогональным свойством. Следовательно, когда m и n различны, интегрирование по α дает ноль. Когда же $m = n$, имеем

$$n! \int \ldots \int K_n(\tau_1, \ldots,\ \tau_n)\, H_n(\tau_1, \ldots,\ \tau_n)\, d\tau_1, \ldots,\ d\tau_n. \tag{5.60}$$

Тогда, если известно среднее значение $r(t,\ \alpha)$, то получим соотношение

$$\lim_{T\to\infty}\frac{1}{2T}\int_{-T}^{T} r(t,\ \alpha)\,dt = \sum_{n} n!\int\ldots\int K_n(\tau_1,\ldots,\tau_n)\times$$
$$\times H_n(\tau_1,\ldots,\tau_n)\,d\tau_1,\ldots,d\tau_n. \tag{5.61}$$

Это соотношение означает, что среднее по времени реализации процесса *r(t, α)* на выходе умножителя равно сумме интегралов от произведения ядер неизвестной и известной систем. Но известную систему можно взять какой угодно, например H_n представить в виде произведения функций Лагерра:

$$H_n(\tau_1,\ldots,\tau_n)=\prod_{i=1}^{n}\varphi_k(\tau_i), \tag{5.62}$$

где φ_k — функции Лагерра *(k* = 0, 1, 2, ...). Если множество {*H*} типа (5.62), то среднее по времени *r(t, α)* будет с точностью до *n*! совпадать с интегралом по времени от произведения K_n и известной лагерровской функции. Учитывая соотношения (5.61) и (5.62) таким образом, можно получить разложение K_n в ряд по произведениям лагерровских функций и, следовательно, можно решить задачу анализа и синтеза в том смысле, как это обсуждалось ранее.

Нетрудно видеть, что практическая реализация этой теории требует эксперимента, так как для решения задачи анализа и синтеза необходимо получение среднего по времени произведения реакций реальной системы и «математически» описываемой функциями Лагерра. В сложных случаях это можно выполнить лишь с помощью вычислительной машины.

Методика анализа и синтеза. Теперь рассмотрим методику анализа и синтеза систем консультирования более детально. Пусть гауссов случайный процесс *x(t)* с единичным уровнем спектральной плотности одновременно подается на вход исследуемой системы консультирования и некоторой схемы, построенной путем параллельного соединения устройств, каждое из которых можно описать ортогональной функцией Лагерра $\varphi_k(t)$. Если осреднить по времени реакцию схемы, реализующей $\varphi_k(t)$, то получаются коэффициенты A_i ортонормированного разложения Лагерра для *x(t)*. Эти коэффициенты статистически независимы, а их вероятность распределения подчиняется закону Гаусса:

$$p(A_0,\ldots,A_n)=\frac{2}{(2\pi)^{n/2}}\exp-\frac{1}{2}\left(A_0^2+\ldots+A_n^2\right). \tag{5.63}$$

В силу физической осуществимости системы консультирования ее реакция *r(t)* представляет собой функцию предыдущих значений воздействия *x(t).* Выразим *r(t)* через коэффициенты Лагерра для *x*(*t*):

$$r(t)=\lim_{n\to\infty} R(A_0,\ A_1,\ldots,\ A_n). \tag{5.64}$$

Здесь *R* является функцией переменных A_i, которые могут быть распределены на интервале $-\infty<A_i<\infty$. функцию *R* можно представить полиномом Эрмита для многих переменных. В этом рассмотрении переменными являются случайные величины $A_0, A_1 \ldots, A_n$; для выхода системы *r(t)* имеем

$$r(t)=\lim_{n\to\infty}\sum_{a=0}^{\infty}\ldots\sum_{k=0}^{\infty} c_{a,b,\ldots,k} H_a(A_0) H_b(A_1)\ldots H_k(A_n). \tag{5.65}$$

Умножая обе части этого уравнения на

$$H_{k'}(A_0) H_{b'}(A_1)\ldots H_{k'}(A_n)\, e^{-\frac{1}{2}\left(A_0^2+A_1^2+\ldots+A_n^2\right)} \tag{5.66}$$

и усредняя его на всем интервале $-\infty< t <\infty$, получаем:

$$\lim_{T\to\infty}\frac{1}{2T}\int_{-T}^{T} r(t) H_{a'}(A_0) H_{b'}(A_1)\ldots$$

$$\ldots H_{k'}(A_n)\, e^{-\frac{1}{2}\left(A_0^2+A_1^2+\ldots+A_n^2\right)}\, dt=\lim_{T\to\infty}\sum_{a=0}^{\infty}\sum_{b=0}^{\infty}\ldots$$

$$\ldots\sum_{k=0}^{\infty} c_{a,b,\ldots,k}\cdot\frac{1}{2T}\int_{-T}^{T} H_a(A_0) H_{a'}(A_b) H_b(A_1) H_{b'}(A_1)\ldots$$

$$\ldots H_k(A_n) H_{k'}(A_n)\, e^{-\frac{1}{2}\left(A_0^2+A_1^2+\ldots+A_n^2\right)}\, dt. \tag{5.67}$$

Ясно, что для обеспечения высокой точности аппроксимации *r(t)* необходимо, чтобы $n\to\infty$. Здесь для краткости в (5.67) $\lim_{n\to\infty}$ опущен. Теперь согласно принципу эргодичности усреднение по времени, применяемое в правой части этого уравнения, можно заменить усреднением по множеству. Тогда при многомерном распределении вероятностей вида (5.63) уравнение (5.67) можно записать так:

$$\lim_{T\to\infty}\frac{1}{2T}\int_{-T}^{T} r(t)H_{a'}(A_0)H_{b'}(A_1)\ldots H_{k'}(A_n)e^{-\frac{1}{2}\left(A_0^2+A_1^2+\ldots+A_n^2\right)}\times$$

$$\times dt=\frac{1}{(2\pi)^{1/2}}\sum_{a=0}^{\infty}\sum_{b=0}^{\infty}\ldots\sum_{k=0}^{\infty}\int_{-\infty}^{\infty}H_a(A_0)\,H_{a'}(A_0)e^{-\frac{A_0^2}{2}}\times$$

$$\times dA_0\times\int_{-\infty}^{\infty}H_b(A_1)H_{b'}(A_1)\,e^{-\frac{A_1^2}{2}}dA_1\times\ldots$$

$$\ldots\times\int_{-\infty}^{\infty}H_k(A_n)\,H_{k'}(A_n)\,e^{-\frac{A_n^2}{2}}dA_n. \tag{5.68}$$

Вследствие свойства ортогональности полиномов Эрмита с весом $e^{-A_i^2/2}$ на прямой $-\infty<A<\infty$

$$\int_{-\infty}^{\infty}H_i(A)\,H_j(A)\,e^{-\frac{A^2}{2}}dA=\begin{cases}\sqrt{2\pi}\,i!, & i=j,\\ 0, & i\neq j,\end{cases} \tag{5.69}$$

где

$$H_i(A)=(-1)^i e^{\frac{A^2}{2}}\frac{d^i}{dA^i}\left(e^{-\frac{A^2}{2}}\right),$$

причем

$$H_0(A)=1;\ H_1(A)=A,$$

получим

$$\lim\frac{1}{2T}\int_{-T}^{T} r(t)\,H_a(A_0)\,H_b(A_1)\ldots$$

$$\ldots H_k(A_n)\,e^{-\frac{1}{2}\left(A_0^2+A_1^2+\ldots+A_n^2\right)}dt=\frac{c_{a,b,\ldots,k}}{(2\pi)^{n/2}}. \tag{5.70}$$

Множество коэффициентов c_a, c_b..... c_k ряда Эрмита, определяющих коэффициенты Лагерра A_i для гауссова процесса $x(t)$, может быть определено с помощью вычислительного устройства, схема которого, только для воспроизведения двух функций Лагерра и двух полиномов Эрмита, приведена на рис. 5.20.

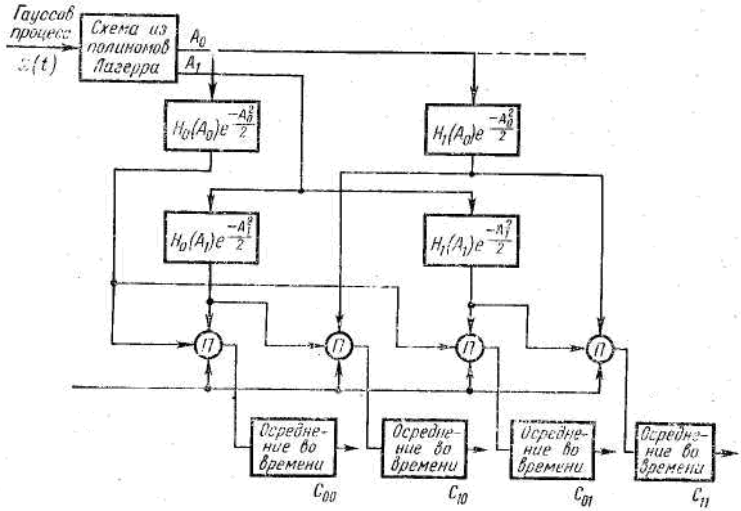


Рис. 5.20.

Теперь, пусть имеется устойчивая система консультирования, заданная коэффициентами c_a, $c_{b,}$....., c_k. Изложенным способом можно получить реакцию системы $r(t)$ для любого воздействия $x(t)$. Начнем с аппроксимации $r(t)$ конечным числом слагаемых ряда Эрмита многих переменных:

$$r(t)=\sum_{a=0}^{\infty}\sum_{b=0}^{\infty}\ldots\sum_{k=0}^{\infty}c_{a,b,\ldots,k}H_a(A_0)H_b(A_1)\ldots H_k(A_n), \quad (5.71)$$

где A_0, $A_{1,}$ ..., A_n — коэффициенты разложения по функциям Лагерра известной входной величины;

$$x(t)=A_0\varphi_0(t)+A_1\varphi_1(t)+\ldots+A_n\varphi_n(t). \quad (5.72)$$

Эти коэффициенты можно получить, если подать воздействие $x(t)$ на вход электрической схемы системы, реализующей функции Лагерра. Функциональное преобразование, соответствующее полиномам Эрмита, может быть также реализовано либо с помощью электрической схемы, либо с помощью вычислительной машины. Тогда «математическая» система, о которой говорилось выше, может быть построена по схеме, приведенной на рис. 5.21, где белый гауссов процесс $x(t)$ поступает на вход схемы, реализующей функции Лагерра, которые аппроксимируют линейную часть системы с памятью. Коэффициенты A_i (i=l, 2, ..., n), являющиеся значениями случайной величины, поступают на вход схемы, реализующей полиномы Эрмита, которые аппроксимируют безынерционную нелинейную часть

системы. В итоге получаем оценку сигнала *r*(t),* являющегося реакцией математической системы.

Рис. 5.21.

Таким образом, используя схему измерений, приведенную на рис. 5.20, можно определить коэффициенты A_0, A_1, ..., A_n и c_a, c_b,..... ,c_k с помощью которых на основании (5.71) можно определить реакцию системы на какой-либо другой сигнал.

Наборы коэффициентов $\{A_n\}$ и $\{ c_a, c_b,..... ,c_k \}$ характеризуют свойства и поведение системы консультирования, поэтому их можно использовать также для решения задач технической и медицинской диагностики.

5.5. Определение ядер Винера методом взаимной корреляции

Построение однородных функционалов. В основу определения ядер системы консультирования положим метод взаимной корреляции. Пусть, как и выше, на вход системы консультирования поступает белый гауссов процесс. Суть получения ядер заключается в соответствующей совместной обработке входного и выходного сигналов исследуемой системы консультирования.

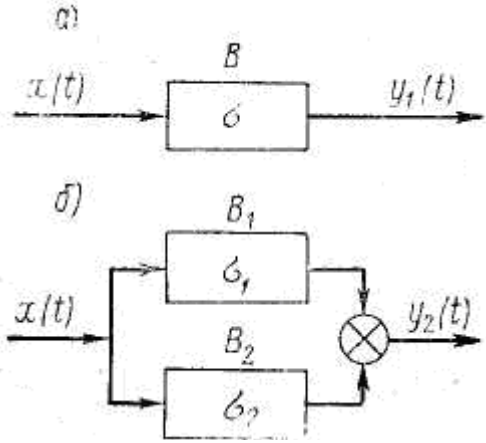


Рис. 5.22.

Сначала рассмотрим прохождение белого гауссова процесса по цепям с запаздыванием во времени. Выход цепи с запаздыванием σ (рис. 5.22, *а)* можно записать в виде однородного функционала первой степени:

$$y_1(t)=\int_{E_1}\delta(t-\sigma)\,x(t-\tau)\,d\tau. \tag{5.73}$$

Выходной сигнал двумерной цепи (рис. 5.22, *б)* с запаздыванием можно записать в виде однородного функционала второй степени:

$$y_2(t)=\int_{E_2}\delta(\tau_1-\sigma_1)\,\delta(\tau_2-\sigma_2)\,x(t-\tau_1)\,x(t-\tau_2)\,dv_\tau \tag{5.74}$$

и вообще цепь с *n*-мерным запаздыванием можно представить в виде однородного функционала степени:

$$y_n(t)=\int_{E_n}\delta(\tau_1-\sigma_1)\ldots\delta(\tau_n-\sigma_n)\prod_{r=1}^{n}x(t-\tau_r)\,dv_\tau. \tag{5.75}$$

Построив такую последовательность функционалов, рассмотрим возможность экспериментального определения ядер Винера с их помощью; ядро нулевого порядка

$$k_0=\overline{y}\,(t), \tag{5.76}$$

т. е. равно среднему значению реакции системы консультирования. Это следует из того, что

$$\overline{y(t)}=\overline{\sum_{n=0}^{\infty}G_n[k_n,\ x(t)]}=k_0. \tag{5.77}$$

Определим ядро первого порядка. Для этого вычислим взаимокорреляционную функцию *y(t)* и сигнала *x(t),* прошедшего одномерную цепь задержки, т. е. $y_1(t)$. Учитывая ортогональное свойство полиномов Винера, получим

$$\begin{aligned}\overline{y(t)\,y_1(t)}&=\overline{\left\{\sum_{n=0}^{\infty}G_n[k_n,\ x(t)]\right\}y_1(t)}=\\&=\overline{\left\{\int_{E_1}k_1(\tau_1)\,x(t-\tau_1)\,d\tau_1\right\}x(t-\sigma)}=\\&=\int_{E_1}k_1(\tau_1)\,\overline{x(t-\tau_1)\,x(t-\sigma)}\,d\tau_1=\int_{E_1}k_1(\tau_1)\,N\delta_0(\sigma-\tau)\,d\tau_1=\\&=Nk_1(\sigma),\end{aligned} \tag{5.78}$$

т. е.

$$k_1(\sigma)=\frac{1}{N}\,\overline{y(t)\,y_1(t)}. \tag{5.79}$$

Следовательно, при использовании функционалов Винера первое

ядро определится так же, как и весовая функция линейной системы.

Определим теперь ядро второго порядка. Используя вновь ортогональное свойство полиномов Винера, получаем

$$\overline{y(t)\,y_2(t)}=\overline{\left\{\sum_{n=0}^{\infty}G_n[k_n,x(t)]\right\}y_2(t)}=\overline{G_0[k_0,\ x(t)]\,y_2(t)}+$$
$$+\overline{G_1[k_1,\ x(t)]\,y_2(t)}+\overline{G_2[k_2,\ x(t)]\,y_2(t)}. \tag{5.80}$$

Вычислим первый член в (5.80):

$$\overline{G_0[k_0,x(t)]\,x[t-\sigma_1]\,x[t-\sigma_2]}=\overline{k_0x(t-\sigma_1)\,x(t-\sigma_1)}=$$
$$=k_0N\delta(\sigma_1-\sigma_2), \tag{5.81}$$

второй:

$$\overline{G_1[k_1x(t)]\,x[t-\sigma_1]\,x[t-\sigma_2]}=\left[\int_{E_1}\overline{k_1(\tau_1)\,x(t-\tau_1)\,d\tau_1}\right]\times$$
$$\times\,\overline{x(t-\sigma_1)\,x(t-\sigma_2)}=\int_{E_2}k_1(\tau_1)\,\overline{x(t-\tau_1)\,x(t-\sigma_1)}\times$$
$$\times\,\overline{x(t-\sigma_2)}\,d\tau_1=0. \tag{5.82}$$

Равенство нулю этого члена обусловлено тем, что среднее от произведения нечетного числа сигналов типа белого процесса с нулевым математическим ожиданием равно нулю. Далее, применяя свойство среднего от произведения четного числа сигналов типа белый гауссов процесс, вычисляем третий член в (5.80):

$$\overline{G_2[k_2,\ x(t)]\,x(t-\sigma_1)\,x(t-\sigma_2)}=\left[\int_{E_2}\overline{k_2(\tau_1,\ \tau_2)\,x(t-\tau_1)\,x(t-\tau_2)}\,d\tau_1d\tau_2-N\int_{E_1}k_2(\tau_2,\ \tau_2)\,d\tau_2\right]x(t-\sigma_1)\,x(t-\sigma_2)=$$
$$=\int_{E_2}k_2(\tau_1,\ \tau_2)\,\overline{x(t-\tau_1)\,x(t-\tau_2)\,x(t-\sigma_1)\,x}\,(t-\sigma_2)\,d\tau_1\,d\tau_2-$$
$$-N^2\delta(\sigma_1-\sigma_2)\int_{E_1}k_2(\tau_2,\ \tau_2)\,d\tau_2=$$
$$=\int_{E_2}k_2(\tau_1,\ \tau_2)\,N^2[\delta(\tau_1-\tau_2)\,\delta(\sigma_1-\sigma_2)+\delta(\tau_1-\sigma_1)\,\delta(\tau_2-\sigma_2)+$$
$$+\delta(\tau_1-\sigma_2)\,\delta(\tau_2-\sigma_1)]\,d\tau_1d\tau_2-N^2\,\delta(\sigma_1-\sigma_2)\int_{E_1}k_2(\tau_2,\tau_2)\,d\tau_2=$$
$$=N^2\left[\delta(\sigma_1-\sigma_2)\int_{E_1}k_2(\tau_1,\ \tau_1)\,d\tau_1+k_2(\sigma_2,\ \sigma_1)+k_2(\sigma_1\sigma_2)-\right.$$
$$\left.-\delta(\sigma_1-\sigma_2)\int_{E_1}k_2(\tau_2,\ \tau_2)\,d\tau_2\right]=2N^2k_2(\sigma_1,\ \sigma_2). \tag{5.82}$$

При выводе соотношения (5.82) учитывалось, что ядра Винера симметричны, т. е. что $k_2(\sigma_1, \sigma_2) = k_2(\sigma_2, \sigma_1)$. Суммируя (5.81) и (5.82), окончательно получаем для (5.80):

$$\overline{G_0[k_0, x(t)]\,x[t-\sigma_1]\,x[t-\sigma_2]} = \overline{k_0 x(t-\sigma_1)\,x(t-\sigma_1)} = k_0 N \delta(\sigma_1 - \sigma_2), \tag{5.83}$$

Выражение (5.83) определяет ядро второго порядка при всех $\sigma_1 \neq \sigma_2$, т. е.

$$k_2(\sigma_1, \sigma_2) = \frac{1}{2N^2}\,\overline{y(t)\,y_2(t)} \text{ для } \sigma_1 \neq \sigma_2. \tag{5.84}$$

Определим теперь ядро n-го порядка, вычисляя для этого среднее от произведения реакции системы консультирования и n-мерной цепи с запаздыванием:

$$\overline{y(t)\,y_n(t)}. \tag{5.85}$$

Благодаря свойству ортогональности полиномов Винера (5.85) будет иметь вид

$$\overline{y(t)\,y_n(t)} = \overline{\left\{\sum_{m=0}^{\infty} G_m[k_m, x(t)]\right\} \times \left\{\prod_{r=1}^{n} x(t-\sigma_r)\right\}}. \tag{5.86}$$

Для вычисления (5.86) запишем однородный функционал, представляющий n-мерную цепь с запаздыванием в виде

$$y_n(t) = G_n[l_n, x(t)] - F, \tag{5.87}$$

где F— сумма однородных функционалов степени меньше n. При этом ядро l_n в (5.87) равно

$$l_n(\tau_1, \ldots, \tau_n) = \delta(\tau_1 - \sigma_1) \ldots \delta(\tau_n - \sigma_n).$$

Используем соотношение (5.86) и свойство ортогональности, тогда для $m = n$

$$\overline{G_n[k_n, x(t)]\,[G_n[l_n, x(t)] - F]} = \overline{G_n[k_n, x(t)\,G_n[l_n, x(t)]}. \tag{5.88}$$

Применяя формулы для среднего значения произведения функционалов Винера, получаем

$$\begin{aligned}
&\overline{G_n[k_n, x(t)]\,G_n[l_n, x(t)]} = \\
&= n!\; N^n \int_{E_n} k_n(\tau_1, \ldots, \tau_n)\, l_n(\tau_1, \ldots, \tau_n)\, d\tau_1 \ldots d\tau_n = \\
&= n!\; N^n \int_{E_n} k_n(\tau_1, \ldots, \tau_n)\, \delta(\tau_1 - \sigma_1) \ldots \delta(\tau_n - \sigma_n)\, d\tau_1 \ldots d\tau_n = \\
&= n!\; N^n k_n(\sigma_1, \ldots, \sigma_n).
\end{aligned} \tag{5.89}$$

Определим теперь функции взаимной корреляции для $m<n$. Если m и n — четные, то высшая степень функционала из (5.89) для $m<n$ и четных равна n—2. В этом случае усредняемый член при вычислении взаимнокорреляционной функции получается таким:

$$\overline{x(t-\tau_1)\ldots x(t-\tau_{n-2})\,x(t-\sigma_1)\ldots x(t-\sigma_n)} \text{ для } n>2. \quad (5.90)$$

Среднее от произведения сигналов типа белый шум может быть заменено произведением средних, взятых во всевозможных комбинациях пар переменных. С другой стороны, в формуле (5.89) число переменных σ_n на две единицы больше, чем переменных τ_i. Поэтому при усреднении в (5.90) для равных между собой σ_n возникают две δ-функции или больше. Таким образом, для четных значений m и n больше двух получим

$$\left.\begin{array}{l}\overline{x(t-\sigma_1)\ldots x(t-\sigma_j)\ldots x(t-\sigma_n)}\\ \overline{x(t-\tau_1)\,x(t-\tau_2)\,x(t-\sigma_1)\ldots x(t-\sigma_n)}\\ \cdots\cdots\cdots\cdots\cdots\cdots\\ \overline{x(t-\tau_1)\ldots x(t-\tau_{n-2})x(t-\sigma_1)\ldots x(t-\sigma_n)}\end{array}\right\}\begin{array}{l}=0 \text{ для } \sigma_i\neq\sigma_j\\ \text{δ-функция при}\\ \sigma_i=\sigma_j\end{array} \quad (5.91)$$

Если же m и n будут противоположной четности, то из-за свойств среднего от произведения нечетного числа сигналов типа белый шум, $\overline{G_m G_n}$ будет равно нулю. Если m и n нечетны, то, рассуждая аналогично (когда m и n четны), получаем

$$\left.\begin{array}{l}\overline{x(t-\tau_1)\,x(t-\sigma_1)\ldots x(t-\sigma_j)\ldots x(t-\sigma_n)}\\ \overline{x(t-\tau_1)x(t-\tau_2)\,x(t-\tau_3)\ldots x(t-\sigma_1)\ldots x(t-\sigma_n)}\\ \cdots\cdots\cdots\cdots\cdots\cdots\\ \overline{x(t-\tau_1)\ldots x(t-\tau_{n-2})\,x(t-\sigma_1)\ldots x(t-\sigma_n)}\end{array}\right\}\begin{array}{l}=0 \text{ для } \sigma_i\neq\sigma_j\\ \text{δ-функция при}\\ \sigma_i=\sigma_j\end{array} \quad (5.92)$$

Таким образом, окончательно получаем

$$K_n(\sigma_1,\ldots,\sigma_n)=\frac{1}{n!\,N^n}\overline{y(t)\,y_n(t)} \text{ для } \sigma_i\neq\sigma_j\,(i,j=1,2,\ldots,n). \quad (5.93)$$

Хотя (5.91), (5.92), (5.93) и не позволяют определять ядра при равных значениях времени запаздывания, однако всегда можно сколь угодно приблизить эти значения. Однако попытаемся снять указанное ограничение. Ограничение на запаздывание σ_i накладывается при определении ядер порядка выше первого. Для ядра второго порядка дельта-функция возникает из-за усреднения (5.81). Поэтому если из реакции системы консультирования *y(t)* вычесть $G_0 = \overline{y}$ (это возможно, поскольку G_0 вычисляется раньше, чем G_2), то дельта-функция исчезает и будет получено соотношение

$$\overline{[y(t) - G_0[k_0,\ x(t)]]\, y_2(t)} = 2N^2 k_2(\sigma_1,\ \sigma_2) \tag{5.94}$$

или

$$k_2(\sigma_1,\ \sigma_2) = \frac{1}{2N^2}\,\overline{[y(t) - G_0[k_0,\ x(t)]]\, y_2(t)}. \tag{5.95}$$

Аналогично для $n>2$ дельта-функции возникают из-за усреднения функционалов (все они определяются раньше, чем G_n)

$$k_n(\sigma_1, \ldots,\ \sigma_n) = \frac{1}{n!\ N^n}\,\overline{\left\{y(t) - \sum_{m=0}^{n-1} G_m[k_m,\ x(t)]\right\} y_n(t)}, \tag{5.96}$$

причем равенство (5.96) оказывается справедливым для всех σ_i (i= 1, 2, ..., n).

Определение ядер Винера при не белом процессе.

Во многих задачах при исследовании систем консультирования спектр воздействия может отличаться от спектра белого гауссова процесса, иметь, например, дробно-рациональную спектральную плотность. Приведем обобщение изложенной теории на этот случай. Рассмотрим систему Θ (рис. 5.23) с воздействием $z(t)$, представляющим собой не белый гауссов процесс с дробно-рациональной спектральной плотностью $S_z(\omega)$.

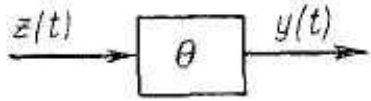


Рис. 5.23.

Эту плотность $S_z(\omega)$ можно разложить на множители:

$$S_z(\omega) = S_z^+(\omega)\, S_z^-(\omega), \tag{5.97}$$

где $S^+_z(\omega)$ — комплексная сопряженная величина $S^-_z(\omega)$. Кроме того, все полюса и нули $S^+_z(\omega)$ находятся в левой половине комплексной плоскости s, где $s = c_0 + j\omega$. Таким образом, $S^+_z(\omega)$ и $1/S^+_z(\omega)$ могут быть реализованы путем преобразования белого гауссова процесса линейной системой.

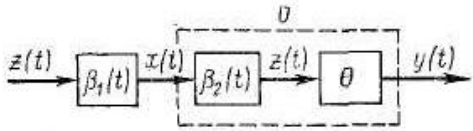


Рис. 5.24.

Тогда систему Θ можно рассмотреть в эквивалентной форме, представленной на рис. 5.24, где передаточные функции двух линейных систем консультирования $\beta_1(t)$ и $\beta_2(t)$ имеют вид

$$W_1(j\omega)=\frac{1}{S_z^+(\omega)};\ W_2(j\omega)=S_z^+(\omega). \tag{5.98}$$

Таким образом, система *O* образуется последовательным соединением линейной системы с импульсной переходной функцией $\beta_2(t)$ и системы Θ. Тогда $x(t)$—воздействие системы консультирования, представляет собой белый гауссов процесс с единичным уровнем спектральной плотности. В соответствии с уравнением (5.93) ядра Винера системы *O* будут иметь вид

$$W_1(j\omega)=\frac{1}{S_z^+(\omega)};\ W_2(j\omega)=S_z^+(\omega). \tag{5.99}$$

за исключением случая, когда два или более значений τ_i равны между собой. Из (5.99) видно, что для определения $\{k_n\}$ необходимо знать функцию взаимной корреляции:

$$R_{yx}(\tau_1,\ldots,\tau_n)=\overline{y(t)\,x(t-\tau_1)\ldots(t-\tau_n)}. \tag{5.100}$$

Однако мы располагаем лишь воздействием $z(t)$, и поэтому выразим искомую функцию R_{yx} через взаимную корреляционную функцию между реакцией $y(t)$ и многомерной задержкой воздействия $z(t)$. При подстановке соотношения

$$x(t)=\int_{-\infty}^{\infty}\beta_1(\sigma)\,z(t-\sigma)\,d\sigma \tag{5.101}$$

в уравнение для R_{yx} получим формулу

$$R_{yx}(\tau_1,\ldots,\tau_n)=\int_0^{\infty}\beta_1(\sigma_1)\,d\sigma_1\ldots\int_0^{\infty}\beta_1(\sigma_n)\,d\sigma_n R_{yz}[(\tau_1-\sigma_1),\ldots,(\tau_n-\sigma_n)], \tag{5.102}$$

где

$$R_{yz}(\tau_1,\ldots,\tau_n)=\overline{y(t)\,z(t+\tau_1)\ldots z(t+\tau_n)} \tag{5.103}$$

— взаимная корреляционная функция между реакцией $y(t)$ и многомерной задержкой воздействия $z(t)$.

В частотной области уравнение (5.103) можно представить в виде

$$S_{yx}(\omega_1, \ldots, \omega_n) = W_1(\omega_1) W_1(\omega_2) \ldots W_1(\omega_n) S_{yz}(\omega_1 \ldots, \omega_n), \quad (5.104)$$

где

$$S_{yz}(\omega_1, \ldots, \omega_n) = \frac{1}{(2\pi)^n} \int_{-\infty}^{\infty} \exp(-j\omega_1\tau_1)\, d\tau_1 \ldots$$

$$\ldots \int_{-\infty}^{\infty} \exp(-j\omega_n\tau_n)\, d\tau_n R_{yz}(\tau_1, \ldots, \tau_n), \quad (5.105)$$

а передаточная функция $W_1(\omega)$ определяется формулой

$$W_1(\omega) = \int_{-\infty}^{\infty} \beta_1(t) \exp(-j\omega t)\, dt. \quad (5.106)$$

Подставив $W_1(\omega)$ в формулу (5.104), получим искомое выражение в частотной области:

$$S_{yx}(\omega_1, \ldots, \omega_n) = \frac{S_{yz}(\omega_1, \ldots, \omega_n)}{S_z^+(\omega_1) \ldots S_z^+(\omega_n)}. \quad (5.107)$$

Теперь для определения ядер $\{k_n\}$ в соответствии с (5.99) с помощью взаимной корреляции R_{yz} можно использовать соотношение (5.102) или (5.105).

После того как ядра $\{k_n\}$ определены, систему консультирования можно представить в виде, изображенном на рис. 5.25.

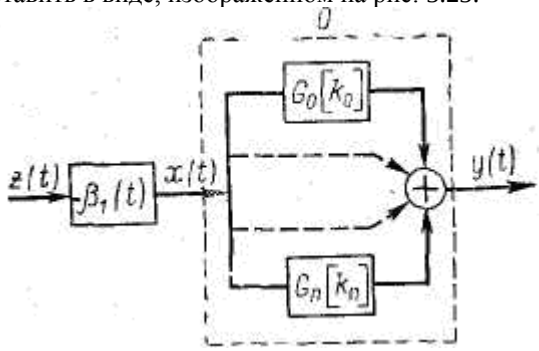


Рис. 5.25.

Модифицированное представление этой системы показано на рис. 5.26.

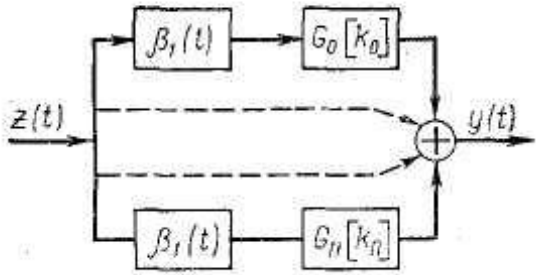


Рис. 5.26.

На этом рисунке видно, что выходы параллельных ветвей ортогональны для *z(t)*. Таким образом, нелинейная система консультирования Θ разложена на ряд функционалов, которые ортогональны для гауссовых реакций со спектральной плоскостью $S_z(\omega)$.

В заключение покажем, каким образом можно вычислить функционалы, ортогональные относительно *z(t)*, не прибегая к построению дополнительных линейных систем консультирования $\beta_1(t)$ и $\beta_2(t)$. Это можно сделать подстановкой (5.101) в (5.87). Обозначим эти функционалы через $\aleph_n$. Первые три функционала имеют вид

$$\aleph_0[k_0, z(t)] = k_0,$$

$$\varkappa_1[k_1, z(t)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_1(\sigma_1)\beta_1(\tau_1-\sigma_1)z(t-\tau_1)\,d\sigma_1 d\tau_1,$$

$$\varkappa_2[k_2, z(t)] = \int_{\infty-\infty}^{\infty}\int_{-\infty-\infty}^{\infty}\int_{-\infty-\infty}^{\infty} k_2(\sigma_1, \sigma_2)\beta_1(\tau_1-\sigma_1)\beta_1(\tau_2-\sigma_2)\times$$

$$\times z(t-\tau_1)z(t-\tau_2)\,d\sigma_1 d\sigma_2 d\tau_1 d\tau_2 -$$

$$-2\pi\int_{-\infty}^{\infty} k_2(\sigma_1, \sigma_2)\,d\sigma_2, \qquad (5.108)$$

где

$$\beta_1(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{\exp(j\omega t)}{S_z^+(\omega)} d\omega. \quad (5.109)$$

Тогда выходной сигнал системы Θ, выраженный через эти функционалы, при входном сигнале *z(t)* может быть представлен ортогональным разложением:

$$y(t) = \sum_{n=0}^{\infty} \varkappa_n [k_n,\ z(t)]. \quad (5.110).$$

Теперь рассмотрим определение ядер Винера по экспериментальным данным и приведем некоторые примеры этих ядер. Как уже говорилось, ядра Винера целесообразно определять особенно в тех случаях, когда весьма сложно осуществить декомпозицию системы консультирования. В качестве такой «системы» возьмем лицо, формирующее рекомендации, выполняющего консультационные операции по формированию рекомендаций решения задач консультируемой проблемы. Используя специальный прибор, будем показывать ему некоторую реализацию белого гауссова процесса и регистрировать реакцию на этот процесс. (Лицо, формирующее рекомендации должно отслеживать реализацию случайного процесса с помощью пишущего визира.) Будем иметь две реализации случайного процесса: *x(t)*—вход и *y(t)* —выход. Теперь на основе изложенной теории можно рассчитать ядра Винера любого порядка. Это можно сделать с помощью обработки данных реализаций *x(t)* и *y(t)* на ЭВМ.

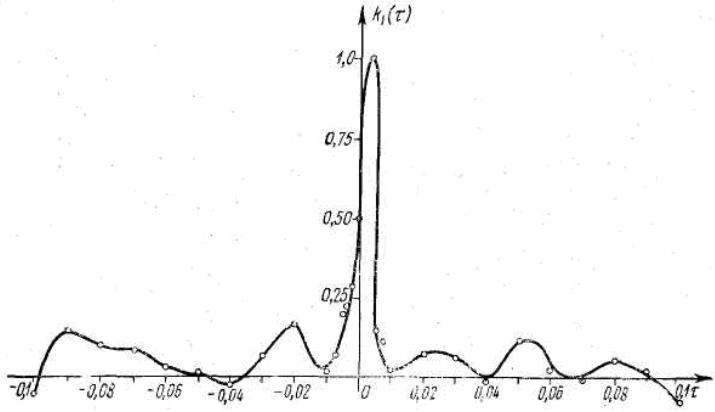


Рис. 5.27.

Примеры ядер первого и второго порядка приведены на рис. 5.27 и 5.28.

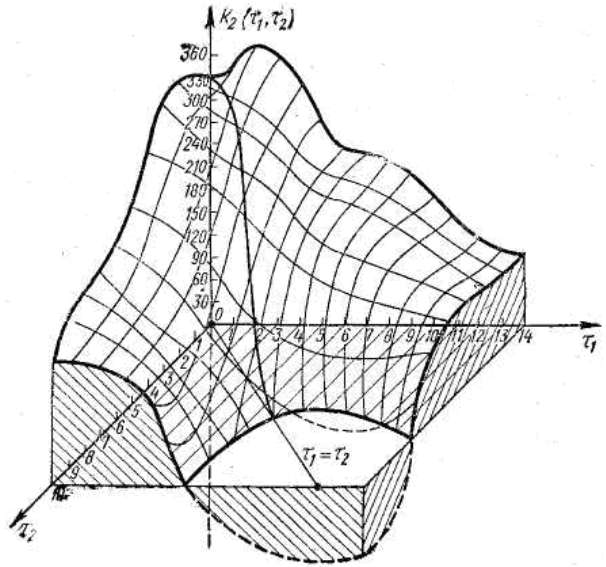


Рис. 5.28.

Анализируя ядро первого порядка, можно видеть, что оно является типичным для импульсной переходной функции линейной системы консультирования с запаздывающим аргументом. Здесь запаздывание составляет порядка 0,2 с. На рис. 5.28 показано ядро Винера второго порядка. Здесь, как и на рис. 5.27, дано множество дискретных значений; ядра соединены на графике плавными кривыми. Очевидно, что эти наборы значений могут быть аппроксимированы аналитическими зависимостями. Ясно, что ***предлагаемый материал может служить исходным при исследованиях систем консультирования***.

Таким образом, ***используя описание системы консультирования с помощью ряда Вольтерра или ряда Винера, можно либо осуществить анализ системы консультирования, либо синтезировать требуемые характеристики системы консультирования, либо выполнить диагностику.***

5.6. Основы построения систем автоматизированного консультирования (САК)

5.6.1. Цели создания и функции САК

В любом консультационном процессе на самом общем уровне можно рассматривать следующие виды деятельности ЛФР,а: формирование новых рекомендаций, их анализ и выбор рациональных вариантов.

Формирование новых рекомендаций относится к творческому виду деятельности, во многом определяемому индивидуальными факторами, присущими ЛФР,у. Этот вид деятельности практически не формализуем, хотя отдельные эвристические приемы поиска новых вариантов рекомендаций можно алгоритмизировать.

Анализ вариантов рекомендаций проводится ЛФР,ом на основе физических, математических и информационных моделей консультируемых проблем. Рассматривая этот вид его деятельности, можно отметить, что он в наибольшей степени может быть возложен на САК, поскольку основан на формальном описании различных свойств консультируемых проблем и в широкой степени может использовать различные математические модели.

На этапе выбора рациональных вариантов рекомендаций одна из основных проблем, стоящих перед ЛФР,ом, заключается в оценке вариантов рекомендаций по многим критериям, как правило, в условиях существенной неопределенности. Перебор значительного количества вариантов с учетом их оценок по многим критериям возможен лишь при наличии в САК специального ***комплекса моделей, поддерживающих процедуру выбора в диалоговом режиме***.

Таким образом, ***САК будем рассматривать как человеко-машинную систему, способную оказать помощь ЛФР,у на всех этапах консультационного процесса***.

Процесс создания САК следует рассматривать как весьма сложную комплексную проблему, успешное решение которой во многом зависит от характера консультируемых проблем, достигнутого уровня и методологии консультирования, используемой вычислительной техники.

Тем не менее можно предложить достаточно ***общие теоретические и методологические*** основы построения САК и состава обеспечивающих подсистем.

Для САК различного класса в качестве основного структурного элемента будем принимать ***функциональную подсистему***, которая призвана реализовывать отдельные фрагменты консультационного процесса.

Функциональные подсистемы будем подразделять на ***консультирующие и обслуживающие***. К первым будем относить подсистемы, непосредственно выполняющие консультационные операции по формированию рекомендаций решения задач консультируемой проблемы, ко вторым — поддерживающие работоспособность вспомогательных подсистем (например, подсистем графического отображения информации).

Работа функциональных подсистем во многом определяется программно-информационными комплексами САК, позволяющими конкретизировать деятельность ЛФР,а , обеспечить систему реальными данными из предметной области консультирования и алгоритмами обработки имеющейся информации.

Накопленный опыт консультирования проблем в различных предметных областях показывает, что автоматизация формирования рекомендаций по решению задач консультируемых проблем— это область наиболее эффективного использования ЭВМ. Но в то же время становится ясным, что главное направление здесь — это не автоматизация формирования рекомендации по решению отдельной задачи консультируемой проблемы, а ***автоматизация получения комплекса формируемых рекомендаций, реализация которых позволит решить всю консультируемую проблему в целом***.

Такой подход к созданию САК базируется на стремлении существить ***основную задачу — повысить качество и эффективность формируемых рекомендаций, применять методы оптимального консультирования.*** Следует, однако сознавать, что ***САК — это вспомогательное средств, а но замена консультанта.***

Цель создания САК. ***Под автоматизацией консультирования будем понимать систематическое применение ЭВМ в процессе консультирования при научно обоснованном разделении функций между ЛФР,ом и ЭВМ, научно обоснованном выборе методов машинного решения консультационных задач.***

Цель автоматизации консультирования — повысить качество формируемых рекомендаций, снизить материальные затраты, сократить сроки консультирования и снизить рост числа работников, участвующих в формировании рекомендаций по решению задач консультируемых проблем.

Научно обоснованное распределение функций межд человеком и ЭВМ подразумевает, что человек должен решать задачи творческого характера, а ЭВМ — задачи удовлетворяющие двум требованиям: 1) возможности алгоритмизации; 2) большей эффективности исполнена алгоритма на ЭВМ по сравнению с ручным решением.

Эффективность машинных методов консультирования заключается в возможности осуществления ***математического моделирования консультируемых проблем***. При этом следует иметь в виду, что при консультировании число альтернатив необозримо. Поэтому нельзя ставить задачу создания универсальной САК, а необходимо решать вопросы автоматизации консультирования для конкретного семейства САК.

Системы автоматизированного консультирования предназначены для выполнения консультационных операций (процедур) в автоматизированном режиме. САК создаются в различных консультационных организациях с целью: ***повышения качества и технико-экономического уровня формируемых рекомендаций по решению задач консультируемых проблем; повышения эффективности сформированных рекомендаций, уменьшения затрат на их разработку и реализацию; сокращения сроков, уменьшения трудоемкости консультирования и повышения качества консультационной документации.***

Достижение указанных целей создания САК возможно при условиях:

- систематизации и совершенствования консультационных процессов на основе применения математических методов и средств вычислительной техники;
- комплексной автоматизации консультационных работ в консультационной организации с необходимой перестройкой ее структуры и кадрового состава;
- повышения качества управления консультированием;
- применения эффективных математических моделей консультируемых проблем;
- использования методов многовариантного консультирования и оптимизации формируемых рекомендаций;
- автоматизации трудоемких и рутинных консультационных работ;
- замены натурных испытаний и макетирования математическим моделированием;
- создания единых банков данных, содержащих систематизированные сведения справочного характера, необходимые для автоматизированного консультирования;
- унификации и стандартизации методов консультирования.

Для достижения целей создания САК необходимы:

— совершенствование консультационных процессов на основе применения математических методов и средств вычислительной техники;

— автоматизация процесса поиска, обработки и выдачи информации;

— использование методов оптимизации и многовариантного консультирования; применение эффективных математических моделей консультируемых проблем;

— создание банков данных, содержащих систематизированные сведения справочного характера, необходимые для автоматизированного консультирования;

—повышение качества оформления консультационной документации;

— повышение творческой доли труда консультантов за счет автоматизации нетворческих работ;

— унификация и стандартизация методов консультирования;

— подготовка и переподготовка специалистов;

— взаимодействие с автоматизированными системами различного уровня и назначения.

Комплекс средств автоматизации консультирования включает в себя ***методическое, лингвистическое, математическое, программное, техническое, информационное и организационное обеспечение***.

Состав САК. Система автоматизированного консультирования — система, объединяющая технические средства, математическое и программное обеспечение, параметры и характеристики которых выбирают с максимальным учетом особенностей задач консультирования. В САК обеспечивается удобство использования программ за счет применения средств оперативной связи консультанта с ЭВМ, специальных проблемно-ориентировочных языков и информационно-справочной базы.

Структурными составляющими САК являются подсистемы, обладающие всеми свойствами систем и создаваемые как самостоятельные системы.

По назначению, как мы уже говорили, подсистемы САК будем подразделять на два вида: консультационные и обслуживающие.

К консультационным будем относить подсистемы, выполняющие консультационные процедуры и операции.

К обслуживающим будем относить подсистемы, предназначенные для поддержания работоспособности консультационных подсистем:

— графического отображения консультируемых проблем;

— документирования;

— информационного поиска и др.

Структурное единство подсистем САК обеспечивается строгой регламентацией связей между компонентами различных видов обеспечения, объединенных общей для данной подсистемы целевой

функцией. ***Компонента представляет собой элемент обеспечения, выполняющий определенную функцию в подсистеме.***

Структурное объединение подсистем в систему обеспечивается связями между компонентами, входящими в подсистемы.

Составными структурными частями САК являются подсистемы, обладающие всеми свойствами системы и создаваемые как самостоятельные системы.

В зависимости от отношения к консультируемой проблеме будем различать два вида консультируемых подсистем: объектно-ориентированные (объектные); объектно-независимые (инвариантные).

Объектные подсистемы выполняют одну, или несколько консультационных процедур или операций, непосредственно зависимых от конкретной консультируемой проблемы.

Инвариантные подсистемы выполняют унифицированные консультационные процедуры и операции.

Подсистема состоит из компонентов САК (далее — компонентов), объединенных общей для данной подсистемы целевой функцией и обеспечивающих функционирование этой подсистемы. Компонента представляет собой элемент обеспечения, выполняющий определенную функцию в подсистеме:

- ***методическое обеспечение*** — документы, в которых отражены состав, правила отбора и эксплуатации средств автоматизации консультирования;
- ***лингвистическое обеспечение***—языки консультирования, терминология;
- ***математическое обеспечение*** — методы, математические модели, алгоритмы;
- ***программное обеспечение***—документы с текстами программ, программы на машинных носителях и эксплуатационные документы;
- ***техническое обеспечение*** — устройства вычислительной и организационной техники, средства передачи данных, измерительные и другие устройства или их сочетания;
- ***информационное обеспечение*** — документы, содержащие описания стандартных консультационных процедур, типовых рекомендаций, а также файлы и блоки данных на машинных носителях с записью указанных документов;
- ***организационное обеспечение*** — положения, инструкции, приказы, штатные расписания, квалификационные требования и другие документы, регламентирующие организационную структуру подразделений и их взаимодействие с комплексом средств автоматизации консультирования.

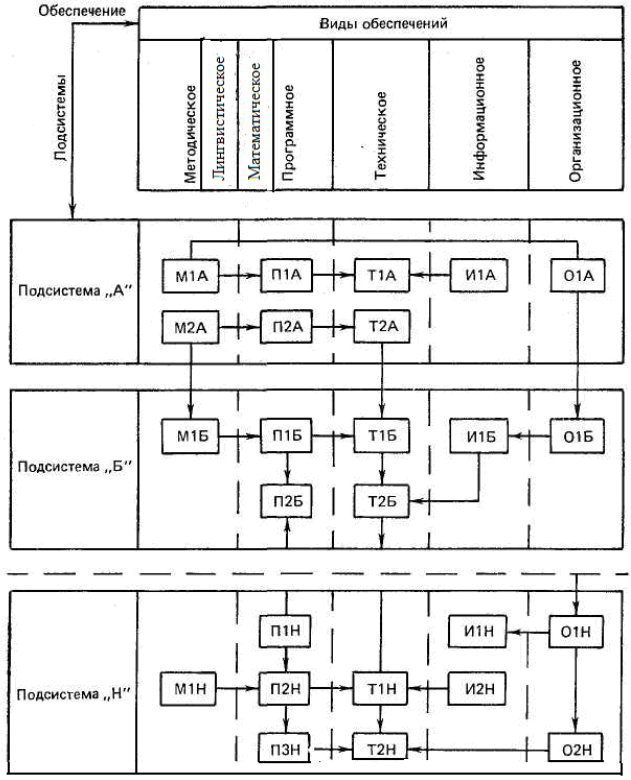


Рис. 5.29. Матричная структура САК:
□ — компонента САК

Введение структурного понятия «компонента» как некоторого элементарного «кирпичика» системы позволяет раскрыть внутреннюю структуру подсистемы и указать конкретные связи между подсистемами не только иерархические, но и методические, информационные и т. д. На рис. 5.29 приведена двумерная структурная схема САК, основанная на понятии компоненты. Как видно из рисунка, матричная структура является открытой как по количеству подсистем, так и по видам обеспечения. Связи между подсистемами «А» и «Б» показывают, что иерархически подсистема «Б» подчинена подсистеме «А», а компонента организационного обеспечения 01А является определяющей как для самой подсистемы «А» (задает требования к

компоненте методического обеспечения М1А), так и для подсистемы «Б» (является исходным для компоненты 01 Б). Компоненты могут иметь многократное применение, т. е. одна и та же типовая или унифицированная компонента может применяться в различных подсистемах.

Анализ структурной схемы САК позволяет сделать вывод, что структурное единство подсистемы САК обеспечивается связями между компонентами различных средств обеспечения САК, образующими подсистему, а структурное объединение подсистем в систему — связями между компонентами, входящими в подсистемы.

5.6.2. Классификация САК

В начальный период создания САК должны быть разработаны единый метод и признаки классификации, основные классификационные группировки и правила обозначения САК. При разработке классификации и обозначений САК нами был использован фасетный метод классификации объектов, при котором классификационные признаки (т. е. объект классификации) характеризуются с разных сторон.

По каждому признаку установлены классификационные группы, их характеристики и коды (рис. 5.30).

Рис. 5.30. Классификационные группы САК

Основные классификационные группы САК приведены в табл. 5.1—5.7.

5.1. Классификационные группировки САК по типу консультируемой проблемы

Код	Наименование
1	САК проблем в машиностроения и приборостроения
2	САК проблемных процессов в машиностроении и приборостроении
3	САК проблем в строительства
4	САК проблем в экономике
5—9	Резерв

5.2. Классификационные группировки САК по разновидности консультируемой проблемы

Код	Наименование

Код и наименование группировки устанавливаются по действующим обозначениям документации на консультируемые проблемы, которые консультируются данной системой

5.3. Классификационные группировки САК по сложности консультируемой проблемы

Код	Наименование	Число составных частей консультируемой проблемы
1	САК простых проблем	$<10^2$
2	САК проблем средней сложности	$10^2...10^3$
3	САК сложных проблем	$10^3...10^4$
4	САК очень сложных проблем	$10^4...10^6$
5	САК проблем очень высокой сложности	$>10^6$

5.4. Классификационные группировки САК по уровню автоматизации консультирования

Код	Наименование	Объем автоматизированных работ от общего количества консультационных процедур
1	Система низкоавтоматизированного консультирования	<25
2	Система среднеавтоматизированного консультирования	25...50
3	Система высокоавтоматизированного консультирования	>50 (применяются методы многовариантного оптимального консультирования)

5.5. Классификационные группировки САК по комплексности автоматизации консультирования

Код	Наименование
1	Одноэтапная САК
2	Многоэтапная САК
3	Комплексная САК (выполняет все этапы консультирования)

5.6. Классификационные группировки САК по количеству выпускаемых проектных документов

Код	Наименование	Число выпускаемых за год консультационных документов в пересчете на формат А4
1	САК малой производительности	$\leq 10^5$
2	САК средней производительности	$10^5 ... 10^6$
3	САК высокой производительности	$\geq 10^6$
4 - 9	Резерв	—

5.7. Классификационные группировки САК по числу уровней в структуре технического обеспечения

Код	Наименование	Характеристика технических средств системы
1	Одноуровневая САК	ЭВМ среднего или высокого класса со штатным набором периферийных устройств, который может быть дополнен средствами обработки. графической информации
2	Двухуровневая САК	ЭВМ среднего или высокого класса и одно или несколько автоматизированных рабочих мест консультанта (АРМ), включающих в себя мини-компьютеры.
3	Трехуровневая САК	ЭВМ высокого класса, одно или несколько АРМ и периферийное программно-управляемое оборудование.
4—9	Резерв	

5.6.3. Основные принципы построения САК.

Разработка САК представляет собой крупную научно-техническую проблему, а ее внедрение требует значительных капиталовложений. Можно выделить следующие основные принципы их построения:

САК — человеко-машинная система. Все созданные и создаваемые с помощью ЭВМ системы консультирования являются автоматизированными, важную роль в них играет человек — специалист по разработке САК.

В настоящее время и, по крайней мере, в ближайшие годы создание САК «не угрожает» монополии консультанта при формировании рекомендаций в процессе консультирования. Консульант должен формировать рекомендации, во-первых, по решению всех задач консультируемой проблемы, формализация которых не достигнута, во-вторых, рекомендации, формирование которых осуществляется консультантом на основе эвристических способностей более эффективны, чем рекомендации современной ЭВМ на основе вычислительных возможностей. Тесное взаимодействие консультанта и ЭВМ в процессе консультирования — один из принципов построения и эксплуатации САК.

САК — иерархическая система. Она реализует комплексный подход к автоматизации всех уровней консультирования. Блочно-иерархический подход к консультированию должен быть сохранен при применении САК. Иерархия уровней консультирования отражается в структуре специального программного обеспечения САК в виде иерархии подсистем.

Следует особо подчеркнуть целесообразность обеспечения комплексного характера САК, так как автоматизация консультирования на одном из уровней при сохранении старых форм консультирования на соседних уровнях оказывается значительно менее эффективной, чем полная автоматизация всех этапов. Иерархическое построение относится не только к специальному программному обеспечению, но и к техническим средствам САК, разделяемых на ***центральный вычислительный комплекс и автоматизированные рабочие места консультантов (АРМК)***.

САК — совокупность информационно согласованных подсистем. Этот очень важный принцип должен относиться не только к связям между крупными подсистемами, но и к связям между более мелкими частями подсистем. Информационная согласованность означает, что ***все или большинство возможных последовательностей задач консультирования обслуживаются информационно согласованными программами***.

Две программы являются информационно согласованными, если все те данные, которые представляют собой объект переработки в обеих программах, входят в числовые массивы, не требующие изменений при переходе от одной программы к другой. Так, информационные связи могут проявляться в том, что результаты решения одной консультационной задачи будут исходными данными для другой задачи. Если для согласования программ требуется существенная переработка общего массива данных с участием человека, который добавляет недостающие параметры, вручную перекомпоновывает массив или изменяет значения отдельных параметров, то это значит, что программы плохо согласованы. Ручная перекомпоновка массива ведет к существенным временным задержкам, росту числа ошибок и поэтому снижает эффективность работы САК. Плохая информационная согласованность превращает САК в совокупность автономных программ, при этом из-за неучета в подсистемах многих факторов, оцениваемых в других подсистемах, снижается также качество формируемых рекомендаций.

Принцип информационной согласованности подсистем часто представляют близким по смыслу принципу оптимальности связей человека с ЭВМ внутри САК. При этом подчеркивается сторона автоматизированного консультирования, требующая рационального распределения функций между человеком и ЭВМ.

Автоматическая связь программ без ручной перекомпоновки массивов — элемент оптимальности связей между человеком и ЭВМ. Техническое обеспечение оперативной связи человека с ЭВМ посредством дисплея — другой элемент такой связи.

Еще одним близким по смыслу, но не полностью совпадающим с рассмотренными является принцип оптимальности связей между САК и внешней средой. Если каждый раз при консультировании очередной проблемы заново вводятся в систему не только действительно специфические новые исходные данные, но и сведения справочного характера, например параметры унифицированных рекомендаций, то тем самым получают нерациональную организацию связей САК с окружающей средой. Очевидно, что ***все данные, используемые многократно при консультировании разных проблем, должны храниться системой в банке данных — информационном обеспечении САК.***

САК — открытая и развивающаяся система. Существует, по крайней мере, две существенные причины, по которым САК должна быть изменяющейся во времени системой. Во-первых, разработка

столь сложного объекта, как САК, занимает продолжительное время и экономически выгодно вводить в эксплуатацию части системы по мере их готовности. Введенный в эксплуатацию ***базовый вариант системы*** в дальнейшем расширяется. Во-вторых, постоянный прогресс вычислительной техники и вычислительной математики приводит к появлению новых, более совершенных математических моделей и программ, которые должны заменять старые, менее удачные аналоги. Поэтому САК должна быть открытой системой, т. е. обладать свойством удобства включения новых методов и средств.

САК — специализированная система с максимальным использованием унифицированных модулей. Требования высокой эффективности и универсальности, как правило, противоречивы. Применительно к САК это положение справедливо. Высокой эффективности САК, выражаемой прежде всего малыми временными и материальными заратами при решении консультационных задач, добиваются за счет специализации систем. Но очевидно, что при этом растет число различных САК. Чтобы снизить расходы на разработку многих специализированных САК, целесообразно строить их на основе максимальною использования ***унифицированных составных частей***. Необходимым условием унификации является поиск общих положений в моделировании, анализе и синтезе разнородных объектов.

5.7. Комплекс средств обеспечения САК

Структурное единство каждой из подсистем и системы в целом обеспечивается с помощью системных компонентов САК, которые представляют собой совокупность методического, информационного, лингвистического, математического, программного, технического и организационного обеспечений. Каждая компонента выполняет соответствующие функции в САК и характеризуется определенным набором средств. Остановимся на большинстве средств обеспечения САК более детально.

Методическое обеспечение. Как уже отмечалось, методическое обеспечение — это есть совокупность документов, устанавливающих состав и правила отбора и эксплуатации средств автоматизации консультирования. Оно должно поддерживать консультационный процесс формирования рекомендаций по решению задач консультируемой проблемы с помощью САК в целом. Формирование рекомендаций в САК характеризуется логической схемой консультационного автоматизированного процесса, определяющей порядок и правила формирования рекомендаций в ходе консультирования проблемы.

Один из возможных вариантов схемы процесса автоматизированного консультирования представлен на рис. 5.31.

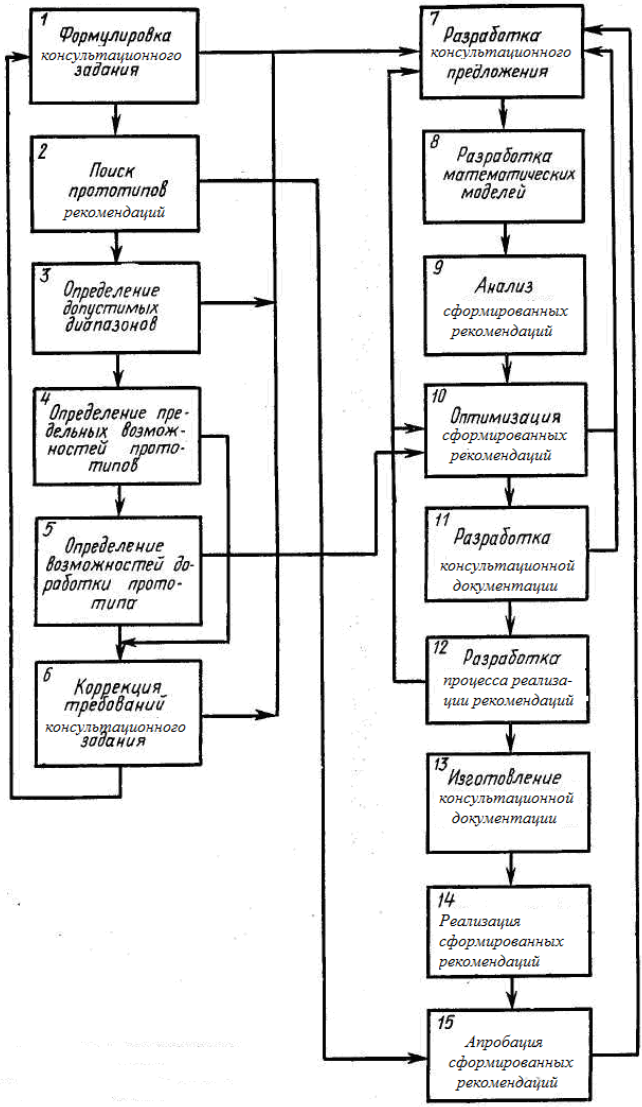


Рис. 5.31. Схема структурная процесса автоматизированного консультирования

Схема предполагает следующий порядок этапов проведения консультационных работ.

1. Разрабатывается консультационное задание на формирование рекомендаций по решению задач консультируемой проблемы.

2. Осуществляется поиск среди существующих рекомендаций, таких, которые удовлетворяют требованиям на формирование рекомендаций. Этот этап дает возможность исключить неоправданные затраты на формирование новых рекомендаций, а также позволяет наиболее эффективно использовать номенклатуру существующих рекомендаций. Если в банке данных САК есть рекомендация с требуемыми характеристиками, то ЭВМ выдает все необходимые сведения о ней. Если нет сведений о рекомендациях с требуемыми характеристиками, то ЭВМ констатирует этот факт и консультант в диалоговом режиме выделяет «критические» параметры требований и затем осуществляется переход к следующему этапу консультирования. (Под «критическими» параметрами будем понимать параметры и характеристики, не удовлетворяющие требованиям консультационного задания.)

3. Консультант назначает допустимые отклонения по «критическим» параметрам. Производится поиск рекомендаций-прототипов с характеристиками, близкими к требуемым. Если их нет, то осуществляется переход к седьмому этапу; если есть, то к четвертому этапу консультирования.

4. Определяются предельно допустимые возможности рекомендаций-прототипов («запас») по выделенным «критическим» параметрам. Если по уточненным данным рекомендация обеспечивают выполнение консультационных требований в полном объеме, то осуществляется переход к 6-му этапу, а если нет, то к 5-му этапу.

5. Определяется возможность доработки рекомендации (например, параметрическая, структурная коррекция, оптимальный выбор параметров, изменение методов формирования рекомендаций и т. п.). Если доработка дает желаемый результат, т. е. требуемые характеристики выполнены, то осуществляется переход к 10-му этапу. Если нет, то к следующему этапу.

6. Определяется возможность коррекции исходных требований консультационного задания с учетом полученных данных о рекомендациях-прототипах и возможностях их доработки. Если коррекция возможна, она осуществляется, и процесс консультирования повторяется с первого этапа. В противном случае коррекция невозможна, осуществляется переход к 7-му этапу.

7. Принимаются принципиальные решения о реализации сформированных рекомендаций, строится схемная модель рекомендаций, отражающая протекающие процессы и явления в консультируемой проблеме. Модель может быть задана консультантом в виде условной схемы на экране дисплея или получена с помощью ЭВМ, в диалоговом режиме.

8. Производится построение или выбор математического аппарата по анализу сформированных рекомендаций, используемых при решении задач консультируемой проблемы.

9. Осуществляется анализ принятых сформированных рекомендаций по решению задач консультируемой проблемы. Если они имеют характеристики, близкие к требуемым, то переходят к 10-му этапу.

10. Проводится оптимизация сформированных рекомендаций по сформированным функциям качества. Если в ходе оптимизации найдены рекомендации, обеспечивающие наилучшим образом выполнение требований консультационного задания и дополнительных рекомендаций, то осуществляется переход к 11-му этапу, если нет— к 7-му этапу.

11. Проводится проработка отобранных рекомендаций, окончательный выбор вариантов реализации рекомендаций, выпускается консультационная документация.

12. Осуществляется подготовка к реализации выбранных сформированных рекомендаций (проектируются технологические процессы реализации рекомендаций, выбираются основные и вспомогательные средства реализации рекомендаций и т. п.). В зависимости от полученных результатов переходят к 7-му, 10-му или следующим этапам консультирования.

13. Изготовляется консультационная и проектно-сметная документация. Создаются управляющие программы.

14. Осуществляются реализация сформированных рекомендаций. На этом этапе функции САК могут быть ограничены только контролем за общим ходом процесса реализации рекомендаций, накоплением и обработкой определенных данных.

15. Проводится апробация реализованных сформированных рекомендаций. На этом этапе в САК можно моделировать процесс апробации с целью выбора наиболее рациональной программы проведения апробационных работ (экономия ресурсов, времени и т. п.), а также проводить обработку полученных результатов для дальнейшего использования их в ходе консультационной деятельности.

В случае отрицательного результата апробации переходят в зависимости от характера выявленных недостатков к 7, 10, 12, 13-му этапам консультирования. При положительном результате апробации окончательно формируются рекомендации для решения задач консультируемой проблемы.

Методическое обеспечение является ключом для понимания, назначения и взаимосвязи всех других компонентов САК (информационного, математического, программного обеспечения и т. д.).

Информационное обеспечение. Оно включает в себя совокупность сведений, необходимых для поддерживания процесса автоматизированного консультирования. Разработка информационного обеспечения (ИО) обусловлена решением задач, связанных с организацией хранения, поиска, обеспечения защиты информации, санкционированности доступа, удобства и своевременности представления необходимых сведений пользователям системы и т. д. ИО включает средства для описания и накопления входной, выходной и промежуточной информации, необходимой для консультирования. К средствам описания различных видов информации относятся архивы, библиотеки, банки и базы данных.

Информационное обеспечение включает также традиционные средства формирования и обновления информационных массивов, алгоритмы оптимального размещения и поиска информации. Кроме того, средства информационного обеспечения САК должны осуществлять:

- прием запросов как от операторов-консультантов, так и от подсистем к программам САК;
- обработку запросов и выдачу результатов поиска непосредственному источнику запроса в требуемой для него форме;
- хранение и работу с информацией, представляющей результат всех стадий как ручных, так и автоматизированных процессов формирования рекомендаций;
- реализацию принципа доступности системы, т. е. возможность формирования и приема запроса на рабочем месте консультанта;
- оперирование информацией, определяемой как путем указания на нее, так и по смысловому запросу;
- быстрое внесение изменений и корректировку информации, а также доведение этих сведений до пользователя;
- проверку корректности вводимой информации и гарантию правильности хранимой и выдаваемой информации;
- получение копий требуемых документов в буквенно-цифровой и графической формах и т. п.

Сформулированные задачи обработки информации в САК делают еще более актуальными вопросы широкого использования в составе ИО систем управления базами данных (СУБД).

Под СУБД понимается специальная программная система, поддерживающая структуры и взаимосвязи данных как на логическом (уровне описания данных), так и на физическом уровнях (уровне хранения данных в ЭВМ).

Характерной чертой современных СУБД является их ориентация не на конкретные приложения, а на поддержание принятых структур данных. Это позволяет хранить в одной базе данных самую разнообразную информацию, характеризующую консультируемую проблему с различных точек зрения.

Таким образом, информационный комплекс САК в качестве основных элементов включает в себя базы данных и их системы управления.

Лингвистическое обеспечение. Это сокупность лингвистических средств, включающих в свой состав терминологию и языки консультирования, с помощью которых осуществляется взаимодействие консультанта с системой. Термины и определения, используемые в САК, оговариваются в соответствующих документах (руководящих материалах, нормалях, ГОСТах), что необходимо для однозначного понимания специалистами различных аспектов консультирования.

Особое положение в лингвистическом обеспечении занимают языки программирования, одни из которых обеспечивают общение консультанта с ЭВМ, а другие служат для описания алгоритмов обработки информации в ЭВМ. Существующие языки программирования можно разделить на три класса: машиино-ориентированные, процедурно-ориентированные и проблемно-ориентированные.

Программирование на ***машинно-ориентированных языках*** требует знаний не только сущности задачи и алгоритма ее решения, но и структуры, технических особенностей ЭВМ, способов программирования на ней. Типичным представителем машинно-ориентиромшшых языков является язык ассемблер. Как правило, эти языки используются при написании программ, например, операционных систем. Использование машинно-зависимых языков позволяет делить программы более компактными и быстродействующими. Однако программирование с помощью этих языков является наиболее трудоемким.

Существование ***процедурно-ориентированных языков*** в значительной степени упрощает процесс программирования за счет

включаемых в эти языки специальных средств описания процессов решения различных классов задач. Представление алгоритма на языке данного класса заключается в описании алгоритма в виде последовательности процедурных шагов, детализирующих вычислительный процесс. К наиболее широко используемым процедурно-ориентированным языкам относятся Фортран, Паскаль, Бейсик, Ада, С и др.

Паскаль удобен при программировании логических задач и задач обработки символьных данных.

Язык Ада был разработан группой специалистов с учетом тех возможностей вычислительных систем, которые связаны с их использованием в реальном масштабе времени и организацией параллельных вычислений.

Язык программирования С первоначально был разработан как язык для реализации операционной системы UNIX. Но опыт программирования показал, что этот язык обладает широкими возможностями для программирования различных прикладных задач. Можно сделать вывод, что С — это универсальный язык программирования, достаточно компактный, обладающий механизмами управления вычислениями и структурами данных.

Процедурно-ориентированные языки важны для САК, так как решают задачу совместимости программ для различных типов машин, облегчают взаимодействие человека с ЭВМ, упрощают процессы написания, отладки программ и обучения программированию.

Проблемно-ориентированные языки характеризуются наличием непроцедурных средств, указывающих в основном на то, что должно быть сделано алгоритмом, а не как, т. е. языки отражают сущность, а не способ реализации вычислительного процесса. В программе формулируются соотношения, а не последовательность вычислений, и, таким образом, программист освобождается от обязанности разрабатывать шаги алгоритмов и определять их порядок. К проблемно-ориентированным языкам можно отнести язык Пролог. В Прологе не пишут формул, вместо этого определяют соотношения между объектами и величинами. Язык состоит только из описания и не имеет инструкций.

К лингвистическим средствам САК можно отнести также ***языки запросов к базе данных, манипулирования данными и описания данных.*** Характер этих языков определяется системой управления базой данных, используемой в САК.

В настоящее время все большее внимание уделяется разработке взаимодействия консультантов с ЭВМ на естественном профессиональном языке.

САК должна располагать знаниями, необходимыми для консультирования, процедурами обработки знаний, методами выполнения консультационных процедур, и консультант может обращаться к ЭВМ на естественном языке, инициировать к действию соответствующие программы и алгоритмы, облегчающие ему процесс консультирования.

Математическое обеспечение. Такое обеспечение представляет собой математические методы, модели и алгоритмы, на основе которых осуществляется процесс автоматизированного консультирования. Для правильного понимания роли и назначения этого вида обеспечения в САК рассмотрим вопросы, возникающие в ходе консультирования проблем. На этапах начального формирования рекомендаций по решению задач консультируемой проблемы осуществляются выработка концепций и просмотр общих схемных решений, поиск оптимальных параметров, сравнение альтернативных вариантов рекомендаций, определение функций и структуры формируемых рекомендаций и детальное исследование процессов, сопровождающих их работу. Проработки, выполненные на этих этапах, оказывают решающее влияние на качественные и количественные характеристики рекомендаций. Как уже отмечалось, эти этапы характеризуются сложностью формализации задач консультирования, необходимостью учета большого количества разнообразных требований, критериев и ограничений, большими затратами труда.

Поэтому для выработки обоснованных рекомендаций необходимо в максимальной степени использовать ***математические модели функционирования консультируемых проблем, а также модели консультационного процесса.***

Степень обоснованности выбранных рекомендаций повышается за счет точности описания протекающих процессов и явлений в консультируемых проблемах, а также оптимизации принимаемых консультационных рекомендаций на базе разрабатываемого математического аппарата.

Таким образом, ***разработка математического обеспечения САК является одним из самых ответственных этапов в построении САК, на котором закладываются точность и достоверность принимаемых консультационных рекомендаций, а значит и качество и эффективность работы всей САК в целом.***

В САК должны использоваться множество математических моделей и методов, охватывающих практически все разделы математики.

Математические методы в САК можно условно разбить на следующие группы: имитационного и анимационного моделирования; логического синтеза; оптимизации; синтеза геометрических и чертежно-графических моделей консультируемой проблемы.

Методы ***имитационного и анимационного моделирования*** используются для создания имитационной и анимационной модели консультируемой проблемы и экспериментирования с ней в условиях реальных ограничений, а также для оценки соответствия консультируемой проблемы предъявляемым к ней требованиям и условиям функционирования.

Методы ***логического синтеза*** предназначены для синтеза рекомендаций по решению задач консультируемой проблемы на основе формального описания ее функционирования, так называемого сквозного логического синтеза, начиная от ввода в ЭВМ формализованного задания на консультирование до выдачи ЭВМ структуры рекомендаций с учетом заданных ограничений.

Методы ***оптимизации*** позволяют консультанту на любом этапе формирования рекомендаций провести оптимизацию отдельных рекомендаций или всей их совокупности в целом по формируемым функциям качества. Для этого в системе должны быть предусмотрены возможности:

- назначения параметров оптимизации;
- формирования функций качества, которые позволили бы количественно оценить рассматриваемые рекомендации;
- формирования ограничений, выделяющих в пространстве множество возможных состояний проблемы области, которые соответствуют требованиям консультационного задания, возможностям их реализации и дополнительным рекомендациям;
- выбора методов поиска экстремума функции качества в зависимости от количества параметров оптимизации, количества и вычислительной сложности ограничений, а также от характера и трудоемкости вычисления функции качества;
- выбора исходной точки поиска, которая может быть определена волевым решением или с помощью вычислительных процедур.

Формализация процесса консультирования требует разработки математических моделей консультационных процессов. Информация, касающаяся консультационных процессов, является, как правило, тем связующим звеном, на котором базируется множество математических

моделей, связывающих консультационные характеристики с различными характеристиками функционирования проблемы.

Эффективность математического обеспечения САК во многом определяется достигнутым уровнем единства физических и математических принципов, применяемых для разработки моделей, а также возможностями приспосабливаемости имеющихся в САК моделей к решению конкретных задач на всех этапах консультирования.

Программное обеспечение. Представляет собой совокупность программ, обеспечивающих реализацию функций САК. Анализ задач и особенностей автоматизированного консультирования показывает, что программное обеспечение САК должно обеспечивать выполнение следующих основных функций:

- пользовательских, с помощью которых непосредственно проводится автоматизированное консультирование;
- обеспечивающих, с помощью которых осуществляются структурное и функциональное объединение системы, ее работоспособность и развитие.

С учетом задач и функций, выполняемых различными компонентами ПО (рис. 5.32), его можно разделить на две группы: системное (общее) и специальное (прикладное).

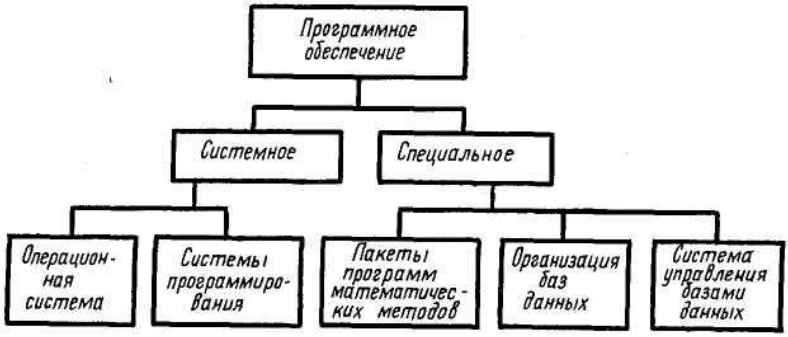


Рис. 5.32. Схема структурная программного обеспечения САК

К системному программному обеспечению относятся системы программирования (трансляторы, редакторы, стандартные программы) и программы операционной системы, включающие управляющие программы, которые выполняют планирование использования ресурсов и координацию работы отдельных устройств, входящих в состав вычислительной системы САК.

Операционная система (ОС) — это программный комплекс, разработанный специально для управления ресурсами ЭВМ, автоматизации разработки прикладных программ и управления процессом их выполнения.

Существуют три основных типа ОС: мультипрограммирования, с разделением времени и реального времени.

Мультипрограммирование — это способ организации работы вычислительной системы, который позволяет нескольким заданиям совместно использовать ее ресурсы. Задание на обработку данных передается ОС из пакета заданий в порядке очереди с учетом присвоенного ему приоритета.

Особенностью мультипрограммирования является то, что после того, как какое-либо задание начинает выполняться, оно полностью занимает ЭВМ до тех пор, пока не будет выполнено, либо не будет вынуждено остановиться по какой-то причине. Такой режим использования ЭВМ, называемый обычно пакетным, позволяет максимально загружать машину, однако создает трудности в организации единого разветвленного вычислительного процесса, когда в системе должно выполняться несколько заданий.

В ОС с разделением времени все задания выполняются одновременно путем циклического выделения ресурсов машины каждому из заданий. Поскольку этот процесс происходит весьма быстро, у пользователей создается иллюзия их одновременного обслуживания.

В тех случаях, когда обработка данных должна проводиться в реальном масштабе времени прохождения какого-либо процесса, применяются ОС реального времени.

Специальное программное обеспечение (СПО) предназначено для решения конкретных консультационных задач. В основе прикладных программ лежат конкретные методы научных дисциплин, используемых в ходе консультирования, а также методы вычислительной математики и программирования. Состав СПО всегда индивидуален и зависит от консультируемой проблемы, специфики и объема задач, решаемых конкретной САК.

Разработка прикладных программ проводится на основе математического обеспечения и является одной из наиболее трудоемких и ответственных задач при создании САК.

В результате разработки ПО, входящего в САК, должна быть оформлена документация, необходимая для разработки, составления и сопровождения программ. Такие документы носят название программных документов. В их состав входят:

- спецификация — состав программ и документация на них;

- текст программ — запись программ на некотором языке программирования с необходимыми комментариями;
- описание программ — сведения о логической структуре и функционировании программ;
- программа и методика испытаний — требования, поэтапная проверка при испытании программ, а также порядок и методы их контроля;
- техническое задание — назначение и область применения программы, технические, технико-экономические и специальные требования, предъявляемые к программе, необходимые стадии и сроки разработки, виды испытаний;
- пояснительная записка — схема алгоритма, общее описание алгоритма и функционирования программ, а также обоснование принятых технических и технико-экономических решений;
- эксплуатационные документы — сведения для обеспечения функционирования и эксплуатации программы.

Техническое обеспечение (ТО). Это совокупность устройств, вычислительной и организационной техники, предназначенных для выполнения автоматизированного консультирования или консультационного процесса. Это ТО представляет собой физическую интеграцию всех компонентов САК в единое целое, обеспечивающее функционирование САК.

В техническое обеспечение САК входят универсальные ЭВМ, персональные ЭВМ, средства представления и переработки информации, средства дистанционной передачи данных и др..

Минимальный набор технических средств, позволяющий эффективно решать задачи консультирования при непосредственном участии человека, **называют базовыми конфигурациями САК.** Базовые конфигурации САК могут быть одно- и многоуровневые.

Одноуровневой базовой конфигурацией является автоматизированное рабочее место консультанта (АРМК).

Технические средства АРМК состоят из мини ЭВМ (компьютера) с расширенным составом периферийного оборудования. Последовательность действий пользователя АРМК обусловлена спецификой задач, решаемых в консультируемой области, набором директив и используемых прикладных программ, а также характером требуемой конечной документации (чертежи, таблицы, графики).

Многоуровневые конфигурации САК включают кроме одного или нескольких АРМК центральную универсальную ЭВМ средней или высокой производительности, связанную с АРМК средствами передачи данных. Обычно центральная ЭВМ используется для решения

сложных расчетных задач. Вместо нее могут использоваться многопроцессорные вычислительные комплексы или универсальная ЭВМ, соединенная с высокопроизводительными спецпроцессорами, например, логического моделирования.

Организационное обеспечение. Это обеспечение включает в себя положения, инструкции, приказы, штатные расписания и другие документы, регламентирующие организационную структуру подразделений и их взаимосвязь с комплексом средств автоматизированного консультирования.

Рассмотрим более подробно такие важные для консультанта, на наш взгляд, виды обеспечений: лингвистическое, информационное и техническое.

5.8. Лингвистическое обеспечение САК

5.8.1. Назначение, классификация языков консультирования и требование к ним

Лингвистическое обеспечение САК представляет собой совокупность языков консультирования, включая термины и определения, правила формализации естественного языка и методы сжатия и развертывания текстов, необходимых для выполнения автоматизированного консультирования, представленных в заданной форме.

Языки консультирования предназначены для представления и преобразования описаний в процессе автоматизированного консультирования. Основными объектами описаний в САК являются: проблемы консультирования, консультационное задание, консультационные процедуры, операции и процессы, консультационные рекомендации (начальные, промежуточные, конечные, типовые) и консультационные документы. Языки консультирования являются важнейшей составной частью САК и должны обладать как многими качествами универсальных языков программирования в представлении данных и действий над ними, так и достаточными уровнями выразительности, гибкости и проблемной ориентации в построении языковых конструкций, привычных консультанту.

Универсальные алгоритмические языки типа АЛГОЛ, ФОРТРАН, ПЛ/1, Паскаль, Ада, С достаточно эффективно используются для реализации САК, так как обладают развитыми возможностями для описания разнообразных алгоритмов, характерными для программного

обеспечения САК. Однако при их использовании в качестве языков консультирования программа, как правило, громоздка и неудобна. Это объясняется, во-первых, необходимостью специальной подготовки в области программирования и, во-вторых, сложностью самих процессов трансляции, генерации и отладки программ для управления последовательностью консультационных процедур и операций в соответствии с заданием на консультирование, которое составляется на этих языках.

Альтернативой этому подходу является создание специализированных проблемно-ориентированных языков консультирования и трансляторов к ним.

Языки консультирования будем классифицировать по следующим основным признакам: месту в процессе автоматизированного консультирования; связи с универсальными языками программирования; оперативности; преимущественному способу представления информации.

По месту в процессе автоматизированного консультирования будем различать языки *входные, внутренние, промежуточные, выходные, сопровождения* и *управления*.

Входные языки предназначены для задания исходной информации о консультируемой проблеме и целях консультирования и представляют собой совокупность *языков описания консультируемой проблемы* (ЯОКП) и *языков описания задания* (ЯОЗ).

Под описанием консультируемой проблемы понимают описание структуры консультируемой проблемы, ее свойств и характеристик, включая описание взаимодействия между частями консультируемой проблемы и ее взаимодействия с внешней средой, а также описание схемы функционирования консультируемой проблемы. Описание процесса как объекта консультирования включает также описание результата процесса и заданных характеристик его выполнения во времени и пространстве.

Для задач анализа и оптимизации с помощью ЯОКП описываются структура и исходные параметры консультируемой проблемы, а для задач структурного синтеза — консультационное задание и, возможно, исходный вариант консультируемой проблемы или ее аналога. Для описания процессов используются специальные классы процедурных языков — *языки моделирования.*

Язык ЯОЗ предназначен для идентификации заданий, описания их характеристик и указания последовательности выполнения консультационных процедур на ЭВМ.

Внутренние и промежуточные языки предназначены для представления информации на определенных стадиях ее обработки в ЭВМ. Появление этих языков объясняется выделением в САК некоторых подсистем (например, графического ввода, графического документирования, архива, чертежей и т. п.), инвариантных к классам консультируемых проблем, и необходимостью унификации представления входных или (и) выходных данных для этих подсистем.

Промежуточные языки позволяют легко включать инвариантные подсистемы в различные САК путем разработки специальных программ, называемых конверторами, которые выполняют преобразование данных из входных языков различных систем в единый унифицированный промежуточный язык определенной инвариантной подсистемы. Или, наоборот, промежуточный язык может быть преобразован во входной язык какой-либо другой специализированной подсистемы.

Примерами внутренних и промежуточных языков могут служить: язык представления графической и текстовой информации (ЯГТИ), языки графических метафайлов для хранения данных в архивах чертежей (графиков, диаграмм) и т. п.

Выходные языки консультирования предназначены для представления результатов выполнения консультационных процедур на ЭВМ, в том числе каких-либо рекомендаций, включая результаты консультирования в форме, удовлетворяющей их дальнейшее применение.

Языки сопровождения и управления служат для непосредственного общения пользователя с ЭВМ в процессе решения задач. Эти языки, как правило, включают средства для корректировки и редактирования входных данных и заданий на консультирование и поэтому содержат элементы входных и выходных языков, а также язык диагностических сообщений о допущенных ошибках.

По связи с универсальными языками программирования будем различать *автономные* и *расширяющие* языки. Автономные языки имеют собственные грамматики, соответствующий транслятор и могут применяться независимо от других языков программирования. Расширяющие языки строятся на основе грамматики другого языка и являются его проблемно-ориентированными дополнениями. Базой расширения чаще всего служат алгоритмические языки ПЛ/1, С и др. Такой подход позволяет использовать в языках консультирования все имеющиеся в базовом языке мощные средства обработки данных и упростить связь языков консультирования с другими программными

средствами системы, а также обеспечить в значительной степени независимость языков консультирования от типа используемой ЭВМ. К недостаткам расширяющих языков относится преимущественно их пакетный режим использования.

По оперативности языки будем разделять на *диалоговые* и *пассивные.* Диалоговые языки обеспечивают взаимодействие консультанта с ЭВМ на основе взаимного обмена сообщениями в реальном масштабе времени. Это позволяет оперативно получать все промежуточные результаты и управлять процессом консультирования на ЭВМ. Пассивные языки позволяют задавать входные данные и последовательность консультационных операций и процедур в виде некоторого символического описания с последующей трансляцией этих описаний и выполнением в режиме пакетной обработки заданий. По преимущественному способу представления информации будем выделять *алфавитно-цифровые, графические, голосовые* и *смешанные* языки консультирования.

В алфавитно-цифровых (символических) языках описания задаются в виде строк символов или в виде таблиц. В графических языках информация представляется в виде чертежей, графиков, схем, диаграмм и т. п. Для вывода информации в такой форме используют графопостроители, дисплеи, а для ввода — различные устройства кодирования графической информации, алфавитно-цифровую и функциональную клавиатуры и др. При решении задач многих консультируемых проблемах для консультантов привычной является графическая форма представления информации, поэтому графические языки являются наиболее эффективными для САК.

Разработка голосовых языков общения человека с ЭВМ основывается на использовании устройств распознавания и синтеза речи. Их применение в качестве языков управления совместно с другими способами представления информации является перспективным направлением развития диалоговых языков консультирования.

К языкам консультирования предъявляют следующие основные требования: эффективность, полноту, расширяемость, выразительность и проблемную ориентацию.

Эффективность языка подразумевает точность передачи заданий пользователя и лаконичность записей.

Полнота языка понимается как возможность описания любых проблемных ситуаций, на консультирование которых ориентирована САК, а также задания всех действий, имеющих отношение к цели консультирования.

Расширяемость алфавита и синтаксиса языка должна обеспечить возможность развития языка в соответствии с развитием предметной области САК.

Выразительность и проблемная ориентация должны обеспечить простоту изучения и использования языков консультантами-непрограммистами. С этой точки зрения языки консультирования должны быть близкими к естественным по своим грамматикам, что обеспечивает простоту и минимальные затраты времени на их изучение.

5.8.2. Представление языков с помощью формальных грамматик

Искусственные языки лингвистического обеспечения САК относятся ***к формальным языкам, которые определяются как множество цепочек в некотором конечном алфавите.***

Для задания описания формального языка необходимо указать его алфавит и формальную грамматику.

Алфавит представляет собой совокупность объектов, называемых символами (или буквами), каждый из которых можно воспроизвести в неограниченном количестве экземпляров.

Для языков консультирования, которые в большинстве своем близки к естественным, а слова, словосочетания и графические образы являются простейшими элементами, термин «алфавит», принятый в теории формальных языков, можно отождествить с термином «словарь», который употребляется в лингвистике.

Формальная грамматика представляет собой систему правил для описания множества конечных последовательностей символов. Конечные последовательности символов (цепочек), входящие в это множество, называют предложениями, а само множество — языком, который описывается этой формальной грамматикой.

В теории формальных языков правила формальных грамматик рассматриваются как продукции (правила вывода) — элементарные операции, которые, будучи применены в определенной последовательности к исходной цепочке (аксиоме), порождают лишь правильные цепочки. Сама ***последовательность правил, использованных в процессе порождения некоторой цепочки, является ее выводом.***

По способу задания правильных цепочек формальные грамматики разделяются на *порождающие* и *распознающие.*

К ***порождающим*** относятся грамматики, которые устанавливают правила построения любой правильной цепочки с указанием ее структуры так, что нельзя построить ни одной неправильной цепочки.

Распознающая грамматика позволяет установить, правильна ли произвольно выбранная цепочка, и если она правильна, выяснить ее строение.

Такое деление является несколько условным, так как любая распознающая грамматика по сути задает правила построения всех предложений.

Порождающей грамматикой или, кратко, грамматикой называется упорядоченная четверка

$$G = (V_T, V_H, \sigma_0, P),$$

где $V_T = \{a_1, a_2, ... a_n\}$ —основной терминальный алфавит; V_H — конечный вспомогательный (нетерминальный) алфавит; $\sigma_0 \in V_H$ — начальный (нетерминальный) символ или аксиома; $P = \{u_i \rightarrow v_i / i = 1, 2 ... k\}$ — конечная система подстановок (продукций), левые и правые части которых есть цепочки $u_i v_i$, содержащие символы основного V_T и вспомогательного V_H алфавитов, т. е. $u_i, v_i \in F(V)$, где $F(V)$ — свободная полугруппа над объединенным алфавитом $V = (V_T \bigcup V_H)$.

Символы основного алфавита V_T являются элементарными единицами определяемого языка, а символы алфавита V_H — метапеременными, которые используются при выводе правильных цепочек. В естественных языках ***метапеременным*** соответствуют грамматические классы: ***существительное, глагол*** и т. п. Начальный символ $\sigma_0 \in V_H$ — метапеременная аксиома, из которой выводятся все правильные цепочки, например, в естественных языках такой ***аксиоме*** соответствует грамматический класс «***предложение***». Множество P — грамматические правила определяемого языка. Язык, порождаемый грамматикой G, обозначают через $L(G)$.

Задача описания языка состоит в определении множеств, составляющих его грамматику.

Множество терминальных символов V_T в наибольшей степени отражает проблемную ориентацию языка консультирования. Рациональное определение множества V_T путем включения в него привычных для консультанта понятий позволяет создать проблемно-ориентированную языковую среду, естественную для него как для специалиста.

В качестве начального нетерминального символа σ_0 во входных языках консультирования чаще всего выбирают понятие «***директива***». Остальные нетерминальные символы определяют лексические классы (лексические переменные) входного языка. Например, для диалоговых языков можно определить следующее подмножество нетерминальных символов: ДМ — директива, Д — действие, ОД — объект действия, СД

— способ действия, УД — условия действия, ЧД — число действий; УВВ — устройство ввода; УВН — устройство вывода и т. д.

Решающее влияние на свойства языка, сложность порождения и распознавания цепочек оказывают правила подстановок множества *P*. В зависимости от правил подстановок, различают следующие четыре основных класса грамматик.

1. Грамматики непосредственных составляющих или контекстные грамматики (НС-грамматики) с продукциями вида

$$u_1 \psi u_2 \rightarrow u_1 z u_2,$$

где u_1 и u_2 — произвольные (возможно, пустые) цепочки над $V_T \cup V_H$; ψ — нетерминальный сивол, z—непустая цепочка над $V_T \cup V_H$. Цепочки u_1 и u_2 называют соответственно левым и правым контекстами данной продукции.

Каждое правило вывода НС-грамматики указывает подстановку некоторой цепочки вместо нетерминального символа. Язык, порождаемый НС-грамматикой, называется НС-языком или контекстным языком.

2. Контекстно-свободные или бесконтекстные грамматики (КС-грамматики) имеют продукции вида $\psi \rightarrow z$, где ψ— нетерминальный символ, а z — произвольная непустая цепочка над $V_T \cup V_H$.

Языки, порождаемые КС-грамматиками, называют контекстно-свободными или КС-языками. КС-грамматики играют главную роль при формальном описании и анализе языков программирования. Это объясняется тем, что, во-первых, средствами КС-грамматик удается достаточно полно описать синтаксическую структуру языков программирования, в том числе и языков консультирования, и, во-вторых, достаточно хорошо проработаны алгоритмы распознавания КС-языков, которые составляют основу блоков синтаксического анализа трансляторов этих языков.

3. Линейные грамматики (ЛН-грамматики) имеют продукции вида $\psi \rightarrow z$, где $\psi \in V_H$, а $z = F(V_T)$ либо $z = u\tau v$, где $z, u, v \in F(V_T)$, $\tau \in V_H$. Иными словами правые части продукции ЛН-грамматик являются терминальными цепочками либо содержат лишь один терминальный символ.

Продукция $\psi \rightarrow z$ — называется праволинейной (леволинейной), если $z \in (V_T)$ либо $z = u\tau$, $(z = \tau u)$, т. е. если единственный нетерминал τ входит в правые части линейных продукций, то он всегда крайний справа (слева). Соответственно грамматики называются праволинейной, леволинейной, а языки праволинейным, леволинейным.

4. Автоматные грамматики (А-грамматики) имеют продукции вида $\psi \to b\tau$, где $\psi, \tau \in V_H$, $b \in V_T$ и порождают автоматные языки (А-языки).

Алгоритмы распознавания линейных и автоматных языков достаточно просты. Линейные и автоматные грамматики используются в основном для описания простых языков и для предварительного преобразования программ с целью распознавания некоторых составных конструкций более сложных языков и представления их в форме, более удобной для анализа. К таким конструкциям относятся идентификаторы и выражаемые через них понятия, все типы числовых констант, бесскобочные выражения.

Некоторые языки консультирования можно полностью описать А-грамматикой G^a — (V_T, V_H, a, P); может быть ассоциирован конечный автомат A = (V_T, S, σ, φ, Г), где V_T — входной алфавит автомата A, совпадающий с терминальным алфавитом грамматики G^a;
$S = V_H \bigcup \{\varphi\}$ ($\varphi \in V_H$) — множество состояний автомата; σ и φ — соответственно начальное и конечные состояния; Г — граф переходов, определяющий функционирование автомата (рис. 5.33).

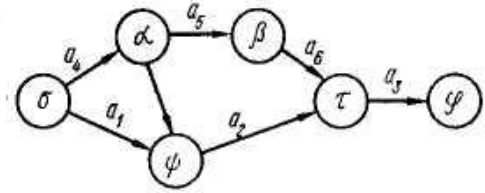


Рис. 5.33. Граф функционирования конечного автомата

Каждому нетерминалу $\psi \in V_H$ грамматики G^a соответствует вершина графа Г, помеченная символом ψ. В графе есть еще одна вершина, помеченная символом заключительного состояния φ. Единственному начальному состоянию σ автомата A соответствует вершина, помеченная аксиомой σ. Каждой продукции $\psi \to a\tau$ соответствует ребро на графе из вершины ψ в вершину τ, помеченное терминальным символом $a \in V_T$. При подаче на вход такого автомата цепочки терминальных символов осуществляется смена состояний в соответствии с движением по ребрам графа от вершины σ до φ. Причем в каждом состоянии автомат воспринимает очередной терминальный символ и переход осуществляется по ребру, помеченному этим символом.

К отдельному виду формальных грамматик порождающего типа, которые описывают класс контекстно-свободных языков, относятся

Бекуса нормальные формы (БНФ) или металингвистические формулы. Такие формулы или такие языки предназначены для описания свойств других языков и называются метаязыками. БНФ предложил американский математик Бекус для синтаксиса языка АЛГОЛ-60. Они широко применяются для формального описания различных КС-языков. Основными классами объектов, которые используются в БНФ, являются основные символы и имена конструкций описываемого языка в виде металингвистических переменных. Значения металингвистических переменных — это цепочки основных символов описываемого языка.

Каждая металингвистическая формула описывает правила построения некоторой конструкции языка и состоит из двух частей. В левой части находится металингвистическая переменная, которая обозначает соответствующую конструкцию. Затем следует так называемая металингвистическая связка ::=, имеющая смысл глагола «быть». Она соединяет левую и правую части формулы. В правой части формулы указывается один или несколько вариантов построения конструкции, определенной в левой части. Варианты разделяются металингвистической связкой | (вертикальная черта), имеющей значение «или». Металингвистические переменные обозначаются словами, которые заключаются в угловые скобки < >. Эти слова поясняют смысл описываемой конструкции.

Для того чтобы построить определяемую формулой конструкцию, нужно выбрать некоторый вариант построения из правой части формулы и, используя соответствующие формулы, подставить вместо каждой металингвистической переменной некоторые цепочки основных символов. Особенностью металингвистических формул является наличие в них рекурсий, т. е. использование для описания некоторых конструкций самих описываемых конструкций. Наличие рекурсий несколько затрудняет чтение и усвоение металингвистических формул, однако рекурсии небходимы для того, чтобы язык, описываемый формулами, был бесконечным, т. е. включал в себя бесконечное число цепочек основных символов.

Примером БНФ может служить задание синтаксиса целого числа в описании языка:

<целое> :: = <целое без знака> | — <целое без знака> | + <целое без знака>
<целое без знака> :: = <цифра> | <целое без знака> <цифра>
<цифра> :: = 0|1|2|3|4|5|6|7|8|9

Здесь <целое>, <целое без знака>, <цифра> — суть метапеременные, а цифры от 0 до 9 — основные терминальные символы языка.

Определение целого числа без знака является рекурсивным, так как любая цифра — это целое без знака, или любое целое без знака, к которому справа приписана цифра, также есть целое без знака.

Для описания синтаксиса формальных языков на практике нашли применение также графические представления языковых конструкций, так называемые синтаксические диаграммы. На рис. 5.34 представлен один из вариантов синтаксической диаграммы целого числа.

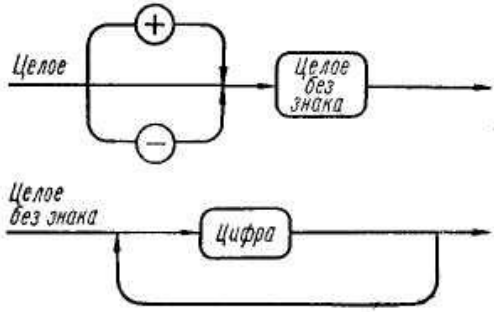


Рис. 5.34. Вариант синтаксической диаграммы для распознавания описания целого числа

5.8.3. Входные языки

Входные графические языки (ВГ-языки) находят широкое распространение в САК, так как часто исходная информация о консультируемой проблеме имеет графическую форму представления в виде чертежей, эскизов, функциональных зависимостей его характеристик и т. п. ВГ-языки составляют основу лингвистического обеспечения в подсистемах геометрического моделирования и машинной графики.

Для описания геометрии объектов консультируемой проблемы на графических языках используются следующие четыре способа: координатный, структурно-символический, аналитический и рецепторный.

При *координатном способе* задаются координаты всех характерных точек объекта и их связность, например вершины граней тела или вершины ломаных линий и т. п.

Структурно-символический способ предусматривает предварительное выделение и описание типовых или базовых графических элементов (БГЭ), создание библиотеки БГЭ и дальнейшее ее использование для составления из базовых элементов форм и чертежей объекта консультируемой проблемы.

Аналитический способ основан на описании графического изображения в виде математических соотношений, например уравнений поверхностей и линий.

Рецепторный способ предусматривает использование мозаичного представления изображения, например, в виде матрицы, элементы которой — булевы величины (их единичные значения соответствуют темным, а нулевые — светлым частям изображения).

Операторы символических ВГ-языков задаются в виде текстовых строк или вводятся в ЭВМ с терминалов с помощью алфавитно-цифровой и (или) функциональной клавиатуры, устройств управления маркером или устройства указания и считывания координат на планшете ввода (кодографе).

Терминология ВГ-языков должна быть близка к обычной терминологии, чтобы облегчить процесс освоения языка и ввода графических данных непосредственно специалистами прикладной области без посредника-специалиста в области программирования. ВГ-языки могут быть ориентированы на описание объекта и на ввод изображения. Особенность ВГ-языков первого типа состоит в том, что в результате трансляции описания в ЭВМ формируется модель геометрии объекта в трехмерном пространстве, которая может быть представлена на устройствах отображения в виде изображения произвольных проекций, сечений, разрезов. Результатом трансляции описания на языках второго типа является лишь то изображение, которое введено. Эти языки могут использоваться в САК для ввода типовых графических элементов чертежей (ТЭЧ). ВГ-языки для описания изображений основаны на использовании некоторых общих подмножеств команд, которые обеспечивают: построение графических примитивов; задание атрибутов графических примитивов; построение графических изображений произвольной конфигурации; построение изображения из ограниченного множества элементов, имеющих типовую конфигурацию; сокращение избыточности описания на основе использования принципа умолчания и признаков повторения; преобразование изображения (афинные и другие преобразования); документирование информации в графическом, текстовом виде или запись на машинные носители; прием и передачу информации; управление устройствами вывода. Подмножества этих команд могут быть расширены или сокращены в зависимости от области и условий использования конкретного языка.

Для описания изображений в САК используются символические входные языки (СВГ-языков). Вводимая графическая информация представляется на СВГ-языках в виде последовательности описаний

характерных точек и формирования графических примитивов, которые являются специальными типами переменных в программах на этих языках.

К графическим переменным относятся точка, линия, окружность, строка текста, прямоугольник и т п. Обычно СВГ-языки допускают полную и сокращенную форму идентификации этих переменных, например, для перечисленных типов переменных могут использоваться соответственно следующие ключевые символы: POINT или P, LINE или L, CIRCLE или C, TEXT или T, BOX или B и т. д. Идентификатор графической переменной кроме ключевых символов, задающих ее тип, может содержать последовательность цифр, задающую номер переменной в пределах своего типа, например, POINT2 или P2, LINE5 или L5 и т. п. Идентификаторы используются для ссылки на эти переменные при описании следующих за ними графических переменных или геометрических построений.

Операторы задания графических переменных на таком языке имеют следующую общую синтаксическую структуру:

<графическая переменная> :: = <идентификатор графической переменной> <разделитель оператора> <последовательность операндов> <признак конца оператора>

<идентификатор графической переменной> :: = <ключевое слово типа графической переменной> | <ключевое слово типа графической переменной> <целое без знака>

<разделитель оператора> :: = <=>

<признак конца оператора> ::= <;>

<последовательность операндов>::= <операнд> | <последовательность операндов> <разделитель операндов> <операнд>

<разделитель операндов> :: = <,>

<операнд> :: = <число> | <идентификатор графической переменной> | <строка текста> | <простое арифметическое выражение> | <идентификатор>

<простое арифметическое выражение> :: = <операнд> <знак арифметической операции> <операнд>.

В описании графической переменной *точка* используются различные способы задания координат, например:

а) с использованием числовых констант:

*P*1 =30.5, 40.28;

б) с использованием идентификаторов переменных, содержащих значения координат:

P3= *X*5, *Y*6;

в) с использованием простых арифметических выражений над переменными:

P4 = *X*5+0.7, *Y*2/3;

г) в приращениях к ранее определенной точке (в примере к точке P3):

P = P3, 0.6, —20.5;

P = P3, *DX*1, *DY*1;

д) в координатах узла некоторой ранее описанной размерной сетки с указанием номеров сетки в квадратных скобках и идентификатора сетки, если сетка не единственная:

P3 = [2,5];

P4 = *GR*2*:* [3,4]; и т. п.

Возможны различные комбинации из приведенных вариантов:

P=20.7, *Y*6;

P3= *X*5 + 0.7, 80.5;

P4 = P3,4, *Y*2; и т. д.

Если точка является операндом, то вместо ее идентификатора может быть записана правая часть описания, заключенная в скобки, например, последовательность

*P*1 = *X, Y*;

P2 = P1, 20.15;

равнозначна операнду

P2 = (*X, Y*)20.15;

Размерные сетки представляют собой прямоугольные сетки с осями параллельными координатным осям (рис. 5.35, *а, б*).

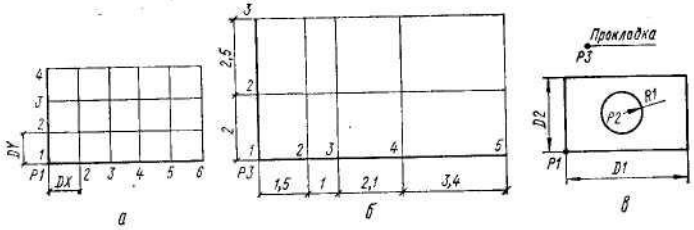


Рис. 5.35. К объяснению использования графического языка:
а — сетка регулярная; *б* — сетка нерегулярная; в — чертеж прокладки

Они могут быть регулярными, т. е. с равномерной дискретизацией по осям *OX* и *OY,* и нерегулярными с неравномерной дискретизацией. Для идентификации регулярных и нерегулярных сеток используются соответственно ключевые символы *GR* и *GN*. Как и графические переменные сетки могут нумероваться, например: *GR*4, *GN*3.

В описании регулярной сетки задаются в порядке перечисления следующие операнды: точка левого нижнего угла сетки; значения шагов вдоль координатных осей *OX* и *OY;* количество шагов по осям *OX* и *OY*. Для сетки (рис. 5.35, *а*) оператор имеет следующий вид:

*GR*1 = P1, *DX*, *DY*, 5,3;

Для описания нерегулярной сетки необходимо предварительно задать списки значений шагов по осям *OX* и *OY* с помощью операторов *SGX* и *SGY*.

Пример:

*SGX*2= 1.5, 1, 2.1, 3.4;
*SGY*2 = 2, 2.5;
*GN*2 = *P*3, *SGX*2, *SGY*2, 4,2;

Первые два оператора задают размеры шагов по осям *OX* и *OY*, а третий — нерегулярную сетку с номером 2 и четырьмя разбиениями по оси *OX* и двумя по оси *OY* (рис. 5.35, *б).*

Операндами графической переменной *линия* являются две точки. Точки могут задаваться всеми допустимыми для их задания способами. Пример:

*L*1 = P1, P2;
L= (10, 20), (45, 20);
L = P2, (40, 55);
*L*3 = ([4,6]), ([15,3]);

Окружность на графическом языке может быть задана двумя способами: *C*-0, <точка центра>, <точка на окружности>; и *C*-1, <точка центра>, <радиус>;

Пример:

*C*1 = Ø, *P8, P*4;
*C*2= 1, *P*5, 10;
*C*3= 1, *P*5, *R*2;

Описание графической переменной *прямоугольник* включает: точку, задающую левый нижний угол фигуры; размеры сторон вдоль осей *OX* и *OY,* которые могут быть определены графическими переменными типа линия, константами или простыми переменными; значение угла наклона к оси *OX* горизонтальной стороны фигуры (угол отсчитывается в положительном направлении и задается в градусах).

Пример:

B=*P*1, *L*3, *L* 4, ANG: 45;
*B*3 = *P*2, *DX, DY,* ANG: 30;

ANG: ключевое слово для идентификации значения угла поворота. Для задания *строки текста* указываются: точка начала строки; символы

строки, заключенные в апострофы, или идентификатор текстовой переменной; угол наклона строки к оси *OX*.
Пример:

T = *P*2, 'ДЕТАЛЬ' ANG: 0;

В ТЭЧ обычно выделяется базовая (опорная) точка, относительно которой будет определяться вектор сдвига для переноса всех координат элемента при размещении его на конкретном чертеже в результате выполнения программы вызова ТЭЧ из библиотеки. *Базовая точка* задается с помощью оператора

BASE= <координаты точки>;

Программа на графическом языке начинается оператором

NAME =<имя программы>, <тип ТЭЧ>;

Имя программы задается идентификатором, который затем используется для поиска элемента в библиотеке:

<тип ТЭЧ> :: = 0 | 1 | 2 | 3, что соответствует заданию ТЭЧ следующих типов: ТЭЧ-К — графические константы, вид которых не зависит от масштаба чертежа; ТЭЧ-М — масштабируемые графические константы; ТЭЧ-ПК — параметрические элементы, характеристики которых зависят от входных параметров, но не зависят от масштаба формируемого чертежа; ТЭЧ-ПМ — параметрические элементы, которые при выводе приводятся к масштабу формируемого чертежа. Для описания входных параметров в графическом языке имеется соответствующий оператор

PAR = <список параметров>;

Программа для формирования чертежа прокладки (рис. 5.35, *б*) на графическом языке имеет следующий вид:

```
NAME = PROKL, 3;
PAR = D1, D2, R1;
BASE = 0,0;
P1 = 0,0;
B = P1, Dl, D2;
P2 = P1, Dl/2, D2/2;
C= 1, P2, R1;
P3 = P2, — 10, D2/2 + 20;
T = P3, 'ПРОКЛАДКА', ANG: 0;
 L = P3, (P3, 20,0);
 FIN;
```

СВГ-языки бывают пассивными и диалоговыми. Описание графических данных на пассивных СВГ-языках переносится на носители с помощью традиционных устройств подготовки данных, а затем вводится и обрабатывается в ЭВМ в пакетном режиме. Для внесения

изменений в описание может потребоваться повторный ввод данных о всем изображении. Описание изображений с использованием диалоговых СВГ-языков производится, как правило, за пультом дисплея с возможным выводом промежуточных результатов на одно из пассивных графических устройств. Это позволяет оперативно вносить изменения и корректировать ошибки с дисплея.

Общим недостатком СВГ-языков является необходимость достаточно глубокого и полного знания синтаксиса и семантики языка пользователями САК, а также часто точного предварительного определения координат всех характерных точек графического изображения. Эти недостатки практически отсутствуют в графосимволических языках (ГСВГ-языках).

ГСВГ-языки, как правило, диалоговые и основаны на использовании средств оперативного ввода графической информации для задания графических команд и их параметров. Характерными особенностями этих языков является возможность оперативного контроля и корректировки вводимого изображения, а также простота формирования команд описания изображения. В качестве средств ввода используются функциональная и алфавитно-цифровая клавиатуры, одно из устройств ввода координат экрана дисплея (курсор, клавиши изменения направления движения маркера) или планшет ввода с указателем координат.

ВГ-язык представляет собой язык графического диалога консультанта с прикладной программой, которая ориентирована на построение графического изображения. Мнемоника команд построения графических элементов (точка, прямая, дуга, кривая, ломаная и т. п.), способов задания координат, а также команд редактирования и преобразования изображения (стереть, вставить, сдвинуть, повернуть и т. п.) может представляться на экране в виде функциональных кнопок, которые указываются пользователем с помощью курсора при формировании команд. В системах с функциональной клавиатурой каждой команде ставится в соответствие определенная клавиша. Менее удачной считается реализация ввода кода команды с помощью алфавитно-цифровой клавиатуры. Основной недостаток реализации входного языка с использованием дисплея состоит в трудности точного задания координат элементов создаваемого изображения.

Применение планшетов ввода в качестве устройств задания координат позволяет значительно облегчить этот процесс. Координаты точек для графических построений в этом случае могут задаваться непосредственно с эскиза чертежа. Поле ввода планшета может быть разбито на ряд зон: рабочую и несколько командных. В рабочей зоне

помещается эскиз вводимого изображения, а в командных зонах размещаются так называемые программно-интерпретируемые таблицы мнемонических наименований команд, операций и других понятий, на основе которых строится язык графического ввода. При соответствующей организации программного обеспечения ввода по координатам точек командных зон сравнительно просто могут быть определены коды этих команд и понятий. Последовательность таких кодов поступает на вход интерпретатора с языка графического ввода, который посредством вызова требуемых программ выполняет графические команды и (или) действия, соответствующие введенным командам.

Данный подход обеспечивает возможность смены или расширения графических языков в зависимости от класса решаемых задач или кодируемых чертежей при сохранении общей методики ввода операторов языка пользователем-консультантом. Смена графических языков обеспечивается заменой мнемоники программно-интерпретируемых таблиц и настройкой интерпретатора на программы, которые выполняют обработку действий, задаваемых этими таблицами.

Рассмотрим типичный ВГ-язык, ориентированный на описание объекта. Такой язык включает в себя операторы трех типов: служебные, операторы описания, операторы композиции.

К *служебным* операторам относятся операторы задания границ программы, масштаба описания объекта:

<оператор начала программы> :: = НАЧАЛО <имя программы>.

<оператор конца программы>::= КОНЕЦ <имя программы>.

<оператор задания системы координат >:: = СИСТЕМА XYZ.

<оператор размерности> :: = РАЗМЕРНОСТЬ <единицы>.

<оператор масштаба> :: = МАСШТАБ <отношение>.

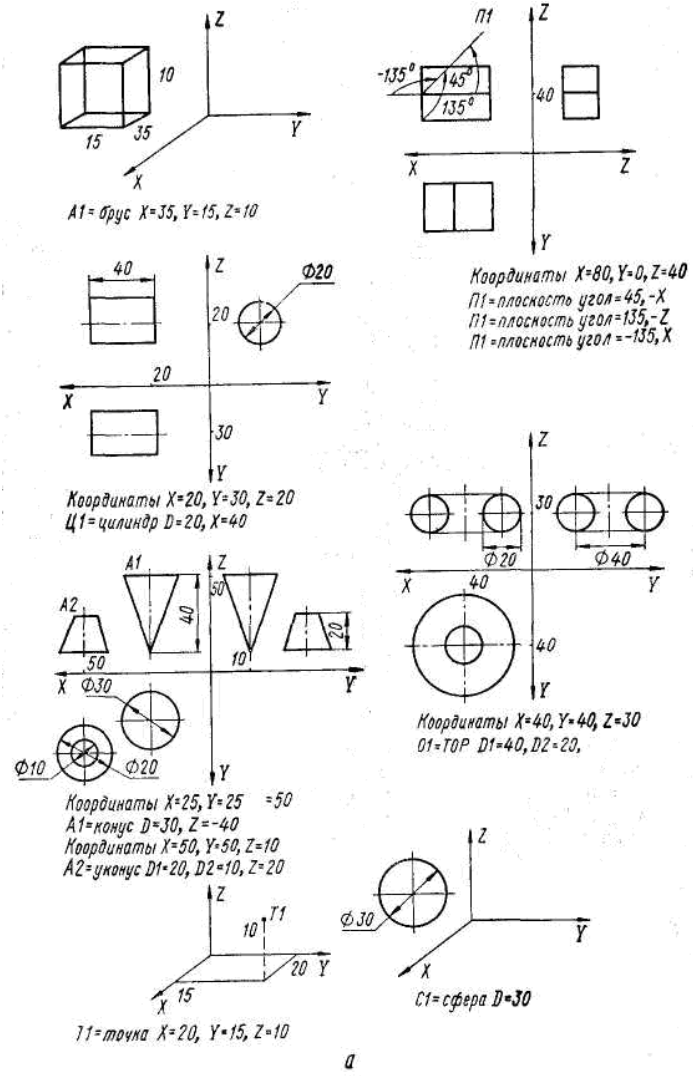
<единцы> :: = ММ | СМ

<масштаб> :: = 1 : 1 | 1 : 2 | 1 : 10 | 2 : 1 | 5 : 1 | 10: 1.

Для задания базовых элементов форм (БЭФ), из которых формируется объект, используются соответствующие *операторы описания* для следующих трехмерных БЭФ: прямоугольный параллелепипед (БРУС); прямой круговой цилиндр (ЦИЛИНДР); прямой круговой конус (КОНУС); плоскость, параллельная одной из координатных плоскостей или оси координат (ПЛОСКОСТЬ); тор (ТОР); сфера (СФЕРА); точка (ТОЧКА); прямая призма с треугольным (ПРИЗМА) или криволинейным основанием (ЦИКЛИН и СКРКЛИН), усеченный конус (УКОН):

<оператор описания БЭФ>::=<идентификатор БЭФ>=
<наименованияБЭФ><параметры БЭФ>

Примеры задания операторов БЭФ приведены на рис. 5.36.



а

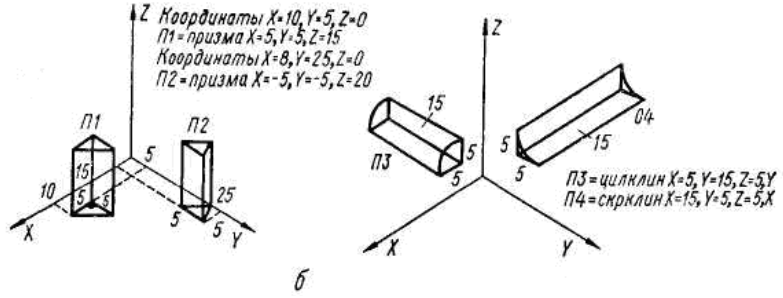


Рис. 5.36. Примеры задания базовых элементов форм *(а)* и треугольных и криволинейных призм *(б)* на графическом языке

Операторы композиции графического языка позволяют задавать следующие операции над БЭФ: сложения, вычитания и отсечения:

<оператор сложения> :: = <идентификатор БЭФ> =
<идентификатор БЭФ> + <знак + или —>
<идентификатор БЭФ>
<оператор отсечения> :: = <идентификатор БЭФ>/
<идентификатор секущей плоскости>, <знак> <ось>

На рис. 5.37 показано применение операторов сложения, вычитания, отсечения БЭФ на графическом языке для формирования детали. Оператор отсечения позволяет отсечь от композиции БЭФ часть по направлению указываемой оси.

На рис. 5.37 оператором *D*3 = *D*2/П1, + *X* задается отсечение части тела *D*2, расположенной в сторону увеличения координаты по *X*, считая от секущей плоскости П1. Для графического языка реализованы различные способы ввода и трансляции описаний, включая диалоговый графический режим задания.

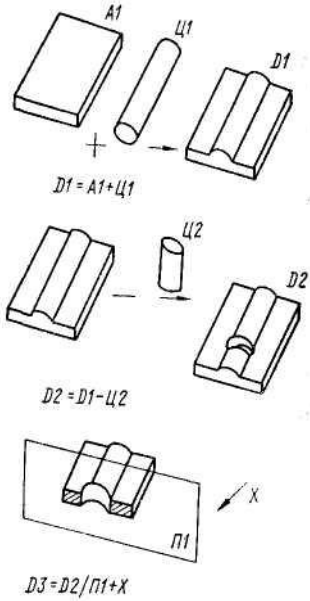


Рис. 5.37. К объяснению использования операторов композиции БЭФ

Языки описания схем и моделирования. Схемные языки предназначены для ввода данных о консультируемой проблеме, представленной в виде структурных, функциональных или принципиальных схем, которые отображают множество элементов и связи между ними с точки зрения функционирования проблемы. Применение схем характерно для радиоэлектроники, вычислительной техники, автоматики, гидравлики, а также для прочностных расчетов конструкций, механизмов и т. п.

Описание схем с помощью схемных языков состоит из совокупности предложений, каждое из которых содержит сведения об одном элементе схемы и его связях. Сведения об элементе включают его тип, имя (номер) и числовые значения параметров. Связи задаются номерами узлов, к которым подсоединяются внешние входы (выходы) элементов, а также типами связей, например, в расчетных схемах конструкций.

Различают форматные и бесформатные схемные языки. На *форматных схемных языках* описание чаще всего представляется в виде таблиц или входных документов. Каждая строка таблицы представляет собой предложение на языке описания, а колонки соответствуют определенной части предложения (тип элемента,

номера узлов связи, параметры и т. п.). Табличная форма описания удобна для ввода схем с однородными элементами по числу связей и количеству параметров. Правила заполнений таблиц просты и не требуют специальных знаний в области программирования.

В *бесформатных схемных языках* части предложений и сами предложения отделяются друг от друга специальными разделителями типа запятая, точка с запятой, наклонная черта и т. п. Бесформатные языки более удобны для описания схем с разнохарактерными элементами. На практике также нашли применение и смешанные формы схемных языков, в которых используются элементы форматирования в виде полей записей с фиксированным и переменным числом позиций, а также описание последовательности параметров с применением разделителей. Для полей с переменным числом позиций допускается использование специальных признаков строк продолжения.

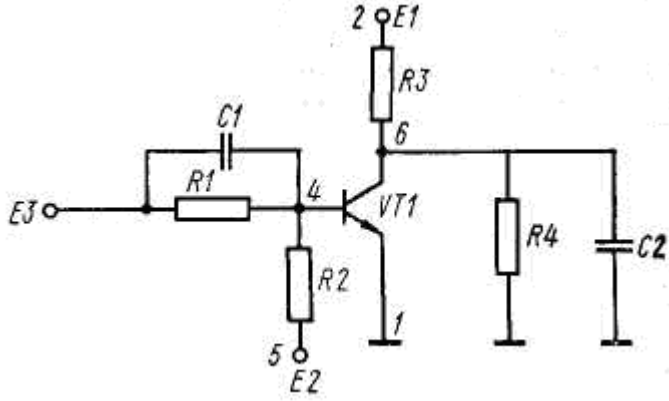


Рис. 5.38. Схема для примера применения языка

На рис. 5.38 приведен пример принципиальной электрической схемы, описание которой на языке описания схем имеет следующий вид:

```
R1, 3 — 4 = 2.7
R2, 4 — 5 = 5.1
R3, 2 — 6= 1.3
C1, 3 — 4 = 0.068
C2, 6— 1 =0.01
R4  6 — 1 = 3.
Ql = 2T312 (4 — 6— I)
E1,2— 1=6.3
E2,  1—5=5.
E3, 3—1 = ТАБ 1 (TIME).
```

Структура описания ясна из сопоставления с рисунком схемы. Каждое предложение содержит идентификатор типа элемента и его номер, номера узлов связи и номинал в единицах по умолчанию (килоом, пикофарада, вольт). Описание транзистора VT1 имеет идентификатор Q1, номера узлов для выводов транзистора (4—6 — 1) даны в заранее оговоренной последовательности (база — коллектор — эмиттер). El, E2, E3 описывают источники напряжения. Тип функциональной зависимости источника переменного напряжения E3 от времени задается в таблице ТАБ 1, описание которой здесь не приведено.

Для описания и логического моделирования схем используются различные входные языки. С помощью этих языков сведения об объекте задаются в виде описания схем и протекающих в них процессах. Перечисленные языки используются при формировании рекомендаций по реализации процесса проектирования логических и функциональных схем ЭВМ. Языки моделирования GPSS, СИМСКРИПТ, НЕДИС и другие используются для описания информационных процессов в АСУ, вычислительных системах и т. п. Чаще всего их модели представляются в виде систем массового обслуживания.

Входные языки автоматизированных информационных систем (АИС) являются составной частью лингвистического обеспечения САК. Они служат для составления предписаний на ввод, обновление, поиск, обобщение, редактирование и выдачу информации с баз нормативных данных, автоматизированных архивов и т. п. Входные языки АИС имеют в своем составе средства, позволяющие задавать следующие предписания: идентификационные признаки пользователей (авторов предписаний); признаки массивов информации, к которым производится обращение; операции над массивами; исходные данные для операций, т. е. вводимые или запрашиваемые сведения; имена программ или последовательности программ, реализующих операции над массивами; адреса, по которым следует направлять результаты выполнения операций; редакционные признаки выдаваемой информации, т. е. форматы таблиц и т. п. Одним из основных дополнительных требований САК к этим языкам и к АИС в целом является наличие интерпроцессорного (автоматического) режима работы, т. е. режима, при котором источником запросов и потребителем информации является не только человек, но и некоторая подсистема САК. Это может быть обеспечено на основе выделения в АИС некоторого промежуточного языка, удобного для обмена данными между АИС и программами САК в автоматическом режиме.

5.8.4. Диалоговые языки

Диалоговые языки служат средством оперативного взаимодействия консультанта с ЭВМ, при котором происходит чередование запросов и ответов между человеком и ЭВМ в реальном масштабе времени Диалог человека с ЭВМ в САК рассматривается как метод решения задачи при котором человек знает и ставит задачу консультирования, а ЭВМ используется для ее решения. В процессе диалога с ЭВМ создается ***цифровая модель консультируемой проблемы*** (ЦМКП), выполняются расчет и анализ характеристик консультируемой проблемы, формируются необходимые для решения консультационных задач рекомендации, создается необходимая консультационная документация. Одним из основных требований к диалоговым языкам является близость к естественным для человека.

Многие операции по формулированию и корректировке ЦМКП могут быть представлены в виде различных комбинаций следующих четырех действий: выбора элемента, ввода нового элемента, удаления элемента, изменения характеристик элемента или его связей. При формировании задания автоматическим консультационным процедурам необходимо указывать: вид консультационной процедуры, входные данные или их место во внешней памяти, тип, форму и направление выдачи результатов.

По способам ввода команд различают более десяти типов представления языка диалога, среди которых наибольшее распространение получили языки типа «запрос — ответ» на основе: директив пользователя; выбора альтернативных возможностей; заполнения пользователем форматов, представляемых машиной на экране дисплея.

В *директивных* языках основным форматом представления операторов является текстовая строка, а основным устройством ввода — алфавитно-цифровая клавиатура. Недостатком языков с алфавитно-цифровым вводом является то, что пользователь должен помнить все элементы словаря данного языка и правила формирования предложений. Иногда объем такой информации бывает достаточно велик. Кроме того, достаточно сложным является этап лексического и синтаксического анализа в процессе трансляции предложения.

В языках, *основанных на процедуре выбора альтернативных возможностей,* конструирование предложения происходит путем указания предоставляемых на экране дисплея элементов словаря или выполнения действий с устройствами ввода. Предпосылкой использования альтернативных языков в ДГС автоматизированного консультирования является то, что основная информация, представляемая пользователю, имеет контрольно-справочный характер

о возможностях системы и правилах работы с ней, о состоянии цифровой модели проблемы и базы данных, а также следующие преимущества по сравнению с директивными языками:

- достигается более высокое быстродействие ввода, поскольку пользователь не должен полностью печатать слово или фразу и не должен помещать выбранный элемент данных на определенную позицию формата;
- от пользователя не требуется знания форматов и символики множества директив, а достаточно знать лишь простые правила работы с устройствами интерактивного ввода для выбора нужной команды из набора предоставляемых;
- существенно упрощается этап лексического и синтаксического анализа в процессе разбора предложения, так как элементы словаря вводятся в ЭВМ в заведомо правильном виде.

Процедура трансляции предложений языка существенно упрощается, если синтаксический разбор начинается не по окончании ввода всего предложения, а по мере ввода его отдельных членов. В таком случае пользователю предоставляется не весь словарь, а только та его часть, из которой выбирается очередной член предложения.

Языки с такой организацией ввода получили название *диалоговых языков со сменными наборами команд* (СНК-языки). Построение альтернативных СНК-языков дает дополнительные преимущества, так как позволяет организовать оперативное эхо-отображение процесса разбора предложения. Представление словаря на экране дисплея называют ***меню***, а его элементы —***функциональными кнопками***. В принципе, весь ввод для этих языков можно организовать на основе использования устройства выбора альтернатив и соответствующих меню, однако некоторые элементы словаря удобнее вводить с помощью других устройств ввода символов, координат и чисел. При этом возможно такое построение языка и системы разрешения прерываний от устройств ввода, которая обеспечивает соответствие каждому терминальному символу языка лишь одного обобщенного действия пользователя (диалогового прерывания, связанного с выбором альтернативы, вводом строки текста или координат позиции и т. п.). Диалоговые языки, основанные на использовании графических изображений и устройств ввода графических данных, называют *диалоговыми графическими языками* (ДГ-языками).

В ДГ-языках выделяют языки изображений и действий. Язык изображений предназначен для вывода из ЭВМ графического представления консультируемой проблемы, а также сведений о состоянии обрабатываемых данных, вычислительного процесса и о вариантах

действий, которые может предпринять пользователь. Язык действий — это язык, на котором пользователь вводит в ЭВМ свои ответы и задания.

Выбор типа ДГ-языка оказывает существенное влияние на структуру диалога. Применение ДГ-языков альтернативного типа позволяет реализовать только первый и второй из следующих трех наиболее типичных режимов диалога:

- «инициатор — ЭВМ» с жесткой структурой последовательности сообщений;
- «инициатор — ЭВМ» с гибкой структурой последовательности сообщений»;
- «инициатор — человек» (при этом пользователь взаимодействует с ЭВМ на проблемно-ориентированном языке директивного типа). Первый и второй режимы диалога обеспечивают минимальное время решения для широкого класса задач автоматизированного консультирования.

Для учета психологических факторов и улучшения процесса взаимодействия человека с ЭВМ в диалоговых графических системах (ДГС) используют ряд приемов в организации языка изображений и сервисных программных средств. Поле экрана дисплея обычно разделяют на ряд областей по функциональному назначению: главную (рабочую), в которой воспроизводится собственно графическое представление консультируемой проблемы; процессов, предназначенную для отображения ключевых слов команд пользователя, допустимых в данном состоянии системы; графических данных — для отображения стандартных или построенных ранее графических объектов, используемых в качестве элементарных для построения сложных изображений; сопровождения диалога, в которой выводятся системные указания пользователю, вопросы системы и диагностические сообщения; контроля данных — для вывода эхо-отображения при вводе данных с клавиатуры.

Сервисные средства ДГС обеспечивают консультанту определенный уровень «комфорта» в процессе освоения и эксплуатации системы. Одной из разновидностей таких средств является система подсказок, которая полезна для начинающих консультантов, но может оказать негативное воздействие на опытных консультантов. Поэтому ДГС должны иметь адаптивные сервисные средства, которые позволяют настраиваться на уровень подготовленности консультанта или предусматривают возможность обхода (отключения) соответствующих блоков сервисных средств самим консультантом.

Во многих случаях функционирование ДГС представляется в виде периодического процесса, управляемого с помощью команд консультанта, которые последовательно переводят систему из одного состояния в другое. Состояние характеризуется постоянством в течение ограниченного интервала времени содержимого функциональных зон экрана дисплея. При этом описание диалога представляют в виде диаграмм состояний, условные обозначения и примеры которых приведены на рис. 5.39 и 5.40.

Грамматика ДГ-языка такой системы задается четверкой

$$G = \{V_T, V_H, s_0, P\},$$

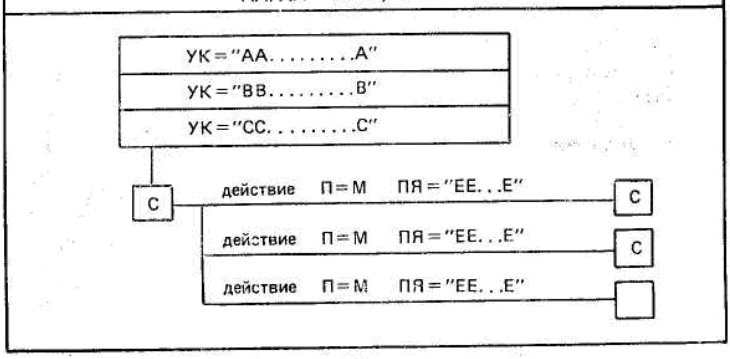
где V_T — терминальный алфавит, включающий конечный набор слов и действий пользователя с устройством ввода; V_H — нетерминальный алфавит, составленный из номеров состояний системы с начальным состоянием s_0; P—правила подстановок вида $\beta \rightarrow u_i c$. Подстановка $\beta \rightarrow u_i c$ интерпретируется как переход системы из состояния с номером $\beta \in V_H$ в состояние $c \in V_H$ при вводе цепочки терминальных символов $u_i \in V_H$, представляющих i-ю команду управления диалогом. Например, команда замены некоторого элемента изображения элементом из набора библиотечных графических изображений может быть сформирована из следующих терминальных символов:

a_1 — ЗАМЕНИТЬ; a_2 — ИМЯ ЭЛЕМЕНТА изображения; a_3 — ИМЯ БИБЛИОТЕЧНОГО ЭЛЕМЕНТА.

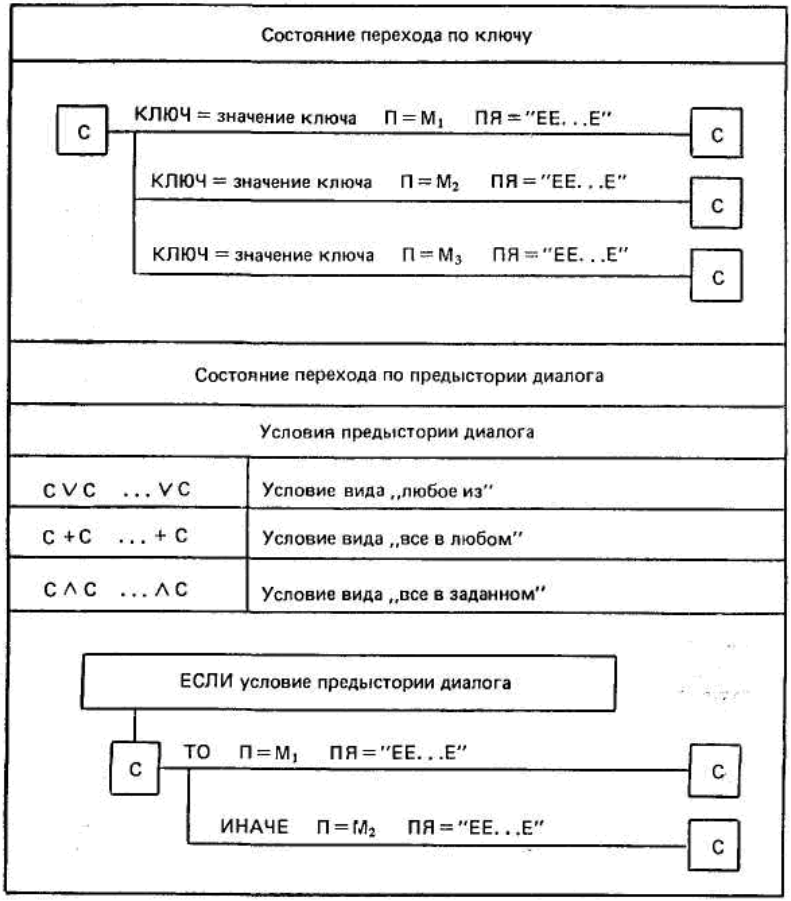
Правило подстановки имеет вид $\beta \rightarrow a_1 a_2 a_3 c$.

Представленным классом формальных грамматик определяется большинство ДГ-языков САК. Это — подмножество контекстно-свободных грамматик (КС-грамматик), называемых праволинейными грамматиками. Для альтернативных ДГ-языков имеет место однозначное отображение элементов терминального алфавита V_T на результаты действия пользователя с устройствами ввода графических терминалов (имя сегмента, номер альтернативы, координаты позиции, строка текста, введенное число).

Общие обозначения	
— С —	Состояние системы: активное: перехода по ключу, перехода по предыстории диалога
— ВХ —	Состояние выхода из подъязыка
— СД —	Состояние системной диагностики
$ДО_i$="АА...А"	i-е диагностическое сообщение (ДС) АА...А – текст ДС
ПЯ="ВВ...В"	Переход к диаграммам с именами ВВ...В
П=М	Вызов программы № М
Описание действий пользователя с устройствами ввода	
СФК="СС...С"	Кнопка функциональной клавиатуры (СФК)
ФК=Ф1 -- Ф2	Кнопки функциональной клавиатуры с номерами от Ф1 до Ф2
УКЗ	Указатель элемента изображения
ЛОК	Локатор положения следящего перекрестия
СТР	Алфавитно-цифровая клавиатура
ЧИС	Ввод К числовых значений
Активное состояние	
УК="АА...А"	Указание (подсказка) пользователю АА...А – текст указания



а



б

Рис. 5.39. Условные обозначения для построения диаграмм состояний: *а* — активные состояния: *б* — внутренние соотояния

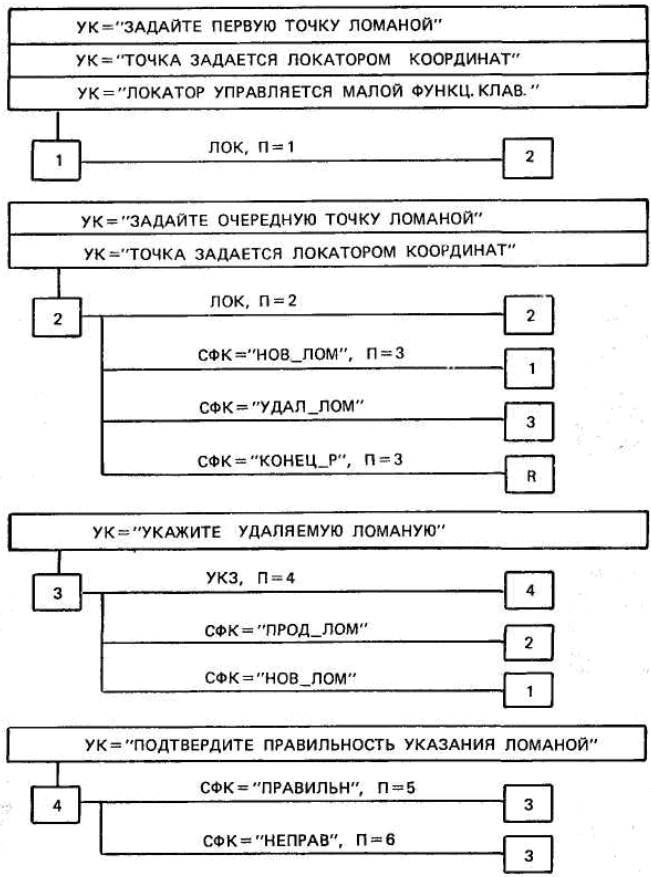


Рис. 5.40. Диаграмма состояний

Любой праволинейной грамматике может быть поставлена в соответствие автоматная грамматика G_a, которая с точностью до пустого слова порождает тот же язык. Это обеспечивается введением дополнительных нетерминальных символов с целью исключения вхождения в множество продукций грамматики правил подстановки вида $\beta \rightarrow u_i c$ и $\beta \rightarrow u_i$, где u_i — терминальная цепочка, длина которой $|u_i|>1$; $\beta, c \in V_H$. Так, правило подстановки приведенного примера может быть заменено новыми правилами:

$$\beta \rightarrow a_1 c_1;\ c_1 \rightarrow a_2 c_2;\ c_2 \rightarrow a_3 c.$$

где c_1, c_2 — дополнительные нетерминальные символы.

Подобному преобразованию соответствует переход от альтернативного ДГ-языка со свободным выбором к альтернативному ДГ-языку со сменными наборами команд.

Интерпретаторы ДГ-языков составляют основу диалоговых графических программ и предназначены для анализа правильности действий пользователя в процессе диалога и для организации выполнения в ЭВМ программ, соответствующих ожидаемой реакции на эти действия.

Моделью интерпретатора с автоматного языка является конечный автомат (преобразователь)

$$A_G=(S, X, Y, s_0, Z, \Phi, \Psi),$$

где S — непустое конечное множество состояний; X — непустое конечное множество входных символов; Y — непустое конечное множество выходных символов; s_0 — начальное состояние $s_0 \in S$; Z — непустое конечное множество заключительных состояний, $Z \in S$; Φ — функция перехода $\Phi : X \times S \rightarrow S$; Ψ — функция выхода $\Psi : X \times S \rightarrow Y$.

При семантической интерпретации конечного автомата как модели диалоговой программы его элементам придается следующий смысл:

X — множество допустимых воздействий консультанта — команды языка графического взаимодействия; Y — множество имен прикладных (семантических) программ, осуществляющих интерпретацию команд консультанта. Преобразователь A_G осуществляет перевод входной цепочки терминальных символов X в символы выходного алфавита Y. Каждому преобразователю A_G может быть поставлен в соответствие ориентированный граф Г, который называют диалоговым графом (рис. 5.41, *а*).

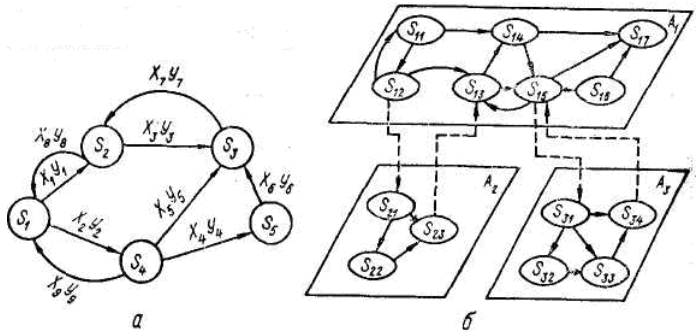


Рис. 5.41. Графическое представление модели интерпреатора диалоговых языков

Каждой вершине графа соответствует состояние автомата, а каждому переходу автомата из состояния S_i в S_j соответствует ориентированное ребро (S_i, S_j), помечаемое парой (X_k, Y_k).

При этом множество входных символов, допускаемых автоматом A_{ij}, совпадает с языком $L(G)$, порождаемым грамматикой G, т. е. можно говорить, что грамматика G задает ДГ-язык, допускаемый автоматом A_G.

Представление интерпретатора в виде конечного автомата широко использовалось при разработке различных ДГС. Однако, как показывает практика разработки ДГС, существует большое число ДГ-языков, принципиально не сводимых к автоматным языкам. Более подходящей для САК является модель диалоговой системы в виде автомата с магазинной памятью (МП-автомат), допускающего интерпретацию более широкого класса языков — контекстно-свободных (КС) языков. В качестве одного из недостатков такой модели можно отметить то, что операция над магазинной памятью не имеет эффективного графического представления.

Контекстно-свободные языки также допускает и система конечных автоматов, которая является эквивалентом автомата с магазинной памятью. В такой системе каждый автомат является автоматом вида A_G, а переходам между автоматами соответствуют операции с магазином. Представление интерпретатора в виде иерархической системы конечных автоматов является более наглядным и допускает простое отображение на традиционные структуры программирования. Если каждому конечному автомату такой системы поставить в соответствие диалоговую подпрограмму, интерпретирующую диалоговый подъязык, то в таком случае ДГ-программу можно представить в виде совокупности диалоговых подпрограмм, т. е. в виде структуры, принятой в традиционном программировании на процедурных языках высокого уровня и привычной большинству программистов. Представление системы конечных автоматов в виде графа приведено на рис. 5.41, *б*, на котором изображены связи между автоматами.

Однако и представление интерпретатора в виде магазинного автомата или системы автоматов, допускающих контекстно-свободный язык, не решает полностью всех проблем, связанных с разработкой ДГ-языков в САК, так как их практическая реализация, как правило, связана с необходимостью учета предыстории процесса автоматизированного консультирования, что эквивалентно введению контекста в ДГ-языки.

При этом можно выделить два типичных случая, когда продолжение диалога зависит от результатов семантических программ: первый

случай, когда продолжение диалога зависит от результатов работы последней выполнившейся семантической программы, и второй случай, когда продолжение диалога зависит от факта успешного выполнения группы семантических программ, выполнившихся в разных состояниях системы. Этот случай определим, как учет предыстории диалога.

Первый случай соответствует наличию в семантической программе более чем одного выхода, по крайней мере, существует еще и выход по сбою. Примером сбоя может быть случай, когда объект, указанный на экране дисплея, не принадлежит к классу объектов, обрабатываемых данной программой. Интерпретация ДГ-языков таких систем обеспечивается магазинным автоматом $A_м$. Автомат $A_м$ может быть построен на основе автомата A_G, дополненного памятью магазинного типа и множеством внутренних состояний $\hat{S}$, в которых производится анализ содержимого магазина, а также определяется значение соответствующих функций перехода и выхода. Над магазином определена операция записи SET (α), которая может производиться в семантической программе. В этом случае значение α соответствует выходу программы. Значение функций перехода и выхода во внутреннем состоянии определяется в зависимости от значения символа, находящегося на вершине магазина. Введение внутренних состояний $\hat{S}$ позволяет кроме синтаксического анализа ДГ-языков проводить анализ семантических условий, что повышает устойчивость ДГС к сбоям, связанным с некорректными действиями консультанта.

Во втором случае при определенной организации ДГ-языка успешному выполнению семантической программы можно поставить в соответствие прохождение диалоговой графической программы через некоторое состояние, а учет предыстории диалога может быть задан путем анализа трех типов зависимостей над состояниями диалога.

Состояние S_j функционально зависит от состояний $X_1, X_2,.....,X_n$, если к моменту перехода в состояние S_j система находилась в каждом x_i $(i = 1,2,\ldots, n)$ хотя бы по одному разу. Состояние S_j слабо зависит по контексту от состояний x_i $(i = 1,2,\ldots, n)$, если к моменту перехода в состояние S_j система находилась в каждом x_i $(i = 1,2,\ldots, n)$ строго один раз. Состояние S_j сильно зависит по контексту от состояний x_i $(i = 1,2,\ldots, n)$, если к моменту перехода в состояние S_j система находилась в каждом x_i $(i = 1,2,\ldots, n)$ строго один раз, и при этом все состояния должны быть в строго заданной последовательности.

Для анализа зависимостей состояний вводятся соответственно три типа условий перехода: u_1 — «любое из»; u_2 — «все в любом»; u_3 — «все в заданном». Условие «любое из» истинное, если ДГ-программа побывала в любом из заданного множества состояний, условие «все в любом» истинно, если система побывала во всех состояниях из заданного множества в любом порядке, а условие «все в заданном» истинно, если система побывала во всех состояниях из заданного множества в строго заданном порядке.

Интерпретатор ДГ-языка, позволяющий делать анализ приведенных условий, кроме функций преобразователя A_G и $A_м$ должен в процессе функционирования ДГС фиксировать в специальной памяти номера и внутреннее автоматное время пройденных состояний и иметь соответствующие внутренние состояния, в которых анализируются предписанные этим состояниям контекстные условия по предыстории диалога типа u_1, u_2, u_3, и определять пути дальнейшего ведения диалога.

Программа интерпретатор, реализующая представленную модель, выделяется в независимую часть ДГС и составляет основу унификации средств интерпретации ДГ-языков рассмотренного класса**.**

5.8.5. Средства разработки и поддержки языков консультирования

Общая схема обработки формальных описаний на ЭВМ. Применение языков консультирования позволяет сформулировать описание консультируемой проблемы или задание на консультирование, программу управления процессом консультирования или выдачи документации в виде, удобном для человека и для ввода в ЭВМ, для документирования и для обмена информацией между ЭВМ различных типов. Такие описания будем называть представлением объектов (проблем), процессов или заданий на исходном языке.

Для обработки формальных описаний на ЭВМ требуется их преобразование. Так, представление проблемы на исходном языке преобразуется во внутримашинные структуры данных в виде таблиц, массивов, списков, векторов, что обеспечивает быстрый доступ программ к элементам данных. Форматы данных соответствуют стандартным двоичным машинным форматам чисел с фиксированной и плавающей запятой, символьных строк, логических переменных и других, что позволяет использовать их в качестве операндов машинных команд.

Задание на консультирование преобразуется во внутримашинное представление данных и вызовов, необходимых для его использования консультационными процедурами, в результате выполнения которых генерируются промежуточные или окончательные рекомендации в форматах некоторых внутренних языков консультирования. Использование процедур подсистем документирования позволяет получить результаты консультирования на выходных языках в виде текстовых документов, таблиц, графиков, чертежей и т. д.

Представленная схема преобразования информации в ЭВМ позволяет сформулировать абстрактную модель САК с точки зрения лингвистического обеспечения в виде совокупности обрабатывающих подсистем. Каждая обрабатывающая подсистема определяется как пара $P_k = (L_k, J_k)$, где L_k — язык, J_k — транслирующая программа или языковый процессор для этого языка. Абстрактная модель представляет собой иерархию таких информационно связанных подсистем, причем на k-м уровне информация представляется на языке L_k, преобразуется языковым процессором J_k в представление на языке (k+1)-й обрабатывающей подсистемы $P_{k+1} = (L_{k+1}, J_{k+1})$ и т. д. Верхний уровень такой модели представлен входными языками, средние уровни — внутренними и промежуточными языками, а нижний — выходными языками. В абстрактную модель в качестве одной из обрабатывающих подсистем может быть включен и консультант, который анализирует результаты консультирования и управляет процессом консультирования, например, с помощью устройств ввода и языка графического диалога.

Совокупность языков и языковых процессоров составляет так называемую систему программирования САК и выполняет функции лингвистического обеспечения систем автоматизированного консультирования, так как формальный язык без соответствующего языкового процессора не может стать средством консультирования. Основные типы языковых процессоров — трансляторы и интерпретаторы.

Транслятор преобразует описания с одного языка на другой за один или несколько проходов в зависимости от класса и структуры входного языка и генерирует так называемый объектный модуль представления входного описания на выходном языке. Этот модуль при необходимости можно многократно выполнять или использовать без повторной обработки на языковом процессоре.

Интерпретатор анализирует представление каждого предложения на исходном языке и сразу осуществляет действия, предписанные этим предложением. Это приводит к необходимости выполнять

преобразования с помощью языкового процессора при каждом повторном использовании описания. В большинстве случаев применение трансляторов приводит к меньшим затратам времени и большим затратам оперативной памяти ЭВМ по сравнению с затратами при интерпретации.

В общем случае языковый процессор состоит из следующих функциональных блоков: блока ввода исходного описания, лексического анализатора, синтаксического анализатора, блока управления и семантической интерпретации, блока выдачи диагностических сообщений.

Блок ввода исходного описания предназначен для ввода и преобразования информации с алфавитно-цифровой или графической формы во внутримашинную двоичную. Для алфавитно-цифрового описания блок ввода обычно представлен простой подпрограммой, которая вызывается лексическим анализатором. Для графических языков блок ввода выполняется в виде программного процессора, который осуществляет преобразование данных от разнотипных графических устройств в единую промежуточную форму на основе унификации *виртуальных* устройств ввода.

Лексический анализатор распознает введенные терминальные символы исходного языка, заменяет цепочку терминальных символов некоторым внутренним представлением, которое более эффективно для обработки синтаксическим анализатором.

Синтаксический анализатор проверяет правильность синтаксиса введенной цепочки (предложения). Все синтаксически правильные цепочки передаются для исполнения в *блок семантической интерпретации,* который может выполнять требуемые подстановки для порождения цепочек выходного языка или осуществлять вызов программ, необходимых для выполнения действий, предписанных входной цепочкой.

Блок выдачи диагностических сообщений обычно выполняется в виде специализированной программы, которая по входному коду считывает текст сообщения из соответствующего файла и выводит его на экран дисплея и (или) в файл протокола работы и на устройство печати.

Средства поддержки лингвистического обеспечения. В зависимости от типа исходных языков и методов их обработки различают следующие три основные группы средств поддержки лингвистического обеспечения: проблемно-ориентированные системы или генераторы пакетов прикладных программ; макрогенераторы и другие расширяющие системы программирования, построенные по

схеме предтрансляторов; метасистемы или системы построения трансляторов по схеме параметрических систем программирования.

Проблемно-ориентированные системы предназначены для построения на их основе специализированных программ. В качестве параметров настройки в системе этого класса может использоваться описание модели консультируемой проблемы (объекта) или предметной области. Система обеспечивает автоматизированный подбор программных модулей и конструирует на их основе программы для консультирования проблем (объектов) заданного класса. Она может обеспечить также автоматический обмен данными между внутренней и внешней памятями ЭВМ или совмещение модулей, написанных на различных языках программирования.

Системы этого класса ориентированы на создание определенного сервиса для разработки программного обеспечения, и задачи автоматизации разработки и поддержки различных языков САК при этом практически не решаются.

Макрогенераторы и предтрансляторы представляют средства для расширения языков программирования и занимают промежуточное положение между пакетами прикладных программ и специализированными языками консультирования. С помощью набора макроопределений создается надстройка над базовым языком, которая отражает специфику предметной области применения. Общая схема использования метода предтрансляции показана на рис. 5.42.

Развитие возможностей этого метода основано на использовании синтаксических макропроцессоров, в которых выполнению макроподстановок предшествует синтаксический анализ программ на исходном языке. Ряд синтаксических макропроцессоров допускают возможность использования различных пар входных и выходных языков, что приближает их к системам построения трансляторов или метасистемам.

Рис. 5.42. Схема обработки входных языков методом предтрансляции

Метасистемы можно разделить на многоязыковые системы программирования и метатрансляторы. Многоязыковые системы программирования создаются на основе выделения общей части трансляторов с нескольких конкретных языков — ядра системы, которое реализует алгоритмы синтаксического контроля и анализа для определенного класса языков.

Если грамматика проектируемого языка L_k относится к классу языков, распознаваемых ядром системы, то создание языкового процессора J_k сводится к описанию грамматики языка L_k на специальном метаязыке системы и генерации таблиц описания грамматики G_k, которые обеспечивают настройку синтаксического анализатора на язык L_k.

Метатрансляторы типа МТ-системы позволяют создавать языковые процессоры для входных языков с различными грамматиками. В качестве параметров в таких системах задается информация о синтаксической структуре языка и его семантике. Для описания

синтаксиса часто используется нормальная форма Бэкуса — Наура, а семантика задается на специальном метасемантическом языке. Структура метатрансляторов типа МТ-систем приведена на рис. 5.43.

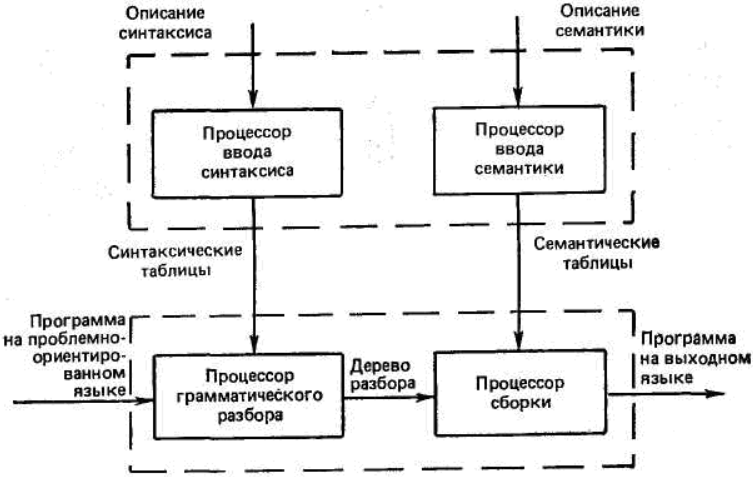


Рис. 5.43. Структура системы метатранслятора

В результате работы системы программа на входном языке преобразуется в выходной, которым в МТ-системе может быть любой язык программирования. Таким образом, метатрансляторы типа МТ-системы позволяют автоматизировать создание языковых процессоров для различных входных языков, обеспечивая его перевод на любой выбранный язык программирования.

Для построения лексических и синтаксических анализаторов входных языков в составе операционных систем имеются соответствующие генераторы.

Автоматизация разработки диалоговых графических языков в системах автоматизированного консультирования достигается за счет применения унифицирешенного интерпретатора с класса диалоговых языков и двуязыковой концепции проектирования диалоговых программ. Основу двуязыковой концепции составляет использование простого специализированного языка для описания диаграмм состояний параллельно с применением традиционных языков программирования.

Диалоговая программа состоит из описания диаграмм состояний на специализированном языке и семантического описания реакции системы на языке программирования. Тексты этих частей могут быть

совмещены, если, например, включить строки описания диаграмм в программу семантического описания в качестве отмеченных комментариев используемого языка программирования. На рис. 5.44 приведен пример такой программы для диаграммы на рис. 5.41. Текст диалоговой программы подвергается трансляции двумя трансляторами: с языка описания диаграмм состояний и с языка программирования. В результате первой трансляции создается файл описания алфавита и синтаксиса ДГ-языка.

```
C   Пример описания ДГ-языка и ДГ-программы для построения  изо-
C   бражения, состоящего из ломаных линий.
C   Операторы с коментарием CD-операторы диалогового   языка,
C   включенные в текст исполняющей программы.
C
            SUBROUTINE PLINE
            NAME = Ø                        !Имя сегмента
C    Определение диалогового языка
CD  SUBLANG = „DRPLIN", STATE = 3, DIAGN = Ø;
 C
        !Ø Ø CALL INTERP
        CALL INTPAR (IRET, NPR)
C    Если IRET = Ø, то выходим из языка
          IF(IRET.EQ.l) GOTO 999
C    NPR — номер прикладной программы, подлежащей выполнению
          GOTO (1 Ø, 2 Ø, 3 Ø, 4Ø) NPR
C
C_____________________ Описание    состояния N 1_____________________
CD  S1:    MS =«Задайте первую точку ломаной линии»;
CD          MS = «Точка задается локатором координат»;
CD          MS = «Локатор  координат — следящее перекрестье»;
C
CD   LOC        P = 1 Ø,   S = 2;
     1Ø     NAME = NAME+ 1
            CALL GCTLC (NPT,X,Y)  !Получить координаты точки
           CALL GCRSG (NAME)       !Открыть сегмент
            CALL MOVA2 (X,Y)          !Перейти в начало ломаной
            GOTO 1 Ø Ø
C
C_____________________   Описание состояния N 2 _____________________
CD  S2:    MS = «Задайте очередную точку ломаной линии»;
CD          MS = «Точка задается локатором координат»;
```

```
C
CD  LOC,      P = Ø;   S = 2;
          CALL GGTLC (NRT.X.Y) !Получить координаты точки
             CALL LINA2 (X,Y)          !Вычертить отрезок ломаной
             GOTO 1ØØ
 C
 CD, CHC = "НОВ—ЛОМ", P = 3Ø,   S = 1;
               CALL GCLSG                    !Закрыть сегмент
               GOTO 1ØØ
 C
CD  CHC, = "УДАЛ — ЛОМ"            S = 3;
CD CHG = "КОН — РАБ", P = 5Ø, S = R;
         5Ø     CALL GCLSG                   !Закрыть сегмент
                GOTO 1ØØ
C
C_____  Описание состояния  N 3_____
CD  S3:    MS = "Укажите удаляемую линию";
C
C     PIC, P = 4Ø,   S = 3;
               CALL GGTPC (NAMSEG, IPIC)   !Получить имя сегмента
               CALL GDSG (NAMSEG)                !Удалить сегмент
               GOTO 1ØØ
CD            CHC = «НОВ — ЛОМ», S = 1;
CD    END;
C
       999 RETURN
             END
```

Рис. 5.44. Текст диалоговой программы для диаграмм состояний рис. 5.41.

Применение двуязыковой концепции дает возможность использовать единственный язык для описания диаграмм состояний вместе с различными языками программирования высокого уровня. Это исключает необходимость расширять каждый язык программирования диалоговыми компонентами, улучшает технологию построения и отладку диалоговых программ. В зависимости от стадии консультирования и уровня подготовки специалистов в области программирования на практике используются диаграммы состояний и несколько уровней спецификации семантического описания.

На низшем уровне используется обычное текстовое описание необходимых действий для преобразования данных и результатов, ориентированное на заказчика системы. Для реализации модельной системы с ограниченными возможностями задается следующий уровень. На этом уровне для описания семантики используется простой язык программирования. Это позволяет быстро и недорого построить модельную систему с необходимыми функциями, но с ограниченными возможностями по сложности консультируемой проблемы. Самый высокий уровень предназначен для профессиональных программистов. Он охватывает реализацию системы на одном из языков программирования высокого уровня, с учетом всех требований заказчика к функциям и эффективности системы.

5.9. Информационное обеспечение САК

5.9.1. Определения, состав и общие требования к информационному обеспечению САК

Под информационным обеспечением (ИО) САК будем понимать совокупность сведений, необходимых для выполнения автоматизированного (автоматического) консультирования, представленных в заданной форме. ИО САК предназначено для организации, использования (получения), хранения и поддержания в актуальном и корректном состоянии всех сведений (данных), необходимых для процесса консультирования. Функции ИО реализует информационная система САК.

Компонентами ИО являются документы, содержащие описания стандартных консультационных процедур, типовых рекомендаций, типовых элементов, комплектующих изделий, материалов и другие данные, а также файлы и блоки на машинных носителях с записью указанных документов, обеспечивающие функционирование соответствующих подсистем САК. Совокупность компонентов ИО образует информационную базу САК. Структура информационной базы САК с классификацией ее компонентов приведена на рис. 5.44.

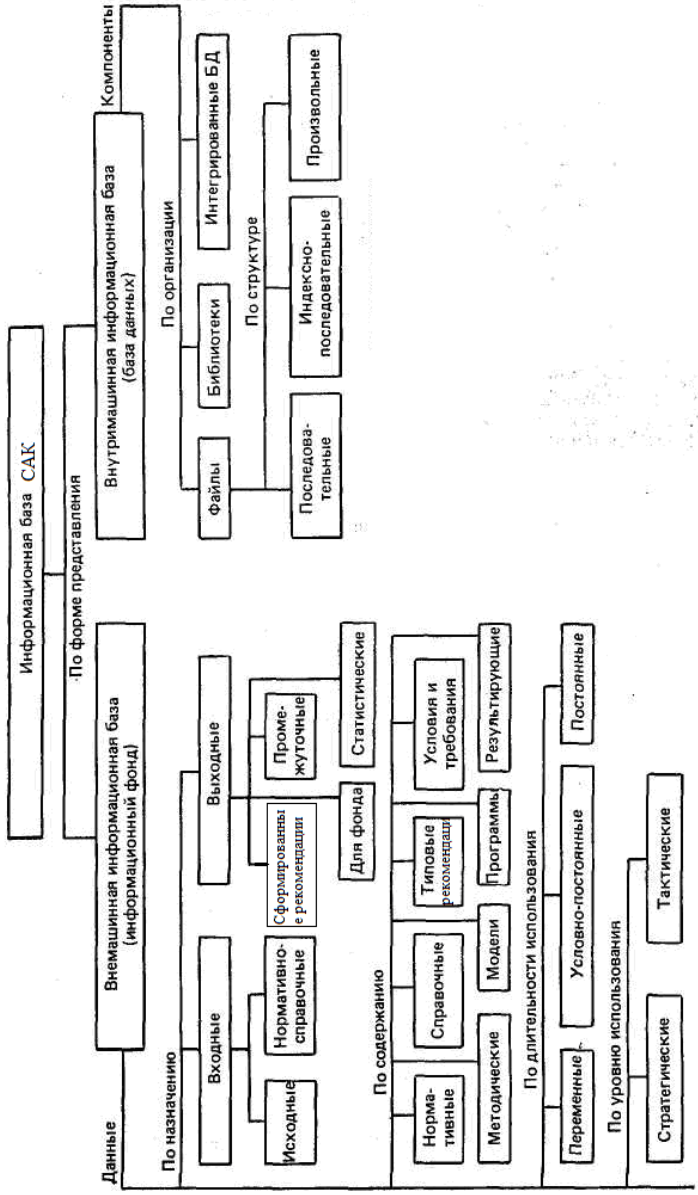


Рис. 5.44 Структура информационного обеспечения САК

Информационную базу САК по форме представления данных можно подразделить на две составляющие: внемашинную (информационный фонд) и внутримашинную (базу данных).

Информационный фонд. В состав информационного фонда входят текстовые и графические документы, классификаторы, справочники, кодификаторы, паспорта и описания типовых консультационных рекомендаций, картотеки, микрофильмы и другие источники информации для восприятия непосредственно персоналом САК.

По назначению данные информационного фонда подразделяют на входные и выходные. Входными являются данные, используемые при проведении консультационных работ. Входные данные подразделяют на исходные и нормативно-справочные.

К *исходным данным* относятся сведения, которые задаются для консультирования конкретной проблемы (серии проблем). Такими данными являются функции и характеристики консультируемой проблемы, требования к решению задач консультируемой проблемы (приводятся в консультационном задании), а также результаты предконсультационных исследований.

Нормативно-справочные данные составляет информация, общая для всей области применения САК, либо для группы консультируемых проблем. К нормативно-справочным данным относят стандарты, нормали, руководящие и методические документы и материалы, сведения о проблемах, типовые рекомендации, математические модели, справочные сведения, сведения о функционировании САК и т. д.

Выходными являются данные, получаемые в результате функционирования САК. К выходным данным относят *принятые сформированные рекомендации по решению задач консультируемой проблемы* (расчетные значения, текстовые документы, чертежи, схемы), *промежуточные данные* (являющиеся исходными для последующих этапов формирования рекомендаций), *результирующие данные* для пополнения нормативно-справочного фонда и *статистические данные* (о функционировании САК).

По содержанию данные можно разделить на следующие группы: нормативные, справочные, методические, модели, типовые рекомендации, программы, условия и требования, результирующие.

Нормативные данные содержатся в документах нормативного характера (государственные стандарты, отраслевые стандарты, стандарты консультирующих служб, нормали, руководящие и инструктивные документы, директивные документы и т. д.).

Справочные данные содержатся в справочниках, каталогах, кодификаторах, информационных материалах, консультационной документации и литературе, отчетах по консультационным работам и т. д.

Источниками нормативной и справочной информации являются вышестоящие, информационные и патентные службы и организации, а также научно-технические библиотеки.

Методические данные содержатся в методических материалах и инструкциях, документах и литературе с изложением теории, математических методов, методик, специальных языков и т. д. Источниками методической

информации являются организации и службы, ведущие разработки по САК и в проблемных областях.

Модели представляют собой математические зависимости, наборы эвристических или алгоритмических процедур и данных о консультируемой проблеме (характеристиках, технических и экономических данных), ее функционировании. Наиболее часто используют математические, эвристические, цифровые и имитационные модели. Источниками моделей являются разработчики САК.

Типовые рекомендации формируют на основе опыта консультирования с целью фиксации данных по разработанным (созданным) рекомендациям и использования их в последующих консультационных работах. К типовым рекомендациям относятся общие (обобщенные) типовые рекомендации, типовые рекомендации по отдельным аспектам консультирования проблем на различных этапах, типовые чертежи и схемы, типовые графические элементы (условные обозначения, унифицированные чертежи элементов консультируемых проблем, элементы оформления графических материалов).

Программы вне машины хранятся в виде текстов программ и программной документации.

Условия и требования — это данные, являющиеся исходными для различных этапов и стадий консультирования проблем. Их формируют как исходные перед консультированием проблемы, либо как промежуточные при завершении определенного этапа или стадии консультирования.

Результирующие данные формируют на заключительных этапах консультирования в виде сформированных рекомендаций (характеристики, параметры, ограничения) и консультационной документации.

По длительности использования данные можно подразделить на переменные, условно-постоянные и постоянные.

Переменные данные характерны для конкретной консультируемой проблемы. Они изменяются при переходе от одной проблемы (серии проблем) к другой. К переменным данным относят условия, требования и результирующие данные.

Условно-постоянные данные характерны для группы консультируемых проблем. Они изменяются в процессе анализа результатов консультирования и в целом всего жизненного цикла проблемы. К условно-постоянным относят справочные данные и типовые рекомендации.

Постоянные данные характерны для всей области применения САК. Они изменяются при поступлении директивных указаний от

вышестоящих организаций и служб, а также при развитии САК, если изменяется область применения и методология консультирования. К постоянным данным относят нормативные, методические, модели и программы.

По уровню использования данные можно подразделить на стратегические и тактические.

Стратегические данные предназначены для принятия принципиальных рекомендаций на уровне главного консультанта или руководителей служб САК. К стратегическим относят постоянные и частично условно-постоянные данные.

Тактические данные предназначены для принятия конкретных рекомендаций на уровне разработчиков-консультантов. К тактическим относят условно-постоянные и переменные данные.

База данных (БД) включает определенным образом организованные данные информационного фонда, хранится на машинных носителях и используется непосредственно в процессе автоматизированного (автоматического) консультирования.

Компоненты БД можно классифицировать по организации и по носителям записи.

По организации компонентов выделяют файлы (или наборы данных), библиотеки и интегрированные базы данных.

Файл — это совокупность данных, определенным образом оформленных для хранения и обработки в ЭВМ.

Библиотека представляет собой каталогизированную совокупность информационных компонентов (данных, программ, рекомендаций и т. д.).

Интегрированная БД представляет собой совокупность взаимосвязанных информационных компонентов (файлов, наборов данных и их составных частей) и описания структуры и взаимосвязи данных.

Компоненты БД хранят на носителях записи.

Компоненты ИО САК должны обеспечивать возможность: надежного хранения информации в течение требуемых сроков; гибкой организации и открытой структуры, приспособленной к пополнению и объединению; логической структуризации данных по формальным признакам; эффективного доступа к данным (по запросам пользователей и программ пользователей); реализации ведения БД (просмотр, дополнение, корректировка и удаление данных); поддержки необходимой достоверности и релевантности (уместности) данных; реализации защиты от несанкционированного (неразрешенного) и некомпетентного (приводящего к порче данных) доступа;

максимального использования серийных технических и программных средств.

Надежность хранения обеспечивают комплексом организационных и программных мер и средств: организацией контроля (визуального, по формату, логического, по паритету, циклического, по контрольным суммам, с использованием специальных кодов); организацией защиты (от несанкционированного и некомпетентного доступа, помех, сбоев и т. д.); организацией корректировки (автоматической и по запросу); организацией протоколирования работы и анализа состояния БД; организацией копирования и восстановления БД.

Гибкость организации и открытость структуры позволяет пополнять и реорганизовывать данные без коренной перестройки или повторного создания всей системы информационного обеспечения.

Логическая структуризация данных по формальным признакам обеспечивает возможность минимизации данных, устранения излишней избыточности, организации эффективной логической и физической взаимосвязи данных, эффективного поиска данных.

Доступ к данным должен осуществляться по достаточно простым запросам пользователей в режиме диалога или запроса-ответа и по достаточно просто реализуемому обращению из прикладной программы.

Ведение БД должно быть в максимально возможной степени автоматизировано и реализовано в режиме диалога и обращения из прикладных программ.

Необходимая достоверность и релевантность данных должны определяться по требуемой точности результатов и принятых рекомендаций с учетом полноты данных и возможных ошибок при работе технических средств и программ.

Реализация *защиты от несанкционированного и некомпетентного* доступа позволяет разрешить доступ к соответствующим данным только лицам и программам, имеющим на это право, и предохранить данные от порчи.

Использование *серийных технических и программных средств* в значительной степени облегчает и ускоряет процесс создания системы ИО.

5.9.2. Принципы построения системы инфорационного обеспечения

Организация данных и их взаимодействие с программами пользователей в своем развитии прошли три основных этапа: разобщенный фонд данных, централизованный фонд данных, интегрированная база данных (банк данных).

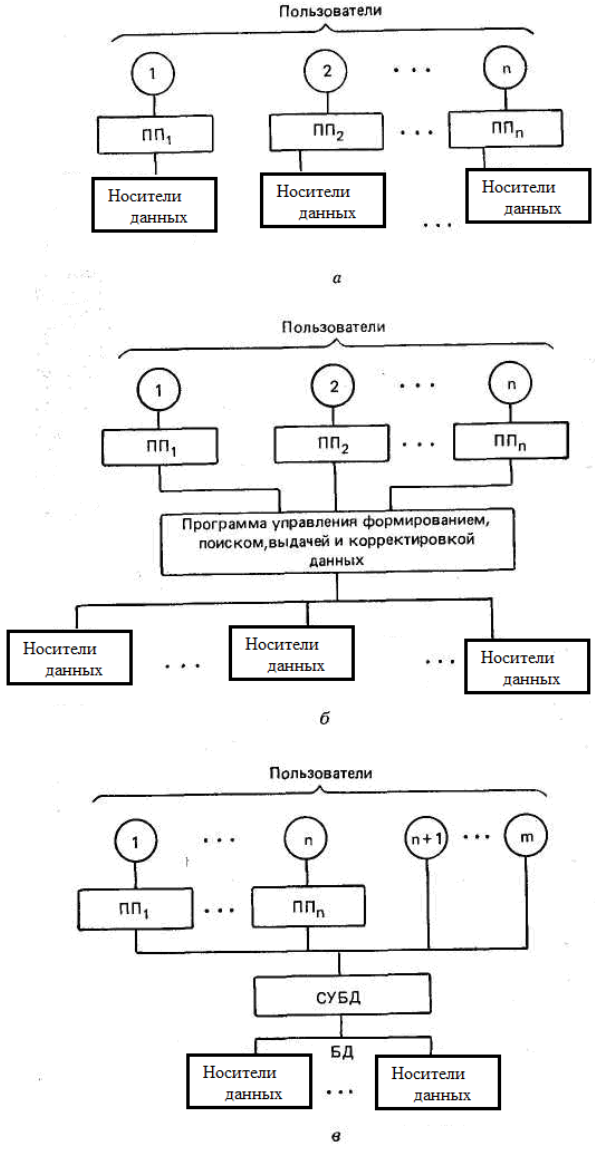


Рис. 5.45. Схемы организации данных:

а — разобщенный фонд данных; *б* — централизованный фонд данных; в —интегрированная БД (БнД); *ПП* — программма пользователя.

Разобщенный фонд данных (рис. 5.45, *а*) характеризуется наличием для каждой программы пользователя своих массивов данных (файлов). Эти массивы, как правило, другими задачами не использовались. Описание данных осуществлялось в программе пользователя. Для корректировки данных разрабатывались специальные программы (фрагменты программ). Такая организация обладает следующими недостатками: большой избыточностью информации, полной зависимостью данных и программ пользователя, сложностью реализации в программе пользователя доступа к данным.

Централизованный фонд данных (рис. 5.45, *б)* характеризуется наличием совокупности общих (для всех программ пользователей) не связанных между собой массивов данных и программы (комплекса программ), управляющей формированием, поиском, выдачей и корректировкой данных. Данные отделены от задач и организованы чаще всего в библиотеку массивов общего пользования. Форматы данных и структуры массивов унифицированы. При этом недостатки, присущие разобщенному фонду, хотя и имеют место, но значительно снижаются (особенно упрощается реализация доступа к данным из пользовательских программ), однако разработка сложной программы (комплекса программ) управления является достаточно трудоемкой. Для централизованного фонда характерна двухуровневая организация взаимодействия программы пользователя с данными: логический уровень обработки — физический уровень хранения.

Интегрированная база данных — банк данных (рис. 5.45, *в)* характеризуется объединением в одну систему программных средств управления данными и описаний данных и их взаимосвязей. При этом комплекс программ, управляющий накоплением, ведением, поиском, реорганизацией, выполнением сервисных функций, получил название системы управления базой данных (СУБД), а сама информационная система — банк данных (БнД). Достигаются минимально возможная избыточность данных, максимальная независимость программ и данных, простота организации запросов на доступ к данным от пользователей и программ. Для БнД характерна трехуровневая организация взаимодействия программы пользователя с данными: логический уровень пользователя — логический уровень внутрисистемной обработки — физический уровень хранения. Высокая степень унификации структур данных позволяет осуществлять централизованную (промышленную) разработку СУБД и поставку пользователям для широкого внедрения.

Анализ методов организации данных показывает, что наиболее эффективной формой системы информационного обеспечения САК

является БнД, однако это не исключает достаточно широкого использования централизованного фонда (особенно при организации библиотек) и разобщенного фонда (при необходимости работы с отдельными обособленными файлами).

Для описания данных разработаны специальные языки описания данных (ЯОД). Такое описание проводится в терминах имен и характеристик элементов БД, а также связей, которые существуют и должны поддерживаться в ней между экземплярами этих элементов.

5.9.3.Типы и модели данных

5.9.3.1. Основные понятия. Множества и отношения

Как уже отмечалось, составление программы для ЭВМ заключается, собственно, в записи соответствующего алгоритма обработки данных на языке, понятном ЭВМ. Естественно, что алгоритм, а значит и программа будут зависеть от того, как организованы данные, относящиеся к решаемой задаче. В связи с этим остановимся детально на вопросах представления данных и их структур в ЭВМ.

Наиболее простая структура данных соответствует случаю, когда между отдельными данными отсутствуют какие-либо внутренние связи. Их набор можно рассматривать как множество.

Множество есть любое собрание определенных и различных между собой объектов, которые можно рассматривать как единое целое. Так, например, можно рассмотреть множества людей, машин, телевизоров. Отдельные объекты множества называются элементами.

Для любых объектов $a_1, a_2, \ldots a_n$ множество обозначается через $\{a_1, a_2, \ldots a_n\}$, т. е. путем перечисления всех объектов, входящих в множество. Другая форма обозначения множества состоит в указании общего свойства объектов, из которых оно образуется $M = \{x|P(x)\}$.

Эта запись читается «множество элементов x таких, что имеет место $P(x)$». Вертикальная черта в записи расшифровывается как фраза «таких, что», а $P(x)$ обозначает свойство, характеризующее все элементы данного множества. Например, множество четных чисел M можно записать так:

$$M\{x|x \text{ —целое число, делящееся на } 2\}.$$

К основным операциям над множествами относятся операции объединения, пересечения, разности. Объединением множеств M_1 и M_2 ($M_1 \bigcup M_2$) называют множество тех элементов, которые содержатся по крайней мере в одном из множеств M_1 или M_2:

$$M_1 \bigcup M_2 = \{x|x \in M_1 \text{ или } x \in M_2\}.$$

Пересечением множеств M_1 и M_2 ($M_1 \bigcap M_2$) называют такое множество, которое содержит элементы, одновременно принадлежащие как M_1 так и M_2:

$$M_1 \bigcap M_2 = \{x | x \in M_1 \text{ и } x \in M_2\},$$

Разностью множеств M_1 и M_2 ($M_1 \backslash M_2$) называют множество тех элементов, которые принадлежат M_1 и не принадлежат M_2:

$$M_1 \backslash M_2 = \{ x | x \in M_1 \text{ и } x \notin M_2\}.$$

Прежде чем перейти к отношениям, рассмотрим еще одно важное понятие — упорядочение элементов. Если будем рассматривать только два элемента, то интуитивно ясно, что упорядоченная пара элементов — это просто совокупность, состоящая из двух предметов, расположенных в некотором определенном порядке. Когда это понятие используют в математике, то говорят, что упорядоченной парой элементов множества M называется объект $< x_1, x_2>$, состоящий из двух (не обязательно различных) элементов $x_1, x_2 \in M$ с указанием, какой из них следует считать первым, а какой — вторым. Так, если рассмотреть множество $M = \{ x_1, x_2, x_3, x_4, x_5\}$, то упорядоченные пары $< x_1, x_2>$, $<x_2, x_1>$ следует считать различными. К упорядоченным парам элементов из множества M относят также пары $< x_1, x_1> < x_2, x_2>$ и т. д. Множество всех упорядоченных пар $<x_i, x_i>$ из M называют декартовым (или прямым) произведением множества на себя и обозначают $M \times M$:

$$M \times M = \{ < x_i, x_i > | \, x_i \in M, x_i \in M\}.$$

Аналогичным образом вводится и прямое произведение множества M_1 на M_2:

$$M_1 \times M_2 = \{ < x, y > | \, x \in M, y \in M\}.$$

По аналогии с упорядоченными парами можно ввести понятие упорядоченной тройки $< x_1, x_2, x_3>$, упорядоченной четверки $< x_1, x_2, x_3, x_4>$ и, вообще, упорядоченной n-ки, где n — произвольное число. Упорядоченная n-ка элементов из M — это n (не обязательно различных) элементов множества M, данных в определенной последовательности $<a_1, a_2, ..., a_n>$. Упорядоченные n-ки элементов из множества M называют иногда кортежами над M (длиной n).

Кортежи как раз и используются для формализации отношений. Вообще термин «отношения» применяется для обозначения какой-либо связи между объектами или понятиями. Например, мы говорим «объект x лучше объекта y», «x меньше, чем y», «a равно b», «ДИОД Д2 включен в схему R15». Здесь примерами отношений являются выражения: «лучше», «меньше», «чем», «равно», «включен в», которые показывают, в каком отношении друг к другу находятся элементы.

Отношения бывают двуместные (или бинарные), трехместные (или тернарные) и вообще *n*-местные или *n*-арные, где *n* — произвольное число. Термин «бинарное (двуместное) отношение» говорит о том, что для его представления используется упорядоченная пара элементов.

Формально бинарное отношение (обозначим его *R)* между двумя множествами M_1 и M_2 может быть определено как подмножество прямого произведения $M_1 \times M_2$:

$$R\,(M_1, M_2) = \{<x, y> |\; x \in M_1, y \in M_2\}.$$

Другими словами, бинарное отношение *R* представляет собой множество упорядоченных пар $<x, y>$. Если $x \in M_1$, $y \in M_2$, и данная упорядоченная пара находится в отношении *R*, то это записывается в виде *xRy*.

n-местным отношением на множествах $M_1 \times M_2 \times \ldots \times M_n$, называется подмножество декартового произведения $M_1 \times M_2 \times \ldots \times M_n$:

$$R\,(M_1, M_2, \ldots, M_n) = \{< x_1, x_2, \ldots, x_n> |\; x_1 \in M_1, x_2 \in M_2, \ldots, x_n \in M_n\}.$$

Множества $M_1, M_2, \ldots, M_n$ называют областями определения (доменами), т. е. домены — множества, из которых черпаются конкретные значения элементов кортежа, *n*-местное отношение $R(M_1, M_2, \ldots, M_n)$ удобно представлять в виде таблицы, где каждая *i*-я строка есть кортеж $<x^i_1, x^i_2, \ldots, x^i_n>$, а каждый столбец соответствует одному и тому же компоненту декартова произведения, т. е. в нем могут появляться только элементы из соответствующей области определения (соответствующего домена).

Очень важное требование к отношениям заключается в том, что каждая строка (кортеж) рассматривается как элемент множества *R*, а в множествах дубликаты не имеют смысла, т. е., например, множества {1, 2, 3} и {3, 2, 3, 1, 2} считаются эквивалентными. Поэтому если в результате операций над отношениями появляются одинаковые строки, то в результирующем отношении должна остаться только одна из них.

Можно отметить, что *n*-местное отношение может быть использовано для представления набора объектов. Рассмотрим отношение, названное ТЕЛЕВИЗОР (табл. 5.8).

Таблица 5.8

Наименование телевизора	Индекс	Диаметр кинескопа, см	Цена, руб.
Горизонт	Ц-355	51	610
Рубин	Ц-266,д	67	1040
Темп	Ц-280д	61	755
Электрон	Ц-283	61	755

Каждая строка в табл. 5.8 выполняет роль описания отдельного объекта, а в данном случае атрибуты объекта интерпретируются столбцами отношений. Множество допустимых значений атрибута интерпретируется соответствующим доменом. Поэтому столбцы отношений называют также атрибутами и присваивают им имена. В этом случае можно говорить об отображении имен атрибутов в значения, принадлежащие соответствующим доменам. В общем случае при разработке структуры хранения данных используют различные виды таблиц.

Важную роль в процессе обработки данных играет упорядоченность строк таблиц. Иногда к таблице подсоединяют дополнительный «внешний» столбец, содержащий последовательность целых чисел от 1 до *n,* который отражает упорядоченность строк таблицы. При этом пользователь не имеет доступа к этому столбцу. Такая таблица называется **последовательной таблицей**. В ней каждой текущей строчке можно поставить в соответствие предыдущую строку и последующую.

Упорядочение таблицы может быть произведено и по значениям одного из столбцов, соответствующему арбитру. В этом случае она называется **упорядоченной.**

В системе обработки данных атрибут или совокупность атрибутов, позволяющие уникально определять каждую строчку таблицы, называют **ключом**. В общем случае в таблице может быть несколько ключей, один из которых назначается **первичным ключом**, а остальные будут представлять **возможные ключи.**

Следует помнить, что существует понятие — **вторичный ключ** или **ключ поиска**. Это ключ, посредством которого можно выделять из таблицы все кортежи, имеющие определенные значение этих атрибутов. Таким образом, вторичный ключ позволяет выделять из таблицы кортежи, обладающие интересующими нас свойствами.

Список имен атрибутов отношения называется **схемой отношения.** Если отношение называется R и его схема имеет атрибуты с именами $A_1, A_2,...,A_n$, то схема отношений будет

$$R (A_1, A_2,...,A_n).$$

Для табл. 5.8 схема отношения будет иметь вид: ТЕЛЕВИЗОР (наименование, индекс, диаметр кинескопа, цена).

Можно заметить, что существует аналогия между отношением и файлом, между кортежем и логической записью, между схемой отношения и форматом записи.

Графы. В математике хорошо разработано теоретическое направление, позволяющее выражать отношения между объектами в виде

специальных рисунков — графов. Граф представляет собой набор некоторых точек на плоскости *X,* которые называют **вершинами**, и множество отрезков *U,* соединяющих все или некоторые вершины. Отрезки *U* называют **дугами** или **ребрами** графа. Например, на рис. 5.35 изображен граф, состоящий из пяти вершин $X=\{x|x=a, b, c, d, e\}$ и трех ребер $U=\{u|u= (a,b), (a,d), (c,e)\}$.

Математически граф *G* можно определить как пару множеств *X* и *U: G(X, U).*

Форма представления отношения *R* в виде графа, заданного на множестве $M\times M$, позволяет элементы множеств *M* выбрать в качестве вершин, а в качестве дуг изобразить упорядоченные пары элементов $< x_i, x_j > \in M\times M$ характеризующие само отношение *R.* Так как дуга изображается стрелкой, то началу стрелки ставится в соответствие первый элемент кортежа $< x_i, x_j >$, а окончанию стрелки — второй.

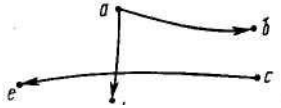
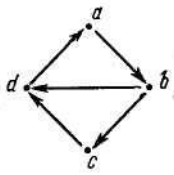


Рис. 5.45. Пример графа



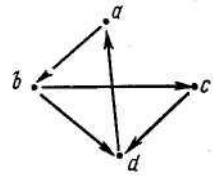


Рис. 5.46. Эквивалентные графы

Так, графу (5.45) можно поставить в соответствие некоторое отношение (<*a, b*>, <*a, d*>, <*c, e*>), заданное на прямом произведении множества *M* на себя, где *M*={*a, b, c, d, e).* Таким образом, при отображении с помощью графов некоторого отношения *R* между объектами вершины графа будут представлять объекты, а дуги — отношения между объектами.

Поскольку граф представляет собой абстрактное геометрическое отображение связей между объектами, дуги не обязательно должны быть прямыми, а вершины на плоскости могут быть расположены произвольно (рис. 5.46).

В практических задачах типична ситуация, когда связи между объектами являются симметричными, т. е. отношение *R* выполняется как между объектами $< x_i, x_j >$, так и между $< x_i, x_i >$. В этом случае вместо двух дуг, соединяющих объекты (рис. 5.47, *а*), можно нарисовать одну дугу (рис. 5.47, *б*).

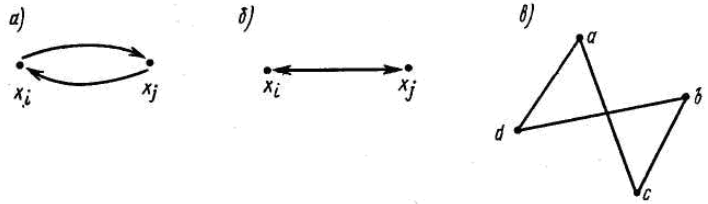


Рис. 5.47. Неориентированный граф

Когда все дуги в графе изображаются двойной стрелкой, стрелки можно не рисовать (рис. 5.47, *в).*

Дуги в этом случае называют ребрами, а сам граф — **неориентированным графом.** Граф, в котором задана ориентация дуг, называется соответственно **ориентированным графом** или **орграфом.**

Некоторые свойства графов. Иногда удобно представлять графы в виде некоторых матриц, в частности смежности и инциденций.

Две вершины x_i и x_j являются **смежными,** если они различны и существует дуга, идущая из x_i в x_j. Дуга называется инцидентной вершине *x,* если она заходит в эту вершину или исходит из нее.

Обозначим через x_1, x_2,..., x_i,..., x_j,..., x_n вершины графа, а через u_1,.....,u_k,.....,u_l его дуги.

Введем числа:

$$r_{ij} = \begin{cases} 1, \text{ если имеется дуга, соединяющая} \\ \text{вершины } x_i \subset x_j; \\ 0, \text{ если такой дуги нет.} \end{cases}$$

Квадратная матрица $\|r_{ij}\|$ порядка $n \times n$ называется **матрицей смежности** графы.

Для построения матрицы инциденций введем числа s_{il}.

$$s_{il} = \begin{cases} +1, \text{ если } u_l \text{ исходит из } x_i; \\ -1, \text{ если } u_l \text{ заходит в } x_i; \\ 0, \text{ если } u_l \text{ не инцидентна } x_i. \end{cases}$$

Матрица $S = \|s_{il}\|$ порядка $n \times k$ называется **матрицей инциденций** для дуг графа**.**

Подграфом G_A графа $G(X, U)$ называется граф, в который входит лишь часть вершин графа G, образующих множество $A_x \subseteq X$ вместе с дугами, соединяющими эти вершины.

Путем в графе G называется такая последовательность дуг $\mu=(u_1,.....,u_k)$, в которой конец каждой предыдущей дуги совпадает с началом следующей.

Контур — это замкнутый путь $\mu= (u_1,.....,u_k)$, у которого начальная вершина x_i совпадает с конечной x_k. Контур единичной длины, образованный дугой вида (x_i, x_i) называется петлей.

5.9.3.2. Типы и модели данных

Напомним, что когда речь идет о представлении данных в ЭВМ, следует различать физическую и логическую организацию данных (рис. 5.48).

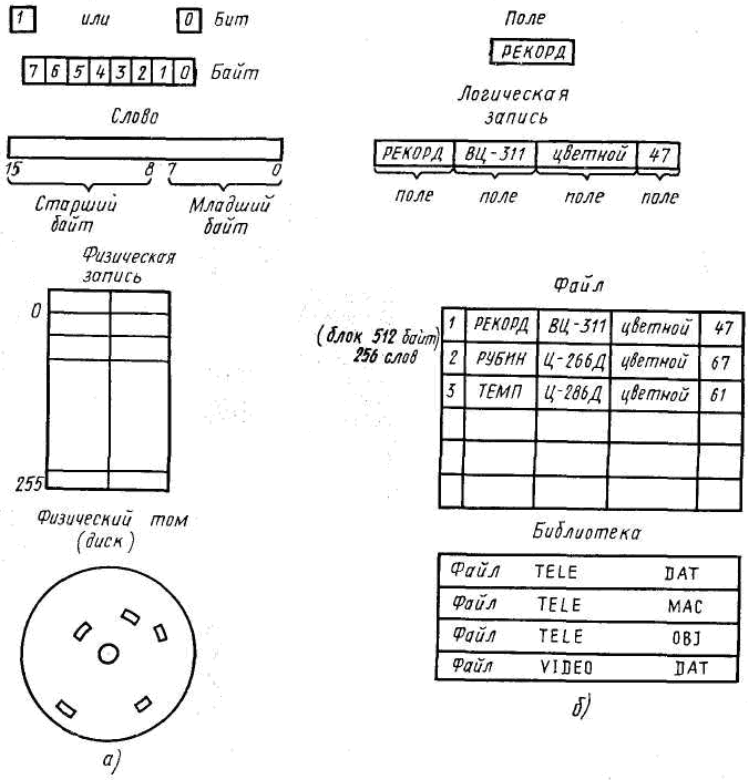


Рис. 5.48. Физическая *(а)* и логическая (б) организации данных

С другой стороны, когда говорится о данных применительно к какой-то конкретной задаче, следует помнить, что данные, закодированные в машине в виде нулей и единиц, несут в себе информацию о реальном объекте.

Если составляется алгоритм решения задачи, то программист пытается изобразить предметную область путем некоторого набора символов в виде структуры данных. При этом, естественно, его

интересуют не все объекты, а лишь некоторые, которые имеют непосредственное отношение к решаемой задаче.

Совокупность таких объектов называется **предметной областью,** а сами объекты — **объектами предметной области.**

Например, объектами могут быть: материалы; предметы, элементы, агрегаты, изделия; станки, оборудование; проблемы консультирования; рекомендации; консультационные процессы.

Для каждого объекта предметной области должны быть описаны характеристики, являющиеся наиболее важными для решаемой задачи. Эти характеристики называют **атрибутами.** Каждый атрибут может принимать определенные значения.

Например, когда речь идет о телевизоре как объекте предметной области, в качестве атрибутов (в зависимости от решаемой задачи) могут быть использованы характеристики (марка, индекс, изображение, размер кинескопа по диагонали, цена). Для определенного типа телевизора они примут конкретные значения. Например «Рекорд», ВЦ-311, цветной, 47 см, 640 руб.

Для любого объекта существует совокупность его параметров, которую будем называть **записью.** Она, в свою очередь, содержит **поля.** Семейство записей образует **файл.** Несколько файлов образуют **библиотеку.** При этом можно отметить следующие соответствия между объектами предметной области и логическим представлением:

- число объектов равно числу записей в файле;
- число атрибутов, описывающих объект, равно числу полей в каждой записи.

Очевидно, что при описании атрибутов могут использоваться числовые величины, строки символов, значения проводимых измерений. Чтобы осуществить обработку этих данных с помощью ЭВМ, необходимо как-то учитывать тот факт, что внутреннее представление информации в ЭВМ сводится в конечном счете к последовательности нулей и единиц. Действительно, пусть в ходе некоторой операции в регистре центрального процессора оказалась двоичная величина 1101010001010011. Как интерпретировать ее?

Для того чтобы машина могла однозначно интерпретировать данные, необходимо снабдить ее некоторой дополнительной информацией, сообщающей о «типе» обрабатываемой величины. Поэтому данные, закладываемые в ЭВМ, классифицируются по ***типам***. ***Понятие типа играет центральную роль в языках программирования высокого уровня***, поскольку при программировании на ассемблере программист должен сам контролировать, с какими данными он имеет дело.

В языках высокого уровня тип данных задается путем явного описания в тексте программ. В различных языках, в общем случае, имеются различные типы данных, но основные классифицируются следующим образом: простые; структурированные; ссылочные.

Простые типы не обладают внутренней структурой и представляют собой конечный набор типов, называемых также ***базисными***. К простым типам данных относятся: целый; плавающий; символьный; логический.

Целый и плавающий типы данных предназначены для представления числовых значений, символьный используется для образования текстов из символов (кода КОИ-7 или ДКОИ). Логический тип состоит из логических значений «истина» и «ложь» и применяется для выражения значений логических условий.

Данные любого простого типа характеризуются тем, что в любой момент времени ***каждому данному соответствует только одно значение.***

Но для решения большинства задач с помощью программ требуется возможность использования таких типов данных, которые могут содержать много значений. Поэтому в рассмотрение кроме базовых типов данных вводят структурированные.

Структурированные типы данных предназначены для конструирования из конечного набора базисных типов сложных структурных данных.

К этим типам данных относят: массивы (одно- и многомерные), последовательности (файлы, стеки).

Под **массивом** понимают группировку набора данных идентичного типа. Массиву присваивается имя, обозначающее всю группу данных. Причем к каждому элементу группы возможен индивидуальный доступ с помощью целого индекса, указывающего позицию элемента в группе. В *n*-мерных массивах позиция элемента задается с помощью *n* индексов.

Ссылочный тип данных предназначен для обеспечения ссылок на другие данные и называется ***указателем***. Этот тип применяется для динамического построения сложных структур данных и в ряде языков отсутствует.

Таким образом, из сказанного видно, что при написании программ на языке высокого уровня для того, чтобы ЭВМ правильно интерпретировало получаемую информацию, следует самым внимательным образом описывать типы используемых в программе данных.

Одним из основных способов структуризации данных является использование абстракций.

Абстракция предполагает, что несущественные детали должны быть опущены, а внимание должно быть сконцентрировано на основных общих свойствах множества объектов. Так, общее понятие ТЕЛЕВИЗОР — есть абстракция, отражающая множество наших представлений о конкретных телевизорах. Абстракция может быть многоуровневой т. е. объект абстракции одного уровня может рассматриваться как объект абстракции другого уровня и т. д. Таким образом, абстракция может использоваться для формирования нового типа из других типов. Например, БЫТОВАЯ РАДИОАППАРАТУРА определяется как абстракция типов ТЕЛЕВИЗОР, МАГНИТОФОН, ПРОИГРЫВАТЕЛЬ и т. д. Механизм абстракций широко используется при построении моделей данных.

Модели данных. Разработка САК любых типов предполагает, что данными, заложенными в систему, будут пользоваться не одним человеком, а группой консультантов, имеющих доступ к системе. Возможность совместного использования одних и тех же данных требует создания в системе консультирования единой информационной базы — базы данных. Под **базой данных** будем понимать массив данных, хранимый в ЭВМ и предназначенный для совместного использования группой людей. Использование баз данных позволяет:

- представлять в памяти ЭВМ сложные структуры информации, когда объектом хранения являются не только данные, но и структуры, в которые они организованы;
- сокращать дублирование информации за счет структурирования данных, что приводит к экономии памяти на внешних носителях и повышает надежность информации;
- повышать сохранность данных от несанкционированного доступа;
- обеспечивать независимость прикладных программ от изменений данных;
- повышать достоверность информации и сокращать затраты на обслуживание систем.

Если в случае создания отдельной программы пользователь может сам проводить подготовку данных, определять их типы и предъявлять ЭВМ, то при создании базы данных сразу для нескольких пользователей требования к организации данных в ЭВМ несколько усложняются.

Для того чтобы некоторую предметную область представить в базе данных, прежде всего необходимо выделить те понятия, которые интересны с точки зрения всех потенциальных пользователей базы. В

силу этих причин создание базы данных требует широкого использования механизма абстрагирования. На абстрактном уровне проводится классификация понятий и объектов предметной области. Сам процесс абстрагирования информации о предметной области, требуемой для создания базы данных, называют **логическим проектированием базы данных (проектирование данных).**

При логическом проектировании физические характеристики ЭВМ и особенности программ отдельных пользователей не учитываются. Логическое проектирование осуществляется в рамках некоторой заранее принятой модели абстрагирования, которую называют **модель данных.** Точнее говоря, под ней понимают основные понятия и способы, используемые при выполнении абстрагирования.

Описание предметной области в терминах некоторой модели данных называют **концептуальной схемой,** или **моделью.** Связь между предметной областью и концептуальной моделью может быть отражена определенным образом (рис. 5.49).

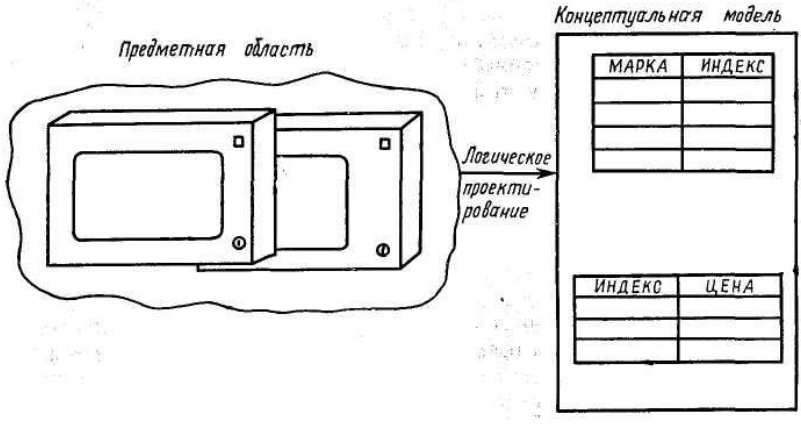


Рис. 5.49. Связь между предметной областью н концептуальной моделью

Концептуальная модель характеризует логическое представление данных и является тем мостиком, посредством которого связываются возможности физического представления данных в ЭВМ и в то же время их независимого использования в различных прикладных программах.

Концептуальную модель, описанную и скорректированную с точки зрения представления данных в памяти ЭВМ, называют **внутренней**

моделью. Поскольку база данных (БД) рассчитана на то, что каждый пользователь имеет доступ к определенным данным, концептуальная модель должна отражать еще один уровень логического представления данных для каждого конкретного пользователя. Такое представление концептуальной модели, непосредственно связанное с применением, называют **внешней моделью** (схемой). Так как каждый отдельный пользователь в большинстве случаев имеет отношение лишь к небольшой, вполне определенной части данных, хранимых в базе, то внешняя модель у него может быть своя. Таким образом, архитектура БД может быть упрощенно представлена, как на рис. 5.50.

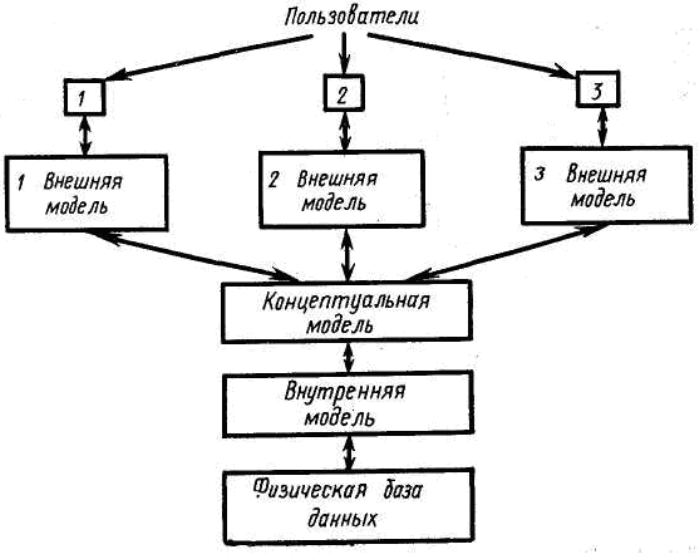


Рис. 5.50. Архитектура базы данных

Можно отметить, что на концептуальном уровне осуществляется формализованное описание информационной базы в терминах конкретной СУБД. Это означает, что на концептуальном уровне одна и та же информационная база описывается средствами различных СУБД. Поэтому при проектировании базы вводится еще один уровень описания. Модель этого уровня должна выражать информацию о предметной области в виде, независимом от конкретно используемой СУБД.

Этот уровень называют информационно-логическим или инфологическим (рис. 5.51).

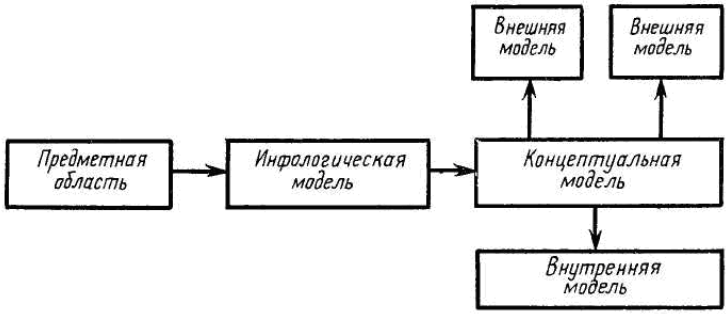


Рис. 5.51. Место инфологической модели в процессе разработки моделей данных

Инфологическая модель служит для формализации представления о предметной области до привязки к конкретным средствам реализации баз знаний.

Таким образом, при моделировании данных возникает проблема многоуровневого представления данных. Поэтому при проектировании БД уровни представления данных подразделяют на инфологический и даталогический. ***Причем, если инфологические модели используются для построения семантических (смысловых) моделей, отражающий информационное содержание конкретной предметной области, даталогические модели служат для реализации информационных баз в определенной вычислительной среде.***

Целостность данных. При создании БД всегда следует учитывать логические ограничения на значения данных и их соотношения. Они обычно представляют собой условия, при которых имеют смысл те или иные данные. Так, например, если будем рассматривать значение стоимость детали, то оно не может превышать значение СТОИМОСТИ ИЗДЕЛИЯ, в которое детали входят составной частью. Или, например, значение ДАТА ИЗГОТОВЛЕНИЯ телевизора не может принять значение 1912 г.

Логические ограничения, накладываемые на данные, рассматриваются как свойства, присущие данным и обеспечивающие адекватное отображение предметной области в БД. Если эти ограничения записать в БД, то их можно использовать для контроля целостности содержимого БД. Отсюда возникает понятие **целостности данных,** т. е. данные, хранимые в БД, не должны противоречить

заданным логическим ограничениям, которые называют **ограничениями целостности.** Они обычно задаются для множества объектов. Их можно разбить на два основных типа: внутренние и явные.

Внутренние ограничения обусловлены самой структурой принятой модели данных. Так, например, в **реляционных моделях данных** дубликаты записей не размещаются. Или **в иерархической модели** данных связи ограничены древовидной иерархической структурой.

В некоторых моделях данных вводятся ограничения, которые описываются в ***явном*** виде, с помощью специальных конструкций языка описания данных. К явным ограничениям целостности можно отнести **ограничения на значение атрибутов объекта**. Естественно, что ограничения в явном виде задаются не только для атрибутов, но и для типов объектов (сущностей) и связей. Так, например, если рассматривать сущность СТУДЕНТ, то может быть ограничено число студентов, обучающихся в одной группе. Для того чтобы уточнить, какие бывают ограничения на связи, необходимо рассмотреть основные типы связей.

Связь один к одному (1:1). Она определяет такой вид связи между двумя типами сущностей *A* и *B,* при котором каждому экземпляру сущности *A* соответствует только один *B,* и наоборот. Например, связь студент курса — номер зачетной книжки (рис. 5.52).

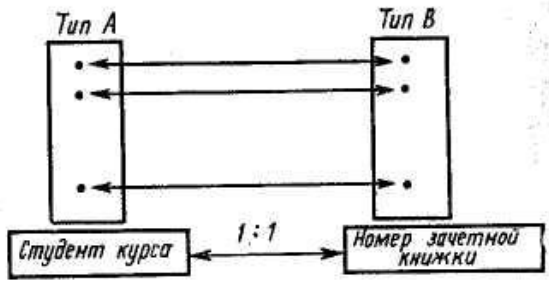


Рис. 5.52. Связь один к одному (1:1)

Здесь каждый экземпляр одного типа сущности однозначно определяет другой.

Связь один ко многим (1:M). Соответствует случаю, когда для двух типов сущностей *A* и *B,* одному экземпляру сущности *A* соответствует несколько (0, 1, 2,...,*M)* экземпляров сущности *B*. Однако каждому *B*

соответствует только один экземпляр сущности *A,* например, связь группа — фамилия, имя, отчество студента (рис. 5.53).

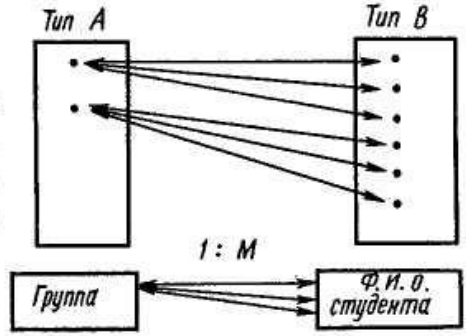


Рис. 5.53. Связь один ко многим (1 : М)

Связь многие к одному (М:1). Является вариантом связи, обратных к связи 1 : М, т. е. в этом случае многим экземплярам сущности типа *A* соответствует только один *B.* Например, Ф. И. О. студента — группа (рис. 5.54).

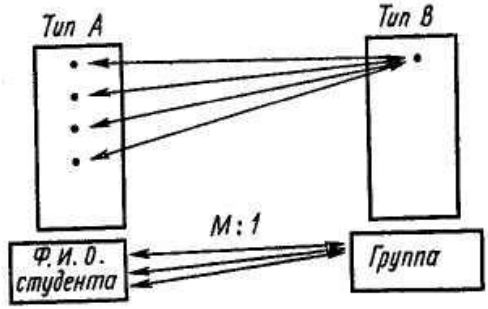


Рис. 5.44. Связь многие к одному (М : 1)

Связи типа 1:1, 1 :М, М: 1 называют ***функциональными***.

Связь многие ко многим (М:М). Соответствует случаю, когда каждому экземпляру сущности *A* может соответствовать несколько экземпляров сущности *B,* и наоборот. Например, телевизор — резистор (рис. 5.55).

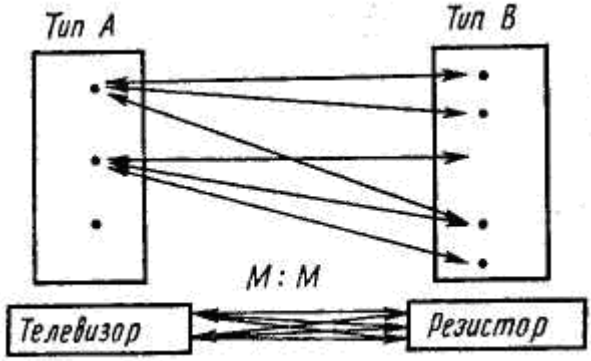


Рис. 5.55. Связь многие ко многим (М:М)

До сих пор рассматривались двусторонние связи между типами сущностей. Иногда целесообразно рассматривать одностороннюю связь от сущности *А* к сущности *В* (такие связи называют ***ассоциациями***). Ассоциации, в свою очередь, подразделяют на три типа: простая, сложная и условная.

Ассоциация простая (тип 1). Соответствует случаю, когда экземпляр сущности *А* определяет один и только один экземпляр сущности *В,* т. е. идентификация экземпляра сущности *В* является уникальной. Например, связь типа группа — фамилия старосты группы (рис. 5.56).

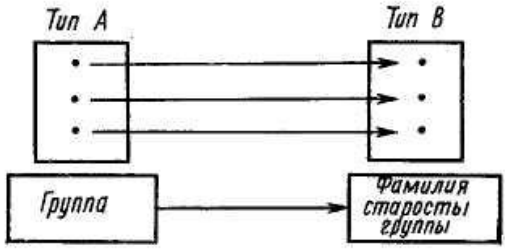


Рис. 5.56. Ассоциация простая по типу 1

Естественно, что в частном случае может оказаться, что в разных группах могут оказаться старосты с одинаковыми фамилиями, и поэтому обратная связь (от *В* к *А)* здесь не рассматривается.

Сложная ассоциация (тип М). Соответствует случаю, когда каждый экземпляр сущности *А* определяет несколько (нуль, один, два и т. д.)

экземпляров сущности *B.* При этом идентификация экземпляра типа *B* не обязательно является уникальной: например, связь между сущностями УЗЛЫ ИЗДЕЛИЯ и ПОСТАВЩИК (рис. 5.57).

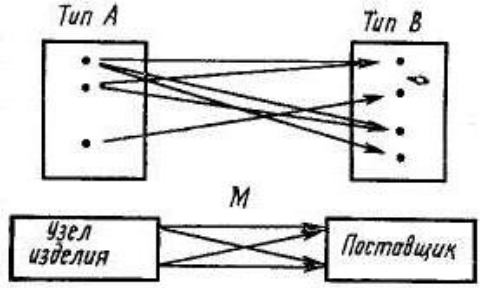


Рис. 5.57. Сложная ассоциация по типу *M*

Условная ассоциация (тип C). Описывает связь, когда для двух типов сущностей *A* и *B* может не существовать экземпляра типа сущности *B,* но если существуют, то он относится к единственному экземпляру сущности *A.* Например, связь между сущностями служащий — дата увольнения (рис. 5.58).

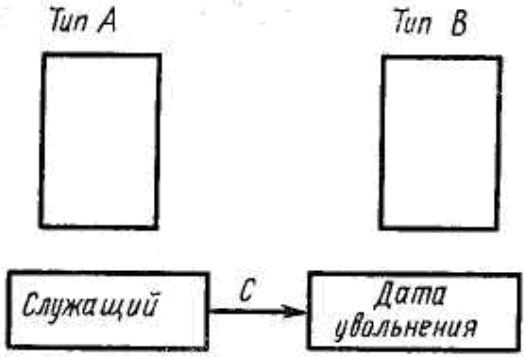


Рис. 5.58. Условная ассоциация по типу *C*

Приведенные типы связей не исчерпывают все множество видов ограничений, которые могут иметь место. Поэтому контроль выполнения ограничений в явной форме в конкретных моделях данных представляет собой сложную задачу и требует от СУБД целого набора актов доступа к БД и использования средств реляционного исчисления.

Операции над данными. В процессе обработки данных на ЭВМ приходится осуществлять над ними некоторые операции, которые необходимы для заданной конкретной операции.

Селекцию можно осуществить на основе использования логической позиции данного, его значений и связей между ними.

Использование для селекции **логической позиции данного** базируется на определенной упорядоченности данных в памяти системы. Упорядоченность данных позволяет использовать для селекции такие понятия, как первый элемент, последний, текущий, *n*-й. Относительно текущего элемента можно указать предыдущий элемент и последующий. Этот тип селекции называют также селекцией **посредством текущий.**

Использование для селекции **значений данных** требует задания в явном виде значений атрибутов, являющихся ***критериями селекции***. Простое условие определяется на одном атрибуте и одном его значении и обычно имеет вид

Имя атрибута—оператор условия — значения атрибута.

Под ***оператором-условием*** понимается один из арифметических операторов (= , ≠, <, >, ≤, ≥). Приведем пример простого условия селекции по значению данных:

средний балл студента ≥4,0.

Это условие означает, что из всего списка студентов следует отобрать студентов, имеющих средний балл не меньше 4,0.

Более сложные критерии селекции могут быть заданы выражениями, построенными на простых условиях с помощью логических операторов (И, ИЛИ, И). Например, условие

средний балл ≥4,0 и физика =5

означает селекцию тех студентов, у которых средний балл не менее 4,0 и которые сдали физику на 5.

Селекцию данных может производить не только по логической позиции или по значению, но и в соответствии с логическими связями между ними. Например, при наличии связи типа РАБОТАЕТ между сущностями ПРЕПОДАВАТЕЛЬ и КАФЕДРА можно осуществить селекцию всех лиц, работающих преподавателями на кафедрах университета и всех кафедр, имеющихся в университете.

Селекция такого рода называется ***селекцией по связности данных***.

По характеру производимого действия над данными различают следующие виды операций:

1. Идентификация данного и определение его позиции в базе;
2. Выборка (получение требуемых данных из базы);
3. Включение (запись новых данных в базу);

4. Удаление данных;
5. Обновление (модификация) данных.

Эти действия могут быть применены как к атрибутам, так и к типам сущностей и связям.

По характеру способа получения результата различают навигационные и спецификационные операции. Если результат представлен единичным объектом (значением атрибута, реализацией сущности или связи), полученным при прохождении по логическому пути (т. е. при **навигации)** в структуре БД, то соответствующую операцию называют **навигационной.**

Навигационные операции всегда предполагают селекцию посредством текущей. Отсюда возникает задача манипулирования текущими. Для определения текущих могут использоваться специальные индексы, которые могут совпадать с ключами.

Если в операции определяются только требования к результату, но не задается способ его получения, то операции называется **спецификационными.** В спецификационных операциях текущие на пользовательском уровне не видны. Они могут быть определены в терминах операций теории множеств: объединение, пересечение, разность. Поэтому результату спецификационной операции в общем случае соответствует некоторое множество объектов.

В процессе обработки данных возникает необходимость использования более сложных действий над данными, чем позволяют навигационные и спецификационные операции. Например, при выполнении операций селекции необходимо автоматически поддерживать целостность данных. Этого можно добиться путем создания специальной программы, которая будет осуществлять проверку большого количества данных. Эта программа, естественно, потребует использования локальных операций манипулирования над данными. С другой стороны, она может рассматриваться как глобальная операция проверки целостности данных.

Такие глобальные операции называют **процедурами базы данных.** Другими словами — ***это обобщенные операции изменения состояния базы данных.*** Процедура базы данных рассматривается как единая макрооперация, при выполнении которой ни одна другая процедура или программа не может обратиться к данным. Поэтому ***такие процедуры или операции называют еще транзакциями***.

Один из видов процедур БД — вычисление значений, которые непосредственно в ней не хранятся, например вычисление сумм, подсчет числа экземпляров, определение минимума, максимума. Процедуры этого вида называют **функциями агрегации.**

Важный вид процедур БД — вычисление значений атрибута, например, вычисление возраста студента по дате его рождения и текущей календарной дате. Процедура базы данных, обеспечивающая вычисление значения какого-либо атрибута, называется **виртуальным атрибутом.** Для пользователя он представляется как обычный атрибут, обладающий теми же свойствами, что и любой другой атрибут.

Процедуры БД применяются также для контроля целостности, контроля доступа к данным, расширения языка данных операциями, первоначально в них не предусмотренными.

Особый вид процедур БД составляют процедуры, активизирующиеся при определенных условиях и выполняющих одну или более операций включения, удаления или модификации. Такие процедуры называют ***запускаемыми включением, удалением, обновлением***.

Анализируя различия между процедурами БД и операциями, можно отметить следующее:

1. Процедуры при своем выполнении могут захватывать обширные области данных;
2. Процедуры могут реализовать широкий круг действий;
3. Вызовы процедур не выполняются пользователем;
4. Процедуры обычно описываются в схеме данных, в то время как операции включаются в пользовательскую программу.

Появление и использование БД привело к необходимости создания специальных программ, обеспечивающих управление данными, хранящимися в базе, т. е. систем управления базой данных (СУБД).

Таким образом, **СУБД — *это специальный пакет программ, посредством которого реализуется централизованное управление БД и обеспечивается доступ к данным***. База данных вместе с системой управления его являются составными частями **банка данных.**

Развитие СУБД привело к появлению ряда языков описания состояния предметной области, которое в СУБД интерпретируется состоянием БД. Множество допустимых состояний БД определяется схемой БД, задаваемой на **языке определения данных (ЯОД).**

Изменение состояния БД и извлечение данных из базы для последующей обработки обеспечивается средствами **языка манипулирования данными (ЯМД).**

Описание структуры данного некоторого типа на формализованном языке называют ***схемой*** этого данного. Язык описания данных — это язык высокого уровня, предназначенный для схемы БД. С его помощью описываются типы данных, хранящихся в базе, и их структура. Язык манипулирования данными (ЯМД) используется при написании прикладных программ, обращающихся к данным, храня-

щимся в базе. Основной функцией ЯМД является выполнение операций ввода-вывода при обработке информационной базы. Собственно говоря, совокупность ЯОД и ЯМД и определяет ***модель данных, понимаемую как совокупность методов и средств определения логической структуры БД.***

В качестве моделей данных, наиболее широко используемых при создании БД, обычно применяются: реляционная; сетевая; иерархическая. Рассмотрим их детально.

5.9.4. Реляционная модель данных

Основные понятия. В основе реляционной модели данных (РМД) лежит математическая теория отношений. Этим определяется и название модели (RELATION — отношение). ***Отношение служит средством структуризации данных.***

Таким образом, представив *n*-местное отношение в виде таблицы, тем самым определенным образом структурируем данные. Поэтому, естественно, подобные образования называют ***реляционной структурой*** или ***реляционным типом***. ***Массив данных***, представленный набором реляционных структур, образует ***реляционную*** **БД**, и схема реляционной БД будет представлена набором схем отношений:

$$R_1(A^1_1, A^1_2,..., A^1_k);$$
$$R_2(A^2_1, A^2_2,..., A^2_l);$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$R_m(A^m_1, A^m_2,...,A^m_n),$$

где A_1^j— ***имя атрибута***, R_l— ***имя отношения***.

Ограничения модели. Число ограничений в реляционной модели невелико, что обеспечивает достаточную свободу в выборе представления ***типов связей и сущностей***. Основным ограничением является невозможность представления в отношении дубликатов строк. Это ограничение позволяет уточнить понятие ключа отношения. В РМД ***ключ*** определяется как ***подмножество атрибутов, позволяющих однозначно идентифицировать кортеж***. Так как дубликаты строк в отношении запрещаются, то это означает, что каждое отношение имеет, по крайней мере, один ключ (состоящий из всех атрибутов). Отношение может иметь и несколько ключей, называемых ***возможными ключами***. Один из возможных ключей выбирается в качестве **первичного.** Следует иметь в виду, что первичный ключ не разрешается обновлять и никакой из его компонентов не может принимать значение «неопределено».

Второе ограничение модели состоит в том, что ***порядок столбцов в таблице является значимым.*** Пренебрегать упорядочением столбцов

можно только в том случае, если каждому столбцу присвоено уникальное имя.

На значения атрибутов в модели можно задавать ограничения в явном виде. Большинство явных ограничений, встречающихся на практике, это ограничения на зависимости между атрибутами. Явное задание ограничений обеспечивает возможность самостоятельного исследования зависимостей между атрибутами.

Зависимость отражает тот факт, что один объект зависит от другого. В РБД в качестве таких объектов рассматриваются либо ***аргументы, либо кортежи***. Одним из основных типов зависимостей, рассматриваемых в реляционных БД, являются **функциональные зависимости.**

Рассмотрим формальное определение функциональной зависимости, принимая следующие обозначения. Большими буквами *A, B, C,...* обозначим одиночные атрибуты, *X, Y, Z* — множество атрибутов, а *a, b, c,...* и *x, y, z,...* — соответствующие им значения. Большие буквы *R, S* будем применять для обозначения отношений.

Предположим, что существует некоторое **универсальное отношение** *U,* в котором каждый атрибут имеет уникальное имя. При этом будем считать, что множество атрибутов любого другого отношения представляет собой некоторое подмножество атрибутов универсального отношения *U.*

Пусть *A* и *B* атрибуты отношения *R.* Тогда говорят, что атрибут *B* отношения *R* функционально зависит от атрибута *A,* если в каждый момент времени каждому значению *a* соответствует не более одного значения *b.* Функциональную зависимость *f* атрибута *B* от атрибута *A* обозначают: $f{:}A \rightarrow B$.

Эту зависимость *f* можно также представить множеством упорядоченных пар $\{<a, b> | a \in A, b \in B\}$, в которых каждому значению *a* соответствует только одно значение *b.* При этом говорят, что *B* **функционально зависит** (или просто **зависит)** от *A***, а** *A* **функционально определяет** (или **определяет)** *B.*

Если существует единственная функциональная зависимость *B* от *A,* то ее обозначают просто $A \rightarrow B$. В случае отсутствия между ними функциональной зависимости вводят обозначения $A \neq B$.

Если $A \rightarrow B$ и одновременно $B \rightarrow A$, то между *A* и *B* существует взаимно однозначное соответствие, что записывается как $A \leftrightarrow B$.

Пусть: $f{:}A_1, A_2, \ldots, A_n \rightarrow B$ и $g{:}A_1, A_2, \ldots, A_m \rightarrow B$, где $m<n$. Так как атрибуты $A_1, A_2, \ldots, A_m$ функционально определяют *B,* то $A_{m+1}, A_{m+2}, \ldots, A_n$ называют ***посторонними*** в *f*. В этом случае *B* неполно зависит от

$A_1, A_2, \ldots, A_n$. Если для данного f не существует g с вышеуказанными свойствами, т. е. левая часть f не содержит посторонних атрибутов, то говорят, что реализуется ***полная функциональная зависимость***.

Пусть имеется множество атрибутов $A_1, A_2, \ldots, A_n$ отношения R, а также множество F функциональных зависимостей $X \rightarrow Y$, где X и Y — подмножества атрибутов множества $A_1, A_2, \ldots, A_n$. Тогда из функциональных зависимостей, входящих в F, могут быть выведены другие функциональные зависимости, присущие отношению R.

Обозначим через F^+ замыкание множества функциональных зависимостей F, т. е. полное множество зависимостей, которое можно получить из F. Множество зависимостей F^+ можно построить из F на основе следующих правил вывода функциональных зависимостей (ФЗ):

1. ФЗ1 (свойство рефлексивности).
2. ФЗ2 (свойство пополнения).
3. ФЗ3 (свойство транзитивности).

Это полный набор правил, т. е. он позволяет по заданному множеству F определить полное множество функциональных зависимостей F^+, присущих рассматриваемой схеме отношений $R(A_1, A_2, \ldots, A_n)$.

Рассмотрим эти правила более подробно.

Правило ФЗ1 (свойство рефлективности). Если $X \subseteq U$, $Y \subseteq U$ и $Y \subseteq X$, то имеет место функциональная зависимость $X \rightarrow Y$. Например, задано отношение $U(A_1, A_2, A_3, A_4, A_5)$.

Рассмотрим два множества атрибутов:

$$X = \{A_1, A_2, A_3, A_4\} \text{ и } Y = \{A_1 A_4\}.$$

Исходя из свойства транзитивности, можно сказать, что существует функциональная зависимость $X \rightarrow Y \in F^+$. Данное правило говорит о том, что, имея исходную функциональную зависимость $A_i A_j A_k A_n \rightarrow A_i A_j$, можно в состав множества атрибутов левой части выражения вводить любые атрибуты из множества U. При этом функциональная зависимость будет сохраняться. Можно также в правую часть включать атрибуты уже располагающиеся в левой части.

Правило ФЗ2 (свойство пополнения). Если $X \subseteq U$, $Y \subseteq U$, $Z \subseteq U$ и имеет место функциональная зависимость $X \rightarrow Y$, то $X \bigcup Z \rightarrow Y \bigcup Z$. В отличие от правила ФЗ1 данное говорит о том, что для его применения не существенно выполнение условий $Y \subseteq X$. Т. е. любые атрибуты из множества U можно одновременно подставлять в левую и правую части выражения функциональной зависимости F.

Например, имеется универсальное отношение $U(A_1, A_2, A_3, A_4, A_5)$ и заданы наборы атрибутов $X = \{A_1, A_3\}$, $Y=\{A_2, A_4\}$, $Z=\{A_5\}$. Тогда из условия, что существует функциональная зависимость $X \rightarrow Y$:

$$\{A_1,A_3\} \rightarrow \{A_2,A_4\}$$

следует, что имеет место зависимость

$$\{A_1, A_3, A_5\} \rightarrow \{A_2, A_4, A_5\}.$$

Правило ФЗ3 (свойство транзитивности). Если $X \subseteq U$, $Y \subseteq U$, $Z \subseteq U$ и имеют место зависимости $X \rightarrow Y$ і $Y \rightarrow Z$, то $X \rightarrow Z$. Например, имеются подмножества атрибутов $X=\{A_1, A_3\}$, $Y=\{A_2, A_4\}$, $Z=\{A_5\}$. Тогда из условия существования зависимостей $\{A_1, A_3\} \rightarrow \{A_2, A_4\}$, $\{A_2, A_4\} \rightarrow \{A_5\}$ следует, что имеет место зависимость $\{A_1, A_3\} \rightarrow \{A_5\}$.

Кроме этих правил часто используют дополнительные правила следствия ФЗ1, ФЗ2 и ФЗ3.

Правило ФЗ4 (свойство расширения). Если $X \subseteq U$, $Y \subseteq U$ и задана функциональная зависимость $X \rightarrow Y$, то тогда для любого $Z \subseteq U$ имеет место функциональная зависимость $X \bigcup Z \rightarrow Y$.

Правило ФЗ5 (свойство продолжения). Если $X \subseteq U$, $Y \subseteq U$, $W \subseteq U$, $Z \subseteq U$ и задана функциональная зависимость $X \rightarrow Y$, то для любых $W \subseteq Z$ имеет место зависимость $X \bigcup Z \rightarrow Y \bigcup W$.

Правило ФЗ6 (свойство псевдотранзитивности). Если $X \subseteq U$, $Y \subseteq U$, $W \subseteq U$, $Z \subseteq U$ и заданы функциональные зависимости $X \rightarrow Y, Y \bigcup W \rightarrow Z$, то имеет место функциональная зависимость $X \bigcup W \rightarrow Z$.

Правило ФЗ7 (свойство аддитивности). Если $X \subseteq U$, $Y \subseteq U$, $Z \subseteq U$ и заданы функциональные зависимости $X \rightarrow Y$, $X \rightarrow Z$, то имеет место функциональная зависимость $X \rightarrow Y \bigcup Z$.

Правило ФЗ8 (свойство декомпозиции). Если $X \subseteq U$, $Y \subseteq U$, $Z \subseteq U$ и при этом $Z \subseteq Y$ и задана функциональная зависимость $X \rightarrow Y$, то будет иметь место функциональная зависимость $X \rightarrow Z$.

Очевидно, что даже при небольшом числе зависимостей F число функциональных зависимостей F^+ имеет быть весьма велико.

Функциональная зависимость играет важную роль в моделировании данных. Вместе с тем можно отметить, что это отнюдь не общий вид зависимости. Существенную роль играет также многозначная зависимость.

Многозначные зависимости. Рассмотрим отношение $R(X, Y, Z)$, где X, Y, Z — множество атрибутов. Кортеж отношения $R(X,Y,Z)$ обозначим через $\langle x, y, z \rangle$. Если в отношении $R(X, Y, Z)$ присутствуют кортежи $\langle x, y, z \rangle$; $\langle x, y', z \rangle$;... (x, y, z'); $\langle x, y', z' \rangle$, то говорят, что существует

многозначная зависимость атрибутов Y и Z от X. Эта зависимость обозначается $X \rightarrow\rightarrow Y$.

Пример. Рассмотрим таблицу, в которой указаны школьники-победители олимпиад по предметам:

№ п/п	Ф. И. О. школьника	Школа №	Предмет
1	Давыдов В. А.	154	Математика
2	Давыдов В. А.	154	Физика
3	Кулешов К. Г.	820	Биология
4	Жукова А. М.	154	Математика
5	Жукова А. М.	154	Физика

В нашем примере имеют место две многозначные зависимости: ШКОЛА №→→ПРЕДМЕТ (например, № 154→→ (математика, физика), № 820→→(биология)) и ШКОЛА №→→Ф. И. ШКОЛЬНИКА

Другими словами, X многозначно определяет Y, если и только если множество $Y=\{y|(x, y, z)\in R\}$ определяется только X.

Многозначную зависимость определяют путем следующей проверки.

Если для двух кортежей t и s отношения $R(X, Y, Z)$ справедливо первое условие:

$$t[X]= s [X]$$

т. е. t и s совпадают по значениям атрибутов X и существует третий кортеж u, такой, что выполняется второе условие:

$$u[X, Y]=t[X, Y]; \quad u[Z]=s[Z],$$

то существует многозначная зависимость.

Пример. Рассмотрим отношение ПОБЕДИТЕЛИ ОЛИМПИАДЫ (Ф. И. О. ШКОЛЬНИКА, ШКОЛА №, ПРЕДМЕТ).

Проверим на основании вышеприведенного условия многозначную зависимость ШКОЛА № →→ ПРЕДМЕТ

1) t [№154]= s [№154].

Можно отметить, что существует четыре кортежа, у которых значение атрибута ШКОЛА И (№ 154) совпадает, т. е. первое условие выполняется. В качестве t и s возьмем, например, кортежи

t [Давыдов В. А. № 154 математика],

s [Жукова А. М. № 154 физика].

Рассмотрим, существует ли такой кортеж u, для которого будет выполняться и второе условие, определяющее многозначную зависимость.

Можно выделить кортежи, для которых выполняется соотношение u[№ 154, математика] =t [№ 154, математика]; это следующие кортежи:

t [Давыдов В. А., № 154, математика],

u [Жукова А. М., № 154, математика],

т. е. первому уравнению во втором условии — кортежи удовлетворяют. Анализируя таблицу, заметим, что для того, чтобы существовала многозначная зависимость ШКОЛА №→→ПРЕДМЕТ, должно выполняться условие

u [Жукова А. М.]=*s* [Жукова А. М.],

что действительно имеет место.

Другой способ проверки многозначной зависимости $X \rightarrow\rightarrow Y$ на отношение *R(X, Y, Z)* может быть осуществлен на основе проверки существования кортежа $<x, y, z>$ при условии, что существуют кортежи $<x, y, z'>$ и $<x, y', z>$.

Следует иметь в виду, что в общем случае формальная проверка должна выполняться на множестве всех возможных экземпляров кортежей отношения.

Правила вывода многозначных зависимостей сходны с правилами вывода функциональных зависимостей:

МЗ1 (дополнение). Если $X \subseteq U$, $Y \subseteq U$, $X \rightarrow\rightarrow Y$, то имеет место многозначная зависимость $X \rightarrow\rightarrow U - X - Y$.

МЗ2 (присоединение). Если $X \subseteq U$, $Y \subseteq U$, $Z \subseteq W$, $X \rightarrow\rightarrow Y$, то имеет место зависимость $X\,W \rightarrow\rightarrow YZ$.

МЗ3 (транзитивность). Если $X \subseteq U$, $Y \subseteq U$, $X \rightarrow\rightarrow Y$, $Y \rightarrow\rightarrow Z$, то $X \rightarrow\rightarrow Z - Y$.

Существуют также правила вывода для совокупности ФЗ и МЗ:

ФМ1. Если $X \subseteq U$, $Y \subseteq U$, $X \rightarrow Y$, то $X \rightarrow\rightarrow Y$.

ФМ2. Если $X \subseteq U$, $Y \subseteq U$, $Z \subseteq U$, $W \subseteq U$ и W не пересекается с Y (т. е. $W \cap Y = 0$), и $X \rightarrow\rightarrow Y$, $W \rightarrow Z$, то имеет место зависимость $X \rightarrow\rightarrow Z$.

ФМ3. Если $X \rightarrow\rightarrow Y$, $XY \rightarrow Z$, то $X \rightarrow Z - Y$.

Функциональная и многозначная зависимости являются свойствами, существующими между двумя атрибутами (множествами). С их помощью осуществляют декомпозиции (разбиение) отношения на два или восстанавливают исходное отношение, соединяя два отношения (рис. 5.59).

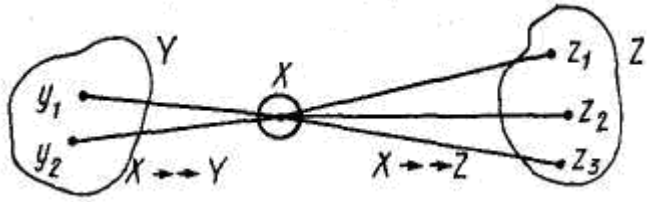


Рис. 5.59. Пример изображения многозначных зависимостей

Одним из основных вопросов любой организации данных является вопрос сохранения жизнеспособности прикладных программ при изменениях в базе данных для того, чтобы избежать трудностей поддержания базы данных и назначения ключей для каждого отношения.

Рассмотрим отношение $R\{A_1, A_2, \ldots, A_n\}$. Возможный ключ k отношения R — это комбинация атрибутов (возможно, состоящих из одного атрибута), обладающих следующими свойствами.

1. В каждом кортеже отношения R величина k единственным образом определяет этот кортеж.

2. Не существует атрибута в возможном ключе k, который мог бы быть удален без нарушения свойства 1.

Всегда существует, по крайней мере, один возможный ключ, т. е. комбинация всех атрибутов R удовлетворяет свойству 1.

Если в отношении R имеется несколько возможных ключей, то один из них выбирается в качестве первичного.

Атрибут A_i отношения R называется также первичным, если он входит в состав любого ключа (возможного или первичного) отношения.

Нормальные формы схем отношений. Рассмотрим четыре уровня нормализации схем отношений и соответственно четыре нормальные формы отношений: 1НФ, 2НФ, 3НФ, 4НФ. Их взаимное отношение можно представить в виде, представленном на рис. 5.60.

Из рис. 5.60 следует, что отношения являются как бы вложенными друг в друга по возрастанию номеров.

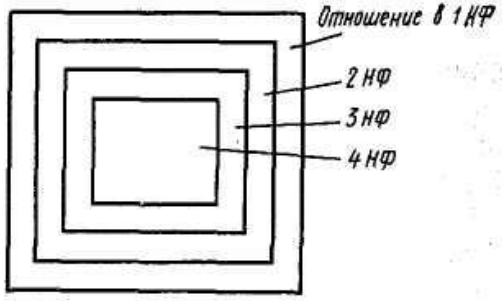


Рис. 5.60. Взаимное отношение нормальных форм

Так, например, если отношение находится в 4НФ, то оно будет удовлетворять и условиям 3НФ, 2НФ, 1НФ.

Отношение находится в **первой нормальной форме** (1НФ), если каждый атрибут отношения является простым (***атомарным***) атрибутом, т. е. отсутствуют составные. Рассмотрим схему отношения:

АВТОМОБИЛЬ (МОДЕЛЬ, МАРКА, МОЩНОСТЬ, СТОИМОСТЬ, ИЗГОТОВИТЕЛЬ (НАЗВАНИЕ ЗАВОДА, ГОРОД)).

В данном случае атрибут ИЗГОТОВИТЕЛЬ составной. Для приведения к 1НФ отношения необходимо избавиться от составного отношения — ИЗГОТОВИТЕЛЬ. Этого можно добиться, рассматривая вместо составного атрибута его составляющие:

АВТОМОБИЛЬ (МОДЕЛЬ, МАРКА МОЩНОСТЬ, СТОИМОСТЬ, НАЗВАНИЕ ЗАВОДА ИЗГОТОВИТЕЛЯ, ГОРОД).

Приведение отношения к 1НФ достаточно для реализации языков запросов.

Чтобы рассмотреть 2НФ, введем понятие полной зависимости. Пусть X и Y — подмножества атрибутов отношения R и $X \rightarrow Y$. Если Y функционально не зависит от любого подмножества A множества X (причем A не совпадает с X), то Y называется полностью зависимым от X в R.

Тогда говорят, что отношение R находится во второй нормальной форме (2НФ), если оно нормализовано, т. е. находится в первой нормальной форме, и каждый непервичный атрибут полностью зависит от первичного ключа.

Отношение R находится в третьей нормальной форме (3НФ), если оно находится во второй нормальной форме и каждый непервичный атрибут в отношении R не содержит транзитивных зависимостей от первичного ключа.

Транзитивная зависимость наблюдается в R, если существует такой атрибут A, что $X \rightarrow Y$ и $Y \rightarrow A$, $Y \leftrightarrow X$, где X — ключ.

Можно заметить, что 2НФ и 3НФ накладывают ограничения на зависимости только в связи с непервичными атрибутами. Поэтому в рассмотрение вводят усиленную третью нормальную форму (или нормальную форму Бойса-Кодда).

Отношение R находится в усиленной третьей нормальной форме, если для всех зависимостей $X \rightarrow A$, когда A не принадлежит X, X является возможным ключом отношения R. Обычно атрибут, от которого функционально полно зависит другой, называют ***детерминантой.*** Поэтому говорят, что ***отношение R находится в усиленной третьей нормализованной форме, если все <u>детерминанты</u> являются <u>ключами</u>.***

Можно отметить следующее отличие этих нормальных форм по информационному содержанию. Вторая нормальная форма более информативна, чем первая, а третья более информативна, чем вторая.

Поэтому 3НФ и усиленная 3НФ более понятны пользователю и приспособлены к реализации.

Существует также нормальная форма, учитывающая многозначные зависимости, ее называют четвертой нормальной формой (4НФ). Отношение R находится в четвертой нормальной форме, если всякий раз, когда существует многозначная зависимость $X \rightarrow\rightarrow Y$ (где $Y \neq \emptyset$, $Y \not\subset X$ и XY состоит не из всех атрибутов R), также существует зависимость $X \rightarrow A$ для любого атрибута A в R.

Получение отношений нормальной формы достигается декомпозицией их схем. Под декомпозицией схемы отношения $R=\{A_1, A_2,..., A_n\}$ понимается замена схемы совокупностью отдельных схем $\rho=\{R_1, R_2,...,R_k\}$ таких, что $R_1 \bigcup R_2 \bigcup ... \bigcup R_n=R=\{A_1, A_2,..., A_n\}$. При этом не накладывается никаких ограничений на пересечение любых двух элементов совокупности ρ.

Для получения нужных данных из базы разрабатываются специальные языки манипулирования данными, обеспечивающие выполнение необходимых операций. Для разработки и исследования языков манипулирования данными используют математический аппарат, основанный на следующих трех подходах:

- реляционная алгебра;
- реляционное исчисление с переменными-кортежами;
- реляционное исчисление с переменными-атрибутами.

Реляционная алгебра. В ней определяются основные операции над данными реляционного типа. Все операции, вводимые в реляционной алгебре, можно разделить на традиционные над множествами и специализированные, вводимые для удобства поиска в БД.

К операциям первой группы относятся: объединения, пересечение, разность, декартово произведение. К операциям второй группы следует отнести: проекцию, ограничение, соединение, деление.

Объединение. В результате применения этой операции получается отношение, объединяющее кортежи, содержащиеся в исходных отношениях. Пусть имеем два исходных отношения R_1 и R_2. Операция объединения этих отношений будет обозначаться $R_1 \bigcup R_2$:

$$R_1 \bigcup R_2 = \{r \mid r \in R_1 \text{ или } r \in R_2\}.$$

Объединяемые отношения должны иметь одинаковые атрибуты (должны быть объединимыми).

Пример.

R_1:

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

R_2:

A	B	C
a_3	b_4	c_2
a_4	b_5	c_3
a_2	b_2	c_2

$R_1 \cup R_2$

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3
a_3	b_4	c_2
a_4	b_5	c_3
a_2	b_2	c_2

Пересечение. В данной операции (обозначаемой $\cap$) получают отношение, включающее кортежи, общие для R_1 и R_2:

$$R_1 \cap R_2 = \{r \mid r \in R_1 \text{ и } r \in R_2\}.$$

Пример. Для отношения R_1 и R_2из предыдущего примера получим

$R_1 \cap R_2$:

A	B	C
a_2	b_2	c_2

Разность. В результате применения этой операции $(R_1 \backslash R_2)$ получается отношение, содержащее кортежи, являющиеся кортежами отношения R_1 и не являющиеся кортежами отношения R_2:

$$R_1 \setminus R_2 = \{r \mid r \in R_1 \text{ и } r \notin R_2\}.$$

Пример.

$R_1 \setminus R_2$:

A	B	C
a_1	b_1	c_1
a_3	b_3	c_3

Декартово (прямое) произведение. В этой операции $(R_1 \times R_2)$ из n-местного отношения R_1 и n-местного отношения R_2 получают $(m+n)$ -

местное отношение. Причем первые m элементов представляют кортежи из отношения R_1, а последние n элементов — кортежи из отношения R_2:

$$R_1 \times R_2 = \{\langle r_1, r_2, \rangle \mid r_1 \notin R_1 \text{ и } r_2 \in R_2\}.$$

Пример.

$R_1 \times R_2$:

A	B	C	A	B	C
a_1	b_1	c_1	a_3	b_4	c_2
a_2	b_2	c_2	a_3	b_4	c_2
a_3	b_3	c_3	a_3	b_4	c_2
a_1	b_1	c_1	a_4	b_5	c_3
a_2	b_2	c_2	a_4	b_5	c_3
a_3	b_3	c_3	a_4	b_5	c_3
a_1	b_1	c_1	a_2	b_2	c_2
a_2	b_2	c_2	a_2	b_2	c_2
a_3	b_3	c_3	a_2	b_2	c_2

Проекция. Операция проекции предназначена для изменения числа столбцов в отношении, т. е. в том случае, когда из строк-кортежей требуется исключить какие-либо атрибуты.

Обозначим через $j_1, j_2, \ldots, j_n$ — номера столбцов n-местного отношения R. Операцию определения проекции отношения R обозначим через $\pi_{j_1, j_2, \ldots, j_n}(R)$, а сама операция заключается в том, что из отношения R выбираются столбцы и компонуются в указанном порядке $j_1, j_2, \ldots, j_n$.

Пример.
Рассмотрим отношение ТЕЛЕВИЗОР.

Наименование	Индекс	Диаметр кинескопа, см	Цена, руб.
Рекорд	ВЦ-311	47	640
Рубин	Ц-266Д	67	1040
Темп	Ц-280Д	61	755
Электроника	Ц-283	61	755
Горизонт	Ц-355	57	610

Тогда можно получить проекцию $\pi_{1,4}$ (ТЕЛЕВИЗОР):

Наименование	Цена, руб.
Рекорд	640
Рубин	1040
Темп	755
Електроника	755
Горизонт	610

$\pi_{4,1,3}$ (ТЕЛЕВІЗОР):

Цена, руб.	Наименование	Диагональ кинескопа, см
640	Рекорд	47
1040	Рубин	67
755	Темп	61
755	Електроника	61
610	Горизонт	51

π_4 (ТЕЛЕВИЗОР):

Цена, руб.
640
1040
755
755
610

Можно отметить, что в последней проекции оказались одинаковые строки.

Ограничение. Ограничением называют такую операцию, в которой отношение исследуют по строкам и выделяют множество строк, удовлетворяющих заданным условиям.

Обозначим через r строку (кортеж) отношения R, а через θ определим одно из отношений: $=, \neq, <, \leq, >, \geq$. Тогда θ-ограничение между выбранным атрибутом A в отношении R и некоторой величиной C определяют следующим образом: $R[A\theta C]=\{r|r\in R$ и $r[A]\theta C\}$, где $r[A]$ соответствует значению атрибута A в строке r.

Таким образом, θ-ограничение обеспечивает получение среди строк отношения *R* только тех строк, в которых значение атрибута *A* и значение величины *C* удовлетворяют условию сравнения θ.

Например, требуется из отношения ТЕЛЕВИЗОР выделить те марки телевизоров, которые имеют стоимость меньше 700 руб.

ТЕЛЕВИЗОР [цена < 700]: Рекорд ВЦ-311 47 640
Горизонт Ц-355 51 610

Следует отметить, что в частном случае в операции сравнения вместо величины *C* можно использовать другой домен *B*. Тогда операция ограничения определяется как

$$R(A\theta B) = \{r \mid r \in R \text{ и } r(A)\theta r(B)\}$$

Пример. Пусть отношение *R* имеет вид:

$$R: \quad \begin{matrix} (A, & B, & D, & E) \end{matrix}$$
$$\begin{bmatrix} a & 15 & 17 & p \\ k & 22 & 28 & q \\ e & 39 & 11 & e \\ m & 16 & 43 & m \end{bmatrix}$$

Так как операция θ-ограничения предусматривает выбор среди строк отношения *R* только тех, в которых значения атрибутов *A* и *B* удовлетворяют условию сравнения θ, то для условий (*B*<*D*) и *(A=E)* получим следующие отношения:

$$R[B < D]: \quad \begin{matrix} A & B & D & E \end{matrix}$$
$$\begin{bmatrix} a & 15 & 17 & p \\ k & 22 & 28 & q \\ \boldsymbol{m} & 16 & 43 & m \end{bmatrix}$$

$$R[A = E]: \quad \begin{matrix} A & B & D & E \end{matrix}$$
$$\begin{bmatrix} e & 39 & 11 & e \\ \boldsymbol{m} & 16 & 43 & m \end{bmatrix}$$

Соединение. Операция соединения обратна операции проекции. Рассмотрим для простоты два бинарных отношения $R_1(A, B)$ и $R_2(B, C)$. Соединением (обозначается $\bowtie$) отношений R_1 и R_2 $(R_1 \bowtie R_2)$ называют операцию, при которой соединяют два отношения, используя в качестве признака соединения общий атрибут *Y:*

$$R_1 \bowtie R_2 = \{\langle A, B, C\rangle \mid \langle A, B,\rangle \in R_1 \text{ и } \langle B, C,\rangle \in R_2\}.$$

Отношение $R_1 \bowtie R_2$ является отношением с атрибутами <*A, B, C*>.

Например:

R_1: Наименованние	Индекс	R_2: Индекс	Цена
Рекорд	ВЦ-311	ВЦ-311	640
Темп	Ц-280Д	Ц-280Д	755
Електроника	Ц-283	Ц-283	755
Горизонт	Ц-355	Ц-355	610

Отношение $R_1 \rhd\lhd R_2$ будет иметь вид:

Наименование	Индекс	Цена
Рекорд	ВЦ-311	640
Темп	Ц-280	755
Электроника	Ц-283	755
Горизонт	Ц-355	610

Операция соединения соответствует случаю, когда просто стыкуются таблицы отношений. Но поскольку в получающейся таблице атрибуты с одинаковым содержанием присутствуют дважды, один из одинаковых столбцов исключают. Такую операцию называют естественным соединением. Операцию соединения можно использовать не только для бинарных, но и для *n*-местных отношений: $R^1 \rhd\lhd R_2 \rhd\lhd ,..., \rhd\lhd R_n = \{M | M$ — кортежи, образованные соединением атрибутов отношений $R_1, R_2,..., R_n\}$.

Отметим, что рассматривались в основном операции соединения фактически по условию равенства двух атрибутов в двух отношениях. В общем случае соединение можно осуществлять не только по условию равенства, но и по любому другому θ: =, ≠, <, ≤, >, ≥:

$$R_1 \underset{A\theta B}{\rhd\lhd} R_2 = \{(r,\ s) | r \in R_1 \text{ и } s \in R_2 \text{ и } (r(A)\theta r(B))\},$$

когда θ — оператор равенства, то операцию называют ***эквисоединением.***

Если сравнение θ возможно, то для атрибутов *A* и *B* одинаковые имена необязательны.

Пример. Пусть имеются отношения $R_1(A, B, C)$ и $R_2(D, E, F)$

R_1:

A	B	C
8	4	
2	7	
5	5	
8	3	

R_2:

D	E	F
d_1	8	3
d_2	8	2
d_3	8	10
d_4	5	12

Для условий $(C=E)$, $(B>F)$ получим

$R_1 \underset{C=E}{\bowtie} R_2$:

A	B	C	D	E	F
a_2	5	8	d_1	8	3
a_2	5	8	d_2	8	2
a_2	7	8	d_1	8	3
a_2	7	8	d_2	8	2
a_3	2	5	d_4	5	12

$R_1 \underset{B>F}{\bowtie} R_2$:

A	B	C	D	E	F
a_1	3	4	d_2	8	2
a_2	5	8	d_1	8	3
a_2	5	8	d_2	8	2
a_2	7	8	d_1	8	3
a_2	7	8	d_2	8	2

Таким образом, с помощью операции соединения можно объединять строки различных отношений по критерию сравнения значений каких-нибудь двух атрибутов.

Деление. Рассмотрим деление *m*-местного отношения R_1 на *n*-местное отношение R_2.

Пусть из общего количества *m* атрибутов отношения R_1 выделим несколько атрибутов: *A, B,...,F* и из них составляем список, обозначив его через *M.* Набор значений атрибутов из *M* столбцов можно рассматривать как проекцию отношения *R* на список атрибутов *M,* т. е. $\pi_M(R_1)$. Тогда через $\bar{M}$ будут обозначаться атрибуты дополнительные к *M,* т. е. атрибуты отношения R_1, не вошедшие в список *M,* и соответственно значения атрибутов из списка $\bar{M}$ определяются $\pi_{\bar{M}}(R_1)$. Будем полагать, что делимое (отношение R_1) может быть представлено таким образом, что его атрибуты сгруппированы в порядке $R_1(\bar{M}, M)$. В отношении R_2 также выделим несколько атрибутов *G, K,...,P* и составим из них список, обозначив его через *N* (формально этот список представляет собой проекцию $\pi_N(R_2)$). Если проекции $\pi_N(R_1)$ и $\pi_N(R_2)$ объединимы, т. е. имеют одинаковое количество атрибутов, то можно рассматривать операцию деления R_1 по *M* на R_1 по *N* (что обозначается как $R_1[M \div N]R_2$). Операция деления

$R_1[M \div N]R_2$ представляет собой операцию, которая определяет такое наибольшее множество значений атрибутов из $\pi_{\bar{M}}(R_1)$ (обозначим его через $r_1(\bar{M})$), что прямое произведение этого множества $r_1(\bar{M})$, с $\pi_N(R_2)$ содержится в R_1.

Операции деления можно определить с помощью уже ранее введенных операций следующим образом:

$$R_1[M \div N]R_2 = \pi_{\bar{M}}(R_1) \setminus \pi_{\bar{M}}((\pi_{\bar{M}}(R_1) \times \pi_N(R_2)) \setminus R_1),$$

где $\pi_{\bar{M}}$ —это проекция отношения на атрибуты списка $\bar{M}$.

Пример.

R_1:

A	B	C	D
a_1	b_1	c_3	d_1
a_2	b_1	c_1	d_1
a_3	b_2	c_2	d_2
a_1	b_3	c_1	d_3
a_3	b_2	c_1	d_1
a_3	b_1	c_2	d_2

R_2:

C	D
c_1	d_1
c_2	d_2

Включим в список матрицы M атрибуты C и D (из отношения R_1), тогда в список $\bar{M}$ войдут атрибуты A и B, а в список N — атрибуты C и D из отношения R_2:

$$R_1[(C, D) \div (C, D)]R_2$$

A	B
a_3	b_2

Действительно,

$\pi_{\overline{M}}(R_1)$:

A	B
a_1	b_1
a_2	b_1
a_3	b_2
a_1	b_3
a_3	b_1

$\pi_{\overline{M}}(R_1) \times \pi_N(R_2)$:

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_1	c_2	d_2
a_2	b_1	c_1	d_1
a_2	b_1	c_2	d_2
a_3	b_2	c_1	d_1
a_3	b_2	c_2	d_2
a_1	b_3	c_1	d_1
a_1	b_3	c_2	d_2
a_3	b_1	c_1	d_1
a_3	b_1	c_2	d_2

$\pi_{\overline{M}}(R_1) \times \pi_N(R_2) \setminus R_1$:

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_1	c_2	d_2
a_2	b_1	c_2	d_2
a_1	b_3	c_1	d_1
a_1	b_3	c_2	d_2
a_3	b_1	c_1	d_1

$$\pi_{\overline{M}}\left(\left(\pi_{\overline{M}}(R_1)\times\pi_N(R_2)\right)\setminus R_1\right):$$

A	B
a_1	b_1
a_2	b_1
a_1	b_3
a_3	b_1

$$R_1[(C,\ D)\div(C,\ D)]\ R_2=\pi_{\overline{M}}(R_1)\setminus\pi_{\overline{M}}\left(\left(\pi_{\overline{M}}(R_1)\times\pi_N(R_2)\right)\setminus R_1\right):$$

A	B
a_3	b_2

При построении языка манипулирования данными на основе реляционной алгебры каждый оператор языка реализует некоторый набор алгебраических операций, в результате выполнения которых получается желаемое выходное отношение.

Другой подход к построению языка манипулирования данными основан на использовании моделей реляционного исчисления.

Суть этого подхода заключается в том, что желаемый результат определяется не заданием набора операций над отношениями, а путем описания требований, которым должно удовлетворять результирующее отношение. СУБД представляется самой подобрать последовательность операций, ведущих к поставленной цели. Естественно, что языки, основанные на использовании моделей реляционного исчисления, представляют собой языки очень высокого уровня.

Реляционное исчисление. В реляционном исчислении, так же как и в реляционной алгебре, имеется набор понятий и операций, которые позволяют записывать любое отношение в виде некоторой формулы или формального выражения (α-выражения).

Формулы в реляционном исчислении помимо арифметических операций (= , ≠, <, ≤, >, ≥) включают дополнительные **логические** операции. К ним относят операции квантификации:

$\forall$ — квантор общности и $\exists$ — квантор существования.

Квантор общности $\forall$ читается как: «все для всех», «для каждого», «каков бы ни был».

Например, запись $\forall x$ читается «для любого x» или «для всех x».

Квантор существования $\exists$ читается «для некоторого», «существует хотя бы один». Например, $\exists y$ читается «для некоторого y» или «существует y такой, что». Кроме кванторов в реляционном исчислении используются логические операции: $\vee$, $\wedge$, $\neg$.

Операция $\vee$ носит название **логическое сложение,** или **дизъюнкция,** и по смыслу соответствует слову «ИЛИ». Операция $\wedge$ **логическое умножение, или конъюнкция,** и соответствует слову «И». Операция $\neg$ — операция **отрицания**, соответствует «НЕ». Запись $R_1 \wedge R_2$ будет читаться как отношение R_1 **И** отношение R_2, запись $R_1 \vee R_2$ — «отношение R_1 **ИЛИ** отношение R_2».

При записи выражений в реляционном исчислении используется понятие **свободных или связных переменных.**

Вхождение переменных x в формулу реляционного исчисления $\psi(x)$ связано, если в ψ она находится в части формулы, начинающейся квантором $\forall$ или $\exists$, за которым непосредственно следует переменная x. В таких случаях говорят, что квантор связывает переменную x. В остальных случаях вхождение переменной x в формулу ψ свободно.

Например, в формуле

$$\forall x(R_1(x, y) \vee (\exists y) R_2(x, y, z))$$

переменная x связана, переменная y в отношении R_1 свободна, а в R_2 — связана, переменная z свободна.

Формулы в реляционном исчислении строятся из ***атомов и совокупности арифметических и логических операторов***. Атомы в реляционном исчислении представляются по-разному. В зависимости от того, что используется в качестве переменной — кортеж (строка) или атрибут (столбец). Поэтому различают реляционное исчисление с переменными-кортежами и переменными-доменами.

Реляционное исчисление с переменными-кортежами. Выражение такого исчисления может иметь следующий вид:

$$\{r \mid \psi(r)\},$$

где r —кортеж: ψ — некоторая формула исчисления. Например, выражение $\{r|R_1(r) \wedge R_2(r)\}$, где в качестве формулы $\psi(r)$ используется выражение $R_1(r) \wedge R_2(r)$, означает, что необходимо получить множество всех кортежей r, таких, что они принадлежат одновременно как отношению R_1, так и отношению R_2. В том случае, когда в качестве переменных в реляционном исчислении используются кортежи атомы, из которых конструируются формулы, они могут быть следующих типов:

1. Атом — отношение $R(r)$, где r — кортеж в отношении R.
2. Атом — конструкция типа $s[i]\theta v[j]$ или $s[i]\theta c$, где c — некоторая константа; θ — арифметический оператор $(=, \neq, <, \leq, >, \geq)$;

s, v — кортежи; i, j — номера (или имена) атрибутов (столбцов) в соответствующих кортежах. Например, атом $(s[1] = q[7])$ означает, что первая компонента кортежа s равна седьмому кортежа q, а атом

s[5]<10 означает, что пятая компонента меньше 10. Два перечисленных выше типа вида атомов являются единственными в реляционном исчислении.

Сами формулы рекурсивно конструируются из атомов по следующим правилам.

1. ***Любой атом—это формула***. Все вхождения переменных-кортежей, упомянутые в атоме, являются свободными.
2. Если ψ формула, то $\neg\psi$ — тоже формула ($\neg$—символ логического отрицания).
3. Если ψ_1 и ψ_2 формулы, то и выражения $\psi_1 \wedge \psi_2$ и $\psi_1 \vee \psi_2$ также являются формулами. Причем свободными (связными) являются те и только те вхождения переменных, которые происходят от свободных (связных) вхождений переменных ψ_1 и ψ_2.
4. Если ψ формула и r — свободная переменная-кортеж этой формулы, то $\forall r(\psi))$ и $\exists r(\psi)$ также формулы, переменная в этом случае становится связанной.
5. Формулы могут при необходимости заключаться в скобки.
6. ***Ничто иное не является формулой***.

Реляционное исчисление с переменными-доменами. В этом случае в качестве переменных вместо кортежей используются домены. Реляционное исчисление с переменными на доменах строится с использованием тех же операторов, что и с переменными кортежами. Но в качестве атомов формул исчисления используются следующие типы:

1. $R(x_1, x_2, ..., x_n)$, где R — n-арное отношение, x_i — константа или переменная на некотором домене.

Атом $R(x_1, x_2, ..., x_n)$ указывает, что значения x_i, которые являются переменными, должны быть выбраны так, чтобы $(x_1, x_2, ..., x_n)$ было кортежем отношения.

2. $x\theta y$, где x и y — константы или переменные на некотором домене, θ — арифметический оператор сравнения.

В остальном формулы реляционного исчисления с переменными-доменами строятся аналогично формулам исчисления с переменными-кортежами.

Выражение реляционного исчисления с переменными на доменах имеет вид $\{x_1, x_2.....x_n|\psi(x_1, x_2, ..., x_n)\}$, где ψ — формула, обладающая тем свойством, что только ее свободные переменные на доменах являются различными переменными $x_1, x_2, ..., x_n$.

Выражения реляционного исчисления

$$\{r|\psi(r)\} \text{ или } \{x_2, x_2, ..., x_n|\psi(x_1, x_2, ..., x_n)\}$$

служат основой реальных языков манипулирования данными.

В реляционном исчислении доказано, что для любого простого выражения исчисления существует эквивалентное ему выражение реляционной алгебры. Поэтому может быть построена универсальная процедура для перевода выражений реляционного исчисления в эквивалентное по смыслу алгебраическое выражение.

5.9.5. Сетевая модель данных

В основе разработки сетевых моделей данных лежит возможность представления связей между данными в графической форме.

Наиболее развитой сетевой моделью является модель, предложенная Рабочей группой по базам данных (РГБД) Ассоциации по языкам обработки данных (КОДАСИЛ). Нужно отметить, что в основе модели КОДАСИЛ лежат понятия «сущность» и «связь», а к основным типам структур модели относят: элемент данных, агрегат, запись, набор.

Сущность — это собирательное понятие, некоторая абстракция реально существующего объекта предметной области, процесса или явления. Набор однородных объектов или явлений определяет **тип** сущности, а каждый конкретный объект в наборе представляет **экземпляр** сущности. Связи между сущностями фиксируются заданием множества отношений. При анализе связей между сущностями наиболее часто используются бинарные связи, т. е. связи между двумя сущностями. По характеру бинарные связи между типами сущностей различают:

- один к одному (1:1);
- один ко многим *(*1*:М);*
- многие к одному *(М* : 1);
- многие ко многим *(М:М).*

Элемент данных — это наименьшая единица данных, которой можно оперировать в БД и выполнять построение всех остальных структур. Можно отметить, что элемент данных представляет собой аналог поля в файловых системах. Элемент данных имеет имя, которое хранится в БД как часть описания базы. Именами элементов данных могут быть, например, ИНДЕКС ИЗДЕЛИЯ, ДАТА ВЫПУСКА, СТОИМОСТЬ и т. д. В сетевых моделях элементы данных используются для представления атрибутов сущности.

Агрегат данных — совокупность элементов данных, имеющих общее имя, которую можно рассматривать как единое целое. Например, агрегат данных ДАТА состоит из элементов данных: ЧИСЛО, МЕСЯЦ, ГОД.

Запись — совокупность элементов данных, которые описывают конкретный экземпляр объекта (сущности). Предположим, что сущность ТЕЛЕВИЗОР описывается элементами данных: МАРКА; ИНДЕКС, ЦЕНА. Тогда запись в этом объекте для конкретного изделия может быть: РЕКОРД, ВЦ-311, 640. Можно отметить, что запись эквивалентна кортежу в реляционных моделях данных.

Сетевая модель РГБД КОДАСИЛ в качестве базовых использует понятия «экземпляр» и «набор».

Тип — это общее понятие, представляющее собой собрание экземпляров записи. Каждый тип записи состоит из некоторого числа элементов данных, значения которых размещаются в экземплярах записи данного типа. В качестве связей между типами записей используются наборы. Каждый набор представляет собой отношение (связь) между двумя или несколькими типами записей. Он отображает множество связей между экземплярами записей типа «владелец» и «член». Для каждого типа набора один тип записи может быть объявлен «владельцем», а остальные — его «члены». При этом любой экземпляр записи типа «член» может быть связан не более чем с одним экземпляром типа «владелец».

Графическая интерпретация сетевой модели данных представляет собой ориентированный граф без петель. Причем, вершинам графа соответствуют типы записей, а дугам — наборы, отражающие связи между соответствующими типами записей. Направленные стрелки на дуге ориентированы от записи типа «владелец» к записи типа «член».

Подмножество дуг, соединяющих одну запись — владельца с несколькими записями— членами, называется экземпляром набора.

Рассмотрим, например, граф, отражающий упрощенную БД комплектующих деталей телевизора (рис. 5.61).

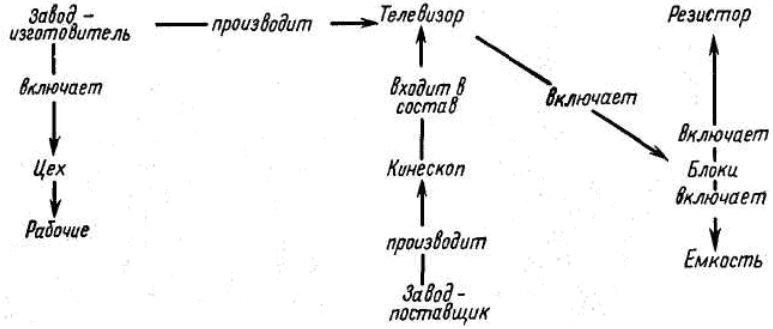


Рис. 5.61. Пример построения базы данных комплектующих телевизора

Стрелки между вершинами соответствуют наборам данных, отражающих связи между записями, а надписи над стрелками — именам наборов.

Как тип записи, так и набор данных в общем случае могут быть представлены таблицами. Но в отличие от таблиц реляционных моделей, в сетевых моделях данных они могут допускать дубликаты строк или записей.

В модели КОДАСИЛ вводится особый тип набора данных, называемый ***сингулярным*** и имеющий только один экземпляр набора этого типа. В нем запись «владелец» отсутствует (владельцем является система управления базой данных). Этот тип набора обычно используется для создания традиционного файла, состоящего из однотипных записей.

В сетевой модели данных КОДАСИЛ имеются несколько ограничений, которые надо учитывать при построении модели. Основным внутренним ограничением являются функциональность связей, так как нельзя реализовать связи типа *M:M*. В модели это ограничение соответствует положению: в конкретном экземпляре набора экземпляр записи числа может иметь не более одного экземпляра записи владельца.

Для того чтобы отобразить принятую схему данных в памяти ЭВМ, требуется описать все таблицы, соответствующие записям и наборам. Для этого группой КОДАСИЛ был предложен язык описания данных, позволяющий задать схему данных с помощью четырех типов статей.

Статья схемы задает имя схемы БД. Статья запишется как:

SCHEMA NAME IS имя схемы.

Статья области характеризует область памяти, в которой размещаются экземпляры записей БД. С помощью этой статьи можно в случае необходимости распределять БД по различным ЗУ. Поскольку в общем случае можно выделить под БД несколько различных областей, каждая из них должна иметь собственное уникальное имя и описываться следующим образом:

AREA NAME IS имя области;

Статья записи содержит описание типа записи, включающее имя записи и характеризующее все элементы данных, входящие в ее состав. Каждому типу записи соответствует своя статья. Статья записи начинается предложением:

RECORD NAME IS имя записи;

Следующий за этим предложением текст зависит от варианта реализованного языка КОДАСИЛ. Наиболее распространенными являются варианты ЯОД КОДАСИЛ. Поскольку ряд действующих

СУБД реализуют вариант ЯОД-73, далее будем рассматривать случай, когда статья записи формируется на его основе. Тогда вторым предложением в схеме записи будет идти предложение:

LOCATION MODE IS { DIRECT | CALC (имя процедуры) USING (имя calc-элемента) | DUPLICATES ARE (NOT) ALLOWED | VIA имя набора SET | SYSTEM }

где в фигурных скобках указано одно из возможных описаний.

DIRECT используется в том случае, когда предполагается, что номер страницы области для размещения записи будет определяться программой, выполняющей включение записи в БД.

CALC применяется, когда предполагается, что специальная программа будет использовать значение ключа БД.

Ключ базы данных — это идентификатор, уникально определяющий запись, помещенную в БД.

Для того, чтобы можно было различать отдельные экземпляры записей, хранящихся в БД, каждому экземпляру записи присваивается значение ключа, который играет роль внутрисистемного индетификатора.

Зная значение ключа, можно быстро отыскать соответствующую запись. В формате CALC USING имя calc-элемента, в качестве имен calc-элемента используются имена элементов данных записи.

В том случае если ключи не имеют дубликатов значений, то в варианте CALC следует указать DUPLICATES ARE NOT ALLOWED. Если ключ имеет дубликаты значений, например в качестве ключа задан элемент данных ФАМИЛИЯ в записи типа СТУДЕНТ, то следует указать DUPLICATES ARE ALLOWED.

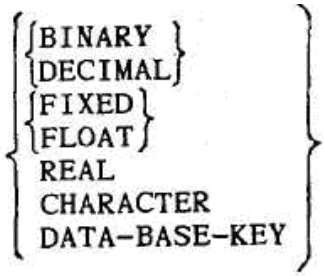
VIA SET используется в том случае, когда требуется запись разместить физически как можно ближе к соответствующему экземпляру набора, в который она будет включена.

SYSTEM применяется в случае, если размещение записей возлагается на саму СУБД (в соответствии с заложенными в нее алгоритмами).

Для того чтобы приписать рассматриваемый тип записи к некоторой области, используется предложение

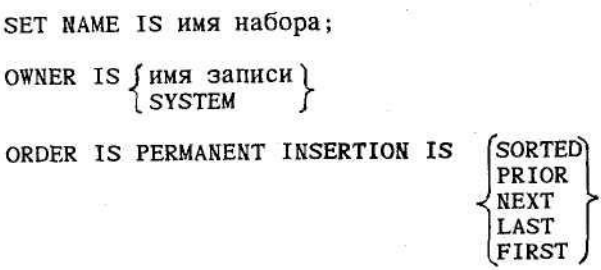
WITHIN имя области AREA

Для описания внутренней структуры записи в статью записи включается подсистема данных, имеющая вид



С помощью этой подсистемы каждому элементу данных приписывается тип значений данных: двоичное, десятичное, фиксированное, плавающее, натуральное, символьное, ключ БД.

Статья набора позволяет описать наборы БД. Формат задания набора имеет вид



Первое предложение статьи задает имя описываемого набора. Второе указывает имя типа записи, являющейся записью владельца набора. И последнее — на способ включения экземпляров записей— членов в экземпляры описываемого типа набора.

Обычно предполагается, что на внутреннем уровне экземпляры набора, включающие несколько членов записей, организованы в виде цепочки.

Цепочка записей реализуется с помощью специального служебного элемента — **указателя,** который содержит (указывает на) адрес записи, логически связанной с рассматриваемой. Посредством указателей можно организовать обращение не только к последующим записям, но и к предыдущим.

Команды, входящие в фигурные скобки третьего предложения схемы, как раз и позволяют организовать цепочку требуемым образом и означают:

FIRST — запись включается первой в цепочку перебора записей, т. е. сразу же после записи владельца (рис. 5.62).

LAST — запись включается последней (рис. 5.62).

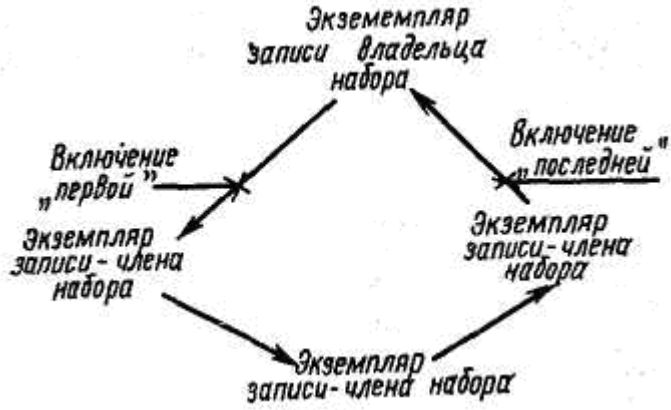


Рис. 5.62. Реализация вариантов FIRST и LAST

NEXT — включение следующей записи, т. е. записи, следующей за текущей записью набора (рис. 5.63).

Рис. 5.63. Реализация вариантов NEXT і PRIOR

Под текущей записью набора понимается тот экземпляр записи конкретного экземпляра набора, на который указывает «текущая» запись, в типе набора.

PRIOR — включение предыдущей записи, т. е. перед текущей записью набора (рис. 5.63).

SORTED — СУБД поддерживает упорядоченность экземпляров однотипных записей, входящих в экземпляр набора, в соответствии в возрастанием и убыванием ключевого данного.

Описание числа набора проводится с помощью подстатьи члена набора, имеющей вид

В зависимости от способа включения экземпляра записи-члена в экземпляр набора различают два типа членства в наборе:

AUTOMATIC — автоматический;

MANUAL — ручной.

Чтобы поместить в БД новую запись, необходимо сформировать ее в рабочей области программ и только затем с помощью операторов языка манипулирования данными записать в БД. Если необходимо экземпляр записи-члена поместить в БД, но при этом не соединять ни с каким экземпляром набора, то указывают оператор MANUAL. Последующее включение этой записи в набор должно быть осуществлено программистом явным образом с помощью соответствующих средств языка манипулирования данными. Тип членства AUTOMATIC предусматривает, что в момент помещения в БД запись должна быть автоматически включена в набор. Операторы MANDATORY (обязательный) и OPTIONAL (необходимый) характеризуют вид исключения экземпляра записи из экземпляра набора описываемого типа. Так тип OPTIONAL означает, что запись может исключаться из набора данных в произвольный момент времени.

Поскольку БД предполагает соответствующее манипулирование данными рабочей группой КОДАСИЛ был также предложен соответствующий язык, позволяющий осуществлять запись в базе данных: STORE (запомнить), ERASE (стереть), MODIFY (изменить), CONNECT (присоединить), DISCONNECT (исключить) и др.

Оператор STORE (запомнить)—помещает данные, подготовленные в рабочей области программы, в базу.

Оператор ERASE (стереть) —удаляет из БД экземпляр записи.

Оператор MODIFY (изменить) обновляет текущую запись банка данных.

Оператор CONNECT (присоединить) включает текущую запись в набор.

Оператор DISCONNECT (исключить) исключает текущую запись из набора, но при этом она остается в БД.

Можно отметить, что в языке манипулирования данных существуют и другие операторы, но приведенный набор операторов ЯМД является функционально полным, т. е. позволяет образовывать в БД произвольное допустимое схемой базы данных состояние.

5.9.6. Иерархическая модель данных

Иерархическая модель данных, так же как и сетевая, основаны на возможности представления структур данных в виде графов. Но в отличие от сетевой на иерархическую модель накладываются более жесткие ограничения. Граф иерархической базы данных имеет древовидную структуру связей.

Древовидная структура или дерево — это граф, не содержащий циклов. Дерево представляет собой связанный граф, так как каждая вершина в нем завершает по крайней мере одно ребро (рис. 5.64).

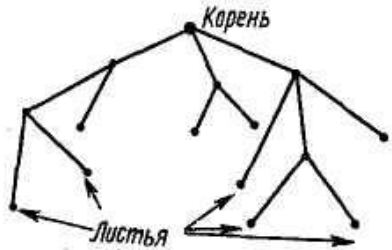


Рис. 5.64. Граф дерева

При работе с иерархической моделью данных граф, описывающий структуру данных, является направленным или ориентированным, т. е. дерево в этом случае будет ориентированным.

В зависимости от направления дуг в ориентированном графе выделяют такие типы вершин, как корень и лист.

Корень — это вершина, имеющая одну или несколько исходящих дуг и ни одной входящей.

Лист — это вершина, имеющая одну или несколько входящих дуг и ни одной исходящей.

Корень можно рассматривать как источник направленной древовидной структуры. От корня до любого листа легко проследить элементарную последовательность дуг. ***Число дуг*** между корнем и листом называется ***уровнем листа***. Вершину графа, не являющуюся ни корнем, ни листом, называют **узел ветвления.**

Примером иерархической структуры в виде дерева служит схема управления большинства организаций. Корнем схемы является ди-

ректор. От него исходят одна или несколько дуг к его заместителям. От них отходят дуги к более низким уровням управления и т. д. до непосредственных исполнителей, которые на графе соответствуют листьям.

Основными понятиями в иерархической модели данных являются **тип записи** и **иерархические отношения.** Вершины в дереве соответствуют типу сущности и называются типом записи. Тип записи состоит из одного или более **элементов данных.** Во многих иерархических моделях вместо понятия «тип записи» используют эквивалентное понятие «тип сегмента».

Иерархическое отношение (ветвь дерева) соединяет два типа записей и представляет собой множество связей между экземплярами записей этих двух типов.

Дуги (ветви) дерева соответствуют функциональному типу связи, т. е. типам 1 : 1, 1 : *M, M*: 1, и их называют **связью исходной порожденной.** Дуга исходит из **типа родительской записи** и заходит в **тип порожденной записи.** Таким образом, рассматривая последовательность связей — исходной-порожденный, можно выделить типы родительских и порожденных записей. Каждый экземпляр родительского типа записей может иметь связь с несколькими (в том числе и с нулем) экземплярами порожденных записей. В свою очередь, каждый экземпляр записи порожденного типа подчинен ровно одному экземпляру записи родительского типа. Другими словами, ***иерархическое отношение можно рассматривать как функцию, признаком которой служит экземпляр порожденного типа записи, а ее значением является экземпляр родительского типа.***

Иерархическая модель накладывает жесткие ограничения на иерархические отношения между записями. Поскольку любые два типа записей могут быть связаны не более чем одним иерархическим отношением, то иерархическим связям не требуются собственные имена. Каждая из иерархических связей может быть однозначно идентифицирована указанием родительской и порожденной записи.

Модели данных «сущность-связь». Они появились как обобщение и развитие иерархических и сетевых моделей. Модель «сущность-связь» была задумана как средство представления предметной области, не зависимого от особенностей среды хранения и не связанного соображениями физической эффективности. Базовыми структурами в этих моделях являются типы сущностей и связей. Тип сущности в модели носит название множество сущностей. Каждая сущность при этом идентифицирует объект предметной области. Связь между ними фиксируется заданием множества отношений. Различают два вида

отношений — ***слабые и стандартные***. Слабые связи идентифицируют иерархические отношения.

Множество связей (МС) в данной модели можно представить как математическое отношение *n* типов сущностей. Если МС есть множество, то его можно определить следующим образом:

$$МЗ=\{<m_1, m_2,...,m_n> | m_1 \in M_1, \ m_2 \in M_2,..., \ m_n \in \ M_n\},$$

где m_1 — сущность, принадлежащая множеству сущностей M_i, а $<m_1, m_2,...,m_n>$ — связь, принадлежащая множеству МС.

Домен в модели «сущность-связь» называют **множеством значе-, ний.** В этом случае значение представляет собой конкретный экземпляр множества значений.

Для изображений множеств сущностей и отношений могут быть использованы диаграммы, аналогичные графам. МС изображается ***прямоугольником***, множество отношений — ***ромбом***. Множествам обоих типов присваиваются имена; множества сущностей соединяются с множествами отношений, в котором они участвуют с помощью ненаправленных линий. Множество значений представляется ***овалом*** (рис. 5.65).

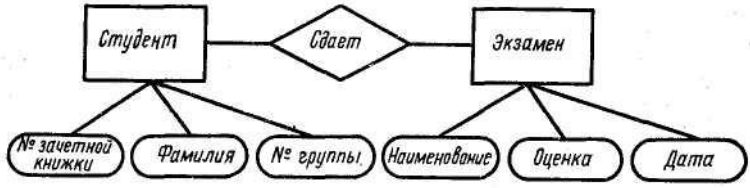


Рис. 5.65. Модель данных «сущность-связь»

Использование моделей «сущность-связь» является удобным средством для представления концептуальной информации.

Бинарные модели. Отношения, используемые для описания предметной области, в общем случае могут быть *n*-мерными. Интенсивно идут исследования по представлению информации с помощью только бинарных отношений, поскольку они позволяют лаконично представить сложные связи.

Бинарные модели, основанные на использовании бинарных отношений, имеют простые базовые структуры, способные обеспечить эффективное представление предметной области. Поскольку графы позволяют наглядно задавать отношения, их использование в бинарных моделях также позволяет лучше уяснить особенности модели. Вершины графа в бинарных моделях соответствуют классификационному обобщению экземпляров данных в типы и называются *категориями,* а дуги — *бинарным отношением* категорий. Граф, удо-

влетворяющий этим структурным представлениям, носит название *графа типов.* Каждому бинарному отношению ставится в соответствие отношение, имеющее противоположное направление. Например:

СТУДЕНТ —УЧИТСЯ У — ПРЕПОДАВАТЕЛЬ — ОБУЧАЕТ —

Обоим направлениям бинарного отношения присваиваются уникальные имена, которые называются *функциями доступа.*

В бинарном отношении категорий СТУДЕНТ и ПРЕПОДАВАТЕЛЬ направление от категории СТУДЕНТ к категории ПРЕПОДАВАТЕЛЬ есть функция доступа УЧИТСЯ У. Представленное отношение может быть охарактеризовано следующим образом:

СТУДЕНТ УЧИТСЯ У ПРЕПОДАВАТЕЛЯ

Функция доступа для противоположного направления будет

ПРЕПОДАВАТЕЛЬ ОБУЧАЕТ СТУДЕНТА

Расширение бинарного графа типов позволяет рассматрить понятия двух родов-объектов и поименованных бинарных отношений (связей). Взаимосвязи, в которых участвует более двух объектов, в свою очередь, интерпретируются как объекты.

Объект — это реализация категории. Например, объект — это конкретные студенты и преподаватели. Объекты подразделяются на абстрактные и конкретные. Причем подразумевается, что абстрактные всегда существуют, в то время как конкретные появляются и исчезают в описании реального объекта. Так, абстрактные объекты используются для представления чисел, дат и др.

Объекты соединены связями, которые отражают реализацию бинарного отношения. Каждой связи в обоих направлениях присваиваются имена, соответствующие функциям доступа.

Для создания категорий в бинарной модели данных используют оператор CATEGORY. Например, условие

СТУДЕНТ = CATEGORY

говорит о создании новой категории с именем СТУДЕНТ.

Для создания бинарного отношения следует задать его имя, функции доступа и категории соответствующих данному отношению объектов. Например:

СТУДЕНТ —ПРЕПОДАВАТЕЛЬ = RELATION (СТУДЕНТ, ПРЕПОДАВАТЕЛЬ, УЧИТСЯ У, ОБУЧАЕТ) определяет бинарное отношение СТУДЕНТ — ПРЕПОДАВАТЕЛЬ категорий СТУДЕНТ и ПРЕПОДАВАТЕЛЬ с функциями доступа УЧИТЬСЯ У, ОБУЧАЕТ.

В бинарных моделях отсутствует явно выраженное понятие «свойство объекта». Свойства объектов могут быть определены посредством бинарного отношения, заданного на множестве объектов

и других элементарных объектов, с помощью которых задаются значения свойств.

Как уже отмечалось, отношение характеризуется двумя функциями доступа, каждая из которых определяет минимальное и максимальное число объектов в присоединяемой категории. Ограничение на число объектов задается оператором AFN, позволяющим задать граничные величины подмножеств значений функций доступа. Так, например, если для функции доступа ОБУЧАЕТ запишем:

$$\text{ОБУЧАЕТ} = \text{AFN}\ (0, \infty),$$

то это будет означать, что преподаватель может вообще никого не обучать, или обучать любое число студентов.

Логический доступ к данным бинарной модели обеспечивается **с** помощью программ, реализующих элементарные операции доступа.

Семантические сети.

Для представления семантических (смысловых) текстов, задаваемых на естественном языке, разработаны семантические сетевые модели.

Семантическая сеть представляет собой ориентированный граф с намеченными вершинами и дугами. При этом если вершины обозначаются только в целях ссылок к ним, то метки дуг содержат сведения о некоторых их семантических свойствах и значениях.

Для представления данных используются четыре типа вершин: **концепты** (или понятия), **события, характеристики** (свойства) и **значения.**

Концепты — константы или параметры, которые специфицируют физические или абстрактные объекты.

События — соответствуют действиям, наблюдаемым в представляемой области.

Характеристики — вершины, соответствующие свойствам концепты.

Значения — вершины, соотносящиеся с областями значений, которые могут принимать характеристики.

Поскольку имеется четыре типа вершин, необходима соответствующая зависящая от этих типов интерпретация дуг, соединяющих различные вершины. Модели семантической сети предусматривают возможность распределения вершин по типам. В этом случае следует различать вершины-концепты и вершины-классы, которые собственно и представляют определенные типы вершин. Например, КУЛЕШОВ— концепт, СТУДЕНТ— класс.

Различие между классом и концептом весьма близко к тому же, что между типом и экземпляром в других моделях. Отличие заключается в

том, что граф семантической сети включает как классы, так и концепты. Кроме того, концепт может быть соотнесен с несколькими классами.

Поскольку семантические сети предусматривают задавать на графе в явном виде различие между вершинами-концептами и вершинами-классами, то в рассмотрение вводятся три вида дуг: утверждение; порождение экземпляра; бинарное отношение.

Утверждение — дуга, соединяющая два концепта.

Порождение экземпляра — дуга, между классом и концептом.

Бинарное отношение — дуга, связывающая два класса.

Рассмотрим пример семантической сети (рис. 5.66), иллюстрирующей сказанное.

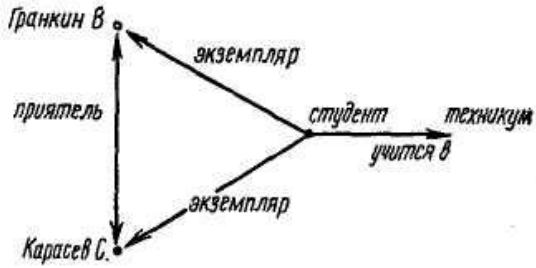


Рис. 5.66. Пример семантической сети

В данной семантической сети, вершины ГРАНКИН В. и КАРАСЕВ С. —концепты, СТУДЕНТ и ИНСТИТУТ —классы. Дуга ПРИЯТЕЛЬ — утверждение. Дуги, связывающие вершину СТУДЕНТ с вершинами КАРАСЕВ С. и ГРАНКИН В., отражают связь экземпляров с классами. Дуга УЧИТСЯ В отображает бинарное отношение между классами СТУДЕНТ и ИНСТИТУТ.

Классы могут быть связаны в иерархию в соответствии со связями ЕСТЬ НЕК и ЕСТЬ — ЧАСТЬ. Эти связи позволяют из отдельных понятий и классов строить более общие понятия и классы.

Для представления в семантической сети некоторых событий и действий вводится набор простых отношений, характеризующих основные компоненты события. Для построения с помощью семантической сети структуры события в первую очередь выделяют из него само действие, описываемое обычно глаголом. После этого выделяют лиц, совершающих действие и объекты, над которыми оно осуществляется. Лицо, осуществляющее действие, называется **агентом.** Вещи, над которыми действие осуществляется, называют объектами.

Лицо, получающееся результатом действия или испытывающее его, называется **адресат.**

Рассмотрим предложение: «Мастер починил телевизор». В этом предложении выделим действие: ПОЧИНИЛ. Очевидно, что объектом является МАСТЕР (рис. 5.67).

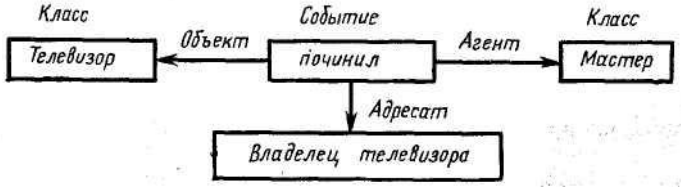


Рис. 5.67. Сеть предложения «Мастер починил телевизор»

В данном предложении адресат явно не указан, но его наличие можно предположить. Рассмотрим более сложное предложение:

«Вчера мастер Босин П. А. починил магнитофон «Юность», принадлежащий студенту Афонину К. А.».

Семантическая сеть для данного предложения представлена на рис. 5.68.

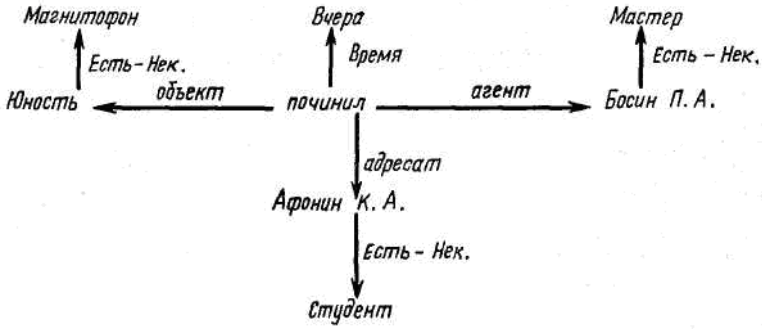


Рис. 5. 68. Пример семантической сети

В этом предложении появилась новая дуга ВРЕМЯ, указывающая на то, когда происходит событие. В общем случае в семантических сетях кроме простых отношений: агент, объект, адресат выделяют и другие, позволяющие ***описать широкий класс событий***. К ним относят: время, место, инструмент, цель и др.

Операции, совершаемые над данными, задаваемые семантической сетью, разбиваются на два подмножества: операции над классами и над бинарными отношениями.

Над <u>классами</u> могут быть совершены четыре операции:

- создание экземпляра некоторого класса или установление принадлежности существующего экземпляра некоторого класса к еще одному;

- устранение принадлежности экземпляра к некоторому классу или полное его исключение;

- выборка экземпляров, принадлежащих к одному классу;

- определение принадлежности экземпляра указанному классу.

Над <u>бинарными отношениями</u> могут быть совершены три операции:

- установление связи между классами;

- выборка всех экземпляров, связанных в данном бинарном отношении с указанным экземпляром;

- установление наличия связей между двумя экземплярами.

В общем случае реализация всех вышеперечисленных операций требует создания специальных программ, учитывающих рассматриваемую предметную область.

Разработка семантических сетевых моделей явилась следствием повышения требований к интегрированному представлению данных, включающему не только данные, но и их ***категории, свойства категорий и операции над данными***. Важную роль в развитии моделей данных этого класса сыграли проблемы алгоритмизации процессов естественного языка. Модели семантических сетей широко используются при разработке систем искусственного интеллекта.

Инфологические модели данных.

Инфологическое представление полностью независимо от физических параметров среды хранения. Эта модель в качестве базовой использует понятия: **объект, свойства и связи.**

Под **объектом** понимается нечто, представляющее интерес для решаемой задачи. Предполагается, что существование объекта связано с такими событиями, как появление (возникновение), изменение и исчезновение.

Объекты подразделяются на атомарные и составные.

Атомарный объект,— это любой объект, дальнейшее разложение которого на другие объекты невозможно.

Составной объект включает в себя множество объектов.

С каждым объектом связывается определенный набор свойств. Важным свойством существования объекта является время (время его возникновения и исчезновения).

С помощью базовых концепций объектов, свойств связей и времени формируется **элементарный факт**. Элементарный факт задается формально как тройка <*x, y,* z> или *(*<x_1, x_2, ..., x_n>rz*)*, где <x_1, x_2, ..., x_n> —кортежи объектов, *y* — свойство, *r* — отношение, *z* — время. Такую базовую структуру называют **элементарной плеядой**.

Так как в рамках инфологической модели все может быть объявлено объектом, то свойства и отношения, в свою очередь, также могут рассматриваться как объект.

В данной модели типы объектов вводятся путем группирования объектов и их свойств. Объектная группа *O(p),* соотнесенная со свойством *p,* определяется как совокупность объектов, потенциально (вне связи со временем) имеющих свойство *p.* Объекты, обладающие свойством *p,* в определенный момент времени образуют подмножество $O_t(p)$ множества *O(p),* называемое ***временным срезом объектной группы.***

В инфологической модели данных понятие ***атрибута*** вводится посредством понятий ***объекта, свойства, связи и времени***.

Атрибут определяется как множество свойств A = {p_i} объектной группы O(p), такое, что в каждый момент времени каждый объект x, принадлежащий O_t(p), содержится, по крайней мере, в одной O_t(p).

Свойства p — это значение атрибута A объектной группы O(p).

Базовые операции включения, обновления и удаления в инфологической модели связаны с вводом новых элементарных сообщений и предполагают наличие механизмов интерпретации ***элементарного сообщения.***

Процедуры обращения к БД **(транзакции)** могут быть представлены ***парой (оператор, параметр*)**.

К основным операциям, изменяющим БД, относят <ДОБАВИТЬ, сообщение> <УБРАТЬ сообщение> <ЗАМЕНИТЬ сообщение 1, сообщение 2>.

На основе базовых процедур возможно конструирование более универсальных запросных средств.

Следует иметь в виду, что возможности инфологической модели существенно меньше возможностей естественного языка и их удобно использовать лишь в тех случаях, когда модель предметной области позволяет описать только очевидные факты.

5.9.7. Проектирование данных

В процессе проектирования данных можно выделить два этапа: инфологический и даталогический.

На **инфологическом этапе** проектирования рассматриваются вопросы, связанные со смысловым содержанием данных.

Даталогический этап проектирования направлен на решение вопросов представления данных в памяти машины.

Даталогическое проектирование, в свою очередь, подразделяют на ***логическое и физическое***.

Таким образом, процесс проектирования БД представляет собой сложный многоуровневый процесс, охватывающий все аспекты ее использования: от удобства обращения к базе пользователей до конкретного представления данных в ЭВМ.

Весь процесс проектирования можно представить в виде последовательности операций, начинающихся с описания предметной области и заканчивающихся схемой внутренней модели базы данных.

На первом этапе проектирования (инфологическом) изучается предметная область и проводится ее описание. С этой целью идентифицируются все типы объектов (сущностей), представляющие интерес для введения в БД, определяются связи между этими объектами, а также выявляются ограничения. Для описания предметной области обычно используют концепцию моделей данных: «сущность—связь» или «инфологическую модель». Эти описания позволяют обеспечить взаимодействие между пользователями и проектировщиками системы. Проектирование начинается с предварительной структуризации предметной области. Обычно для облегчения этого процесса составляется перечень вопросов, на которые требуется ответить:

какие типы объектов входят в состав предметной области?

каковы имена каждого типа объектов?

каково значение (семантика) каждого типа?

какими средствами обладает каждый тип объекта?

какие атрибуты объектов представляют интерес?

каковы имена каждого атрибута?

На основе собранной информации о типах объектов выявляются типы существующих связей для систематизации собираемой информации, а также предлагается использовать вопросы следующего плана:

какие типы связей (отношений) могут иметь место между каждой парой типов объектов?

каковы имена каждого типа связи?

каково значение каждого типа связи?

Собранную информацию оформляют в виде специальных диаграмм. Часто для обозначения объектов используют прямоугольники, и для атрибутов — овалы, соединяя их с соответствующими объектами

ненаправленными ребрами. Для обозначения связей используются ромбы, которые также соединяют их с объектами ребрами.

Пример. Рассмотрим графическую диаграмму, соответствующую описанию предметной области производства телевизоров (рис. 5.69).

Рис. 5.69. Пример графической диаграммы

При создании описания предметной области проектировщик разбивает ее на ряд локальных областей, которые потом объединяются. Выбор размеров локальных областей в общем случае является произвольным. Но для удобства проектирования в одной локальной области рекомендуют использовать не более 6—7 объектов.

Моделирование локальных представлений. Для представления информации в модели «сущность-связь» конструктивными элементами модели являются: ***сущность, атрибуты*** **и** ***связи.***

Основным элементом локального представления некоторого явления, процесса или объекта предметной области, о котором необходимо собрать информацию, является ***сущность.*** При формулировании сущностей следует различать такие понятия, как **тип и экземпляр.**

Понятие **тип сущности** относится к набору однородных предметов или явлений, выступающему как целое. **Экземпляр сущности** относится к конкретному элементу набора. Например, типом сущности может быть ТЕЛЕВИЗОР, а экземпляр сущности — РУБИН. На концептуальном этапе проектирования необходимо сформулировать сущности, требуемые для описания локального представления. При этом возникает проблема ее выделения в качестве конструктивного элемента, так как некоторая информация может быть представлена как **атрибут, сущность или связь**. Например, тот факт, что конкретный студент учится в институте, может быть выражен сущностью СТУДЕНТ, либо связью УЧИТСЯ между сущностью СТУДЕНТ и ИНСТИТУТ, либо как атрибут в сущности ГРУППА ИНСТИТУТА.

При возникновении такой неоднозначности формулирования сущностей рекомендуется руководствоваться следующими правилами:

1. Необходимо выбирать вариант, более гибкий с точки зрения представления информации, т. е. позволяющий представлять не только всю часть некоторой информации, но и ее отдельные фрагменты.

2. Для моделирования порции информации должна использоваться одна и только одна конструкция. Другими словами, следует избегать избыточности в использовании конструктивных элементов.

Другое важное положение, связанное с формулированием сущностей, касается выбора наименований сущности. Так как она представляет собой информационный факт, то этому факту должно быть дано четкое наименование, что имеет важное значение для стадии объединения локальных представлений.

Выбор атрибутов сущности. Свойства сущностей определяются с помощью атрибутов.

Атрибут — это характеристика сущности, имеющая имя. Атрибуты используются для определения того, какая информация должна быть собрана о сущности. Примерами атрибутов для сущности СТУДЕНТ могут быть: НОМЕР ЗАЧЕТНОЙ КНИЖКИ, ПОЛ, НОМЕР ГРУППЫ и т. д.

Несмотря на то что совокупность атрибутов не может служить основой для выделения сущностей, из множества атрибутов обычно выделяют несколько (или один) атрибутов, позволяющих однозначно распознавать экземпляр сущности.

Атрибут (или совокупность атрибутов), значение которого единственным образом определяют экземпляр сущности, называют **ключом.** Если для описания типа сущностей выбрана совокупность атрибутов, не содержащих ключа, то создается специальный атрибут, играющий роль ключа. В общем случае сущность может иметь несколько ключей.

Таким образом, ***атрибуты могут быть разделены на два класса: те, которые служат для идентификации экземпляров сущности, т. е. являются ключами, и те, которые описывают свойства сущностей.***

На этапе построения локальных представлений в процессе выбора атрибутов рекомендуется каждому ставить в соответствие следующие характеристики:

- наименование, т. е. уникальное обозначение атрибута;
- описание — словесное изложение смысла атрибута;
- роль, т. е. конкретное использование атрибута.

Спецификация связей. После выделения сущностей, характеризующих предметную область, и соответствующих атрибутов локальное

представление дополняется информацией, раскрывающей зависимости между экземплярами сущностей.

Одна из неформальных процедур для этого шага заключается в попарном объединении между собой всех экземпляров сущностей, входящих в рассматриваемое локальное представление, и установлении существования некоторой связи для каждой пары сущностей.

После их выявления определяются связи необходимые и избыточные. Каждой необходимой связи присваивается имя и определяются ее характеристики, которые включают тип связи (1:1, 1:*M, M:M,M:*1*).*

Объединение моделей локальных представлений. В результате объединения локальных представлений получается единая глобальная информационная структура. Объединение может быть осуществлено на базе трех основополагающих подходов: **идентичности, агрегации и обобщении.**

Идентичность позволяет объединять несколько сущностей путем объединения двух или более элементов синонимами.

Для проверки согласованности результата объединения локальных представлений на основе понятия идентичности предложено следующее правило:

Если объект из одного локального представления идентичен объекту из другого представления, ни один из этих объектов не должен в дальнейшем принимать участие в каком-либо другом объединении идентичности между этими двумя представлениями.

В нескольких локальных представлениях рассматривается один и тот же объект, но его отдельные составляющие могут различаться. Например, имеется два локальных представления ТЕЛЕВИЗОР (рис. 5.70, *а).*

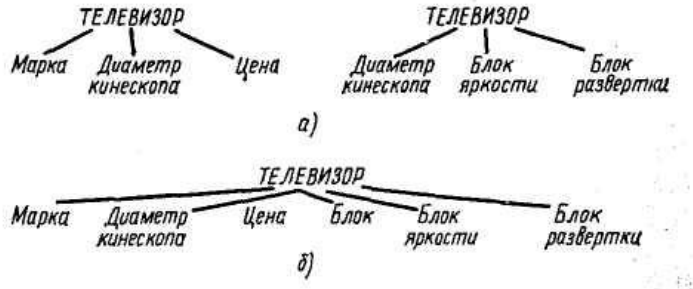


Рис. 5.70. Объединение идентичности

В результате объединения идентичности вместо отдельных локальных представлений будет построено новое (рис. 5.70, *б).*

Агрегация позволяет рассматривать связь между элементами модели как новый элемент. Например, сущность ФАКУЛЬТЕТ может быть рассмотрена как агрегация сущностей КАФЕДРА, ДЕКАНАТ.

При объединении представлений агрегация встречается в следующих двух формах.

1. В одном представлении агрегатный объект определяется как целое, а в другом — в виде составных частей.

Например, в одном локальном представлении определены в качестве сущности объект ТЕЛЕВИЗОР, а в другом — блоки: КИНЕСКОП, БЛОК ЯРКОСТИ, БЛОК РАЗВЕРТКИ, являющиеся составными частями объекта ТЕЛЕВИЗОР. Причем во втором представлении не указан явно тот факт, что вышеперечисленные блоки — составные части телевизора.

Простое объединение позволяет слить эти два локальных представления, не выражая явным образом, что ТЕЛЕВИЗОР является агрегацией частей КИНЕСКОП, БЛОК ЯРКОСТИ, БЛОК РАЗВЕРТКИ. Чтобы включить эту информацию в модель объединенного представления, необходимо выполнять объединение с использованием агрегации.

2. Агрегатный объект в одном локальном представлении до конца как единое целое не определен.

Например, в одном представлении определены КИНЕСКОП и БЛОК ЯРКОСТИ, а в другом БЛОК РАЗВЕРТКИ, БЛОК СИНХРОНИЗАЦИИ, БЛОК ЦВЕТНОСТИ, являющиеся составными частями объекта ТЕЛЕВИЗОР, который не назван ни в одном представлении. Для повышения возможностей совместного использования данных можно ввести в рассмотрение агрегат ТЕЛЕВИЗОР (рис. 5.71).

Рис. 5.71. Объединение агрегации

Понятие **обобщения** близко к понятию агрегации, но в отличие от последней, которая может быть представлена в виде составных частей, образующий некоторое «целое», обобщение связано только с «целыми». Обобщение относится к типу абстракции, в которой группа подобных элементов воспринимается как родовой элемент. При этом различия между отдельными элементами опускаются.

Например, УЧАЩИЙСЯ может быть воспринят как УЧАЩИЙСЯ школы, УЧАЩИЙСЯ ПТУ, УЧАЩИЙСЯ техникума.

Так же как и агрегация, обобщение может встречаться в двух формах:

1. В одном локальном представлении определено некоторое множество объектов, которое может быть объединено общим для этих объектов родовым понятием, а само оно указано в другом локальном представлении.

Пример.

I представление	II представление
Цветные телевизоры	Телевизоры
Черно-белые телевизоры	

Здесь родовым понятием, объединяющим оба представления, будет ТЕЛЕВИЗОР.

2. Ни одно из объединяемых локальных представлений не содержит родового понятия.

I представление	II представление
Цветные телевизоры	Переносные телевизоры
Черно-белые телевизоры	

В этом случае установить наличие родовой связи между специфичными типами объектов можно только в процессе сопоставления объектов из различных локальных представлений.

Использование объединения обобщением позволяет повысить эффективность доступа пользователей к данным, хранящимся в базе.

При формировании глобального представления данных путем комбинированного использования идентичности, агрегирования и обобщения можно отобразить в модели сложные связи, существующие между объектами и понятиями в предметной области.

Сам процесс объединения локальных представлений обычно носит интерактивный характер. В целях упрощения процесса объединения обычно осуществляют бинарное объединение, т. е. на каждом этапе объединяются только два локальных представления. Процесс попарного объединения повторяется для всех локальных представлений до тех пор, пока все они не будут интегрированы в одно глобальное (рис. 5.72).

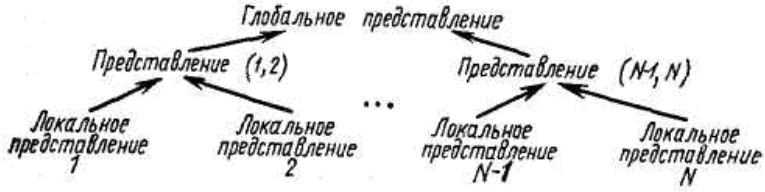


Рис. 5.72. Бинарное объединение

В процессе объединения могут выявляться противоречия между отдельными локальными представлениями. Противоречия обусловлены обычно неполнотой или ошибочностью спецификаций или же некорректностью требования. Так, например, в одном локальном представлении связь между объектами может быть отнесена к типу 1 :1, а в другом —типу 1 *:M* или *M:*1. Большинство из противоречий такого вида может быть разрешено на этапе объединения путем выполнения соответствующих коррекций. Процесс объединения требует согласования и устранения всех выявленных противоречий.

После завершения объединения полученное глобальное представление служит исходной информацией для **даталогического** этапа проектирования БД. Данный этап предусматривает полученное инфологическое описание предметной области отобразить в описание БД.

Естественно, что отображение одного описания в другое будет зависеть от принятой за основу модели данных, поддерживаемой соответствующей СУБД. Существующие СУБД по виду поддерживаемой модели могут быть отнесены к одному из трех классов: реляционные, сетевые, иерархические.

В случае ***реляционной*** СУБД описание предметной области трансформируется в реляционную схему. ***Объекты при этом отображаются в отношения БД.***

В случае ***сетевой*** СУБД ***описание предметной области отображается в граф***. Типы объектов (сущностей) представляются в типы записей, а типы связей — в типы наборов.

В случае ***иерархической*** СУБД ***описание предметной области преобразуется в множество деревьев***. Типы сущностей при этом также будут отображены в типы записей, а типы связей — в типы «исходный-порожденный».

Процесс этих преобразований является неформальным и во многом зависит от проектировщика.

Принципы проектирования физической базы данных. Физическая организация данных оказывает существенное влияние на эксплуатационные характеристики проектируемой БД, такие, как объем занимаемой памяти, время отклика базы на запрос пользователя и т. д.

Под проектированием физической БД будем понимать процесс создания эффективной ее структуры на выбранной логической структуре.

Физическая база данных представляет собой совокупность совместно хранимых взаимосвязанных данных, состоящих из одного или нескольких типов хранимых записей.

Понятие структуры физической БД включает: **формат хранимой записи, структуру путей доступа к данным и размещение записей на физических устройствах.**

Наиболее простой формой хранения данных в памяти ЭВМ является линейный список. **Линейный список** представляет собой конечное и упорядоченное множество объектов {*x*[1], *x*[2], ..., *x*[*n*]}, структурные свойства которого связаны только с линейными относительным расположением элементов данных. Порядковый номер, расположенный в квадратных скобках, указывает на относительное положение элементов в списке.

Линейные списки, естественно, используются в тех случаях, когда встречаются упорядоченные множества данных переменного размера и где операции включения, поиска, удаления элемента данных должны выполняться в произвольных местах.

Одномерный линейный список, используемый для хранения данных в памяти ЭВМ, называют также **вектором данных.**

Для линейного списка существует две возможности представления в ЭВМ: **последовательное и связанное**.

Последовательное представление. Является более простым и предполагает, что элементы списка размещаются в последовательных элементах памяти ЭВМ (рис. 5.73).

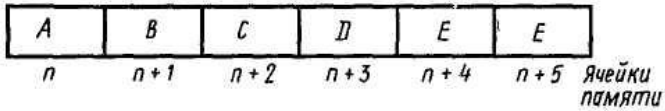


Рис. 5.73. Последовательное распределение памяти для представления линейного списка

При таком представлении списка возникают определенные сложности в реализации вставки нового элемента в середину. Например, чтобы включить в список между элементами *D* и *E* новый элемент *K,* необходимо изменить место элементов *E* и *F*. Точно так же удаление элемента из списка ведет к появлению в списке пустой ячейки и для его уплотнения необходимо осуществлять смещение оставшихся элементов.

Связное представление. Оно предусматривает задание для каждого элемента списка отношений следования и предшествования с помощью указателей, задающих связь между данными. При таком

представлении каждая ячейка содержит элемент данных и указатель (адрес) на последующий элемент списка. При связанном распределении не требуется, чтобы список хранился в последовательных элементах памяти (рис. 5.74).

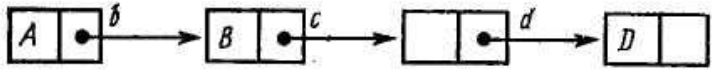


Рис. 5.74. Связанный список данных

Добавление или исключение некоторых данных в этом случае можно выполнить с помощью простой операции изменения значения указателя (рис. 5.75).

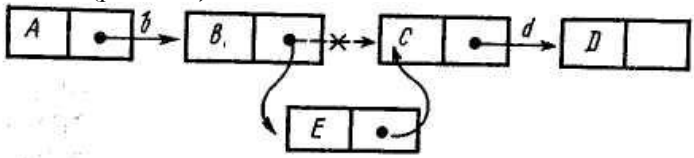


Рис. 5.75. Пример включения элемента в связанный список данных

Таким образом, использование связанных списков более удобно в случае динамически изменяющихся линейных структур.

Списки могут быть и двусвязанными (рис. 5.76).

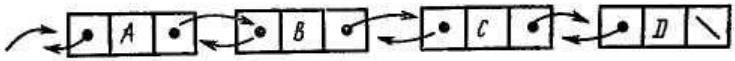


Рис. 5.76. Двусвязанный список данных

При таком задании списка необходимо кроме прямого указателя для каждого элемента вводить в рассмотрение и обратный.

Структура линейного списка, представленная с помощью связанного распределения, называется также **цепной структурой или цепью.** Физическая последовательная и связанная структуры являются основными для большинства числа методов доступа к данным.

Методы доступа. Под методом доступа понимается совокупность технических и программных средств, обеспечивающих возможность хранения и выборки данных, расположенных на физических устройствах ЭВМ. В методе доступа выделяют два компонента: **структура памяти и механизм поиска.**

Наиболее широко используемыми методами доступа являются: **последовательный; прямой произвольный; индексно-последовательный; индексно-прямой; основанный на использовании явных древовидных структур.**

Последовательный доступ. Он реализует доступ к данным базы путем последовательного просмотра записей.

Рассмотрим случай, когда физически последовательная структура содержит несколько смежных записей. Записи могут быть неупорядочены или упорядочены по значениям первичного ключа. Обычно каждая запись располагается в отдельном блоке.

Среднее количество физических блоков $N_{ср}$, к которым осуществляется доступ при поиске произвольной записи, равно $N_{ср}=(1+N)/2$. Данное выражение справедливо как для упорядоченных, так и неупорядоченных записей при условии, что искомая запись существует. В случае если искомая запись отсутствует, то для неупорядоченного файла $N_{ср} = N$, т. е. будут проверены все записи. Таким образом, неупорядоченный файл является неэффективным, если приходится часто обращаться к поиску отсутствующей записи.

Поскольку внесение изменений в произвольном порядке в последовательную структуру требует большого количества операций по перемещению записей, режим внесения изменений строго ограничивают.

Прямой доступ. В том случае, когда имеется возможность выделить в памяти для каждой записи место, определяемое уникальным значением ее первичного ключа, можно построить простую функцию преобразования ключа в адрес, обеспечивающую запоминание и выборку каждой записи в точности за один произвольный доступ к блоку.

Использование прямого доступа позволяет эффективно с точки зрения временных затрат осуществлять поиск данных в базе.

Методы прямого доступа подразделяются на две группы:

1. Доступ с помощью ключа, эквивалентного адресу;
2. Хэширование (метод произвольного доступа или рассеянной памяти).

Использование первой группы методов доступа возможно в тех случаях, когда в качестве атрибута в запись включается адрес памяти, в котором будет размещена запись. При работе с БД этот атрибут будет использоваться в качестве ключа.

Методы второй группы широко используются для обеспечения быстрой выборки и обновления записей по заданному значению первичного ключа.

Основная идея хэширования заключается в том, что каждый экземпляр записи размещается в памяти по адресу, вычисляемому с помощью специальной ***хэш-функции***. В этом случае экземпляры записей хранятся не в последовательных ячейках, выделенного блока памяти, а случайным образом рассеиваются по всему блоку. Причем каждая новая запись помещается в блок памяти таким

образом, что ее присутствие или отсутствие может быть установлено без поиска по всему блоку.

Другими словами, ***в методе хэширования строится отображение множества ключевых значений экземпляров записей во множество физических адресов.***

В качестве хэш-функций обычно выбирается некоторое правило, преобразующее значение типа записи в адрес.

Рассмотрим на примере алгоритм построения хэш-функции.

Пусть в памяти машины для хранения экземпляров записей выделен блок *B* из 1024 слов. Экземплярами записей z_i, которые необходимо хранить в *B,* являются наборы символов длиной в два слова. Таким образом, в блоке *B* можно разместить 512 отличающихся записей z_i.

Предположим, что начальный адрес блока в памяти α. Хэш-адресом для данного случая может быть цепочка I_z из девяти битов, поскольку формула $\alpha+2I_z$ будет давать адрес, попадающий внутрь блока. Цепочка I_z может быть вычислена для каждой записи длиной в два слова по следующему алгоритму. Пусть z_i хранится в словах *a* и *b*. Тогда:

1. Умножим *a* на *b* и получим в результате *c* (произведение занимает два слова).
2. Сложим два слова, составляющие *c,* получим в результате *d.*
3. Возводим *d* в квадрат, получим в результате *e.*
4. Извлечем девять центральных битов из *e* и примем, что это и есть искомое значение I_z.

В зависимости от свойств битовых цепочек, представляющих экземпляры записей или их ключей, могут использоваться различные хэш-функции. При их подборе важно, чтобы хэш-функция равномерно распределяла все множество z_i по выделенному блоку памяти.

Обычно выбранная хэш-функция не может быть полной гарантией того, что различные экземпляры записей получат различные хэш-адреса. Всегда в процессе хэширования возникает ситуация, когда два (или более) экземпляра записей получают один и тот же адрес, что приводит к коллизии. Коллизия возникает, когда при добавлении новой записи в блоке памяти выясняется, что позиция записи, задаваемая хэш-адресом, уже занята записью, отличной от той, которую собираются туда поместить.

Для разрешения коллизий имеется много методов. К ним относятся методы последовательного сканирования и цепочки.

Метод последовательного сканирования. Он состоит в том, что при возникновении коллизии просматриваются последовательно (сканируются) участки памяти, отведенные для хранения записей, до

тех пор, пока не будет найдена свободная позиция, куда и помещается запись.

Метод цепочки. Вместо хранения самих элементов блока можно хранить в нем указатели на связанные списки экземпляров, записей, имеющих одинаковый хэш-адрес.

После хэширования экземпляра записи (ее ключа), если участок памяти по вычисленному адресу свободен, запись размещается по этому адресу. Если же участок памяти по вычисленному адресу занят, то происходит обращение по указанию к следующему участку памяти (элементу списка), на который и помещается запись.

При поиске записей действия выполняются в той же последовательности. Вначале проверяется участок памяти по вычисленному адресу. Если там находится запись с другим значением ключа, то по указателю обращаются к следующей записи, и так до тех пор, пока не будет найдена необходимая запись.

Память, выделяемую для организации списков, называют ***областью переполнения.***

Индексно-последовательный метод доступа. Он строится на основе упорядоченного физически последовательного файла и иерархической структуры индексов блоков, каждый из которых упорядочен по значениям первичных ключей подобно записям в файле данных. Данный метод позволяет обеспечивать как последовательный, так и произвольный доступ к данным. С этой целью в рассмотрение вводится новый параметр — индекс блока. **Индекс блока** —представляет собой упорядоченную таблицу значений первичных ключей, в которой каждый элемент блока содержит наибольшее значение ключа среди всех записей в указанном блоке. С каждым значением ключа в индексе связан указатель соответствующего блока (рис. 5.77).

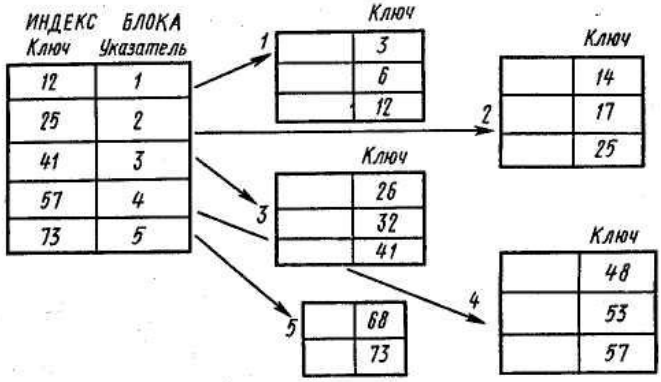


Рис. 5.77. Организация индекса блока

Использование упорядоченного индекса блока требует таких же записей данных в файле. Благодаря этой упорядоченности можно в каждом элементе индекса указать границы соответствующего блока: значение ключа представляет верхнюю границу, а указатель задает нижнюю границу, так как указывает адрес блока, точнее, адрес первой записи в блоке, содержащей наименьшее значение ключа.

В этом случае поиск экземпляра записи осуществляется посредством первоначального установления номера блока, в котором может находиться запись, а затем последовательного поиска в этом блоке, пока не будет установлено местонахождение искомой записи или ее отсутствие.

Данный метод позволяет обеспечивать быстрый доступ к записям баз данных и файлов большой размерности, в которых поиск по одному только индексу приводит к значительным затратам времени.

Индексно-произвольный метод доступа. Данный метод обеспечивает доступ к экземплярам записей на основе использования индекса. Поэтому метод называют также «метод доступа с полным индексом».

Полный индекс представляет собой такую организацию файла, при которой для каждого конкретного экземпляра записи предусмотрен соответствующий индекс. Этот индекс составляется из значения первичного ключа и указателя экземпляра записи, содержащий это значение (рис. 5.78).

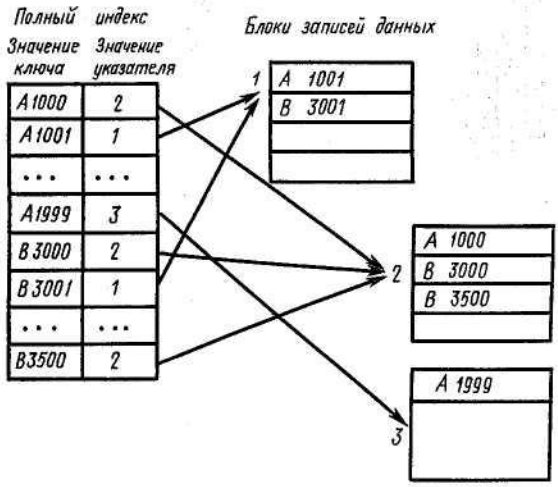


Рис. 5.78. Схема доступа с полным индексом

Обычно для ускорения поиска индексы упорядочиваются. При этом упорядочивание или физически близкое размещение хранимых записей не требуется.

После того как в индексе обнаружено искомое значение ключа, доступ к записи можно осуществить с помощью указателя, который хранится в индексе рядом со значением ключа. В этом случае, если искомое значение ключа в индексе не найдено, поиск завершается на уровне индекса. Допускается произвольный доступ к индексу посредством хэширования.

Метод доступа с полным индексом обеспечивает эффективную выборку одиночных записей, а также простоту операций обновления.

Методы доступа, основанные на использовании древовидных структур. Получили широкое распространение методы представления в памяти ЭВМ данных в виде древовидных сетевых структур и соответствующие методы доступа к ним. Эти методы стали конкурентами классических, таких, как индексно-последовательный метод или хэширование. К основным древовидным структурам обычно относят: бинарное дерево и *B*-дерево.

Бинарным (двоичным) деревом называется древовидный граф с двоичным ветвлением, т. е. с таким ветвлением, когда из каждой вершины (кроме концевых) выходят две дуги.

Бинарные деревья представляют собой широко используемый вид структур баз данных, обеспечивающий как произвольную, так и последовательный выбор данных.

Важную роль проектирования древовидных структур данных играет понятие ***сбалансированного дерева***. Прежде чем его рассмотреть, введем некоторые дополнительные понятия. Уровень вершины *i* определяется длиной пути от корневой вершины *T* до вершины *i*. Корневая вершина *T* имеет нулевой уровень. Ветвь дерева определяется его максимальным уровнем.

Дерево называется <u>сбалансированным</u>, если разница уровней любых двух конечных вершин не превышает единицы.

Построение сбалансированного дерева делает равновероятным обращение к любой из его вершин, что позволяет минимизировать среднюю длину доступа. Механизм поиска по бинарному дереву основан на том, что каждая вершина дерева помечена отдельным ключом. Ключи упорядочены следующим образом: $k_{i_1} \le k_i \le k_{i_2}$, где k_i — ключ *i*-й вершины; k_{i_1} —ключ i_1 -вершины, соответствующей левой дуге *i*-й вершины *(—);* k_{i_2} — ключ вершины, лежащей на

правой дуге. При поиске некоторого ключа k вначале просматривается корневая вершина дерева T и сравнивается ключ k с ключом k_T корневой вершины. В случае $k=k_T$ поиск завершен успешно. В том случае, когда $k<k_T$, поиск продолжается в левом поддереве, а при $k>k_T$ поиск продолжается в правом поддереве (рис. 5.79).

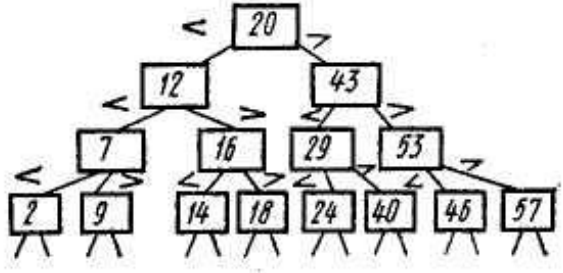


Рис. 5.79. Бинарное дерево поиска

Кроме бинарных деревьев в БД часто используются так называемые *B*-деревья. *B*-дерево представляет собой обобщение понятия бинарного дерева. В нем из каждой вершины могут выходить более двух ветвей. Обычно *B*-деревья используются только для организации индекса. Записи данных располагаются в отдельной области, для которой возможен произвольный доступ. Каждая вершина *B*-дерева состоит из совокупности значений первичного ключа, указателей индексов и ассоциированных данных. Указатели индекса используются для перехода на следующий, более низкий уровень вершины в *B*-дереве.

Ассоциирование данных фактически представляют собой совокупность указателей данных и служат для определения физического местоположения данных, ключевые значения которых хранятся в этой вершине индекса.

Имеется множество различных гибридных древовидных организаций, сочетающих в себе лучшие качества метода хэширования, бинарных деревьев, *B*-деревьев и их многочисленных вариантов.

5.9.8. Архитектура автоматизированных банков данных

Автоматизированные банки данных (АБД) являются составной частью информационного обеспечения САК и состоят из баз данных САК и системы управления базами данных. АБД являются обслуживающей подсистемой САК и предназначены для автоматизированного обеспечения необходимыми данными подсистем САК. АБД представляют собой человеко-машинную систему, включающую специальные структуры организации информации, алгоритмы, специализированные языки и административный персонал.

В совокупности они обеспечивают создание и эксплуатацию эффективных систем накопления информации, поступающей от нескольких источников, ее обновление, корректировку, многоцелевое использование в различных подсистемах консультирования, а также прямую связь с пользователем для получения ответов на произвольные вопросы, заданные в требуемой форме.

Архитектура АБД является многоуровневой. Выделяют внутренний, концептуальный и внешний уровни.

Внутренний уровень отражает требуемую организацию данных в среде хранения, т. е. он связан со способом физического хранения данных во внешних устройствах.

Концептуальный уровень обеспечивает интегральное (общее) логическое представление о предметной области, т. е. представляет собой описание структуры всей БД. Уровень отражает обобщенное представление всех пользователей о БД.

Внешний уровень обеспечивает представление данных, требуемое конкретным пользователем, т. е. представляет собой описание структуры данных, требуемых пользователю в процессе решения задач или реализации запросов. Уровень отражает частные представления пользователей о БД.

Внутренний уровень определяют описанием соответствующих файлов. Концептуальный уровень определяют концептуальной моделью (или схемой), которая описывает интегральную логическую структуру данных, определяет типы хранимых данных, их структуру, логические связи между отдельными элементами. Внешний уровень определяют внешней схемой (или подсхемой), которая описывает логическую структуру данных, необходимую для реализации конкретной задачи или запроса пользователя.

Способы описания внутреннего отображения данных определяются операционной системой и составом устройств внешней памяти. Для логического описания данных (схем и подсхем) в СУБД имеются (или разрабатываются) специальные языки описания данных (ЯОД). Основные концепции ЯОД разработаны CODASYL.

Язык схемы является спецификацией (набором правил, форматов и условий) общего ЯОД. Он не зависит от других языков, требуемых системе, оперирующей БД, но является общим для них. Унификация ЯОД схемы оказывает существенное влияние на разработку функционально совместимых СУБД и облегчает перенос программ с одной вычислительной системы на другую. Схема состоит из статей (оформленных совокупностей предложений) ЯОД и является полным описанием БД. Она содержит имена и описания всех областей, типов

наборов и записей, а также соответствующих элементов и агрегатов данных, существующих в БД и известных СУБД. Для описания подсхем используют, как правило, либо тот же ЯОД схем, либо подобный ему. Соотношения между схемой и подсхемой следующие.

Отдельный программист (пользователь) имеет дело не со всей БД, а только с ее частями, относящимися к программе (запросу), которую он создает. Так как БД может содержать данные, относящиеся ко многим применениям, такое разделение существенно облегчает работу программистов (пользователей) и снижает зависимость программ от изменений БД.

На основе любой схемы может быть объявлено произвольное число подсхем, причем объявление одной подсхемы не влияет на объявление других подсхем. Подсхемы могут перекрывать друг друга частично или полностью. Возможна ситуация, когда схема и подсхема совпадают. На одну и ту же подсхему может ссылаться произвольное число программ. Программа может использовать только области, записи, элементы и наборы данных, описанные в подсхеме, на которую она ссылается.

Схема, написанная на ЯОД, состоит из статей четырех типов:

- ***статьи схемы,*** которая именует схему и связывает с ней определенные средства СУБД (по доступу, защите и т. д.). Такая статья должна быть единственной в схеме и не подразделяться на подстатьи;
- ***статьи области,*** которая именует область и определяет ее атрибуты; каждая область в схеме описывается отдельной статьей и не подразделяется на подстатьи;
- ***статьи записи,*** которая именует тип записи, ее агрегаты и элементы данных и определяет их атрибуты; каждый тип записи в схеме описывают отдельной статьей; статья записи содержит одну подстатью записи, за которой следует произвольное число подстатей данных (в частности, они могут отсутствовать);
- ***статьи набора,*** которая именует тип набора и определяет его атрибуты; каждый тип набора в схеме описывают отдельной статьей; статья набора содержит одну ***подстатью набора***, за которой следует одна или несколько подстатей ***членов набора.***

Достаточно эффективным средством повышения независимости программ и данных является описание хранения данных, называемое ***схемой хранения данных.*** Схема хранения указывает, каким образом описанные в схеме данные могут быть организованы в терминах среды хранения, независимой от операционной системы и от устройств. Схема хранения может быть описана на языке описания хранения

данных (ЯОХД). Схема хранения создается для каждой схемы. Для описания хранения данных используется следующая терминология :

- ***область хранения*** — поименованное подразделение пространства памяти, выделенного для БД (она содержит хранимые записи и индексы, причем каждая хранимая запись и индекс целиком содержатся в одной области хранения);
- ***страница*** — индивидуально пронумерованный подраздел области хранения (область хранения состоит из целого числа страниц одинаковой емкости);
- ***хранимая запись*** — адресуемая единица хранения переменной длины, используемая для хранения данных и указателей, ассоциированных с одной записью схемы (хранимая запись обычно целиком размещается на одной странице);
- ***индекс*** — структура, с помощью которой обеспечиваются пути доступа к хранимым записям (индекс состоит из последовательности элементов, каждый из которых содержит указатель и, если это специфицировано, ключ);
- ***схема хранения*** — состоит из статей ЯОХД, описывающих области хранения, хранимые записи и индексы, соответствующие записям и наборам, описанным в схеме.

Кроме средств описания БД пользователю необходимы ***средства указания действий с данными.*** Такие указания выполняются с помощью специальных языков пользователя. Основными группами таких языков являются языки манипулирования данными (ЯМД) и интерактивные языки запросов (диалога).

Язык манипулирования данными программист использует для описания организации передачи данных между его процессом (выполнением одной или нескольких программ) и БД. ЯМД не является полным языком, он является подъязыком включающего его языка программирования. Язык программирования обеспечивает ЯМД процедурные средства, необходимые для манипулирования данными.

Интерактивные языки запросов (диалога) предназначены для спецификации запросов и функций манипулирования с данными при организации интерактивной работы пользователей с БД (запрос — ответ, диалог). Для работы пользователя с данными выделяют рабочие области (буферы) пользователя, являющиеся зонами загрузки и разгрузки. В рабочую область помещают данные, предназначенные для передачи в БД, и СУБД направляет данные по результатам запроса.
Общая функциональная структура АБД приведена на рис. 5.80.

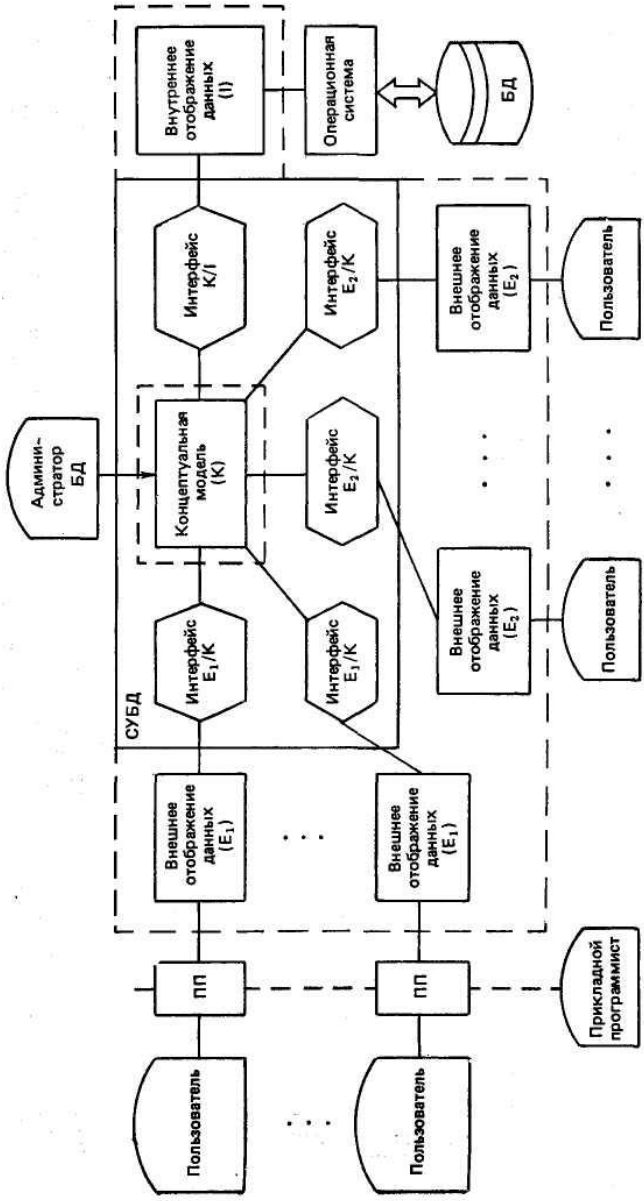


Рис. 5.80. Функциональная структура АБД

Пользователи АБД подразделяются на две основные категории: пользователи готовых программ и интерактивные пользователи.

Пользователи готовых программ свои запросы могут сформулировать заранее, и эти запросы реализуют в прикладных программах, разработанных программистами на языке, допускающем включение ЯМД.

Интерактивные пользователи формулируют свои запросы в процессе работы по мере необходимости. Для этого используют диалоговые (интерактивные) языки. Внешнее отображение данных (подсхема) предназначено для описания структуры данных, необходимых конкретному пользователю (группе пользователей).

Концептуальная модель (схема) описывает общую структуру данных для всех возможных (предполагаемых) пользователей. Преобразование осуществляется из представления пользователя (находится в рабочей области) в представление в базе данных. Преобразование касается только представления данных, а не их значения с тем, чтобы сохранить их смысл при реализации доступа с помощью включающих языков. Такие преобразования обеспечивают специальные программы, входящие в СУБД и являющиеся интерфейсами между внешним и концептуальным описанием данных. В СУБД имеется также программный модуль, обеспечивающий интерфейс между концептуальным и внутренним отображениями (описание физической структуры) данных. Поиск данных и манипуляции с ними в устройствах внешней памяти осуществляют соответствующие программы операционной системы.

Администратор (администрация) БД — это лицо (или группа лиц), осуществляющее создание, сопровождение и обслуживание БД и ее схемы с целью эффективного удовлетворения информационных потребностей пользователей.

Функционирование АБД. АБД функционирует в режимах генерации БД и эксплуатации. Схема функционирования АБД приведена на рис. 5.81. Генерация БД осуществляется на этапе организации БД, включающем следующие работы: составление схемы и ввод ее в СУБД (генерация схемы); составление схемы хранения и ввод ее в СУБД (генерация схемы хранения); загрузка БД; назначение паролей и ключей управления; назначение устройств и носителей для данных. Эти работы проводит администрация БД. Пользователи совместно с администрацией составляют подсхемы. Подсхемы вводят в СУБД (генерация подсхем).

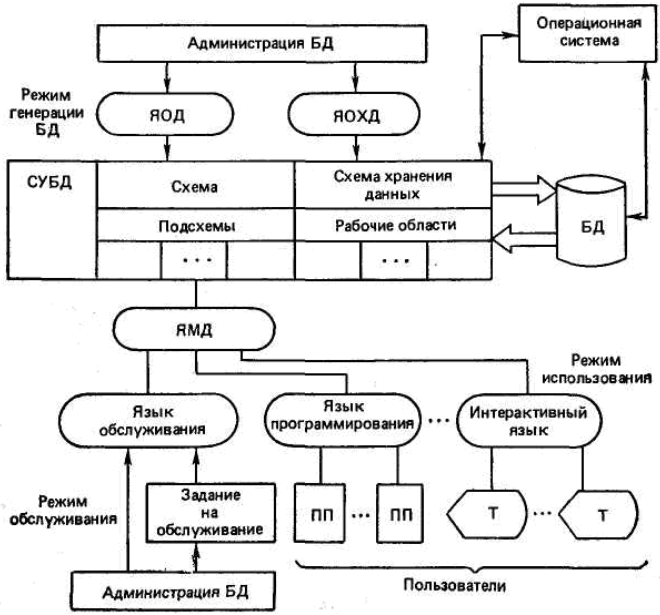


Рис. 5.81. Схема функционирования АБД: ПП — программа пользователя; Т — терминал .

Режим эксплуатации предназначен для использования и обслуживания БД.

Использование БД предполагает запуск программ по графику (в пакетном режиме) и по требованиям пользователей (разделение времени), ввод запросов пользователей и ведение диалога (реальное время) по поиску данных, формирование и выдачу результатов. Запрос к БД описывают с помощью ЯМД, который реализуют в рамках языков пользователя — языков программирования и интерактивных (диалоговых) языков. Запрос (так же как и выдача результатов) интерпретируется в рамках соответствующей подсхемы. СУБД выделяет прикладной программе рабочую область, в которую программа вводит параметры обращения, анализирует эти параметры (используя схему и подсхему) и обращается к операционной системе для организации ввода-вывода. Операционная система организует пересылку данных из внешней памяти в системные буферы и передает управление СУБД, которая пересылает данные из системных буферов в рабочую область.

Обслуживание БД предполагает решение следующих задач: наблюдения за использованием БД; реорганизации и реструктуризации БД; восстановления БД после сбоев системы.

В функции наблюдения входит контроль за действиями пользователей по обращению к БД, санкционированности доступа к информации, получения ответов, нарушений управления доступом. Для выполнения этих функций требуется организовать сбор статистики использования БД и ведение протокола изменений в БД.

В результате анализа собранной информации о работе с БД или появления новых требований может возникнуть необходимость изменения БД, т. е. проведения реорганизации и реструктуризации. При этом выполняют следующие работы: изменение схемы и схемы хранения (в том числе и их объектных модулей); изменение БД в соответствии со схемой и схемой хранения; удаление записей, ставших недоступными; уплотнение хранимых данных с целью использования освободившегося пространства; перераспределение данных на различные устройства и носители в соответствии с требованиями по времени и пространству; редактирование требуемой части БД.

Для восстановления БД требуется: выполнять дамп (вывод содержимого) частей БД на вспомогательную запоминающую среду; восстанавливать части БД, используя ранее полученный дамп или протоколы изменений.

Работы по обслуживанию АБД выполняет администрация БД. Для ввода заданий на обслуживание БД разрабатывают специальные языки обслуживания.

Особенности реализации АБД САК. Процесс взаимодействия пользователей с АБД реализуется выполнением совокупности транзакций, представляющих собой запросы на манипулирование данными и вызывающих последовательность операций чтения-записи. В АБД общего назначения транзакции относительно просты, изменения содержания БД легко регистрировать после завершения транзакций, действия транзакции, как правило, долговременны (данные, порожденные транзакцией, хранятся в БД сравнительно длительное время).

Под **консультационной транзакцией** следует понимать целый комплекс запросов (в общем случае недетерминированный), реализация которого приводит к получению законченной версии сформированной рекомендации. Продолжительность консультационной транзакции может быть весьма высокой (часы и даже дни). Все варианты сформированных рекомендаций до получения законченной версии должны временно храниться в БД и

восстанавливаться после разрушения, АБД общего назначения не приспособлены для хранения такой информации, к тому же при восстановлении БД после разрушения все незавершенные транзакции аннулируются.

В АБД САК должна быть реализована параллельная работа множества пользователей, поэтому СУБД должна иметь средства управления одновременным обращением к данным. В основном методе управления используется система замков, которые допускают обращение к определенным данным в конкретный момент времени только одному пользователю.

Анализ показывает, что АБД общего назначения не совсем удовлетворяют в полном объеме специфическим требованиям САК. Разработки по созданию СУБД САК в настоящее время ведутся в различных предметных (проблемных) обастях, и в скором времени будет возможна поставка и тиражирование таких систем.

База данных САК (особенно сложных) имеет большой объем, и работа с ней (доступ к данным) требует значительного машинного времени. Поэтому большие БД не реализуют в центральной ЭВМ, а применяют одну из следующих организаций: использование ЭВМ БД, реализация нескольких автономных БД, реализация распределенных БД.

Использование ЭВМ БД предполагает создание в центральном вычислительном комплексе (ЦВК) САК центрального АБД на специально выделенной для этих целей ЭВМ (рис. 5.82, *а)*. При этом к БД могут обращаться как ЭВМ ЦВК, так и пользователи, работающие на АРМК. Для целей коммуникации, управления очередями и сопряжением между ЭВМ ЦВК, ЭВМ БД и АРМК можно использовать препроцессор.

Автономные БД реализуют в ЦВК и АРМК (рис. 5.82, *б)*. Если ЦВК территориально распределен, то в нем может быть несколько автономных БД (несколько АБД). Такая структура характеризуется значительной избыточностью данных, но обладает наибольшим быстродействием.

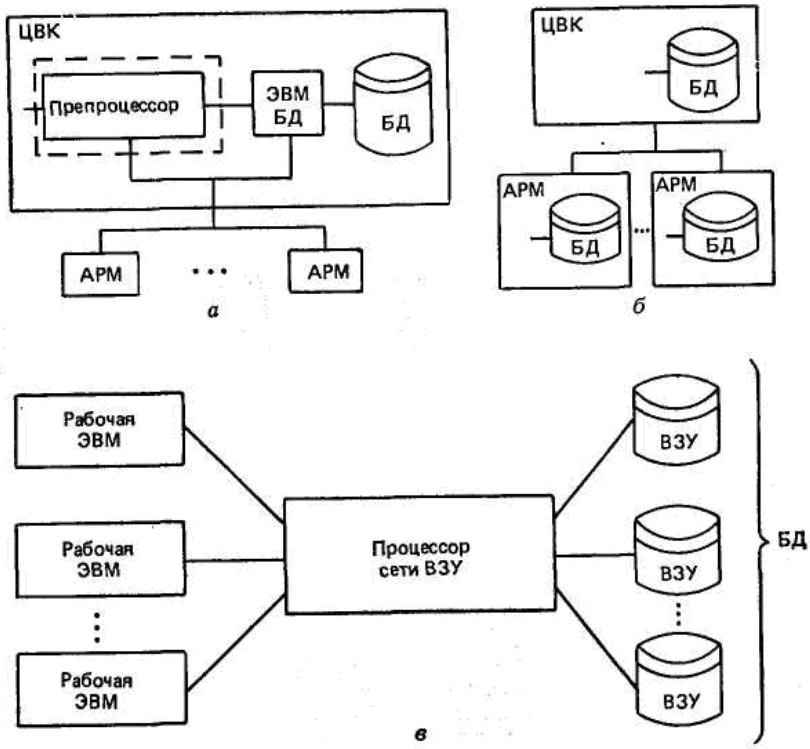


Рис. 5.82. Схемы использования ЭВМ БД (*а*); реализации автономных БД (*б*) и сети ВЗУ (*в*)

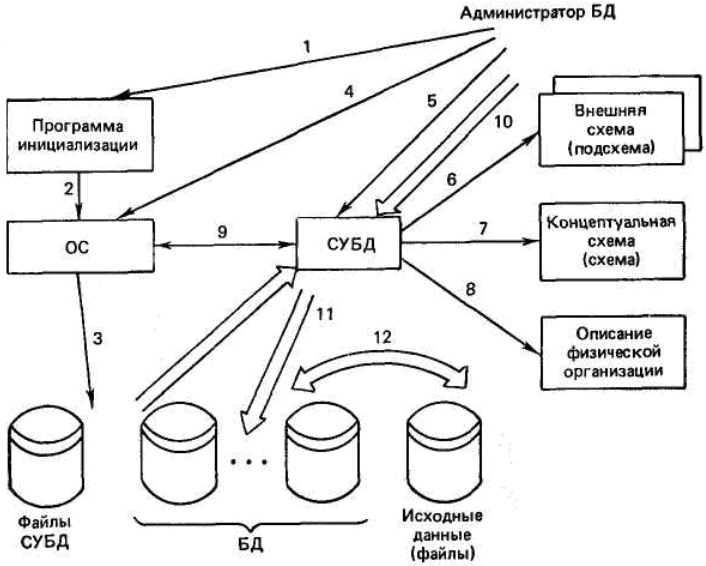
Распределенные БД реализуют на сетях ЭВМ (локальных или глобальных). При этом используют два способа хранения информации: разделение и дублирование (или их комбинации). Разделение информации между БД позволяет накапливать большие объемы данных, но время доступа может быть велико. Дублирование сокращает реакцию на запрос, но приводит к существенной избыточности данных. Процессы и протоколы обмена в распределенных БД реализуют средства управления. Распределенные БД можно создавать также на сети внешних запоминающих устройств (рис. 5.82, *в)*. В таких сетях необходимо иметь один или несколько процессоров, управляющих работой ВЗУ и осуществляющих связь с рабочими ЭВМ. В реальных САК может использоваться комбинация различных способов организации БД.

5.9.9. Система управления базами данных (СУБД)

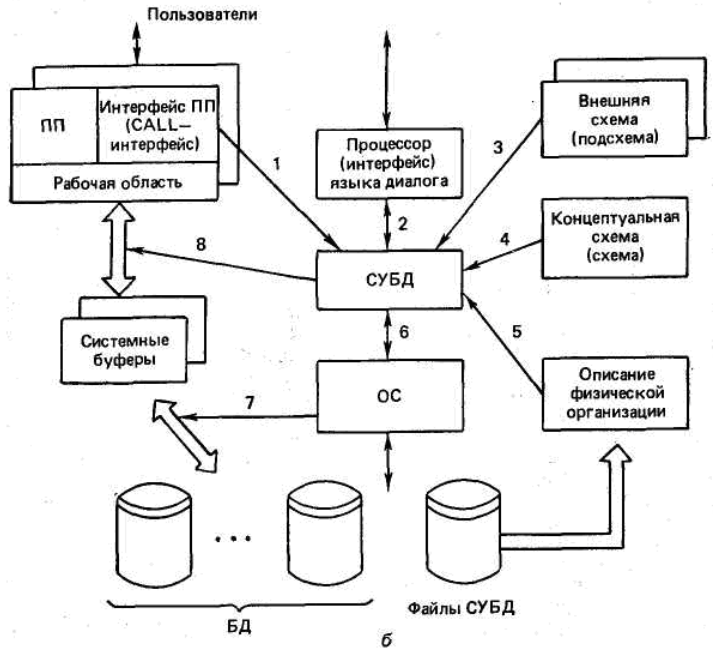
Одним из основных элементов программного обеспечения САК являются системы управления базами данных (СУБД). Их применение позволяет существенно упорядочить работу с данными в системе, обеспечить их защиту и целостность, резко сократить трудоемкость программирования многообразных процессов работы с ними. Рассмотрим основные характеристики СУБД, используемой в САК.

СУБД представляет собой комплекс программ, обеспечивающий централизованное хранение, накопление, модификацию и выдачу данных, входящих в БД. К основным функциям СУБД относят следующие: определение и инициализацию БД; организацию хранения данных; предоставление пользователям доступа к БД; защиту целостности БД; управление доступом к БД; поддержание функций системного персонала; поддержание технологического процесса функционирования АБД. Управление процессом функционирования осуществляется службой администратора АБД. В СУБД (а также в ППП окружения) для обеспечения процесса функционирования АБД включаются программные средства внутримашинной организации данных для следующих целей: ведения словаря-справочника данных, содержащего сведения о составе и свойствах БД; ввода, контроля и преобразования данных; документирования выдаваемых данных (генерации отчетов); загрузки и актуализации БД; ведения журнала для отображения функционирования системы с целью восстановления БД в случае ее разрушения; обеспечения интерактивного режима работы; сбора системной статистики для анализа функционирования; реструктуризации и реорганизации БД; восстановления БД; поддержки механизмов управления доступом; обеспечения стандартных выходов (интерфейсов) на программно-технические компоненты.

Можно выделить три основных группы функций СУБД: формирование БД; управление доступом к данным; актуализацию и сохранение БД. Схемы функционирования СУБД для этих групп приведены на рис. 5.83.


Администратор БД
1
4
5
10
Внешняя схема (подсхема)
Программа инициализации
2
6
ОС
9
СУБД
7
Концептуальная схема (схема)
8
3
11
12
Описание физической организации
Файлы СУБД
БД
Исходные данные (файлы)

а


Пользователи
ПП
Интерфейс ПП (CALL–интерфейс)
Рабочая область
1
Процессор (интерфейс) языка диалога
3
Внешняя схема (подсхема)
2
8
4
Концептуальная схема (схема)
СУБД
Системные буферы
6
5
7
ОС
Описание физической организации
БД
Файлы СУБД

б

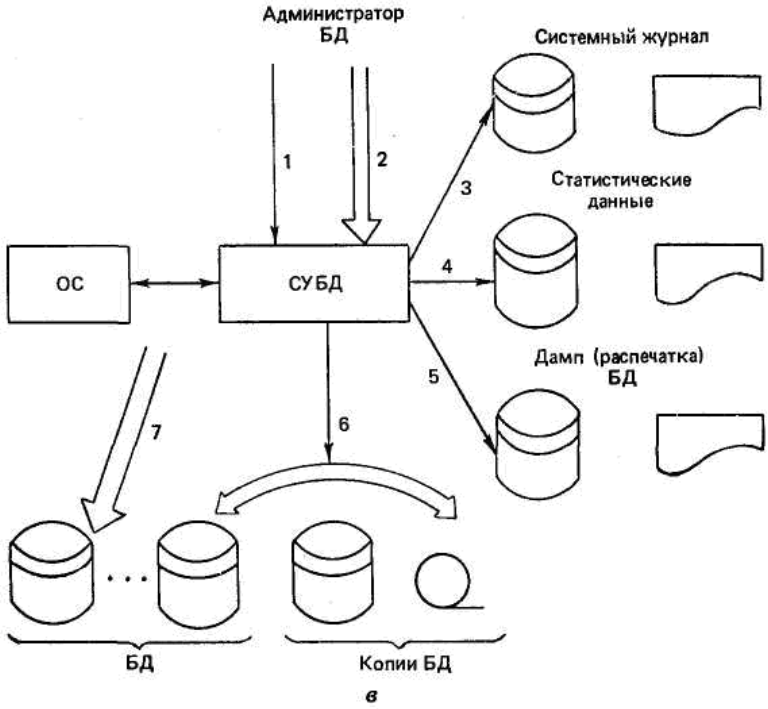


Рис. 5.83. Схемы функционирования СУБД по формированию БД (*а*); по управлению доступом (*б*); по актуализации и сохранению БД (*в*)

К функциям *формирования* БД относят следующие (рис. 5.83, *а*): *1* — выдачу запроса на инициализацию СУБД (запуск программы инициализации или специального командного файла); *2* — инициализацию и настройку СУБД; *3* — формирование загрузочных модулей СУБД и каталогизацию их в системе; *4*—запуск СУБД (запускается управляющая программа СУБД, в результате чего СУБД готова к работе); *5* — выдачу директивных запросов (запуск соответствующих программ) на генерацию схем, загрузку, контроль и корректировку БД; *6* — генерацию внешнего описания данных (внешней схемы, подсхемы); 7— генерацию концептуального описания (схемы); 5 — генерацию внутреннего описания (физической организации) данных; *9* — обращение к супервизору (для запуска системных средств) и передачу управления СУБД; *10* — ввод данных

по настройке СУБД и формирование БД; *11* — запись (корректировку) данных в БД; *12* — загрузку БД из исходных файлов.

Управление доступом к данным включает следующие основные функции (рис. 5.83, *б*): *1* — запрос на доступ к данным из прикладной программы; *2* — запрос—ответ по доступу к данным от интерактивных пользователей; *3* — использование подсхемы для описания данных, по которым получен запрос; *4* — использование схемы для определения логических типов данных; *5* — использование описаний физической организации БД для формирования запроса к ОС; *6* — обращение к ОС (супервизору) для реализации доступа и возврат управления СУБД; 7— обмен данными (записями) между физической БД (на ВЗУ) и системными буферами по реализации запроса; *8* — обмен данными с необходимыми преобразованиями (с учетом внешнего описания) между системными буферами и рабочей областью ПП по реализации запроса.

Актуализация и сохранение БД включают следующие основные функции (рис. 5.83, *в*): *1* — запрос к СУБД на выполнение функций актуализации и сохранения БД; *2* — ввод данных по запросу; *3* — ведение системного журнала, обращение к системному журналу, распечатка журнала; *4* — ведение, использование и распечатка статистических данных о работе АБД; 5 — вывод БД (дампирование, распечатка); *6* — копирование и восстановление БД; 7 — запись (корректировка) данных.

5.9.10. Целостность и сохранность баз данных

Эффективность САК и, в конечном итоге, качество сформированных рекомендаций в значительной степени зависят от достоверности и целостности БД. Достоверность предполагает соответствие хранимых данных реальному состоянию консультируемой проблемы (а точнее, состоянию модели консультируемой проблемы, сведения о которой хранятся и перерабатываются в системе). Под целостностью БД понимают отсутствие физических нарушений в хранимой информации.

Снижение достоверности и нарушения целостности могут быть вызваны ошибками в данных и разрушением данных. Основными источниками ошибок и разрушения данных являются: неточные исходные данные (вводимая информация); искажение и потери информации при подготовке и вводе ее персоналом САК; искажение информации при работе оборудования (неустойчивость характеристик, сбои, помехи); неисправность и отказы оборудования; некорректность программных средств; несанкционированный и некомпетентный доступ к данным.

Основными мерами поддержания достоверности и целостности БД являются контроль информации, исправление ошибок и искажений, обеспечение сохранности БД. Эти меры реализуются следующими методами и средствами: визуальными (при подготовке, вводе и выводе информации); техническими (аппаратными); программными и организационными.

Контроль информации. Для обнаружения ошибок существенное значение имеют четыре фактора: грамматический характер ошибок, уровень агрегации (иерархии) контролируемых элементов, вид и форма используемой для контроля избыточности, момент внесения избыточности в контролируемые элементы.

По грамматическому характеру ошибки подразделяются на синтаксические и семантические.

Синтаксические ошибки влияют на структуру и лексику представления данных, например пропуск или добавление символа, внесение постороннего или запрещенного символа в реквизит, сдвиг по строкам или колонкам отформатированного документа (сообщения).

Для обнаружения синтаксических ошибок применяют сравнительно простые методы, например, проверку шаблонов и форматов.

Семантические ошибки искажают смысл (содержание) информации, оставляя правильными структуру и лексику, например, замена символов другими допустимыми символами. Семантические ошибки разделяют на три вида: орфографические, смысловые и блочные.

Орфографические ошибки искажают значения символов реквизита.

Смысловые ошибки искажают содержимое (смысл) записи, например, перемена местами одноименных реквизитов различных записей.

Блочные ошибки искажают блок записей (например, отсутствие некоторых записей, их перестановки). Следовательно, можно выделить три уровня агрегации контролируемых элементов: реквизит, запись, блок записей (или файл).

Для контроля информации используют следующие виды избыточности:

избыточные разряды (например, контроль по модулю, при котором контрольные разряды определяются путем преобразования значений информационных, контроль по четности);

избыточные реквизиты, дополняющие запись (они вычисляются как некоторая функция от значений информационных реквизитов, например, контрольное суммирование по записи);

избыточные записи, дополняющие блок (например, итоговое суммирование);

двукратную избыточность, заключающуюся в повторной фиксации информации и сравнении ее с первоначальной (например, дубль-копирование, верификация);

естественную избыточность, основанную на свойстве конкретного вида информации иметь определенные формат, размерность, диапазон разрешенных значений, а также определенным образом сочетаться между собой (например, реквизит «месяц» имеет разрешенный диапазон 01—12).

Внесение ошибок (искажение информации) возможно на следующих основных этапах переработки данных: заполнения документов (сбора данных); подготовки данных на машинных носителях; ввода данных; обработки данных; записи (считывания) информации на внешние запоминающие устройства; передачи данных. На этапе заполнения возможен визуальный контроль; на этапе подготовки — визуальный, повторная подготовка и сравнение, контрольное суммирование; на этапе ввода данных — визуальный (по отображению вводимой информации), технический (по четности, по модулю), программный (логический по формату, структуре, диапазону и по контрольной сумме); на этапе обработки — технический (предусмотренный в аппаратуре ЭВМ) и программный (логический, двойная обработка и сравнение); на этапе записи (считывания) — в основном аппаратные средства (на четность, циклический, контрольное суммирование); на этапе передачи — аппаратные (с помощью устройств защиты от ошибок, обеспечивающих помехозащитное и исправляющее некоторые ошибки кодирование и контроль), а также программные, если передача осуществляется в (из) ЭВМ.

На всех этапах могут применяться организационные методы контроля, заключающиеся в осуществлении, сочетании, сопоставлении результатов других методов, организации исправления ошибок, а также принятия решений о возможности продолжать процесс переработки данных.

Исправление ошибок и искажений. При обнаружении искажения данных в БД возможны два пути корректировки: корректировка отдельных записей (их элементов) и осуществление мер по обеспечению сохранности БД.

Исправление (корректировка) отдельных записей осуществляется, если записи локализованы и если искажена незначительная часть БД. В противном случае пользуются системой обеспечения сохранности БД.

Обеспечение сохранности БД. Основными мерами обеспечения сохранности являются: копирование и восстановление БД; защита от несанкционированного и некомпетентного воздействия; организационно-методическая работа.

Для копирования и восстановления БД применяется целый ряд общих методов.

Полное копирование томов предполагает снятие полных копий томов памяти. *Восстановление* заключается в перезаписи копии на том (иногда перезаписывается часть тома). Сохраняется, как правило, последняя копия. Основные недостатки: большие временные затраты и невозможность использования БД во время копирования.

Выборочное копирование (копирование файлов) предполагает снятие копий только модифицированных файлов, которые добавляются к уже имеющимся копиям. Основной недостаток: трудоемкость учета и периодической чистки файлов.

Системная журнализация предполагает фиксацию (протоколирование) действий по модификации БД в системном журнале (специальный файл) и хранение корректур. Выполняется обычно средствами СУБД и используется в сочетании с другими методами копирования-восстановления, облегчая их реализацию за счет наличия предыстории модификации. Основные недостатки: относительно большое время восстановления, а также трудоемкость реализации системной журнализации (при использовании уже разработанной СУБД с этими функциями данный недостаток не имеет значения).

Регенерация поколений предполагает копирование по схеме «дед — отец — сын», т. е. наличие трех текущих копий. Самая последняя копия называется «сын», предыдущие — соответственно «отец» и «дед». Если происходит очередное копирование, появляется новое поколение «сын», бывшее поколение «сын» становится «отцом», бывшее поколение «отец» — «дедом», а копия бывшего поколения «дед» уничтожается. При разрушении оригинала (копии «сын») восстановление осуществляется путем повторения процедур актуализации. Если разрушена копия «отец», то восстановление осуществляется в два этапа: «дед», «отец» — «сын». Для восстановления существенное значение имеет системная журнализация.

Предварительное замещение ориентировано на восстановление информации, разрушенной во время сеанса актуализации БД. Предполагает создание копии модифицируемой части перед выполнением модификации, сама модификация выполняется с копией.

Если информация во время модификации будет разрушена, оригинал останется в сохранности.

В реальных системах информационного обеспечения наиболее эффективно использовать комбинации общих методов копирования — восстановления. Наиболее распространены три основных класса комбинированных моделей копирования-восстановления: периодической разгрузки, дублирования-регенерации поколений, периодической разгрузки-регенерации поколений.

Защита от несанкционированного и некомпетентного воздействия (запроса) основывается на системе *замков* и *ключей,* идентифицирующих пользователей. Система реализуется, как правило, в рамках СУБД.

Организационно-методические работы по обеспечению сохранности БД проводит администрация БД. В состав работ входят: анализ требований пользователей по защите данных; идентификация пользователей (назначение паролей, ключей); определение функций (операций) над данными, допустимых для каждого пользователя, и фиксация этой информации в СУБД; контроль и анализ нарушений санкционированного доступа; анализ системного журнала (выявление особых ситуаций, необходимости корректировки, копирования, восстановления); проверка логической непротиворечивости информации БД (запуск специальных тестирующих программ по графику или по необходимости, анализ результатов тестирования); контроль модификаций информации, проводимых пользователями в БД; организация корректировки БД (службой администрации БД и пользователями); организация копирования БД (подготовка носителей, запуск программ копирования по графику или внепланово, регистрация и учет копий); организация восстановления (подготовка носителей, запуск программ восстановления по графику или внепланово, протоколирование восстановления); методическая работа с пользователями программ, программистами и интерактивными пользователями по повышению эффективности взаимодействия с БД (выбору наилучших стратегий поиска), приобретению правильных навыков взаимодействия с БД (повышение компетентности), поддержанию достоверности и целостности БД.

5.9.11. Организация данных по типовым рекомендациям и элементам

Процесс стандартизации в области САК включает два подпроцесса: нормативное упорядочение деятельности по созданию САК и формирование нормативно-технического потенциала типовых средств создания САК.

Нормативное упорядочение деятельности по созданию САК направлено на выявление структурных характеристик (элементов, свойств и отношений) процессов и результатов консультирования, представление этих характеристик в виде нормативного конечного результата и типовой последовательности состояний консультационного процесса и их отображение в множество нормативных рекомендаций, определяющих содержание рекомендаций.

Формирование научно-технического потенциала типовых средств создания САК направлено на выявление состава и разработки типовых для процессов стандартизируемой деятельности нормативных элементов (методов, рекомендаций, частей конечного результата) и их отображение в виде стандартов, нормативных и методических документов. Типовые средства подразделяют на две составляющие — типовые методы (ТМ) и типовые рекомендации (ТР).

Типовые методы предназначены для упрощения реализации и ведения процессов автоматизированного (автоматического) консультирования, тиражирования различных САК, обеспечения большей гибкости системы. ТМ представляют собой оформленные для привязки к конкретным условиям методы и модели, в которых интерфейсы (связи) унифицированы. К ТМ относятся методы формирования рекомендаций (выбор варианта, наиболее соответствующего требованиям решения задачи консультируемой проблемы и стратегии консультирования, генерация вариантов рекомендаций, расчет консультационных параметров и т. п.); модели консультируемых проблем (математические, цифровые, имитационные); методы оценки сформированных рекомендаций (оценка эффективности, качества, надежности и т. п.); методы описания и отображения сформированных рекомендаций (компоновка консультационной документации, редактирование текста и графики, оперативное визуальное отображение, представление и вывод текстовых и графических документов). Методы оценки сформированных рекомендаций могут входить в состав методов формирования рекомендаций и моделей консультируемых проблем. Вне машины ТМ представляют в виде компонентов методического (в том числе и математического) обеспечения: постановки консультационной задач, алгоритмы, методики, инструкции. Внутри машины ТМ представляют в виде программ (комплекса программ) и данных, необходимых для работы этих программ.

Типовые рекомендации представляют собой оформленные для использования при автоматизированном (автоматическом) консультировании представления (описания) возможных компонентов

консультируемых проблем, в которых унифицированы однородные параметры, структура и интерфейсы. К ТР относят типовые проекты сформированных рекомендаций, типовые рекомендации (ТР) и типовые элементарные рекомендации (ТЭР).

Типовой проект сформированных рекомендаций представляет собой полностью оформленное консультационное описание типового (т. е. тиражируемого или имеющего достаточно широкое распространение) объекта-аналога, в котором предусмотрены возможности варьирования в допустимых пределах параметрами, характеристиками, компонентами и составными частями с целью привязки к заданным условиям.

Типовая рекомендация представляет собой оформленное описание относительно самостоятельной части (компонента) объекта для всего процесса консультирования или его части (этапа) с возможностью варьирования указанными характеристиками в заданных пределах с целью привязки рекомендации к заданным условиям. ТР могут быть расчетно-описательными, графическими и смешанными (т. е. включающими и расчетно-описательную, и графическую часть). ТР могут иметь следующее назначение — являться базовыми составляющими для выработки рекомендаций или основой для компоновки консультационной документации; использоваться комплексно как для выработки рекомендаций, так и для компоновки консультационной документации.

Типовой элемент рекомендации представляет собой определенным образом оформленное описание элементарной части (компонента) объекта или фрагмента рекомендации, используемое в качестве исходного (совместно с другими ТЭР соответствующего назначения) для синтеза рекомендаций и документов. ТЭР имеют более ограниченный (по сравнению с ТР) набор варьируемых параметров (например, типоразмер, изменяемая характеристика ряда, масштаб). ТЭР могут быть характеристическими (ТЭРХ) и графическими (ТЭРГ).

ТЭРХ содержит описание параметров, характеристик компонента или фрагмента, а также условий и ограничений по их применению.

ТЭРГ содержит описание графического изображения фрагмента чертежа, схемы или рисунка. В зависимости от возможных изменений формы и составляющих выделяют следующие группы ТЭРГ: графические константы (ТЭРГ-К) — элементы, формы начертания и размеры которых не изменяются (например, обозначения условные графические, изображения основных надписей чертежа); масштабируемые константы (ТЭРГ-М) имеют постоянный набор составляющих, размеры которых можно изменять в допустимых

пределах; параметрические элементы (ТЭРГ-П) имеют переменный состав компонент с постоянными или изменяемыми размерами, который зависит от входных условий (параметров).

Вне машины ТР хранятся в виде специально оформленных документов, являющихся текстово-графическими описаниями соответствующих рекомендаций. Эти документы не являются обязательными, и с повышением надежности машинного хранения, емкости носителей ВЗУ, расширением возможностей отображения документальная форма ТР исчезнет.

Внутри машины ТР могут храниться в общей БД (типовые проекты, ТР), в БД пользователя (ТР, ТЭРХ), в специальной БД (ТР, ТЭРХ, ТЭРГ), в специальных библиотеках (ТЭРХ, ТЭРГ, графические ТР). Все ТР должны быть каталогизированы. Типовые рекомендации целесообразно снабжать паспортами (рефератами), в которых должны быть указаны все параметры рекомендации, компоненты или объекта. Паспорт ТР используется при реализации рекомендаций, просмотре и анализе ТР. Описательная часть ТР используется для компоновки документации.

5.9.12. Рекомендации по проектированию информационного обеспечения

При проектировании ИО САК решают следующие основные задачи: определение функций и требований; разработка принципов построения и структуры системы ИО; определение структуры, содержания и характеристик данных; выбор (разработка) СУБД; описание логической структуры и структуры хранения данных, подготовка каталогов — перечней данных; подготовка к испытаниям и проведение испытаний; подготовка документации.

Управляющая часть ИО (комплекс программ) является, как правило, инвариантным средством САК. Поэтому при проектировании необходимо в максимальной степени учесть возможности применения уже разработанных, поставляемых и апробированных средств, особенно СУБД. Тип используемой СУБД определяют либо выбором, либо директивно. Директивное назначение СУБД характерно для отраслей (объединений, организаций), в которых осуществляется единая техническая политика информационного обслуживания и обеспечения.

Решение о выборе или разработке СУБД принимают на *предпроектной стадии* на основе *технико-экономического обоснования*. На этой стадии определяют функции и требования к ИО, а на их основе обосновывают структурные принципы построения системы ИО (разобщенный, централизованный фонд или АБД).

Задание на разработку АБД может входить в ТЗ на САК, а может быть оформлено в виде частного ТЗ.

В результате информационного обследования организации — пользователя САК и анализа информационных потребностей предполагаемого персонала определяют состав, структуру и характеристики данных, информационные объекты и их атрибуты, модель предметной области (предварительная инфологическая модель). На основании такой модели строят оценочные варианты БД, покрывающей информационные потребности САК.

Возможность применения промышленно сопровождаемых (или разработанных) СУБД определяют на основе оценочных вариантов БД, требований пользователей, технических, функциональных, экономических и сервисных характеристик СУБД. По результатам оценки возможностей реализации БД осуществляют выбор СУБД либо принимают решение о невозможности выбора и необходимости разработки оригинальной СУБД или доработке одной из поставляемых (разработанных) СУБД.

Оценка возможностей использования СУБД является достаточно трудной задачей по следующим причинам: условность оценочных вариантов БД на стадии предпроектных исследований, неоднородность и неоднозначность требований пользователей, сложность привязки критериев выбора (затраты, быстродействие, достоверность, надежность и т. д.) к характеристикам СУБД. Существенным фактором выбора могут стать косвенные «затраты» на использование АБД (технические, экономические, организационные), к которым, в частности, можно отнести затраты: на управление АБД (использование ресурсов вычислительных комплексов на функционирование СУБД и операционных систем); на реализацию запросов данных из прикладных программ и в диалоге в требуемом языке с соблюдением необходимых ограничений; на создание службы (подразделения) администрации БД, ответственной за эксплуатацию и обслуживание АБД; на поддержание достоверности, а также на организацию сохранности БД.

Технико-экономическое обоснование (ТЭО) АБД (в общем случае всей системы ИО) САК документируют в виде разделов общего документа ТЭО САК. Полученные в результате предпроектного обследования материалы по АБД (системе ИО) оформляют как приложение к ТЭО САК. Это приложение может содержать следующие разделы: описание инфологической модели; описание информационных потребностей пользователей; описание схемы документооборота, замыкающегося на АБД (систему ИО); обоснование выбора состава программных средств АБД.

В *частное* ТЗ на АБД (систему ИО) либо в раздел по ИО общего ТЗ на САК включают следующие разделы (подразделы): назначение АБД; основные требования к АБД, основные технические предложения; технико-экономические показатели АБД; состав, содержание и организация работ по созданию АБД; порядок приемки АБД (системы ИО).

На стадии *технического проекта* выполняют следующие основные работы:

- уточнение информационной модели предметной области (детализация элементов модели, устранение избыточности, привязка к выбранной СУБД);
- логическое проектирование БД (определение состава информации, структуризация данных, разработка концептуальной модели, представление связей средствами СУБД);
- физическое проектирование (определение способов организации данных, состава и формата записей и файлов, разработка внутренней схемы, оценка объемных характеристик и распределение данных и файлов по томам);
- проектирование внешнего представления данных (разработка внешних схем для каждой группы приложений);
- разработку состава функций, поддерживаемых средствами СУБД и ППП, а также оригинальными прикладными программами;
- определение параметров генерации и настройки СУБД (с целью выполнения всех требуемых функций);
- постановку оригинальных задач и разработку алгоритмов;
- расчет загрузки оборудования (КТС) средствами АБД;
- разработку организационной структуры службы администрации БД, структурной схемы взаимодействия подразделений с системой ИО, процедур, выполняемых пользователями и обслуживающим персоналом АБД;
- разработку каталогов-перечней данных.

На стадии *рабочего проекта* выполняют следующие основные работы:

- разработку оригинальных программных средств (в соответствии с алгоритмами, разработанными в техническом проекте);
- настройку СУБД и ППП на область применения, включая генерацию всех уровней описания БД;
- разработку контрольных примеров для многоцелевого тестирования
- разработку должностных и технологических инструкций (для служб, подразделений и лиц, использующих АБД).

На стадии *ввода в эксплуатацию* выполняют:

- ввод в действие АБД;
- опытную эксплуатацию АБД;
- ввод в действие АБД в составе САК;
- эксплуатацию АБД в составе САК;
-сдачу АБД (системы ИО) в промышленную эксплуатацию.

5.10. Техническое обеспечение САК

5.10.1. Общие требования

Техническое обеспечение САК представляет собой совокупность взаимосвязанных технических средств (ТС), предназначенных для выполнения автоматизированного консультирования. Структурное единство компонентов технического обеспечения, обеспечивающих функционирование подсистем САК, составляет комплекс технических средств (КТС) САК. Компонентами технического обеспечения являются устройства и системы (сочетания устройств), создаваемые на базе средств вычислительной, организационной, измерительной техники и передачи данных.

Комплекс технических средств САК должен создаваться на базе серийно выпускаемых ТС с применением стандартных программно-аппаратных интерфейсов. При надлежащем техническом и экономическом обосновании могут применяться и специализированные ТС.

Требования к техническому обеспечению САК можно разделить на четыре категории: системные, функциональные, технические и организационно-эксплуатационные.

Системные требования обуславливают спектр свойств, параметров и характеристик КТС САК как технической системы.

Функциональные требования обуславливают свойства КТС с точки зрения выполнения функций САК. Здесь рассмотрены наиболее общие требования к техническому обеспечению.

Технические требования определяют параметры и характеристики КТС и отдельных ТС при функционировании САК.

К ***организационно-эксплуатационным*** относятся требования по технической эстетике, эргономике, безопасности (охрана труда), организации эксплуатации и обслуживания ТС.

Системные требования. К КТС САК предъявляют следующие системные требования: эффективность, универсальность, совместимость, гибкость и открытость, надежность, точность (достоверность), защищенность, возможность одновременной работы достаточно широкого круга пользователей, приемлемая стоимость.

Эффективность. КТС в совокупности с информационным и программным обеспечением САК должен обеспечивать эффективное выполнение персоналом САК всей совокупности функций автоматизированного консультирования с целью получения достаточно качественных (по возможности оптимальных) рекомендаций и консультационной документации в приемлемые сроки.

Универсальность. ТС САК должны быть достаточно универсальны, чтобы обеспечить максимально возможную реализацию совокупности инноваций и изменений по консультируемой проблеме (серии проблем) в течение всего цикла консультирования без перестройки КТС.

Совместимость. Средства, входящие в КТС САК, должны обладать технической, информационной, программной и эксплуатационной совместимостью. Путем достижения совместимости средств обеспечивается нормальное функционирование, развитие и тиражирование всего комплекса.

Гибкость и открытость. Структура КТС САК должна быть гибкой, т. е. допускать перестройку в достаточно широких пределах, и открытой, т. е. допускать замену устаревших средств, их модернизацию и расширение состава. Обеспечение гибкости и открытости позволяет осуществлять модернизацию и развитие САК (что особенно важно при интенсивных инновациях, вносимых консультируемыми проблемами), а также тиражирование САК.

Надежность. КТС САК должен обладать надежностью, достаточной для нормального функционирования в течение всего цикла консультирования. К показателям надежности ТС САК относят среднюю наработку на отказ, среднее время восстановления, средний срок службы, средний срок сохраняемости, коэффициент технического использования. Эти показатели для серийно выпускаемых технических средств заранее известны. И если они не позволяют обеспечить требуемую надежность КТС в целом, необходимо применять системные методы повышения надежности (резервирование, дублирование), эффективные методы восстановления работоспособного состояния, а также средства обеспечения сохранности информации при отказах ТС.

Точность (достоверность). При функционировании КТС САК должен обеспечить требуемый уровень точности (достоверности) формируемых рекомендаций и данных (информации в целом). Точность (достоверность) зависит от достоверности входной информации (точности исходных данных и достоверности ввода), точности ТС (разрядности, методов преобразования, округления и т.

п.), сбоев в оборудовании, отказов ТС и защищенности от внешних воздействий. Для повышения точности (достоверности) информации применяют различные организационные, технические и программные методы и средства контроля, обнаружения ошибок, обеспечения сохранности и восстановления информации.

Защищенность. Комплексы средств САК должны быть защищены от внешних воздействий (помех, сбоев в системе питания, некомпетентного и несанкционированного вмешательства) так, чтобы не нарушалось нормальное их функционирование.

Возможность одновременной работы, достаточно широкого круга пользователей. КТС должен позволять реализовать САК, являющуюся системой коллективного пользования для достаточно большого коллектива специалистов (разработчики САК, консультанты, обслуживающий персонал, административно-управленческий персонал организации-пользователя). Причем терминалы пользователей и вычислительные ресурсы могут быть разнесены территориально на большие расстояния.

Приемлемая стоимость. Стоимость КТС должна быть такая, чтобы созданная на его базе САК обеспечила наибольший или приемлемый (в зависимости от целей создания) экономический эффект.

Функциональные требования. К функциональным требованиям можно отнести (предъявляются к комплексам средств САК и обеспечиваются КТС, программными средствами и информационной базой): реализацию математических моделей (консультируемых проблем, чертежей, функционирования проблем); задач формирования рекомендаций и консультационных процедур; архивов, библиотек сформированных рекомендаций и типовых элементов; системы поиска данных, обеспечение наглядности информации; работы с графическими изображениями и моделями; параллельной разработки отдельных рекомендаций; взаимосвязи этапов консультирования; работы пользователя как в пакетном, так и в диалоговом (в частности, запрос-ответ) режиме с возможностью перехода с одного режима на другой на любом этапе консультирования; документирования результатов консультирования (промежуточных и конечных) с необходимой полнотой и в требуемой форме.

Технические требования вырабатывают в процессе разработки комплексов средств САК и предъявляют к группам и отдельным ТС. Технические требования выражают в виде количественных, качественных и номенклатурных значений характеристик и параметров. К основным характеристикам и параметрам относят следующие: производительность, быстродействие и пропускную

способность оборудования, разрядность устройств (количество разрядов регистров для представления информации); систему кодирования информации; форматы внутреннего представления данных и команд; форматы внешнего представления и отображения данных; разрешающие способности средств отображения и регистрации данных; емкость запоминающих устройств (оперативных, постоянных и внешних); виды носителей данных; типы интерфейсов для сопряжения оборудования.

При необходимости разработки специализированных технических средств для них разрабатываются частные технические задания, в которых указывают все необходимые для разработки, изготовления и испытания этих ТС требования. Для средств вычислительной техники общие технические требования и методы испытаний должны соответствовать действующим стандартам.

Организационно-эксплуатационные требования предъявляются к КТС, вспомогательному оборудованию, рабочим местам, помещениям и персоналу с целью обеспечения нормальных условий эксплуатации и обслуживания САК. Выделяют следующие группы организационно-эксплуатационных требований:

- эргономика и техническая эстетика (требования приведены в соответствующих стандартах);
- безопасность персонала при эксплуатации (требования электробезопасности и пожарной безопасности);
- подготовка персонала (уровень обученности и квалификации персонала должен быть достаточным для обеспечения нормальной эксплуатации и обслуживания САК);
- централизованное техническое обслуживание (комплексы средств САК, за исключением специальных, должны в максимальной степени обеспечиваться централизованным техническим обслуживанием и сопровождением);
- ремонтопригодность (требования по организации, технологии, материальному обеспечению технического обслуживания и ремонта);
- климатические условия помещений (требования по температуре, влажности, атмосферному давлению);
- звукоизоляция (уровень звука, создаваемого при работе ТС, не должен превышать значений, установленных Гигиеническими нормами звукового давления и уровней звука на рабочих местах; для помещений должны быть установлены требования по звукоизоляции, планировке и размещению оборудования, обеспечивающие допустимый уровень звука в зоне расположения персонала);

- типовая планировка и размещение (необходимо в максимальной степени использовать типовые варианты планировки и размещения оборудования, обеспечивающие оптимальные условия эксплуатации).

Наиболее общие требования (в большей части системные и функциональные) приводят в техническом задании на САК. Более детализированные и конкретизированные системные и функциональные требования, а также технические и организационно-эксплуатационные требования указывают в технических заданиях на комплексы средств.

5.10.2. Состав, структура и классификация технических средств САК

Классификацию ТС САК (рис. 5.84) проводят по двум признакам: функциональному (функциональные группы ТС) и структурному (комплексы средств).

По **функциональному признаку** выделяют следующие группы ТС: подготовки и ввода данных; передачи данных; программной обработки данных; отображения и документирования данных; архива сформированных рекомендаций.

Группа ТС подготовки и ввода данных предназначена для автоматизации подготовки, ввода, первичной обработки и редактирования исходных и нормативно-справочных данных для автоматизированного консультирования, а также для ввода запросов и директив САК. ТС подготовки и ввода данных должны обеспечивать кодирование информации, нанесение данных на машинные носители, ввод данных в ЭВМ, визуальный контроль и редактирование данных при вводе и подготовке алфавитно-цифровой и графической информации, а также запросов и директив. Для выполнения указанных функций применяют различные устройства подготовки и ввода данных: устройства подготовки данных на машинных носителях; устройства ввода данных с машинных носителей; устройства ввода графической информации (графоповторители или диджитайзеры); сканирующие устройства; клавиатуры (чаще всего придисплейные) алфавитно-цифровые, функциональные, специальные клавиатуры, реализуемые на экране дисплея, и фотоселекторные средства.

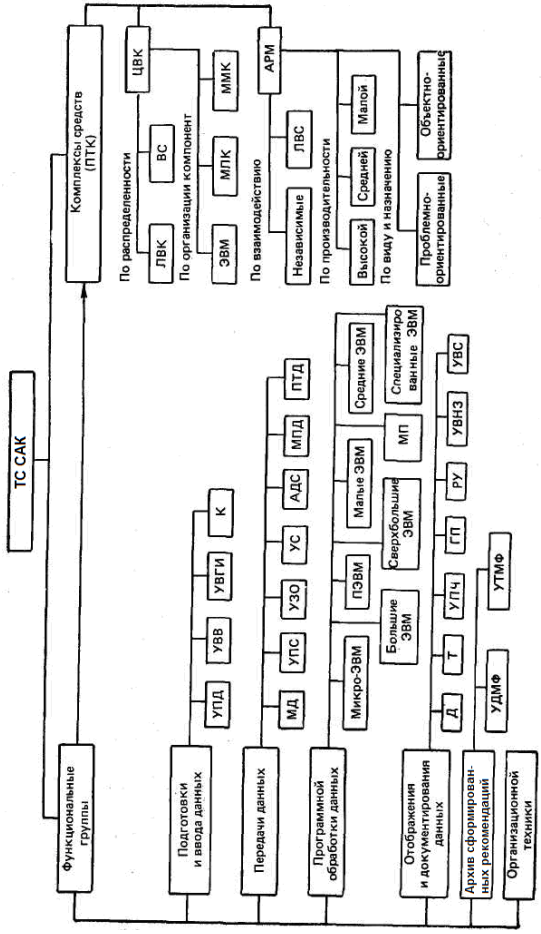


Рис. 5.84. Структура классификации ТС САК:

УПД — устройство подготовки данных; УВВ — устройство ввода; УВГИ — устройство ввода графической информации; К — клавиатура; МД — модем; УПС — устройство преобразования сигналов; УЗО — устройство защиты от ошибок; УС — устройство сопряжения; АДС — адаптер дистанционной связи; МПД — мультиплексор передачи данных; ПТД — процессор телеобработки данных; ПЭВМ — персональная ЭВМ; МП — микропроцессор; Д — дисплей; Т — табло (мнемосхема); УПЧ — устройство печати; ГП — графопостроитель; РУ — регистрирующее устройство; УВНЗ — устройство вывода на носители записи; УВС — устройство вывода специализированное; УДМФ — устройство доступа к микрофильмированным документам; УТМФ — устройство тиражирования микрофильмированных документов; ЛВК — локальный вычислительный комплекс; ВС — вычислительная сеть; МПК — многопроцессорный комплекс; ММК — многомашинный комплекс; ЛВС — локальная вычислительная сеть

Группа ТС передачи данных предназначена для обеспечения дистанционной связи средств САК по каналам связи. Устройства этой группы должны обеспечивать передачу данных между удаленными компонентами САК по телефонным, телеграфным и специальным каналам связи. В целях выполнения временных требований и ограничений следует применять выделенные каналы связи. К устройствам рассматриваемой группы относят аппаратуру передачи данных (модемы, устройства преобразования сигналов, устройства защиты от ошибок) и аппаратуру сопряжения и концентрации (устройства сопряжения, адаптеры дистанционной связи, мультиплексоры передачи данных, процессоры телеобработки данных).

Группа ТС программной обработки данных предназначена для обеспечения приема цифровых данных, их программной обработки, накопления и вывода на машинные носители, устройства отображения и в каналы связи. К устройствам этой группы относят ЭВМ общего назначения (микро-ЭВМ, персональные, малые, средние, большие и сверхбольшие ЭВМ), специализированные ЭВМ и микропроцессоры. ТС программной обработки должны обеспечивать разработку и эксплуатацию программного обеспечения САК, изменение производительности путем замены или наращивания ЭВМ, использование программно-аппаратурных средств учета, хранения и выдачи текущего времени, мультипрограммный режим работы.

Группа ТС отображения и документирования данных предназначена для оперативного представления сформированных рекомендаций и запрашиваемых данных, а также для вывода консультационной документации и промежуточных носителей для консультирования проблем. К ТС этой группы относятся: устройства визуального отображения информации (алфавитно-цифрово-графические дисплеи, панели и табло отображения информации, мнемосхемы); устройства вывода информации на бумагу (устройства печати, графопостроители, регистрирующие устройства); устройства вывода информации на микрофильмы и микрофиши; устройства вывода на машинные носители записи (диски, флеши и др.); устройства вывода специального назначения (координатографы, фотонаборные устройства и т. д.).

ТС визуального отображения информации при необходимости должны комплектоваться совместно с устройствами документирования. ТС документирования данных должны обеспечивать выпуск документов на печатающих и графических устройствах на правах подлинников или в качестве оригиналов для выпуска подлинников, соответствующих требованиям стандартов.

Группа ТС архива сформированных рекомендаций предназначена для обеспечения хранения, контроля, восстановления и размножения данных о сформированных рекомендациях, а также справочных данных (в том числе нормативно-технической документации). К ТС этой группы относят устройства автоматизированного доступа к микрофильмированным документам и устройства тиражирования микрофильмированных документов. Функции архива сформированных рекомендаций по контролю, восстановлению и размножению (тиражированию) данных архива, хранимых на машинных носителях, целесообразно производить с использованием ТС подготовки, ввода, программной обработки и вывода данных.

В состав КТС САК включают в качестве вспомогательного оборудования *средства организационной техники и оформления документации* (шкафы, стеллажи, приспособления для облегчения и рационализации труда консультантов, копировально-множительное, переплетно-брошюровочное оборудование и т. д.).

По структурному признаку рассматривают комплексы средств САК. Под комплексом средств понимают совокупность компонентов и (или) комплексов средств, предназначенную для тиражирования и ориентированную на консультирование проблем определенного класса (вида, типа) и (или) выполнение унифицированных процедур, используемую в соответствующих консультирующих и (или) обслуживающих подсистемах САК.

Различают комплексы средств одного вида обеспечения (информационного, технического, программного) и комбинированные (содержат компоненты различных видов обеспечения). Наибольшее распространение получили комбинированные комплексы средств, которые подразделяют на программно-методические и программно-технические комплексы.

Программно-методические комплексы (ПМК) включают компоненты методического (в том числе, при необходимости, математического и лингвистического), информационного и программного обеспечения и предназначены для реализации сформированных рекомендаций, управляющих и вспомогательных процедур САК на базе определенной совокупности ТС в составе комплексов средств САК.

Программно-технические комплексы (ПТК) представляют взаимосвязанную совокупность ПМК с комплексами и (или) компонентами технического обеспечения.

Классификацию ПТК определяет двухуровневая структура КТС САК (рис. 5.85), которая содержит компоненты *центрального*

вычислительного комплекса (ЦВК) и терминального комплекса, включающего *автоматизированные рабочие места* (АРМ) и отдельные терминалы пользователей САК.

Рис. 5.85. Общая структура КТС САК:
РМ — рабочее место; Т — терминал

В соответствии с этим ПТК подразделяют на ЦВК и АРМ.

ЦВК представляет собой ПТК, предназначенный для объединения действий совокупности АРМ в единый процесс консультирования, хранения и представления общесистемной информации, а также для дополнения вычислительных мощностей отдельных АРМ.

По признаку распределенности ЦВК могут строиться на базе локальных вычислительных комплексов и сетей. С точки зрения структурной организации ЦВК могут включать ЭВМ (сверхбольшие, большие и средние), многопроцессорные и многомашинные вычислительные комплексы.

АРМ представляют собой ПТК, предназначенные для выполнения следующих функций: оперативного ввода, вывода, отображения, редактирования и преобразования текстовой и (или) графической информации; настройки, редактирования, исполнения и контроля программ пользователей в диалоговом режиме; формирования архива сформированных рекомендаций и библиотеки стандартных элементов и процедур (меню); осуществления взаимодействия с другими АРМ и, при необходимости, с ЦВК; дополнения сформированных рекомендаций.

По принципу взаимодействия между собой различают независимые АРМ и локальные вычислительные сети АРМ.

В зависимости от вида и производительности процессоров различают АРМ высокой, средней и малой производительности. По виду и

назначению входящих в них ПМК АРМ подразделяют на проблемно- и объектно-ориентированные.

В состав ЦВК входят ТС группы программной обработки данных и, как вспомогательные, средства передачи данных, отображения и архива. В состав АРМ могут входить ТС различных функциональных групп.

5.10.3. Центральные вычислительные конмплексы САК

Центральные вычислительные комплексы в зависимости от территориальной распределенности САК и сложности консультируемых проблем могут создаваться на базе: глобальных вычислительных сетей (ГВС); локальных вычислительных сетей (ЛВС); многомашинных и многопроцессорных комплексов (ММК и МПК); больших и средних ЭВМ.

Сетевая организация ЦВК предусматривается для САК с высокой сложностью консультируемых проблем, если организации-пользователи расположены на значительном расстоянии друг от друга (до десятков тысяч километров) и их количество превышает 3.

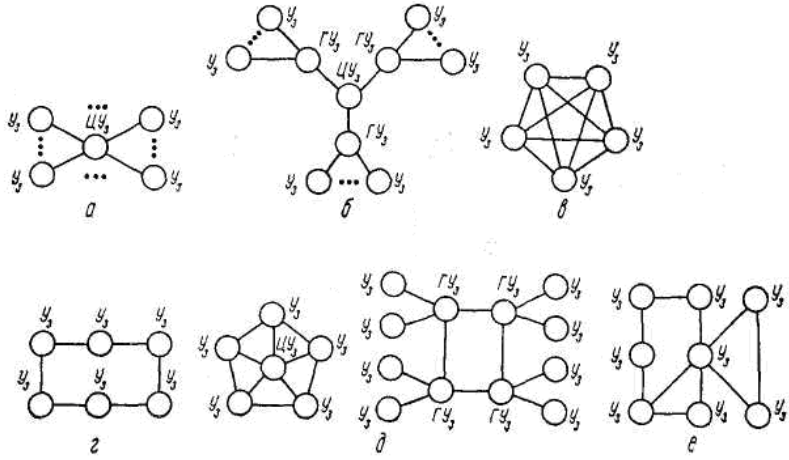


Рис. 5.86. Структура сетей:
а — радиальная; *б* — радиально-узловая; *в* — «каждый с каждым»;
г — петлевая; *д* — радиально-петлевая; *е* — распределенная;
Уз — узел; *ГУз* — групповой узел; *ЦУз* — центральный узел

Глобальные вычислительные сети могут иметь следующую структуру: радиальную, радиально-узловую, по принципу соединений «каждый с каждым», петлевую, радиально-петлевую, распределенную.

Радиальная структура (рис. 5.86, *а)* предусматривает соединение любых узлов сети между собой через центральный узел. Узел сети представляет собой совокупность оконечного оборудования каналов связи, которое включает различные средства концентрации и коммутации, а также линейное оборудование канала. К узлу можно подключить ЭВМ, ММК, АРМ или абонентский терминал (реализующий возможность приема-передачи) пользователя. Радиальная структура очень экономична с точки зрения использования каналов (относительно малая длина абонентских линий) и может быть достаточно просто реализована на телефонной и телеграфной сети общего пользования. Применяется при небольшом количестве узлов и на небольшой территории. Недостатком этой структуры является нарушение связи при выходе из строя центрального узла.

Радиально-узловая структура (рис. 5.86, *б)* реализует два уровня радиального построения: первый уровень (центральный узел — групповые узлы) и второй уровень (групповой узел — узлы). Такая структура применяется на большой территории с большим количеством узлов. Реализуется также достаточно просто, но имеет низкую надежность, так как для соединения любых двух узлов существует только один путь, как правило, состоящий из нескольких каналов и узлов. Применяют его при низкой интенсивности обмена данными между узлами.

Структура по принципу «каждый с каждым» (рис. 5.86, *в*) обладает наивысшей надежностью. Большое число каналов связи между узлами обусловливает высокую стоимость сетей такой структуры. Увеличение числа узлов сети вызывает значительное увеличение числа каналов связи. Поэтому принцип «каждый с каждым» целесообразен только для небольших сетей.

Петлевая структура (рис. 5.86, *г*) предусматривает последовательное соединение узлов в замкнутую петлю (кольцо). В такой сети повреждение одного канала, узла, ЭВМ не приводит к потере работоспособности всей сети. Сеть с петлевой структурой обладает достаточно высокой надежностью и достаточно просто может быть расширена.

Радиально-петлевая структура (рис. 5.86, *д)* представляет собой комбинацию радиальной и петлевой структур. Сети такой структуры обладают высокой надежностью и гибкостью.

В сетях с ***распределенной структурой*** (рис. 5.86, *е)* каждый узел соединен не менее, чем с двумя другими узлами. Такая структура является надежной и мобильной, так как обеспечивает многовариантность маршрутов передачи данных.

Наибольшее распространение получают петлевые, радиально-петлевые и особенно распределенные структуры сетей.

Локальные вычислительные сети. ЦБК на базе ЛВС создают для САК с небольшой территориальной распределенностью (как правило, в одной организации) и с количеством ЭВМ или вычислительных комплексов не менее трех. ЛВС может иметь структуру прямого соединения (шина) и петлевую структуру (кольцо) (рис. 5.87).

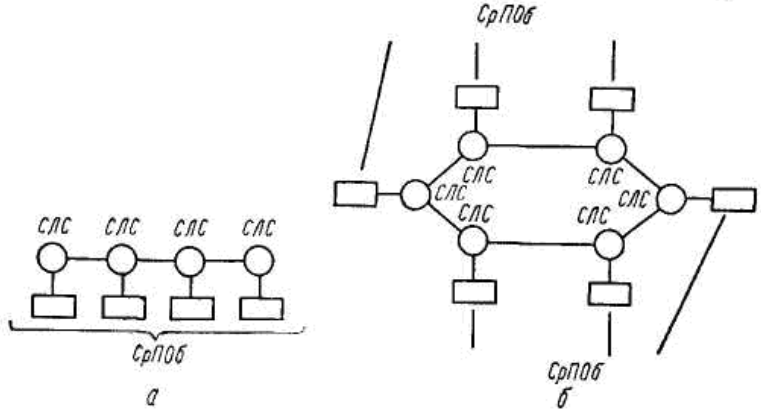


Рис. 5.87. Структура ЛВС:
а — прямое соединение (шина); *б* — петлевая структура (кольцо); СрПОб— средства программной обработки; СЛС — станция локальной сети

Как правило, в ЛВС кроме ЭВМ ЦБК включают АРМ и абонентские терминалы пользователей. Подключение средств программной обработки и терминалов в сеть осуществляют с помощью станций локальной сети, выполняющих функции приема, передачи, пакетирования и распакетирования данных, сопряжения, контроля, концентрации и т. п.

6. Математические модели консультируемых проблем

6.1. Иерархическая система математических моделей

После выбора рациональных вариантов консультирования заданной проблемы и установления соответствующих значений основных параметров консультируемой проблемы возникает задача детализации консультируемой проблемы, вылеления структурных элементов и связей между ними. Данный этап будем называть ***внутренним консультированием***.

Процесс детализации консультируемой проблемы соответствует переходу от одного уровня внутреннего консультирования к другому, более «низкому». На каждом из этих уровней применяются методы математического моделирования, выступающие как *средство определения* ***характеристик, свойств и состояния*** *консультируемой проблемы (ее фрагмента)* путем решения с помощью ЭВМ некоторых уравнений, называемых ***математическими моделями***.

Математическая модель (ММ) *консультируемой проблемы есть совокупность математических объектов (чисел, переменных, матриц, множеств и т. п.) и отношений между ними, которая адекватно отображает свойства консультируемой проблемы, интересующие консультанта данной проблемыу***.**

Выполнение консультационных операций и процедур в САК основано на оперировании с ММ. С их помощью прогнозируются характеристики и оцениваются возможности предложенных вариантов схем формируемых рекомендаций, проверяется их соответствие предъявляемым требованиям КЗ, проводится оптимизация параметров консультируемой проблемы, разрабатывается консультационная документация и т. п.

Используемые математические модели определяются предметной областью консультирования. Предметная область, в свою очередь, соответствует возможно полному физическому и математическому описанию законов и условий функционирования консультируемой проблемы, среды функционирования и способов взаимодействия консультируемой проблемы со средой, состава консультируемой проблемы, элементной базы, способов организации структуры консультируемой проблемы, изменяемых и настраиваемых параметров и т. п. Предметная область может пополняться и уточняться в процессе консультирования.

На каждом уровне внутреннего консультирования превалируют различные виды ***консультационных работ,*** которым соответствуют свои математические модели, свои зависимости между входом и выходом всей консультируемой проблемы, ее фрагментов или элементов с учетом граничных условий. На этих уровнях данные об одних и тех же свойствах консультируемой проблемы многократно используются для решения различных консультационных задач, при этом существующий уровень знаний и требуемая полнота представления данных об отдельных свойствах могут быть весьма различными. Другими словами, ***на различных уровнях и этапах консультирования требуются математические модели различной степени детализации.*** В связи с этим целесообразно создавать в САК ***единую иерархическую систему математических моделей консультируемых проблем***, при этом на различных уровнях иерархии модель, соответствующая консультационному уровню, реализуется объединением моделей уровня иерархии, на единицу меньше рассматриваемого.

В САК для каждого иерархического уровня формулируются основные положения математического моделирования, выбирается и развивается соответствующий математический аппарат, разрабаьываются типовые ММ элементов консультируемых проблем, формализуются методы получения и анализа математических моделей проблем. Сложность задач консультирования и противоречивость требований высокой точности, полноты и малой трудоемкости анализа обусловливают целесообразность компромиссного удовлетворения этих требований с помощью соответствующего выбора моделей. Это обстоятельство приводит к расширению множества используемых моделей и развитию алгоритмов адаптивного моделирования.

Функциональные и структурные модели. В консультационных процедурах, связанных с функциональным аспектом консультирования, как правило, используются ММ, отражающие закономерности процессов функционирования консультируемых проблем. Такие модели будем называть ***функциональными.*** Типичная функциональная модель представляет собой ***систему уравнений,*** описывающих либо проблемы в области электрических, тепловых, механических, экономических, социальных и др. процессов, либо проблемы в области процессов преобразования информации.

В то же время в консультационных процедурах, связанных со ***структурным*** аспектом консультирования, преобладает использование математических моделей, отражающих только структурные свойства консультируемой проблемы, например ее геометрическую

форму, размеры, взаимное расположение элементов в пространстве и др. Такие модели будем называть ***структурными***. Структурные модели чаще всего представляются в виде ***графов, матриц инциденций и смежности, списков и т. п.***

Как правило, функциональные модели более сложные, поскольку в них отражаются также сведения о структуре консультируемых проблем. Однако при решении многих задач консультирования использование сложных функциональных моделей неоправдано, так как нужные результаты могут быть получены на основе более простых структурных моделей. Функциональные модели применяют преимущественно на завершающих этапах верификации описаний формируемых рекомендаций по решению залач консультируемых проблем, предварительно синтезированных с использованием структурных моделей. Еще большая степень обобщения наблюдается в единой системе математического моделирования ИСТРА (Иерархическая Система Трансляции), позволяющей осуществить построение системы иерархических моделей разнородных консультируемых проблем. В этой системе любая консультируемая проблема — технологический процесс, или процесс проектирования, или социальная проблема и др. — ***моделируется одинаковыми математическими методами и средствами***.

Иерархия математических моделей в САК. Блочно-иерархический подход к консультированию различного рода проблем включает в качестве своей основы иерархию математических моделей. Деление моделей по иерархическим уровням (уровням абстрагирования) происходит по степени детализации описываемых свойств и процессов, протекающих в консультируемой проблеме. При этом на каждом иерархическом уровне используют свои понятия «проблема» и «элементы проблемы». Так, проблема *k*-го уровня рассматривается как элемент проблемы на соседнем более высоком *(k*—1)-м уровне абстрагирования.

Представим структуру некоторой консультируемой проблемы в виде множества элементов проблемы (рис. 6.1) и связей между ними.

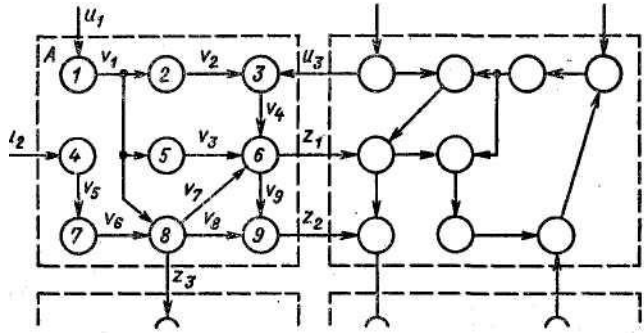


Рис. 6.1. Представление структуры консультируемой проблемы

Выделим в соответствии с блочно-иерархическим подходом в структуре консультируемой проблемы некоторые подмножества элементов и назовем их блоками (на рисунке показаны штриховыми линиями). Пусть состояние каждой связи характеризуется одной фазовой переменной v_i, z_j или u_k. Здесь v_i относится к внутренним связям между элементами данного блока, z_j, u_k относятся к выходам и входам блока соответственно. Теперь поясним важные для функциональных моделей понятия полной модели и макромодели.

Полная модель блока есть модель, составленная из моделей элементов с учетом межэлементных связей, т. е. модель, описывающая как состояние выходов, так и состояние каждого из элементов блока. Моделями элементов блока A являются уравнения, связывающие входные и выходные переменные:

$$\left.\begin{array}{l} f_1(v_1, u_1) = 0; \\ f_2(v_1, v_2) = 0; \\ f_3(v_2, u_3, v_4) = 0; \\ \cdots\cdots\cdots\cdots \\ f_9(v_9, v_8, z_2) = 0. \end{array}\right\}. \qquad (6.1)$$

Полная модель блока есть система уравнений

$$\mathbf{F(V, U)}=0, Z = \mathbf{\Psi(V,U)}, \qquad (6.2)$$

где **V, Z** и **U** — векторы внутренних, выходных и входных фазовых переменных блока. При большом количестве элементов размерность вектора **V** и порядок системы уравнений (6.2) становятся чрезмерно большими и требуются упрощения.

При переходе к более высокому иерархическому уровню упрощения основаны на исключении из модели вектора внутренних

переменных **V**. Полученная модель представляет собой систему уравнений

$$\varphi(\mathbf{Z}, \mathbf{U})=0 \qquad (6.3)$$

существенно меньшей размерности, чем полная модель (6.2), и называется ***макромоделью.*** Следовательно, макромодель уже не описывает процессы внутри блока, а характеризует только процессы взаимодействия данного блока с другими в составе системы блоков.

Модели (6.2) и (6.3) относятся друг к другу как полная модель и макромодель на *n*-м уровне иерархии. На более высоком *(n*—1)-м уровне блок *A* рассматривается как элемент и макромодель (6.3) становится моделью элемента *A*. Следовательно, модели (6.1) и (6.3) относятся друг к другу как модели элементов соседних иерархических уровней. Из моделей типа (6.3) может быть составлена полная модель проблемы на *(n*—1)-м уровне.

Во всех математических моделях различают данные *трех* типов: ***данные об элементах*** *самой консультируемой проблемы моделирования,* ***данные о свойствах*** *и* ***данные об отношениях*** *между элементами и свойствами консультируемой проблемы*. Абстрагирование при моделировании консультируемой проблемы идет по *двум* направлениям: *по глубине структуирования и по степени абстрагирования элементов и свойств консультируемой проблемы, а также отношений между ними*.

По ***глубине*** структуирования сложная консультируемая проблема *A* может рассматриваться как неструктурированная консультируемая проблема, представляющая собой единое целое, или как система взаимосвязанных элементов одного уровня, или как многоуровневая иерархическая система.

По ***степени абстрагирования*** консультируемая проблема моделируется на уровнях *структурных* (методами теории множеств и теории графов), *логических* (методами математической логики) и *количественных свойств и отношений* (методами функционального анализа, теории дифференциальных уравнений, математической статистики с непрерывным или дискретным изменением аргументов). На каждом из этих основных уровней возможны описания консультируемой проблемы с различной степенью полноты и обобщения, соответствующие ***разным уровням абстрагирования структурных, логических и количественных свойств и отношений***. Понятно, что задача «конструирования» нужной приближенной модели, которая достаточно точно отражает характерные свойства консультируемой проблемы или ее элемента на данном уровне

консультирования и в то же время является доступной для анализа и исследования, представляет большие трудности.

На структурном уровне моделируется состав элементов консультируемой проблемы на низшем уровне структурирования в виде некоторого множества $V = (v_1, v_2, ..., v_n)$, свойства и параметры которого представлены «описаниями» $E(v_i)$ наряду с «описанием» консультируемой проблемы $E(V)=(E_1, E_2, \ldots, E_m)$, а также структурными отношениями между элементами и описаниями. К структурным относятся бинарные отношения иерархической подчиненности, отношения порядка, смежности, сопряженности, функциональной связи и т. д. Например, на структурном уровне моделируются ранние этапы формирования рекомендаций по решению задач консультируемой проблемы, когда топологической моделью консультируемой проблемы служит ориентированный граф (орграф) $G(V, E)$, составление которого базируется на содержательном описании состава (множество вершин V) и способа действия консультируемой проблемы (множество ребер E). Вершинами орграфа v_i (элементами консультируемой проблемы) являются, как правило, ***законченные функционально блоки*** (части) консультируемой проблемы, а ребрами e_i — ***информационные связи*** между ними.

При описании консультируемой проблемы и ее частей используются ***вероятностные характеристики функционирования***, а основой применяемого математического аппарата при моделировании служит ***теория динамических систем, теория массового обслуживания и др.***

Консультирование на структурном уровне представляет собой процесс направленного перебора формализованных описаний структуры консультируемой проблемы и алгоритмов ее функционирования методами моделирования на ЭВМ. Рассмотренная модель полностью раскрывает внутреннюю структуру консультируемой проблемы, анализ которой позволяет решить следующие задачи, возникающие на раннем этапе консультирования: ***установить «слабые места» структуры, определить степень нагруженности и значение отдельных элементов в процессе функционирования, а также оценить последствия возможных отказов элементов консультируемой проблемы***.

Структурные отношения между элементами множества V будем описывть матрицей смежности

$$[c_{ij}]_V = [n \times n],$$

строки и столбцы которой соответствуют вершинам орграфа

структурной модели, а ее c_{ij}-й элемент равен числу ребер, направленных от вершины v_i к вершине v_i.

Отношения между элементами множеств V и E, т. е. между вершинами и ребрами орграфа, будем описывть в виде булевых матриц инцидентности

$$[a_{ij}]_{V,E} = [n \times m],$$

строки которой соответствуют вершинам, а столбцы — ребрам орграфа, при этом ее a_{ij}-й элемент равен +1, если v_i - начальная вершина ребра e_i, и равен —1, если v_i — конечная вершина ребра e_i.

С помощью подобных матриц можно выразить и другие бинарные отношения, например между «описаниями» элементов и «описанием» консультируемой проблемы в целом и др.

Пример 1. Для орграфа, показанного на рис. 6.2,

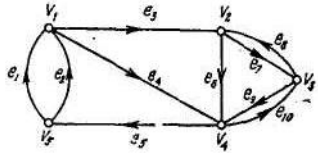


Рис. 6.2**.** Ориентированный граф

матрицы смежности C и инцидентности A имеют вид:

$$C = \begin{array}{c|ccccc|c} & v_1 & v_2 & v_3 & v_4 & v_5 & \\ \hline & & 1 & & 1 & & v_1 \\ & & & 1 & 1 & & v_2 \\ & & 1 & & 1 & & v_3, \\ & & & 1 & & 1 & v_4 \\ & 2 & & & & & v_5 \end{array}$$

$$A = \begin{array}{c|cccccccccc|c} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 & e_{10} & \\ \hline & -1 & -1 & 1 & 1 & & & & & & & v_1 \\ & & & -1 & & & 1 & 1 & -1 & & & v_2 \\ & & & & & & & -1 & 1 & 1 & -1 & v_3. \\ & & & & -1 & 1 & -1 & & & -1 & 1 & v_4 \\ & 1 & 1 & & & -1 & & & & & & v_5 \end{array}$$

В целом на структурном этапе моделируются связи, обусловленные отношениями принадлежности консультируемых проблем, их элементов и свойств к определенным множествам, отношениями

иерархической подчиненности, а также отношениями инцидентности, смежности и порядка.

На логическом уровне моделирования каждому множеству, булевой матрице бинарных отношений или структурному графу соответствуют наборы логических отношений между входящими в них элементами, представленными в виде логических переменных.

Множествам *V* и *E(V)* также соответствуют определенные логические отношения, отражающие причинно-следственные связи. Последние описывают последовательности изменения состояний консультируемой проблемы с учетом состояния других, необязательно смежных с ней, проблем. Например, на логическом уровне моделью консультируемой проблемы, которой является цифровые устройства, служат двудольные ориентированные графы (биорграфы) с двумя непересекающимися подмножествами вершин: подмножеством V_1, образованным элементами (на более глубоком, чем раньше, уровне структурирования) и их межсоединениями, и подмножеством V_2 соответствующим сигналам (логическим переменным). Ребра такого графа хорошо отражают причинно-следственные связи между указанными подмножествами вершин, соответствующих ***статическому*** (подмножество V_1) и ***динамическому*** (подмножество V_2) описанию консультируемой проблемы. Различие между входами и выходами элементов устанавливается с помощью направлений, приписываемых ребрам: выходной сигнал логического элемента исходит из соответствующей вершины, а входной сигнал имеет направление к вершине (рис. 6.3).

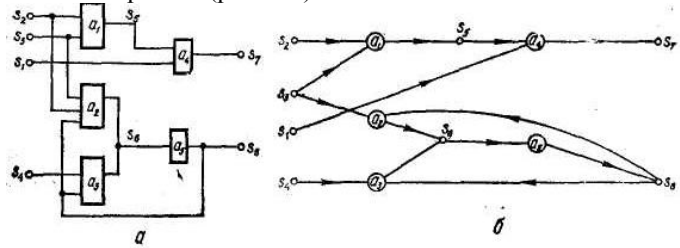


Рис. 6.3. Логическая схема (*а*) и ее модель в виде двудольного ориентированного графа (*б*)

Каждый биорграф можно описать матрицей *B,* определяющей отношение инцидентности элементов и сигналов,

$$B = [A \times S],$$

число строк которой равно числу элементов и число столбцов — числу сигналов, а b_{ij} элемент которой равен +1, если сигнал s_i есть входной сигнал элемента a_i, и равен —1, если s_j выходной сигнал элемента a_i.

Пример 3. Для двудольного ориентированного графа, изображенного на рис. 6.2, матрица *B* принимает вид:

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	
	1	1		−1				a_1
	1	1			−1		1	a_2
			1		−1		1	a_3.
				1		−1		a_4
1					−1		−1	a_5

На **уровне количественных свойств и отношений** моделирования каждому элементу множества булевой матрицы или логической переменной ставится в соответствие алгебраическая и другая количественная переменная, а логические отношения переходят в количественные отношения: уравнения, неравенства и т. п.

Если на предыдущих уровнях моделирования учитывались структурные и причинно-следственные связи в консультируемой проблеме, характеризующие взаимосвязь элементов консультируемой проблемы при абстрагировании естественных действующих факторов, то моделирование на количественном уровне характеризует *функциональные, вещественные, энергетические и пространственные связи:*

- ***функциональные связи*** определяют взаимосвязь между элементами и свойствами консультируемой проблемы;

- ***вещественные связи*** обусловлены физическими и химическими свойствами и отношениями между ними в процессе функционирования;

- ***энергетические связи*** характеризуют энергетическую сторону функционирования консультируемой проблемы.

Все эти связи обычно определяются пространственно-временными соотношениями и выражаются через ***обыкновенные дифференциальные и дифференциальные уравнения в частных производных.***

Важным методом упрощения модели является представление консультируемой проблемы в виде системы таких элементов на глубоких уровнях структурирования, связь между которыми можно с достаточной точностью охарактеризовать функциями только одной переменной (времени). Тем самым *осуществляется переход от модели с* ***распределенными параметрами*** *к более простой модели с* ***сосредоточенными параметрами***.

Пример 4. Построим модель консультируемой проблемы представленной пневмогидравлической системы с газовым редуктором, изображенной на рис. 6.4, *а.*

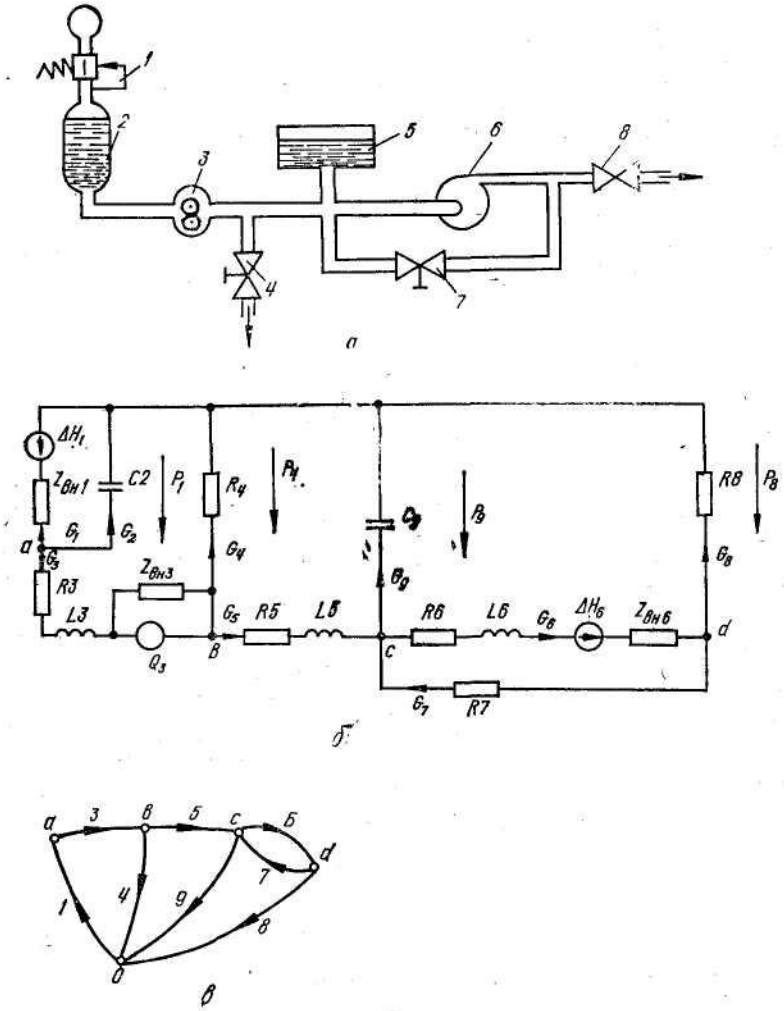


Рис. 6.4. Пневмогидравлическая система (*а*), ее эквивалентная схема (*б*) и ее граф (*в*)

Из сосуда *2* с редуктором *1* жидкость по тракту поступает в объемный насос *3.* Из него частично через заслонку (клапан) *4* жидкость отбирается, частично —она поступает в лопастный насос *6,* откуда часть жидкости через регулирующий дроссель 7 возвращается на вход *5* в насос, а часть отбирается во внешнюю цепь через дроссель *8.* Рассматривая отдельные устройства и тракты системы как линеаризованные элементы с сосредоточенными параметрами, учитываем инерцию и гидравлическое сопротивление для каждого из отдельных участков трактов. Эквивалентная схема системы показана на рис. 6.4, *б,* а модель ее в виде орграфа — на рис. 6.4, *в.* Ветви эквивалентной схемы, соответствующие реальным источникам давления (редуктор) и расхода (насосы), содержат идеальные источники напора с внутренними сопротивлениями $Z_{ВН1}$ — $Z_{ВН6}$. На эквивалентной схеме обозначены также переменные, характеризующие ее ветви: G_1 — G_8 — расход; P_1 — P_8 — давление.

Нетрудно видеть, что при количественном моделировании определяющим является ***состав и полнота описания*** выделенных элементов консультируемой проблемы, которым в орграфе соответствуют отдельные ребра.

Количественные соотношения*, определяющее связь между переменными элементов (компонентов) консультируемой проблемы, будем называть* ***компонентными уравнениями*****.**

Вид и способ получения компонентных уравнений зависит от глубины структурирования консультируемой проблемы. При этом целесообразно получаемые ***модели элементов*** разделить на два класса: ***макромодели и микромодели*****.**

Если при создании модели не рассматриваются внутренняя структура моделируемых элементов консультируемой проблемы и закономерности, определяющие их функционирование, а целью моделирования является обеспечение возможности ***адекватно предсказать реакцию компонента по реакции его модели***, такие модели будем называть ***макромоделями***, или ***аппроксимирующими моделями***. Они, по-существу, ***адекватно аппроксимируют поведение компонента и консультируемой проблемы*** в целом, реализуя, быть может, другие законы функционирования.

Моделирование на ***количественном уровне*** с использованием макромоделей будем называть ***адекватным моделированием*****.**

Модели компонентов, в которых главным является ***отражение внутренних взаимосвязей моделируемой консультируемой проблемы с точностью до минимальных элементов структуры***, будем называть ***микромоделями*****.**

При этом под ***минимальными элементами*** *будем понимать элементы, не имеющие* ***структурного описания*** *и находящиеся на* ***самом высшем уровне структурирования*** *консультируемой проблемы*.

Моделирование на качественном уровне с использованием микромоделей будем называть детальным моделированием.

Пример 5. На рис. 6.5, *а* показана схема логического элемента *И,* определяющего часто глубину структурирования цифрового устройства на этапе логического моделирования.

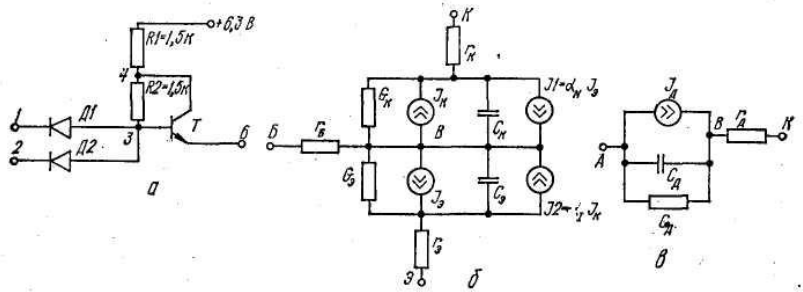


Рис. 6.5. Схема логического элемента (*а*), схема замещения транзистора *(б)* и диода *(в)*

Для построения ее микромодели необходимо заменить используемые транзисторы и диоды соответствующими схемами замещения (рис. 6.5, *б* и *в*) на уровне минимальных элементов, в качестве которых при ***детальном (схемотехническом) моделировании*** выступают компоненты *R, L, C, E, I* и четыре типа зависимых источников. Одна из возможных макромоделей элемента *И* для адекватного моделирования, также выраженная совокупностью минимальных элементов, показана на рис. 6.6, *а*.

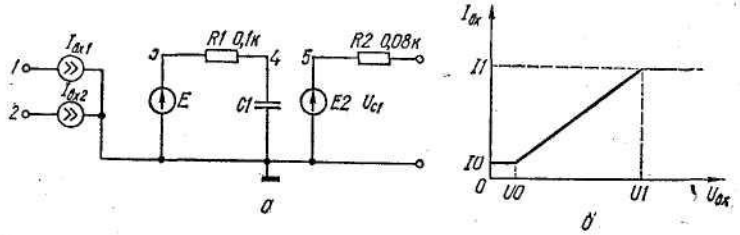


Рис. 6.6. Схема макромодели логического элемента (*а*) и его кусочно-линейная входная характеристика *(б)*

Вошедшие в нее элементы описываются следующим образом. Источник, моделирующий входную цепь логического элемента (рис. 6.6, *б*):

$$I_{вх} = \varphi(U_{вх}) = \begin{cases} I0, \ U_{вх} \leqslant U0; \\ I0 + (I1 - I0)\ (U_{вх} - U0) \mid (U1 - U0), \ U0 < U_{вх} < U1; \\ I1, \ U_{вх} \geqslant U1, \end{cases}$$

где *I0* и *U0* — ток и напряжение логического состояния «0»; *I*1 и *U*1 — ток и напряжение логического состояния «1».

Источник *E,* моделирующий логическую функцию элемента, в общем случае описывается выражением

$$E = \psi(U0, U1, \ U_{BX_1} \ldots . U_{BX_N}),$$

которое содержит пороговую булевую функцию *TF* (равную 1, если $U_{BX} > U_T$, и нулю, если $U_{BX} < U_T$, где U_T — пороговое напряжение) и функцию *LF*, зависящую от логического закона функционирования элемента. В описываемой макромодели функция *LF* определяется по разному для элементов типа *И, ИЛИ, И—НЕ, ИЛИ—НЕ, И—ИЛИ, ИЛИ—И, И— ИЛИ—НЕ* и др.

Нелинейная емкость, моделирующая фронты и задержки переключения, определяется соотношением:

$$C = f(U_C) = \begin{cases} C_0 + C_1\ (U_C - U0)/(U1 - U_C), \ U < 0; \\ C_0 = \alpha_{11} FZ + \alpha_{12} DZ, \ C_1 = \alpha_{21} FZ + \alpha_{22} DZ; \\ C_0^1 + C_1^1\ (U1 - U_C)/(U1 - U0), \ U > 0; \\ C_0^1 = \alpha_{11} FP + \alpha_{12} DP; \ C_1^1 = \alpha_{21} FP + \alpha_{22} DP, \end{cases}$$

где *FZ* и *FP* — фронты переключения в логические состояния «1» и «0» соответственно; *DZ* и *DP*—задержки переключения в логические состояния «1» и «0».

В табл. 6.1 с учетом рис. 6.7 сравниваются результаты моделирования логического элемента *И* с использованием макромодели и микромодели, полученные с помощью программы СПАРС.

Как видно из табл. 6.1, точность адекватного моделирования достаточно высокая.

Таблица 6.1

Сравнение результатов моделировавания логического элемента с помощью микро- и макромоделей

Выделенные временные точки и интервалы	Микромодель, нс	Макромодель, нс
t_{p1}	1,0000	1,0204
t_{p5}	1,1095	1,1098
t_{p9}	1,3776	1,3532
$t_{\phi p} = t_{p9} - t_{p1}$	0,3776	0,3328
$t_{dp} = t_{p5} - t_{sp}$	0,1095	0,1098
t_{z9}	3,000	3,0123
t_{z5}	3,0940	3,0935
t_{z1}	3,3435	3,3618
$t_{\phi z} = t_{z1} - t_{z9}$	0,3435	0,3495
$t_{dz} = t_{zs} - t_{z5}$	0,0940	0,0935

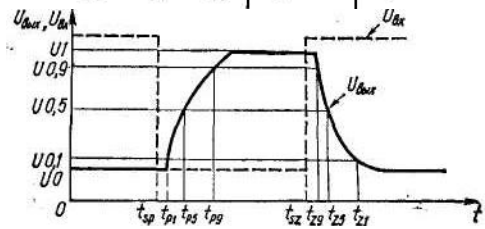


Рис. 6.7. Эпюры (диаграммы) входного и выходного сигналов

При этом использовались следующие данные: для макромодели (*I0* = — 1,6 мА; *I*1 = 0,04 мА; *U0* = 0,643 В; *U*1=3,285 В; U_T = 1,976 В; *FZ* = 3,435 нс; *FP* = 3,776 нс; *DZ* = 0,94 нс; *DP* = 1,095 нс; α_{11} = — 14,703333; α_{12} = 9,189584; α_{21} = 40,383148, α_{22} = —12,739465) и микромодели

$$(J_э = 0{,}1 \cdot 10^{-10}(e^{38{,}46U_{BЭ}} - 1)_{мА};$$

$$J_к = 0{,}2 \cdot 10^{-8}(e^{33{,}33U_{BK}} - 1)_{мА};$$

$$J_1 = 0{,}99\, J_э;\ J_2 = 0{,}89 J_к;$$

$$C_э = 19{,}28 \cdot 10^{-8}\,(e^{38{,}46U_{BЭ}} - 1) + 50\ пФ;$$

$$C_к = 33{,}33 \cdot 10^{-7}\,(e^{33{,}33U_{BK}} - 1) + 50\ пФ;$$

$$r_б = 50\ Ом;\ r_к = 23\ Ом;\ r_э = 0;$$

$$G_э = 1\ мкСм;\ G_к = 0{,}1\ мкСм;\ G_д = 1\ мкСм;$$

$J_д = 0{,}1 \cdot 10^{-10}\,(e^{31{,}25U_{AB}} - 1)$ мФ; $C_д = 31{,}25 \cdot 10^{-9}\,(e^{31{,}25U_{AB}} - 1) + 50$ пФ; $r_д$ = 50 Ом).

Совокупность моделей консультируемой проблемы на ***структурном, логическом и количественном уровнях моделирования*** представляет собой ***иерархическую систему моделей***, раскрывающую взаимосвязь различных сторон описания консультируемой проблемы и обеспечивающую системную связность ее элементов и свойств на всех стадиях процесса консультирования. При переходе на более высокий уровень абстрагирования осуществляется ***свертка данных*** о моделируемой консультируемой проблеме, при переходе к более детальному уровню описания — ***развертка этих данных.***

Для примера рассмотрим границы возможного применения различных видов моделирования в процессе автоматизированного консультирования проблемы, связанной с проектированием электронных систем на дискретных микросхемах (кривая *1)* и больших интегральных схем (БИС*)* кривая *(2)* (рис. 6.8). Здесь по оси ординат указана сложность проектируемых объектов, выраженная в числе логических элементов (инверторов) *n*, а по оси абсцисс отложены годы, начиная с 1960.

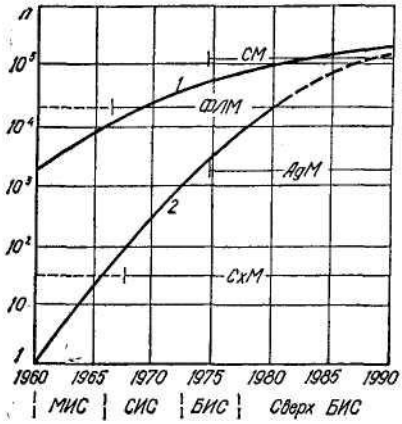


Рис. 6.8. К пояснению границ возможного применения различных видов моделирования в САПР электронных устройств и БИС

Здесь же отмечены периоды широкого использования интегральных схем малой (МИС), средней (СИС), большой (БИС) и сверхбольшой степени интеграции. Горизонтальные линии для определенной функциональной сложности проектируемых объектов соответствуют структурному моделированию *(СМ),* функционально-логическому моделированию *(ФЛМ),* адекватному моделированию

(АдМ) и традиционному схемотехническому *(СхМ),* или детальному моделированию.

Сравнительные характеристики моделей для различных видов моделирования приведены в табл. 6.2, из которой наглядно следует тенденция свертки данных и уменьшения степени связи с порождающей средой (технологией) по мере перехода на более высокие уровни абстрагирования за счет повышения быстродействия самого процесса моделирования.

Таблица 6.2

Сравнительные характеристики различных видов моделирования

Характеристика	СМ	ФЛМ	АдМ	СхМ
Уровень	Функциональный (устройство, блок)	Функциональный (инвертор, счетчик, регистр)	Функциональный (инвертор)	Приборный (транзистор)
Модель	Информационная, имитационная, вероятностно-эвристическая	Функциональная, структурно-функциональная	Аналитическая (макромодель), регрессионная, факторная	Аналитическая (микромодель)
Переменная	Система команд и сигналов управления	1, 0, *X*, задержки	$U(t)$, $i(t)$ (погрешность 20%)	$U(t)$, $i(t)$ (погрешность 1%)
Связь с технологией	(0 %)	Очень слабая (20%)	Слабая (80%)	Отличная (100%)
Алгоритм моделирования	Массового обслуживания, искусственного интеллекта, теории дискретных систем	Итерационный и событийный	Численных методов, методов экстраполяции и аппроксимации, математической статистики, планирование эксперимента	Вычислительной математики (численных методов решения различных типов уравнений)
Используемая процедура	Логическая	Логическая	Арифметическая и логическая	Арифметическая и логическая
Максимальная сложность объекта, инверторов	До 1 000 000	20 000	3000	30—50

Например, для этапов *ФЛМ, АдМ* и *СхМ* соответствующие времена связаны примерным соотношением:

$$t_{АдМ}: t_{ФЛМ} : t_{СхМ} = 10^3:10^2:1.$$

К количественному уровню моделирования относится также этап начального консультирования, учитывающий пространственные связи элементов консультируемой проблемы, на котором решаются задачи нахождения оптимального расположения элементов (их компоновки и размещения) и определения связей между ними. Одна из форм пространственных связей — размерная связь между самими элементами консультируемой проблемы и порождающей среды. ***Размерные связи*** между самими элементами консультируемой проблемы будем называть ***конструктивными***, а между элементами консультируемой проблемы и объектами порождающей среды (технологической оснастки, инструмента, оборудования) — ***технологическими***.

Этому этапу моделирования соответствуют свои модели в виде графов и гиперграфов. Например, для консультируемых проблем,

представляемых объектами, которые расположены на плоских конструкциях, т. е. конструкциях, в которых элементы и соединения располагаются в ограниченном числе параллельных плоскостей (интегральные схемы, многослойные печатные платы, различного рода консультационная документация и др.) целесообразно процесс консультирования разбить на два последовательно выполняемых этапа. На первом этапе осуществляется построение ***предварительного варианта рекомендаций*** по решению консультационной задачи размещения элементов и соединений, удовлетворяющего топологическим ограничениям (например, запрещение расположения трасс в заданных областях монтажного пространства и пересечения различных соединений). На втором этапе полученный вариант размещения преобразуется в реализуемую конструкцию, удовлетворяющую не только топологическим, но и метрическим ограничениям (например, фиксированные размеры элементов, минимальные допустимые расстояния между различными элементами конструкции и др.).

Описанный подход к начальному этапу консультирования будем называть ***топологическим*** и он состоит из следующих основных этапов:

1) построение математической модели плоского объекта;
2) формирование плоской укладки модели объекта (топологический анализ);
3) планаризация модели;
4) построение предварительного варианта размещения элементов и соединений;
5) построение окончательного варианта размещения элементов и соединений.

На первом этапе строится математическая модель объекта, отражающая рассматриваемые ею свойства, на втором — выделяется максимальная часть модели, которая может быть уложена на плоскости без нарушения топологических ограничений. Если выделенная часть содержит не все элементы модели, то выполняется третий этап, на котором исходная модель преобразуется в планарную. При этом учитываются особенности применяемой технологии. Четвертый этап предназначен для автоматической прорисовки предварительного варианта конструкции, которая окончательно отрабатывается на пятом этапе.

Будем различать ***три вида топологических моделей плоских объектов*** — ***граф, гиперграф и многоместный граф***. Представление модели консультируемой проблемы в виде графа можно строить на

основании двух различных отображений. При первом отображении элементы консультируемой проблемы соответствуют вершинам графа, при втором — прообразами вершин являются контакты. Если между некоторыми контактами запрещено прохождение соединений, то вершины, соответствующие таким контактам, соединяются структурными ребрами.

Уточненное представление модели плоского объекта в виде графа (модель *М1)* предполагает отображение элементов и их соединений в вершины графа. При этом граф превращается в двудольный граф, а отношения инцидентности в объекте переходят в отношения смежности на таком графе.

Представление модели консультируемой проблемы в виде гиперграфа предполагает однозначное соответствие элементов вершинам, а соединений — ребрам гиперграфа, однако информационные свойства модели с точки зрения анализа планарности остаются такими же, как у предыдущей модели. В более общей постановке задачи топологического анализа плоского объекта необходимо учитывать порядок расположения контактов его элементов, для чего строятся графы-модели элементов. В таких моделях вершины соответствуют контактам, а ребра — парам соседних контактов. Общая модель консультируемой проблемы (модель *М2)* с учетом графов-моделей элементов представляется многоместным графом *G(A, R),* где *А* — множество вершин, а *R*— множество комплексов (множество подграфов с некоторой центральной точкой).

Пример 6. Построение модели *М2* иллюстрируется рис. 6.9, на котором показана схема (рис. 6.9, *а*), ее потенциальный граф (рис. 6.9, *б),* который отображает элементы в вершины, а соединения — в комплексы, и полная топологическая модель схемы (рис. 6.9, *в*) с учетом графов-моделей элементов.

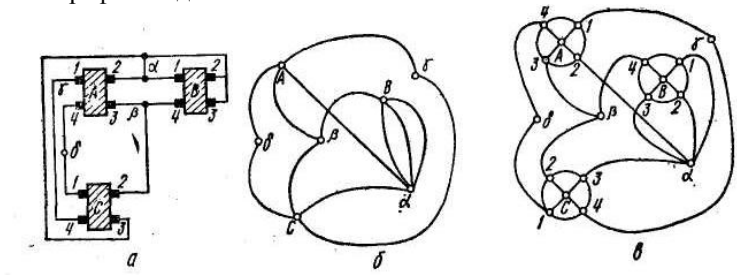


Рис. 6.9. К пояснению построения модели М2 плоского объекта на основании многоместного графа

Модели консультируемых проблем в виде графов и гиперграфов широко используются в машиностроении для описания конструкций машин и механизмов. Например, структура соединений в конструкции при автоматизированном проектировании технологии сборки представляется орграфом *S(D, C),* множеству вершин *D* которого соответствуют сборочные единицы, а ребрам *C* — виды соединений, которые определяются набором параметров. К последним относятся параметры, описывающие разновидность сопряжения и его размерные характеристики с учетом геометрических и конструктивных особенностей сопрягаемых поверхностей, а также параметры, отображающие технические условия на состояние сопрягаемых поверхностей и вид силового замыкания, т. е. вид соединительных элементов.

6.2. Микро-, макро- и метауровни математических моделей.

В зависимости от сложности консультируемой проблемы при ее консультировании используют большее или меньшее число уровней абстракции. Объединение уровней, родственных по характеру используемого математического аппарата, приводит к образованию трех укрупненных уровней — микро-, макро- и метауровня — в иерархии функциональных моделей для большинства консультируемых сложных проблем.

На ***микроуровне*** используют математические модели, описывающие ***физическое состояние и процессы в сплошных средах.*** Для моделирования применяют ***аппарат уравнений математической физики.*** Примерами таких уравнений служат ***дифференциальные уравнения в частных производных***—уравнения электродинамики, теплопроводности, упругости, газовой динамики. Эти уравнения описывают поля электрического потенциала и температуры в полупроводниковых кристаллах интегральных схем, напряженно-деформированное состояние деталей механических конструкций и т. п. К типичным ***фазовым переменным*** на микроуровне относятся ***электрические потенциалы, давления, температуры, концентрации частиц, плотности токов, механические напряжения и деформации***. ***Независимыми переменными*** являются ***время и пространственные координаты***. В качестве операторов *F* и *V* в уравнениях (6.2) фигурируют дифференциальные и интегральные операторы. Уравнения (6.2), дополненные краевыми условиями, составляют ММ консультируемой проблемы на микроуровне. Анализ

таких моделей сводится к решению краевых задач математической физики.

На ***макроуровне*** производится дискретизация пространств с выделением в качестве элементов отдельных деталей, дискретных электрорадиоэлементов, участков полупроводниковых кристаллов. При этом из числа независимых переменных исключают пространственные координаты.

Функциональные модели на макроуровне представляют собой системы алгебраических или обыкновенных дифференциальных уравнений, для их получения и решения используют соответствующие численные методы.

В качестве ***фазовых переменных фигурируют электрические напряжения, токи, силы, скорости, температуры, расходы*** и т. д. Они характеризуют проявления внешних свойств элементов при их взаимодействии между собой и внешней средой в электронных схемах или механических конструкциях.

На ***метауровне*** с помощью дальнейшего абстрагирования от характера физических процессов удается получить приемлемое по сложности описание информационных процессов, протекающих в консультируемых проблемах. На ***метауровне*** для моделирования проблем аналоговой РЭА широко применяют ***аппарат анализа систем автоматического управления, а для моделирования проблем цифровой РЭА — математическую логику, теорию конечных автоматов, теорию массового обслуживания. Математические модели на метауровне — системы обыкновенных дифференциальных уравнений, системы логических уравнений, имитационные модели систем массового обслуживания***.

Формы представления моделей. Для представления моделей используют следующие основные формы:

Инвариантная форма — запись соотношений модели с помощью традиционного математического языка безотносительно к методу решения уравнений модели.

Алгоритмическая форма — запись соотношений модели и выбранного численного метода решения в форме алгоритма.

Аналитическая форма — запись модели в виде результата аналитического решения исходных уравнении модели; обычно модели в аналитической форме представляют собой явные выражения выходных параметров как функций внутренних и внешних параметров.

Схемная форма, называемая также ***графической*** формой, — представление модели на некотором графическом языке, например на языке графов, эквивалентных схем, диаграмм и т. п. Графические

формы удобны для восприятия человеком. Использование таких форм возможно при наличии правил однозначного истолкования элементов чертежей и их перевода на язык инвариантных или алгоритмических форм.

Модели в алгоритмической и аналитической формах называют соответственно ***алгоритмическими*** **и** ***аналитическими***. Среди алгоритмических моделей важный класс составляют ***имитационные модели***, ***предназначенные для имитации физических или информационных процессов в консультируемой проблеме при задании различных зависимостей входных воздействий от времени***. Собственно имитацию названных процессов называют ***имитационным моделированием. Результат имитационного моделирования — зависимости фазовых переменных в избранных элементах системы от времени***. Примерами имитационных моделей являются модели электронных схем в виде систем обыкновенных дифференциальных уравнений или модели систем массового обслуживания, предназначенные для имитации процессов прохождения заявок через систему.

6.3. Требования к математическим моделям

Основными требованиями, предъявляемыми к математическим моделям, являются требования адекватности, универсальности и экономичности.

Адекватность. Модель считается адекватной, если отражает заданные свойства консультируемой проблемы с приемлемой точностью. Точность определяется как степень совпадения значений выходных параметров модели и консультируемой проблемы. Пусть ε_j— относительная погрешность модели по j-му выходному параметру:

$$\varepsilon_j = (\tilde{y}_j - y_j)/y_j,$$

где $\tilde{y}_j$ — j-й выходной параметр, рассчитанный с помощью модели;

y_i — тот же выходной параметр, имеющий место в моделируемой проблеме.

Погрешность модели $\varepsilon_м$ по совокупности учитываемых выходных параметров оценивается одной из норм вектора $\varepsilon=(\varepsilon_1, \varepsilon_2,..., \varepsilon_m)$, например

$$\varepsilon_м = \max_{j=\overline{1,m}} |\varepsilon_j| \text{ или } \varepsilon_м = \sqrt{\sum_{j=1}^{m} \varepsilon_j^2}.$$

Точность модели различна в разных условиях функционирования консультируемой проблемы. Эти условия характеризуются внешними

параметрами. Если задаться предельной допустимой погрешностью $\varepsilon_{пред}$, то можно в пространстве внешних параметров выделить область, в которой выполняется условие

$$\varepsilon_м < \varepsilon_{пред},$$

Эту область называют ***областью адекватности*** (ОА) модели. Возможно введение индивидуальных предельных значений $\varepsilon_{пред\ j}$ для каждого выходного параметра и определение ОА как области, в которой одновременно выполняются все m условий вида $| \varepsilon_j | \le \varepsilon_{пред\ j}$.

Пример ОА (заштрихована) в двумерном пространстве дан на рис. 6.10.

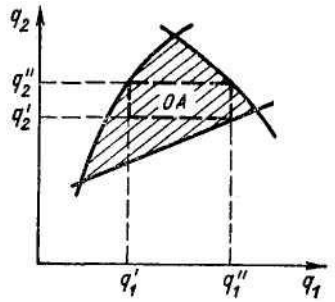


Рис. 6.10. Пример области адекватности

Здесь q_k — k-й внешний параметр.

Определение областей адекватности для конкретных моделей — сложная процедура, требующая больших вычислительных затрат. Эти затраты и трудности представления ОА быстро растут с увеличением размерности пространства внешних параметров. Определение ОА — более трудная задача, чем, например, задача параметрической оптимизации.

Для моделей унифицированных элементов расчет областей адекватности является оправданным в связи с однократностью определения ОА и многократностью их использования при консультировании различных проблем. Знание ОА позволяет правильно выбирать модели элементов из числа имеющихся и тем самым повышать достоверность результатов машинных процедур формирования рекомендаций.

В общем случае ОА может иметь произвольную форму, сведения о которой выражаются громоздко, и неудобна в использовании, поэтому на практике вместо истинных ОА применяют те или иные их аппроксимации. Наиболее просто представляются и используются

сведения об областях, имеющих формулу гиперпараллелепипеда, который задается *p* двусторонними неравенствами:

$$q_k^{'} \leqslant q_k \leqslant q_k^{''},\ k = \overline{1,\ p},$$

где *p* — размерность пространства внешних параметров.

Так, для одной из возможных макромоделей логического элемента транзисторно-транзисторной логики, реализующего функцию И—НЕ, рассчитанная область адекватности выражается следующими неравенствами:

$$4{,}47\text{В} \leqslant E \leqslant 5{,}82\text{ В};\ 3{,}89\text{ нс} \leqslant t_{\text{ф.вх}} \leqslant 11{,}2\text{ нс};$$
$$0 \leqslant N \leqslant 3;\ 3\text{ нс} \leqslant \tau_{\text{вх}} \leqslant 6\text{ нс},$$

где E — напряжение питания; $t_{\text{ф.вх}}$—длительность фронта входного сигнала; N — коэффициент нагружения; $\tau_{\text{вх}}$ — длительность входного сигнала.

В библиотеку моделей элементов наряду с алгоритмом, реализующим модель, и номинальными значениями параметров должны включаться граничные значения внешних параметров q'_k и q''_k, задающие область адекватности.

На рис. 6.10 дано графическое представление области адекватности и аппроксимирующего ее гиперпараллелепипеда. Такое представление удобно для двумерных случаев.

Возможно использование и других аппроксимаций ОА, например областей с линеаризованными границами в виде участков гиперплоскостей, областей в форме гиперсфер и т. п.

Универсальность. При определении ОА необходимо выбрать совокупность внешних параметров и совокупность выходных параметров y_i, отражающих учитываемые в модели свойства. Типичными внешними параметрами при этом являются параметры нагрузки и внешних воздействий (электрических, механических, тепловых, радиационных и т.п.). Увеличение числа учитываемых внешних факторов расширяет применимость модели, но существенно удорожает работу по определению ОА. Выбор совокупности выходных параметров также неоднозначен, однако для большинства консультируемых проблем число и перечень учитываемых свойств и соответствующих им выходных параметров сравнительно невелики, достаточно стабильны и составляют ***типовой набор выходных параметров.*** Например, дли макромоделей логических элементов БИС такими выходными параметрами являются уровни выходного напряжения в состояниях логических «0» и «1», запасы

помехоустойчивости, задержка распространения сигнала, рассеиваемая мощность.

Если адекватность характеризуется положением и размерами ОА, то универсальность модели определяется числом и составом учитываемых в модели внешних и выходных параметров.

Экономичность. Экономичность модели характеризуется затратами вычислительных ресурсов для ее реализации, а именно затратами машинного времени $T_м$ и памяти $П_м$. Общие затраты $T_м$ и $П_м$ на выполнение в САК какой-либо консультационной процедуры зависят как от особенностей выбранных моделей, так и от методов решения.

В большинстве случаев при реализации численного метода происходят многократные обращения к модели элемента, входящего в состав моделируемой проблемы. Тогда удобно экономичность модели элемента характеризовать затратами машинного времени, получающимися при обращении к модели, а число обращений к модели должно учитываться при оценке экономичности метода решения.

Экономичность модели по затратам памяти оценивается объемом оперативной памяти, необходимой для реализации модели.

Требования широких областей адекватности, высокой степени универсальности, с одной стороны, и высокой экономичности, с другой, являются противоречивыми. Наилучшее компромиссное удовлетворение этих требований оказывается неодинаковым в различных применениях. Это обстоятельство обусловливает использование в САК многих моделей для консультируемых проблем одного и того же типа — различного рода макромоделей, многоуровневых, смешанных моделей и т. п.

6.4. Методы получения моделей элементов

Получение моделей элементов (моделирование элементов) в общем случае — процедура неформализованная. Основные решения, касающиеся выбора вида математических соотношений, характера используемых переменных и параметров, принимает консультант. В то же время такие операции, как расчет численных значений параметров модели, определение областей адекватности и другие, алгоритмизированы и решаются на ЭВМ. Поэтому моделирование элементов обычно выполняется специалистами-консультантами конкретных проблемных областей с помощью традиционных средств экспериментальных исследований и средств САК.

Методы получения функциональных моделей элементов делят на теоретические и экспериментальные.

Теоретические методы основаны на изучении физических закономерностей протекающих в проблеме процессов, определении соответствующего этим закономерностям математического описания, обосновании и принятии упрощающих предположений, выполнении необходимых выкладок и приведении результата к принятой форме представления модели.

Экспериментальные методы основаны на использовании внешних проявлений свойств консультируемой проблемы, фиксируемых во время эксплуатации однотипных консультируемых проблем или при проведении целенаправленных экспериментов с консультируемыми проблемами.

Несмотря на эвристический характер многих операций моделирования ***имеется ряд положений и приемов, общих для получения моделей различных проблем***. Достаточно общий характер имеют методика макромоделирования, математические методы планирования экспериментов, а также алгоритмы формализуемых операций расчета численных значений параметров и определения областей адекватности.

Методика макромоделирования. Применение методики состоит из следующих этапов:

1. Определение тех свойств консультируемой проблемы, которые должны отражаться моделью (устанавливаются требования к степени универсальности будущей модели).

2. Сбор априорной информации о свойствах моделируемой проблемы. Примерами собираемых сведений могут служить справочные данные, математические модели и результаты эксплуатации существующих аналогичных проблемных объектов и т. п.

3. Получение общего вида уравнений модели (структуры модели). Этот этап в случае теоретических методов включает выполнение всех присущих этим методам операций, перечисленных выше. Часто составителю модели удобнее оперировать не уравнениями, а эквивалентными схемами, с помощью которых консультанту проще устанавливать физический смысл различных элементов математической модели.

4. Определение численных значений параметров модели. Возможны следующие приемы выполнения этого этапа:

а) использование специфических расчетных соотношений с учетом собранных на этапе 2 сведений;

б) решение экстремальной задачи, в которой в качестве целевой функции выбирается степень совпадения известных значений выходных параметров консультируемой проблемы с результатами использования модели, а управляемыми параметрами являются параметры модели;

в) проведение экспериментов и обработка полученных результатов.

5. Оценка точности полученной модели и определение области ее адекватности. При неудовлетворительных точностных оценках выполняют итерационное приближение к желаемому результату повторением этапов 3—5.

6. Представление полученной модели в форме, принятой в используемой библиотеке моделей.

Методы планирования экспериментов. Для целей моделирования используют пассивные и активные эксперименты.

В ***пассивных экспериментах*** нет возможности выбирать условия опыта по своему усмотрению и устанавливать значения факторов на желаемом уровне.

В ***активных экспериментах*** опыты проводятся по заранее разработанному плану, выражающему количество опытов и значения факторов в каждом опыте.

Выбор вида зависимости выходного параметра макромодели y (в общем случае рассматривается вектор выходных параметров **Y**) от внешних параметров q_k, объединенных в вектор факторов **Q**, осуществляется консультантом. Чаще всего в методах планирования эксперимента используются модели линейные

$$y = \mathbf{AQ} \qquad (6.4)$$

или квадратичные

$$y = \mathbf{AB}, \qquad (6.5)$$

где **A** — вектор-строка коэффициентов (параметров) модели;

B — вектор, включающий факторы q_k, те или иные произведения из двух, трех или более факторов и возможно также квадраты факторов q^2_k; k=1,2,..., p; p — число факторов.

Число опытов N, как правило, должно превышать число определяемых параметров вектора **A**. Параметры рассчитывают по методу наименьших квадратов, т.е. из условия минимизации суммы квадратов отклонений значений $\tilde{y}_l$, определенных по уравнению модели (6.4), и измеренных значений y_l:

$$\min_{\mathbf{A}} \sum_{l=1}^{N} \left(\tilde{y}_l(\mathbf{A}) - y_l\right)^2,$$

где *l* — номер опыта.

В зависимости от способов планирования преимущества активных экспериментов перед пассивными могут выражаться в получении оптимального положения области адекватности, в ее увеличенном объеме, в упрощении оценок точности и т. п.

Регрессионный анализ. Связь между y и **Q** может быть не функциональной, а статистической, что особенно характерно при пассивных экспериментах. Для получения моделей в такой ситуации часто применяют регрессионный анализ. Модель ищется в форме уравнения регрессии (6.4), в котором роль коэффициентов a_k в векторе **A** выполняют коэффициенты относительной регрессии.

Рассмотрим алгоритм вычисления коэффициентов a_k. По результатам пассивных экспериментов получаются оценки математических ожиданий M_y, M_k, среднеквадратичных отклонений σ_y, σ_k соответственно для выходного y и внешних q_k, параметров, а также коэффициенты корреляции r_k между y и q_k, образующие вектор **R**, и коэффициенты корреляции d_{kj} между факторами q_k и q_j, образующие матрицу **D**. Далее решается система линейных алгебраических уравнений

$$\mathbf{D\eta = R} \qquad (6.6)$$

и полученный вектор $\eta=(\eta_1, \eta_2, \ldots, \eta_p)$ используется при расчете относительных коэффициентов регрессии по формуле

$$a_k = \eta_k \sigma_y / \sigma_k.$$

Если факторы q_k некоррелированы, то **D** — единичная матрица и можно обойтись без решения системы (6.6), так $\eta_k=r_k$.

Диалоговое моделирование. Наличие в методике макромоделирования эвристических и формальных операций обусловливает целесообразность разработки моделей элементов в диалоговом режиме работы с ЭВМ. Язык взаимодействия человека с ЭВМ должен позволять оперативный ввод исходной информации о структуре модели, об известных характеристиках и параметрах консультируемой проблемы, о плане экспериментов. Диалоговое моделирование должно иметь программное обеспечение, в котором реализованы алгоритмы статистической обработки результатов экспериментов, расчета выходных параметров эталонных моделей и создаваемых макромоделей, в том числе расчета параметров по методам планирования экспериментов и регрессионного анализа, алгоритмы методов поиска экстремума, расчета областей адекватности и др. Пользователь, разрабатывающий модель, может менять уравнения модели, задавать их в аналитической, схемной или табличной форме,

обращаться к нужным подпрограммам и тем самым оценивать результаты предпринимаемых действий, приближаясь к получению модели с требуемыми свойствами.

6.5. Математические модели консультируемых проблем, используемые на микроуровне

Математические модели консультируемых проблем, связанные с элементами и процессами на микроуровне отражают физические процессы, протекающие в сплошных средах и непрерывном времени. *Независимыми переменными* в этих моделях являются *пространственные координаты и время*. В качестве *зависимых переменных* выступают *фазовые переменные*, такие как потенциалы, напряженности полей, концентрации частиц, деформации и т. п. Взаимосвязи переменных выражаются с помощью уравнений математической физики — интегральных, интегродифференциальных или дифференциальных уравнений в частных производных. Эти уравнения составляют основу ММ на микроуровне.

Для получения законченной математической модели, используемой в задачах консультирования, необходимо дополнительно выполнить ряд процедур:

- ***выбрать краевые условия***. Краевые условия представляют собой сведения о значениях фазовых переменных и (или) их производных на границах рассматриваемых пространственных и временных областей;

- ***дискретизировать задачу***. Дискретизация подразумевает разделение рассматриваемых пространственных и временных областей на конечное число элементарных участков с представлением фазовых переменных конечным числом значений в избранных узловых точках, принадлежащих элементарным участкам;

-***алгебраизировать задачу*** — аппроксимировать дифференциальные и интегральные уравнения алгебраическими.

Используют два основных подхода к дискретизации и алгебраизации краевых задач, составляющие сущность ***методов конечных разностей*** (МКР) и ***конечных элементов*** (МКЭ). С помощью любого из этих методов формируется окончательная модель, исследуемая при выполнении различных процедур анализа консультируемой проблемы.

Пользователь САК средствами входного языка задает исходную информацию о конфигурации консультируемой проблемы, о способе дискретизации — разделения среды на элементы, о физических свойствах участков среды. Формирование модели консультируемой

проблемы, т. е. разделение среды на элементы, выбор математических моделей элементов из заранее составленных библиотек, объединение моделей элементов в общую систему уравнений, так же как и решение получающихся уравнений, осуществляется автоматически на ЭВМ.

Основные уравнения математической физики, используемые в моделях консультируемых проблем. ***Процессы***, протекающие в технической консультируемой проблеме при ее функционировании, по своей физической природе могут быть разделены на ***электрические, тепловые, магнитные, оптические, механические, гидравлические*** и т. п. Каждому типу процессов в математической модели соответствует своя подсистема, основанная на определенных уравнениях математической физики.

Рассмотрим примеры уравнений, составляющих основу математических моделей технических консультируемых проблем на микроуровне.

Электрические процессы в полупроводниковых приборах с достаточной точностью удается описать с помощью ***уравнений непрерывности и Пуассона.***

Уравнения непрерывности выражают скорости изменения концентраций свободных носителей заряда в записываются отдельно для дырок и электронов:

$$\frac{\partial p}{\partial t} = -\frac{1}{q}\,\mathrm{div}\,\mathbf{J}_p + g_p; \tag{6.7}$$

$$\frac{\partial n}{\partial t} = \frac{1}{a}\,\mathrm{div}\,\mathbf{J}_n + g_n, \tag{6.8}$$

где p и n — концентрации дырок и электронов соответственно; q — заряд электрона; g_p и g_n — скорости процесса генерации-рекомбинации соответственно дырок и электронов;

$$\mathbf{J}_p = q\,(-\mu_p p\,\mathrm{grad}\,\varphi - D_p\,\mathrm{grad}\,p); \tag{6.9}$$

$$\mathbf{J}_n = q\,(-\mu_n n\,\mathrm{grad}\,\varphi + D_n\,\mathrm{grad}\,n) \tag{6.10}$$

— плотности дырочного и электронного токов; μ_p, μ_n — подвижности; D_p, D_n — коэффициенты диффузии дырок и электронов; φ — электрический потенциал.

Уравнения (6.7) —(6.8) показывают, что причинами изменения концентрации носителей могут быть неодинаковость числа носителей, втекающих (и вытекающих) в элементарный объем полупроводника (тогда div$\mathbf{J}\neq 0$), и нарушение равновесия между процессами генерации и рекомбинации носителей. Уравнения (6.9) и (6.10), называемые уравнениями плотности тока, характеризуют причины протекания электрического тока в полупроводнике: электрический дрейф под

воздействием электрического поля (grad φ ≠0*)* и диффузию носителей при наличия градиента концентрации.

Уравнение Пуассона характеризует зависимость изменений в пространстве напряженности электрического поля **E**=—gradφ от распределения плотности электрических зарядов ρ:

$$\mathrm{div}\mathbf{E} = \rho /(\varepsilon\varepsilon_0),$$

где ε — относительная диэлектрическая проницаемость среды;

ε_0 — диэлектрическая постоянная.

В качестве краевых условий в моделях полупроводниковых приборов используют ***зависимости потенциалов на контактах от времени,*** принимают значения концентраций носителей на границе между внешним выводом и полупроводником равными равновесным концентрациям p_0 и n_0, для границ раздела полупроводника и окисла задаются скоростью поверхностной рекомбинации g_s, что определяет величины нормальных к поверхности раздела составляющих плотностей тока $\mathbf{J}_p$ и $\mathbf{J}_n$, и т.д.

Результат решения уравнений непрерывности и Пуассона при известных краевых условиях — это ***поля потенциала и концентраций подвижных носителей*** в различных областях полупроводниковой структуры. Знание этих полей позволяет оценить электрические параметры прибора.

В ***основе моделей диффузионных процессов***, используемых, в частности, для описания технологических операций диффузии примесей при изготовлении интегральных схем и полупроводниковых приборов, лежит ***уравнение диффузии***

$$\partial N/\partial t = \mathrm{div}\,(D\mathrm{grad}N),$$

где N — концентрация примеси; D — коэффициент диффузии.

Краевые условия представлены зависимостью распределения примеси N в объеме полупроводника в начальный момент времени и зависимостью поверхностной концентрации от времени.

На использовании закономерностей протекания ***тепловых процессов*** основано действие многих теплофизических установок. В РЭА полезные свойства обусловлены закономерностями электрических процессов, однако рассеяние мощности и изменения температуры оказывают заметное влияние на характер функционирования аппаратуры. Поэтому в моделях РЭА, как и в моделях многих устройств иной природы, приходится учитывать тепловые процессы. ***Теплоперенос в твердых телах описывается уравнением теплопроводности***

$$C\rho \frac{\partial T}{\partial t} = \mathrm{div}\,(\lambda \mathrm{grad}\,T) + g_Q,$$

где T — температура; C — удельная теплоемкость; ρ — плотность; λ - коэффициент теплопроводности; g_Q — количество теплоты, выделяемой в единицу времени в единице объема.

При консультировании различных проблем машиностроения важное значение для определения выходных параметров имеет возможность исследования ***напряженно-деформированных состояний конструкций***. Подобные исследования важны и при проектировании несущих конструкций, механических и электромеханических устройств в составе РЭА. Знание ***напряжений и деформаций*** позволяет оценить ***прочность, долговечность, виброустойчивость аппаратуры.***

Модели для анализа напряжений и упругих деформаций твердых тел формируют с помощью основного уравнения теории упругости — ***уравнения Ламе***. Это уравнение получается из условия равновесия сил, действующих на элемент твердого тела в направлении оси x_i.

$$\sum_{j=1}^{3} \frac{\partial \sigma_{ij}}{\partial x_j} + g_{Mi} = \rho \frac{\partial^2 w_i}{\partial t^2}, \quad i = \overline{1,\ 3}. \tag{6.11}$$

где ρ — плотность; g_{Mi} — проекция вектора $\mathbf{G}_M$ массовых сил, приходящихся на единицу объема, на ось x_i; w_i — перемещение элемента вдоль оси x_i; σ_{ij} — напряжение, действующее вдоль оси x_i в грани элемента, перпендикулярной оси x_j. Напряжения σ_{ij} связаны с деформациями ε_{ij}, а последние — с перемещениями. В случае линейности связи σ_{ij} и ε_{ij} имеем

$$\sigma_{ii} = \lambda \sum_{j=1}^{3} e_{jj} + 2\mu e_{ii}, \tag{6.12}$$

а при $i \neq j$ имеем

$$\sigma_{ij} = 2\mu\varepsilon_{ij}, \tag{6.13}$$

где $\lambda = E_{ю}\nu/[(1+\nu)(1-2\nu)]$ и $\mu = E_{ю}/[2(1+\nu)]$—постоянные Ламе, характеризующие упругие свойства среды; $E_{ю}$ — модуль упругости; ν — коэффициент Пуассона. Учитывая, что

$$e_{ij} = \frac{1}{2}\left(\frac{\partial w_i}{\partial x_j} + \frac{\partial w_j}{\partial x_i}\right), \tag{6.14}$$

подстановкой (6.12) —(6.14) в (6.11) получаем уравнение Ламе

$$(\lambda + \mu)\ \mathrm{grad\ div}\ \mathbf{W} + \mu\Delta\ \mathbf{W} + \mathbf{G}_M = \rho \frac{\partial^2 \mathbf{W}}{\partial t^2}, \tag{6.15}$$

где $\mathbf{W}$ — вектор перемещений; Δ — оператор Лапласа.

Модели для анализа напряжений и деформаций часто оказываются более удобными, если ***представлены в интегральной форме***, вытекающей из вариационных принципов механики. ***Вариационный принцип Лагранжа*** *(принцип потенциальной энергии) гласит, что* ***потенциальная энергия системы*** *получает стационарное значение на тех кинематически возможных перемещениях, отвечающих заданным граничным условиям, которые* ***удовлетворяют условиям равновесия.***

Поэтому ***модель представляют*** в виде ***выражения потенциальной энергии П системы*** как разности энергии деформации Э и работы массовых и приложенных поверхностных сил *А:*

$$П = Э — A,$$

где

$$Э = 0{,}5\int_{R} \varepsilon^t \sigma \, dR;$$

$\varepsilon^t = (\varepsilon_{11}, \varepsilon_{22}, \varepsilon_{33}, \varepsilon_{12}, \varepsilon_{13}, \varepsilon_{23})^t$ —вектор-строка деформаций;
$\sigma=(\sigma_{11},\sigma_{22},\sigma_{33},\sigma_{12},\sigma_{13},\sigma_{23})$—вектор-столбец напряжений.

Вводя матрицу

$$\mathbf{D} = \begin{bmatrix} \lambda+2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & 2\mu & 0 & 0 \\ 0 & 0 & 0 & 0 & 2\mu & 0 \\ 0 & 0 & 0 & 0 & 0 & 2\mu \end{bmatrix},$$

можно (6.12) заменить более лаконичной записью:

$$\sigma = \mathbf{D}\varepsilon.$$

Таким образом, при использовании принципа Лагранжа вместо решения уравнения (6.15) требуется минимизировать функционал

$$П = 0{,}5 \int_{R} \varepsilon^t \mathbf{D}\, \varepsilon \, dR — A. \qquad (6.16)$$

В заключение рассмотрим основные ***уравнения газодинамики***, лежащие в основе ***моделей разнообразных пневматических и гидравлических устройств***.

Уравнение закона сохранения массы называют ***уравнением неразрывности:***

$$\partial\rho/\partial t = —\mathrm{div}(\rho\mathbf{U}),$$

где ρ — плотность; **U** — вектор скорости.

Уравнение закона сохранения количества движения в случае идеальной жидкости называют *уравнением Эйлера:*

$$d(\rho\mathbf{U})/dt = -\rho\,\mathbf{U}\mathrm{div}\mathbf{U} - \mathrm{grad}P. \qquad (6.17)$$

где P — давление.

Учет массовых сил и сил трения приводит к появлению дополнительных членов в правой части уравнения (6.17), которое называют в этом случае ***уравнением Навье — Стокса***.

Таким образом, ***основу большинства моделей консультируемых проблем на микроуровне составляют дифференциальные уравнения***. Далее будем использовать обобщенную форму записи уравнений

$$Lv(\mathbf{X}) = f(\mathbf{X}), \qquad (6.18)$$

где L — дифференциальный оператор; $v(\mathbf{X})$—зависимая фазовая переменная; $\mathbf{X} = (x_1, x_2, x_3)$ —вектор независимых (пространственных) координат; $f(\mathbf{X})$ —заданная функция.

Нестационарные уравнения в операторной форме можно записывать как

$$\partial v/\partial t + Lv = f(\mathbf{X}, t) \qquad (6.19)$$

или

$$\partial^2 v/\partial t^2 + Lv = f(\mathbf{X}, t), \qquad (6.20)$$

где v — функция $\mathbf{X}$ и времени t. Например, для двумерного уравнения диффузии

$$\partial N/\partial t - D(\partial^2 N/\partial x^2_1 + \partial^2 N/\partial x^2_2) = 0$$

используем запись $\partial N/\partial t = LN$, где

$$L = D(\partial^2/\partial x^2_1 + \partial^2/\partial x^2_2).$$

При необходимости применим и более лаконичную запись нестационарных уравнений в виде

$$\Lambda v(\mathbf{X},t) = f(\mathbf{X},t),$$

где Λ — дифференциальный оператор, включающий дифференцирование по всем независимым переменным.

Математическая модель должна содержать не только уравнения типа (6.19) или (6.20), но и краевые условия, представленные уравнениями

$$\alpha_1 v + \alpha_2 \Lambda v = \varphi(\mathbf{X}, t), \qquad (6.21)$$

где α_1, α_2 — коэффициенты; $\varphi(\mathbf{X}, t)$ —заданная функция.

Краевые условия состоят из граничных и начальных. Для граничных условий в (6.21) задается $\mathbf{X} \in \mathbf{\Gamma}$, где $\mathbf{\Gamma}$ — граница рассматриваемой пространственной области. Для начальных условий в (6.21) задается $t = t_{\text{нач}}$, где $t_{\text{нач}}$ — начальный момент времени.

Основные положения метода конечных разностей. Рассмотрим уравнение (6.18). Пусть $\mathbf{X} \in \mathbf{R}$ ($\mathbf{R}$ — ограниченная область изменения независимых переменных), заданы граничные условия.

Дискретизация задачи заключается в покрытии **R** сеткой и замене множества **R** конечным множеством точек $\mathbf{X}_k$, являющихся узлами сетки. Сетка может быть прямоугольной, косоугольной, с постоянными или переменными межузловыми расстояниями вдоль координатных осей (величинами шагов). Наиболее часто используют прямоугольную сетку с постоянными величинами шагов. На рис. 6.11 представлен фрагмент такой сетки для двумерной задачи с величинами шагов h_1 и h_2 вдоль координатных осей x_1 и x_2.

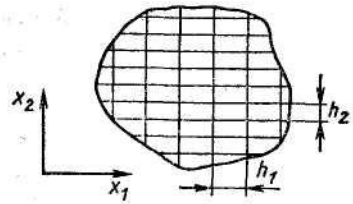


Рис. 6.11. Дискретизация пространтва с помощью сетки

Алгебраизация задачи заключается в замене дифференциального оператора Lv разностным. Это означает, что непрерывная переменная $v(\mathbf{X})$ заменяется конечным множеством значений $v_k=v(\mathbf{X}_k)$ в узлах сетки, а производные $\partial v/\partial \mathbf{X}$ аппроксимируются конечноразностными выражениями.

Применяется несколько способов выражения производных через значения v_k. Вид разностных операторов удобно представлять графически в форме шаблонов. На рис. 6.12, *а—г* даны примеры шаблонов для одномерных, а на рис. 6.12, *д, ж* — для двумерных стационарных задач.

Шаблон представляет собой часть сетки, включающую множество узлов $\mathbf{X}_k$, значения переменных в которых используются при аппроксимации производных в заданном узле $\mathbf{X}^*$. Узлы $\mathbf{X}_k$ на рис. 6.12 показаны темными кружками, а узел $\mathbf{X}^*$ обведен дополнительной окружностью. В левой части рисунков указан аппроксимируемый дифференциальный оператор, а рядом с узлами сетки записаны значения коэффициентов, с которыми соответствующие величины v_k входят в конечноразностное аппроксимирующее выражение.

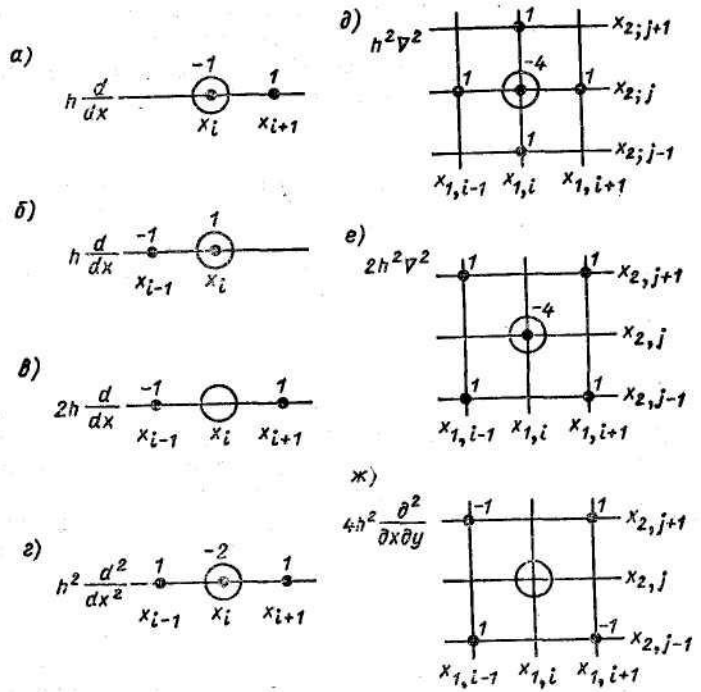


Рис. 6.12. Примеры шаблонов для разностных операторов

При этом принято $h_1=h_2=h$. Так, для рис. 6.12, *а* получаем

$$h\left.\frac{\partial v}{\partial x}\right|_{x^*} = v_{i+1} - v_i,$$

для рис. 6.12, *г*

$$h^2\left.\frac{d^2 v}{dx^2}\right|_{x^*} = v_{i+1} - 2v_i + v_{i-1}, \tag{6.22}$$

для рис. 6.12, *д*

$$h^2 \nabla^2 v|_{(x_{1i}, x_{2j})} = v_{i-1}^j + v_{i+1}^j + v_i^{j-1} + v_i^{j+1} - 4v_i^j \tag{6.23}$$

и т. д. В последнем выражении использовано обозначение

$$v_i^j = v(x_{1i}, x_{2j})$$

Подстановка выбранных аппроксимаций производных в исходное уравнение (6.18) преобразует его в систему разностных уравнений

$$\mathbf{F}(\mathbf{V})=0, \qquad (6.24)$$

где **V** — вектор, элементами которого являются значения фазовой переменной во внутренних узлах сетки. Аналогичная подстановка в исходное нестационарное уравнение (6.19) преобразует его в систему обыкновенных дифференциальных уравнений (ОДУ)

$$d\mathbf{V}/dt = \mathbf{F}(\mathbf{V}), \qquad (6.25)$$

которая вместе с заданными начальными условиями представляет собой задачу Коши.

Следовательно, ***дискретизация и алгебраизация уравнений*** в МКР сводит задачу ***анализа моделей на микроуровне к численному решению систем конечных (6.24) или обыкновенных дифференциальных (6.25) уравнений***.

Следует отметить, что точность аппроксимации растет с уменьшением величин шагов, однако при этом увеличивается порядок систем уравнений (6.24) или (6.25). Так, если окажется, что для достижения приемлемой точности рассматриваемую область **R** нужно делить вдоль каждой из координатных осей на 10^2 участков, то порядки систем уравнений (6.24) или (6.25) в одно-, дву- и трехмерных задачах составляют соответственно около 10^2, 10^4 и 10^6. Очевидно, что решение двумерных и особенно трехмерных задач требует значительных вычислительных ресурсов и тщательного отбора соответствующего математического обеспечения. Методы решения таких уравнений, применяемые в САК, рассматриваются в следующем разделе.

Основные положения метода конечных элементов. Рассмотрим применение МКЭ к решению задачи

$$L\nu(\mathbf{X}) = f(\mathbf{X}).$$

Дискретизация исследуемой пространственной области **R** в МКЭ осуществляется ее разделением на непересекающиеся подобласти — конечные элементы (КЭ). В одномерных задачах КЭ представляют собой ***отрезки линий***, в двумерных имеют ***форму треугольников***, прямоугольников, в трехмерных — ***тетраэдров, параллелепипедов*** и т. п. В пределах каждого КЭ выбирают конечное число узловых точек $\mathbf{X}_k$. Непрерывную фазовую переменную ν, фигурирующую в модели (6.18), заменяют конечным числом значений ν_k этой фазовой переменной в точках $\mathbf{X}_k$.

Возможности использования КЭ различной формы, размеров и пространственной ориентации обусловливают легкость дискретизации граничных условий при произвольной форме области **R**. Это обстоятельство — одно из основных преимуществ МКЭ перед МКР,

объясняющее широкое применение конечноэлементных представлений при ***моделировании процессов в элементах сложной конфигурации***.

Второе отличие МКЭ от МКР заключается в ***способе алгебраизации дифференциальных уравнений*** $Lv(\mathbf{X}) = f(\mathbf{X})$.

Если в МКР ***аппроксимируются производные*** $\partial v/\partial \mathbf{X}$, то в МКЭ ***аппроксимируется решение*** $v(\mathbf{X})$ ***некоторой функцией*** $u(\mathbf{X})$ с ***неопределенными коэффициентами***. Решение исходной задачи получается путем вычисления этих коэффициентов. В свою очередь задача вычисления коэффициентов формулируется как задача минимизации функционала, характеризующего качество аппроксимации решения $v(\mathbf{X})$ функцией $u(\mathbf{X})$, а эта задача сводится к решению системы алгебраических уравнений.

Очевидно, что чем сложнее применяемые аппроксимирующие функции и чем шире класс этих функций, тем сложнее задача формализации метода и его реализации в САК. Особенностью МКЭ является выбор аппроксимирующих функций для каждого КЭ в отдельности. Малые размеры КЭ позволяют использовать простые аппроксимирующие функции, причем одного и того же типа для всех КЭ определенной формы. Обычно в качестве $u(\mathbf{X})$ для отдельного КЭ применяют ***полиномы степени не выше третьей***, например в одномерном случае

$$u(x) = \sum_{i=0}^{r} a_i x^i.$$

В МКЭ $u(\mathbf{X})$ представляют в форме

$$u(\mathbf{X}) = \sum_{i=0}^{r} q_i \varphi_i(\mathbf{X}), \qquad (6.26)$$

где коэффициенты q_i имеют вполне определенный физический смысл — это значения аппроксимирующей функции в узловых точках;
$\varphi_i(\mathbf{X})$ —функции, называемые координатными (базисными, пробными или функциями формы); r — число узловых точек в конечном элементе.

В общем случае аппроксимации вектора $\mathbf{V}(\mathbf{X})$ в m-мерном пространстве выражение (6.26) принимает вид

$$\mathbf{U}(\mathbf{X}) = \mathbf{NQ}, \qquad (6.27)$$

где $\mathbf{U}$ — вектор размера $m \times 1$; $\mathbf{Q}$ — вектор размера $(mr) \times 1$;
$\mathbf{N}$ — интерполяционная матрица размера $m \times (mr)$, ее элементами являются координатные функции.

Важной составной частью МКЭ является выбор функционала, характеризующего качество используемой аппроксимации.

Примером такого функционала может служить следующий;

$$I = \int_{\mathbf{R}} (L\mathbf{U}(\mathbf{X}) - f(\mathbf{X}))^2 \, d\mathbf{R}, \tag{6.28}$$

где $\mathbf{X} \in \mathbf{R}$, **R** — исследуемая область. Очевидно, что интеграл *I*, относящийся ко всей области **R**, получается суммированием интегралов, подобных (6.28), но относящихся к отдельным КЭ.

Иногда математические модели консультируемых проблем на микроуровне уже в своем исходном виде могут быть представлены в ***вариационной формулировке***, т. е. в виде ***задачи минимизации функционала.*** Типичным примером таких моделей служат модели, описывающие статические напряженно-деформированные состояния элементов. В этих моделях в качестве минимизируемого функционала используется выражение полной потенциальной энергии (6.16)

$$\Pi = \frac{1}{2} \int_{\mathbf{R}} \boldsymbol{\varepsilon}^t \mathbf{D} \boldsymbol{\varepsilon} d\mathbf{R} - A, \tag{6.29}$$

а в качестве искомой экстремали функционала фигурирует вектор перемещений $\mathbf{W}(\mathbf{X})$, $\mathbf{X} \in \mathbf{R}$. Деформации ε_{ij} связаны с перемещениями w_i соотношениями (6.14), что можно выразить в матричной форме записью

$$\begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \varepsilon_{12} \\ \varepsilon_{13} \\ \varepsilon_{23} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 2\frac{\partial}{\partial x_1} & 0 & 0 \\ 0 & 2\frac{\partial}{\partial x_2} & 0 \\ 0 & 0 & 2\frac{\partial}{\partial x_3} \\ \frac{\partial}{\partial x_2} & \frac{\partial}{\partial x_1} & 0 \\ \frac{\partial}{\partial x_3} & 0 & \frac{\partial}{\partial x_1} \\ 0 & \frac{\partial}{\partial x_3} & \frac{\partial}{\partial x_2} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}, \tag{6.30}$$

или более лаконично

$$\boldsymbol{\varepsilon} = \mathbf{SW}, \tag{6.31}$$

где **S** — фигурирующая в (6.30) матрица-оператор дифференцирования.

Подставляя (6.31) в (6.29) и заменяя вектор **W** его аппроксимацией **U = NQ** в соответствии с (6.27), получаем

$$\Pi = 0{,}5 \int_{\mathbf{R}} \mathbf{Q}^t \mathbf{B}^t \mathbf{D}\mathbf{B}\mathbf{Q} d\mathbf{R} - A,$$

где **B** = **SN**. Обозначим

$$\mathbf{K} = \int_{\mathbf{R}} \mathbf{B}^t \mathbf{D}\mathbf{B} d\mathbf{R} \qquad (6.32)$$

и назовем ***матрицей жесткости***. Тогда Π=0,5$\mathbf{Q}^t\mathbf{K}\mathbf{Q}$—$A$. В соответствии с принципом Лагранжа дифференцируем Π по вектору **Q** и приравниваем результат нулю, получаем систему алгебраических уравнений

$$\mathbf{K}\mathbf{Q} = \mathbf{P}, \qquad (6.33)$$

где $\mathbf{P}=\partial A/\partial \mathbf{Q}$— вектор правых частей, называемый ***вектором нагрузок***.

Матрица жесткости **К** всего исследуемого элемента составляется из матриц жесткости $\mathbf{K}_{ij}$ отдельных КЭ. Матрицы $\mathbf{K}_{ij}$ несут информацию о конфигурации и упругих свойствах материала конечных элементов и подсчитываются по формуле (6.32), в которой при этом под **R** понимается подобласть, относящаяся к рассматриваемому КЭ.

Система уравнений типа (6.33) получается при применении МКЭ к решению и других стационарных уравнений.

При этом требуется минимизировать некоторый функционал, например (6.28), что также приводит к решению системы алгебраических уравнений

$$\partial I/\partial \mathbf{Q} = 0.$$

В случае нестационарных уравнений основные положения МКЭ — деление на КЭ, подбор аппроксимирующей функции и минимизируемого функционала — по-прежнему применяют по отношению к пространственной области. Тогда получаем систему обыкновенных дифференциальных уравнений (ОДУ)

$$d\mathbf{Q}/dt = \mathbf{F}(\mathbf{Q}),$$

решением которой являются зависимости фазовых переменных $\mathbf{Q}(t)$ в узловых точках от времени.

6.6. Предпосылки автоматического формирования аналитических моделей консультируемых проблем

Основная часть реализуемых в САК расчетных работ (по объему и затратам) выполняется на количественном уровне моделирования (детальном и адекватном). В связи с этим целесообразно автоматизировать везде, где возможно, процедуру формирования аналитической модели консультируемой проблемы (ее элементов) на

выбранном уровне структурирования и абстрагирования. Модель консультируемой проблемы (ее элемента) при детальном и адекватном моделировании чаще всего представляется ориентированным графом (схемой замещения на уровне минимальных элементов) с подробным описанием весов (передач) всех ребер графа (компонентных уравнений для элементов схемы замещения). На основании такой модели может быть построена ее математическая форма (математическая модель) в виде системы уравнений (алгебраических, дифференциальных, смешанных, линейных или нелинейных, непрерывных или разностных и т. д.), описывающих закон функционирования и характер изменения переменных консультируемой проблемы (компонента).

При количественном моделировании консультируемых проблем, модели которых представлены орграфами, важную роль играют следующие законы равновесия:

1. Сумма последовательных переменных у каждой вершины графа модели (или в любом его сечении) равна нулю.

2. Сумма параллельных переменных по любому замкнутому гонтуру графа модели (или в любом его сечении) равна нулю.

По аналогии с электрическими цепями эти законы часто называют ***обобщенными законами Кирхгофа*** и интерпретируют в зависимости от физической природы консультируемой проблемы в виде ***уравнений балансов токов и напряжений*** (радиоэлектронные цепи), ***расходов и падения давления*** (пневмогидравлические цепи), ***сил и скоростей*** (механические устройства), ***тепловых потоков и разности температур*** (теплотехнические цепи) и др.

Положим для общности, что консультируемая проблема состоит из ***двухполюсных*** и ***многополюсных*** компонентов (рис. 6.13) с ***последовательными переменными*** η и ***параллельными переменными*** ξ, которые могут быть описаны соответствующими полюсными графами.

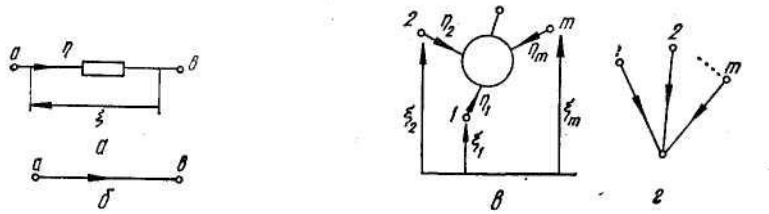


Рис. 6.13. Схема обобщенного двухполюсника (*а*) и его полюсный граф *(б);* схема многополюсного компонента (*в*) и его полюсный граф (*г*)

Физический смысл переменных η и ξ для различных физических консультируемых проблем поясняется в табл. 6.3.

Таблица 6.3

Физический сигнал переменных η и ξ

Объекты	η	ξ
Электрические	Ток	Напряжение
Механические поступательные	Сила	Скорость
Механические упругие	Сила	Деформация
Механические вращательные	Вращательный момент	Угловая скорость
Тепловые	Тепловой поток	Температура
Гидравлические и пневматические	Поток (расход)	Давление

Тогда обобщенные законы Кирхгофа в матричной форме принимают вид

$$\Pi\eta = 0; \qquad (6.34)$$

$$P\xi = 0, \qquad (6.35)$$

где П, Р — матрицы независимых сечений и контуров соответственно. Независимым сечением (контуром) является такое сечение (контур), которое отличается от предыдущих сечений (контуров) хотя бы одним новым ребром.

Известно, что если граф модели консультируемой проблемы содержит v вершин и l ребер, то размерности указанных матриц соответственно

$$\Pi = [v - 1] \times [v - 1];$$
$$P = [l - v + 1] \times [l - v + 1],$$

а общее число уравнений (6.34) и (6.35) равно l, хотя число переменных $\xi + \eta = 2l$.

Независимость сечений и контуров удобно устанавливать с помощью дерева графа, которое соответствует произвольной совокупности ребер, объединяющих все вершины *1*—7 и не образующих ни одного замкнутого контура (рис. 6.14, *а)*.

Ясно, что выбор дерева неоднозначен, поэтому следует установить определенную иерархию приоритетов для ребер графа при выборе дерева, которая позволяет реализовать некоторые дополнительные условия (например, получить специфический вид уравнений математической модели).

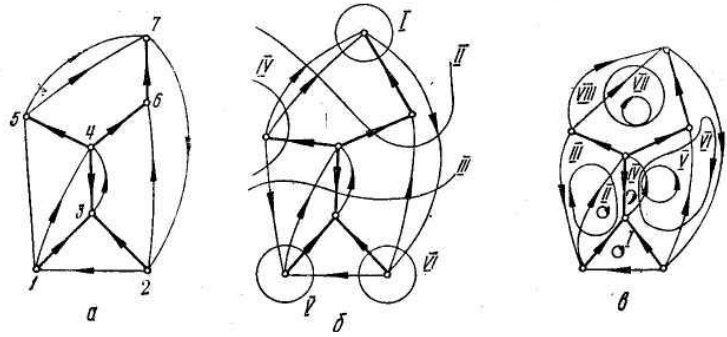


Рис. 6.14. Граф с выделенным деревом (*а*), его независимые сечения *(б)* и независимые контуры (в)

Тогда каждому ребру дерева графа соответствует сечение *I—VI* (рис. 6.14, *б),* инцидентное только одному ребру дерева, а также инцидентное другим ребрам графа, называемым хордами. Последние образуют дополнение графа. Нетрудно видеть, что при удалении совокупности всех ребер, принадлежащих сечению, граф распадается на две или более частей. Каждой хорде, т. е. ребру, не вошедшему в дерево, соответствует контур *I—VIII* (рис. 6.14, *в),* в который, кроме данной хорды входят (инцидентны) только ребра дерева графа.

Сечении и контуры можно рассматривать как некоторую систему координат, относительно которой записываются уравнения консультируемой проблемы. Если принять в качестве независимых параллельные переменные ребер дерева и последовательные переменные хорд η_N, то остальные переменные (ξ и η_T) могут быть найдены на основании обобщенных законов Кирхгофа из соответствующих сечений и контуров. Тогда

$$\eta_T = -\pi\eta_N, \qquad (6.36)$$

или

$$[1\pi]\begin{bmatrix}\eta_T\\ \eta_N\end{bmatrix} = \Pi\eta = 0,$$

если осуществить преобразование формы матрицы сечений к виду $[1\pi]$, где π— матрица сечений для хорд размером $[v - 1] \times [l-v+1]$;

$$\xi_N = -\rho\xi_T \qquad (6.37)$$

или

$$[1\rho]\begin{bmatrix}\xi_N\\ \xi_T\end{bmatrix} = \rho\xi = 0,$$

если осуществить преобразование формы матрицы контуров к виду $[\rho 1]$,

где ρ — матрица контуров для ребер дерева размером

$$[l - v + 1] \times [v - 1].$$

Каждому ребру графа приписывается некоторое направление, совпадающее с направлением ибследовательной переменной η_i, поэтому сечения и контуры также имеют направления, определяемые направлением инцидентных ребер дерева или хорд соответственно.

К решению задачи о выборе дерева имеется два подхода. В соответствии с первым из них дерево строится на множестве вершин графа без учета структуры последнего, и дерево представляет собой самостоятельный граф (например, совокупность узловых величин (v_1 — v_4), совмещенный с графом объекта в его вершинах *1—4* (рис. 6.15).

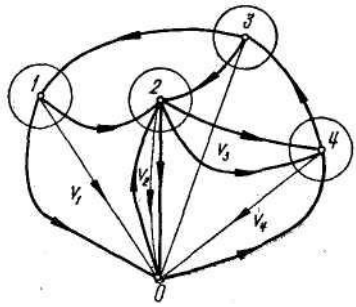


Рис. 6.15. К пояснению выбора дерева в виде совокупности узловых величин v_i

В этом случае базовые уравнения законов равновесия преобразуются к виду:

$$A\eta = 0; \qquad (6.38)$$

$$\xi = A^t v, \qquad (6.39)$$

где A — матрица инциденций, которая отображает взаимоотношение вершин и ребер графа и имеет размерность такую же, как матрица сечений П. По существу, матрица A и есть матрица сечений (кононических), выбранных так, что каждое сечение охватывает отдельную вершину графа, которой инцидентно только одно ребро дерева — соответствующая узловая величина.

При другом подходе в качестве ребер выбирают ребра самого графа (рис. 6.14). При этом выбор дерева и формирование топологического описания графа в виде уравнений законов равновесия составляет отдельную процедуру, целью которой является получение одной из топологических матриц π или ρ (они связаны зависимостью $\rho = -\pi^t$).

Искомые топологические матрицы можно сформировать путем преобразований исходной матрицы инциденций графа A к виду $[1\pi]$, если использовать алгоритм Гаусса. При этом ребра, определяющие столбцы этой преобразованной матрицы, которые относятся к единичной матрице размера $[v - 1] \times [v - 1]$, являются ребрами дерева графа, а остальные — хордами. Более экономно формировать матрицы π и ρ непосредственно на множестве ребер и вершин графа консультируемой проблемы при использовании только логических операций. Исходными данными служат: число вершин графа v, число ребер графа l и массив ребер *MVT*, представляющий собой упорядоченные пары инцидентных им вершин графа (начальная и конечная вершины). Дерево формируется методом Краскала (методом стягивания вершин на графе). Процедура формирования сводится к последовательному просмотру ребер и их распределению между деревом и хордами, или дополнением. Очередное ребро включают в дерево, если оно не образует контура с выбранной ранее совокупностью-ребер дерева.

Пример 7. Для графа, изображенного на рис. 6.16.

$v = 5$; $l=8$;

$$MVT\ (5,8) = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline 2 & 5 & 5 & 4 & 1 & 4 & 2 & 1 \\ \hline 3 & 3 & 4 & 3 & 4 & 2 & 1 & 5 \\ \hline \end{array}.$$

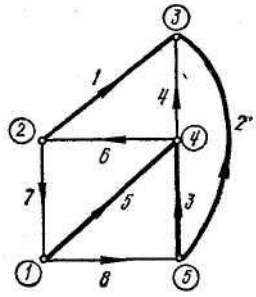


Рис. 6.16. Граф, для которого строится матрица π

Для иллюстрации работы логического алгоритма выбора дерева и матрицы π построим вспомогательные массивы ребер дерева *MDR*(v — 1), хорд *MDO*(l—v+1) и массив вершин *MVR*(v), начальное содержимое которых для нашего примера устанавливаем в виде:

$$MDR = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 4 \\ \hline & & & \\ \hline \end{array}; \quad MDO = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 4 \\ \hline & & & \\ \hline \end{array};$$

$$MVR = \begin{array}{|c|c|c|c|c|}\hline 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 2 & 3 & 4 & 5 \\ \hline \end{array}.$$

Полагая, что ребра графа пронумерованы в соответствии с некоторым приоритетом, включаем ребро *1* в дерево и стянем его вершины *(2, 3)* в одну, которой присваиваем номер начальной вершины. В результате содержимое массивов *MDR* и *MVR* изменится следующим образом:

$$MDR = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 4 \\ \hline 1 & & & \\ \hline \end{array};$$

$$MVR = \begin{array}{|c|c|c|c|c|}\hline 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 2 & 2 & 4 & 5 \\ \hline \end{array}.$$

Если включить в дерево ребро *2* с инцидентными вершинами *5, 2 а* также стянуть их в одну, то

$$MDR = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 4 \\ \hline 1 & 2 & & \\ \hline \end{array};$$

$$MVR = \begin{array}{|c|c|c|c|c|}\hline 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 5 & 5 & 4 & 5 \\ \hline \end{array}.$$

После включения в дерево ребра *3* и стягивания его преобразованных вершин получаем:

$$MDR = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 4 \\ \hline 1 & 2 & 3 & \\ \hline \end{array};$$

$$MVR = \begin{array}{|c|c|c|c|c|}\hline 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 5 & 5 & 5 & 5 \\ \hline \end{array}.$$

Очередное ребро *4* с начальными номерами инцидентных вершин *(4, 3)* к данному этапу превращается в петлю, так как вершины *2, 3, 4* и *5* уже стянуты в одну комплексную вершину. Поэтому ребро *4* относится к хордам:

$$MDO = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 4 \\ \hline 4 & & & \\ \hline \end{array}.$$

Ребро 5, соединяющее вершины *1, 4,* оказывается подключенным к вершинам (*1*, 5) и может быть отнесено к дереву. В результате

$$MDR = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 4 \\ \hline 1 & 2 & 3 & 5 \\ \hline \end{array};$$

$$MVR = \begin{array}{|c|c|c|c|c|}\hline 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline \end{array}.$$

Все остальные ребра отнесены к дополнению (хордам):

$$MDO = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 4 \\ \hline 4 & 6 & 7 & 8 \\ \hline \end{array}.$$

После того, как сформировано дерево, начинается формирование матрицы сечений π методом исключения висячих вершин.

Вершину называют висячей, если она инцидентна лишь одному ребру дерева. Ребро дерева, инцидентное висячей вершине, называют висячим ребром. Очевидно, что в сечение висячего ребра дерева войдут те и только те хорды, которые инцидентны висячей вершине. Сначала осуществляется поиск висячих вершин (ВВ) на множество ребер дерева *MDR* о учетом информации о их начальных (НВ) и конечных (КВ) вершинах:

$$MDR = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 4 \\ \hline 1 & 2 & 3 & 5 \\ \hline \end{array};$$

$$MVR\ (\text{НВ}) = \begin{array}{|c|c|c|c|}\hline 2 & 5 & 5 & 1 \\ \hline \end{array};$$

$$MVR\ (\text{КВ}) = \begin{array}{|c|c|c|c|}\hline 3 & 3 & 4 & 4 \\ \hline \end{array}.$$

Просмотр подмассива *MVR,* относящегося к ребрам дерева, позволяет идентифицировать висячие вершины *1* и *2,* а также висячие ребра дерева *5* и *1.*

Затем формируются строки матрицы π, соответствующие висячим ребрам дерева графа (1-я и 4-я строки). Если в общем случае v_i — висячая вершина, а l_j — висячее ребро, то множество элементов матрицы π пополняется группой ненулевых единичных элементов, соответствующих строке l_j матрицы с индексами столбцов, определяемыми по элементам массива хорд *MDO* с признаками НВ *(MDO(k))* = v_i или КВ *(MDO(k))* = v_i. Знаки этих элементов будут

положительными, если ребро дерева и хорда одинаково ориентированы относительно висячей вершины, и отрицательными - в противном случае. Висячих вершин может быть несколько (число их тем больше, чем разветвленнее дерево графа). В этом случае за один шаг алгоритма заполняются несколько строк матрицы π.

Для данного примера после определения висячих вершин *1* и *2* просматривается массив *MDO* и находится местоположение номеров этих вершин среди НВ и КВ ребер массива *MDO:*

	1	2	3	4
MDO =	4	6	7	8

MVR (НВ) =	4	4	2	1

MVR (КВ) =	3	2	1	5

Следовательно, соответствующие строки матрицы π принимают вид:

	4	6	7	8		
π =		—1	1		1	Номера ребер дерева
					2	
					3	
			—1	1	5	

Далее во все ячейки начального массива *MVR,* содержащие номера висячих вершин, заносятся номера вершин, связанных с ВВ, т. е. производится стягивание вершин висячих ребер графа.

Для данного примера массив *MVR* видоизменяется следующим образом:

	1	2	3	4	5
MVR =	4	3	3	4	5

Из массива ребер дерева *MDR* исключаем использованные ветви (1 и 5) для ограничения поля поиска ВВ на следующем шаге, для которого

$$MDR = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 4 \\ \hline & 2 & 3 & \\ \hline \end{array};$$

$$MVR\ (\text{НВ}) = \begin{array}{|c|c|}\hline 5 & 5 \\ \hline \end{array};$$

$$MVR\ (\text{КВ}) = \begin{array}{|c|c|}\hline 3 & 4 \\ \hline \end{array},$$

поэтому на втором шаге вершины *3* и *4* становятся висячими. Формируем соответствующие 2-ю и 3-ю строки матрицы π:

$$MDO = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 4 \\ \hline 4 & 6 & 7 & 8 \\ \hline \end{array};$$

$$MVR\ (\text{НВ}) = \begin{array}{|c|c|c|c|}\hline 4 & 4 & 3 & 4 \\ \hline \end{array};$$

$$MVR\ (\text{КВ}) = \begin{array}{|c|c|c|c|}\hline 3 & 3 & 4 & 5 \\ \hline \end{array};$$

$$\pi = \begin{array}{|c|c|c|c|}\hline 4 & 6 & 7 & 8 \\ \hline & & & & 1 \\ \hline 1 & 1 & -1 & & 2 \\ \hline -1 & -1 & 1 & -1 & 3 \\ \hline & & & & 5 \\ \hline \end{array}.$$

Снова стягиваем вершины висячих ребер дерева и получаем

$$MVR = \begin{array}{|c|c|c|c|c|}\hline 1 & 2 & 3 & 4 & 5 \\ \hline 5 & 5 & 5 & 5 & 5 \\ \hline \end{array},$$

$$MDR = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 4 \\ \hline & & & \\ \hline \end{array}.$$

Так как все ребра дерева уже использованы, то на этом формирование матрицы π прекращается. Путем транспонирования и изменения знаков всех элементов на противоположные из матрицы π получаем матрицу контуров для ветвей дерева ρ.

Таким образом, для графа, изображенного на рис. 6.16, справедливы следующие соотношения между продольными и параллельными переменными его ребер, определяемые структурой графа:

$$\begin{bmatrix}\eta_1\\ \eta_2\\ \eta_3\\ \eta_4\end{bmatrix} = -\begin{bmatrix}0 & -1 & 1 & 0\\ 1 & 1 & -1 & 0\\ -1 & -1 & 1 & -1\\ 0 & 0 & -1 & 1\end{bmatrix}\begin{bmatrix}\eta_4\\ \eta_6\\ \eta_7\\ \eta_8\end{bmatrix};$$

$$\begin{bmatrix}\xi_4\\ \xi_6\\ \xi_7\\ \xi_8\end{bmatrix} = \begin{bmatrix}0 & 1 & -1 & 0\\ -1 & 1 & -1 & 0\\ 1 & -1 & 1 & -1\\ 0 & 0 & -1 & 1\end{bmatrix}\begin{bmatrix}\xi_1\\ \xi_2\\ \xi_3\\ \xi_5\end{bmatrix}.$$

Следует отметить, что топологические уравнения в любой форме (6.34)—(6.39), отражающие структурные связи в консультируемой проблеме, всегда линейны.

Помимо топологических уравнений при количественном моделировании консультируемых проблем важную роль играют также ***компонентные уравнения***, характеризующие взаимосвязи продольных и параллельных переменных ребер графа:

$$K(\xi, \eta) = 0. \qquad (6.40)$$

В отличие от топологических компонентные уравнения, как правило, нелинейны. В процессе вычислений на ЭВМ уравнение (6.40) в пределах временного шага линеаризуется в соответствии с выражением

$$\frac{\partial K}{\partial \xi}(\xi^{(j)} - \xi^{(j-1)}) + \frac{\partial K}{\partial \eta}(\eta^{(j)} - \eta^{(j-1)}) = 0;$$

$$\frac{\partial K}{\partial \xi}\xi^{(j)} + \frac{\partial K}{\partial \eta}\eta^{(j)} = \frac{\partial K}{\partial \xi}\xi^{(j-1)} + \frac{\partial K}{\partial \eta}\eta^{(j-1)},$$

или

$$G^{(j)}\xi^{(j)} + R^{(j)}\eta^{(j)} = W^{(j)}, \qquad (6.41)$$

где

$$G^{(j)} = \frac{\partial K}{\partial \xi};\ R^{(j)} = \frac{\partial K}{\partial \eta}.$$

Принято различать *Y*-ребра (компоненты) и Z-ребра (компоненты) в зависимости от выбора управляющей переменной в выражении (6.40), для которых соответственно:

$$\eta = K_1(\xi);$$

$$\eta^{(j)} = \frac{\partial K_1}{\partial \xi}\Big|_{\xi^{(j-1)}} \xi^{(j)} + W^{(j)},$$

где вводится независимый источник

$$W^{(j)} = K_1(\xi^{(j-1)}) - \frac{\partial K_1}{\partial \xi}\Big|_{\xi^{(j-1)}} \xi^{(j-1)},$$

$$\xi = K_2(\eta);$$

и

$$\xi^{(l)} = \left.\frac{\partial K_2}{\partial \eta}\right|_{\eta^{(l-1)}} \eta^{(l)} + W^{(l)},$$

где вводится независимый источник

$$W^{(l)} = K_2(\eta^{(l-1)}) - \left.\frac{\partial K_2}{\partial \eta}\right|_{\eta^{(l-1)}} \eta^{(l-1)}.$$

Временно́й шаг, а следовательно, интервал линеаризации автоматически выбирается, исходя из заданной погрешности вычислений.

Компонентные уравнения (6.40) отражают помимо ***нелинейности*** также ***инерционность***, присущую консультируемой проблеме и ее компонентам. Часто инерционные свойства отдельных компонент выражаются уравнениями вида

$$\eta_C = M\frac{d\xi_C}{dt}, \tag{6.42}$$

или

$$\xi_L = N\frac{d\eta_L}{dt}. \tag{6.43}$$

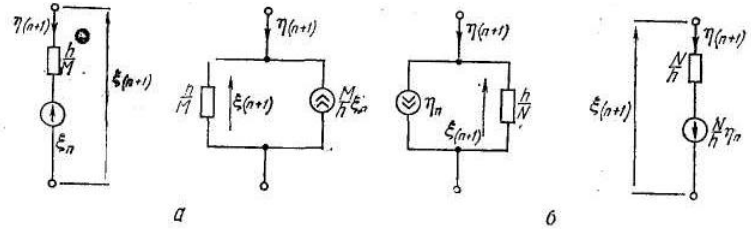
В процессе вычислений на ЭВМ уравнения (6.42-6.43) алгебраизируются применением аппроксимирующих выражений для производной, полученных на основе численных методов интегрирования дифференциальных уравнений. Например, для уравнения (6.42), учитывая неявный метод Эйлера

$$\xi_{C(n+1)} = \xi_{C(n)} + h\dot{\xi}_{C(n+1)},$$

получаем:

$$\left.\begin{aligned} \xi_{C(n+1)} &= \xi_{C(n)} + \frac{h}{M}\eta_{C(n+1)}; \\ \eta_{C(n+1)} &= \frac{M}{h}\xi_{C(n+1)} - \frac{M}{h}\xi_{C(n)}. \end{aligned}\right\} \tag{6.44}$$

Выражениям (6.44) соответствуют дискретные схемы замещения (рис. 6.17, *а),* справедливые для рассматриваемого интервала времени $t_n < t < t_{n+1}$.



а *б*

Рис. 6.17. Схемы замещения инерционного компонента с уравнением $\eta = Md\xi/dt$ *(а)* и уравнением $\xi = Nd\eta/dt$ *(б)*

Аналогично для уравнения (6.43) на основе неявного метода Эйлера находим выражения:

$$\left.\begin{aligned}\eta_{L(n+1)} &= \eta_{L(n)} + \frac{h}{N}\xi_{L(n+1)};\\ \xi_{L(n+1)} &= \frac{N}{h}\eta_{L(n+1)} - \frac{N}{h}\eta_{L(n)},\end{aligned}\right\} \qquad (6.45)$$

которым соответствуют дискретные схемы замещения, показанные на рис. 6.17, *б.* Таким образом, инерционность элемента консультируемой проблемы учитывается введением в ее схему замещения ***дополнительных независимых источников, характеризующих состояние переменных элемента в предыдущий момент времени***. На каждом новом временном шаге параметры компонентов такой схемы замещения обновляются. Этот же принцип сохраняется, если вместо метода Эйлера использовать другие более сложные методы численного интегрирования.

Сложность исходных компонентных уравнений (6.40) зависит от того, используются ли микро- или макромодели для элементов проблемы. При переходе к макромоделям упрощается структура модели консультируемой проблемы, а следовательно, ее топологические уравнения, но, как правило, существенно усложняются компонентные уравнения.

Применение макромоделей гарантирует возможность моделирования сложных консультируемых проблем, поэтому рассмотрим основные методы их получения.

1. Огрубление микромодели консультируемой проблемы (элемента). В этом случае из исходной модели последовательно отключаются отдельные компоненты (закорачиваются или размыкаются) до тех пор, пока входные и выходные характеристики консультируемой проблемы не станут отличаться от начальных значений на заданную величину. Избежать перебора можно, если воспользоваться анализом чувствительности характеристик микромодели к вариациям параметров компонентов на отдельных участках рабочего режима консультируемой проблемы.

Понятно, что число оставшихся компонент, характеризующихся наибольшей чувствительностью и образующих макромодель, будет зависеть от заданной ее точности. При точности 80—85% существенного упрощения исходной микромодели при таком подходе достигнуть не удается, хотя такую процедуру получения макромодели консультируемой проблемы можно автоматизировать.

2. Аппроксимация внешних характеристик консультируемой проблемы. Исследуются внешние реакции консультируемой

проблемы, на основании которых строится структура макромодели. Параметрами последней служат входное и выходное сопротивления, задержки, коэффициенты передач и другие величины, характеризующие свойства консультируемой проблемы на внешних зажимах. При этом внутренняя структура макромодели не совпадает с действительной структурой консультируемой проблемы и даже может не иметь физического смысла.

Для формирования такой модели требуется выполнить большой объем исследований характеристик консультируемой проблемы и анализа их результатов. Успех определяется хорошим знанием консультируемой проблемы разработчиком макромодели.

3. Формирование табличных массивов. Результаты всестороннего исследования консультируемой проблемы оформляются в виде многомерного массива табличных данных, содержащих информацию о входных и выходных переменных консультируемой проблемы для различных режимов функционирования. При этом возможно сокращение объема вычислений, так как решение сложных нелинейных уравнений заменяется процедурой выбора нужных данных из табличного массива с необходимой интерполяцией.

Для повышения точности описания консультируемой проблемы такой массив табличных данных быстро растет. В принципе процедура формирования табличных массивов поддается автоматизации.

4. Использование математических выражений и уравнений. При этом подходе стремятся получить аналитические выражения связи входных и выходных переменных консультируемой проблемы. Эти выражения могут включать в себя полиномиальные функции (для зависимых источников с задержками и нелинейных компонентов), логические функции *И* и *ИЛИ* при «сшивании» отдельных формул, функции распределения вероятности для описания вариаций времен задержек и уровней переменных.

От консультанта (разработчика) при выводе таких выражений требуется высокая квалификация, сам же процесс вывода автоматизировать трудно.

5. Использование математических методов редукции. В отличие от первого подхода, базирующегося на удалении отдельных компонент, в рассматриваемом случае сокращению подвергается исходная математическая модель консультируемой проблемы путем матричных преобразований, переупорядочения, Гауссова исключения, анализа собственных значений и др. Формально понижают порядок описывающей системы уравнений для консультируемой проблемы.

При этом процесс понижения порядка систем уравнений можно автоматизировать.

6. Комбинированные методы. На практике при разработке макромоделей консультируемых проблем и их элементов используют в различном сочетании перечисленные выше методы, а при моделировании сложных консультируемых проблем — макромодели различных типов.

Например, в макромоделях логических цифровых элементов могут использоваться математические выражения (4-й метод) для отображения реализуемых логических отношений, а аппроксимация (2-й метод) входных и передаточных характеристик базовой электронной схемы описывается в табличной форме (3-й метод) и т. д. Аппроксимация может быть универсальной или зависеть от типа логического элемента. Часто используется кусочно-линейная аппроксимация, а параметры аппроксимации определяются заданными значениями логических уровней, порогов срабатывания и др., или получаются в результате факторных экспериментов.

6.7. Типы аналитических моделей консультируемых проблем

Различные типы математических моделей консультируемой проблемы получают различным сочетанием уравнений равновесия и компонентных уравнений.

Модели ПГМ и ПГМ$_0$ (полная гибридная и табличная). Уравнения ПГМ консультируемой проблемы размером

$$[2l + v - 1] \times [2l + v - 1]$$

образуются простым сочетанием уравнений равновесия (6.38), (6.39) и компонентных уравнений (6.41):

$$\begin{bmatrix} A^t & & -1 \\ & A & \\ & \frac{\partial K}{\partial \eta} & \frac{\partial K}{\partial \xi} \end{bmatrix} \begin{bmatrix} v \\ \eta \\ \xi \end{bmatrix} = \begin{bmatrix} \\ \\ W \end{bmatrix}. \tag{6.46}$$

Более распространена табличная модель ПГМ$_0$, имеющая несколько меньшую размерность $[2l \times 2l]$ и образованная сочетанием уравнений (6.34), (6.35) с уравнениями (6.41):

$$\begin{bmatrix} \Pi & \\ & P \\ \frac{\partial K}{\partial \eta} & \frac{\partial K}{\partial \xi} \end{bmatrix} \begin{bmatrix} \eta \\ \xi \end{bmatrix} = \begin{bmatrix} \\ \\ W \end{bmatrix}. \tag{6.47}$$

Повышенная размерность этих моделей не служит препятствием при их использовании на ЭВМ. Этому способствует специальная методика кодирования и обработки разреженных матриц, т. е. матриц, имеющих незначительное число ненулевых элементов (НЭ).

В соответствии с этой методикой матрицы уравнений (6.46) и (6.47) могут быть представлены в ЭВМ, например, в виде списков *IC, IR* и *VA* или других подобных одномерных массивов. При этом в списке *IC* хранятся индексы столбцов НЭ, в списке *VA* — абсолютные значения этих элементов, а в списке *IR* — информация о числе НЭ в каждой строке исходной матрицы, так что содержимое ячейки списка *IR* указывает на номер ячейки списка *IC,* начиная с которой в нем хранятся данные о НЭ рассматриваемой строки.

Пример 8.

Информация, содержащаяся в матрице

$$\begin{bmatrix} a_{11} & & a_{13} & & & a_{16} \\ & a_{22} & & & & \\ & & a_{33} & & & a_{36} \\ a_{41} & & & & a_{45} & \\ & a_{52} & & & & \\ a_{61} & & & & & a_{66} \end{bmatrix},$$

кодируется следующими списками:

IR =	1	4	5	7	9	10					
IC =	1	3	6	2	3	6	1	5	2	1	6
	1	2	3	4	5	6	7	8	9	10	11
VA =	a_{11}	a_{13}	a_{16}	a_{22}	a_{33}	a_{36}	a_{41}	a_{45}	a_{52}	a_{61}	a_{66}

Число ненулевых элементов в строке определяется разностью содержимого двух соседних ячеек массива *IR*. Важно отметить, что все последующие процедуры вычислений строятся с учетом возможности непосредственного обращения к этим или подобным спискам (например, выбор по ним требуемого НЭ или занесения в списки нового НЭ), при этом используются только ненулевые элементы матриц уравнений консультируемой проблемы.

Имеется еще одна возможность кодирования разреженных матриц, так называемое «позиционное» кодирование, когда каждой позиции элемента матрицы сопоставляется свой номер:

1	2	...	...	n
$n+1$	$n+2$	...	...	$2n$
$2n+1$	$2n+2$	...	...	$3n$
...	...	...	...	...
n^2-n+1	...	...	...	n^2

.

В этом случае используются два списка: список *NR* абсолютных номеров позиций НЭ и список *VA*. В частности, для матрицы из примера 8

$$NR = \boxed{1 \mid 3 \mid 6 \mid 8 \mid 15 \mid 18 \mid 19 \mid 23 \mid 26 \mid 31 \mid 36};$$

$$VA = \boxed{a_{11} \mid a_{13} \mid a_{16} \mid a_{22} \mid a_{33} \mid a_{36} \mid a_{41} \mid a_{45} \mid a_{52} \mid a_{61} \mid a_{66}}.$$

Такой способ кодирования удобен, когда матрица заполняется не в строгой последовательности (например, несколько строк одновременно).

Модель OM_1 (однородная по методу параллельных переменных ξ).

Для случая, когда объект содержит только *Y*-компоненты, описываемые соотношениями $\eta=K(\xi)$, уравнения консулируемой проблемы строятся на основе уравнений сечений (6.34), в которые вводятся компонентные соотношения, и уравнений контуров (6.35):

$$\Pi K(\xi) = 0;$$
$$P\xi = 0,$$

или

$$\boxed{\begin{array}{c}\Pi \frac{\partial K}{\partial \xi} \\ \hline P\end{array}} [\xi] = \boxed{\begin{array}{c} b \\ \hline \end{array}} f, \qquad (6.48)$$

где f — вектор задающих источников.

Размерность получаемых уравнений равна $[l] \times [l]$.

Модель OM_2 (однородная по методу сечений). Уравнения консультируемой проблемы строятся только на основании уравнений сечений (6.34), в которые вводятся компонентные соотношения

$$\eta = K(\xi) = K(\xi_T,\ \xi_N) = K(\xi_T,\ -\rho\xi_T) = K^*(\xi_T),$$

предварительно преобразованные на основе соотношения (6.37).

Уравнения имеют вид:

$$\left[\Pi \frac{\partial K^*}{\partial \xi_T}\right] \xi_T = bf,$$

или

$$\left[\Pi \frac{\partial K}{\partial \xi} \Pi^t\right] \xi_T = bf, \tag{6.49}$$

если используются исходные компонентные уравнения и зависимость

$$\xi = [1,\ -\rho]^t \xi_T = \Pi^t \xi_T,$$

вытекающая из соотношения (6.37).

При выборе в консультируемой проблеме полюсного дерева, ветвями которого являются узловые величины v, уравнения (6.49) приводятся к виду (модель OM_{20}):

$$\left[A \frac{\partial K}{\partial \xi} A^t\right] v = bf, \tag{6.50}$$

где A- матрица инциденции. Размер уравнений (6.49)и (6.50) равен

$$[v - 1] \times [v - 1].$$

Модель OM_3 (однородная по методу последовательных переменных η). Для случая, когда консультируемая проблема содержит только Z-компоненты, описываемые соотношениями $\xi = K(\eta)$, уравнения консультируемой проблемы строятся на основе уравнений контуров (6.35), в которые вводятся компонентные соотношения, и уравнений сечений (6.34);

$$P K(\eta) = 0;$$
$$\Pi\eta = 0$$

или

$$\begin{vmatrix} P \frac{\partial K}{\partial \eta} \\ \hline \Pi \end{vmatrix} [\eta] = \begin{vmatrix} b \\ \hline \end{vmatrix} f. \tag{6.51}$$

Размер уравнений $[l] \times [l]$.

Модель OM_4 (однородная по методу контуров). В этом случае уравнения консультируемой проблемы строятся только на основании

уравнений контуров (6.35), в которые вводятся компонентные соотношения

$$\xi = K(\eta) = K(\eta_T, \eta_N) = K(-\pi\eta_N, \eta_N) = K^*(\eta_N),$$

предварительно преобразованные на основании соотношения (6.36), Уравнения имеют вид

$$\left[\mathrm{P}\frac{\partial K^*}{\partial \eta_N}\right]\eta_N = bf,$$

или

$$\left[\mathrm{P}\frac{\partial K}{\partial \eta}\mathrm{P}^t\right]\eta_N = bf, \qquad (6.52)$$

если учесть зависимость

$$\eta = [-\pi,\ 1]^t\eta_N = \mathrm{P}^t\eta_N.$$

При выборе в графе объекта независимой канонической системы контуров, образованной ячейками графа, уравнения (6.52) приводятся к виду (модель OM_{40}):

$$\left[\mathrm{P}'\frac{\partial K}{\partial \eta'}\mathrm{P}'^t\right]\eta' = bf, \qquad (6.53)$$

где P' — матрица канонической системы контуров; η' — вектор контурных величин. Размерность уравнений (6.52)и (6.53) равна

$$[l - v + 1] \times [l - v + 1].$$

Модель $\mathrm{ГМ}_1$ (гибридная или таблично-узловая). Предыдущие однородные модели не удобны для консультируемых проблем, содержащих идеальные компоненты (например, источники, короткозамкнутые ребра и др.). В общем случае, когда в структуре консультируемой проблемы имеется k Z-компонентов, описываемых уравнениями $\xi_2 = K_2(\eta_2)$, и $(l - k)$ Y-компонентов, описываемых уравнениями $\xi_1 = K_1(\eta_1)$, удобно расширить вектор переменных модели OM_2 и ввести в него η последовательных переменных Z-компонентов, неудобных при описании консультируемой проблемы в терминах параллельных переменных (например, последовательные переменные, используемые для управления зависимыми источниками, и др.). Тогда уравнения консультируемой проблемы строятся преимущественно на основе уравнений сечений (6.34) и дополняются компонентными уравнениями Z-компонентов:

$$[\Pi_1\Pi_2]\begin{bmatrix}\eta_1\\ \eta_2\end{bmatrix} = \Pi_1 K_1(\xi_1) + \Pi_2\eta_2 = 0;$$

$$\begin{bmatrix}\xi_1\\ \xi_2\end{bmatrix} = \begin{bmatrix}D_1\\ D_2\end{bmatrix}^t \xi_T.$$

Реализуется система уравнений размерностью

$$[v - 1 + k] \times [v - 1 + k]$$

вида

$$\left|\begin{array}{c|c} \Pi_1 \frac{\partial K_1}{\partial \xi_1} D_1^t & \Pi_2 \\ \hline -D_2^t & \frac{\partial K_2}{\partial \eta_2} \end{array}\right| \left[\begin{array}{c} \xi_T \\ \eta_2 \end{array}\right] = bf, \tag{6.54}$$

которая автоматически адаптируется к компонентному составу консультируемой проблемы.

Аналогично на базе узловых величин, выступающих в качестве параллельных переменных дерева $\xi_T = v$, можно получить уравнения (модель $ГМ_{10}$):

$$\left|\begin{array}{c|c} A_1 \frac{\partial K_1}{\partial \xi_1} A_1^t & A_2 \\ \hline -A_2^t & \frac{\partial K_2}{\partial \eta_2} \end{array}\right| \left[\begin{array}{c} v \\ \eta_2 \end{array}\right] = bf, \tag{6.55}$$

так как в этом случае $\Pi = A$ и $D^t = A^t$.

Модель FM_2 (гибридная или таблично-контурная). При наличии в консультируемой проблеме (l — k) Y-компонентов и k Z-компонентов с компонентными уравнениями $\eta_1 = K_1(\xi_1)$ и $\xi_2 = K_2(\eta_2)$ соответственно уравнения консультируемой проблемы можно строить преимущественно на основе уравнения контуров (6.35) с введением в вектор неизвестных параллельных переменных Y-компонентов и дополнением уравнений консультируемой проблемы компонентными уравнениями этих Y-компонентов:

$$[\mathrm{P}_1\mathrm{P}_2]\left[\begin{array}{c} \xi_1 \\ \xi_2 \end{array}\right] = \mathrm{P}_1\xi_1 + \mathrm{P}_2 K_2(\eta_2) = 0;$$

$$\left[\begin{array}{c} \eta_1 \\ \eta_2 \end{array}\right] = \left[\begin{array}{c} Q_1 \\ Q_2 \end{array}\right]^t \eta_N.$$

Общая размерность реализуемой системы уравнений вида

$$\left|\begin{array}{c|c} \mathrm{P}_2 \frac{\partial K_2}{\partial \eta_2} Q_2^t & \mathrm{P}_1 \\ \hline -Q_2^t & \frac{\partial K_1}{\partial \xi_1} \end{array}\right| \left[\begin{array}{c} \eta_N \\ \xi_1 \end{array}\right] = bf \tag{6.56}$$

Равна

$$[2l - k - v + 1] \times [2l\text{-}k\text{-}v + 1].$$

Аналогично на базе контурных величин η', когда $\eta' = \eta_N$ и $Q^t = \mathrm{P}^t$ можно построить систему уравнений

$$\begin{array}{|c|c|}\hline \mathrm{P}_2\dfrac{\partial K_2}{\partial \eta_2}\mathrm{P}_2^t & \mathrm{P}_1 \\ \hline -\mathrm{P}_1^t & \dfrac{\partial K_1}{\partial \xi_1} \\ \hline \end{array} \left[\begin{matrix}\eta' \\ \xi_1\end{matrix}\right] = bf. \tag{6.57}$$

Модель ГМ$_3$ (классическая гибридная). В этом случае уравнения консультируемой проблемы размерностью [*l*] × [*l*] строятся на основе уравнений сечений (6.34), в которые вводятся компонентные уравнения *Y*-компонентов $\eta_1 = K_1(\xi_1)$ и уравнений контуров (6.35), в которые вводятся компонентные уравнения Z-компонентов $\xi_2 = K_2(\eta_2)$.

Вектор переменных выражается через ξ_T и η_N

$$[\Pi_1\Pi_2]\left[\begin{matrix}\eta_1 \\ \eta_2\end{matrix}\right] = \Pi_1\eta_1 + \Pi_2\eta_2 = \Pi_1 K_1(\xi_1) + \Pi_2\eta_2 = 0;$$

$$[\mathrm{P}_1\mathrm{P}_2]\left[\begin{matrix}\xi_1 \\ \xi_2\end{matrix}\right] = \mathrm{P}_1\xi_1 + \mathrm{P}_2\xi_2 = \mathrm{P}_1\xi_1 + \mathrm{P}_2 K_2(\eta_2) = 0;$$

$$\left[\begin{matrix}\xi_1 \\ \xi_2\end{matrix}\right] = \left[\begin{matrix}D_1 \\ D_2\end{matrix}\right]^t \xi_T; \quad \left[\begin{matrix}\eta_1 \\ \eta_2\end{matrix}\right] = \left[\begin{matrix}Q_1 \\ Q_2\end{matrix}\right]^t \eta_N.$$

Тогда

$$\begin{array}{|c|c|}\hline \Pi_1\dfrac{\partial K_1}{\partial \xi_1}D_1^t & \Pi_2 Q_2^t \\ \hline \mathrm{P}_1 D_1^t & \mathrm{P}_2\dfrac{\partial K_2}{\partial \eta_2}Q_2^t \\ \hline \end{array} \left[\begin{matrix}\xi_T \\ \eta_N\end{matrix}\right] = bf. \tag{6.58}$$

При выборе в консультируемой проблеме полюсного дерева, ветвями которого являются узловые напряжения *v*, и канонической системы последовательных величин η' контуров, образованных ячейками графа, уравнения (6.58) приводятся к виду (модель ГМ$_{30}$, или каноническая гибридная модель):

$$\begin{array}{|c|c|}\hline A_1\dfrac{\partial K_1}{\partial \xi_1}A_1^t & A_2\mathrm{P}_2'^t \\ \hline \mathrm{P}_1' A_1^t & \mathrm{P}_2'\dfrac{\partial K_2}{\partial \eta_2}\mathrm{P}_2'^t \\ \hline \end{array} \left[\begin{matrix}v \\ \eta'\end{matrix}\right] = bf,$$

так как $\Pi = A$, $D^t = A^t$, $\mathrm{P} = \mathrm{P}'$ и $Q^t = \mathrm{P}'^t$.

Модель ГМ$_4$ (сокращенная гибридная). В отличие от модели ГМ$_3$ вектор переменных уравнений консультируемой проблемы содержит только параллельные переменные *Y*-компонентов дерева ξ_{UT} и последовательные переменные Z-хорд η_{ZN}. При этом уравнения консультируемой проблемы образуются из совокупности только тех Π'

независимых сечений, которые инцидентны Y_T-компонентам, и тех P′ независимых контуров, которые соответствуют Z_N-компонентам:

$$\left|\begin{array}{c|c} \Pi_1' \frac{\partial K_1}{\partial \xi_1} D_1'^t & \Pi_2' Q_2'^t \\ \hline P_1' D_1'^t & P_2' \frac{\partial K_2}{\partial \eta_2} \end{array}\right| \left[\begin{array}{c} \xi_{YT} \\ \eta_{ZN} \end{array}\right] = bf. \tag{6.59}$$

В модели $ГМ_4$ по сравнению с моделью $ГМ_3$ число уравнений в первом приближении уменьшается на число вырожденных контуров в графе консультируемой проблемы, содержащих только *Y*-компоненты или *Y*-компоненты и независимые источники ξ, и число вырожденных сечений, инцидентных только Z-компонентам или Z-компонентам и независимым источникам η.

Модель ПС (переменных состояния). В этом случае формируется система дифференциальных уравнений, описывающих консультируемую проблему, на основе уравнений любой из гибридных моделей (ПГМ, $ПГМ_0$, $ГМ_1$— $ГМ_4$). Для этого в гибридном векторе переменных этих моделей нужно выделить субвектор переменных состояния $x=[\xi_C \eta_L]^t$, соответствующих параллельным переменным эквивалентных емкостных компонентов и последовательным переменным эквивалентных индуктивных компонентов, а остальные переменные объединить в субвектор x_0 переменных безынерционных компонентов.

Тогда, например, уравнения гибридной модели консультируемой проблемы $ГМ_4$ приводятся к виду

$$\left[\begin{array}{cc} pw_{11} + a_1 w_{12} \\ a_2 \end{array}\right] \left[\begin{array}{c} x \\ x_0 \end{array}\right] = \left[\begin{array}{c} b_1 \\ b_2 \end{array}\right] f. \tag{6.60}$$

Исключив x_0 и обратив w_{11}, получаем каноническую форму уравнений переменных состояния:

$$px = \dot{x} = w_{11}^{-1} [(a_1 - w_{12} w_{22}^{-1} a_2) x - (b - w_{12} w_{22}^{-1} b_2) f] = Ax + Bf. \tag{6.61}$$

Разделение и исключение зависимых переменных в уравнении (6.60) можно выполнить по следующей алгоритмической процедуре.

Представим уравнение (6.60) в виде

$$\left[\begin{array}{cc} pw_{11} w_{12} \\ 0 \;\; w_{22} \end{array}\right] \left[\begin{array}{c} x \\ x_0 \end{array}\right] = \left[\begin{array}{c} -a_1 \\ -a_2 \end{array}\right] x + \left[\begin{array}{c} b_1 \\ b_2 \end{array}\right] f,$$

или

$$W^{(0)}z^{(0)} = A^{(0)}x^{(0)} + B^{(0)}f, \quad z = [x, \ x_0]^t, \tag{6.62}$$

и используем алгоритм Гауссова — Жордана для приведения матрицы $W^{(0)}$ к единичной форме.

Если в процессе преобразования матрица $W^{(0)}$ будет содержать нулевую j-строку среди строк подматрицы $w^{(0)}{}_{22}$, то составляющие субвектора x окажутся линейно независимыми. Тогда, по крайней мере, одну переменную состояния нужно исключить. В этом случае j-строка принимает вид

$$[0]\, x_0 = [a_{j1}, \ a_{j2}, \ \ldots, \ a_{jn}]\, x + [b_{j1}, \ b_{j2}, \ \ldots, \ b_{jm}]\, f.$$

Определяя первый ненулевой элемент a_{jk}, исключаем из уравнений (6.62) переменную

$$x_k = -\frac{1}{a_{jk}}(a_{j1}x_1 + \cdots + b_{j1}f_1 \ldots) \tag{6.63}$$

и ее производную

$$\dot{x}_k = -\frac{1}{a_{jk}}(a_{j1}\dot{x}_1 + \cdots + b_{j1}\dot{f}_1 + \cdots). \tag{6.64}$$

Исключение переменной x_k достигается процедурой гауссова исключения элементов в k-м столбце матрицы A (сведения их к нулевым значениям) с последующим вычеркиванием этого столбца. Исключение производной x_k из правой части выражения (6.62) достигается вычеркиванием k-го столбца матрицы $W^{(0)}$ и подстановкой выражения (6.64) в k-ю строку уравнений (6.62), при этом

оставшиеся члены

$$\frac{a_{js}}{a_{ik}},$$

$s = k + l, \ \ldots, \ n$-й строки матрицы A вносятся в k-ю строку подматрицы $w^{(0)}{}_{11}$, а члены

$$\frac{b_{jl}}{a_{jk}},$$

l=1.....m j-й строки матрицы B вносятся в k-ю строку дополнительной матрицы B_1, соответствующей вектору $\dot{f}$. В результате уравнения (6.62) принимают вид

$$W^{(1)}z^{(1)} = A^{(1)}x^{(1)} + B^{(1)}f + B_1^{(1)}\dot{f} \tag{6.65}$$

и процесс исключения рекурсивно повторяется. Отметим, что субвектор переменных $x^{(1)}$ содержит на одну компоненту меньше, чем субвектор $x^{(0)}$.

6.8. Логические процедуры формирования уровней моделей консультируемых проблем

Логические процедуры формирования уровней моделей консультируемых проблем по уравнениям (6.48)—(6.59) может выполняться без матричных операций путем только логических процедур. Классическим примером является алгоритм логического заполнения матрицы модели $ОМ_{20}$, или матрицы проводимости. Если для упрощения записи в уравнении (6.48) обозначить вектор частных производных компонентных уравнений ветвей через $Y_B = \partial K/\partial \xi$, то из формулы (6.48) видно, что диагональный элемент матрицы уравнений

$$y_{kk} = [\pi_{k_1}, \pi_{k_2}, \ldots, \pi_{kl}]\,[Y_в]\,[\pi_{k_1}, \pi_{k_2}, \ldots, \pi_{kl}]^t = \sum_{i=1}^{l} \pi_{kl}^2 y_{вi}, \tag{6.66}$$

а недиагональный элемент, расположенный на пересечении k-й строки и s-го столбца, определяется как

$$y_{ks} = [\pi_{k_1}, \pi_{k_2}, \ldots, \pi_{kl}]\,[Y_в]\,[\pi_{k_1}, \pi_{k_2}, \ldots, \pi_{kl}]^t = \sum_{i=1}^{l} \pi_{kl}\pi_{sl} y_{вi}. \tag{6.67}$$

Учитывая, что $\pi_{ki}=\pm 1$, находим, что диагональный элемент y_{kk} равен сумме проводимостей ребер, пересекаемых k-м сечением, а элемент y_{ks} равен сумме проводимостей ребер, общих для k-го и s-го сечений. Общие проводимости суммируются со знаком «плюс», ели сечения по отношению к данному ребру направлены одинаково, и со знаком «минус», если сечения направлены противоположно. Аналогично k-я компонента вектора b в правой части выражения (6.48) указывает на совокупность принадлежащих k-му сечению ребер, определяющих задающие источники, с учетом их направления относительно сечения. Таким образом, элементы матриц $Y = [\Pi Y_в \Pi^t]$ и b уравнения (6.48) можно записать непосредственно из графа консультируемой проблемы (эквивалентной схемы замещения) без каких-либо операций над матрицами и векторами, например, поочередным рассмотрением независимых сечений и их комбинаций.

Сформулированные выше правила логического заполнения элементов матриц уравнений модели консультируемой проблемы можно обобщить и на другие типы моделей. Например, для таблично-узловой модели $ГМ_{10}$, определяемой выражением (6.55), матрица коэффициентов уравнений имеет структуру

$$\left|\begin{array}{c|c} Y & N \\ \hline M & Z \end{array}\right|, \tag{6.68}$$

в которой ее верхняя часть соответствует независимым сечениям графа консультируемой проблемы с учетом инцидентности вводимых последовательных переменных η_2 Z-компонентов этим сечением, при этом субматрица *Y* формируется по аналогии с матрицей проводимости. Нижняя часть матрицы уравнений соответствует компонентным уравнениям Z-компонент.

Пример 9. Расссмотрим последовательность логических процедур получения модели $ГМ_{10}$ на примере консультируемого объекта в виде электрической схемы, изображенной на рис. 6.18, *а.*

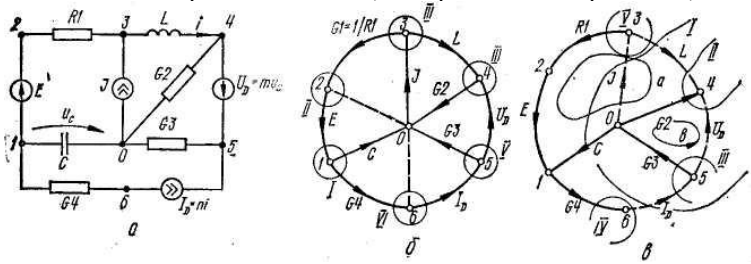


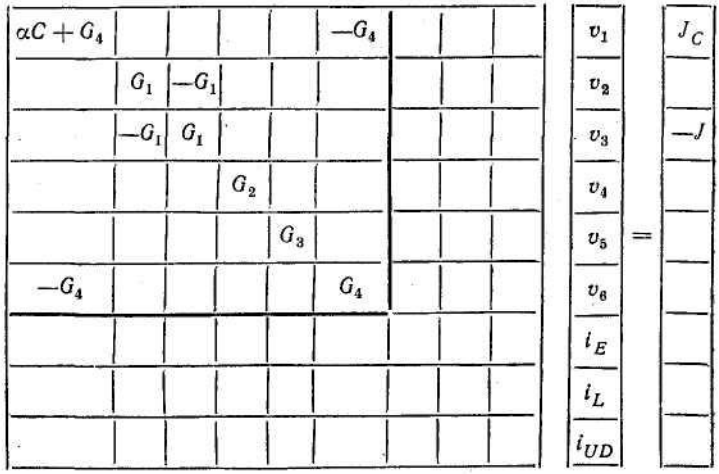
Рис. 6.18. Электрическая схема *(а),* ее граф для таблично-узловой модели $ГМ_{10}$ *(б)* и сокращенной гибридной модели $ГМ_4$ *(в)*

В схему входят два зависимых источника: тока I_D, управляемого током индуктивности i_L, и напряжения U_D, управляемого напряжением емкости u_c. С учетом операций линеаризации и алгебраизации компонентные уравнения схемы можно записать в виде:

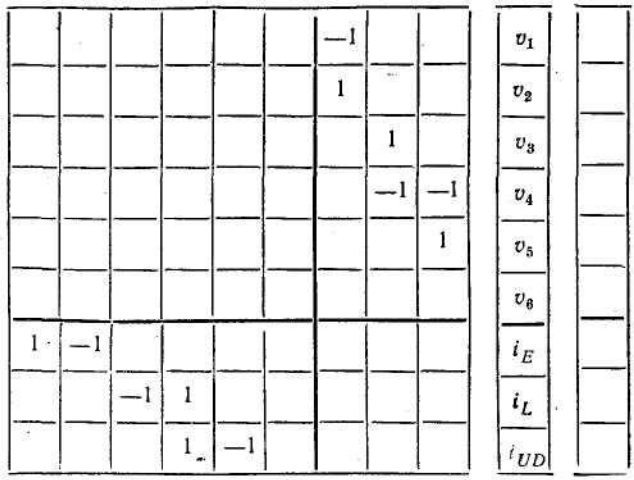
$$i_C = \alpha C u_C + J_C;$$
$$u_L = \alpha L i_L + E_L;$$
$$i_{R1} = G_1 u_{R1}, \text{ или } u_{R1} = R_1 i_{R1};$$
$$i_{G2} = G_2 u_{G2};$$
$$i_{G3} = G_3 u_{G3};$$
$$i_{G4} = G_4 u_{G4};$$
$$I_D = ni = ni_L;$$
$$U_D = mu = mu_C.$$

Кроме того, в схеме имеется два независимых источника: напряжения *E* и тока *J*. Соответствующий схеме граф с выделенной совокупностью независимых сечений для узловых напряжений показан на рис. 6.18, *б.* Таких сечений шесть. Кроме того, в схеме имеется три «неудобных» Z-компонента *(E, L, U_D)*, токи которых следует ввести в вектор переменных. Таким образом, искомая система уравнений схемы будет иметь 9-й порядок.

1. Вначале по описанной выше методике формируем субматрицу проводимости, не обращая внимания на Z-компоненты:



2. Определяем инцидентность введенных токов i_z (i_E, i_L, i_{UD},) независимым сечением, находим соответствующую подматрицу инциденции A_2 (см. уравнение (6.55)), по ней $-A^t_2$, которые вносим в подматрицы N и M структуры модели (6.68):



3. Учитываем в матрице модели схемы зависимые источники типа $I_i = = nI_j$ и $U_i = mU_j$, а также собственные сопротивления Z-компонентов и источники, входящие в добавляемые компонентные уравнения. Параметр n проставляем на пересечении строк матрицы,

определяемых вершинами включения ребра с током I_i зависимого источника, и столбца, соответствующего ребру с управляющим током I_j. Аналогично, параметр m вписываем в элемент матрицы, расположенной на пересечении строки, определяемой ребром U_i (среди добавочных компонентных уравнений) и столбцом, соответствующим вершинам включения управляющего ребра U_i. На этом этапе получаем

									v_1	
									v_2	
									v_3	
									v_4	
									v_5	=
									v_6	
									i_E	E
									i_L	$-E_L$
									i_{UD}	

							$-n$			
							n			
							αL			
$-m$										

Следует отметить, что в модели $ГМ_{10}$ легко учитываются параметры зависимых источников всех известных четырех типов. Окончательно получаем уравнения таблично-узловой модели схемы, изображенной на рис. 6.18 *а,* в виде

$\alpha C + G_4$					$-G_4$	-1			v_1	J_C
	G_1	$-G_1$				1			v_2	
	$-G_1$	G_1					1	-1	v_3	$-J$
			G_2				-1	1	v_4	
				G_3			$-n$		v_5	
$-G_4$					G_4		n		v_6	
1	-1								i_E	$-E$
		-1	1				αL		i_L	$-E_L$
$-m$			1	-1					i_{UD}	

Для сравнения сформируем сокращенную гибридную модель $ГМ_4$ для схемы рис. 6.18, *а,* чтобы проиллюстрировать однотипность используемых логических процедур заполнения матриц различных типов моделей. Соответствующий этому случаю граф схемы с выделенной совокупностью независимых сечений и контуров показан на рис. 6.18, *в*. При этом *Y*-компонентами являются компоненты *C, G_2, G_3, G_4, I_D* и *J*, а *Z*-компонентами — *E, R_1, L, U_D*. Как видно из рис. 6.18, *в,* в дерево графа вошли ребра, соответствующие *E, C, G2, G3, G4, R1*, а к хордам отнесены— I_D, *J, L, U_D* (штриховые линии на рис. 6.18, *в).* В модели $ГМ_4$ независимые сечения определяются *Y*-компонентами дерева (*C, G2, G3, G4)* и Z-хордами (*L, U_D),* поэтому матрица искомой системы уравнений будет 6-го порядка.

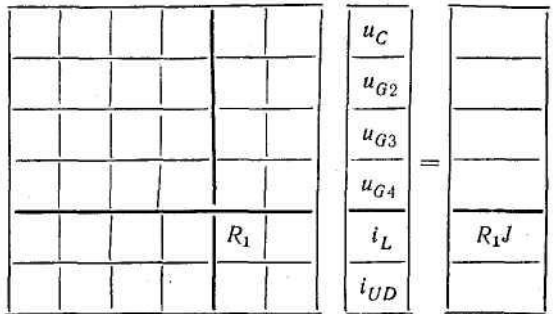
1. Вначале заполняются собственные проводимости (сопротивления) невырожденных сечений (контуров), определяемых Y_T и Z_N, с учетом инциденции их токов и напряжений, вошедших в вектор переменных уравнений модели, а также независимых источников. Рассматривая сечения *I—IV* графа рис. 6.18, *в* и контуры *а* и *в* для *L* и U_D, и принимая во внимание компонентные уравнения, записываем

αC				-1		u_C	$-J-J_C$
	G_2			1	1	u_{G2}	
		G_3			1	u_{G3}	
			G_4			u_{G4}	
1	-1			αL		i_L	$E+E_L$
	-1	-1				i_{uD}	

2. Далее заносятся в элементы матрицы параметры избыточных ребер Y_N и Z_T. Например, при Z_T их параметры проставляются в строках матрицы, соответствующих тем Z_N, с которыми Z_T образуют вырожденные сечения, и в столбцах, соответствующих инцидентным невырожденным контурам. При Y_N их параметры вносятся в элементы матрицы, расположенные на пересечении строк, соответствующих тем Y_T (определяющих невырожденные сечения), с которыми Y_N образуют вырожденные контуры, и столбцов, соответствующих инцидентным невырожденным сечениям. Знаки параметров определяются согласованностью направлений рассматриваемых ребер, вырожденных и невырожденных сечений и контуров.

Для схемы, изображенной на рис. 6.18, *а,* избыточным ребром является Z-ребро дерева R_1, входящее в вырожденное сечение *V (*рис.

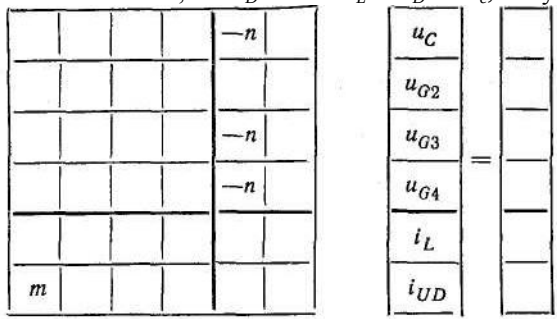
6.18, *в*), отражающее соотношение $i_{R_1} + i_L — J = 0$. В свою очередь, ребро R_1 входит в невырожденный контур для L, поэтому



3. Учитываются параметры зависимых источников по процедуре, аналогичной предыдущей. Так, параметры зависимых источников $I_{Di}=gU_j$ и $I_{Di}= nI_j$ вносятся в элементы матрицы на пересечении строк, соответствующих невырожденным сечениям, инцидентным ребру источника I_{Di}, и столбцов, соответствующих сечениям, инцидентным управляющему ребру U_j, или контуров, инцидентных управляющему ребру I_j.

Параметры зависимых источников напряжения типа $U_{Di}= mU_j$ и $U_{Di} = rI_j$ вносятся в элементы матрицы на пересечении строк, соответствующих невырожденным контурам, инцидентным ребру источника U_{Di}, и столбцов, соответствующих невырожденным сечениям (или контурам), инцидентных управляющему ребру j.

Для графа рис. 6.18, *в* ребро I_D инцидентно невырожденным сечениям *I*, *III* и *IV*, при этом направления этих сечений и ребра противоположны. Учитывая, что $I_D = ni = ni_L$ и $U_D= mu_c$, получаем:



Суммируя вклады отдельных ребер графа схемы, получаем ее уравнения в сокращенном гибридном базисе $ГМ_4$:

$$\begin{array}{|cccc|c|}
\hline
\alpha C & & & & -1-n & \\
\hline
 & G_2 & & & 1 & 1 \\
\hline
 & & G_3 & & -n & 1 \\
\hline
 & & & G_4 & -n & \\
\hline
1 & -1 & & & \alpha L + R_1 & \\
\hline
m & -1 & -1 & & & \\
\hline
\end{array}
\begin{array}{|c|}
\hline
u_C \\
\hline
u_{G2} \\
\hline
u_{G3} \\
\hline
u_{G4} \\
\hline
i_L \\
\hline
i_{UD} \\
\hline
\end{array}
=
\begin{array}{|c|}
\hline
-J-J_C \\
\hline
 \\
\hline
 \\
\hline
 \\
\hline
E + R_1 J + E_C \\
\hline
 \\
\hline
\end{array}.$$

Отметим, что для данного примера разреженность модели $ГМ_{10}$, равная отношению числа НЭ к общему числу элементов матрицы, выше, чем модели $ГМ_4$ (32% и 44% соответственно). Это следует учитывать при использовании методов кодирования и обработки разреженных матриц. Матрицы заполнением, превышающим 33%, рекомендуется обрабатывать как полностью заполненные.

Пример 10. Рассмотрим процедуру автоматического формирования уравнений переменных состояния схемы, содержащей зависимый источник тока i_D, управляемый током короткозамкнутой ветви i (рис. 6.19, *а*).

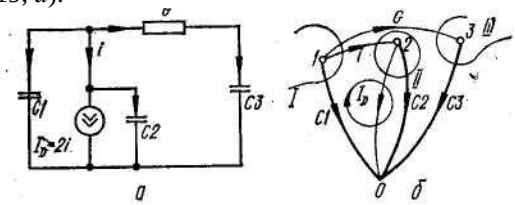
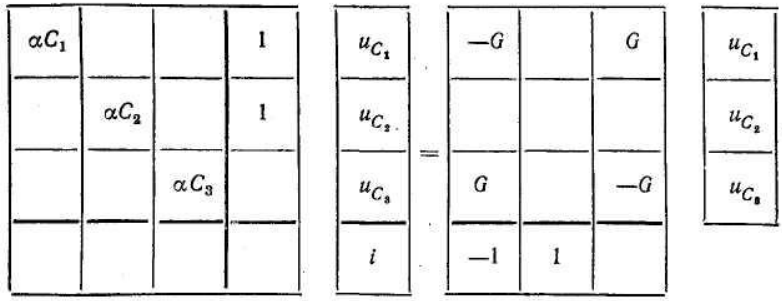


Рис. 6.19. Электрическая схема (*а*) и ее граф (*б*)

Выберем на графе схемы дерево (рис. 6.19, *б)*. Порядок модели $ГМ_4$ будет определяться тремя невырожденными сечениями для *С1, С2, С3* и одним контуром для короткозамкнутой ветви. По аналогии с предыдущим получаем

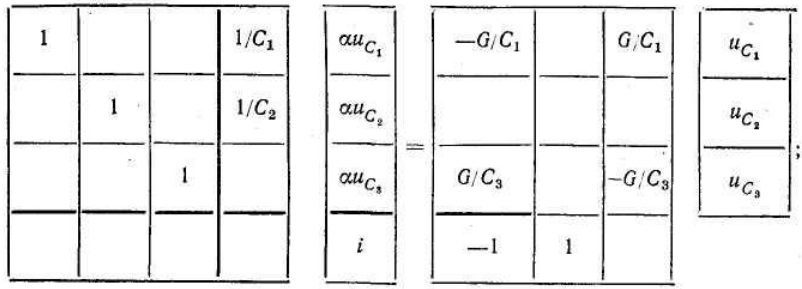
$$\begin{array}{|cccc|}
\hline
\alpha C_1 + G & & -G & 1 \\
\hline
 & \alpha C_2 & & 2-1 \\
\hline
-G & & \alpha C_3 + G & \\
\hline
1 & -1 & & \\
\hline
\end{array}
\begin{array}{|c|}
\hline
u_{C_1} \\
\hline
u_{C_2} \\
\hline
u_{C_3} \\
\hline
i \\
\hline
\end{array}
=
\begin{array}{|c|}
\hline
 \\
\hline
 \\
\hline
 \\
\hline
 \\
\hline
\end{array}.$$

В соответствии с выражением (6.62) преобразуем полученные уравнения к виду

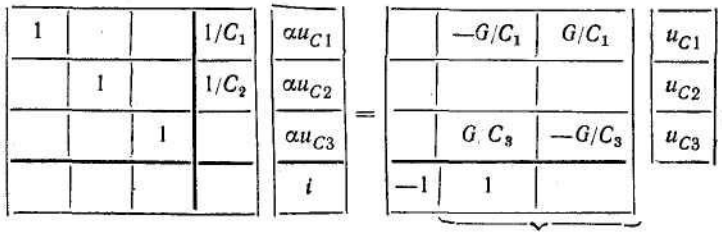


В подматрице w_{22} существует нулевая строка, поэтому в соответствии с выражениями (6.63) и (6.64) выполняются следующие преобразования:

- выделяются элементы единичной матрицы



- проводится гауссово исключение элементов первого столбца матрицы A для исключения u_{C1}



- исключается производная u_{C1} перенесением выделенных элементов четвертой строки матрицы A в первую строку матрицы w_{11}

$$\begin{array}{|c|c|c|}\hline 1 & & 1/C_1 \\ \hline 1 & & 1/C_2 \\ \hline & 1 & \\ \hline \end{array} \quad \begin{array}{|c|}\hline \alpha u_{C2} \\ \hline \alpha u_{C3} \\ \hline i \\ \hline \end{array} = \begin{array}{|c|c|}\hline -G/C_1 & G/C_1 \\ \hline & \\ \hline G/C_3 & -G/C_3 \\ \hline \end{array} \quad \begin{array}{|c|}\hline u_{C2} \\ \hline u_{C3} \\ \hline \end{array}; \qquad (6.69)$$

- повторно выделяются элементы единичной матрицы

$$\begin{array}{|c|c|c|}\hline 1-C_2/C_1 & & \\ \hline 1 & & 1/C_2 \\ \hline & 1 & \\ \hline \end{array} \quad \begin{array}{|c|}\hline \alpha u_{C2} \\ \hline \alpha u_{C3} \\ \hline i \\ \hline \end{array} = \begin{array}{|c|c|}\hline -G/C_1 & G/C_1 \\ \hline & \\ \hline G/C_3 & -G/C_3 \\ \hline \end{array} \quad \begin{array}{|c|}\hline u_{C2} \\ \hline u_{C3} \\ \hline \end{array}.$$

Из первой и третьей строк полученной системы уравнений формируются уравнения переменных состояния

$$\begin{bmatrix} \dot{u}_{C2} \\ \dot{u}_{C3} \end{bmatrix} = \begin{bmatrix} -G/(C_1-C_2) & G/(C_1-C_2) \\ G/C_3 & -G/C_3 \end{bmatrix} \begin{bmatrix} u_{C2} \\ u_{C3} \end{bmatrix}.$$

Если $C_1=C_2$, то, как следует из выражения (6.69), в процессе гауссова исключения будет обнаружена новая нулевая строка в матрице $W^{(1)}$, в результате потребуется новая итерация исключения еще одной зависимой переменной, так как в данном случае $u_{C2} = u_{C3}$.

6.9. Математические модели консультируемых проблем, используемые на макроуровне

Компонентные и топологические уравнения. Для одной и той же консультируемой проблемы (элемента) на микро- и макроуровнях используют разные математические модели. На ***микроуровне*** ММ должна отражать внутренние по отношению к консультируемой проблеме ***процессы, протекающие в сплошных средах***. На ***макроуровне*** ММ той же консультируемой проблемы служит для ***отражения только тех ее свойств, которые характеризуют взаимодействие этой консультируемой проблемы с другими элементами в составе исследуемой консультируемой проблемы***.

Математические модели элементов на макроуровне получают одним из способов, рассмотренных в п.6.4. Математические модели консультируемых проблем (ММКП) формируют из математических моделей элементов (ММЭ) с помощью методов, излагаемых ниже.

Уравнения, входящие в ММКП, будем называть ***компонентными***. Наряду с компонентными уравнениями в ММКП обязательно входят ***уравнения,*** отражающие ***способ связи элементов между собой*** в

составе системы и называемые ***топологическими***. Топологические уравнения могут выражать законы сохранения, условия неразрывности, равновесия и т. п.

В используемых в САК методах формирования ММКП будем моделируемую консультируемую проблему представлять в виде совокупности физически однородных подпроблем. Каждая подпроблема описывает процессы определенной физической природы, например механические, электрические, тепловые, гидравлические. Как правило, для описания состояния одной подпроблемы достаточно применять ***фазовые переменные*** двух типов — ***потенциала и потока***. В первых столбцах табл. 6.4 конкретизированы типы фазовых переменных применительно к ряду встречающихся подпроблем.

Таблица. 6.4

Подпроблема	Физическая переменная		Параметры простых элементов		
	типа потенциала U	типа потока I	C	L	R
Механическая	Скорость	Сила	Масса	Гибкость (обратная величина жесткость	Механическое сопротивление
Механическая вращательная	Угловая скорость	Вращательный момент	Момент инерции	Вращательная гибкость	Вращательное сопротивление
Электрическая	Электрическое напряжение	Электрический ток	Электрическая емкость	Электрическая индуктивность	Электрическое сопротивление
Тепловая	Температура	Тепловой поток	Теплоемкость	-	Тепловое сопротивление
Гидравлическая и пневматическая	Давление	Расход	Гидравлическая емкость	Гидравлическая индуктивность	Гидравлическое сопротивление

Особенностью топологических уравнений является то, что каждое из них связывает однотипные фазовые переменные, относящиеся к разным элементам системы. Примером могут служить уравнения законов Кирхгофа, записываемые относительно либо токов, либо напряжений ветвей. Для компонентных уравнений характерно то, что они связывают разнотипные фазовые переменные, относящиеся к одному элементу. Так, уравнение закона Ома связывает ток и напряжение резистора.

Формы представления моделей. Элементы подпроблем могут быть простыми и сложными. Элемент называют ***простым,*** если соответствующая ему ММЭ может быть представлена в виде ***одного линейного уравнения***, связывающего переменную типа потенциала *U* и переменную типа потока *I*, характеризующие состояние данного элемента.

В физически однородных подпроблемах различают три типа простых элементов. Это элементы ***емкостного, индуктивного и резистивного типов.*** Соответствующие им ММЭ имеют вид

$$CdU/dt = I,\ LdI/dt = U,\ U = RI, \qquad (6.70)$$

где *C, L, R* — параметры элементов, физический смысл которых поясняется в табл.6.4.

Элементы подпроблем в зависимости от числа однотипных фазовых переменных, входящих в ММЭ, делят на двухполюсники и многополюсники.

Двухполюсник характеризуется парой переменных типа *U* и *I*, определяется так же, как простой элемент, если снять условие линейности уравнения.

Многополюсник можно представить как совокупность взаимосвязанных двухполюсников.

Для представления математических моделей на макроуровне применяют несколько форм.

Инвариантная форма — представление модели в виде системы уравнений, записанной на общепринятом математическом языке, безотносительно к методу численного решения.

Применительно к системам обыкновенных дифференциальных уравнений различают две инвариантные формы — ***нормальную и общую,*** определяемые тем, в каком виде — явном или неявном относительно вектора производных — представлена система.

Ряд форм модели получается при преобразовании ее уравнений на основе формул и требований выбранного численного метода решения. Так, численное решение дифференциальных уравнений как в

частных производных, так и обыкновенных требует их предварительного преобразования—***дискретизации и алгебраизации***.

Дискретизация заключается в замене непрерывных независимых переменных (времени и пространственных координат) дискретным множеством их значений.

Алгебраизованная форма — результат представления дифференциальных уравнений в полученных после дискретизации точках в алгебраизованном виде с помощью формул численного интегрирования.

Ряд численных методов решения основан на линеаризации исходных уравнений,

Линеаризованная форма модели — представление ее уравнений в линейном виде.

Алгебраизация и линеаризация могут осуществляться по отношению ко всем или только избранным переменным, уравнениям или их частям, что увеличивает разнообразие возможных форм представления моделей.

Формы представления моделей определяются также используемыми языковыми средствами. Наряду с традиционным математическим языком применяют алгоритмические языки, а также те или иные графические изображения, облегчающие пользователю восприятие модели и приводящие к представлению модели в той или иной ***схемной форме***, например представление моделей в виде эквивалентных схем, графов, к таким формам относится также представление разностных уравнений с помощью шаблонов (см. п.6.5).

Рассмотрим особенности представления моделей в виде ***эквивалентных схем***.

В разных проблемных областях применяют специфические системы обозначений элементов на эквивалентных схемах. Будем использовать в дальнейшем единую систему обозначений для элементов всех подпроблем, обычно применяемую при изображении электрических эквивалентных схем. При этом элементы представляют собой ***двухполюсники***, которые могут быть ***пяти различных видов***, их условные обозначения приведены на рис. 6.18, *а.*

Получение эквивалентных схем — обычная для консультантов-схемотехников операция, выполняемая при анализе функционирования консультируемой проблемы, представленной радиоэлектронными устройствами. Переход от принципиальной электрической схемы к эквивалентной заключается в замене обозначений электронных приборов обозначениями двухполюсников (рис. 6.18, *а*) и добавлении ветвей, отображающих учитываемые паразитные параметры.

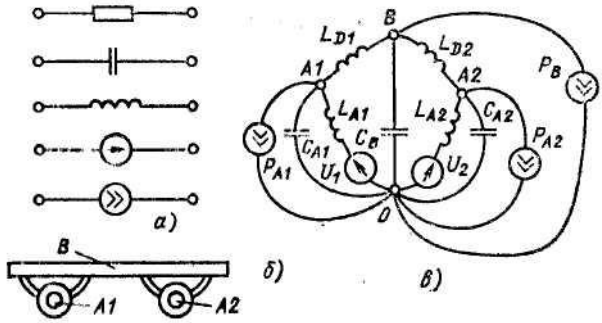


Рис. 6.18. Условные обозначения дпухполюсиых элементов

Не вызывает затруднений и составление на основе электрогидравлических и электротепловых аналогий эквивалентных схем, отражающих гидравлические, пневматические и тепловые процессы в консультируемых проблемах.

Составление эквивалентных схем для механических систем начинается с выбора системы координат, начало *О* которой должно быть связано с инерциальной системой отсчета. Далее формируются *n* эквивалентных схем, где *n* — число степеней свободы. *В общем случае возможны* ***три эквивалентные схемы****, соответствующие* ***поступательным движениям*** *вдоль координатных осей, и* ***три эквивалентные схемы****, соответствующие* ***вращательным движениям*** *вокруг осей, параллельных координатным осям.* Рассмотрим правила составления эквивалентных схем на примере одной из эквивалентных схем для поступательного движения:

1) для каждого тела A_i с учитываемой массой C_i в эквивалентной схеме выделяется узел *i* и между узлом *i* и узлом *О* включается двухполюсник массы C_i ;

2) трение между контактируемыми телами A_p и A_q отражается двухполюсником механического сопротивления, включаемым между узлами *p* и *q;*

3) пружина, соединяющая тела A_p и A_q, а также другие упругие взаимодействия контактируемых тел A_p и A_q отражаются двухполюсником гибкости (жесткости), включаемым между узлами *p* и *q*.

В качестве примера на рис. 6.18, *в* приведена эквивалентная схема, моделирующая вертикальные скорости и усилия, возникающие в элементах движущегося транспортного устройства, условно изображенного на рис. 6.18, *б* в виде платформы *В* и колес *А1* и *А2*.

Здесь учитываются массы платформы C_B и колес C_A, жесткости колес L_A и рессор L_D, а также веса P_B, P_{A1}, P_{A2} платформы и колес. Внешние воздействия отражены источниками скорости U.

Часто на эквивалентных схемах рядом с обозначением нелинейного элемента указан его тип или записано его компонентное уравнение.

Для отражения взаимосвязей подпроблем различной физической природы, из которых состоит моделируемая консультируемая проблема, в эквивалентные схемы подпроблем вводят специальные ***преобразовательные элементы***. Различают ***три вида связей*** подпроблем. ***Трансформаторная и гираторная связи*** выражают соотношения между фазовыми переменными двух подпроблем, этим типам связей соответствуют преобразовательные элементы, представляемые парами источников ***тока или напряжения***. ***Третий вид связи выражает влияние фазовых переменных одной подпроблемы на параметры элементов другой и задается в виде зависимостей C, L или R от фазовых переменных.***

Варианты эквивалентных схем трансформаторной *(а)* и гираторной *(б, в)* связей даны на рис. 6.19, где запись вида *A(B)* означает, что фазовая переменная *A* является функцией фазовой переменной *B,*

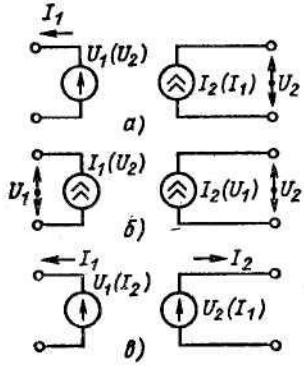


Рис. 6.19. Эквивалентные схемы трансформаторной *(а)* и гираторных (*б, в)* связей

Так, в гидравлическом приводе связь механической и гидравлической подсистем является гираторной и соответствует рис. 6.19, *б*, если для источника объемного расхода в гидравлической подсистеме использовать выражение $g = SV$, а для источника силы в механической подсистеме — выражение $F = SP$, где V — скорость

перемещения поршня; S — площадь поршня; P — давление жидкости в цилиндре.

Примеры математических моделей элементов электронных схем. Для конденсаторов, катушек индуктивности и резисторов чаще всего применяют простые модели (6.70). Примерами сложных элементов являются транзисторы, диоды, трансформаторы. На рис. 6.20, *а* представлена эквивалентная схема биполярного транзистора, используемая во многих программах анализа электронных схем.

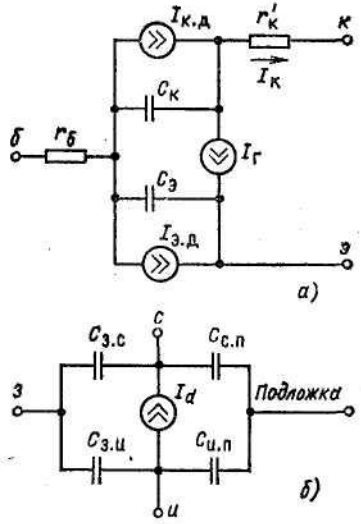


Рис. 6.20. Эквивалентные схемы биполярного *(а)* и МДП (*б*) транзисторов

Этой схеме соответствуют следующие компонентные уравнения:

$$dU_{C_э} / dt = (I_э - I_г - I_{э.д})/C_э; \quad dU_{C_к} / dt = (I_к + I_г - I_{к.д})/C_к,$$

где $C_э$, $C_к$— емкости и $I_г$, $I_{э.д}$, $I_{к.д}$— токи переходов, являющиеся заданными функциями напряжений $U_{сэ}$ и $U_{ск}$ на емкостях, а для $r_б$ и $r'_к$ используются уравнения закона Ома.

На рис. 6.20, *б* представлена эквивалентная схема МДП-транзистора, в которой I_d является функцией потенциалов на электродах прибора, а емкости между затвором и стоком $C_{з.с}$, затвором и истоком $C_{з.н}$, истоком и подложкой $C_{и.п}$, стоком и подложкой $C_{с.п}$ считаются либо постоянными, либо зависящими от потенциалов электродов.

Примеры математических моделей элементов консультируемых проблем неэлектрической природы. Простыми элементами механических поступательных систем являются элементы массы и гибкости (жесткости). Математическая модель массы выражает закон Ньютона

$$F = mdU/dt,$$

где F — сила; m — масса; U — скорость.

Математическая модель упругого стержня получается из закона Гука

$$\sigma = E\Delta l/l, \qquad (6.71)$$

где σ и Δl —напряжение и изменение длины стержня в продольном направлении; E — модуль упругости; l — длина стержня.

Так как $\sigma = F/S$, где S — площадь поперечного сечения стержня, то после дифференцирования (6.71) по времени имеем

$$dF/dt = U/L_M,$$

где $L_M = l/(SE)$ — гибкость стержня (напомним, что обратная величина гибкости — жесткость).

Сочетания элементов массы, гибкости и механического трения образуют разнообразные сложные элементы многих механических систем, например манипуляторов.

В качестве примера на рис. 6.21 приведена эквивалентная схема плоского сложного элемента «шарнирная связь двух твердых тел», где $C1$, $C2$ — массы, а $C3$, $C4$ — моменты инерции соединенных тел.

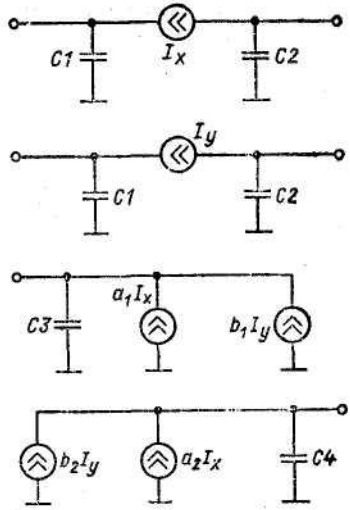


Рис. 6.21. Эквивалентная схема «шарнирная связь двух твердых тел»

Математическая модель представляет собой систему уравнений, отражающих геометрические соотношения, действующие в системе шарнирно связанных тел:

$$U_{x1} - a_1\omega_1 - U_{x2} + a_2\omega_2 = 0;\ U_{y1} + b_1\omega_1 - U_{y2} - b_2\omega_2 = 0, \quad (6.72)$$

где U_x и U_y — проекции вектора скорости центра масс на оси координат x и y (индексы 1 и 2 относят соответствующую величину к первому или второму телу); ω —угловая скорость относительно центра масс тела; a_i и b_i — коэффициенты, зависящие от угла поворота i-го тела в выбранной системе координат. Реакции в шарнирах I_x и I_y определяются в результате решения системы уравнений, состоящей из компонентных уравнений (6.72), компонентных уравнений других элементов и топологических уравнений. Реакции I_x и I_y фигурируют в топологических уравнениях, составляемых для внешних узлов эквивалентной схемы. В случае связи тел с помощью упругой тяги в эквивалентной схеме появляется элемент гибкости, при учете трений — элемент механического сопротивления.

Процессы теплопереноса в твердых телах отображаются элементами ***теплопроводности и теплоемкости***. Математическая модель теплового сопротивления вытекает из уравнения Фурье

$$\mathbf{J} = -\lambda \mathrm{grad} T,$$

где **J** — плотность теплового потока; K — коэффициент теплопроводности; T — температура.

Рассматривая распределение теплоты вдоль одной координатной оси по телу с площадью поперечного сечения S и длиной l, имеем тепловой поток

$$I = U/R_T, \quad (6.73)$$

где U — перепад температур на участке длиной l; $R_T = l/(S\lambda)$ — тепловое сопротивление.

Теплоперенос с помощью ***конвекции и лучеиспускания*** на макроуровне описывается также уравнениями вида (6.73) с той разницей, что величины R_T определяются иначе, чем в случае теплопроводности.

Из определения удельной теплоемкости

$$C_{\text{уд}} = \frac{1}{\rho}\frac{dQ}{dT}$$

следует уравнение теплоемкости

$$I = dQ/dt = C_T\, dU/dt. \quad (6.74)$$

Здесь Q — количество теплоты; ρ — плотность; $C_T = \rho C_{\text{уд}}$ — теплоемкость тела.

Из элементов с компонентными уравнениями (6.73) и (6.74) составляются эквивалентные схемы для анализа тепловых процессов во многих объектах, например в конструкциях РЭА.

В ***гидравлических системах*** наличие ***вязкого трения*** обусловливает появление в эквивалентных схемах ***гидравлического сопротивления***. ***Математическая модель гидравлического сопротивления*** для участка трубопровода круглого сечения при ламинарном течении жидкости имеет

$$U = R_{\text{г}} I,$$

где U — перепад давлений на рассматриваемом участке длиной l и радиусом r; I — массовый ноток жидкости;

$$R_{\text{г}} = 8\nu l/(\pi r^4);$$

ν— кинематическая вязкость.

При ***турбулентном*** характере течения жидкости

$$U = R_{\text{г}} I\,|I|; \quad R_{\text{г}} = \lambda l/(4\,\pi^2 r^5)$$

где

$$\lambda = 0.37 \sqrt[4]{\pi r \nu / |I|}.$$

Кроме сопротивления участок трубопровода характеризуется ***гидравлической индуктивностью*** $L_{\text{г}} = l/S$, а любая полость в гидравлической системе может рассматриваться как ***гидравлическая емкость:***

$$C_{\text{г}} = Sl\rho/E.$$

Поэтому трубопровод в гидравлической и пневматической системах можно представлять состоящим из нескольких секций, заменяя каждую секцию эквивалентной схемой рис. 6.22.

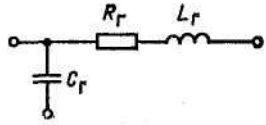


Рис. 6.22. Эквивалентная схема одной секции трубопровода

Консультант, учитывая конкретные условия, может пренебрегать какими-либо из элементов R_T, $C_{\text{г}}$ или $L_{\text{г}}$ в отдельных секциях и выбирать нужное число секций.

6.10. Построение математических моделей консультируемых проблем на макроуровне

Математическая модель консультируемой проблемы (ММКП) на макроуровне представляет собой ***систему обыкновенных дифференциальных уравнений*** (ОДУ)

$$\mathbf{F}\left(\frac{d\mathbf{V}}{dt}, \mathbf{V}, t\right) = 0, \tag{6.75}$$

где **V** — вектор зависимых переменных, называемых также базисными или определяющими координатами; *t* — время.

При получении системы (6.75) исходными являются ***компонентные и топологические уравнения***. Поскольку выбор как формы исходных топологических уравнений, так и формы итоговой модели неоднозначен, для получения ММКП возможно применение ряда методов. Будем использовать три основные формы представления ММКП (6.75) на макроуровне:

- ***нормальная форма***

$$d\mathbf{V}/dt = \varphi(\mathbf{V}, t),$$

ее применение удобно при ориентации на последующее численное решение явными методами;

- ***линеаризованная форма***

$$\mathbf{D}\frac{d\mathbf{V}}{dt} + \mathbf{H}\mathbf{V} = \mathbf{B}, \tag{6.76}$$

где **D** и **H** — матрицы; **B** — вектор. Форма (6.76) используется в программах, открытых по отношению к методам интегрирования;

- ***алгебраизованная и линеаризованная форма***

$$\mathbf{Я}_n\mathbf{V}_n = \mathbf{B}_n, \tag{6.77}$$

где **Я** — матрица; **B** — вектор. Их вид определяется выбранными методами алгебраизации и линеаризации. Обозначения $\mathbf{Я}_n$, $\mathbf{V}_n$, $\mathbf{B}_n$, используются для значений переменных **Я, V, B**, соответствующих моменту времени t_n.

В подавляющем большинстве программ анализа применяют форму (6.77). Для получения ММКП в такой форме применяют ***методы узловых потенциалов*** (МУП) и ***табличные методы***. В этих методах для алгебраизации реализуют одну из неявных разностных формул численного интегрирования

$$\left.\frac{d\mathbf{V}}{dt}\right|_n = \eta_n \mathbf{V}_n + \mu_n, \tag{6.78}$$

где $(d\mathbf{V}/dt)_n$ и $\mathbf{V}_n$ — значения векторов $d\mathbf{V}/dt$ и **V** в момент времени t_n; *n* — номер шага интегрирования; η_n — коэффициент, зависящий от выбранного метода интегрирования и значения шага h_n; μ_n — вектор, зависящий также от значений векторов **V** на *p* предыдущих шагах интегрирования, *p* — порядок метода.

Линеаризацию выполняют с помощью разложения нелинейных элементов вектора $\mathbf{F}(d\mathbf{V}/dt, \mathbf{V}, t)$ в ряд Тейлора с сохранением в разложении только линейных членов.

Различия между МУП и табличными методами заключаются в выборе исходных топологических уравнений и вектора базисных координат.

Для получения ММКП в нормальной форме наиболее приемлем метод переменных, характеризующих состояние системы, называемой обычно более коротко — метод переменных состояния (МПС).

Метод узловых потенциалов. Исходные топологические уравнения в МУП — уравнения закона токов Кирхгофа (ЗТК) или аналогичные им уравнения, выражающие равновесие переменных типа потока во всех узлах эквивалентной схемы, за исключением лишь одного узла, принимаемого за базовый,

$$\mathbf{AI} = 0, \tag{6.79}$$

где **A** — матрица иициденций; **I** — вектор токов.

Матрица инцидеиций характеризует связи узлов и ветвей эквивалентной схемы. В матрице инциденций *i*-я строка соответствует *i*-му узлу, а *j*- й столбец — *j*-й ветви дерева. Всего в матрице α столбцов и β строк, где α и β — число ветвей и узлов в эквивалентной схеме. Элемент матрицы a_{ij}= +1, если *i*-й узел инцидентен *j*-й ветви и положительное направление тока в этой ветви выбрано от i-го узла; a_{ij}=—1 при тех же условиях инцидентности, но при противоположном направлении тока, иначе a_{ij}=0.

Исходные компонентные уравнения в классическом варианте МУП должны иметь вид

$$\mathbf{I}_n = \mathbf{G}\left(\left.\frac{dU}{dt}\right|_n, \mathbf{U}_n, t_n\right), \tag{6.80}$$

где $\mathbf{I}_n$ и $\mathbf{U}_n$ — векторы токов и напряжений ветвей на *n*-м шаге.

Алгебраизация с помощью (6.78) приводит к системе уравнений

$$\mathbf{I}_n = \mathbf{G}\left[(\eta_n \mathbf{U}_n + \mu_n), \mathbf{U}_n, t_n\right].$$

После линеаризации относительно вектора $\mathbf{U}_n$ компонентные уравнения имеют вид

$$\mathbf{I}_n = \mathbf{Y}_n\mathbf{U}_n+\mathbf{Q}_n, \tag{6.81}$$

где $\mathbf{Y}_n=\partial\mathbf{I}_n/\partial\mathbf{U}_n$ —матрица проводимостей; $\mathbf{Q}_n$ — вектор, зависящий от значений вектора **U**, полученных на предыдущих шагах интегрирования.

В качестве основных базисных координат в МУП используют узловые потенциалы, вектор которых на *n*-м шаге обозначим φ_n. Отметим, что связь между векторами $\mathbf{U}_n$ и φ_n выражается с помощью матрицы инциденций

$$\mathbf{U}_n = -\mathbf{A}^t\varphi_n. \tag{6.82}$$

Подставив (6.82) в (6.81) и (6.81) в (6.79), имеем

$$-\mathbf{A}\mathbf{Y}_n\mathbf{A}^t\varphi_n + \mathbf{A}\mathbf{Q}_n = 0,$$

т. е. получена алгебраизованная и линеаризованная форма ММКП в виде

$$\mathbf{Я}_n\varphi_n = \mathbf{B}_n, \qquad (6.83)$$

где $\mathbf{Я}_n = \mathbf{A}\mathbf{Y}_n\mathbf{A}^t$—матрица Якоби; $\mathbf{B}_n = \mathbf{A}\mathbf{Q}_n$— вектор правых частей.

В классическом варианте МУП имеются ограничения на вид компонентных уравнений. Применительно к схемной форме представления моделей эти ограничения выражаются в недопустимости таких ветвей, как идеальные источники напряжения и любые ветви, параметры которых зависят от каких-либо токов. В модифицированном варианте МУП эти ограничения снимаются благодаря расширению вектора базисных координат — дополнительно к узловым потенциалам к базисным координатам относят также токи особых ветвей. Особыми ветвями при этом называют: 1) ветви источников напряжения; 2) ветви, токи которых являются управляющими (аргументами в выражениях для параметров зависимых ветвей); 3) индуктивные ветви.

Обозначим через $\mathbf{I}_1$ и $\mathbf{I}_2$ векторы токов, а через $\mathbf{A}_1$ и $\mathbf{A}_2$ — подматрицы инциденций узлов с неособыми и особыми ветвями соответственно. Тогда уравнение (6.79) перепишем в виде

$$\mathbf{A}_1\mathbf{I}_1 + \mathbf{A}_2\mathbf{I}_2 = 0. \qquad (6.84)$$

Компонентные уравнения неособых ветвей имеют вид

$$\mathbf{I}_1 = \mathbf{Y}_1\mathbf{U} + \mathbf{Y}_2\mathbf{I}_2 + \mathbf{Q},$$

где в дополнение к уравнению (6.81) в правой части фигурирует слагаемое, определяемое токами особых ветвей. Здесь $\mathbf{Y}_1 = \partial\mathbf{I}_1/\partial\mathbf{U}$, $\mathbf{Y}_2 = \partial\mathbf{I}_1/\partial\mathbf{I}_2$; n — индекс, указывающий номер шага, опущен, но подразумевается, что все переменные величины относятся к n-му шагу интегрирования, Подстановка $\mathbf{I}_1$ в (6.84) дает подсистему из (β—1)-го уравнения

$$-\mathbf{A}_1\mathbf{Y}_1\mathbf{A}_1^t\varphi + (\mathbf{A}_1\mathbf{Y}_2 + \mathbf{A}_2)\mathbf{I}_2 + \mathbf{A}_1\mathbf{Q} = 0,$$

в которую входят β + γ—1 неизвестных (γ —число особых ветвей). Эта подсистема доопределяется с помощью γ компонентных уравнений особых ветвей.

Алгоритм вычисления матрицы Якоби в методе узловых потенциалов. Вычисление матрицы Якоби как матричного произведения $\mathbf{AYA}^t$ без учета разреженности матриц **A** и **Y** нерационально, так как приводит к излишне большим затратам машинных времени и памяти. Например, в схеме средней сложности, включающей β = 51 и α = 80, матрица **A** имеет размер 50×80, а матрица

Y—размер 80×80, т. е. только эти две матрицы для хранения всех их элементов требуют около 10 000 ячеек памяти. В то же время статистические исследования показывают, что ненулевыми в этих матрицах оказываются лишь около 240 элементов. Поэтому на практике используют алгоритмы формирования матрицы Якоби, учитывающие сильную разреженность матриц **A** и **Y**.

Рассмотрим алгоритм вычисления матрицы Якоби в классическом варианте МУП. Этот же алгоритм используют в модифицированном варианте МУП для вычисления подматрицы $\mathbf{A_1 Y A}^t_1$, входящей в матрицу Якоби. Исходная информация о матрицах **A** и **Y** в этом алгоритме задается в виде списка, где *k*-я строка списка соответствует *k*-й ветви эквивалентной схемы. В этой строке указываются номера *i* и *j* узлов, инцидентных *k*-й ветви, местонахождение (адрес) проводимости $\partial I_k/\partial U_k$ (где I_k и U_k — ток и напряжение *k*-й ветви) и, если I_k зависит также от потенциала φ_p некоторого узла *p,* отличного от *i* и *j,* — местонахождение производной I_k по потенциалу этого узла. Каждая *k*-я строка приводит к добавлению проводимостей в некоторые клетки матрицы **Я**. Строки списка просматриваются поочередно. Просмотр *k*-й строки приводит к добавлению проводимости $\partial I_k/\partial U_k$ в диагональные клетки $я_{ii}$ и $я_{jj}$ и к вычитанию этой проводимости из клеток $я_{ij}$ и $я_{ji}$. Если имеется $\partial I_k/\partial U_k \neq 0$, то эта производная добавляется в клетки $я_{ip}$, и $я_{jp}$, причем если ток I_k направлен от узла *j* к узлу *i*,то добавление в $я_{ip}$ происходит со знаком «плюс», а в $я_{jp}$ — со знаком «минус». Пример формирования матрицы Якоби приведен ниже.

Табличный метод. В качестве базисных координат исполъзуют токи и напряжения всех ветвей схемы, а в качестве исходных топологических уравнений — уравнения Кирхгофа. Эти уравнения записывают для системы контуров и сечений, выбранной в схеме так, чтобы получить α топологических линейно независимых уравнений, где α — число ветвей в схеме. В этих уравнениях фигурируют 2α неизвестных токов и напряжений, поэтому система уравнений доопределяется с помощью α компонентных уравнений

Применяемый способ выбора системы независимых контуров и сечений основан на построении фундаментального дерева в графе схемы. Используется полюсный граф, повторяющий структуру эквивалентной схемы. Фундаментальное дерево связного графа есть связный подграф, включающий β—1 ребро и не имеющий циклов. Ребра, вошедшие в дерево, образуют множество ветвей дерева (ВД), а остальные ребра — множество ветвей, называемых хордами (ВХ). Контуром *k*-й хорды называют подмножество ребер графа (ветвей схемы), входящих в замкнутый контур, образуемый при подключении

k-й хорды к дереву. Сечения образуются следующим образом: отделим часть нершин графа от остальных с помощью замкнутой линии сечения, проведя ее так, чтобы ни одно ребро не пересекалось более одного раза и при этом пересекалась одна и только одна ветвь дерева. Следовательно, каждому сечению соответствует определенная ветвь дерева. На рис. 6.23, *а* для примера приведена некоторая схема, а на рис. 6.23, *б* — ее граф с выделенным жирными линиями фундаментальным деревом.

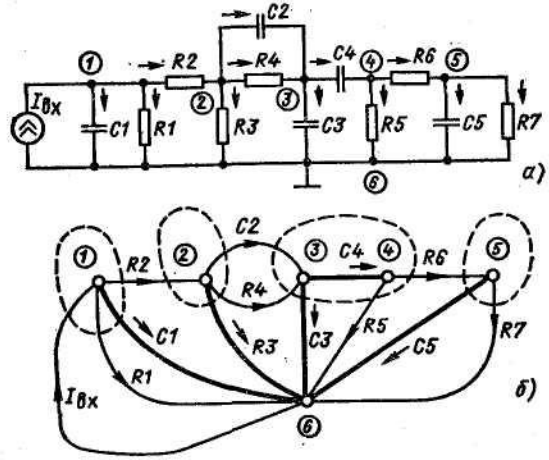


Рис. 6.23. Эквивалентная схема (*а*) и ее граф *(б)*

Штрихом показаны линии сечения. Уравнения токов Кирхгофа для сечений ветвей дерева и напряжений Кирхгофа для контуров хорд образуют систему независимых топологических уравнений

$$\left.\begin{aligned} \mathbf{I}_{в.д} - \mathbf{M}^{t}\mathbf{I}_{x} &= 0, \\ \mathbf{U}_{x} + \mathbf{M}\mathbf{U}_{в.д} &= 0, \end{aligned}\right\} \qquad (6.85)$$

$\mathbf{I}_{в.д}$, $\mathbf{I}_{x}$, $\mathbf{U}_{в.д}$, $\mathbf{U}_{x}$ — векторы токов и напряжений ветвей и хорд соответственно; **M** — матрица контуров и сечений.

Деление ветвей на хорды и ветви дерева может выполняться по различным правилам. В табличном методе рекомендуется выбирать фундаментальное дерево так, чтобы минимизировалось количество ненулевых элементов в матрице **M**. Элемент m_{ij} матрицы **M** равен ±1, если в контур *i*-й хорды входит *j*-я ветвь дерева, иначе m_{ij} =0. Знаки у $m_{ij}\neq 0$ выбирают по специальному правилу: m_{ij}=+1, если положительные направления токов в *i*-й хорде и *j*-й ветви дерева при обходе контура

совпадают; m_{ij} =—1, если эти направления не совпадают. В столбцах источников напряжения записываются +1, если ток подключенной хорды втекает в выбранный за положительный полюс источника.

На рис. 6.23 матрица **M** имеет вид табл. 6.5.

Т а б л и ц а 6.5

	C1	*C3*	*C4*	*C5*	*R3*
C2	0	+1	0	0	—1
R1	—1	0	0	0	0
R2	—1	0	0	0	+1
R4	0	+1	0	0	—1
R5	0	—1	+1	0	0
R6	0	—1	+1	+1	0
R7	0	0	0	—1	0
$I_{вх}$	+1	0	0	0	0

Компонентные уравнения $\mathbf{F}(\mathbf{V}', \mathbf{V}, t)=0$, где $\mathbf{V}=(\mathbf{U}_x, \mathbf{I}_{в.д}, \mathbf{U}_{в.д}, \mathbf{I}_x)$, $\mathbf{V}'=d\mathbf{V}/dt$, предварительно алгебраизуются с помощью (6.78) и линеаризуются относительно вектора $\mathbf{V}_n$:

$$\mathbf{W}\mathbf{V}_n = Q, \qquad (6.86)$$

где $\mathbf{W}=\eta\partial F/\partial\mathbf{V}'+\partial\mathbf{F}/\partial\mathbf{V}$; **Q** — вектор правых частей.

Окончательно ММКП представляет собой систему из 2*n* уравнений (6.85) и (6.86). Сравнительно высокий порядок получающейся системы уравнений — недостаток табличного метода.

Метод переменных состояния. Метод ориентирован на получение ММКП в виде системы обыкновенных дифференциальных уравнений в нормальной форме с последующим ее интегрированием явными методами. В качестве базисных координат (переменных состояния) выбирают независимые емкостные напряжения и индуктивные токи, а в качестве исходных топологических уравнений — уравнения типа (6.85). Подмножество ветвей дерева ВД образуется путем включения в него ветвей схемы в следующем порядке — ветви источников напряжения, емкостные, резистивные, индуктивные. Именно такой приоритет ветвей обусловливает последующее получение ММКП в форме либо нормальной, либо легко преобразуемой в нормальную. После выбора дерева и построения матрицы **M** систему (6.85) преобразуют — из нее исключают все токи

и напряжения, не относящиеся к переменным состояниям. Исключення производят с помощью компонентных уравнений.

Наиболее просто такие исключения выполняют в том случае, если в схеме отсутствуют топологические вырождения (рис. 6.24), под которыми понимаются емкостные контуры (*а*) и индуктивные звезды *(б).*

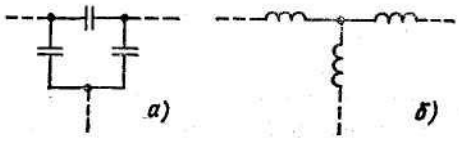


Рис. 6.24. Топологические вырождения

К топологическим вырожденям относят также такие ситуации, при которых в контур резистивной хорды оказывается включенной хотя бы одна резистивная ветвь дерева. Наличие топологических вырождений усложняет процедуру получения ММКП в нормальной форме и ее использование — требуется либо решение систем линейных алгебраических уравнений на каждом шаге численного интегрирования системы обыкновенных дифференциальных уравнений, либо предварительное устранение топологических вырождений с помощью изменений схемы. Очевидно, что последний прием может привести к дополнительным погрешностям решения или к ухудшению обусловленности системы уравнений.

Пример получения ММКП различными методами. Для примера рассмотрим схему, показанную на рис. 6.23. Предполагается, что численное интегрирование уравнений будет выполняться с помощью метода первого порядка точности. Неявная формула такого метода

$$\mathbf{V}_n = (\mathbf{V}_n\text{-}\mathbf{V}_{n-1},)/\ h_n, \qquad (6.87)$$

где h_n — величина n-го шага интегрирования.

В методе узловых потенциалов компонентные уравнения

$$I_n = y_i U_n;\ I = C_i dU/dt.$$

Последнее уравнение, алгебраизованное на n-м шаге интегрирования с помощью (6.87), принимает вид

$$I_n — y_i U_n = Q_n. \qquad (6.88)$$

Здесь $y_i = 1/R_i;\ y_i = C_i/\ h_n;\ Q_n = — C_i U_{n-1}/\ h_n.$

Использование вышеописанного алгоритма формирования матрицы Якоби непосредственно приводит к ММКП вида (6.89).

$$\begin{array}{|c|c|c|c|c|}
\hline
y_{R1}+y_{R2}+y_{C1} & -y_{R2} & 0 & 0 & 0 \\
\hline
-y_{R2} & y_{R2}+y_{R3}+y_{R4}+y_{C2} & -y_{R4}-y_{C2} & 0 & 0 \\
\hline
0 & -y_{R4}-y_{C2} & y_{R4}+y_{C2}+y_{C3}+y_{C4} & -y_{C4} & 0 \\
\hline
0 & 0 & -y_{C4} & y_{R5}+y_{R6}+y_{C4} & -y_{R6} \\
\hline
0 & 0 & 0 & -y_{R6} & y_{R6}+y_{R7}+y_{C5} \\
\hline
\end{array} \times$$

$$\times \begin{array}{|c|} \hline \varphi_1 \\ \hline \varphi_2 \\ \hline \varphi_3 \\ \hline \varphi_4 \\ \hline \varphi_5 \\ \hline \end{array} = \begin{array}{|c|} \hline I_{\text{вх}}+y_{C1}\varphi_1' \\ \hline y_{C2}\left(\varphi_2'-\varphi_3'\right) \\ \hline y_{C2}\left(\varphi_3'-\varphi_2'\right)+y_{C3}\varphi_3'+y_{C4}\left(\varphi_3'-\varphi_4'\right) \\ \hline y_{C4}\left(\varphi_4'-\varphi_3'\right) \\ \hline y_{C5}\varphi_5' \\ \hline \end{array} .$$

(6.89)

Здесь y_p — проводимость ветви p; φ_k— потенциал k-го узла на данном шаге интегрирования; φ'_k — то же на предыдущем шаге интегрирования.

В табличном методе полученная матрица **M** дана в табл. 6.6.

Таблица 6.6

	C1	*C2*	*C3*	*C4*	*C5*
R1	—1	0	0	0	0
R2	—1	+1	+1	0	0
R3	0	—1	—1	0	0
R4	0	—1	0	0	0
R5	0	0	—1	+1	0
R6	0	0	—1	+1	+1
R7	0	0	0	0	—1
$I_{\text{вх}}$	—1	0	0	0	0

Итоговая ММКП есть следующая система линейных алгебраических уравнений:

1	0	M	0	U_x	=	0
0	1	0	$-M^t$	$I_{в.д}$		0
0	1	$-Y_{в.д}$	0	$U_{в.д}$		Q1
$-Y_x$	0	0	1	I_x		Q2

где **1**—единичная подматрица; $\mathbf{Y}_{в.д}$ и $\mathbf{Y}_x$ — диагональные подматрицы с проводимостями ветвей дерева и хорд соответственно на диагонали; **Q**1 и **Q**2 — подвекторы правых частей компонентных уравнений ветвей (6.88). Здесь первые две строки суть топологические уравнения (6.85), а две последующие — компонентные уравнения.

В методе переменных состояния граф и дерево, выбранное в соответствии с приоритетами ветвей, показаны на рис. 6.25.

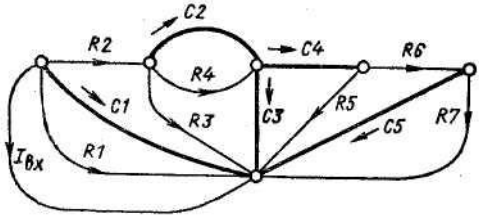


Рис. 6.25. Граф и выбранное дерево в методе переменных состояния

Матрица **M** получается в виде табл. 6.6. Вектор правых частей **φ**(**V**, *t)* в системе уравнений **V**′=φ(**V**, *t)* с помощью полученной матрицы **M** находят по следующему алгоритму:

1. Вычисление вектора резистивных напряжений с помощью второго из уравнений (6.85).

$$\mathbf{U}_x = -\mathbf{M}\mathbf{U}_{в.д}.$$

Это матричное уравнение удобно развернуть в следующую совокупность уравнений путем сканирования матрицы **M** по строкам:

$$U_{R1} = U_{C1};$$
$$U_{R2} = U_{C1} - U_{C2} - U_{C3};$$
$$U_{R3} = U_{C2} + U_{C3};$$
$$U_{R4} = U_{C2};$$
$$U_{R5} = U_{C3} - U_{C4};$$
$$U_{R6} = U_{C3} - U_{C4} - U_{C5};$$
$$U_{R7} = U_{C5}.$$

Отметим, что значения фигурирующих в правой части переменных состояния известны — на первом шаге это начальные условия, на каждом из последующих шагов — это значения, полученные на предыдущем шаге.

2. Вычисление вектора резистивных токов с помощью компонентных уравнений

$$I_{Ri} = U_{Ri}/R_i.$$

3. Вычисление вектора емкостных токов с помощью первого из топологических уравнений (6.85)

$$\mathbf{I}_{в.д} = \mathbf{M}^t \mathbf{I}_x.$$

Это матричное уравнение при вычислениях разворачивается в системе уравнений путем сканировании матрицы **M** по столбцам:

$$I_{C1} = -I_{R1} - I_{R2} - I_{вх};$$
$$I_{C2} = I_{R2} - I_{R3} - I_{R4};$$
$$I_{C3} = I_{R2} - I_{R3} - I_{R5} - I_{R6};$$
$$I_{C4} = I_{R5} + I_{R6};$$
$$I_{C5} = I_{R6} - I_{R7}.$$

4. Вычисление вектора производных переменных состояния с помощью компонентных уравнений

$$dU_{Ci}/dt = I_{Ci}/C_i.$$

Далее применяют ту или иную формулу численного интегрирования, преобразующую вектор производных в вектор переменных состояния для очередного момента моделируемого времени, после чего переходят к новому шагу интегрирования.

6.11. Математические модели на метауровне

Математические модели консультируемой проблемы, представленной аналоговой РЭА. Использование рассмотренных положений схемотехнического моделирования при формировании рекомендаций для проектирования сложной аналоговой РЭА на метауровне оказывается затруднительным из-за чрезмерно больших размерностей задач. Необходимы упрощения. Основой снижения размерности задач является ***макромоделирование***. Часто используют ряд дополнительных упрощений и допущений. Главные из них формулируются следующим образом:

1. Однонаправленность в передаче сигналов, т. е. использование макромоделей, в которых отсутствует влияние выходных переменных на состояние входных цепей.

2. Отсутствие влияния нагрузки на параметры и состояние моделируемых консультируемых проблем.

3. Использование вместо фазовых переменных двух типов (напряжение и ток) переменных одного типа, называемых ***сигналами***. При этом ***компонентные уравнения элемента*** представляют собой ***уравнения связи сигналов на входах и выходах этого элемента***.

4. Линейность моделей инерционных элементов.

Перечисленные допущения характерны для ***функционального моделирования,*** широко используемого при формировании рекомендаций для анализа консультируемых проблем в области систем автоматического управления. Элементы (звенья) систем при функциональном моделировании делят на три группы:

1) линейные безынерционные звенья для отображения таких функций, как повторение, инвертирование, чистое запаздывание, идеальное усиление, суммирование сигналов;

2) нелинейные безынерционные звенья для отображения различных нелинейных преобразований сигналов (ограничение, детектирование, модуляция и т. п.);

3) линейные инерционные звенья для выполнения дифференцирования, интегрирования, фильтрации сигналов. Инерционные элементы представлены отношениями преобразованных по Лапласу или Фурье выходных и входных фазовых переменных. ***При анализе во временной области применяют преобразование Лапласа, модель инерционного элемента с одним входом и одним выходом есть передаточная функция, а при анализе в частотной области — преобразование Фурье, модель элемента есть выражения амплитудно-частотной и частотно-фазовой характеристик. При наличии нескольких входов и выходов ММ элемента представляется матрицей передаточных функций или частотных характеристик.***

Рассмотрим примеры моделей элементов аналоговой РЭА. Анализ однокаскадного RC-усилителя, проводимый в курсах основ электроники, позволяет получить следующее приближенное выражение для передаточной функции каскада:

$$h(p) = \frac{K_0}{(1 + p\tau_в)\ (1 + 1/(p\tau_н))},$$

где K_0 — коэффициент усиления на средних частотах; $\tau_в$ и $\tau_н$ — постоянные времени каскада на высоких и низких частотах соответственно.

Выражение частотной характеристики того же усилителя

$$h(j\omega) = \frac{K_0}{(1 + j\omega\tau_в)\ (1 + 1/(j\omega\tau_н))}.$$

Для однокаскадного резонансного *LC*-усилителя передаточная функция имеет вид

$$h(p) = S\frac{pL + r}{LCp^2 + Crp + 1}, \tag{6.90}$$

где S — крутизна усиления на резонансной частоте; L, C, r — индуктивность, емкость и последовательное сопротивление потерь параллельного колебательного контура.

Примером нелинейного элемента является амплитудный модулятор, для которого при функциональном моделировании используют модель в виде

$$U_{вых}(t) = U_m(1 + mU_{вх}(t))\sin(\omega t + \varphi),$$

где $U_{вых}$ и $U_{вх}$ — напряжения модулированных и модулирующих колебаний; m — коэффициент модуляции; ω и φ — частота и фаза несущей; U_m— амплитуда несущей на выходе модулятора.

Для автогенераторов в качестве модели используется описание формы генерируемых колебаний, так, в случае генератора гармонических колебаний имеем

$$U(t) = U_m \sin(\omega t + \varphi),$$

где $U(t)$, U_m, ω, φ — напряжение, амплитуда, частота и фаза колебаний.

Допущения, принимаемые при функциональном моделировании, существенно упрощают алгоритмы получения математических моделей консультируемых проблем (ММКП) из математических моделей элементов (ММЭ). Математическая модель консультируемой проблемы представляет собой совокупность ММЭ, входящих в консультируемую проблему, при отождествлении переменных, относящихся к соединяемым входам и выходам.

Пример 6.11.1. Электронный усилитель работает в малосигнальном режиме и состоит из двух каскадов и цепи обратной связи, имеющих передаточные функции $K_1(p)$, $K_2(p)$ и $K_3(p)$ соответственно. Математическая модель может быть получена непосредственно по схеме усилителя, представленной на рис. 6.26:

$$U_{вых}(p) = K^*(U_{вх}(p) + K_з(p)\,U_{вых}(p)),$$

т. е. передаточная функция усилителя

$$h(p) = U_{вых}(p)/U_{вх}(p) = K^*/[1 - K^*K_з(p)], \tag{6.91}$$

где $K^* = K_1(p)K_2(p)$.

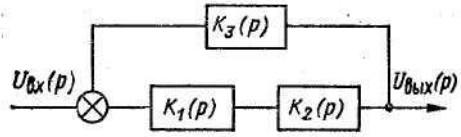


Рис. 6.26. Схема усилителя

Для исследования линейных систем во временной области на основе модели типа (6.91) можно использовать два подхода.

Первый подход связан с применением ***правил операционного исчисления*** и требует выполнения прямого преобразования Лапласа над входными сигналами и обратного преобразования над выходными. Второй подход связан с представлением модели (6.91) в виде ***системы обыкновенных дифференциальных уравнений с ее последующим численным интегрированием.***

Пример 6.11.2. Модель резонансного усилителя в виде системы обыкновенных дифференциальных уравнений получается из (6.90) с учетом того, что $U_{вых}(p) = h(p)U_{вх}(p)$ и $p = d/dt$:

$$LC\frac{d^2U_{вых}}{dt^2} + Cr\frac{dU_{вых}}{dt} + U_{вых} = S\left(L\frac{dU_{вх}}{dt} + rU_{вх}\right)$$

или, вводя обозначение $I = CdU_{вых}/dt$, образуем систему уравнений в нормальной форме Коши

$$I = CdU_{вых}/dt;\ LdI/dt + rI + U_{вых} = S\,(LdU_{вх}/dt + rU_{вх}),$$

где $U_{вх}$, $U_{вых}$ — заданная и рассчитанная функции времени соответственно.

При наличии в схеме нелинейных звеньев применяют второй из вышеназванных подходов.

При формировании рекомендаций по проектированию систем, в которых информация представлена в виде огибающей высокочастотных колебаний, возможны два способа введения переменных в модели.

При первом способе несущей переменные изображают высокочастотные модулированные колебания. При анализе приходится имитировать поведение консультируемой проблемы в течение большого числа периодов несущей, что зачастую делает неприемлемо крупными затраты машинного времени.

При втором способе огибающей переменные отображают огибающие высокочастотных колебаний. Отражение только низкочастотной огибающей существенно ускоряет вычисления, однако построение моделей может оказаться затруднительным.

Математические модели логических схем цифровой РЭА.

На функционально-логическом уровне необходим ряд положений, упрощающих модели устройств и тем самым позволяющих анализировать более сложные консультируемые проблемы по сравнению с консультируемыми проблемами, анализируемыми на схемотехническом уровне. Часть используемых положений аналогична положениям, принимаемым для моделирования аналоговой РЭА. Во-первых, это положение о представлении состояний консультируемой проблемы с помощью однотипных фазовых переменных (обычно напряжений), называемых сигналами. Во-вторых, не учитывается влияние нагрузки на функционирование элементов-источников.
В третьих, принимается допущение об однонаправленности, т. е. о возможности передачи сигналов через элемент только в одном направлении — от входов к выходам. Дополнительно к этим положениям при моделировании цифровой РЭА принимается положение о дискретизации переменных, их значения могут принадлежать только заданному конечному множеству — алфавиту, например двоичному алфавиту {0,1}.

Моделирование цифровой РЭА возможно с различной степенью детализации. На логическом *(вентильном)* подуровне функционально-логического проектирования в качестве элементов аппаратуры рассматривают простые схемы типа ***вентилей***, на регистровом подуровне элементами могут быть как отдельные вентили, так и любые более сложные сочетания простых схем, например ***регистры, счетчики, дешифраторы, сумматоры, арифметико-логические устройства*** и т. п.

Рассмотрим ***математические модели элементов на логическом подуровне.*** Для одновыходных комбинационных элементов ММ представляет собой выражение (в общем случае алгоритм), позволяющее по значениям входных переменных (значениям входов) в заданный момент времени t вычислить значение выходной переменной (значение выхода) в момент времени $t+t_з$, где $t_з$ — задержка сигнала в элементе. Такую модель элемента называют ***асинхронной***. При $t_з=0$ модель элемента называют ***синхронной***. Модель многовыходного элемента должна включать в себя алгоритм вычисления задержек и значений всех выходных сигналов.

Пример 6.11.3. Модель элемента 3И—НЕ с фиксированной задержкой $t_з$, входами u_1, u_2, u_3 и выходом

$$y(t+t_з)=\overline{u_1(t)\,\&\,u_2(t)\,\&\,u_3(t)}, \qquad (6.92)$$

где & — символ операции конъюнкции.

Для элементов последовательностных схем (элементов с памятью) используют модели, в которых аргументами выходных переменных y_j могут быть как входные u_i, так и внутренние u_k переменные. Вектор внутренних переменных V отражает состояние элемента (состояние его памяти).

Пример 6.11.4. Модель J—K триггера с входами J и K и состояниями V_0 в предыдущем такте и V в последующем такте

$$V = V_0 \& \overline{J} \& \overline{K} + \overline{V}_0 \& J \& K + J \& \overline{K},$$

где «+» — символ операции дизъюнкции.

Объединение моделей элементов в общую математическую модель консультируемой проблемы выполняется на основе вышеперечисленных допущений отождествлением переменных на соединяемых входах и выходах элементов.

Пример 6.115. На рис. 6.27 представлен фрагмент схемы цифрового устройства.

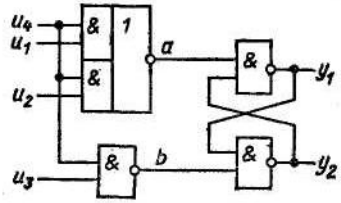


Рис. 6.27. Фрагмент схемы цифрового устройства

Соответствующая ему модель есть следующая система логических уравнений:

$$a = \overline{u_4 \& (u_1 + u_2)};$$
$$b = \overline{u_4 \& u_3};$$
$$y_1 = \overline{a \& y_2};$$
$$y_2 = \overline{b \& y_1}.$$

Двузначные и многозначные модели. Применение двузначного алфавита приводит к наиболее экономичным алгоритмам моделирования, однако двузначный алфавит ограничивает возможности анализа работоспособности схем. Поэтому чаще используют ***многозначное моделирование***.

В трехзначном алфавите сигналы могут принимать значения из множества {0, 1, X}, где 0 и 1 соответствуют низкому и высокому уровням сигнала, а значение X интерпретируется как неопределенное (неизвестное). В моделях элементов с трехзначным (иначе —

троичным) алфавитом, называемых трехзначными (троичными) моделями, сигналы могут принимать значение *X* во время переходных процессов, например во время своего переключения из единичного в нулевое состояние и обратно, а также в установившихся режимах при неопределенных значениях входных и внутренних переменных. Трехзначный алфавит используют для выявления ситуаций, в которых возможны сбои аппаратуры из-за рассогласования времен прохождении сигналов по различным цепям (из-за состязаний сигналов).

Дополнительные возможности обнаружения сбойных ситуаций предоставляет использование пятизначного алфавита {0, 1, *X, Д, Е*}, в котором с помощью значений *Д* и *Е* различают переходы сигналов из состояния 1 в состояние 0 и из состояния 0 в состояние 1 соответственно. Иногда алфавит расширяют в еще большей степени, добавляя в него символы, соответствующие таким ситуациям, как импульсные помехи, воздействие некоторых внешних факторов и т. п.

Результаты выполнения основных логических операций над аргументами *a* и *b* в дву-, трех- и пятизначном алфавитах представлены в табл. 6.7 (здесь аргументом для операции НЕ служит переменная *b).*

Таблица 6.7

a	0	1	*X*	*E*	*Д*
b	0 1 X E Д	0 1 X E Д	0 1 X E Д	0 1 X E Д	0 1 X E Д
Результат операции И	0 0 0 0 0	0 1 X E Д	0 X X X X	0 E X E X	0 Д X X Д
Результат операции ИЛИ	0 1 X E Д	1 1 1 1 1	X 1 X X X	E 1 X E X	Д 1 X X Д
Результат операции НЕ	1 0 X Д E	– – – – –	– – – – –	– – – – –	– – – – –

При этом к дву- и трехзначному моделированию относятся только те столбцы, в которых оба аргумента имеют значения соответственно из множества {0, 1} и {0, 1, *X}.*

Синхронные и асинхронные модели.

Синхронные модели применяют для анализа установившихся состояний логических схем. Они представляют собой системы логических уравнений вида

$$\mathbf{V} = \mathbf{F}(\mathbf{V}, \mathbf{U}), \qquad (6.93)$$

где **V** — вектор, состоящий из выходных *y,* и внутренних переменных; **U** — вектор входных переменных.

Примером синхронной модели может служить вышеприведенная система уравнений для схемы, показанной на рис. 6.27.

Система уравнений (6.93) для последовательных схем имеет столько решений, сколько устойчивых состояний при заданном **U** имеет моделируемая схема. Как правило, для анализа синхронных

моделей используют методы, позволяющие получить то решение, которое соответствует исходному значению вектора **V** и заданному входному набору **U.** Получение такого решения называют ***синхронным моделированием.***

Синхронное моделирование на основе двузначного алфавита позволяет проверить схему на отсутствие грубых ошибок типа неправильных соединений элементов. Дополнительную информацию о наличии в схеме рисков сбоя получают при применении трех- и пятизначного алфавита.

Трехзначное синхронное моделирование позволяет обнаружить статические риски сбоя. ***Статический риск сбоя*** выражается в появлении ложных сигналов на выходе схемы при неблагоприятном рассогласовании времен переключения входных сигналов.

Пример 6.11.6. Па рис. 6.28 приведены возможные временные диаграммы для входов u_1, u_2 и выхода y схемы И.

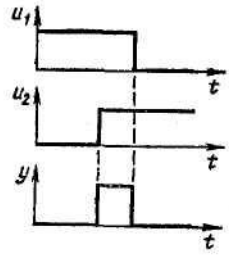


Рис. 6.28. Диаграммы статического риска сбоя

Из рисунка видно появление ложного сигнала на выходе схемы, что и является проявлением статического риска сбоя.

Для обнаружения статического риска сбоя требуется на каждом такте синхросигналов двукратное решение уравнений синхронной модели. Первое решение проводится при промежуточных значениях входных переменных: все изменяющиеся из состояний 1 или 0 входные переменные получают значение *X*, не изменяющиеся сохраняют свои исходные значения. Второе решение проводится при итоговых значениях входных переменных. Если у какой-либо переменной в схеме исходное, промежуточное и итоговое значения имеют последовательности 0—X—0 или 1—X—1, то данная переменная изображает ложный сигнал, т. е. указывает на наличие статического риска сбоя.

Пример 6.11.7. Для временной диаграммы (рис. 6.28) входная переменная u_1 имеет последовательность исходного, промежуточного и

итогового значений 1—Х—0; переменная u_2 — последовательность 0—Х—1 (значение Х используется для отображения неопределенного состояния переменных при их переключении). В соответствии с правилами выполнения операции И в трехзначном алфавите (табл. 6.7) имеем для переменной *y* последовательность 0—Х—0, что отождествляется со статическим риском сбоя.

Пятизначное синхронное моделирование позволяет дополнительно обнаруживать динамические риски сбоя. ***Динамический риск сбоя*** выражается в возможности многократного изменения некоторой переменной вместо правильного однократного изменения в течение одного такта синхронизации схемы.

Для выявления динамического риска сбоя выполняют двукратное решение системы логических уравнений при промежуточных и итоговых значениях входных переменных. Если у какой-либо изменяющейся переменной последовательность исходного, промежуточного и итогового значений отличается от возможных корректных последовательностей (корректными являются последовательности 0—Е—1 и 1—Д—0), то в схеме имеет место динамический риск сбоя.

Пример 6.11.8. На рис. 6.29,*б* приведена временная диаграмма, соответствующая такому сочетанию входных сигналов, при котором в схеме (рис. 6.29, *а*) проявляется динамический риск сбоя.

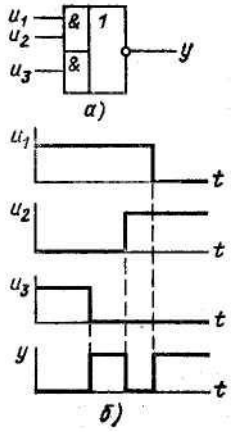


Рис. 6.29. Схема (*а*) и диаграммы *(б)* для иллюстрации динамического риска сбоя

Этой временно́й диаграмме соответствуют следующие последовательности значений переменных: для u_1—1—Д—0, для u_2—0—Е—1, для u_3—1—Д—0. Для переменной y по правилам выполнения логических операций в пятизначном алфавите получаем 0—Х—1, что отождествляется с динамическим риском сбоя.

Следует, однако, заметить, что синхронное моделирование указывает на возможность сбоев, которые в действительности происходят лишь при неблагоприятных рассогласованиях моментов переключения входных сигналов. Так, если (рис. 6.28) переключение переменной u_1 произойдет раньше переключения переменной u_2, то в действительности ложного сигнала на выходе не будет, хотя синхронное моделирование указывает на риск сбоя. В этом отношении результаты синхронного моделирования аналогичны результатам анализа на наихудший случай — они могут привести к отрицательным заключениям о работоспособности вполне корректных схем.

Асинхронные модели схем составляют из асинхронных моделей элементов и применяют для анализа переходных процессов в цифровой РЭА. Время t в асинхронных моделях дискретизируется и измеряется в количестве тактов. Продолжительность такта достаточно малая — не должна превышать допустимую погрешность расчета временных параметров.

Асинхронную модель схемы можно представить в виде

$$\mathbf{V}' = \mathbf{F}(\mathbf{V}, \mathbf{U}), \qquad (6.94)$$

где **U** — вектор входных переменных; **V** — вектор текущих значений внутренних и выходных переменных; **V′** — вектор будущих значений тех же перемелных. Примером логического выражения, входящего в систему (6.94), является выражение (6.92). Алгоритм асинхронного моделирования заключается в подстановке в текущий момент времени t известных значений **V** и **U** в правую часть выражений (6.94), в вычислении новых значений переменных и их задержек, в увеличении модельного времени на длительность такта и корректировке вектора **V** с учетом результатов расчета задвижек и значений вектора **V′**. Далее такты асинхронного моделирования повторяются до исчерпания заданного временного интервала анализа.

Асинхронные модели обычно используют с двузначным или трехзначным представлением переменных. *Трехзначное асинхронное моделирование* позволяет учесть разбросы задержек распространения сигналов в элементах. Пусть в момент времени t_1 на вход элемента приходит сигнал, изменяющий состояние элемента с 0 на 1с задержкой t_3, лежащей в интервале $[t_{3\min}, t_{3\max}]$. Тогда в асинхронной модели элемента значение выходной переменной

$$y = \begin{cases} 0 \text{ при } t_1 \leqslant t < t_1 + t_{3\min}; \\ X \text{ при } t_1 + t_{3\min} \leqslant t < t_1 + t_{3\max}; \\ 1 \text{ при } t \geqslant t_1 + t_{3\max}. \end{cases}$$

С помощью асинхронных моделей можно проанализировать прохождение сигналов во времени в цифровой РЭА с учетом реальных задержек в элементах при различных последовательностях входных сигналов. Однако асинхронное моделирование обычно требует заметно больших затрат машинного времени по сравнению с синхронным.

Математические модели функциональных схем цифровой РЭА на регистровом подуровне. Первая особенность ММ на регистровом подуровне связана с разнообразием типов функциональных узлов, рассматриваемых в качестве элементарных при моделировании. ***Разнообразие типов элементов влечет за собой разнообразие их математических моделей***. В ММ элементов могут использоваться различные ***типы данных***, в частности ***величины булевы, целые, вещественные.*** Эти ***величины*** могут быть ***скалярными и векторными***. Введение векторных переменных позволяет лаконично описывать многоразрядные счетчики, регистры, их входные и выходные сигналы. С помощью вещественных величин и операций над ними, которые присущи алгоритмическим языкам общего назначения, можно описать разнообразные алгоритмы, реализуемые в функциональных узлах различной сложности.

Математические модели на регистровом подуровне могут быть алгоритмического и схемного типов. ***Модели алгоритмического типа*** описывают алгоритмы функционирования устройств без привязки к их схемной реализации. ***Модели схемного типа*** отражают связи между переменными на входах и выходах функциональных узлов, составляющих анализируемую схему. Возможны смешанные модели, состоящие из алгоритмических и схемных описаний.

В большинстве случаев ***модели схемного типа*** представляют в виде ***конечных автоматов***

$$\text{КА} = \{\mathbf{V_0}, \mathbf{U}, \mathbf{Y}, \mathbf{V}, \varphi, \Psi\}$$

и называют ***автоматными моделями.*** Здесь $\mathbf{V}_0$ — исходное состояние автомата; **U, Y, V** — векторы входных, выходных и внутренних переменных, принимающих значения в конечных множествах; **φ** — функция переходов; **Ψ** — функция выходов.

Функция переходов задает преобразование наборов входных и внутренних переменных на предыдущем такте в набор внутренних переменных на последующем такте:

$$\mathbf{V}(t + t_3) = \varphi(\mathbf{V}(t), \mathbf{U}(t)),$$

где $t_з$ — задержка, а функция выходов задает преобразование векторов **V***(t)* и **U***(t)* в вектор **Y**(*t*).

Функции **φ** и **Ψ** могут выражаться с помощью формул, таблиц (матриц), описаний на входных языках прикладных программ. Задержки $t_з$ могут быть различными для разных путей прохождения сигналов от входов к выходам и в общем случае функционального узла с *n* входами и *m* выходами задаются в виде матрицы $[\tau_{ij}]$, i=1, 2,…,n, j=1, 2,…, m, где τ_{ij} — задержка распространения сигнала от i-го входа к j-му выходу.

В отдельных случаях модель функционального узла может быть представлена в виде алгоритма, в котором действия выполняются над переменными **U** и **Y** вещественного типа. В таком виде удобно представлять сложные устройства, например арифметико-логические, выполняющие действия над числами с плавающей запятой.

Пример модели функционального узла

Y: = **X**1 * **X**2,

где **Y** — идентификатор выхода; **X**I и **X**2 — идентификаторы входов;

* — символ выполняемой в функциональном узле операции (алгоритма) над операндами **X**1 и **X**2. Переменные **Y**, **X**1 и **X**2 могут быть скалярами или векторами булевого или вещественного типа. Подобным образом описываются шифраторы, сумматоры, схемы контроля четности и т. п.

6.12. Консультируемые проблемы – как системы массового обслуживания

6.12.1. Предмет теории массового обслуживания

Одним из математических методов исследования стохастических сложных консультируемых проблем является теория массового обслуживания, занимающаяся ***анализом эффективности функционирования*** так называемых ***систем массового обслуживания***. Работа любой такой системы заключается в обслуживании поступающего на нее потока требований, или заявок. Заявки поступают на систему одна за другой в некоторые, вообще говоря, случайные моменты времени. Обслуживание поступившей заявки продолжается какое-то время, после чего система освобождается для обслуживания очередной заявки. Каждая такая система может состоять из нескольких независимо функционирующих единиц, которые называют каналами обслуживания, или обслуживающими аппаратами. Примерами таких систем могут быть: телефонные станции, билетные кассы, аэродромы, вычислительные центры, радиолокационные станции и т. д. Типичной системой массового

обслуживания является автоматизированная система управления производством.

Математический аппарат теории массового обслуживания позволяет оценить эффективность обслуживания системой заданного потока заявок в зависимости от характеристик этого потока, числа каналов системы и производительности каждого из каналов.

В качестпе критерия эффективности системы обслуживания могут быть использованы различные величины и функции, например: вероятность обслуживания каждой из поступающих заявок, средняя доля обслуженных заявок, среднее время ожидания обслуживания, среднее время простоя каждого из каналов и системы в целом, закон распределения длины очереди, пропускная способность системы и т. д. Численное значение каждого из этих критериев в той или иной степени характеризует степень приспособленности системы к выполнению поставленной перед ней задачи — удовлетворение потока поступающих в систему требований.

Часто термин «пропускная способность» используется в следующем узком смысле: среднее число заявок, которое система может обслужить в единицу времени. Эффективность систем обслуживания может быть оценена также величиной относительной пропускной способности—средним отношением числа обслуженных заявок к числу поступивших.

В силу случайного характера моментов поступления заявок процесс их обслуживания представляет собой случайный процесс. Теория массового обслуживания позволяет получить математическое описание этого процесса, изучение которого дает возможность оценить пропускную способность системы и сформировать рекомендации по рациональной организации обслуживания.

Все системы массового обслуживания имеют вполне определенную структуру, схематически изображенную на рис. 6.30.

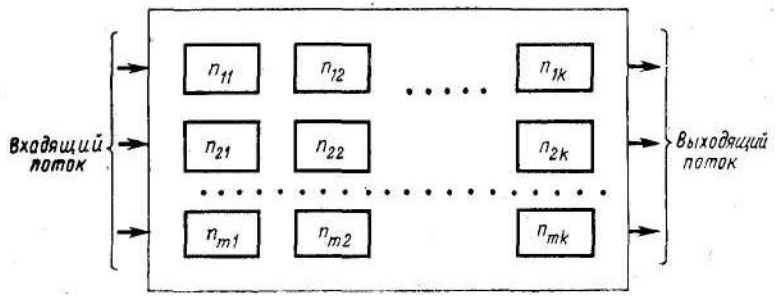


Рис. 6.30.

В соответствии с рисунком в любой системе массового обслуживания будем различать следующие основные элементы: входящий поток, выходящий поток, собственно система обслуживания.

Поток требований, нуждающихся в обслуживании и поступающих в систему обслуживания, называется ***входящим.*** Поток требований, покидающих систему обслуживания, называется ***выходящим.*** Совокупность обслуживающих аппаратов вместе с системой правил, устанавливающих организацию обслуживания, образуют ***систему обслуживания.***

6.12.2. Входящий поток. Простейший поток и его свойства

События, образующие входяший поток, вообще говоря, могут быть различными, но здесь будет рассматриваться лишь однородный поток событий, отличающихся друг от друга только моментами появления. Такой поток можно представить в виде последовательности точек t_1, t_2, ..., t_k, ... на числовой оси (рис. 6.31), соответствующих моментам появления событий.

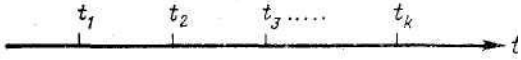


Рис. 6.31.

Поток событий называется ***регулярным,*** если события следуют одно за другим через строго определенные промежутки времени. Такие потоки редко встречаются в реальных консультируемых проблемах, для которых типичным является именно случайность моментов поступления требований. Рассмотрим случайный входящий поток, обладающий особенно простыми свойствами.

Введем ряд определений.

1. Поток событий (проблем) называется ***стационарным,*** если вероятность поступления заданного числа событий (проблем) в течение интервала времени фиксированной длины зависит только от продолжительности этого интервала, но не зависит от его расположения на временной оси.

2. Поток событий (проблем) называется ***ординарным,*** если вероятность появления двух или более событий (проблем) в течение элементарного интервала времени Δt есть величина бесконечно малая по сравнению с вероятностью появления одного события на этом интервале.

3. Поток событий (проблем) называется ***потоком без последействия,*** если для любых неперекрывающихся интервалов

времени число событий (проблем), попадающих на один из них, не зависит от числа событий (проблем), попадающих на другие.

Если поток событий (проблем) ***удовлетворяет*** всем трем перечисленным условиям (т. е. он стационарен, ординарен и не имеет последействия), то он называется ***простейшим потоком.*** Для простейшего потока число событий (проблем), попадающих на любой фиксированный интервал времени, распределено но закону Пуассона, поэтому его иначе называют ***стационарным пуассоновским***

Условию стационарности удовлетворяет поток заявок, вероятностные характеристики которого не зависят от времени. В частности, постоянной является ***плотность потока— среднее число, заявок в единицу времени***. Заметим, что свойство стационарности выполняется, по крайней мере на ограниченном отрезке времени, для многих реальных процессов.

Условие ординарности означает, что заявки поступают в систему поодиночке, а не парами, тройками и т. д. Например, поток обстрелом, которому подвергается воздушная цель в зоне действия комплекса ЗРВ, является ординарным, если стрельба ведется одиночными ракетами, и не является ординарным, если стрельба идет одновременно двумя или тремя ракетами.

Условие отсутствия последействия является наиболее существенным для простейшего потока. Выполнение этого условия означает, что заявки поступают в систему независимо друг от друга. Например, можно сказать, что ***последействие отсутствует*** для потока пассажиров, входящих в метро, так как ***отсутствует зависимость между причинами***, вызвавшими приход каждого из пассажиров на станцию. Но как только эта зависимость появляется, условие отсутствия последействия нарушается. Например, поток пассажиров, покидающих станцию метро, уже не обладает свойством последействия, так как моменты выхода для пассажиров, прибывших на станцию одним и тем же поездом, зависимы между собой.

Вообще следует заметить, что выходящие потоки заявок, покидающих систему обслуживания, обычно имеют последействие, даже если входящий поток его не имеет. В этом легко убедиться на примере рассмотрения выходящего потока для одноканальной системы массового обслуживания с фиксированным временем обслуживания $t_{об}$. Выходящий поток такой системы обладает тем свойством, что минимальный интервал между последовательными обслуженными заявками будет равен $t_{об}$. При этом, если в некоторый момент t_1 систему покинула заявка, то можно утверждать, что на интервале

(t_1, $t_1 + t_{об}$) обслуженных заявок больше не появится и, таким образом, имеется зависимость между числом событий на неперекрывающихся интервалах.

Отметим, что, если на систему обслуживания поступает самый простой, на первый взгляд, регулярный поток, анализ процессов функционирования системы является существенно более сложным, чем, например, при поступлении простейшего потока, именно вследствие жесткой функциональной зависимости, которая имеет место для заявок регулярного потока.

В дальнейшем будет рассматриваться только простейший входящий поток в силу особой его роли в общей теории консалтинга.

Дело в том, что простейшие или близкие к простейшим потоки заявок часто встречаются в практике консультирования проблем. Кроме того, при анализе систем консультационного обслуживания во многих случаях можно получить вполне удовлетворительные результаты, заменяя входящий поток проблем любой структуры простейшим с той же плотностью. Наконец, важное свойство простейшего потока проблем состоит в том, что при суммировании большого числа ординарных, стационарных потоков проблем с практически любым последействием получается поток проблем, сколь угодно близкий к простейшему. Условия, которые должны при этом соблюдаться, аналогичны условиям центральной предельной теоремы: *складываемые потоки проблем должны оказывать на сумму равномерно малое влияние*.

Получим аналитическое описание простейшего потока проблем и рассмотрим его свойства подробнее.

Рассмотрим на оси 0*t* простейший поток проблем *П* (рис. 6.32) как неограниченную последовательность случайных точек.

Рис. 6.32.

Выделим произвольный интервал времени длиной τ. Как уже отмечалось, если поток проблем является простейшим, то число проблем, попадающих на интервал τ, распределено по закону Пуассона с математическим ожиданием

$$a = \lambda\tau,$$

где λ — плотность потока.

В соответствии с законом Пуассона вероятность того, что за время τ насступит (поступит) ровно *m* проблем, равна

$$P_m(\tau) = \frac{(\lambda\tau)^m}{m!} e^{-\lambda\tau}, \quad m = 0,\ 1,\ 2, \ldots \tag{6.95}$$

Тогда вероятность того, что не наступит (поступит) ни одной проблемы, будет

$$P_0(\tau) = e^{-\lambda\tau}. \tag{6.96}$$

Отсюда вероятность того, что за время τ наступит (поступит) хотя бы одна проблема, равна

$$P_{\geq 1}(\tau) = 1 - P_0(\tau) = 1 - e^{-\lambda\tau}. \tag{6.97}$$

Введом величину $a = \lambda\tau$ (среднее число проблем, происходящих за время τ). При этом формул» (6.95) перепишется в виде

$$P_m(\tau) = \frac{a^m}{m!} e^{-a}.$$

С ростом *a* распределение Пуассона асимптотически нормально со средним и дисперсией, равными *a.* Это значит, что если *Z(a)* — случайная величина, распределенная по закону Пуассона со средним значением *a,* то для больших *a*

$$P\{Z(a) \leqslant m\} \approx \frac{1}{\sqrt{2\pi a}} \int_{-\infty}^{m+1/2} \exp\left\{-\frac{(t-a)^2}{2a}\right\} dt = \Phi\left(\frac{m - a + 1/2}{\sqrt{a}}\right),$$

где

$$\Phi(x) = \frac{2}{\sqrt{2\pi}} \int_0^x \exp\left[-\frac{t^2}{2}\right] dt.$$

Важной характеристикой потока является закон распределения длин интервалов между проблеми. Пусть *T* — случайная длина интервала времени между двумя произвольными соседними проблеми в простейшем потоке (рис. 6.37) и *F(t)=P(T<t)*—искомый закон распределения продолжительности временного интервала между последовательными проблеми. С другой стороны, вероятность *P(T< t)* может быть интерпретирована как вероятность появления хотя бы одной проблемы в течение временного интервала продолжительностью *t,* начинающегося в момент поступления в систему некоторой проблемы.

Поскольку простейший поток не обладает последействием, наличие проблемы в начале интервала *t* не оказывает никакого влияния на вероятность появления проблем в дальнейшем. Поэтому вероятность *P(T<t)* может быть вычислена по формуле

$$P(T<t) = 1 - P(T \geq t) = 1 - P_0(t), \tag{6.98}$$

откуда, имея в виду (6.96),

$$P(T<t) = F(t)=1-e^{-\lambda t} \quad (t>0). \qquad (6.99)$$

Дифференцируя (6.99), находим плотность распределения длин интервалов между последовательными проблемами

$$f(t) = \lambda e^{-\lambda t} \quad (t>0). \qquad (6.100)$$

Закон распределения с плотностью (6.100) называется *показательным с параметром* λ.

Найдем математическое ожидание и дисперсию величины T, распределенной по показательному закону (6.100),

$$M[T] = \int_0^\infty t f(t)\,dt = \lambda \int_0^\infty t e^{-\lambda t} dt = \frac{1}{\lambda},$$

$$D[T] = M[(T - M[T])^2] = \int_0^\infty t^2 f(t)\,dt - \frac{1}{\lambda^2} =$$

$$= \lambda \int_0^\infty t^2 e^{-\lambda t} dt - \frac{1}{\lambda^2} = \frac{1}{\lambda^2}. \qquad (6.101)$$

Выявим одно весьма важное свойство показательного закона, состоящее в следующем. Если промежуток времени, распределенный по показательному закону, уже длился некоторое время τ, то это никак не влияет на закон распределения оставшейся части промежутка: он будет таким же, как и закон распределения всего промежутка T. Для этого рассмотрим случайный промежуток времени T с функцией распределения

$$F(t)=1-e^{-\lambda t}. \qquad (6.102)$$

Предположим, что этот промежуток уже длился некоторое время τ, т. е. $T > \tau$. Найдем условный закон распределения оставшейся части промежутка $T_1 = T - \tau$, обозначив его через $F_\tau(t)$,

$$F_\tau(t) = P(T - \tau < t / T > \tau).$$

По теореме умножения вероятностей

$$P((T>\tau)(T-\tau<t)) = P(T>\tau)\,P(T-\tau<t/T>\tau) =$$

$$= P(T>\tau)\,F_\tau(t).$$

Отсюда

$$F_\tau(t) = \frac{P(T>\tau,\ T-\tau<t)}{P(T>\tau)}.$$

Но событие $(T > \tau) \wedge (T - \tau < t)$ равносильно событию $\tau < T < t + \tau$, вероятность которого равна

$$P(\tau<T<t+\tau) = F(t+\tau) - F(\tau).$$

С другой стороны,

$$P(T > \tau) = 1 - F(\tau),$$

следовательно,

$$F_\tau(t) = \frac{F(t+\tau) - F(\tau)}{1 - F(\tau)}$$

Отсюда, используя (6.102), имеем

$$F(t) = (e^{-\lambda\tau} - e^{-\lambda(t+\tau)}/e^{-\lambda\tau}) = 1 - e^{-\lambda t} = F(t), \quad (6.103)$$

что и требовалось.

Таким образом, показано, что если промежуток времени распределен по показательному закону, то любая информация о его протяженности не влияет на закон распределения оставшегося времени. Доказано, что показательный закон является единственным, обладающим этим свойством. Свойство отсутствия последействия, присущее простейшему потоку, позволяет использовать для его анализа аппарат марковских цепей.

Введем состояния системы консультирования следующим образом: будем считать, что система консультирования находится в состоянии E_s в момент времени t, если к этому моменту в систему консультирования поступило s требований (проблем). Вычислим w_{ss} вероятность того, что в момент времени $t+dt$ система консультирования останется в том же состоянии. Ясно, что этому соответствует ситуация, когда за интервал dt в систему консультирования не поступит ни одного требования (проблемы). В соответствии с (6.96) эта вероятность равна

$$w_{ss} = P_0(dt) = e^{-\lambda dt}, \quad s = 0, 1, 2, \ldots$$

Разлагая $e^{-\lambda dt}$ в ряд, имеем

$$w_{ss} = 1 - \lambda dt + \frac{1}{2!}(\lambda dt)^2 - \ldots = 1 - \lambda dt + o(dt) = 1 - \lambda dt, \quad s = 0, 1, 2, \ldots \quad (6.104)$$

Вероятность поступления в систему консультирования хотя бы одного требования (проблемы) за интервал dt в соответствии с (6.97) равна

$$P_{\geq 1}(dt) = 1 - P_0(dt) = \lambda dt.$$

Учитывая свойство ординарности простейшего потока, получим

$$w_{s,\,s+1} = P_{\geq 1}(dt) = \lambda dt, \quad s = 0, 1, 2, \ldots,$$

$$w_{s,\,m} = 0, \quad m = s+2, s+3, \ldots; \quad s = 0, 1, 2, \ldots \quad (6.105)$$

Ввиду стационарности простейшего потока, можно считать, что рассчитанные по формулам (6.104) и (6.105) вероятности перехода для момента времени t имеют то же значение и для любого другого момента времени. Тогда матрица переходов для простейшего потока приобретает вид рис. 6.33.

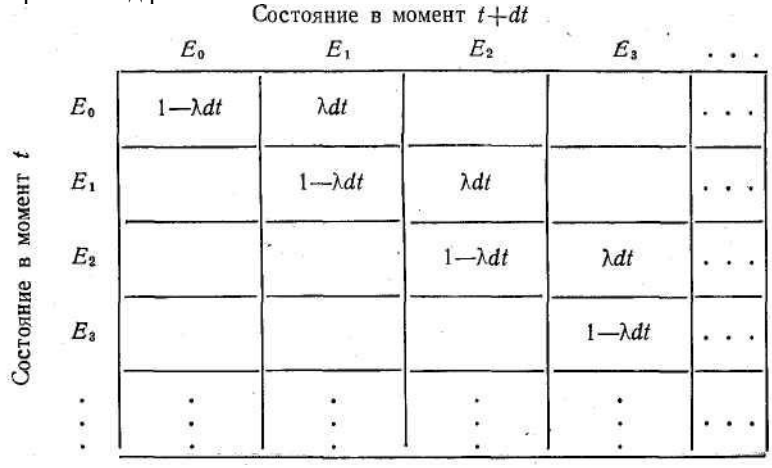


Рис. 6.33.

Соответствующий полученной матрице переходов граф изображен на рис. 6.34.

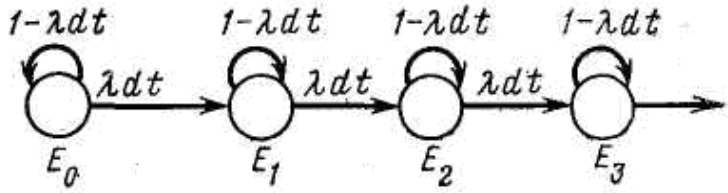


Рис. 6.34.

6.12.3. Нестационарный пуассоновский поток

Если вероятность попадания на интервал фиксированной длины τ заданного числа событий зависит не только от длины этого интервала, но и от его расположения на временной оси, то входящий поток является нестационарным. Естественно, что в этом случае ***плотность потока*** λ — ***среднее число требований****, поступающих в систему в единицу времени,* уже не является постоянной величиной. В связи с этим характеристикой нестационарного потока является ***мгновенная***

плотность $\lambda(t)$. Мгновенной плотностью потока называется предел отношения среднего числа событий, приходящихся на элементарный интервал времени $(t,\ t+\Delta t)$, к длине интервала, когда последняя стремится к нулю, т. е.

$$\lambda(t) = \lim_{\Delta t \to 0} \frac{m(t+\Delta t) - m(t)}{\Delta t},$$

где $m(t)$ - математическое ожидание числа событий на интервале $(0, t)$.

Пусть в систему обслуживания поступает ординарный и без последействия, но нестационарный поток однородных событий с переменной плотностью $\lambda(t)$. Такой поток называется ***нестационарным пуассоновским потоком.***

Показано, что для такого потока число событий, попадающих на временной интервал длиной τ, начинающийся в момент t_0, распределено по закону Пуассона:

$$P_m(\tau,\ t_0) = \frac{\mu^m}{m!}\, e^{-\mu(t_0,\ \tau)},\quad m = 0,\ 1,\ 2,\ \ldots, \tag{6.106}$$

где $\mu(t_0,\ \tau)$ —математическое ожидание числа событий на интервале $(t_0,\ t_0+\tau)$, равное

$$\mu(t_0,\ \tau) = \int_{t_0}^{t_0+\tau} \lambda(t)\, dt.$$

Здесь величина $\mu(t_0,\ \tau)$ зависит и от длины интервала и от его положения на оси $0t$.

Аналогично тому, как это было сделано для простейшего потока, найдем закон распределения промежутка времени T между соседними событиями. Предположим, что первое из двух последовательных событий появилось в момент t_0. При этом условии закон распределения времени T между поступившим и следующим за ним событиями $F_{t_0}(t)$ запишем в виде

$$F_{t_0}(t) = P(T < t) = 1 - P(T \geq t).$$

Вероятность того, что на интервале $(t_0,\ t_0+\ t)$ не появится ни одного события, равна вероятности выполнения неравенства $T \geq t$ и вычисляется по формуле

$$P(T \geq t) = P_{0,\ t_0}(t) = \exp\left[-\mu(t_0,\ t)\right] = \exp\left[-\int_{t_0}^{t_0+t} \lambda(t)\, dt\right],$$

откуда

$$F_{t_0}(t) = 1 - \exp\left[-\int_{t_0}^{t_0+t} \lambda(t)\, dt\right]. \tag{6.107}$$

Дифференцируя (6.107), получаем плотность распределения

$$f_{t_0}(t) = \lambda(t_0 + t) \exp\left[-\int_{t_0}^{t_0+t} \lambda(t)\,dt\right], \quad t > 0. \tag{6.108}$$

Этот закон распределения уже не является показательным. Его характер зависит от вида функции $\lambda(t)$ и от момента t_0. Например, при линейном изменении $\lambda(t)$:

$$\lambda(t)=a+bt$$

плотность (6.108) имеет вид

$$f_{t_0}(t) = [a + b(t_0 + t)] \exp\left[-at - bt_0 t - \frac{bt^2}{2}\right].$$

6.12.4. Поток с ограниченным последействием (поток Пальма)

Рассмотренный в предыдущем пункте нестационарный пуассоновский поток является естественным обобщением простейшего потока. Обобщением этого простейшего потока в другом направлении является ***поток с ограниченным последействием***.

Рассмотрим ординарный поток однородных событий. Этот поток называется *потоком с ограниченным последействием* (потоком Пальма), если ***промежутки времени*** между последовательными событиями ***независимы***. Ясно, что простейший поток является частным случаем потока Пальма, когда независимые промежутки времени между последовательными событиями распределены по показательному закону.

Заметим, что нестационарный пуассоновский поток не является потоком Пальма, ибо, как уже было указано, для нестационарного пуассоновского потока закон распределения длины интервала между двумя последовательными событиями зависит от положения начала этого интервала на оси *t*.

Поскольку начало этого интервала совпадает с концом предыдущего интервала, между ними в силу вышесказанного имеется зависимость. Поток заявок, покидающих какую-либо систему массового обслуживания, может быть разбит на два: поток обслуженных и поток необслуженных заявок. Относительно характера потока необслуженных заявок справедлива следующая теорема Пальма, которая здесь приводится без доказательства.

Пусть на систему массового обслуживания поступает поток заявок типа Пальма, причем заявка, заставшая все каналы занятыми, получает отказ (не обслуживается). Если при этом время

обслуживания имеет показательный закон распределения, то поток необслуженных заявок является также потоком типа Пальма.

В частности, если входящий поток является простейшим, то поток необслуженных заявок, уже не являясь простейшим, все же будет иметь ограниченное последействие.

Примером потоков с ограниченным последействием являются так называемые *потоки Эрланга,* образуемые «просеиванием» простейшего потока. Пусть на вход системы обслужиания поступает простейший поток требований. Если теперь исключить из потока каждое второе требование, то оставшиеся требования образуют поток, называемый *потоком Эрланга первого порядка. Поток Эрланга второго порядка* получится, если сохранить в простейшем потоке каждое третье требование. Вообще, *потоком Эрланга k-го порядка называется поток,* получаемый из простейшего, если сохранить каждое (k +1)-е требование, исключив остальные. С точки зрения этого определения простейший поток представляет собой поток Эрланга *нулевого порядка*.

Введем случайную величину T_k, равную интервалу между соседними событиями в потоке Эрланга k -го порядка. Показано, что плотность распределения $f_k(t)$ величины T_k имеет вид

$$f_k(t) = \frac{\lambda(\lambda t)^k}{k!} e^{-\lambda t}, \quad t > 0. \tag{6.109}$$

Закон распределения с плотностью (6.109) называется законом Эрланга k -го порядка. Очевидно, что при k =0 он превращается в показательный

$$f_0(t) = \lambda e^{-\lambda t}, \quad t > 0. \tag{6.110}$$

Поток Эрланга любого порядка представляет собой поток Пальма, поскольку из независимости двух соседних промежутков между требованиями в простейшем потоке с неизбежностью следует независимость сумм этих промежутков для любого числа слагаемых в потоке Эрланга соответствующего порядка.

6.12.5. Время обслуживания

Как уже отмечалось, эффективность консультируемой системы зависит не только от характеристик входящего потока проблем, но и от производительности самой консультируемой системы, т. е. от числа каналов и быстродействия каждого из них. В связи с этим *время обслуживания одной проблемы* $T_{об}$ является важной характеристикой консультируемой системы. В силу самых различных причин время обслуживания в реальных консультируемых системах может меняться от одной проблемы к другой. Поэтому в общем случае следует считать время обслуживания случайной величиной.

Введем закон распределения времени обслуживания

$$G(t)=P(T_{об}<t), \quad t>0$$

и плотность его распределения

$$g(t)=G'(t).$$

Для консультационной практики особый интерес представляет случай, когда продолжительность времени обслуживания имеет ***показательный закон распределения***, т. е.

$$G(t) = 1 - e^{-\mu t} \qquad (6.111)$$

Параметр μ, входящий в (6.111), имеет простой физический смысл. Величина, обратная 6.111, равна математическому ожиданию времени обслуживания. Действительно,

$$M[T_{об}] = \int_0^\infty t dG(t) = te^{-\mu t}\Big|_0^\infty + \int_0^\infty e^{-\mu t} dt = \frac{1}{\mu}.$$

Важная роль, которую играет показательный закон времени обслуживания связана с уже упоминавшимся свойством этого закона. Применительно к данному случаю оно формулируется следующим образом: *если в какой-то момент происходит обслуживание требования, то закон распределения оставшегося времени обслуживания не зависит от того, сколько времени обслуживание уже продолжалось.*

Таким образом, процесс обслуживания заявок не обладает последействием и поэтому для его анализа может быть использован аппарат теории марковских процессов.

Показательный закон распределения времени обслуживания имеет место во многих консультационных задачах, когда обслуживание сводится к последовательности попыток, каждая из которых приводит к необходимому результату с некоторой вероятностью.

Примером такого обслуживания является обстрел цели, заканчивающийся после поражения цели. Предположим, что последовательность выстрелов, каждый из которых поражает цель с вероятностью *p,* образует простейший поток с плотностью λ.

Из этого потока выделим поток успешных выстрелов (выстрел будем называть успешным, если имеет место попадание в цель). Поскольку каждый из выстрелов независимо от других может оказаться успешным, поток успешных выстрелов так же, как и исходный, будет простейшим с плотностью $\Lambda = \lambda p$.

Закон распределения интервала времени между попаданиями имеет вид

$$G(t) = P(T_{об}<t)=1-e^{-\Delta t},$$

откуда плотность распределения времени обслуживания

$$g(t) = \Lambda e^{-\Delta t},$$

что соответствует показательному закону с параметром Λ.

Количество примеров реальных консультируемых проблем, в которых обслуживание сводится к последовательности попыток, можно значительно увеличить. К такому типу можно отнести формирование рекомендаций на обслуживание по устранению неисправностей технических устройств, когда поиск неисправного элемента ведется путем использования ряда тестов. Совершенно аналогичной является консультационная задача, заключающаяся в формировании рекомендаций по обнаружению воздушной цели радиолокатором, многократно зондирующим исследуемое пространство, причем цель может с некоторой вероятностью обнаруживаться в каждом из циклов обзора.

Поскольку показательный закон распределения вполне приемлемым образом соответствует большому количеству реальных консультируемых проблем обслуживания, а также в связи с тем, что основные характеристики консультируемых проблем обслуживания зависят, главным образом, не от вида закона распределения, а от среднего значения времени обслуживания, в практических исследованиях обычно используется допущение о показательности закона распределения времени обслуживания. Важно также, что эта гипотеза позволяет существенно упростить математический аппарат, применяемый для анализа консультируемых проблем, использующих математический аппарат теории систем массового обслуживания.

6.12.6. Основные типы систем массового обслуживания и показатели эффективности их функционирования

Важным признаком классификации систем массового обслуживания является поведение поступившего в систему требования (проблемы) в ситуации, когда все обслуживающие аппараты заняты. При этом в одних случаях требование не может ждать момента освобождения системы обслуживания и покидает ее необслуженным. Требование, поступившее в систему обслуживания и получившее отказ, потеряно для системы. Поэтому такие системы обслуживания называют *системами с отказами* или *системами с потерями.*

В других случаях требование может более или менее долго ожидать начала обслуживания, т. е. момента освобождения одного из обслуживающих аппаратов системы. Совокупность таких требований образует очередь. Если при этом время ожидания для каждого из

требований не ограничено, система обслуживания называется *чистой системой с ожиданием* или *системой без потерь.* В противном случае, когда это время ограничено какими-либо условиями, систему называют *системой обслуживания смешанного типа.* Характер ограничений в системах смешанного типа может быть различным. Во многих случаях ограничение накладывается на *продолжительность ожидания в очереди,* т. е. каждое из поступивших требований покидает систему, если обслуживание не началось до определенного момента времени, однако начатое обслуживание доводится до конца. В других случаях более естественным является наложить ограничение сверху *на общее время пребывания требования в системе.* Наконец, ограничение может быть наложено на *длину очереди,* т. е. требование становится в очередь и ожидает обслуживания только в том случае, если длина очереди (число ожидающих требований) не слишком велика.

Естественным критерием эффективности системы обслуживания с отказами является вероятность отказа в обслуживании (вероятность потери требования). Так как отказ происходит только в том случае, когда все обслуживающие аппараты заняты, соответствующие вероятности равны между собой.

Степень загрузки системы обслуживания с отказами *характеризует закон распределения числа занятых аппаратов.* Во многих случаях для характеристики эффективности системы обслуживания с отказами достаточно указать *среднее число занятых аппаратов.*

В системе обслуживания без потерь требование находится до тех пор, пока не будет закончено его обслуживание. Исходя из этого, могут быть сформулированы основные критерии эффективности функционирования таких систем. Это, прежде всего, *длина очереди.* Поскольку число требований, ожидающих начала обслуживания в очереди, случайно, наиболее полной характеристикой этой величины является закон ее распределения. Знание этого закона позволяет рассчитать среднее число требований, ожидающих обслуживания, вероятность того, что длина очереди превысит заданную и т.д. Другим важным критерием для оценки эффективности таких систем является *время ожидания начала обслуживания,* наиболее полно характеризуемое своим законом распределения. С использованием этого закона может быть вычислено среднее значение времени ожидания, вероятность того, что обслуживание будет начато в течение некоторого заданного интервала времени и т. п. Наконец, характеристикой таких систем является закон распределения числа аппаратов, занятых обслуживанием, позволяющий рассчитать среднее

число занятых аппаратов, вероятность занятости числа аппаратов, превышающее заданное, и т. п.

Для оценки эффективности систем обслуживания смешанного типа могут быть использованы все перечисленные выше критерии. Кроме них, используются и некоторые специфические критерии. Например, для системы, в которой ограничено общее время пребывания требования в системе, определенный интерес представляет расчет времени, затраченного на обслуживание требований, которые покидают систему *до момента окончания их обслужившим.* Если частичное обслуживание не обеспечивает решения задачи обслуживания, то имеют место непроизводительные потери, учет которых характеризует эффективность системы.

Все перечисленные критерии в той или иной степени информативно характеризуют приспособленность рассматриваемой системы для выполнения поставленных перед ней задач. Анализ численных значений критериев позволяет сделать выводы относительно реальной эффективности системы и выработать рекомендации по ее повышению.

6.12.7. Система массового обслуживания с отказами

Пусть имеется *n*-канальная система массового обслуживания с отказами. Представим ее в виде некоторой системы консультирования с конечным множеством состояний:

E_0— свободны все каналы,

E_1 — занят ровно один канал,

….....

E_k — занято ровно *k* каналов,

……

E_n — заняты все *n* каналов.

Проведем оценку эффективности такой системы при следующих допущениях:

1) поток заявок — простейший, с плотностью λ*;*
2) время обслуживания $T_{об}$ — показательное, с параметром

$$\mu = \frac{1}{M[T_{об}]}.$$

Понятно, что параметры λ и μ по своему смыслу аналогичны. Действительно, если λ есть среднее число требований, поступающих в систему в единицу времени, то μ — среднее число требований, которое система в единицу времени в состоянии обслужить.

Важно отметить, что при выполнении принятых допущений процесс перехода рассматриваемой системы из одного состояния в другое является *марковским.*

Действительно, пусть система находится в состоянии E_s (в некоторый момент времени t занято s аппаратов). Моменты поступления новых заявок не зависят от того, что было до момента t, так как поток требований, по предположению, простейший. Моменты освобождения занятых аппаратов также не зависят от прошлого системы до момента t в силу показательности времени обслуживания. Таким образом, моменты переходов системы в новое состояние зависят только от текущего состояния системы, но не от того, как система пришла в это состояние, т. е. система обладает марковским свойством.

В связи с этим для ***оценки эффективности*** такой системы массового обслуживания может быть использован ***аппарат теории марковских процессов.*** Заметим, что все характеристики эффективности системы массового обслуживания (вероятность отказа, вероятность обслуживания, среднее число занятых каналов и т. д.) так или иначе определяются при использовании закона распределения вероятностей состояний системы в установившемся стационарном режиме. Вместе с тем показано, что для любой системы массового обслуживания такой режим устанавливается (т. е. система облагает эргодическим свойством) в том и только в том случае, если выполняется следующее условие:

$$\lambda/\mu<n. \qquad (6.112)$$

При выполнении условия (6.112) предельный вектор существует и может быть рассчитан с использованием элементов стохастической матрицы системы **W.** Следует обратить внимание на то, что вопрос о наличии предельного вектора не возникает при анализе системы с отказами, так как число возможных состояний в этой системе конечно и каждое из них, не являясь периодическим, достижимо из любого другого. Этого достаточно для того, чтобы система была эргодической. В системах массового обслуживания смешанного типа, а также в системах без потерь выполнение условия (6.112) для существования предельного вектора является необходимым и достаточным.

Рассчитаем предельный вектор системы с отказами. Пусть $\mathbf{P} = (P_0\ P_1\ P_2 \ldots P_n)$—вектор вероятностей различных состояний системы в установившемся режиме. Для отыскания компонент вектора используем векторно-матричное уравнение

$$\mathbf{P= PW.} \qquad (6.113)$$

Вычислим элементы матрицы переходов системы **W.** Расчет проведем по столбцам матрицы последовательно, начиная с нулевого.

$w_{00}(\Delta t)$ есть вероятность того, что за время Δt система, свободная к моменту начала интервала Δt, не будет занята к концу этого интервала.

Пусть $P_s(\Delta t)$ —вероятность поступления в систему s требований в течение Δt, а $P'_s(\Delta t)$ —вероятность обслуживания s требований в течение Δt. Тогда, строго говоря,

$$w_{00}(\Delta t) = P_0(\Delta t) + P_1(\Delta t)P'_1(\Delta t)+P_2(\Delta t)P'_2(\Delta t) + \ldots \quad (6.114)$$

Однако в соответствии с (6.95)

$$P_1(\Delta t) = (\lambda \Delta t)\, e^{-\lambda \Delta t} = \\ = (\lambda \Delta t)\left[1 - \lambda \Delta t + \frac{1}{2}(\lambda \Delta t)^2 - \ldots\right] = \lambda \Delta t + o(\Delta t) \quad (6.115)$$

и, с другой стороны, в силу (6.111)

$$P'_1(\Delta t) = 1 - e^{-\mu \Delta t} = \\ = 1 - \left[1 - \mu \Delta t + \frac{1}{2}(\mu \Delta t)^2 - \ldots\right] = \mu \Delta t + o(\Delta t). \quad (6.116)$$

Поэтому, перемножая (6.115) и (6.116), имеем

$$P_1(\Delta t))P'_1(\Delta t)) = o(\Delta t).$$

Одновременно

$$P_s(\Delta t)P'_s(\Delta t) = o(\Delta t), \qquad s = 2, 3.....$$

так как входящий поток является простейшим и обладает ординарностью.

Тогда, пренебрегая членами, имеющими меньший порядок малости по сравнению с Δt, упростим (6.114) к виду

$$w_{00}(\Delta t) = P_0(\Delta t) = e^{-\lambda \Delta t} \approx 1 - \lambda \Delta t. \quad (6.117)$$

Далее, $w_{10}(\Delta t)$ есть вероятность того, что система, в котором к началу интервала Δt имеется один занятый канал, освободится к моменту окончания интервала. Повторяя предыдущие рассуждения, получим

$$w_{10}(\Delta t) = P_0(\Delta t)\, P'_1(\Delta t) = (1 - \lambda \Delta t)\, \mu \Delta t \approx \mu \Delta t. \quad (6.118)$$

Наконец, можно непосредственно убедиться в том, что

$$w_{s0}(\Delta t) = o(\Delta t) \text{ для всех } s=2, 3, \ldots \quad (6.119)$$

Для произвольного столбца матрицы **W** с номером k $(0<k<n)$ имеем

$$w_{k,k}(\Delta t) = P_0(\Delta t)\, P'_0(\Delta t) + P_1(\Delta t)P'_1(\Delta t) + \ldots = \\ = P_0(\Delta t)\, P'_0(\Delta t) + o(\Delta t),$$

$$w_{k-1,k}(\Delta t) = P_1(\Delta t)\, P'_0(\Delta t) + P_2(\Delta t)\, P'_1(\Delta t) + \ldots = \\ = P_1(\Delta t)\, P'_0(\Delta t) + o(\Delta t),$$

$$w_{k+1,k}(\Delta t) = P_0(\Delta t)\, P'_1(\Delta t) + P_1(\Delta t)\, P'_2(\Delta t) + \ldots = \\ = P_0(\Delta t)\, P'_1(\Delta t) + o(\Delta t).$$

Таким образом,

— переходная вероятность $w_{k,k}(\Delta t)$ равна вероятности того, что за интервал Δt не поступит ни одной новой заявки и ни один из k ранее занятых каналов не освободится;

— переходная вероятность $w_{k-1,k}(\Delta t)$ равна вероятности того, что за интервал Δt в систему поступит одна новая заявка и ни один из k теперь занятых каналов не освободится;

— переходная вероятность $w_{k+1,k}(\Delta t)$ равна вероятности того, что за интервал Δt не поступит ни одной новой заявки и какой-либо один из $k+1$ ранее занятых каналов освободится.

Поэтому

$$w_{k,k}(\Delta t) = e^{-\lambda \Delta t}\,(e^{-\mu \Delta t})^k = e^{-(\lambda + k\mu)\,\Delta t} \approx 1 - (\lambda + k\mu)\,\Delta t, \tag{6.120}$$

$$w_{k-1,k}(\Delta t) = (\lambda \Delta t)\, e^{-\lambda \Delta t}\,(e^{-\mu \Delta t})^k = \lambda \Delta t e^{-(\lambda + k\mu)\Delta t} \approx \lambda \Delta t, \tag{6.121}$$

$$w_{k+1,k}(\Delta t) = e^{-\lambda \Delta t}\,(1 - e^{-\mu \Delta t})\,(k+1) = (k+1)\mu \Delta t. \tag{6.122}$$

Кроме того, как легко убедиться,

$$w_{sk}(\Delta t) = o(\Delta t), \tag{6.123}$$

если $|s-k| \geq 2$, $s = 0, 1, 2, \ldots, n$, $k = 0, 1, 2, \ldots, n$.

Совершенно аналогично рассчитаем элементы последнего n-го столбца матрицы переходов. Имеем

$$w_{n-1,n}(\Delta t) = P_1(\Delta t)\,P_0(\Delta t) + o(\Delta t) \approx \lambda \Delta t, \tag{6.124}$$

$$w_{n,n}(\Delta t) = P'_0(\Delta t) + o(\Delta t) \approx 1 - n\mu \Delta t \tag{6.125}$$

и, как это следует из (6.123),

$$w_{sk}(\Delta t) = o(\Delta t), \text{ для } s = 0, 1, 2, \ldots n-2. \tag{6.126}$$

Заметим, что при вычислении $w_{n,n}(\Delta t)$ в отличие от предыдущего случая (когда $k \neq n$) не учитывается вероятность поступления в систему новых требований, так как это не может изменить состояния системы с отказами, если все каналы уже заняты.

Объединяя (6.117) —(6.119) и (6.122) —(6.126), получаем матрицу переходов системы (рис 6.34).

	E_0	E_1	. . .	E_{k-1}	E_k	E_{k+1}	. . .	E_{n-1}	E_n
E_0	$1-\lambda\Delta t$	$\lambda\Delta t$	. . .	0	0	0	. . .	0	0
E_1	$\mu\Delta t$	$1-(\lambda+\mu)\Delta t$	. . .	0	0	0	. . .	0	0
	. .	. .	. . .	. .	. .	. .	. . .	. .	. .
E_{k-1}	0	0	. . .	$1-(\lambda+$ $+\mu(k-1))\Delta t$	$\lambda\Delta t$	0	. . .	0	0
E_k	0	0	. . .	$k\mu\Delta t$	$1-(\mu k+\lambda)\Delta t$	$\lambda\Delta t$	. . .	0	0
E_{k+1}	0	0	. . .	0	$(k+1)\mu\Delta t$	$1-(\lambda+$ $+\mu(k+1))\Delta t$	. . .	0	0
	0	0	. . .	. .	. .	. .	. . .	. .	. .
E_{n-1}	0	0	. . .	. .	0	0	. . .	$1-(\lambda+$ $+\mu(n-1))\Delta t$	$\lambda\Delta t$
E_n	0	0	. . .	. .	0	0	. . .	$n\mu\Delta t$	$1-n\mu\Delta t$

Рис 6.34.

Соответствующий этой матрице граф изображен на рис. 6.35.

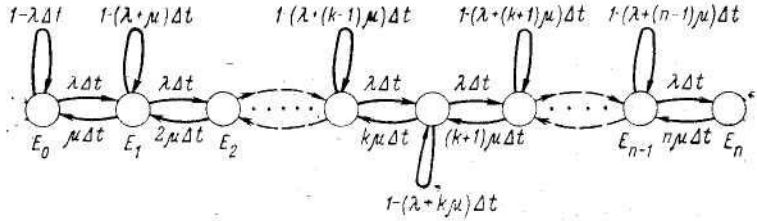


Рис. 6.35.

6.12.8. Формулы Эрланга

Подставляя полученную матрицу переходов в векторно-матричное уравнение (6.113) и преобразуя это уравнение в систему алгебраических уравнений, имеем:

$$P_0 = P_0(1-\lambda\Delta t) + P_1\mu\Delta t,$$

$$P_k = P_{k-1}\lambda\Delta t + P_k[1-(\lambda+\mu k)\Delta t] + P_{k+1}(k+1)\mu\Delta t \quad (1 \leqslant k \leqslant n-1),$$

$$P_n = P_{n-1}\lambda\Delta t + P_n(1-n\mu\Delta t). \qquad (6.127)$$

После приведения подобных членов и сокращения на Δt получим

$$
\begin{gathered}
-\lambda P_0+\mu P_1=0, \\
\lambda P_{k-1}-(\lambda+k\mu)P_k+(k+1)\mu P_{k+1}=0, \quad 0<k<n, \\
\lambda P_{n-1}-n\mu P_n=0.
\end{gathered}
\tag{6.128}
$$

Решим эту систему уравнений относительно $P_0, P_1, \ldots, P_n$, добавив условие нормировки

$$
\sum_{k=0}^{n} P_k=1.
\tag{6.129}
$$

Введем $z_k = \lambda P_{k-1} - k\mu P_k$, $k=1, 2, \ldots, n$. Тогда система уравнений (6.128) очевидным образом преобразуется к виду

$$
\begin{gathered}
z_1=0, \\
z_k-z_{k+1}=0, \quad 1\leqslant k\leqslant n-1, \\
z_n=0
\end{gathered}
\tag{6.130}
$$

и имеет решение $z_1=z_2=\ldots=z_{n-1}=z_n=0$. Отсюда

$$
P_k=\frac{\lambda}{k\mu}P_{k-1}, \quad k=1, 2, \ldots, n.
$$

Тогда

$$
\begin{aligned}
P_1&=\frac{\lambda}{\mu}P_0, \\
P_2&=\frac{\lambda}{2\mu}P_1=\frac{\lambda^2}{2\mu^2}P_0, \\
P_3&=\frac{\lambda}{3\mu}P_2=\frac{\lambda^3}{1\cdot 2\cdot 3\mu^3}P_0.
\end{aligned}
\tag{6.131}
$$

Для любого k ($k= 1, 2, \ldots, n$), таким образом, имеем

$$
P_k=\frac{P_0}{k!}\left(\frac{\lambda}{\mu}\right)^k.
\tag{6.132}
$$

Для определения P_0 используем (6.129). Подставляя (6.132) в (6.129), получаем

$$
P_0\sum_{k=0}^{n}\frac{1}{k!}\left(\frac{\lambda}{\mu}\right)^k=1 \quad \text{и} \quad P_0=\frac{1}{\sum\limits_{k=0}^{n}\frac{1}{k!}\left(\frac{\lambda}{\mu}\right)^k}.
\tag{6.133}
$$

Теперь, подставляя (6.133) в (6.132), имеем

$$P_k = \frac{\frac{1}{k!}\left(\frac{\lambda}{\mu}\right)^k}{\sum_{l=0}^{n}\frac{1}{l!}\left(\frac{\lambda}{\mu}\right)^l}, \quad k = 0, 1, 2, \ldots, n. \tag{6.134}$$

Введем параметр

$$\alpha = \lambda/\mu, \tag{6.135}$$

который назовем *приведенной плотностью потока заявок.*

Так как $1/\mu = M[T_{об}]$, то $\alpha = \lambda M[T_{об}]$ есть среднее число заявок, приходящееся на среднее время обслуживания одной заявки.

С учетом (6.135) формулы (6.134) преобразуются к окончательному виду

$$P_k = \frac{\frac{\alpha^k}{k!}}{\sum_{l=0}^{n}\frac{\alpha^l}{l!}}, \quad k = 0, 1, 2, \ldots, n. \tag{6.136}$$

Соотношения (6.136) называются *формулами Эрланга.*

Эти формулы позволяют установить предельный закон распределения числа занятых каналов в зависимости от плотности потока заявок и производительности системы обслуживания.

Полагая в формуле (6.135) $k = n$, получим *вероятность отказа:*

$$P_{отк} = P_n = \frac{\frac{\alpha^n}{n!}}{\sum_{l=0}^{n}\frac{\alpha^l}{l!}}. \tag{6.137}$$

В частности, для одноканальной системы обслуживания (n=1)

$$P_{отк} = \frac{\alpha}{(\alpha + 1)}. \tag{6.138}$$

Среднее число занятых каналов обслуживания равно

$$M[k]=\sum_{k=0}^{n} kP_k=\sum_{k=1}^{n}\frac{k\frac{\alpha^k}{k!}}{\sum_{k=0}^{n}\frac{\alpha^k}{k!}}=\frac{\sum_{k=1}^{n}\frac{\alpha^k}{(k-1)!}}{\sum_{k=0}^{n}\frac{\alpha^k}{k!}}=$$

$$=\alpha\frac{\sum_{k=1}^{n}\frac{\alpha^{k-1}}{(k-1)!}}{\sum_{k=0}^{n}\frac{\alpha^k}{k!}}=\alpha\frac{\sum_{l=0}^{n-1}\frac{\alpha^l}{l!}}{\sum_{k=0}^{n}\frac{\alpha^k}{k!}}=\alpha\frac{\sum_{l=0}^{n}\frac{\alpha^l}{l!}-\frac{\alpha^n}{n!}}{\sum_{l=0}^{n}\frac{\alpha^l}{l!}}=$$

$$=\alpha\left(1-\frac{\frac{\alpha^n}{n!}}{\sum_{k=0}^{n}\frac{\alpha^l}{l!}}\right)=\alpha(1-P_{\text{отк}}).$$

(6.139)

Заметим, что хотя формулы Эрлаига (6.136) выведены в предположении о показательности закона распределения времени обслуживания, они верны, как это показано Б. А. Севастьяновым, и при произвольном законе распределения времени обслуживания.

Пример. В четырехканальную систему массового обслуживания с отказами поступает простейший поток заявок с плотностью $\lambda = 0{,}2\ c^{-1}$. Время обслуживания $T_{об}$ распределено по показательному закону и $M[T_{об}]=10$ с. Рассчитать вероятность отказа и среднее число занятых каналов в системе.

Решение. Вычислим значение параметра α

$$\alpha = \lambda/\mu = \lambda M[T_{об}] = 0{,}2\cdot 10 = 2.$$

Используя соотношение (6.137), рассчитаем вероятность отказа

$$P_{\text{отк}}=\frac{\frac{\alpha^n}{n!}}{\sum_{l=0}^{n}\frac{\alpha^l}{l!}}=\frac{2^4}{4!\left(1+2+\frac{2^2}{2!}+\frac{2^3}{3!}+\frac{2^4}{4!}\right)}\approx 0{,}1.$$

Подставляя рассчитанное значение для вероятности отказа в (6.139), вычислим среднее число занятых каналов

$$M[k]=\alpha(1-P_{\text{отк}})=2(1-0{,}1)=1{,}8\ \text{(каналов)}.$$

6.12.9. Система массового обслуживании с ожиданием

Как уже отмечалось, система массового обслуживания называется системой с ожиданием, если заявка, заставшая все каналы занятыми, становится в очередь. В таких системах важную роль играет

так называемая «дисциплина очереди». Ожидающие в очереди заявки могут поступать на обслуживание как в порядке очереди, так и в случайном порядке. Существуют системы массового обслуживания с приоритетом, когда некоторые выделяемые по какому-либо признаку заявки обслуживаются в первую очередь.

Каждый тип системы с ожиданием имеет свои особенности и свою математическую теорию. Здесь будет рассмотрен один из самых простых вариантов смешанной системы обслуживания, часто встречающийся при консультировании различных проблем.

Пусть на вход n-канальной системы обслуживания поступает простейший поток требований с плотностью λ. Время обслуживания каждой из заявок $T_{об}$ распределено по показательному закону с параметром $\mu=1/M[T_{об}]$. Заявка, заставшая все каналы системы занятыми, становится в очередь и ожидает обслуживания. Время ожидания $T_{ож}$ будем считать случайным и распределенным по показательному закону

$$H(t)=P(T_{ож}<t)=1-e^{-\nu t} \qquad (6.140)$$

где параметр ν — величина, обратная среднему времени ожидании, т. е. $\nu=1/M[T_{ож}]$.

Благодаря допущениям о том, что входящий поток является простейшим, а распределения времени обслуживании и времени ожидания — показательные, процесс функционирования системы является марковским.

Перечислим состояния системы. Будем нумеровать их не по по числу занятых каналов, как это сделано ранее, а по числу заявок, связанных с системой. При этом будем заявку называть связанной с системы, если она либо обслуживается, либо ожидает в очереди. Возможны состояния системы:

E_0 — свободны все каналы, очереди нет,

E_1 — занят ровно один канал, очереди нет,

……..

E_k — занято ровно k каналов, очереди нет,

……..

E_n— заняты все n каналов, очереди нет,

E_{n+1} — заняты все n каналов, одна заявка стоит в очереди,

……..

E_{n+s} — заняты все n каналов, s заявок — в очереди.

Поскольку число заявок s, ожидающих обслуживания в очереди, может быть сколь угодно большим, система имеет бесконечное (хотя и счетное) число состояний.

Анализ системы с ожиданием проведем аналогично тому, как это было сделано для системы с отказами. Вычислим элементы матрицы переходов системы. Ясно, что элементы первых n столбцов матрицы, начиная с нулевого и кончая $(n—1)$-м, не будут отличаться от соответствующих элементов матрицы переходов для системы с отказами. Однако элементы уже n-го столбца имеют некоторые отличия. Дело в том, что система с отказами, как было показано, может оказаться в состоянии E_n, если предыдущим состоянием было либо E_{n-1} либо E_n. В системе с ожиданием появляется еще одна возможность — переход в E_n из E_{n+1}.

С учетом сказанного получим формулы для расчета элементов n-го столбца. В этом столбце:

$w_{n-1,n}(\Delta t)$ —вероятность того, что за интервал Δt в систему поступит одна новая заявка и ни один из $(n—1)$-го ранее занятых каналов не освободится;

$w_{n,n}(\Delta t)$ —вероятность того, что за интервал Δt не поступит ни одной новой заявки и ни один из n ранее занятых каналов не освободится;

$w_{n+1,n}(\Delta t)$ —вероятность того, что за интервал Δt не поступит ни одной новой заявки и либо освободится один из n ранее занятых каналов (при этом единственная ожидающая в очереди заявка начнет обслуживаться), либо стоящая в очереди заявка покинет систему в связи с окончанием времени ожидания.

Используя рассуждения, аналогичные проведенным в предыдущем пункте, имеем

$$\begin{gathered} w_{n-1,n}(\Delta t) = \lambda \Delta t e^{-\lambda \Delta t}(e^{-\mu \Delta t})^n \approx \lambda \Delta t, \\ w_{n,n}(\Delta t) = e^{-\lambda \Delta t}(e^{-\mu \Delta t})^n = e^{-(\lambda + n\mu)\Delta t} \approx 1 - (\lambda + n\mu)\Delta t, \\ w_{n+1,n}(\Delta t) = e^{-\lambda \Delta t}[(1 - e^{-\mu \Delta t})n + (1 - e^{-\nu \Delta t})] \approx \\ \approx (1 - \lambda \Delta t)(n\mu + \nu)\Delta t \approx (n\mu + \nu)\Delta t. \end{gathered} \quad (6.141)$$

Наконец, для произвольного $(n+s)$ -го столбца:

$w_{n+s-1,n+s}(\Delta t)$—вероятность того, что за интервал Δt в систему поступит одна заявка и ни один из ранее занятых каналов не освободится и ни одна из $(s—1)$ ранее ожидающих в очереди заявок не покинет очереди;

$w_{n+s,n+s}(\Delta t)$—вероятность того, что за интервал Δt в систему не поступит ни одной заявки и ни один из ранее занятых каналов не освободится и ни одна из s ранее ожидающих в очереди заявок не покинет очереди;

$w_{n+s+1,n+s}(\Delta t)$—вероятность того, что за интервал Δt в систему не поступит ни одной заявки и либо освободится один из n ранее занятых каналов (при этом одна из (s+1) ранее ожидающих в очереди заявок начнет обслуживаться), либо одна из (s+1)-й ожидающих обслуживания заявок покинет систему в связи с окончанием времени ожидания.

В соответствии с этим

$$w_{n+s-1,\ n+s}(\Delta t) = \lambda \Delta t e^{-\lambda \Delta t} (e^{-\mu \Delta t})^n (e^{-\nu \Delta t})^{s-1} \approx \lambda \Delta t,$$

$$w_{n+s,\ n+s}(\Delta t) = e^{-\lambda \Delta t} (e^{-\mu \Delta t})^n (e^{-\nu \Delta t})^s = e^{-(\lambda + n\mu + s\nu)\Delta t} \approx$$
$$\approx 1 - (\lambda + n\mu + s\nu)\Delta t,$$

$$w_{n+s+1,\ n+s}(\Delta t) = e^{-\lambda \Delta t}[(1 - e^{-\mu \Delta t})\, n + (1 - e^{-\nu \Delta t})(s+1)] =$$
$$= (1 - \lambda \Delta t)[n\mu + (s+1)\nu]\, \Delta t \approx [n\mu + (s+1)\nu]\, \Delta t. \qquad (6.142)$$

Используем (6.141) и (6.141) для формирования матрицы переходов. Рассчитаем теперь предельный вектор системы. Соответствующая система алгебраических уравнений после естественных упрощений имеет вид:

$$\begin{aligned}
&-\lambda P_0 + \mu P_1 = 0,\\
&\lambda P_0 - (\lambda + \mu) P_1 + 2\mu P_2 = 0,\\
&\lambda P_{k-1} - (\lambda + k\mu) P_k + (k+1)\mu P_{k+1} = 0, \quad 1 \leqslant k \leqslant n-1,\\
&\lambda P_{n-1} - (\lambda + n\mu) P_n + (n\mu + \nu) P_{n+1} = 0,\\
&\lambda P_{n+s-1} - (\lambda + n\mu + s\nu) P_{n+s} + [n\mu + (s+1)\nu] P_{n+s+1} = 0.
\end{aligned} \qquad (6.143)$$

К полученной системе уравнений необходимо добавить еще одно

$$\sum_{k=0}^{\infty} P_k = 1. \qquad (6.144)$$

Применим для решения этой системы алгебраических уравнений уже использованный ранее прием. Введем

$$z_k = \lambda P_{k-1} - k\mu P_k, \quad k = 1, 2, \ldots, n,$$

$$z_{n+s} = \lambda P_{n+s-1} - (n\mu + s\nu) P_{n+s}, \quad s = 1, 2, 3, \ldots$$

При этом система уравнений (6.143) перепишется в виде

$$
\begin{gathered}
z_1 = 0, \\
z_1 - z_2 = 0, \\
\cdots\cdots\cdots \\
z_k - z_{k+1} = 0 \quad (1 \leqslant k \leqslant n-1), \\
\cdots\cdots\cdots \\
z_n - z_{n+1} = 0, \\
\cdots\cdots\cdots \\
z_{n+s} - z_{n+s+1} = 0. \\
\cdots\cdots\cdots
\end{gathered}
$$

Отсюда

$$z_1 = z_2 = \ldots = z_n = z_{n+1} = \ldots = z_{n+s} = z_{n+s+1} = \ldots = 0.$$

Следовательно,

$$
\begin{gathered}
P_k = \frac{\lambda}{k\mu} P_{k-1}, \quad k = 1, 2, \ldots, n, \\
P_{n+s} = \frac{\lambda}{n\mu + s\nu} P_{n+s-1}, \quad s = 1, 2, 3, \ldots
\end{gathered}
\tag{6.145}
$$

Тогда

$$
\begin{gathered}
P_1 = \frac{\lambda}{\mu} P_0, \\
P_2 = \frac{\lambda}{2\mu} P_1 = \frac{1}{2!} \cdot \frac{\lambda^2}{\mu^2} P_0, \\
\cdots\cdots\cdots\cdots\cdots \\
P_k = \frac{\lambda^k}{k!\mu^k} P_0, \quad k \leqslant n, \\
\cdots\cdots\cdots\cdots \\
P_n = \frac{\lambda^n}{n!\mu^n} P_0,
\end{gathered}
\tag{6.146}
$$

$$
\begin{gathered}
P_{n+1} = \frac{\lambda}{n\mu + \nu} P_n = \frac{\lambda^{n+1} P_0}{n!\mu^n (\mu n + \nu)}, \\
P_{n+s} = \frac{\lambda^{n+s} P_0}{n!\mu^n \prod_{x=1}^{s} (n\mu + x\nu)}, \quad s = 1, 2, 3, \ldots
\end{gathered}
\tag{6.147}
$$

Заметим, что первые n формул (6.146) совпадают с формулами (6.132) для системы с отказами.

В формулы (6.146) и (6.147) в качестве сомножителя входит вероятность P_0. Определим ее из (6.144). Подставляя (6.146) и (6.147) в (6.144), получаем

$$P_0 = \left\{ \sum_{k=0}^{n} \frac{\lambda^k}{k!\,\mu^k} + \sum_{s=1}^{\infty} \frac{\lambda^{n+s}}{n!\mu^n \prod_{x=1}^{s} (n\mu + x\nu)} \right\} = 1,$$

Откуда

$$P_0 = \frac{1}{\sum_{k=0}^{n} \frac{\lambda^k}{k!\mu^k} + \sum_{s=1}^{\infty} \frac{\lambda^{n+s}}{n!\mu^n \prod_{x=1}^{s} (n\mu + x\nu)}}. \qquad (6.148)$$

Преобразуем (6.146)— (6.148), введя приведенные плотности

$$\lambda/\mu = \lambda M[T_{об}] = \alpha, \qquad \nu/\mu = \nu M[T_{об}] = \beta. \qquad (6.149)$$

Параметры α и β выражают соответственно среднее число заявок и среднее число уходов заявок, стоящих в очереди, приходящиеся на среднее время обслуживания одной заявки. В новых обозначениях формулы (6.146)— (6.148) примут вид

$$P_k = \frac{\alpha^k}{k!} P_0, \qquad 0 \leqslant k \leqslant n, \qquad (6.150)$$

$$P_{n+s} = \frac{\frac{\alpha^{n+s}}{n!} P_0}{\prod_{x=1}^{s} (n + x\beta)}, \qquad s \geqslant 1, \qquad (6.151)$$

$$P_0 = \frac{1}{\sum_{k=0}^{n} \frac{\alpha^k}{k!} + \frac{\alpha^n}{n!} \sum_{s=1}^{\infty} \frac{\alpha^s}{\prod_{x=1}^{s} (n + x\beta)}}. \qquad (6.152)$$

Подставляя (6.152) в (6.150) и (6.151), получаем окончательные выражения для вероятностей состояний системы:

$$P_k = \frac{\frac{\alpha^k}{k!}}{\sum_{l=0}^{n} \frac{\alpha^l}{l!} + \frac{\alpha^n}{n!} \sum_{s=1}^{\infty} \frac{\alpha^s}{\prod_{\varkappa=1}^{s} (n+\varkappa\beta)}}, \quad 0 \leqslant k \leqslant n, \tag{6.153}$$

$$P_{n+s} = \frac{\frac{\alpha^n}{n!} \frac{\alpha^s}{\prod_{\varkappa=1}^{s} (n+\varkappa\beta)}}{\sum_{l=0}^{n} \frac{\alpha^l}{l!} + \frac{\alpha^n}{n!} \sum_{s=1}^{\infty} \frac{\alpha^s}{\prod_{\varkappa=1}^{s} (n+\varkappa\beta)}}, \quad s \geqslant 1. \tag{6.154}$$

Зная закон распределения вероятностей состояний системы, легко теперь рассчитать другие вероятностные характеристики системы.

Найдем математическое ожидание m_s числа заявок, находящихся в очереди:

$$m_s = M[s] = \frac{\frac{\alpha^n}{n!} \sum_{s=1}^{\infty} \frac{s\alpha^s}{\prod_{\varkappa=1}^{s} (n+\varkappa\beta)}}{\sum_{k=0}^{n} \frac{\alpha^k}{k!} + \frac{\alpha^n}{n!} \sum_{s=1}^{\infty} \frac{\alpha^s}{\prod_{\varkappa=1}^{s} (n+\varkappa\beta)}}. \tag{6.155}$$

Теперь легко определить вероятность P_n того, что заявка покинет систему необслуженной. В самом деле, в установившемся режиме эта вероятность равна отношению среднего числа заявок, покидающих очередь необслуженными в единицу времени, к среднему числу заявок, поступающих в систему в единицу времени. При этом среднее число заявок, поступающих в систему в единицу времени, равно λ, — параметру входящего потока заявок. Среднее же число заявок $m_н$, покидающих систему необслуженными, можно рассчитать, зная сред-

нее число заявок, ожидающих в очереди, m_s и плотность «потока уходов» стоящих в очереди заявок ν:

$$m_н = \nu m_s = \nu \frac{\frac{\alpha^n}{n!}\sum_{s=1}^{\infty}\frac{s\alpha^s}{\prod_{x=1}^{s}(n+x\beta)}}{\sum_{k=0}^{n}\frac{\alpha^k}{k!}+\frac{\alpha^n}{n!}\sum_{s=1}^{\infty}\frac{\alpha^s}{\prod_{x=1}^{s}(n+x\beta)}}.$$

Отсюда

$$P_x = \frac{m_н}{\lambda} = \frac{\nu}{\lambda}\frac{\frac{\alpha^n}{n!}\sum_{s=1}^{\infty}\frac{s\alpha^s}{\prod_{x=1}^{\infty}(n+x\beta)}}{\sum_{k=0}^{n}\frac{\alpha^k}{k!}+\frac{\alpha^n}{n!}\sum_{s=1}^{\infty}\frac{\alpha^s}{\prod_{x=1}^{s}(n+x\beta)}} =$$

$$= \frac{\beta}{\alpha}\cdot\frac{\frac{\alpha^n}{n!}\sum_{s=1}^{\infty}\frac{s\alpha^s}{\prod_{x=1}^{s}(n+x\beta)}}{\sum_{k=0}^{n}\frac{\alpha^k}{k!}+\frac{\alpha^n}{n!}\sum_{s=1}^{\infty}\frac{\alpha^s}{\prod_{x}(n+x\beta)}}. \qquad (6.156)$$

Непосредственное использование формул (6.153)—(6.156) затруднено тем, что в них входят бесконечные суммы. Однако члены этих сумм быстро убывают (если $\alpha<n$). Заметим, что когда параметр $\beta\to\infty$, рассматривавмая система превращается в систему с отказами (заявка мгновенно уходит из очереди). Формулы (6.153) при этом преобразуются в формулы Эрланга, а формулы (6.154) дают нули.

Приведенные выше формулы позволяют получить количественные оценки системы обслуживания и для другого крайнего случая, когда время ожидания в очереди неограниченно велико (чистая система с ожиданием). В такой системе заявки вообще не покидают очереди и поэтому P_n=0.

Как уже отмечалось, в чистой системе с ожиданием не всегда имеется предельный стационарный режим. Такой режим существует лишь в случае, если $\alpha<n$, т. е. среднее число поступающих заявок, приходящееся на среднее время обслуживания одной заявки, не превышает возможностей n-канальной системы. В противном случае $(\alpha \geq n)$ число заявок, ожидающих обслуживания в очереди, будет неограниченно возрастать.

Найдем предельные вероятности P_k состояний чистой системы с ожиданием для $\alpha<n$. Для этого положим в формулах (6.152) - (6.154) параметр β =0.

Имеем

$$P_0 = \frac{1}{\sum_{k=0}^{n} \frac{\alpha^k}{k!} + \frac{\alpha^n}{n!} \sum_{s=1}^{\infty} \frac{\alpha^s}{n^s}}. \qquad (6.157)$$

Суммируя бесконечно убывающую геометрическую прогрессию в знаменателе (6.157), получим

$$P_0 = \frac{1}{\sum_{k=0}^{n} \frac{\alpha^k}{k!} + \frac{\alpha^{n+1}}{n!\,(n-\alpha)}}. \qquad (6.158)$$

Отсюда, используя (6.153) и (6.154) , находим

$$P_0 = \frac{1}{\sum_{k=0}^{n} \frac{\alpha^k}{k!} + \frac{\alpha^{n+1}}{n!\,(n-\alpha)}}. \qquad (6.159)$$

$$P_{n+s} = \frac{\frac{\alpha^{n+s}}{n!\,n^s}}{\sum_{k=0}^{n} \frac{\alpha^k}{k!} + \frac{\alpha^{n+1}}{n!\,(n-\alpha)}}, \quad s \geq 1. \qquad (6.160)$$

Вычислим среднее число заявок, находящихся в очереди. Непосредственная подстановка β=0 в (6.155) с использованием (6.158) дает

$$m_s = \frac{\frac{\alpha^n}{n!}\sum_{s=1}^{\infty}\frac{s\alpha^s}{n^s}}{\sum_{k=0}^{n}\frac{\alpha^k}{k!}+\frac{\alpha^{n+1}}{n!\,(n-\alpha)}}. \qquad (6.161)$$

С целью упрощения (6.161) просуммируем арифметико-геометрическую прогрессию

$$\sum_{s=1}^{\infty} s\alpha^s/n^s.$$

Введя $r = \alpha/n$, имеем

$$\sum_{s=1}^{\infty}\frac{s\alpha^s}{n^s}=\sum_{s=1}^{\infty}sr^s=r\sum_{s=1}^{\infty}sr^{s-1}=r\frac{d}{dr}\left(\sum_{s=1}^{\infty}r^s\right)=$$
$$=r\frac{d}{dr}\left(\frac{r}{1-r}\right)=\frac{r}{(1-r)^2}=\frac{\alpha}{n\left(1-\frac{\alpha}{n}\right)^2}. \qquad (6.162)$$

Подставляя (6.162) в (6.161), получаем

$$m_s = \frac{\frac{\alpha^{n+1}}{n!n\left(1-\frac{\alpha}{n}\right)^2}}{\sum_{k=0}^{n}\frac{\alpha^k}{k!}+\frac{\alpha^{n+1}}{n!\,(n-\alpha)}}. \qquad (6.163)$$

Получим теперь формулу для расчета среднего времени ожидания заявки в очереди.

Если в момент поступления заявки хотя бы один из каналов системы свободен, то время ожидания равно нулю. Если заявка поступает в момент, когда все каналы системы заняты, но очереди нет, то время ожидания в среднем равно $1/(n\mu,)$ (так как поток освобождений в n-канальной системе имеет плотность $n\mu$). Если заявка застанет все каналы занятыми и одну заявку в очереди, то среднее время ожидания равно $2/(n\mu)$ и т. д. Поэтому, среднее время ожидания $\overline{t}_{\text{ож}}$ начала обслуживания равно

$$\bar{t}_{ож} = \sum_{s=1}^{\infty} \frac{s}{n\mu} P_{n+s-1}.$$

Так как в соответствии с (6.160)

$$P_{n+s-1} = \frac{n}{\alpha} P_{n+s},$$

то

$$\bar{t}_{ож} = \sum_{s=1}^{\infty} \frac{s}{\alpha\mu} P_{n+s} = \frac{1}{\lambda} \sum_{s=1}^{\infty} s P_{n+s} = \frac{m_s}{\lambda}. \quad (6.164)$$

Таким образом, среднее время ожидания начала обслуживания равно среднему числу заявок, ожидающих в очереди, деленному на плотность потока заявок.

6.12.10. Система смешанного типа с ограничением по длине очереди

В системах обслуживания смешанного типа с ограничением по длине очереди заявка, заставшая все каналы занятыми, становится в очередь лишь в том случае, если ее длина не превышает некоторого q. Если же число заявок в очереди уже равно q, то вновь поступившая заявка покидает систему необслуженной.

Рассмотрим такую n-канальную систему обслуживания, сохранив прежние допущения о том, что входящий поток заявок простейший и время обслуживания распределено по показательному закону.

Число возможных состояний такой системы конечно, так как общее число заявок, связанных с системой в этом случае, не может превышать $n+q$. Перечислим эти состояния:

E_0 — все каналы свободны, очереди нет.

E_1 — занят ровно один канал, очереди нет,

……

E_k — занято ровно k каналов, очереди нет,

……

E_n — заняты все n каналов, очереди нет,

E_{n+1} — заняты все n каналов, одна заявка стоит в очереди,

……

E_{n+s} — заняты все n каналов, s заявок стоят в очереди,

……

E_{n+q} — заняты все n каналов, q заявок стоят в очереди.

Поскольку число возможных состояний системы конечно и каждое из них достижимо из любого другого, предельный вектор в

такой системе существует. Заметим, кроме того, что в такой системе обслуживания заявка, занявшая очередь, будет ожидать обслуживания неограниченно долго. Это обстоятельство позволяет использовать для описания процесса функционирования такой системы первые $n+q$ уравнений (6.143), полученных для смешанной системы обслуживания с ограничением по длительности (см. п. 6.12.9), считая при этом параметр $v=0$.

Соответствующая совокупность алгебраических уравнений имеет вид

$$\begin{aligned}
&-\lambda P_0+\mu P_1=0,\\
&\lambda P_{k-1}-(\lambda+k\mu)P_k+(k+1)\mu P_{k+1}=0, \quad 1\leqslant k\leqslant n-1,\\
&\lambda P_{n-1}-(\lambda+n\mu)P_n+n\mu P_{n+1}=0,\\
&\lambda P_{n+s-1}-(\lambda+n\mu)P_{n+s}+n\mu P_{n+s+1}=0,\\
&\lambda P_{n+q-1}-n\mu P_{n+q}=0.
\end{aligned} \tag{6.165}$$

Особенность структуры последнего уравнения связана, во-первых, с тем, что поступление нового требования в момент, когда система находится в состоянии E_{n+q}, не может изменить состояния системы, а, во-вторых, с тем, что состояние E_{n+q} является крайним и поэтому переход из E_{n+q+1} в E_{n+q} невозможен.

Решая так же, как и ранее, эту систему уравнений с привлечением дополнительного условия

$$\sum_{k=0}^{n+q} P_k=1,$$

окончательно получаем

$$P_k=\frac{\frac{\alpha^k}{k!}}{\sum\limits_{l=0}^{n}\frac{\alpha^l}{l!}+\frac{\alpha^n}{n!}\sum\limits_{s=1}^{q}\left(\frac{\alpha}{n}\right)^s}, \tag{6.166}$$

$$P_{n+s}=\frac{\frac{\alpha^n}{n!}\left(\frac{\alpha}{n}\right)^s}{\sum\limits_{k=0}^{n}\frac{\alpha^k}{k!}+\frac{\alpha^n}{n!}\sum\limits_{s=1}^{q}\left(\frac{\alpha}{n}\right)^s}, \quad 1\leqslant s\leqslant q. \tag{6.167}$$

Вероятность того, что заявка покинет систему необслуженной, равна вероятности P_{n+q} того, что в очереди уже стоит q заявок.

Нетрудно заметить, что формулы (6.166) и (6.167), как следовало ожидать, могут быть получены из (6.153) и (6.154), если положить в них $\beta = 0$ и ограничить суммирование по s верхней границей q.

Рассчитаем среднее число заявок m'_s, ожидающих обслуживания в очереди. Для этого достаточно использовать соотношение (6.161), ограничив суммирование по s верхней границей q. При этом, так как

$$\sum_{s=1}^{q} \frac{s\alpha^s}{n^s} = \sum_{s=1}^{q} s \cdot r^s = r \sum_{s=1}^{q} s \cdot r^{s-1} = r \frac{d}{dr}\left(\sum_{s=1}^{q} r^s\right) =$$

$$= r \frac{d}{dr}\left(\frac{r - r^{q+1}}{1-r}\right) = r \frac{1 - (q+1)r^q + qr^{q+1}}{(1-r)^2},$$

то

$$m'_s = \frac{\dfrac{\alpha^{n+1}\left[1 - (q+1)r^q + qr^{q+1}\right]}{n \cdot n!\left(1 - \dfrac{\alpha}{n}\right)^2}}{\sum\limits_{k=0}^{n} \dfrac{\alpha^k}{k!} + \dfrac{\alpha^{n+1}}{n!(n-\alpha)}\left[1 - \left(\dfrac{\alpha}{n}\right)^q\right]}. \qquad (6.168)$$

Пример. В двухканальную систему массового обслуживания поступает поток заявок с плотностью λ=2 1/мин. Среднее время обслуживания одной заявки $M[t_{об}]$=2 мин. Допустимая длина очереди равна 3. Рассчитать вероятность отказа, среднее число заявок в очереди, среднее время ожидания в очереди.

Решение. Имеем: n=2, q = 3, λ =2, μ=1/ $M[t_{об}]$=0,5. По формуле (6.162)находим

$$P_{отк} = P_{n+q} = \frac{\dfrac{\alpha^n}{n!}\left(\dfrac{\alpha}{n}\right)^q}{\sum\limits_{k=0}^{n} \dfrac{\alpha^k}{k!} + \dfrac{\alpha^n}{n!}\sum\limits_{s=1}^{q}\left(\dfrac{\alpha}{n}\right)^s} =$$

$$= \frac{\dfrac{4^2}{2!} \cdot 2^3}{1 + \dfrac{4}{1} + \dfrac{4^2}{2} + \dfrac{4^2}{2}\dfrac{2 - 2^4}{1-2}} = \frac{64}{125} = 0{,}512.$$

Среднее число заявок в очереди рассчитаем по формуле (6.168)

$$m'_s = \frac{4^3}{2 \cdot 2 \cdot 125} \cdot \frac{1 - 4 \cdot 2^3 + 3 \cdot 2^4}{(1-2)^2} = \frac{16 \cdot 17}{125} = 2{,}18.$$

Наконец, используя соотношение (6.164), находим среднее время ожидания начала обслуживания для заявки, вставшей в очередь:

$$\bar{t}_{\text{ож}} = \frac{m'_s}{\lambda} = \frac{2,18}{2} = 1,09 \text{ мин.}$$

6.12.11. Система с ожиданием. Произвольные распределения для входящего потока заявок и времени обслуживания

Анализ многоканальной системы с ожиданием для случая, когда законы распределения для входящего потока заявок и времени обслуживания произвольны, весьма затруднителен. Однако задача резко *упрощается,* если система одноканальна.

Рассмотрим одноканальную систему с ожиданием, на вход которой поступает стационарный входящий поток заявок с плотностью λ и произвольным законом распределения длин интервалов между ними. Время обслуживания заявок также распределено по произвольному закону, причем $M[T_{\text{об}}] = 1/\mu$.

Пусть в момент, когда система закончила обслуживание очередной заявки, длина очереди равна u_0. Предположим, далее, что за время обслуживания следующей заявки $T_{\text{об}}$ в систему поступило некоторое случайное число заявок r. Тогда в момент окончания обслуживания очередной заявки число заявок в системе будет равно

$$u = \begin{cases} u_0 + r - 1, & \text{если } u_0 \neq 0, \\ r, & \text{если } u_0 = 0. \end{cases} \tag{6.169}$$

Соотношение (6.169) удобно записать в виде

$$u = u_0 + r - 1 + \delta, \tag{6.170}$$

где

$$\delta = \begin{cases} 1, & \text{если } u_0 = 0, \\ 0, & \text{если } u_0 \neq 0. \end{cases} \tag{6.171}$$

Усредняя соотношение (6.170) по r, имеем

$$\sum_r uP(r) = \sum_r u_0 P(r) + \sum_r rP(r) - 1 + \sum_r \delta P(r). \tag{6.172}$$

Если теперь учесть, что в силу стационарности входящего потока

$$\sum_r u \cdot P(r) = \sum_r u_0 P(r),$$

то соотношение (6.172) преобразовывается к виду

$$\sum_r \delta P(r) = 1 - \sum_r rP(r) = 1 - \lambda T_{об}. \tag{6.173}$$

Усредняя выражение (6.173) по $T_{об}$, получаем

$$\int_0^\infty \varphi(T_{об}) \sum_r \delta P(r) dT_{об} = 1 - \lambda \int_0^\infty T_{об}\varphi(T_{об}) dT_{об},$$

где $\varphi(T_{об})$ — закон распределения случайного времени обслуживания. Так как

$$\int_0^\infty T_{об}\varphi(T_{об}) dT_{об} = M[T_{об}] = \frac{1}{\mu},$$

то окончательно имеем

$$\int_0^\infty \varphi(T_{об}) \sum_r \delta P(r)\, dT_{об} = 1 - \frac{\lambda}{\mu} = 1 - \alpha. \tag{6.174}$$

Заметим, что в левой части полученного соотношения находится среднее значение параметра δ, которое с учетом (6.171) может быть вычислено иначе, а именно

$$M[\delta] = 1 \cdot \text{Вер}\{u_0 = 0\} + 0 \cdot \text{Вер}\{u_0 \neq 0\} = \text{Вер}\{u_0 = 0\}.$$

Таким образом,

$$\text{Вер}\{u_0 = 0\} = P_{св} = 1 - \alpha \text{ и } \text{Вер}\{u_0 \neq 0\} = P_{зан} = \alpha.$$

Итак, вероятность того, что одиночный канал занят, не зависит от характера законов распределения для входящего потока заявок и времени обслуживания и равна приведенной плотности заявок α.

Для определения среднего значения длины очереди возведем в квадрат обе части соотношения (6.170):

$$u^2 = u_0^2 + (r-1)^2 + \delta^2 + 2u_0(r-1) + 2u_0\delta + 2\delta(r-1).$$

Найдем математические ожидания обеих частей полученного равенства, имея в виду, что $u_0\delta = 0$ и $\delta^2 = \delta$:

$$M[u^2] = M[u_0^2] + M[(r-1)^2] + M[\delta] + 2M[u_0(r-1)] + 2M[\delta(r-1)]. \tag{6.175}$$

Из условия стационарности следует, что

$$M[u] = M[u_0] \text{ и } M[u^2] = M[u_0^2].$$

Учтем теперь, что число поступающих в систему заявок r не зависит от наличия очереди и ее длины. Тогда (6.175) перепишется следующим образом:

$$0 = M[(r-1)^2] + M[\delta] + 2M[u_0]M[r-1] + 2M[\delta]M[r-1]. \quad (6.176)$$

С другой стороны,

$$M[r] = \int_0^\infty \varphi(T_{об}) \sum_r rP(r)dT_{об} = \lambda \int_0^\infty \varphi(T_{об})T_{об}dT_{об} = \frac{\lambda}{\mu} = \alpha. \quad (6.177)$$

Кроме того,

$$M[\delta]=1-\alpha \quad (6.178)$$

в силу соотношения (6.174).

Подставляя (6.177) и (6.178) в (6.176), имеем

$$0 = M[r^2] - 2\alpha + 1 + (1-\alpha) + 2M[u](\alpha - 1) + 2(1-\alpha)(\alpha-1),$$

откуда после упрощения получаем соотношения для определения среднего числа заявок в системе

$$M[u] = \alpha + \frac{M[r^2]-\alpha}{2(1-\alpha)}. \quad (6.179)$$

Теперь легко получить формулу для расчета средней длины очереди. Очевидно, что для одноканальной системы

$$m_s = M(u) - P_{зан} = \frac{M[r^2]-\alpha}{2(1-\alpha)}. \quad (6.180)$$

Полученное выражение верно для любых законов распределения, длин интервалов между заявками на входе системы и времени их обслуживания, при условии, что они не зависят от длины очереди, не меняются с течением времени, а также если $\alpha<1$. Таким образом, определение значения $M[u]$ сведено к расчету величины $M[r^2]$. Рассмотрим частный случай, представляющий практический интерес. Входящий поток — пуассоновский, распределение времени обслуживания — произвольное. Для пуассоновского входящего потока вероятность $P_r(T_{об})$ поступления ровно r заявок в течение фиксированного времени продолжительностью $T_{об}$ равна

$$P_r(T_{об}) = \frac{(\lambda T_{об})^r}{r!} e^{-\lambda T_{об}}.$$

Кроме того, для пуассоновского закона, как известно, математическое ожидание $M_{T_{об}}(r)$ и дисперсия $Д_{T_{об}}(r)$ числа заявок, поступивших в течение интервала $T_{об}$, равны $\lambda T_{об}$.

Так как

$$Д_{T_{об}}(r) = M_{T_{об}}[(r - M_{T_{об}}(r))^2] = M_{T_{об}}[r^2] - M^2{}_{T_{об}}[r],$$

то

$$M_{T_{об}}[r^2] = \lambda T_{об} + (\lambda T_{об})^2. \tag{6.181}$$

Усредняя (6.181) по $T_{об}$, имеем

$$M[r^2] = \lambda M[T_{об}] + \lambda^2 M[T^2{}_{об}]. \tag{6.182}$$

Учитывая, что

$$Д(T_{об}) = M[(T_{об} - M(T_{об}))^2] = M[T^2{}_{об}] - M^2[T_{об}],$$

$$M(T_{об}) = 1/\mu,$$

преобразуем (6.182) к виду

$$M[r^2] = \frac{\lambda}{\mu} + \frac{\lambda^2}{\mu^2} + \lambda^2 Д(T_{об}) = \alpha + \alpha^2 + \lambda^2 Д(T_{об}). \tag{6.183}$$

Подставляя (6.183) в (6.180), имеем

$$m_s = \frac{\alpha^2 + \lambda^2 Д(T_{об})}{2(1-\alpha)}. \tag{6.184}$$

Из выражения (6.184) следует, что

$$\lim_{\alpha \to 1} m_s = \infty$$

независимо от распределения времени обслуживания.

Если время обслуживания экспоненциально, то $Д(T_{об}) = 1/\mu^2$. При этом

$$m_s^{exp} = \alpha^2/(1-\alpha).$$

Легко видеть, что средняя длина очереди принимает минимальное значение, если $Д(T_{об}) = 0$, т. е. если время обслуживания постоянно. В этом случае

$$m_s^{const} = \frac{\alpha^2}{2(1-\alpha)}.$$

Теперь получим формулы для расчета среднего времени ожидания обслуживания. Используем для этого соотношение (6.164), которое верно для любого количества каналов в системе и для произвольных распределений входящего потока и времени обслуживания:

$$\bar{t}_{ож}^{exp} = \frac{\alpha^2}{\lambda(1-\alpha)} \quad \text{и} \quad \bar{t}_{ож}^{const} = \frac{\alpha^2}{2\lambda(1-\alpha)}.$$

Таким образом, как среднее число заявок в очереди, так и среднее время ожидания при строго постоянном времени обслуживания вдвое меньше, чем в случае, когда оно распределено по показательному закону.

6.13. Метод статистических испытаний

Выше отмечалось, что если аналитическое описание процесса функционирования системы консультирования в целом получить невозможно или очень трудно, для анализа стохастических систем используется метод статистического моделирования на ЭВМ, обычно называемый методом статистических испытаний или методом Монте-Карло.

Широкое распространение этого метода связано с тем, что во многих практических случаях, когда построение аналитической модели функционирования системы консультирования в целом трудно осуществимо, удается легко описать поведение отдельных элементов системы консультирования или элементарные акты процесса ее функционирования. С другой стороны, потенциальные возможности метода выявились лишь после появления быстродействующих ЭВМ, способных в обозримое время произвести массовые расчеты, необходимые для реализации метода.

6.13.1. Существо метода статистических испытаний

Метод статистических испытаний состоит в следующем. Вместо того, чтобы описывать случайную проблему аналитически, производится ее моделирование с помощью некоторой процедуры, дающей случайный результат. При этом важно, чтобы количество различных исходов указанной процедуры и распределение вероятностей исходов, совпадало с соответствующими характеристиками анализируемой проблемы.

Рассмотрим простой пример. По некоторой цели производится три независимых выстрела, в каждом из которых вероятность попадания p=0,5. При попадании цель поражается. Рассчитать вероятность W поражения цели. Аналитическое решение задачи элементарно:

$$W = 1 - (1 - p)^3 = 1 - 0{,}5^3 = 0{,}875.$$

Решим теперь эту задачу методом статистических испытаний. В качестве случайного механизма используем, например, следующий. Левая страница любой раскрытой книги имеет четный номер, который может быть кратен или не кратен 4. Поскольку кратные и некратные 4 страницы чередуются, вероятность того, что наугад раскрытая книга имеет номер левой страницы, кратный 4, равна 0,5. Пусть теперь случайная процедура состоит в трехкратном открывании книги наугад и анализе всякий раз номера левой страницы на краткость 4. Будем считать исход такого случайного эксперимента благоприятным, если

хотя бы один из номеров кратен 4. Проведем такой эксперимент большое число раз n и подсчитаем число благоприятных исходов m.

При этом в соответствии с теорией вероятностей частота появления благоприятного исхода почти наверняка будет мало отличаться от вероятности этого события. Поэтому отношение m/n можно использовать в качестве удовлетворительной оценки вероятности появления благоприятного исхода описанного эксперимента. Поскольку вероятность появления кратного 4 номера страницы хотя бы один раз при трехкратном открывании книги наугад равна вероятности попадания в цель хотя бы одного выстрела из трех, это отношение m/n является также и оценкой искомой вероятности поражения цели.

В данном примере аналитическое решение гораздо проще, чем решение методом статистических испытаний. Однако можно привести множество примеров консультационных задач, когда аналитический расчет вероятности случайного события или статистических характеристик исследуемого случайного процесса является столь сложным, что получение их методом статистических испытаний оказывается совершенно оправданным.

Пусть, например, нужно рассчитать вероятность P того, что случайная рекомендация, представленная некоторым параметром, имеющем величину ξ с плотностью вероятности $\varphi(x)$ принимает значения в интервале $[a, b]$. Аналитически эта вероятность может быть вычислена по формуле

$$P = \int_a^b \varphi(x)dx. \qquad (6.185)$$

Если интеграл (6.185) не вычисляется в квадратурах, для оценки значения вероятности P целесообразно использовать метод статистических испытаний.

Пусть в нашем распоряжения имеется случайный механизм, формирующий случайные числа ξ с плотностью вероятности $\varphi(x)$ (способы получения такого механизма будут рассмотрены ниже). Тогда можно организовать случайную процедуру, состоящую в многократном формировании случайных чисел ξ с проверкой всякий раз выполнения неравенства $a \le \xi \le b$. Отношение числа благоприятных исходов к общему числу экспериментов дает оценку искомой вероятности P тем лучшую, чем больше экспериментов было проведено.

Очень эффективным является использование метода статистических испытаний при моделировании процессов

функционирования сложных проблем, эволюция которых может быть представлена в виде некоторой последовательности элементарных актов. Каждый из этих актов состоит в переходе проблемы с определенной вероятностью из одного состояния в другое. Если эти вероятности определяются только текущим состоянием проблемы и не зависят от того, как проблема оказалась в этом состоянии, проблема обладает марковским свойством. При этом для ее анализа могут быть использованы методы теории марковских цепей. Однако для многих реальных проблем ее эволюция в каждый момент времени определяется не только текущим состоянием, но и предысторией. В таких случаях аналитическое исследование проблемы оказывается весьма затруднительным, в то время как метод статистических испытаний с этими трудностями легко справляется. При этом производится последовательное моделирование каждого элементарного акта. Пусть, например, в процессе функционирования проблема в некоторый момент времени t_i оказалась в состоянии E_i. В соответствии с траекторией движения проблемы (последовательностью прохождения состояний) до момента t_i вычислены вероятности переходов $\{P_{ij}\}$ $(j \in \mathcal{E}$, $\mathcal{E}$ — множество возможных состояний проблемы) из состояния E_i в другое состояние. Моделирование элементарного акта перехода из E_i осуществляется путем использования датчика случайных чисел, распределенных равномерно в интервале [0; 1]. Поскольку

$$\sum_{j \in \mathcal{E}} P_{ij} = 1,$$

интервал [0; 1] может быть разбит на совокупность подынтервалов $[\alpha^i_{k-1}, \alpha^i_k]$, число которых должно быть равно количеству возможных состояний проблемы. При этом правая граница каждого подынтервала вычисляется из условия

$$\alpha^i_k = \sum_{j=1}^{k} P_{ij}, \quad k \in \mathcal{E}. \tag{6.186}$$

Пусть теперь датчик случайных чисел, распределенных по равномерному закону в интервале [0; 1], выдает случайное число ξ. Поскольку вероятность попадания ξ внутрь каждого из подынтервалов равна его длине и в соответствии с (6.186)

$$\alpha^i_k - \alpha^i_{k-1} = P_{ik},$$

то закон распределения вероятностей $\{P_{ij}\}$ переходов проблемы из E_i в другие состояния совпадает с законом распределения вероятностей

попадания ξ в подынтервалы отрезка [0; 1]. Теперь для того, чтобы определить номер состояния, в которое переходит проблема в данный момент времени t_i, достаточно узнать, внутри какого из подынтервалов оказалось случайное число ξ. Искомый номер подынтервала легко находится последовательной проверкой неравенств

$$a^l_{k-1} \leqslant \xi \leqslant a^l_k, \quad k \in \mathcal{B}.$$

Для решения этой задачи понадобился датчик случайных чисел, распределенных равномерно в интервале [0; 1]. Из дальнейшего станет ясно, что такой датчик нужен и при решении других задач методом статистических испытаний.

Поэтому рассмотрим кратко способы формирования равномерно распределенных случайных величин.

6.13.2. Формирование равномерно распределенных случайных величин

Проблема получения на ЭВМ случайных чисел, распределенных равномерно, может быть решена различными способами.

Простейший способ состоит в использовании хранящейся в памяти машины таблицы равномерно распределенных случайных чисел, подготовленной заранее. В случае необходимости получения случайного числа с помощью специальной команды обращаются к этой таблице и считывают очередную ее строку. Этот способ не нашел широкого распространения, главным образом, потому, что при использовании метода статистических испытаний в процессе решения задачи необходимо получение очень большого количества случайных чисел. Кроме этого, определенные трудности имеет предварительная подготовка достаточно большого (сотни тысяч или даже миллионы) количества равномерно распределенных случайных чисел.

Поэтому будем рассматривать лишь те способы построения генераторов случайных чисел, которые обеспечивают формирование этих чисел непосредственно в процессе работы машины. При этом задача состоит в получении последовательности величин, обладающих статистическими свойствами, аналогичными системе случайных чисел с заданным законом распределения. Такие последовательности принято называть псевдослучайными. Если при этом формируемая последовательность величин обладает статистическими характеристиками системы равномерно распределенных случайных чисел, ее называют квазиравномерной.

ЭВМ работают, как правило, с числами, представленными в двоичной системе счисления, т. е. в виде

$$\xi = \sum_{i=1}^{n} \varepsilon_i 2^{-i}, \tag{6.187}$$

где

$$\varepsilon_i = \begin{cases} 1, & \text{если } 2^{-i} \text{ присутствует в двоичном разложении} \\ & \text{числа } \xi, \\ 0 & \text{в противном случае.} \end{cases}$$

Поэтому понятно, что получение равномерно распределенных n-разрядных случайных чисел сводится к отысканию способа формирования чисел в виде (6.187), обеспечивающего выполнение требования

$$\varepsilon_i = \begin{cases} 1 \text{ с вероятностью } 1/2, \\ 0 \text{ с вероятностью } 1/2, \end{cases} \quad i = 1, 2, \ldots, n.$$

На практике используется способ формирования квазиравномерных распределений случайных чисел программным методом.

Методы программного получения псевдослучайных чисел, как правило, основаны на использовании некоторого рекуррентного вычислительного процесса, определяемого в общем случае соотношением $\xi_n = f(\xi_{n-1}, \xi_{n-2}, \ldots, \xi_{n-k})$.

Программа вычисления последовательности равномерно распределенных случайных чисел должна удовлетворять следующим требованиям:

1) полученная в результате работы программы совокупность псевдослучайных чисел должна отвечать установленным критериям проверки «случайности»;

2) случайные числа полученной совокупности должны быть весьма слабо коррелированы между собой;

3) распределение полученной совокупности псевдослучайных чисел должно достаточно хорошо аппроксимировать равномерное распределение.

Вполне удовлетворительные для практики результаты дает следующая рекуррентная процедура получения псевдослучайных чисел.

Выбирается пара чисел

$$\alpha_0 = \varepsilon_{10} \cdot 2^{-1} + \varepsilon_{20} \cdot 2^{-2} + \ldots + \varepsilon_{n0} \cdot 2^{-n},$$
$$\alpha_1 = \varepsilon_{11} \cdot 2^{-1} + \varepsilon_{21} \cdot 2^{-2} + \ldots + \varepsilon_{n1} \cdot 2^{-n}.$$

Образуем произведение этих двух чисел

$$\alpha_0 \cdot \alpha_1 = \delta_1 \cdot 2^{-1} + \delta_2 \cdot 2^{-2} + \ldots + \delta_{2n} \cdot 2^{-2n}$$

и выберем средние цифры полученного числа в качестве α_2, т. е. считая n четным, имеем

$$\alpha_2 = \delta_{n/2+1} \cdot 2^{-1} + \delta_{n/2+2} \cdot 2^{-2} + \ldots + \delta_{n/2+n} \cdot 2^{-n} =$$
$$= \varepsilon_{12} \cdot 2^{-1} + \varepsilon_{22} \cdot 2^{-2} + \ldots + \varepsilon_{n2} \cdot 2^{-n}.$$

Процесс повторяется для α_1 и α_2 с получением тем же способом α_3 и т. д.

Рассмотрим, наконец, существо программных методов формирования последовательности псевдослучайных чисел, основанных на имитации хаотического перемешивания содержимого разрядов мантиссы псевдослучайных чисел.

Типовая программа такой процедуры имеет следующую структуру:

1) изображение начального числа α_0 сдвигается на некоторое количество разрядов в сторону младших разрядов;

2) полученное при этом число складывается с α_0;

3) вычисляется абсолютная величина суммы. Полученный результат и представляет собой α_1. Далее процедура повторяется нужное число раз.

Недостатком всех программных способов получения псевдослучайных чисел является то, что получаемая при их реализации последовательность оказывается периодической. Поэтому очень длинные последовательности уже не будут случайными. Правда, при удачном выборе процедуры период может быть весьма большим (несколько миллионов). Другой недостаток состоит в трудности теоретической оценки статистических свойств получаемой последовательности.

Важным достоинством программных методов является простота практической реализации. Кроме того, при использовании этих методов возможен контроль работы машины в процессе решения задачи (возможен двойной просчет).

6.13.3. Формирование случайных величин с заданным законом распределения

Для формирования последовательности случайных чисел, имеющих заданный закон распределения, может быть использована последовательность равномерно распределенных случайных чисел.

Пусть η —случайное число с равномерной плотностью распределения в интервале [0; 1]. Введем случайное число ξ с помощью соотношения

$$\xi = \varphi(\eta). \tag{6.188}$$

причем φ(η)-монотонная функция. Тогда однозначно может быть получена обратная функция

$$\eta = \varphi^{-1}(\xi).$$

Запишем функцию распределения $P_{\xi}(x)$ случайной величины ξ. По определению

$$F_{\xi}(x) = p(\xi < x) = \int_{-\infty}^{x} f(x)dx, \tag{6.189}$$

где $f(x)$—плотность распределения случайной величины ξ.

Используя (6.188), имеем

$$P(\xi < x) = P[\varphi(\eta) < x]. \tag{6.190}$$

Так как φ(η) монотонна, то

$$P[\varphi(\eta) < x] = P[\eta < \varphi^{-1}(x)]. \tag{6.191}$$

С другой стороны, поскольку η—равномерно распределенная случайная величина, то

$$P[\eta < \varphi^{-1}(x)] = \int_{0}^{\varphi^{-1}(x)} dy = \varphi^{-1}(x). \tag{6.192}$$

Объединяя (6.189) —(6.192), имеем

$$\int_{-\infty}^{x} f(x)dx = \varphi^{-1}(x), \text{ или } \int_{-\infty}^{\xi} f(x)dx = \varphi^{-1}(\xi) = \eta. \tag{6.193}$$

Соотношение (6.193) позволяет рассчитать последовательность чисел $\xi_1, \xi_2, ..., \xi_n, ...$ с заданным законом распределения $f(x)$ с использованием последовательности $\eta_1, \eta_2, ..., \eta_n, ...$ равномерно распределенных случайных чисел.

Пусть, например, требуется получить случайные числа с показательным законом распределения

$$f(x)=\lambda\cdot e^{-\lambda x},\quad x>0.$$

Используя (6.193), получаем

$$\lambda\int_0^{\xi_i} e^{-\lambda x}dx=\eta_i, \tag{6.194}$$

или после вычисления интеграла (6.194)

$$1-e^{-\lambda\xi_i}=\eta_i.$$

Решая это уравнение относительно ξ_i, будем иметь

$$\xi_i=-\frac{1}{\lambda}\ln(1-\eta_i). \tag{6.195}$$

Если в нашем распоряжении имеется последовательность η_1, η_2, ..., η_n,... чисел, распределенных равномерно, то последовательность чисел ξ_1, ξ_2, ..., ξ_n,... будет распределена по показательному закону.

При решении многих задач непосредственное использование соотношения (6.193) оказывается неудобным (например, когда интервал (6.193) не вычисляется в квадратурах). Поэтому на практике широкое распространение имеют различные приближенные приемы преобразования случайных чисел.

Рассмотрим один такой прием.

Пусть требуется получить последовательность случайных величии с функцией плотности *f(x).* Если область определения соответствующей случайной величины ξ не ограничена, перейдем к усеченному распределению на интервале [*c; d*]. Разобьем [*c; d*] на *n* интервалов. Тогда случайная величина ξ_j может быть представлена в виде суммы

$$\xi_j=a_j+\zeta_j, \tag{6.196}$$

где a_i — абсцисса левой границы *j*-го интервала; ξ_j - случайная величина, возможные значения которой располагаются внутри интервала $[a_j; a_{j+1}]$.

Процедура получения случайного числа ξ_j сводится теперь к следующему:

1) случайный выбор интервала (определение значения a_j),
2) случайный выбор ξ_j из интервала $[a_j; a_{j+1}]$.
3) формирование ξ_j в соответствии с соотношением (6.196).

Таким образцом, для формирования одного случайного числа из их последовательности с заданным законом распределения необходимо дважды использовать датчик случайных чисел.

Понятно, что при выборе интервала на первом шаге указанной процедуры должна учитываться заданная плотность распределения *f(x)*. С этой целью, используя разбиение [*c*; *d*] на *n* интервалов, кусочно-линейно аппроксимируем *f(x)* отрезками прямых, параллельных оси абсцисс (рис. 6.36).

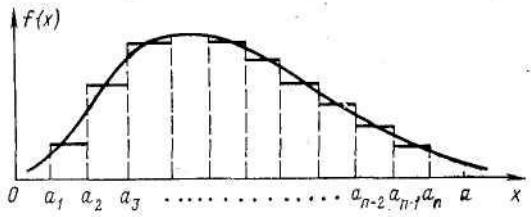


Рис.6.36.

Вычислим площадь каждого из прямоугольников на которые разбивается фигура, ограниченная осью абсцисс и аппроксимирующей *f(x)* ломаной.

Для приближенного вычисления площади *k*-го прямоугольника можно использовать формулу

$$S_k \approx \frac{d-c}{n}\ \frac{f(a_k)+f(a_{k+1})}{2}.$$

Пусть в нашем распоряжении имеется датчик случайных чисел η_1, η_2, ..., распределенных равномерно в интервале [0; 1]. Разобьем интервал [0; 1] на *n* подынтервалов (рис. 6.37): [α_1; α_2], [α_2; α_3], [α_3; α_4], ..., [α_n; 1] таким образом, чтобы выполнялись соотношения

$$\alpha_k = \sum_{i=1}^{k-1} S_i, \quad k = 2, 3, \ldots, n, \ \alpha_1 = 0.$$

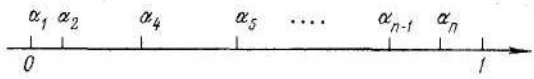


Рис. 6.37.

Теперь понятно, что вероятность попадания случайного числа ξ с плотностью распределения *f(x)* внутрь интервала [a_i; a_{i+1}] равна вероятности попадания равномерно распределенного случайного числа η внутрь интервала [α_i; α_{i+1}]. Поэтому при получении очередного случайного числа ξ_i из последовательности ξ_1, ξ_2, ξ_3,... номер интервала *j* отыскивается путем проверки системы неравенств

$$\alpha_k \leqslant \eta_{j1} \leqslant \alpha_{k+1}, \ k = 1, 2, 3, \ldots, n, \qquad (6.197)$$

после чего по номеру k_i удовлетворившегося неравенства рассчитывается значение a_j.

$$a_j = c + \frac{d-c}{n}(k_j - 1). \tag{6.198}$$

Количество интервалов n для разбиения области определения случайной величины ξ обычно выбирают достаточно большим, чтобы распределение этой случайной величины внутри каждого из интервалов можно было бы приближенно считать равномерным. При этом случайная добавка ξ_j к величине a_j вычисляется по формуле

$$\zeta_j = \frac{d-c}{n}\eta_{j2}. \tag{6.199}$$

В соотношениях (6.198) и (6.199) (η_{i1}, η_{i2}) — пара случайных чисел, распределенных равномерно, используемых для формирования ξ_i.

Достоинством описанной процедуры является простота и возможность применения для формирования случайных чисел с как угодно сложным законом распределения. Недостаток состоит в необходимости проведения некоторой подготовительной работы перед непосредственным применением процедуры (разбиение области определения ξ на интервалы, подсчет величин $\{S_k\}$ и $\{\alpha_k\}$).

6.13.4. Применение метода статистических испытаний для анализа систем консультирования

Выше отмечалось, что задачи анализа широкого класса реально функционирующих систем консультирования могут быть решены методами теории массового обслуживания. Однако следует заметить, что результаты, имеющиеся в литературе по этому вопросу, получены главным образом для сравнительно простых случаев. Это относится прежде всего к описанию входного потока проблем, а также структуры и свойств самих систем консультирования. Вместе с тем реальные входящие потоки проблем по своим свойствам далеко не всегда соответствуют простейшему потоку проблем, время консультирования часто не распределено по показательному закону и, наконец, логика, устанавливающая дисциплину консультирования, может быть достаточно сложной. По этим причинам аналитическое описание многих реальных систем консультирования оказывается затруднительным. В связи с этим для анализа таких систем консультирования целесообразно использовать метод статистических испытаний, справляющийся с перечисленными трудностями.

Метод статистических испытаний позволяет более полно по сравнению с аналитическими методами характеризовать зависимость эффективности систем консультирования от параметров потока проблем и самой системы консультирования. Так, например, метод

статистических испытаний позволяет оценить не только простейшие характеристики эффективности системы консультирования (вероятность отказа, вероятность консультирования, среднее значение доли отказов), но и значения многих других важных показателей системы консультирования (значение дисперсии доли отказов; вероятность того, что значение доли отказов будет не ниже заданного и т.п.).

Метод статистических испытаний позволяет осуществить оценку эффективности многофазных систем консультирования, аналитическое исследование которых ограничивается трудностями описания потока проблем, покинувших очередную фазу консультирования и поступающих на вход следующей.

Сущность метода статистических испытаний применительно к анализу систем консультирования состоит в следующем. С помощью специальных алгоритмов формируются реализации потока проблем с заданным законом распределения интервалов между заявками. Далее моделируется процесс функционирования системы консультирования. Все показатели работы системы консультирования, интересующие исследователя, фиксируются. Общий алгоритм модели многократно воспроизводит случайные реализации процесса функционирования системы консультирования при некоторых заранее заданных условиях (характер и параметры входящего потока, параметры системы и т. д.). Накопленная в результате информация статистически обрабатывается.

Рассмотрим кратко общие принципы построения таких моделей.

Входящий поток проблем однозначно задается последовательностью моментов времени поступления проблем в систему t_1, t_2, ..., t_i. Удобно вместо величин t_1, t_2, ..., t_i, ..., рассматривать случайные величины ξ_1, ξ_2, ..., ξ_i, ..., определяющие длину интервалов между моментами поступления проблем. При этом

$$t_1 = \xi_1,$$
$$t_2 = \xi_1 + \xi_2,$$
$$\cdots\cdots\cdots$$
$$t_i = \xi_1 + \xi_2 + \ldots + \xi_i.$$

Теперь понятно, что для задания входящего потока достаточно получить последовательность случайных величин ξ_1, ξ_2, ..., ξ_i с заданным законом распределения. Методы получения такой последовательности с использованием датчика равномерно распределенных случайных чисел были рассмотрены выше.

Пусть, например, на вход системы поступает простейший поток. Плотность распределения длин интервалов между заявками для такого потока имеет вид

$$f(x)=\lambda e^{-\lambda x}, \quad x>0. \tag{6.201}$$

Тогда, как было показано, для получения последовательности чисел $\xi_1, \xi_2, ..., \xi_i, ...$ используется соотношение (6.195):

$$\xi_i=-\frac{1}{\lambda}\ln(1-\eta_i),$$

где η_i — i-е случайное число из последовательности с равномерным распределением.

Предположим теперь, что на вход поступает стационарный поток с ограниченным последействием и равномерным распределением длин интервалов между проблемами. Функция плотности для такого потока имеет вид

$$f(x)=\frac{1}{b}, \quad 0\leqslant x\leqslant b.$$

Используя соотношение (6.193), получим

$$\int_0^{\xi_i}\frac{1}{b}\,dx=\frac{\xi_i}{b}=\eta_i,$$

Откуда

$$\xi_i=b\eta_i, \quad i=1,\ 2,\ 3,\ \ldots$$

Аналогично может быть получена последовательность $\xi_1, \xi_2, ..., \xi_i, ...$ и для каких-либо других законов распределения длин интервалов между проблемами.

Те же приемы используются для формирования случайных значений времени консультирования.

Принцип построения алгоритма, моделирующего процесс функционирования системы консультирования, рассмотрим на примере простейшей системы консультирования с отказами. На рис. 6.38 представлена логическая граф-схема алгоритма, описывающего работу такой системы консультирования.

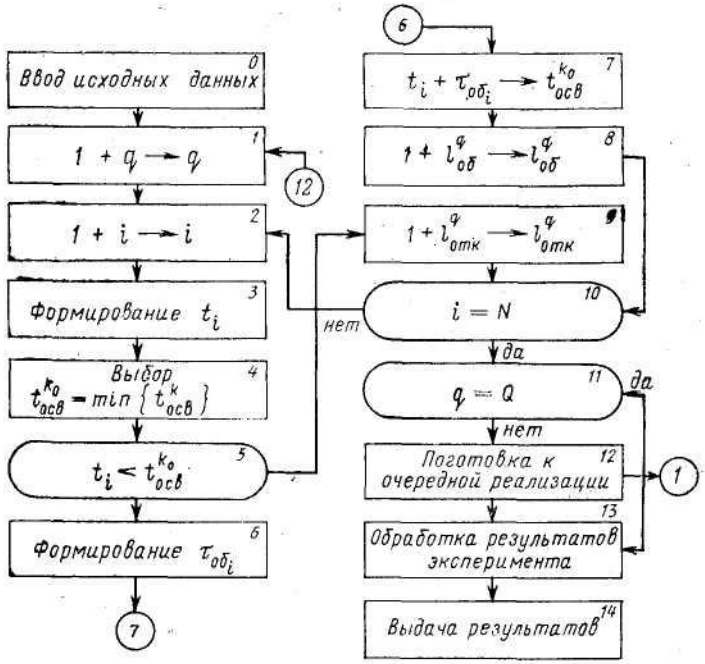


Рис. 6.38.

Оператор 0 осуществляет ввод исходной информации (число каналов системы, параметры входящего потока и закон распределения времени консультирования).

Оператор 1 представляет собой счетчик числа реализаций процесса функционирования системы консультирования.

Оператор 2 фиксирует номер очередной проблемы.

Оператор 3 формирует значения момента поступления очередной проблемы в соответствии с методикой, изложенной выше.

Оператор 4 осуществляет сравнение между собой моментов освобождения каналов системы консультирования и выбирает из них наиболее ранний. Пусть номер соответствующего канала равен k_0.

Оператор 5 производит сравнение величины t_i (момент поступления очередной проблемы) с величиной $t^{k_0}_{осв}$ (момент освобождения канала k_0). Если $t_i < t^{k_0}_{осв}$, управление передается оператору 9, в противном случае—оператору 6.

Оператор 9 работает в случае, если неравенство $t_i < t^{k_0}_{осв}$ выполняется, что соответствует поступлению очередной проблемы до освобождения какого-либо из каналов системы консультирования. При этом проблема получает отказ на консультирование и значение счетчика числа отказов $l^q_{отк}$ увеличивается на единицу.

Оператор 6 работает в случае, если неравенство $t_i < t^{k_0}_{осв}$ не выполняется. При этом канал k_0, раньше всех освободившийся, начинает консультирование очередной проблемы. В соответствии с изложенной выше методикой формируется случайное значение времени консультирования этой проблемы.

Оператор 7 вычисляет время освобождения канала k_0 от консультирования очередной проблемы, начавшегося в момент t_i и продолжающегося в течение интервала $\tau_{об\,i}$.

Оператор 8 представляет собой счетчик числа проконсультировавнных проблем.

Оператор 10 осуществляет сравнение номера очередной проблемы с общим числом проблем N, поступающих в систему. Если равенство $i = N$ не выполняется, осуществляется переход к оператору *2*, в противном случае — к оператору *11*.

Оператор 11 проверяет, выполнено ли запланированное количество консультаций. Если число проведенных консультаций q еще не равно запланированному Q, управление передается оператору *12*, в противном случае — оператору *13*.

Оператор 12 осуществляет подготовку к очередной итерации. При этом очищаются рабочие ячейки памяти, хранящие значения i, $\{t^k_{осв}\}^n_1$, а содержимое ячеек $l^q_{отк}$ и $l^q_{об}$ пересылается в специальный массив для последующей статистической обработки.

Оператор 13 осуществляет статистическую обработку наборов $\{l^q_{отк}\}^Q_1$ и $\{l^q_{об}\}^Q_1$. При этом могут быть вычислены оценки для параметров, характеризующих эффективность системы.

Средняя доля проконсультированных проблем из общего числа поступивших проблем рассчитывается по формуле

$$M[r] = \frac{\sum_{q=1}^{Q} l^q_{об}}{NQ}.$$

Дисперсия случайного значения доли обслуженных заявок рассчитывается по формуле

$$Д[r] = \frac{\sum_{q=1}^{Q} \left(\frac{l_{об}^{q}}{N} - M[r] \right)^2}{Q-1}.$$

Вероятность отказа на консультирование может быть оценена через частоту отказов по формуле

$$P_{отк} = \frac{\sum_{q=1}^{Q} l_{отк}^{q}}{NQ}.$$

Вероятность того, что доля проконсультированных проблем будет не ниже заданной, может быть оценена следующим образом.

Пусть $r_З$ — заданная доля проконсультированных проблем; Q_1 — количество реализаций, для каждой из которых $r = l_{об}/N \geq r_З$. Тогда

$$P(r \geq r_З) = Q_1/Q.$$

С использованием достаточно полной статистической модели системы можно оценить и многие другие показатели ее эффективности.

Сделаем несколько замечаний относительно оценки точности метода. Пусть моделируется процесс, в котором результатом каждого из N независимых консультаций является рекомендация, идентифицируемая случайной величиной ξ_i (i — номер испытания). Предположим, что эта величина обладает конечными математическим ожиданием $M[\xi_i]=a$ и дисперсией $Д[\xi_i]=\sigma^2$. Тогда среднее арифметическое

$$\bar{\xi} = \frac{1}{N} \sum_{i=1}^{N} \xi_i$$

является приближенным значением математического ожидания a. Ошибка оценки может быть установлена с помощью неравенства Чебышева:

$$P\{|\bar{\xi} - a| < \varepsilon\} > 1 - \delta; \quad N > N_0 = \sigma^2/\varepsilon^2\delta. \tag{6.202}$$

Соотношение (6.202) позволяет рассчитать число испытаний N, которые необходимо провести для получения оценки математического ожидания с заданными точностью ε и достоверностью 1—δ.

Обычно трудно заранее оценить дисперсию σ^2. Поэтому при решении конкретных задач вместо теоретического значения σ^2 используют статистическую оценку дисперсии:

$$\Delta = \frac{\sum_{i=1}^{N} (\xi_i - \bar{\xi})^2}{N-1}.$$

Пусть теперь моделируется событие A, вероятность появления которого равна p. Введем величину

$$\xi_i = \begin{cases} 1, & \text{если событие } A \text{ произошло в } i\text{-м испытании,} \\ 0 & \text{в противном случае.} \end{cases}$$

Тогда количество испытаний, в каждом из которых событие A произошло, равно

$$L = \sum_{i=1}^{N} \xi_i,$$

где N— общее число испытаний.

Частота появления события A равна L/N и является случайной величиной, имеющей математическое ожидание

$$M\left[\frac{L}{N}\right] = \frac{1}{N} M[L] = \frac{1}{N} \sum_{i=1}^{N} M[\xi_i] = p$$

и дисперсию

$$Д\left[\frac{L}{N}\right] = \frac{1}{N^2} Д[L] = \frac{1}{N^2} \sum_{i=1}^{N} Д[\xi_i] =$$

$$= \frac{1}{N^2} \sum_{i=1}^{N} M[(\xi_i - p)]^2 =$$

$$= \frac{1}{N^2} \sum_{i=1}^{N} (M[\xi^2_i] - p^2) = \frac{p(1-p)}{N}.$$

В соответствии с законом больших чисел (теорема Бернулли) частота появления события A, равная L/N, при достаточном числе испытаний как угодно мало отличается от соответствующей вероятности p,

Необходимое число испытаний N_0, как и ранее, можно оцепить с помощью неравенства Чебышева

$$N > N_0 = \frac{\sigma^2}{\varepsilon^2 \delta} = \frac{p(1-p)}{\varepsilon^2 \delta},$$

так как

$$\sigma^2 = Д[\xi_i] = M[(\xi_i - p)^2] = M[\xi^2_i] - p^2 = p(1-p).$$

Приложение

Система управления реляционной базой данных ORACLE 7

В приложении рассматривается система управления реляционной базой данных ORACLE 7 с языком доступа к данным SQL. ORACLE 7, появившаяся в 1992 г., представляет собой совершенный программный продукт, реализующий как расширенную реляционную модель, так и архитектуру клиент-сернер и нашедший, к моменту издания настоящей книги, наибольшее применение среди других подобных программных продуктов.

Излагаемый материал распределён на две части.

В первой части рассматриваются: реляционная модель данных, структура базы данных, типы данных и объекты в ORACLE 7, принципы работы системы, обеспечивающие высокую надежность и гибкость данных.

Вторая часть является основной и содержит описание структурного языка запросов SQL стандарта ANSI/ISO. Это мощный непроцедурный язык, который обеспечивает легкость и гибкость манипулирования данными и является основным средством непосредственной работы с данными и разработки прикладного программного обеспечения.

ЧАСТЬ ПЕРВАЯ. КОНЦЕПЦИЯ ORACLE 7.0

1. Реляционная БД и реляционная модель данных

Построение информационных систем, на современном этапе, основано на использовании баз данных (БД) и систем управления базами данных (СУБД). Понятие базы данных появилось в 60-х годах, а в следующих десятилетиях большие фирмы, занимающиеся разработкой программного обеспечения, разрабатывают различные концепции БД и на этой основе разнообразные СУБД. Такими системами являются ORACLE фирмы Oracle Corporation, ACCESS фирмы Microsoft, Paradox фирмы Borland, dBASE фирмы Ashton-Tate, RBASE фирмы Microrim и др.

1.1. Основные понятия

Понятие "база данных" не имеет достаточно четкого определения в специализированной литературе. Его можно рассматривать с разных точек зрения из-за чего и отсутствует точное определение. Однако наиболее общее определение может выглядять следующим образом: база данных это интегрированная совокупность информации какой-либо области человеческой деятельности, ориентированная на совместное использование. В строгом смысле слова база данных - это файл данных, определенный посредством схемы и не зависящий от

программ, которые к нему обращаются. Поняие база данных предполагает ее реализацию в виде запоминающего устройства с прямым доступом.

Это определение имеет ***три основные аспекта***: первый, что ***данные описывают объекты данной области***, второй, что ***информация для этих объектов интегрированная,*** т.е. даны ***взаимосвязи*** между ними, и третий, что ***данные используются одновременно и совместно многими пользователями.***

Область человеческой деятельности, информация для которой сохраняется в базе данных, называется ***предметной областью.*** Предметная область описывается определенным образом посредством так называемой ***концептуальной схемы***. Обычно отдельный пользователь использует информацию, относящуюся к отдельным частям объектов предметной области. Часть концептуальной схемы, описывающая подходящим образом объекты, которые интересуют данного пользователя, называется ***внешней схемой, внешним интерфейсом или клиентным приложением***. Чтобы записать информацию для объектов предметной области в запоминающее устройство компьютера, необходимо концептуальную схему описать с точки зрения представления и записи данных на ***физическом уровне.*** Такое описание называется ***внутренней схемой.***

Для описания данных и взаимосвязи между ними в концептуальной схеме можно использовать различные модели. В практике утвердились три основные модели - ***иерархическая, сетевая и реляционная***. База ORACLE 7 относится к реляционным базам данных.

Реляционная модель описывает связи между объектами посредством операции над множествами. Элементами них множеств являются описываемые в БД объекты. Реляционная алгебра очень хорошо и очень точно описывет эти операции Так как реляционная модель логически ясна и теоретически хороню разработана, она нашла самое широкое распространение в практике при разработке систем управления базами данных.

1.2. Реляционная модель

Объекты реляционной модели описываются посредством ***значений их характеристик***. Любая характеристика называется ***атрибутом*** и принимает ***значения*** из определенной области допустимых значений, которая называется ***доменом. Множество объектов*** с одинаковыми атрибутами и различными значениями называется ***реляцией***. Самое удобное ***представление*** реляции это представление в виде ***таблицы*** и в дальнейшем будем использовать термин таблицы вместо реляции. Столбцы таблицы соответствуют атрибутам и именуются подходящим

образом. Каждая строка таблицы содержит данные, которые характеризируют отдельный объект.

Пример: Пусть объекты базы данных отображают следующие сущности данного вуза: кафедры, преподаватели, студенты и их баллы. Атрибуты кафедры - это порядковой номер кафедры, наименование факультета и наименование кафедры. Атрибуты преподавателей - это номер кафедры, номер преподавателя кафедры, научное звание и его фамилия, имя, отчество. Атрибуты студентов - это факультативный номер, фамилия, имя, отчество, специальность и семестр. Атрибуты баллов это факультативный номер студента, номер кафедры, номер преподавателя, учебная дисциплина, балл и семестр. Значение каждого атрибута принадлежит данному ***домену*** т.е. ***определенному множеству допустимых величин.*** Например, значение атрибута "научное звание" может принимать одно из следующих значений: ассистент, доцент и профессор.

В реляционной алгебре определены следующие операции:

1. *Проекция.* Эта операция создает (возвращает) таблицу T_2, которая содержит часть столбцов заданной таблицы T_1. Обе таблицы имеют одинаковое количество строк, а значения атрибутов T_2 - это соответствующие значения атрибутов строк T_1. Пример: Одной из проекций таблицы преподавателей может быть таблица со столбцами: номер кафедры, номер преподавателя кафедры и фамилия, имя, отчество преподавателя.

2. *Рестрикция (подмножество).* Эта операция возвращает таблицу T_2, которая содержит те же самые столбцы, что и заданная таблица T_1 и часть строк T_1. Значения определенных атрибутов этих строк отвечают некоторым условиям. Пример: Одной из возможных рестрикций таблицы преподавателей может быть таблица преподавателей, которые занимают должности доцента и профессора.

3. *Объединение.* Пусть заданы две таблицы T_1 и T_2 с одинаковыми столбцами. Операция объединение создает таблицу T_3, которая содержит строки двух столбцов. Пример: Таблица T_1 содержит преподавателей кафедры математики, а T_2 - преподавателей кафедры информатики. Если таблица T_3 объединение T_1 и T_2, то она будет содержать преподавателей с обеих кафедр.

4. *Разница.* Пусть даны две таблицы T_1 и T_2 с одинаковыми столбцами. Операция разности создает таблицу T_3, которая содержит строки таблицы T_1, которые не являются строками таблицы T_2. Пример: Таблица T_1 содержит преподавателей всех кафедр, а T_2 - преподавателей кафедры информатики. Если таблица T_3 является

разницей T_1 и T_2, то она будет содержать преподавателей всех кафедр за исключением кафедры информатики.

5. *Сечение.* Пусть даны две таблицы T_1 иT_2 с одинаковыми столбцами. Операция сечения возвращает таблицу T_3 со строками, которые содержатся как в таблице T_1, так и в таблице T_2. Пример: Таблица T_1 содержит преподавателей всех кафедр, а T_2 преподавателей кафедры информатики. Если таблица T_3 является сечением двух таблиц, то она будет содержать преподавателей кафедры информатики.

6. *Декартовое произведение.* Эта операция превращает таблицу, строки которой являются всеми возможными комбинациями строк исходных таблиц.

Пример: Если количество строк одной исходной таблицы - N, а другой – M, то результативная таблица будет иметь N*M строк.

Реляционные системы управления базами данных, как правило, имеют такие преимущества как: независимость физического представления данных от логической структуры БД; разнообразный и легкий доступ ко всем данным; большая гибкость при описании баз данных; уменьшение избыточности информации в базах данных.

2. Структура базы данных ORACLE

Поскольку ORACLE 7 является реляционной системой управления базой данных, то она обеспечивает рациональную логическую и физическую структуру данных, контроль доступа к данным и эффективное управление дисковым пространством. Архитектура базы данных включает определение структур, операций и правил целостности.

Структуры - это хорошо дефинированные объекты, которые служат для накопления данных в базе данных.

Операции являются явно дефинированными действиями, которые позволяют пользователям работать с данными и структурами БД. Операции в БД должны придерживаться предварительно определенных правил интегрированного управления.

Интегрированное управление обеспечивается на основе предварительно установленных пользователями правил целостности. Привила целостности, со своей стороны, это законы, которые управляют работой БД, благодаря которым операции имеют доступ к данным и структурам в БД.

База данных ORACLE - что множество данных, которые рассматриваются как отдельные части. Ее основное предназначение сохранять и восстанавливать связанные данные. Так как в ORACLE используются две структуры - физическая и логическая, то это позволяет в полной мере применять принцип независимости данных.

Это позволяет изменять физическое сохранение данных не влияя на способ доступа к сохраненным логическим структурам.

Физическая структура определяется файловой структурой оперпционной системы. Любая ORACLE-база данных составлена из трех типом файлов: один или несколько файлов с данными, один или несколько файлов для восстановления данных и один или несколько контрольных файлов. Файлы баз данных представляют действительные физические склады информации дня объектов базы данных.

Логическая структура баз данных определяется ***множеством схемных объектов*** БД, таких как ***таблицы, виды, индексы, кластеры, последовательности и процедуры*** для обработки данных, а также ***множеством экстентов и сегментов*** (см. 2.9). Схемные объекты и связи между ними формируют связное описание баз данных.

Для работы с данными в системе управления базами данных ORACLE используется язык SQL -*Structured Querly Language* (структурированный язык запросов). Язык SQL предоставляет пользователям простые, унифицированные средства для определения, переопределения, записи, стирания и других манипуляций данными в базе данных. Для повышения возможностей обработки данных ORACLE предоставляет процедурное расширение языка SQL, а именно PL/SQL. Команды SQL и PL/SQL могут выполняться непосредственно в среде программы SQLPLUS или их можно встраивать в программы языков высокого уровня: ADA, C, COBOL, FORTRAN, Pascal и PL/1. Команды SQL и PL/SQL могут быть изпользованы также и в системах разработки прикладных программ, таких как SQL*Menu, SQL*Foprms, SQL*ReportWriter и другие.

2.1. Типы данных

Так как ORACLE 7 является реляционной базой данных, то она сохраняет данные для объектов в таблицах. Столбцы этих таблиц - это атрибуты объектов. Для дефинирования допустимых значений атрибутов поддерживаются следующие типы данных: *CHAR, VARCHAR2, VARCHAR, NUMBER, DATE, LONG, RAW, LONG RAW, ROWID* и *MLSLABEL.*

Данные типа *CHAR* - это строка (последовательность знаков) фиксированной длины, которая определяется пользователем. Длина строки находится в пределах от 1 до 255 знаков и задается в скобках. Например, *CHAR(5)* задаёт строку длиной 5 знаков.

Данные типа *VARCHAR2* - это строка с изменяющейся длиной, для которой пользователь задал максимально допустимую длину в границах от 1 до 2000 знаков. Длина задается в скобках. Например,

VARCHAR2(25) определяет строку с возможной длиной от 1 до 25 знаков.

Тип данных *VARCHAR* является синонимом *VARCHAR2,* который зарезервирован для будущей версии реляционной базы, в которой будут допускаться различные семантики для сравнения строк. Примером различных семантик является сравнение строк с учетом или без учета пустых интервалом.

Данные типа *NUMBER* - это числовые данные с определенным форматом для сохранения чисел, которые могут задаваться в скобках. Если не задан формат, то число сохраняется в таком виде, в котором оно было введено (количество знаков до и после десятичной запятой сохраняется). Формат может быть задан парой значений (N,M). Число M определяет количество знаков после десятичной запятой. Если вводимое число имеет больше чем M знаков после запятой, оно сохраняется округленным с точностью до M знаков после запятой. Если значение M не задано, то подразумевается, что оно равно нулю. Значение N определяет общее количество знаков числа. Если в качестве значения N задать *, то число сохраняется так, как было введено. Если ввести число с таким количеством знаков до десятичной запятой, что при заданных M и N, сохранение невозможно, то тогда ORACLE выдает сообщение об ошибке. Ниже показано как сохранится число 748547,536 в различных случаях задания формата:

'NUMBER	*748547,536*
NUMBER(9,2)	*748547,54*
NUMBER(,1)*	*748547,5*
NUMBER(6,2)	*сохранение невозможно*
NUMBER(9)	*748548*
NUMBER(9,-2)	*748500*

Тип данных *DATE* служит для сохранения даты. Даты вводятся и сохраняются в формате вида dd-mmm-yyyy, где dd это день, mmm - первые три буквы наименования месяца на английском языке и yyyy - год. Используя функции *TO DATE* и *TO CHAR* для преобразования данных, дату можно вводить и выводить и в других форматах. Новый формат задается параметром соответствующей функции.

Тип данных *LONG* используется для сохранения текста длиной до 2GB. Нет надобности задавать конкретную или максимальную длину текста. При использовании этого типа данных следует иметь ввиду следующие ограничения:

$\Rightarrow$ в одной таблице может быть максимум один столбец этого типа;

$\Rightarrow$ столбец этого типа не индексируется;

⇒ не допускается задание ограничения целостности этого типа данных;

⇒ столбец этого типа не может использоваться с *WHERE, GROUP BY, ORDER BY, CONNECT BY* или *DISTINCT* в команде *SELECT;*

⇒ нет возможности в SQL использовать функции *SUBSTR* и *INSTR;*

⇒ не используется в выражениях;

⇒ нельзя, на основе этого типа данных, создать новую таблицу используя уже существующую таблицу, в которой есть столбец типа *(CREATE TABLE....AS SELECT...)* или записывать новые строки посредством существующих, когда в них есть данные типа *(INSERT INTO... SELECT...);*

⇒ нельзя использовать его для объявления переменной или аргумента в PL/SQL программной конструкции документов или массивов двоичных данных. Данные типа *RAW* и *LONG RAW* эквивалентны соответственно *VARCHAR2* и *LONG* по отношению к длине данных и по отношению к обмену данными в сети через *SQL*NET*. Вспомогательные программы Import и Export не преобразуют знаков во время обмена, если таблица знаков баз данных и пользовательская таблица знаков, установленная параметром *NLS_LANGUAGE* команды *ALTER SESSION*, различаются.

Тип данных *ROWID* - это уникальный идентификационный номер, который связан с физическим адресом строк. Любой строке в некластерной таблице присваивается уникальный *ROWID.* В кластерных таблицах строки различных таблиц, записанные в одном и том же блоке данных, имеют один и тот же *ROWID.* Любая таблица в ORACLE имеет внутренний псевдостолбец с именем *ROWID,* который не выводится при выполнении SQL - команд *SELECT*FROM...* или *DESCRIBE.* Он, однако, может выводиться с использованием зарезервированного ключевого слова *ROWID* в качестве наименования столбца.

Пример: Команда *SELECTROWID, names FROM readers;* выводит

ROWID	NAMES
00000DD5.0000.0001	Иванов
00000DD5.0001.0001	Петров
00000DD6.0002.0002	Алексеев

Физический адрес строк выводится в шестнадцатеричном формате и имеет три части: блок, номер строки и номер файла. Блок, строку и файл можно выводить отдельно командой *SELECT ROWID.* Пример: *SUBSTR(ROW1D,15,4) "FILE", SUBSTR(ROWID, 1,8) "BLOCK", SUBSTR(ROWID,10, 4) "ROW" FROM readers;*

ROWID	FILE	BLOCK	ROW
00000DD5.0000.0001	0001	00000DD5	0000
00000DD5.0001.0001	0001	00000DD5	0001
00000DD6.0002.0002	0002	00000DD6	0002

Количество файлов, которые сохранены в данной таблице можно вывести командой

SELECT COUNT(DISTINCT(SUBSTR(ROWID,15,4))) "FILES" FROM eaders

FILES 2
2

Тип данных *MLSLABEL* имеет переменную длину максимум до 255 байтов, которая не задаётся - это указатель, который определяет путь к двоичной метке записанной в словаре данных. Употребление типа данных *MLSLABEL* позволяет сократить объем используемой дисковой памяти, так как двоичные метки операционной системы обычно типа данных является необходимость сокращения объема использованной дисковой памяти, так как двоичные метки операционной системы обычно бывают очень длинными. *MLSLABEL* является представительной меткой, которая сохраняется в таблице вместо двоичной метки. Во время создания любой таблицы в ORACLE, к ней автоматически добавляется столбец *ROWLABEL* типа *MLSLABEL*.

Для преобразования данных из одного типа в другой можно использовать функции *TO__NUMBER, TO__CHAR, TO__DATE, CHARTOROWID, ROWIDTOCHAR, HEXTOCHAR* и *CHARTOHEX.*

Во время присваивания ORACLE автоматически преобразует типы данных следующим образом:

- *VARCHAR2* или *CHAR* в *NUMBER;*
- *NUMBER* в *VARCHAR2;*
- *VARCHAR2* или *CHAR* в *DATE;*
- *DATE* в *VARCHAR2;*
- *VARCHAR2* или *CHAR* в *ROWID;*
- *ROWID* в *VARCHAR2;*
- *VARCHAR2* или *CHAR* в *HEX;*
- *HEX* в *VARCHAR2.*

При вычислении выражений ORACLE автоматически преобразует типы данных следующим образом:

- *VARCHAR2* или *CHAR* в *NUMBER;*
- *VARCHAR2* или *CHAR* в *DATE.*

2.2. Таблицы

Таблица - это основная часть данных, которые сохраняются в базе данных ORACLE и в ней содержится вся доступная пользователю

информация. Любая таблица задается именем таблицы и определенной совокупностью столбцов. Любой столбец, со своей стороны, определяется именем столбца, типом данных, шириной (ее можно определить автоматически типом данных, например, *DATE),* размером и точностью данных типа *NUMBER.* В процессе использования таблицы, если это необходимо, можно изменять ширину или размер и точность столбцов. Если таблица создана, в ней можно записывать строки с допустимыми данными. Строки таблицы можно обновлять и стирать.

Рассмотренные в 1.2. таблицы можно задать следующим образом. Таблицы кафедр с наименованием *CHAIRS* и столбцы:

- *CHAIRNO NUMBER (3)* - номер кафедры;
- *DEPART VARCHAR2(20)* - имя факультета;
- *NAME VARCHAR2(20)* - имя кафедры;

Таблица CHAIRS

CHAIRNO	NAME	DEPART
10		Информатика
20		Математика

Таблица преподавателей с наименованием *READERS* и столбцы:

- *CHAIRNO NUMBER (3)* - номер кафедры;
- *READERNO NUMBERQ)* - номер преподавателя кафедры;
- *SCDEGREE VARCHAIR2(8)* - научное звание;
- *NAMES VARCHAR2(30)* - фамилия, имя, отчество преподавателей.

Таблица студентов с наименованием *STUDENTS* и столбцы:

- *DEPTNO NUMBER(5)* - факультетский номер студента;
- *NAMES VARCHAR2(30)* - фамилия, имя, отчество студентов;
- *SPES VARCHAR2(20)* -специальность;
- *SEMNUMBER(3)* - номер семестра.

Таблица оценок с наименованием *GRADES* и столбцы:

- *DEPTNO NUMBER(5)* - факультетский номер студента;
- *READERNO NUMBER(2)* - номер преподавателя, который экзаменовал;
- *SUBJECT VARCHAR(40)* - наименование учебной дисциплины;
- *GRADE NUMBER(3,l)* -балл;
- *SEM NUMBER(2)* - семестр, в котором поставлен балл.

Значения в строках таблицы должны вводится в том же порядке, в котором находятся столбцы при задании таблицы. Когда в таблице добавляется новый столбец, он становится последним. Тип значений в строках должен соответствовать заданному для столбцов типу данных. Есть возможность в некоторых строках данного столбца, не вводить

значения. Тогда в этих строках в соответственных столбцах можно записать подразумевающиеся значения, а если такие не определены - записывается *NULL,* т. е. *NULL* показывает отсутствие значения. Например, для столбцов *CHAIRNO* таблицы *READERS* можно задать подразумевающееся значение 10. В строках таблицы *CHAIRS,* приведенных в примере, в столбце *NAME* нет значения и там записано *NULL.*

Из заданной таблицы можно извлекать разнообразные данные. Например, можно выводить справки для всех строк, или для части строк, которые отвечают некоторым условиям. Можно считать, что справка - это ответ на запрос об интересующих пользователя данных. В справке можно вывести значения всех столбцов или только части столбцов данной таблицы. Например, из таблицов с балами можно вывести только столбцы для факультетского номера студента *DEPTNO,* имя учебной дисциплины *SUBJECT* и бал *GRADE.* В одной справке можно вывести и данные из различных столбцов различных связанных таблиц. Например, в рассмотренной таблице можно прибавить столбец для имени студента *NAMES* из таблицы *STUDENTS* соответствующий факультетскому номеру студента. В этом случае связь между таблицей студентов и таблицей оценок производится на основе факультетского номера.

Для более надежного управления табличными данными, в рабочем порядке, можно задавать общие ограничения и триггеры для таблиц, таких как *NOT NULL, UNIQUE* и др. Например, для всех столбцов таблицы баллов *GRADE* можно задать ограничение *NOT NULL,* и таким образом, обеспечить невозможность записи строки, если до того не было задано значение хотя бы одного столбца.

2.3. Виды

Обычно вид - это представление данных одного или нескольких столбцов одной или более таблиц. Вид можно представить и как справку запроса для интересующих пользователя данных. С точки зрения реляционной алгебры вид - это проекция данной таблицы. В действительности, виды не содержат данные, они получают свои данные из таблиц, на основе которых созданы. Эти таблицы считаются базовыми для соответствующего вида. Например, преподаватели кафедры Информатики можно представить с помощью вида *READERSINF,* который включает столбцы *READERNO, SCDEGREE* и *NAMES* базовой таблицы *READERS.* Этот вид можно представить и как результат запроса *SELECT readerno, scdegree, names FROM readers WHERE chairno-10;*

Виды используются чаще всего для:

⇒ введения дополнительного уровня защиты информации;

⇒ повышения степени интегрированности информации (на базе связывания таблиц);

⇒ упрощения команд для пользователя;

⇒ представления данных в разнообразной форме;

⇒ сохранения запросов.

2.4. Индексы

Индексы - это оптимизирующие структуры связанные с таблицами. Они создаются всегда, когда необходимо увеличить скорости выполнения SQL команд. В процессе поиска, ORACLE может использовать несколько или все наличные индексы для более быстрого доступа к данным. Индексы задаются одним или несколькими столбцами таблицы. Когда индекс определяется несколькими столбцами, находящимися в определенном порядке, тогда он называется составным или связанным. Такой индекс увеличивает быстродействие во время поиска данных при заданном условии, включающем все или значительное количество столбцов.

Индексы можно эффективно использовать, когда прикладные программы часто производят запросы на уровне строк (например, запрос о всех преподавателях с научным званием профессора). Для данной таблицы можно создать один или несколько индексов. Например, в таблице баллов *GRADES* можно создать индекс по факультетскому номеру студентов *DEPTNO* для более быстрого доступа к баллу данного студента и индекс по номеру кафедры *CHAIRNO* и номер преподавателя *READERNO* для более быстрого доступа к баллам, поставленным данным преподавателем.

Индексы можно задать как уникальные или как неуникальные. Уникальный индекс гарантирует, что нет двух строк в таблице с дублирующимися значениями в столбце или в столбцах, которые определяют индекс. Такой уникальный индекс должен быть задан столбцом *CHAIRNO* для таблицы *CHAIRS* и столбцами *CHAIRNO* и *READNO* для таблицы *READERS.* Эти индексы не допускают ввода повторяющихся значений, а если такие были до создания индексов, то их, индексы не задаются.

После создания индекса, он поддерживается и используется автоматически ORACLE. Изменения в табличных данных, такие как добавление новых строк, изменение и стирание строк, автоматически отражаются во всех связанных индексах с полной прозрачностью для пользователя. Индексы логически и физически независимы от данных. Они могут создаваться и стираться в любое время, при этом это не сказывается на таблицах или других индексах. Если стереть один

индекс то все приложения будут продолжать функционировать, однако доступ к неиндексированным данным может быть медленным.

Индексы - это структуры баз данных, которые физически сохраняются в ней и определяют местоположение любой строки таблиц, используя значения строк, задающих индекс. Так как индекс определяет данные (их местоположение), его часто путают с понятием ключа. Ключ, в самом деле, определяет данные, однако, определяет их с логической точки зрения и не является отдельной физически сохраняемой структурой. Понятие ключа связано с концепцией целостности данных.

2.5. Последовательности

Последовательность - это список последовательных уникальных значений для записи в арифметических столбцах таблиц БД. Элементы списка - это числовые значения для столбцов одной или нескольких таблиц. Последовательности создаются генератором автоматическим образом.

Предположим, например, что два пользователя, независимо друг от друга, вводят новые имена преподавателей в таблице *READERS*. Не используя последовательность, которая должна генерировать уникальные номера преподавателей, каждый пользователь должен ждать другого, чтобы ввести следующий доступный номер преподавателя. Если в этом случае используется последовательность, то корректные значения для каждого пользователя будут генерироваться автоматически.

Последовательные номера не связаны с конкретной таблицей, они независимы, и одна и та же последовательность может быть использована многими таблицами. После создания последовательности, она становится доступной для различных пользователей, которые могут генерировать действительные последовательности номеров.

2.6. Программные части

Программные части - это сохраненные в базе данных процедуры, функции и пакеты. Процедуры и функции являются совокупностями SQL и PL/SQL команд, которые оформлены как выполняемая часть для решения определенной задачи. Они позволяют комбинировать простоту и гибкость языка структурного программирования с его процедурной функциональностью. Используя PL/SQL, такие процедуры и функции можно определять и сохранять в базе данных для дальнейшего использования. Процедуры и функции, по своей структуре, идентичны, за исключением того, что функции возвращают

только одно значение, а процедуры имеют более сложные связи с остальными частями прикладной программы.

Пакеты обеспечивают метод объединения и сохранения процедур, функций и других пакетных конструкций вместе, как одну часть, в базе данных и предоставляют разработчику прикладной программы организованные средства для увеличения функциональности и производительности работы с базами данных.

2.7. Синонимы

Синоним - это другое имя таблицы, вида, последовательности или программной части. Он не является действительным объектом, но тем не менее указывает на сохраненный в базе данных объект. Синоним можно использовать с целью:

⇒ спрятать действительное имя владельца объекта;

⇒ обеспечить общий доступ к объектам;

⇒ более четко указать место таблицы, вида или программной части отдаленных баз данных при использовании распределенной базы данных;

⇒ упростить использования SQL команд пользователями.

Синоним может быть общим или личным. Отдельный пользователь может создать личный синоним, который доступен только ему. Администратор баз данных часто должен создавать синонимы, чтобы сделать основные схемные объекты общедоступными для всех пользователей БД.

2.8. Кластеры и хэш-кластеры

Кластеры - это оптимизированная технология сохранения табличных данных. Они содержат одну, или, чаще всего, группу таблиц, которые сохраняются вместе. Кластеры позволяют значительно сократить время доступа к диску при чтении и записи строк, связанных с общими столбцами таблиц. Уменьшение времени доступа достигается вследствие того, что строки с одинаковыми значениями их столбцов сохраняются вместе. Связанные столбцы в таблицах называются кластерным ключом. Кластерный ключ индексирован, что позволяет в процессе чтения и записи строк кластера использовать минимум операций ввода-вывода. Так как данные в кластерном ключе индексированного (нехэшированного) кластера сохранены однократно для многих таблиц, то кластеры могут сохранять табличные данные более эффективно, чем индивидуальное сохранение таблиц. Ниже приводится иллюстрация преимущества физического сохранения данных с использованием кластеров.

Таблица CHAIRS

CHAIRNO	NAME
10	Информатика
20	Математика

Таблица READERS

CHAIRNO	READERNO	SCDEGREE	NAMES
10	10	ассистент	Иванов
20	10	доцент	Петров
10	20	профессор	Борисов
20	20	ассистент	Васильев

Кластер

CHAIRNO - КЛАСТЕРНЫЙ КЛЮЧ

10 NAME

Информатика

READERNO	SCDEGREE	NAMES
10	ассистент	Иванов
20	профессор	Борисов

20 NAME

Математика

READERNO	SCDEGREE	NAMES
10	доцент	Петров
20	ассистент	Васильев

Кластеры могут улучшать выполнение операций во время изменения данных в зависимости от их распределения, и от того какие SQL - команды используются наиболее часто. В частности, кластерные таблицы, из которых выводятся данные, более выгодны для пользования, так как строки из связанных таблиц читаются одновременно одной и той же операцией ввода-вывода.

Хэш-кластеры, как и обычные кластеры, тоже являются оптимизированным способом хранения данных. При этом используются хэш-функции. В обычных кластерах данные организованны по значениям в столбцах, формирующих кластерный ключ. В хэш - кластерах строки организованы по значениям кластерного ключа, которые формируются в результате использования хэш-функции. Все строки с одинаковыми значениями хэш-ключа сохраняются вместе на диске.

Метод хэш-кластеров лучше чем метод использования индексов или обычного индексного кластера, в тех случаях, когда в таблице необходимо часто искать данные по запросам одинакового содержания, (например, все строки кафедры 10). В таких случаях результат, который выдается хэш-функцией для соответстующего значения кластерного ключа, указывает на область диска, в которой сохраняются данные строки.

2.9. Блоки с данными, экстентами и сегментами. Связи в базах данных

ORACLE всегда точно контролирует использование дискового пространства логическими структурами, которые включают блоки с данными, экстенты и сегменты.

На самом нижнем уровне логического пространства, данные ORACLE-базы сохраняются в блоке данных. Один блок данных - это определенное количество байтов физического пространства диска, выделенного для базы данных. Размер блока данных определяется для любой базы данных, в процессе ее создания. Система управления базой данных использует и распределяет свободное пространство диска на основе блоков данных.

Следующий уровень логического пространства базы данных называется экстентом. Он представляет собой определенное количество последовательных блоков, полученных в одном отдельном распределении и используемых для сохранения данного типа информации.

Уровень логического пространства для сохранения экстентов называется сегментом. Сегмент - это установленные экстенты, распределенные для определенной логической структуры. Различные типы сегментов включают:

- *сегмент данных.* Любая не кластерная таблица - это сегмент данных. Все данные таблицы сохраняются в экстентах ее сегмента. Любой кластер имеет сегмент данных. Данные любой таблицы кластера сохраняются в сегменте данных кластера;
- *индексный сегмент.* Любой индекс имеет соответствующий индексный сегмент, который сохраняет его данные;
- *rollback сегмент* - сегмент для восстановления предыдущего состояния. Администратор базы данных создает один или несколько *rollback* сегментов для сохранения подлежащей восстановлению информации БД. Эта информация используется: для генерирования информации при согласованном чтении из базы данных; при полном восстановлении БД; для восстановления предыдущего состояния БД при незаконченных транзакциях со стороны пользователя (транзакция - это последовательность логически связанных изменений сохраняемых данных в БД);
- *временной сегмент* создается базой данных ORACLE тогда, когда SQL команда нуждается во временной рабочей области для полного выполнения. Когда команда выполнится, экстенты временного сегмента возвращаются в систему для будущего использования.

ORACLE распределяет дисковое пространство для всех типов сегментов на отдельные экстенты. Когда существующие экстенты для данного сегмента полны, ORACLE определяет другой экстент для нуждающегося сегмента. Так как экстенты распределяются при необходимости, они могут и не сохраняться последовательно на диске.

2.10. Табличные пространства, файлы

Любая ORACLE БД имеет один или больше физических файлов с данными. Все данные, в том числе и логические структуры (таблицы, индексы и другие) физически сохраняются в файлах с данными, которые определены для БД. Файлы с данными имеют следующие характеристики:

$\Rightarrow$ файл с данными может быть связан только с одной базой данных;

$\Rightarrow$ после создания, файл с данными не может изменять свою размерность;

$\Rightarrow$ один или больше файлов с данными формирует логическую часть физического пространства БД, называемую табличным пространством.

При создании определенной ORACLE базы данных формируется и табличное пространство с именем *SYSTEM.* Остальные табличные пространства создаются привилегированными пользователями при необходимости. Обычно это администратор базы данных.

2.10.1. Использование файлов с данными. Данные, которые содержатся в файлах с данными читаются, при нормальной работе, из БД в кэш памяти ORACLE. Предположим, что пользователю требуется доступ к некоторым данным в таблице БД. Если требуемая информация не находится уже в кэш памяти, она читается из соответствующего файла с данными и записывается в память. Модифицированные или новые данные не записываются сразу в файл с данными. Для уменьшения количества записей на диске и увеличения производительности, данные из памяти записываются в соответствующий файл с данными все сразу, как это определено в DBWR-фоновом процессе ORACLE.

2.10.2. Файл сохранения данных и структур объектов до последнего их изменения. Для любой ORACLE базы данных создаётся один или больше файлов, в которых сохраняются данные и структуры объектов до последнего их изменения (лог-файлы или Redo Log Files). Все производимые в процессе работы с базой данных изменения надлежащим образом записываются в этот файл. Если возникнет сбой, или потеря данных, то сделанные изменения и прежнее состояние базы данных можно получить из лог-

файла. Следовательно, эти файлы используются для восстановления базы данных. Лог-файлы имеют очень важное значение для надежной работы БД, в частности, для защиты информации. В связи с этим ORACLE всегда создает резервную копию лог-файла на диске.

Информация в лог-файлах, главным образом, используется для восстановления БД после системного сбоя, из-за которого не произвелась успешная запись в файл с данными. Другим примером является неожиданное прерывание питания персонального компьютера. Это вызвет сбой операций с базой данных и данные из оперативной памяти не смогут быть записаны в файлы с данными. ORACLE автоматически использует информацию, которая содержится в лог-файлах для немедленного восстановления БД и ее файлов с данными такими, какими они были до сбоя питания. Процесс использования этих файлов для восстановления определенного состояния называется возвращением назад (Redo).

2.10.3. Контрольные файлы. Любая ORACLE база данных имеет контрольный файл. Он содержит физическую структуру базы данных и может включать следующие виды информации: имя базы данных; имена и расположение файлов с данными БД и файлы для откорректированных данных; время создания базы данных. ORACLE всегда сохраняет копию контрольного файла с целью его защиты.

Каждый раз, когда запускается ORACLE, его контрольный файл используется для идентификации БД, которая должна быть открытой для выполнения операции над данными и файлами с откорректированными данными. Если физическое состояние БД изменится (например, будет создан новый файл с данными или файл с откорректированными данными), ORACLE автоматически модифицирует контрольный файл БД с отображением всех перемен.

2.11. Словарь данных

Любая ORACLE база данных имеет словарь данных. Словарь базы данных включает описание созданных таблиц; видов и других схемных объектов, которые используются как постоянная, неизменная информация для БД. Например, словарь данных сохраняет информацию о логической и физической структуре БД. В дополнении к этой ценной информации словарь данных сохраняет также следующую информацию: действительные пользователи БД; ограничения целостности БД; какое пространство выделено для схемных объектов и какая его часть использована.

Словарь данных формируется при создании базы данных. Для точного отображения состояния базы данных этот словарь в ORACLE автоматически изменяется в ответ на специфические

действия, например, изменение структуры базы данных. Словарь данных критичен и очень важен для работы базы данных, так как она постоянно читает его содержание в процессе записи, проверки и текущей работы. Например, во время работы базы данных ORACLE читает словарь данных, чтобы проверить существуют ли схемные объекты и действительно ли пользователи имеют право доступа к ним.

В словаре данных включен вид *VIEWS,* который со своей стороны содержит ряд видов из словаря базы данных. Имена этих видов можно вывести командой: *SELECT viewname FROM views;*

Например, следующие виды содержат информацию:

- *TAB* - о таблицах и видах в базе данных;
- *INDEXES* - о индексах в базе данных;
- *DBLINKS* - о связи с базами данных распределенной базы данных;
- *SYNONYMS* - о синонимах;
- *PUBLICSYN* - о публичных синонимах;
- *ALLSEQUENCES* - о последовательностях;
- *USERTABLESPACES* - о табличных пространствах пользователя;
- *ALLTRIGGERS* - о триггерах в базе данных;
- *USEROBJECTS* - о объектах пользователя;
- *USERSOURCE* - определяет пользовательские функции, процедуры и пакеты;
- *DICTIONARY* - о системных таблицах, включенных в словарь базы данных.

Кроме перечисленных видов есть еще ряд других, из которых можно получить полезную информацию о базе данных.

3. Работа с данными в базах данных ORACLE

3.1. SQL - структурный язык для работы с данными

SQL - простой и мощный язык для доступа к базе данных, который стандартизован для систем управления реляционными базами данных. Язык SQL разработан фирмой Oracle Corporation для ORACLE, и он полностью совместим с ANSI/ISO стандартом для SQL. Все операции над информацией в определенной ORACLE-базе данных выполняются SQL-командами. Команда должна быть полным SQL-предложением, как в следующем примере: *SELECT deptno, reademo, names FROM readers;*

Только полная SQL-команда может быть выполнена. Если используется только фрагмент команды, то генерируется сообщение об ошибке (ORACLE сообщает, что нужен еще текст, чтобы SQL- команда могла выполниться). Примером неправильной (неполной) комады является:

SELECT names

SQL-команда может быть сконструирована как очень простой, так и очень мощной компьютерной программой или инструкцией. Команды SQL могут быть разделены на пять категории: *DDL (Data Definition) команды, DML(Data manipulation) команды, команды контроля транзакций, команды контроля сессии* и *связывающие команды.*

DDL-команды определяют и изменяют объекты или стирают объекты, когда они больше не нужны. Они включают также команды, которые позволяют пользователю предоставить другим пользователям привилегии и права для доступа к базе данных и специфическим ее объектам.

DML-команды обрабатывают данные в базе данных. Сюда относятся такие команды как поиск, вставка, изменение и стирание строк таблицы. Закрытие таблицы или вида также являются DML-командами.

Команды контроля транзакций управляют переменами, которые инициируются DML-командами. Они позволяют пользователю или разработчику прикладных программ группировать операции в логически связанные действия, которые называются транзакциями. Примерами таких команд являются *COMMIT, ROLLBACK* и *SAVEPOINT.*

Команды контроля сессии позволяют пользователю контролировать правильность текущей сессии. Они выполняют разрешающие (запрещающие) функции (роли) и языковые установки. Такими командами являются *ALTER SESSION* и *SET ROLE.* Сюда относится и команда *ALTER SYSTEM.* Она изменяет характеристики ORACLE Server сессии, позволяют определить количество распределенных серверов, количество сессий ORACLE запущенных на серверах и др.

Связывающие SQL-команды позволяют объединить DDL, DML-команд и команды контроля транзакций в процедурную языковую программу. Такими, например, являются команды *OPEN, CLOSE, FETCH* и *EXECUTE.*

3.2. Транзакции

Транзакция - это логически определенная часть работы, охватывающая одну или несколько SQL-команд, выполняемых одним и тем же пользователем. Согласно стандарта ANSI/ISO SQL, с которым ORACLE совместим, транзакция начинается с первой выполненной командой пользователя и заканчивается тогда, когда сессия будет полностью выполнена или, по каким-либо причинам, прекращена пользователем. Рассмотрим процесс накопления данных в базе данных при проведении банковских операций. Когда клиент банка переводит деньги со сберегательного счета на расчетный счет, транзакция может

состоять из трех операций: коррекция (уменьшение) сберегательного счета, коррекция (увеличение) расчетного счета и запись транзакции в журнал транзакций. ORACLE должен гарантировать что все три SQL-команды выполнены, чтобы поддерживать точный баланс счетов. В том случае, когда что-то помешало выполнению одной из команд транзакции (например, аппаратурный сбой), остальные команды транзакции должны быть отменены. Это называется "возвращение назад" (rollback).

Выполнение транзакции иллюстрирует приведенная ниже схема.
Начало транзакции
Уменьшение сберегательного счета
UPDATE saving_account sety balance=balance-500 where account=3208;
Увеличение счета расчета
UPDATE checking___account sety balance=balance+500 where account=3209;
Запись в журнале транзакций
INSERT INTO journal VALUES(journal.NEXTVAL,'lB',3209,3208,500);
Конец транзакции
COMMIT WORK
Транзакция закончена.

Операции, которые предписывает данная транзакция могут быть полностью выполнены, или отменены (прерваны) по какой-либо причине. После того как текущая транзакция закончена, начинается выполнение следующей транзакции с первой ее SQL-команды.

Полное выполнение транзакции приводит к окончательному фиксированию всех изменений (они становяться необратимыми), вызванных содержащимися в транзакции SQL-командами. Эти изменения уже становятся "видимыми", доступными для транзакций других пользователей.

Возвращение (прерывание) транзакции отменяет все изменения, которые были произведены содержащимися в ней SQL-командами. После того как выполнение транзакции прервано данные восстанавливаются и приводятся в состояние, которое соответствует состоянию до активизации транзакции (как будто бы SQL-команды текущей транзакции никогда и не выполнялись).

3.3. Точки сохранения

Иногда приходится использовать длинные транзакции, которые содержат много SQL-команд. Это создает некоторые проблемы и неудобства в rex случаях, когда выполнение подобных транзакций не заканчивается полностью. Как было сказано выше, в этом случае все внесенные изменения теряются. Это особенно неприятно, когда

транзакция имеет большую длину. С целью предотвращения этого недостатка в теле транзакции можно указать (вставить) промежуточные маркеры, которые называются точками сохранения. Они используются для деления тела транзакции на более мелкие части. Таким образом, при возникновении сбоя теряются только те данные, изменения и обработки, которые были сделаны только после последней точки сохранения. Все изменения, которые имели место до нее сохраняются, что несомненно практически очень удобно, тем более, что имеется возможность вернуться назад до произвольной точки сохранения Это предоставляет также возможность легко устранять возникшие и длинных транзакциях ошибки.

3.4. Согласованность данных в транзакциях

Транзакции предоставляют пользователям базы данных и прикладных программ возможности гарантированного согласования изменения данных так как SQL- команды в транзакциях группируются по логическому признаку. Все действия в одной транзакции должны быть согласованны и логически связаны. Данные во всех связанных таблицах находятся в согласованном состоянии до начала транзакции и после ее завершения.

Транзакция должна быть согласована с теми SQL-командами, которые охватывают одно совместное изменение данных. Вернемся к примеру выполнения банковской операции. Транзакция здесь заключается в переводе денег из одного счета на другой. Она включает уменьшение одного счета (одна SQL-команда), увеличение другого счета (одна SQL-команда) и запись в журнале транзакций (одна SQL-команда). Все указанные действия либо должны быть произведены в полном объеме, либо не произведены вообще. Нельза допускать, в частности, уменьшения одного счета без коррекции (увеличения) другого. Другие несвязанные действия, такие как открытие нового счета, не должны включаться в эту транзакцию перевода денег. Они должны оформляться в виде отдельной транзакции.

3.5. PL/SQL

Процедурный язык PL/SQL является расширением языка SQL, и разработан фирмой ORACLE Corporation. PL/SQL объединяет простоту и гибкость SQL с процедурной функциональностью структур программных языков, таких как *IF... THEN, WHILE, LOOP* и другие. В процессе разработки прикладных программ для работы с базой данных, разработчик может использовать следующие преимущества PL/SQL:

$\Rightarrow$так как PL/SQL-программу можно сохранять прямо в базе данных, то интенсивность обмена информацией между прикладными

программами и базой данных уменьшается, в результате чего производительность системы увеличивается;

$\Rightarrow$ доступ к данным можно контролировать самой PL/SQL-программой. В этом случае пользователи PL/SQL могут получать доступ к данным только тогда, когда это предусмотрено ее разработчиком;

$\Rightarrow$ PL/SQL блоки могут извлекаться из прикладных программ и направляться в базу данных для выполнения комплексных операций без усложнения обмена. Когда данная PL/SQL процедура не сохраняется в базе данных, программы могут послать SQL блоки в БД, как отдельные SQL команды, что в значительной мере уменьшает объем обмениваемой информации.

Ниже описаны различные программные части, которые можно определять и сохранять прямо в базе данных.

Процедуры и функции. Процедуры и функции создаются на основе соответствующих SQL и PL/SQL-команд и образуют программную часть для решения специфической проблемы или для выполнения заданного набора связанных задач. Они сохраняются в компилированной форме в базе данных и могут использоваться пользователем или прикладной программой. Процедуры и функции по своей структуре идентичны, с той лишь разницей, что функция всегда возвращает пользователю или вызывающей программе одно значение, а процедура может возвращать более сложные структуры данных.

Пакеты. Эти программные части дают возможность объединения и сохранения связанных процедур, функций, переменных и других пакетных конструкций вместе в базе данных. Пакеты позволяют администратору базы данных и разработчику прикладных программ лучше организовать свою работу и свои программы. Использование паектов приводит к увеличению функциональности (например, можно объявить глобальные пакетные переменные и использовать их в любой процедуре пакета) и производительности (например, возможно одновременно компилировать и загружать в память все объекы пакета).

Триггеры базы данных. ORACLE позволяет писать процедуры, которые антоматически выполняются в результате вставки, изменения или стирания строк таблицы. Эти процедуры называются триггерами базы данных. Их можно использовать в различных случаях управления информацией в БД. Например, для автоматического генерирования даты, для проверки модификации даты, реализации сложных ограничений целостности данных и использования сложных процедур по обеспечению защиты информации.

3.6. Целостность данных

Очень важно гарантировать, что к данным будут надежно применяться те рабочие правила, которые предусмотрены администратором базы данных или разработчиком прикладной программы. Примером такого рабочего правила является следующее: строки таблицы *GRADES* не могут содержать числовые значения больше *5* в столбце *GRADE*. Если команды *INSERT* или *UPDATE* пытаются нарушить это правило целостности, ORACLE должен отменить неправомерную команду и вернуть сообщение об ошибке пользователю или программе. ORACLE предлагает ограничения целостности и триггеры БД как средства для решения задач управления данными в БД, удовлетворяя трбованиям правил целостности.

Ограничения целостности. Реализуются на основе декларативного метода определения рабочих правил для соответствующих столбцов таблицы. При задании ограничения целостности следует иметь ввиду, что:

$\Rightarrow$ если ограничение создается для таблицы и существуют табличные данные, для которых ограничение не соблюдается, то оно не может быть применено;

$\Rightarrow$ после того как ограничение определено, если кто-то, в результате выполнения команды манипулирования с данными, его нарушит, то команда отменяется и возвращается сообщение об ошибке.

Ограничения целостности определяются таблицей и сохраняются как часть определения таблицы, непосредственно в словаре данных базы данных, так что все прикладные программы БД должны придерживаться установленных правил. Если возникает необходимость изменения правила, то изменение должно быть произведено на уровне базы данных, а не многократно для каждой прикладной программы.

ORACLE поддерживает следующие ограничения целостности:

- *NOT NULL* - не допускает пустое значение в столбце таблицы;
- *UNIQUE* - не допускает дублирование значений в данном столбце или в общем результате, который получается объединением значений в установленных столбцах;
- *PRIMARY KEY* - не допускает дублированные и пустые значения как в столбцах, так и в полученном результате объединения значений в установленных столбцах;
- *FOREIGN KEY* - требует чтобы любое значение в столбце или общее значение в установленных столбцах, соответствовали значениям в связанном столбце или значениям столбца другой таблицы, которым установлено ограничение UNIQUE или PRIMARY KEY. Например,

значение в столбце DEPTNO с таблицы GRADES должно соответствовать существующему значению столбца DEPTNO таблицы STUDENTS и общее значение столбцов CHAIRNO и READERNO таблицы GRADES должно соответствовать общему значению столбцов CHAIRNO и READERNO таблицы READERS. Ограничение через FOREIGN KEY определяет действия, обеспечивающие целостность связанных данных, особенно в случаях, когда определяющие данные изменяются;

- *CHECK* - определяется логическим выражением и не допускает значений которые не удовлетворяют логическому выражению в ограничении.

Триггеры базы данных позволяют определять и применять ограничения целостности, но они не идентичны с ограничениями целостности. Триггер базы данных, определённый для применения правила целостности, не всегда проверяет данные таблицы, в то время как ограничение целостности всегда применяется. По этой причине рекомендуется использовать триггеры базы данных только тогда, когда правило целостности не может быть выполненно с ограничениями целостности.

Ключи. Термин ключ используется в определениях различных типов ограничений целостности. Ключ - это столбец, который включен в определения известных типов ограничений целостности. Ключи описывают связи между различными таблицами и столбцами в реляционной базе данных. Имеются различные типы ключей:

- первичный ключ - столбец или несколько столбцов, включенные в определение табличного ограничения целостности типа *PRIMARY KEY.* Значения первичного ключа уникальные и определяют строки в таблице. Для определенной таблицы можно задать только один первичный ключ;
- уникальный ключ - столбец или несколько столбцов, которые включаются в строку задания ограничения *UNIQUE;*
- внешний ключ - столбец или несколько столбцов, которые включаются в строку задания ограничения, связывающего соответствия *FOREIGN KEY.*
- связанный ключ - первичный или уникальный ключ из другой таблицы, который связан по соответствию с значением внешнего ключа.

3.7. Согласованность и постоянство данных

В этом разделе коротко рассматриваются механизмы, которые предлагаются программным обеспечением ORACLE для выполнения

следующих важных требований, предъявляемых к системам обработки информациии:

$\Rightarrow$ данные должны читаться и модифицироваься согласовано - в своей логической непротиворечивости;

$\Rightarrow$ при многопользовательских системах должна быть обеспечена максимальная согласованность данных;

$\Rightarrow$ многие пользователи (прикладные программы) требуют обеспечения высокого качества обработки информации при максимально возможной производительности.

Согласованность. Одной из самых важных забот многопользовательских систем управления базами данных является управление и контроль согласованности при одновременном доступе большого количества пользователей к одним и тем же даным базы. При отсутствии подходящего контроля согласованности данные могут быть обновлены или изменены неправильно, что нарушит их интегрированность и полноту. Если многие пользователи имеют доступ к данным, то один из способов обеспечения согласованности это введение дисциплины обслуживания, при которой каждый пользователь ждал бы своей очереди. Целью системы управления базой данных является сокращение времени ожидания в очереди до такой степени, чтобы пользователи чувствовали себя независимыми друг от друга и обслуживание остальных не сказывалось на обслуживании данного пользователя. Все команды языков управления базами данных должны работать таким образом, что, как вмешательство и помехи между ними, так и взаимная зависимость были бы как можно меньшими. Они должны также обеспечивать защиту взаимодействия между согласованными транзакциями. Нельзя жертвовать ни качеством выполнения запросов, ни интегрированностью данных.

ORACLE удовлетворяет этим требованиям путем использования разных типов защиты и многовариантной модели согласования. Оба метода обсуждаются ниже, используя концепции транзакции, так как транзакции являются типичным случаем необходимости удовлетворения требований как согласованности, так и постоянства данных.

Постоянство при чтении (консистентное чтение). В более общем случае, постоянство или консистентность обеспечивает такой режим работы, при котором данные, которые рассматриваются, или изменяются пользователем, не могут быть изменены другим пользователем до тех пор, пока первый пользователь не закончит свою

работу с этими данными. Консистентное чтение, поддерживаемое ORACLE, имеет следующие характеристики:

⇒ гарантирует, что совокупность данных, "рассматриваемая" какой-либо командой, консистентная по отношению к отдельной сущности во времени и не меняется во время выполнения команды (консистентное чтение на уровне команды);

⇒ обеспечивает такой режим работы, при котором пользователь, который читает данные из БД не должен ждать других пользователей, которые хотели бы записать или прочитать те же самые данные;

⇒пользователь, который записывает данные в базу данных не должен ждать других пользователей, которые читалют те же самые данные;

⇒ пользователь, который записывает данные в базу данных должен ждать только тех пользователей, которые пытаются обновить идентичные строки в согласованных транзакциях.

Самый простой способ понимания встроенной в ORACLE-ситемы консистентного чтения, это представить себе, что каждый пользователь имеет собственную копию БД (собственный экземпляр). Следовательно можно говорит о многоэкземплярной консистентной модели.

Консистентность чтения, rollback-сегменты и транзакции. Чтобы управлять многоэкземплярной консистентной моделью ORACLE должен создать консистентно-читаемую совокупность данных во время чтения из данной таблицы и одновременной ее актуализации (запись в таблицу). Когда производится актуализация, оригинальные данные (до актуализации) записываются и сохраняются в rollback сегменте базы данных. Пока эта актуализация остается частью неоконченной транзакции, каждый пользователь, который позже запрашивает модифицированные данные, в самом деле может увидеть значения только оригинальных данных - ORACLE использует текущую информацию из системной глобальной области (СГО) и информацию из rollback сегмента, чтобы конструировать консистентно-читаемый вид таблицы данных для нужд запросов. Только тогда, когда транзакция закончена, изменения, ею вызванные, становятся постоянными. "Увидеть" изменения могут только те команды, которые начнутся после завершения транзакции, вызвавшей изменения.

Следует отметить, что транзакция - это ключ к стратегии ORACLE для обеспечения консистентного чтения. Часть выполненных, или невыполненных SQL-команд определяют:

⇒ начальную точку для консистентно читаемых видов, которые генерируются со стороны читающих пользователей;

$\Rightarrow$ когда модифицированные данные становятся видимыми для остальных транзакций базы данных для чтения или актуализации.

Односторонние (read-only) транзакции. По умолчанию ORACLE гарантирует консистентное чтение на уровне команд. В ответ на одиночный запрос пользователю предлагаются данные, которые являются консистентными по отношению к конкретному моменту времени. В ряде случаев, однако, возможно потребовать соблюдения консистентности чтения на уровне транзакции - возможность посылать множество запросов в рамке одной транзакции. Все они выполняются в условиях консистентного чтения, по отношению к одному и тому же моменту времени таким образом, что в данной транзакции они не "видят" эффект интервенции законченных транзакций.

Если необходимо осуществить определенное количество запросов к множеству таблиц и если при этом не производится актуализация, то рекомендуется использовать односторонную (read-only) транзакцию. После установления факта, что транзакция односторонняя, могут быть выполнены столько запросов, сколько нужно для каждой таблицы, зная что результаты каждого запроса консистентые по отношении к одному и тому же моменту времени.

Закрытие. ORACLE использует замки, чтобы управлять одновременным доступом к данным. Замки - это механизмы, которые предохраняют от таких взаимодействий между пользователями, осуществляющиими доступ к ORACLE, которые привели бы к разрушению данных. Любое взаимодействие, которое приводит к неправильной актуализации данных или к неправильному переключению структур данных называется разрушающим.

Замки используются для реализации следующих двух важных характеристик баз данных:

• консистентность - режим постоянства (неизменчивости) данных, невозможность их изменения со стороны других пользователей, пока данный пользователь не закончит работу с данными;

• интегрированность - базы данных и структуры должны отображать все произведенные над ними изменения в правильной последовательности.

Замки гарантируют интегрированность данных, обеспечивая максимальную согласованность доступов к базе, при практически неограниченном количестве пользователей. Закрытие в ORACLE может производиться как автоматическим, так и ручным способом.

Автоматическое закрытие. Оно производится автоматически и не требует от пользователя предпринимать каких-либо действий.

Безусловное закрытие может задаваться с помощью соответствующих SQL- команд в тех местах и случаях, когда это необходимо, в зависимости от требуемых действий. ORACLE 7 располагает усовершенствованным модулем для управления защитой, который автоматически производит закрытие данных таблицы на уровне строк. Это сводит к минимуму действие эффекта "состезания одинаковых данных". Модуль управления закрытием в ORACLE имеет несколько различных типов закрытий строк, в зависимости от того, какой тип операции устанавливает закрытие. В общем, можно указать два типа закрытий: исключительное и раздельное. Одно и только одно исключительное закрытие применяется к одному ресурсу, например, к строке таблицы, в то время как множество раздельных закрытий могут быть применены к одному ресурсу. Оба типа закрытий разрешают запросы к закрытому ресурсу, но запрещают другие действия (такие как актуализацию, стирание и др.).

Ручное закрытие - в некоторых случаях пользователь может захотеть удалить какое-то подразумеваемое закрытие. ORACLE разрешает ручное удаление автоматически установленного закрытия как на уровне строк, так и на уровне таблицы.

4. Архитектура системы ORACLE 7

В этом разделе рассматривается структура памяти и процессов, которые используются в ORACLE для управления базой данных. Обсуждаются следующие возможности системы ORACLE 7:

$\Rightarrow$ совместного использования общей базы данных многими пользователями;

$\Rightarrow$ достижение высокого качества выполнения запросов пользователей.

4.1. Структуры памяти и процессы

Механизмы действия ORACLE основываются на использовании структуры памяти и процессов. Процессы - это работы или задачи, выполняемые компьютером, и связанные с прикладной программой.

ORACLE создает и использует несколько областей памяти для мыполнения различных видов работ, две из которых основные: системная глобальная область (включает базу данных, буферы сохранения произведенных изменений для целей последующего воссановления иредидущего состояния (redo) и общее, разделённое между пользователями, пространство - общий фонд) и глобальная область прикладных програм.

Системная глобальная область (СГО) - это распределенный участок памяти, который содержит данные и управляющую информацию. СГО и фоновые процессы конструируют один ORACLE-образец. СГО

размещается в памяти, когда данный образец запускается и освобождает место, когда образец исключается. Каждый запущенный образец имеет свою собственную СГО. Данные в СГО распределяются между пользователями, которые подсоединены к базе данных в текущий момент времени. Для оптимальной работы размер СГО должен быть как можно больше, с тем, чтобы сохранить как можно больше данных, минимизировать объем и интенсивность операций ввода/вывода на диск. Информация, которая сохраняется в СГО разделена на несколько типов структур памяти, а именно: буферы баз данных, буферы восстановления данных (redo - буферы) и общие области. Эти области имеют фиксированный размер и создаются в процессе загрузки образца.

Буферы базы данных - здесь СГО сохраняют те блоки базы данных, которые в последнее время были использованы наиболее часто. Совокупность буферов базы данных образца образует буферный кэш базы данных. Эти буферы могут содержать модифицированные данные, которые все еще не записаны на диске. Вследствие хранения в памяти данных, которые использовались последними (или наиболее часто используются), резко уменьшается объем дисковых операций ввода/вывода. Это приводит к увеличению скорости обработки данных и росту производительности системы в целом.

Буфер восстановления данных (БВД). БВД (redo - буфер) сохраняет изменяемые данные с целью их последующего восстановления - это запись изменений, которые были произведены в базе данных. Redo - буфер имеет постоянный объем и используется, если нужно восстановить базу данных.

Общий фонд - это часть СГО, которая содержит общие конструкции в памяти, например, общие SQL-области. Такая общая SQL-область необходима, например, для обработки каждой SQL-команды, которая задается в базе данных. Эта область содержит синтаксическое дерево и план выполнения соответствующей команды. Она используется множеством прикладных программ (здесь помещаются так же процедуры и функции языка SQL или его расширение PL/SQL), которые задают для выполнения одну и ту же команду, высвобождая, таким образом, больше объема общей памяти для других целей.

Указатель (cursor) - это идентификатор (имя) области памяти, связанной со специфической командой. Несмотря на то, что большая часть пользователей ORACLE рассчитывают на автоматическое задание указателей вспомогательными средствами ORACLE, интерфейс системы предлагает разработчикам прикладных программ возможность управления указателями. Эта

возможность позволяет разработчику прикладной программы, например, контролировать фазы выполнения SQL- команд, и, таким образом, улучшить выполнение програмы.

Программная глобальная область (ПГО) - это выделенная область памяти - буфер, который содержит данные и контрольную информацию серверного процесса. ПГО создается одновременно с началом серверного процесса. Информация в ПГО зависит от конфигурации ORACLE.

4.2. Процессы

Процесс - это "нить управления" или механизм функционирования данной операционной системы в ходе выполнения серии действий. В некоторых операционных системах вместо понатия "процесс", используют термин "работа" или "задача". Процесс, обычно, имеет свою область памяти, где записывает необходимые ему данные.

Система управления базой данных ORACLE имеет две основные группы процессов: пользовательские и системные (ORACLE) процессы.

Пользовательские процессы. Пользовательский процесс создается и поддерживается с целью выполнения программного кода прикладной программы или вспомогательной ORACLE-программы (например, SQL*DBA). Пользовательский процесс так же управляет обменом информации с процессами сервера, используя программный интерфейс, описанный дальше.

ORACLE-процессы, Эти процессы вызываются другими процессами, чтобы выполнить некоторые функции вместо вызывающего процесса. К ним относятся серверные и скрытые процессы.

Серверные процессы. ORACLE создает серверные процессы, чтобы выполнить задания со стороны взаимосвязанных пользовательских процессов. Серверный процесс управляет коммуникацией и работает вместе с ORACLE, чтобы выполнить задания ассоциированного пользовательского процесса. Например, если пользователь ищет какие-то данные, которые все еще не включены в буферах базы данных СГО, ассоциированный серверный процесс будет читать необходимые блоки данных из соответствующих файлов и записывать их в СГО. ORACLE может быть конфигурирован таким образом, чтобы имелась возможность изменять количество пользовательских процессов, которые обрабатываются одним серверным процессом. При одиночной серверной конфигурации один серверный процесс выполняет задания одного пользовательского процесса. Многосерверная конфигурация обеспечивает одновременную работу множества пользовательских процессов и небольшого количества

серверных процессов. Это позволяет свести к минимуму количество серверных процессов и в максимально эффективной степени использовать доступные системные ресурсы.

Скрытые процессы (СП). ORACLE создает группу СП для каждого отдельного сеанса (сеанс - это работа ORACLE в отрезке времени от его включения до его выключения). Они консолидируют функции, которые иначе выполнялись множеством ORACLE-программ, по одной для каждого пользовательского процесса. СП асинхронно выполняют операции ввода/вывода и управляют остальными процессами с целью увеличения степени параллелизма и отсюда производительности системы. Данная СГО и СП вместе образует один ORACLE-сеанс.

Любой ORACLE-сеанс может использовать несколько СП, которые имеют следующие имена: *DBWR, LGWR, CKPT, SMON, PMON, RACH, RECO, Dnnn, LSKn* и описаны ниже.

DBWR (Database Writer) - "писатель" в базе данных. Скрытый процесс DHWR записывает модифицированные данные из буферной памяти в файлы с данными. В зависимости от метода, по которому ORACLE сохраняет восстанавливаемые данные, *DBWR* может и не записывать блоки с данными по окончанию транзакции. Этот процесс оптимизирует записи и сводит их количество к минимуму. В общем-то *DBWR* записывает только тогда, когда в СГО необходимо прочитать новые данные и количество свободных буферов базы данных недостаточное.

liiWR (Log Writer)- "писатель" восстанавливаемых данных. Скрытый процесс *LGWR* записывает восстанавливаемые данные на диске. Эти данные генерируются в буферах для восстанавливаемых данных СГО. В ходе выпоннения транзакций, когда буферы восстанавливаемых данных заполнятся, этот процесс записывает восстанавливаемые данные в соответствующиие файлы для восстановления.

CKPT - точка сохранения. В определенные моменты, все модифицированные данные из буферов базы данных в СГО записываются в файлы с данными с помощью процесса *DBWR.* Эту запись называют точкой сохранения. Данный процесс отвечает за то, чтобы сигнализировать *DBWR* о наступления *CKPT* и о необходимости обновления всех файлов с данными и контрольных файлов в базах данных. *CKPT* необязательный процесс; если *CKPT* не включен, *LGWR* берет на себя обязательства СКРТ.

SMON(System monitor) - системный монитор. Этот процесс восстанавливает базу данных предыдущего сеанса во время запуска нового сеанса. В системе с множеством сеансов (в которой используется параллельный сервер), *SMON* во время данного сеанса

может так же восстанавливать другие сеансы, которые были прерваны. *SMON* так же объединяет свободные участки в файлах базы данных, чтобы сделать свободное пространство компактным и легко доступным.

PMON(Process monitor) - процессный монитор. Восстанавливает процессы, при их прерывании. Он служит также для стирания буферной памяти и освобождения ресурсов, которые были заняты процессом. Проверяет так же диспетчер и серверные процессы и перезапускает их, если они не прерваны.

ARCH - архиватор. Он копирует активные файлы из восстанавливающихся данных в архив, когда те заполняются. Архиватор активен только тогда, когда буферы восстанавливающихся данных базы данных используются в режиме *ARCHIVEWG*.

RECO (Reconstructor) - восстановитель. Используется для разрешения распределенных транзакций, которые были прерваны из-за сбоя питания или системы в распределенной базе данных. Через определенные интервалы времени этот процесс пытается связаться с отдаленной базой данных и автоматически заканчивает передачу локальной части прерванных локальных транзакций (если такие имеются).

Dnnn - диспетчер. Диспетчеры присутствуют тогда, когда используется многосерверная конфигурация. Они не являются обязательно скрытыми процессами. Для каждого коммуникационного протокола, который используется в данный момент, создается хотя бы один процесс диспетчеров. Каждый такой процесс (D001, D002, ... , Dnnn) отвечает за передачу запросов из связанных пользовательских процессов к общедоступным серверным процессам и для возвращения ответов обратно к соответствующим пользовательским процессам.

LCKn - замок, (*n* - номер замка). Могут быть использованы от одного до десяти закрывающих процессов (LCK0, ... , LCK9), для внутреннего закрывания во время использования параллельного сервера ORACLE.

5. Запуск и выключение ORACLE

В этом разделе дана концепция запуска и выключения ORACLE-сеанса и ORACLE-базы данных. Здесь рассматриваются:

- ⇒ концепции запуска и выключения;
- ⇒ виды и этапы загрузки базы данных;
- ⇒ методы и причины выключения базы данных;
- ⇒ параметрические файлы.

5.1. Введение в запуск и выключение ORACLE

Данная ORACLE-база данных не всегда доступна всем пользователям. Администратор базы данных может загрузить базу данных таким образом, чтобы она была открытой. Тогда все пользователи имеют доступ к находящейся в ней информации. Если база данных открыта, ее администратор может выключить ее так, чтобы в дальнейшем она была закрытой. Когда база данных закрыта, пользователи не имеют доступ к информации, которую она содержит. Только администратор базы данных может открывать и закрывать ее. Рядовой пользователь не имеет контроля над текущим состоянием базы данных. Защита запуска и выключения базы данных реализуется ограничением количества пользователей, имеющих права на запуск и выключение. Каждый член данной группы является внутренним.

5.2. Подключение внутреннего пользователя

Запуск и закрывание базы данных являются мощными административными функциями, которые исключают возможность подключения произвольного пользователя как внутреннего. Для определения данного пользователя как внутреннего должны соблюдаться следующие условия:

$\Rightarrow$ номер пользователя в операционной системе должен быть определенным, позволяющим такое подключение;

$\Rightarrow$ сам пользователь уполномочен подключаться как внутренний;

$\Rightarrow$ база данных имеет пароль для подключения внутренних пользователей и пользователь знает его.

В дополнение к этому, пользователи могут связываться с базой данных, в качестве внутренних, только для отдельно взятого сервера, но не для всех распределенных серверов. Все это обеспечивает дополнительную защиту и препятствует неуполномоченным пользователям запускать и останавливать базу данных.

5.3. Запуск базы данных.

База данных становится доступной для пользователей после ее запуска, который имеет три этапа: запуск сеанса, инсталяция базы данных и открытие базы данных.

Запуск сеанса. Процесс запуска сеанса включает определение СГО, общей области памяти, которая будет использоваться информацией базы данных, и создание скрытых процессов. Запуск сеанса осуществляется в первую очередь, до инсталации базы данных, сами. Прежде чем сеанс будет действительно создан, ORACLE читает параметрический файл, который определяет инициализацию сеанса.

Этот файл содержит параметры, которые определяют: размер СГО; имя базы данных, с которой сеанс может быть связан и т.д.

Ограниченный вариант запуска сеанса. Сеанс может быть запущен и в ограниченном виде. Это значит, что после запуска базы данных, связь будет предоставляться только тем пользователям, которым присвоена привилегия *RESTRICTED SESSION.*

Запуск сеанса в ненормальной ситуации. В обычных ситуациях, когда база данных выключена, предыдущий сеанс может и не быть "чисто" выключен (например, один из процессов сеанса может и не быть остановлен). В такой ситуации, база данных может сообщить об ошибке при новом запуске сеанса. Чтобы разрешить эту проблему, администратор базы данных должен остановить все ORACLE процессы, оставшиеся с предыдущего сеанса и запустить новый сеанс.

Инстапация базы данных. Инсталировать базу данных значит связать базу данных с запущенным ранее сеансом. После того как это сделано, база данных остается закрытой и доступной только со стороны администратора базы данных. Администратор имеет возможность запустить сеанс и инсталировать базу данных только для специфических поддерживающих операций. После инсталяции базы данных, сеанс находит и открывает управляющие файлы (они определяются параметром *CONTROL FILES* параметрического файла, использованного при запуске сеанса). После нахождения управляющих файлов базы данных, ORACLE читает их, чтобы получить имена файлов с данными и файлов с восстанавливаемыми данными, и тогда подтверждает, что они существуют в таком виде, в котором они определены. Если ORACLE работает в режиме с параллельным сервером, то имеется возможность инсталировать множество конкурирующих сеансов. В этом случае администратор может выбрать вид инсталяции базы данных - *закрытый* или *параллельный.*

В первом случае активизируется только один сеанс. Если первый сеанс, который инсталирует базу данных является сеансом закрытого вида, то этот и только этот сеанс может инсталировать базу данных. Версии ORACLE, которые не поддерживают опцию параллельного сервера, позволяют инсталировать базу данных только в закрытом виде.

Во втором случае, если первый сеанс, который инсталирует базу данных, запущен в параллельном (распределенном) виде, другие сеансы, которые запущены в параллельном виде, могут тоже инсталировать базу данных. Количество сеансов, которые могут инсталировать базу данных, предварительно ограничено определенным числом.

Открывание базы данных. Открывание инсталированной базы данных - это процесс, в результате выполнения которого она делается доступной для нормальных операции. Каждый действительный пользователь может связаться с базой данных и иметь доступ к ее информации, если она уже открыта. Обычно, в большинстве случаев, администратор базы данных открывает ее, делая ее доступной для общего пользования. Вместе с открытием базы данных, открываются также файлы с данными и файлы с информацией для восстановления.

Восстановление сеанса. Если база данных была закрыта, потому что администратор прервал ее сеанс или имелся сбой по питанию во время работы базы данных, то восстановление сеанса выполняется автоматически при повторном открытии базы.

Получение rollback-сегментов. После того как сеанс откроет базу данных, он пытается получить один или несколько rollback-сегментов.

Разрешение сомнительных транзакций распределенного типа. Допустим, что база данных закрыта внезапно (например, сбой питания, или прерван сеанс) и одна или больше распределенных транзакций не закончены или отменены. Когда база данных откроется снова и восстановление сеанса закончится, скрытый процесс *RECO* автоматически, немедленно и последовательно разрешает любую распределенную транзакцию таким образом, чтобы она была либо закончена полностью, либо отменена.

5.4. Закрытие базы данных и сеанса

Закрытие сеанса и базы данных, с которой он связан, происходит в три этапа: закрытие базы данных, деинсталяция базы данных, закрытие сеанса.

Закрытие базы данных. Когда база данных закрывается, все данные базы и восстанавливающие данные из СГО записываются в файлах с данными или в файлах с восстанавливаемыми данными соответственно. После этой операции, все открытые файлы с данными и файлы с восстанавливаемыми данными закрываются. После того как база данных закроется, но все еще будет оставаться инсталированной, управляющие файлы остаются открытыми.

Закрытие базы данных прерыванием сеанса. В ряде непредусмотренных ситуаций, сеанс открытой базы данных может быть прерван, с тем, чтобы немедленно и полностью закрыть базу данных. Этот процесс быстрый, потому что операции записи всех данных из буферов СГО пропускаются. Следующее открытие базы данных автоматически восстанавливает сеанс. Если в системе произойдет сбой, пока база данных находится в открытом состоянии,

то сеанс прерывается и может быть восстановлен только после следующего открывания базы данных.

Деинсталяция базы данных. Это является вторым шагом закрытия базы данных и заключается в прерывании связи базы данных с сеансом. После деинсталяции базы данных управляющие файлы базы закрываются, и в памяти компьютера остается только сеанс.

Закрытие сеанса. Последний шаг закрытия базы данных - это закрытие сеанса. Когда сеанс закроется, СГО деинсталируется из памяти системы и скрытые процессы прерываются компьютером.

Ненормальное прерывание сеанса. При обнаружении остатков предыдущего сеанса (не все структуры в памяти деинсталированы, какой-то из скрытых процессов не закончен) запуск следующего сеанса невозможен. Чтобы устранить эту проблему, администратор базы данных уничтожает в первую очередь следы предыдущего сеанса, после чего может выполнить новый сеанс, запуская его, или используя команду *SHUT DOWN ABORT.*

5.5. Параметрические файлы

Прежде чем запустить сеанс, ORACLE должен прочитать параметрический файл. Этот файл является текстовым файлом, который содержит список конфигурационных параметров сеанса. С помощью этих параметров задаются определенные значения, которые используются для инициализации памяти и процессов сеанса. Этот файл используется только вспомогательной программой SQL*DBA. Параметры этого файла должны содержать следующую информацию:

⇒ имя базы данных, для которой запускается сеанс;

⇒ объем памяти, который следует использовать для структур СГО;

⇒ что делать с заполненными файлами с восстанавливающими данными;

⇒ имена и расположение управляющих файлов базы данных;

⇒ имена частных rollback-сегментов в базе данных.

Пример характерного параметрического файла:

```
db_blckk_buffers = 550
db_name = ORA 7 PROD
db_domain = US.ACME.COM
#
licence_max_users = 64
#
control_files = filename 1, filename2
#
log_archiv_dest = c: \logarch
```

log_archiv_format = arch%S.ora
log_arch_start = TRUE
log_buffer = 64512 #
rolback_segments = rs_one, rs_two

Большинство параметров относятся к одной из следующих групп:

⇒ параметры, задающие имена (например, файлов);

⇒ параметры, устанавливающие пределы (например, максимумы);

⇒ параметры, влияющие на емкость и количество отдельных частей памяти, которые называются переменными параметрами (например, параметр *DB_BLOCK_BUFFERS,* который определяет количество блоков данных, выделенных в компьютерной памяти СГО).

Изменение значения параметров. Администратор базы данных может задавать различные значения переменных параметров с тем, чтобы улучшить производительность системы управления базами данных. Какие точно параметры больше всего влияют на систему зависит от различных характеристик и переменных базы данных.

6. Защита базы данных

Системы баз данных с множеством пользователей, к которым относится и ORACLE, включают меры обеспечения надежности и защиты, которые контролируют и управляют доступом к базе данных и ее использование.Защита имеет следующие основные задачи:

⇒ предотвращает несанкционированный доступ к базе данных;

⇒ предотвращает несанкционированный доступ к схемным объектам;

⇒ контролирует использование диска;

⇒ контролирует использование системных ресурсов (например, время процессора, областей памяти и др.);

⇒ наблюдает действия пользователей.

Защита базы данных может быть разделена на две отдельные части - *защита системы* и *защита данных.*

Защита системы включает механизмы, контролирующие доступ к базе манных и ее использование на системном уровне. Она контролирует:

⇒ действительны ли комбинации имен/паролей пользователей;

⇒ имеет ли данный пользователь право подключаться к распределенной базе данных;

⇒ количество дискового пространства, которое предоставляется на расположение различным объектам пользователя;

⇒ ресурсные ограничения пользователя;

⇒ включен или нет контроль базы данных;

$\Rightarrow$ какие операции в системе может выполнять пользователь.

Защита данных включает механизмы, которые контролируют доступ к данным и их использование на программном уровне. Эта защита, в частности, определяет:

$\Rightarrow$ какие пользователи имеют доступ к определенным схемным объектам и определенным операциям (например, пользователь *SCOT* может выполнять команды *SELECT* и *INSERT,* но не и *DELETE,* когда использует таблицу *EMP);*

$\Rightarrow$ действия, если есть такие, которые контролируются каждым схемным объектом и обеспечивают целостность данных.

Механизмы защиты. Сервер ORACLE предлагает "дискретный контроль доступа", который реализует метод ограничения доступа к определенной информации на основе введения привилегий. Для того, чтобы данный пользователь имел возможность использовать определенные команды, он, для этого, должен иметь соответствующую привилегию. Привилегированные пользователи могут давать привилегию другим пользователям на свою ответственность, из-за чего этот вид защиты называется "дискретной".

ORACLE управляет защитой базы данных, используя несколько различных методов: пользователи базы данных и схем; привилегии; роли; ресурсные ограничения; контроль.

6.1. Пользователи базы данных и схем

Каждая база данных ORACLE имеет список имен пользователей. Чтобы получить доступ к базе данных, пользователь должен использовать заявку на обслуживание базы данных и попытаться включиться с действительным пользовательским именем. Каждое пользовательское имя имеет связанный с ним пароль, с помощью которого предотвращается неправомерное использование имени. С каждым пользователем базы данных ассоциирована схема с тем же именем. Схема представляет логический набор объектов (таблиц, видов, последовательностей, синонимов, индексов, кластеров, процедур, функций, пакетов и связей с базами данных). По умолчанию каждый пользователь создает и имеет доступ ко всем объектам в соответствующей схеме.

Защищенная область. Каждый пользователь имеет свою область с защитой, для которой определяют: действия (привилегии и роли) доступные пользователю; квоты табличного пространства (доступное место на диске) для пользователя; ресурсные ограничения системы (например, время работы процессора) для пользователя.

6.2. Привилегии

Привилегия определяется как право выполнять специфическую SQL-команду. Некоторые примеры привилегий даны ниже:

⇒ право подключения к базе данных (создания сеанса);

⇒ право создания таблицы в своей схеме;

⇒ право выбора строк из чужих таблиц;

⇒ право выполнения сохраняемых чужих процедур.

Привилегии ORACLE - базы данных могут быть разделены на две категории: *системные* и *объектные.*

Системные привилегии. Они разрешают пользователям выполнять определенное действие в рамках всей системы или выполнять это действие с определенным объектом. Например, привилегию создавать табличное пространство или стирать строки всех таблиц в базе данных. Большинство системных привилегий принадлежат только администраторам и разработчикам прикладных программ, потому что они являются очень мощными.

Объектные привилегии. Данный тип привилегий устанавливается при работе пользователей с объектами. Например, привилегия стирать строки определенной таблицы. Объектные привилегии предоставляются конечным пользователям, чтобы они могли использовать прикладные программы базы данных для выполнения специфических задач.

Привилегии предоставляются пользователям для того, чтобы они получали доступ к данным в базе данных и могли модифицировать их по соответствующим правилам работы. Привилегии могут назначаться двумя способами:

⇒ через предоставление определенному пользователю. Например, привилегия записывать новые строки в таблицу *EMP* может быть дана пользователю *SCOT.*

⇒ привилегии сначала предоставляются так названной *роли* (именованная группа привилегий), после чего эта роль может быть предоставлена одному или больше пользователям. Например, привилегия записывать новые строки и таблице *EMP* может быть дана роли *CLERK,* которая со своей стороной предоставляется пользователям *SCOT* и *BRIAN.*

Так как роли позволяют легче и лучше управлять привилегиями, то они находят более широкое практическое применение.

6.3. Роли

Роль определяется как именованная группа привилегий, которая предоставляется другим ролям, или пользователям выполняющим

сходные действия. Если при разработке приложений, в среде ORACLE, используются роли для определения привилегий, то создаются условия для более легкого контроля и управления ими, благодаря следующим свойствам:

⇒ более быстрое предоставление привилегии. Вместо сложного и занимающего много времени назначения одного и того же набора привилегий многим пользователям, администратор базы данных может установить этот набор свойств для определенной роли и затем предоставить ее всей группе близких, по естеству работы, пользователей (группе в целом и каждому члену группы в отдельности);

⇒ динамическое управление привилегиями. Когда приходится изменять состав и вид привилегий данной группы пользователей, то это производится только коррекцией привилегий роли. Сделанные изменения в роли автоматически переносятся на защищенную область всех пользователей, которым предоставлена эта роль;

⇒ выборочный доступ к привилегиям. Роли, предоставленные данному пользователю, могут быть селективно активированными (доступными для пользования) или дезактивированными (недоступными для пользования). Это позволяет осуществлять специфический контроль и управление привилегиями, предоставляемыми пользователям в любой ситуации;

⇒ гибкость проектирования при использовании роли. Данная прикладная программа может быть спроектирована таким образом, чтобы иметь возможность автоматически активировать или дезактивировать выборочным способом роли во время ее выполнения. Кроме того, она может предоставлять свои роли другим ролям или пользователям. Прикладная программа может иметь несколько различных ролей, каждая из которых имеет свой набор привилегий, которые обеспечивают различные режимы доступа к данным во время выполнения программы.

При проектировании роли могут быть использованы и пароли для того, чтобы предотвратить несанкционированное пользование привилегиями, которые предоставляются роли.

7. Сохранение и восстановление базы данных

Здесь рассматриваются структурные и программные механизмы, которые используются ORACLE для того, чтобы обеспечить: восстановление базы данных после возникновения различных видов сбоев; гибкость операций восстановления и их эффективность в любой ситуации; доступность информации во время операций записи и восстановления.

7.1. Важность восстановления

В любой системе управления базой данных всегда существует вероятность возникновения системного или аппаратурного сбоя. Такое событие, как правило, сказывается на базе данных, и она должна быть восстановлена. Задачей восстановления после сбоя является обеспечение возможности записи сделанных всеми выполненными транзакциями перемен в восстановленную базу данных и возврат к нормальному режиму работы как можно быстрее, избавляя пользователей от проблем, причиненных сбоем. Имеются несколько причин, которые могут остановить функционирование базы данных. Наиболее часто встречающиеся типы сбоев описаны ниже.

Ошибки пользователя. Чтобы обеспечить восстановление системы после ошибок пользователей, ORACLE позволяет восстанавливать базу данных в точно заданных прошедших моментах времени. Например, если пользователь случайно попытался стереть таблицу, база данных может быть восстановлена до момента фактического стирания.

Сбой команды и процесса. Сбой команды наблюдается тогда, когда проявляется логическая ошибка во время выполнения ORACLE - команды (например, команда может представлять собой недействительную SQL -конструкцию). Когда появится программный сбой, действия команд (если такие есть) автоматически аннулируются ORACLE и управление возвращается пользователю. Срыв процесса может последовать в результате ошибки в пользовательском процессе, например, ненормальный выход из системы или неправильное выполнение процесса. Пропавший процесс определенного пользователя не может продолжить свою работу, но остальные пользовательские или системные процессы могут. Скрытый процесс *PMON* либо автоматически находит пропавший пользовательский процесс, либо будет информирован о нем от SQL*NET (модуль ORACLE для работы в условиях сети). *PMON* решает проблему путем отмены незаконченной транзакции пользовательского процесса и восстановления всех ресурсов, которые процесс использовал.

Часто встречающиеся проблемы, такие как ошибочные SQL - команды и прерывание пользовательских процессов, не должны останавливать систему управления базой данных в целом. Кроме того, ORACLE автоматически выполняет необходимое восстановление незаконченных транзакционных перемен и закрытие ресурсов с минимальным влиянием на работу всей системы или на других пользователей.

Сбой сеанса. Срыв сеанса наступает тогда, когда возникнет проблема в СГО или в коком-то скрытом процессе, которая

препятствует продолжению работы сеанса. Он может наступить в результате аппаратного сбоя, например, прерывание питания, или в результате программной неувязки, например, срыв операционной системы. Когда наступает сбой подобного характера, данные из буферов СГО не записываются в файлах с данными.

Сбой диска. Этот сбой может появиться при попытке записи или чтения из файла, находящегося на диске и необходимого для работы с базой данных. Дисковый сбой требует восстановления диска, что собственно означает восстановление файлов с данными в таком виде, чтобы информация в них соответствовала состоянию, максимально близкому моменту времени наступления дискового сбоя. Чтобы восстановить дисковой сбой, необходимо сохранить файлы с данными и все открытые и нуждающиеся в архивировании файлы с восстанавливающими данными. Если некоторый из файлов с данными повреждены из-за дискового сбоя, но большая часть базы данных осталась исправной, база данных может остаться открытой, восстанавливая в то же самое время нужное табличное пространство индивидуально. Исправные части базы данных остаются при этом доступными для пользования, а поврежденные в это время восстанавливаются.

7.2. Сохранение базы данных

Так как, в результате дискового сбоя, один или больше файлов может быть физически поврежден, то восстановление диска требует и восстановления поврежденных файлов, приводя их в состояние, которое соответствует моменту времени самого последнего сохранения базы данных системой управления. В системе применяется *полное* и *частичное* сохранение файлов базы данных.

Полное сохранение. Оно производится тогда, когда база данных закрывается и она становится недоступной для пользования. Осуществляется сохранение системой управления всех файлов с данными, открытых файлов с восстанавливаемыми данными и управляющих файлов.

Частичное сохранение. В этом случае производится сохранение системой управления части базы данных. Примерами частичного сохранения являются сохранение файлов с данными одного табличного пространства или сохранение управляющих файлов. Частичное сохранение имеет смысл только тогда, когда система управления базой данных работает с файлами с восстанавливаемыми данными в режиме *ARCH1VELOG,* при котором заполненные файлы с восстанавливаемыми данными архивируются прежде их повторного использования. Можно использовать различные виды частичных

сохранений с тем, чтобы удовлетворить любую стратегию сохранения. Например, файлы с данными и контрольные файлы могут быть сохранены при открытии или закрытии базы данных, или когда некоторое табличное пространство открывается или закрывается. Так как буфер с восстанавливаемыми данными работает в режиме *ARCHIVELOG*, дополнительная запись буфера не нужна.

8. Распределенная база данных и распределенные процессы

В системе управления базами данных ORACLE 7 предусмотрена возможность работы с распределенными процессами в рамках компьютерной сети. В этом разделе коротко рассмотрены основные архитектурные черты ORACLE, которые обеспечивают этот режим работы.

8.1. Архитектура клиент-сервер и распределенные процессы

Распределенные процессы используют больше чем один процессор, чтобы распределить работу по выполнении набора связанных между собой задач. Распределение процессов уменьшает нагрузку каждого отдельного процессора, позволяя различным процессорам "сконцентрироваться" на поднаборе связанных задач, улучшая, таким образом, возможности системы в целом.

На основе своей архитектуры "клиент - сервер", система управления базой данных ORACLE может легко реализовать концепцию распределенных процессов. В этой архитектуре система управления разделена на две части: сторона клиента и сторона сервера.

Сторона клиента, образно говоря, это передняя часть, или передняя сторона системы управления, которая взаимодействует с пользователем через дисплей, клавиатуру и указывающие устройства (мышь). Она не отвечает за доступ к данным. Основные ее функции связаны с запросами, обработкой и предоставлением данных, управляемых серверной частью. Конфигурация рабочей станции клиента, при такой организации, может быть оптимизирована в зависимости от характера выполняемой работы. Например, она может и не нуждаться в большом диске или, наоборот, должна иметь значительные графические возможности.

Сторона сервера выполняется софтвером ORACLE и поддерживает нужные функции конкурентного или общего доступа к данным. Она получает и обрабатывает SQL и PL/SQL-команды, которые поступают из прикладных программ клиента. Конфигурация компьютера,

управляющего серверной частью, может быть тоже оптимизирована с учетом характера выполняемых работ.

8.2. Распределенная база данных

Распределенная база данных представляет собой сеть баз данных, управляемую несколькими серверами, которые воспринимаются пользователем как одна логическая база данных. Основным преимуществом распределенной базы данных является то, что физически распределенные данные могут логически объединяться и возможно сделать их доступными для всех пользователей сети. Каждый компьютер, который управляет базой данных в распределенной базе данных, называется узлом. Базу данных, с которой пользователь связан непосредственно, называют локальной базой данных. Когда локальная база данных связывается с отдаленной базой данных, то эта локальная база данных является клиентом отдаленного сервера.

Разрешая более легкий способ доступа к большому количеству данных в сети, распределенная база данных должна в то же самое время иметь возможность скрывать местонахождение данных и механизмы доступа к базе данных. С концепцией распределенной базы связаны понятия *прозрачность местонахождения* и *автономность* данных.

Прозрачность местонахождения. Прозрачность местонахождения возникает, когда физическое местонахождение данных доступно для прикладных программ и пользователей распределенной базы данных. Одновременно несколько ORACLE-объектов (виды, процедуры, функции, синонимы и др.) могут иметь прозрачное местонахождение. Такую прозрачность, например, может предоставить вид, который связывает табличные данные из таблиц с одинаковым описанием, находящихся, однако, в различных базах данных. Пользователь вида, в этом случае, не должен знать местонахождение данных.

Автономность данных. Автономность значит, что любая база данных, которая включена в распределенную базу данных администрируется отдельно и независимо от остальных баз данных, как будто бы она не связана с сетью. Независимо от того, что каждая база данных может совместно работать с другими, все они являются отдельными системами, которые сохраняют свою индивидуальность.

В процессе работы в условиях распределенной базы данных архитектура ORACLE поддерживает все DML - операции, включая запросы, записи, обновление и стирание удаленных табличных данных. В случаях использования удаленных данных, производится обращение-запрос, которое включает только объектное имя удаленного

глобального объекта. Для доступа к удаленным данным никакого специального кодирования или сложного синтаксиса не нужно. Например, для запроса данных из таблицы с именем *EMP* в удаленной БД с именем *SALES*, обращение производится по глобальному табличному имени объекта, следующим образом:

*SELECT * FROM emp@sales;*

8.3. Двухфазовое окончание транзакции

ORACLE обеспечивает одинаковую надежность работы и целостности данных, как в распределенной, так и в классической, нераспределенной среде. Эта надежность реализуется на основе двухфазной модели транзакций и механизма их выполнения. Первая фаза заключается в заканчивании транзакции на уровне локальной сети, а вторая фаза - на уровне всей системы. Как и в нераспределенных системах, транзакции должны быть внимательно спланированы и должны содержать логические наборы SQL -команд. При наличии сбоя, независимо от его типа (системного или сетевого), распределенная транзакция или выполняется полностью, последовательно проходя через все включенные в транзакцию узлы, или отменяется во всех включенных узлах. Этим обеспечивается сохранение целостности данных во всей глобальной распределенной базе данных.

Двухфазный механизм окончания транзакции является полностью прозрачным для пользователей, выполняющих распределенные транзакции. Самая обычная команда, например, которая задает конец транзакции, автоматически вызывает механизм окончания транзакции. Не нужно никакого кодирования, или сложного синтаксиса, чтобы включить распределенные транзакции в прикладные программы базы данных.

Скрытый процесс *RECO* используется в режиме распределенной обработки информации. Он автоматически разрешает появившиеся сомнительные распределенные транзакции, в которых окончание прервано из-за системного или сетевого сбоя. После того как сбой устранен и нормальная работа и коммуникации восстановлены, процесс *RECO* для любого локального сервера автоматически заканчивает или отменяет всю сомнительную транзакцию распределенного типа, последовательно по всем, включенным в транзакцию, узлам.

В случае долговременного сбоя, ORACLE дает возможность администратору базы данных, для каждого локального узла, вручную закончить или отменить любую распределенную транзакцию, которую можно считать сомнительной (как следствие сбоя). Локальному

администратору предоставлена возможность вручную освобождать все закрытые и недоступные ресурсы, которые может быть стали такими в результате долговременного сбоя.

Если распределенная база данных должна быть восстановлена в состояние, соответствующее определенному моменту времени в прошлом, то вспомогательные средства ORACLE для окончания работы предоставляют возможность администраторам остальных локальных узлов тоже вернуть свои локальные базы данных к тому же состоянию. Этим обеспечивается целостность глобальных баз данных.

8.4. Копирование таблиц

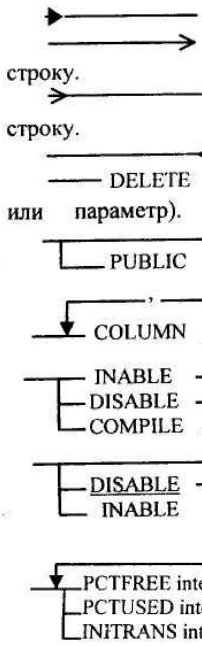
Система распределенной базы данных часто локально копирует удаленные таблицы, к которым наблюдается повышенная интенсивность обращения со стороны локальных пользователей. Имея копию (только для чтения) труднодоступных данных из некоторых узлов, распределенная база данных не должна каждый раз требовать информацию из разных узлов по всей сети. Этим оптимизируется процесс выполнения прикладных программ и повышается производительность системы. ORACLE предоставляет автоматический метод для копирования и обновления таблиц, создавая так называемые моментные таблицы (snapshots). Они представляют собой копии оригинальных таблиц, расположенных в отдаленных узлах. Эти копии разрешены только дая чтения. К моментным таблицам можно отправлять только запросы, но не изменять их. Изменять можно только оригинальные таблицы. Моментные таблицы, однако, обновляются периодически, чтобы переносить в них наступившие в оригинальных таблицах изменения.

8.5. ORACLE и SQL*NET

Если разработчики прикладных программ используют программный модуль SQL*NET, то им не нужно поддерживать сетевые коммуникации для своих программ базы данных. Если по какой-то причине приходится менять коммуникационный протокол передачи и приема данных, то это делается администратором базы данных. Прикладные программы, работающие в среде SQL*NET не требуют никаких модификаций и могут продолжать свое выполнение без посторонних эффектов.

ЧАСТЬ ВТОРАЯ. ЯЗЫК ДЛЯ РАБОТЫ С РЕЛЯЦИОННОЙ БАЗОЙ ДАННЫХ ORACLE 7

В этой части рассматриваются основные команды языка SQL. Правила их записи приводятся с помощью синтаксических диаграмм. Эти диаграммы имеют следующие элементы:



- начало диаграммы.
- продолжение диаграммы на следующую строку.
- продолжение диаграммы на предыдущую строку.
- конец диаграммы.
- обязательный элемент (ключевое слово или параметр).
- необязательный элемент.
- элемент, который может многократно повторяться.
- обязательное использование одного из элементов.
- выбор необязательного элемента. Подчеркнутый элемент подразумевается.
- выбор списка специфицированных элементов.

Для описания команд используются следующие обозначения их параметров:

table - имя объекта, типа определённого параметра. Например, таблица, вид, индекс и т.д.

c - один знак клавиатуры;

'text' - текстовая литера;

char - параметр должен быть выражением, переменной, или литерой типа CHAR или VARCHAR2;

condition - условие, которое имеет значение TRUE или FALSE. (Условия описаны в разделе 2.2);

date или *d* - величина типа даты;

expr -выражение числового типа. (Описание приводится в разделе 2.3);

integer - целое число;

label - величина типа MLSLABEL;

number, m или *n* - выражение типа NUMBER;

raw - величина типа RAW;

rowid - величина типа ROWID;

subquery - команда SELECT языка SQL, но с ограничениями.

:host_variable - имя переменой в процедурах и функциях. Для определения типов данных используют также *:host_integer* и *:host_string;*

statement_name - идентификатор SQL, или PL/SQL-команды;

bloc_name - блок.

1 Элементы SQL

1.1. Схемные объекты

Схемные объекты SQL - это таблицы, виды, индексы, последовательности, синонимы, функции, процедуры, пакеты, триггеры базы данных и кластеры о которых было рассказано в первой части. Совокупность таких объектов называется схемой. Схема принадлежит пользователю базы данных и ее имя совпадает с его именем. Каждый пользователь имеет определенное имя и пароль для доступа, которые задаются администратором базы данных. При распределенной базе данных в схеме могут содержаться связи баз д;шных.

Существуют некоторые объекты, которые сохраняются в базе данных, создаются и обрабатываются командами SQL, но они не являются частью схемы. Это профайлы, роли, сегменты для возвращения предыдущего состояния (rollback segments) и табличные пространства. В командах SQL используются и некоторые части объектов, такие как столбцы таблицы и виды, ограничения целостности таблиц, пакетные процедуры, пакетные функции и другие объекты сохраняемые в процедурах.

Имена объектов создаются в соответствии со следующими правилами:

1. Они должны иметь длину до 30 символов за исключением имен базы данных, которые содержат до 8 символов и имен связей распределенной базы данных, которые имеют длину до 128 символов.
2. В наименование нельзя включать комментарии.
3. В состав имен могут включаться как прописные, так и строчные буквы. Все они воспринимаются одинаковым образом.
4. Имена должны начинаться с буквы.
5. Имена могут содержать буквы, цифры и знаки $, _ и #, однако не рекомендуется использовать знаки $ и #. Имена связи распределенной базы данных могут содержать так же знаки . (точка) и @.
6. Имена не должны совпадать с зарезервированными словами, такими как *ACCESS, ADD, ALL, ALTER, AND* и т.д.

7. Слово *DUAL* нельзя использовать как имя объекта или части объекта, так как это имя фиктивной таблицы, которая может быть использована программами *SQL*PLUS* и *SQL*FORMDS.*

8. Имена не должны совпадать с ключевыми словами, такими как *ADMIN, ALTER, ALLOCATE* и т. д.

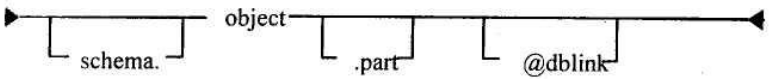
9. Имя должно быть уникальным в своей области имен.

ORACLE поддерживает несколько областей имен. В соответствующей области включаются имена: таблиц, видов, последовательностей, частных синонимов, процедур и функций; индексов; ограничений целостности; кластеров; триггеров базы данных; частных связей распределенной базы данных; пользователей и ролей; общих синонимов; общих связей распределенной БД; табличных пространств; *rollback*-сегментов; профайлов.

Столбцы в одной таблице или виде не могут иметь одинаковые имена, однако в различных таблицах это допускается. Процедуры и функции, содержащиеся в одном пакете могут иметь одинаковые имена, когда их аргументы различаются по количеству и типу. Не рекомендуется, однако, использовать большое количество процедур или функций с одинаковыми именами и различными аргументами в одном и том же пакете.

10. Имя может быть записано в кавычках. Такие имена могут содержать любые знаки, игнорируя правила 3, 4, 5, 6 и 7. Эти имена могут включать и пробелы. В данном случае прописные и строчные буквы имеют значение. Например, имена *"Depart"* и *"DEPART"* различные. Имена, записанные в кавычках, должны использоваться всегда в кавычках.

При задании объектов или части объектов можно использовать следующую синтаксическую диаграмму:



Здесь: *schema* - имя схемы; *object* - имя объекта; *part* - имя части объекта, например, столбец таблицы; *@dblink* - связь в распределенной базе данных, которая задает базу данных отличную от локальной.

1.2. Значение *null,* псевдостолбцы, коментарии

Когда в некоторой строке нет значения для данного столбца, то это отсуствующее значение называется *null.* Говорят, что столбец имеет мшчение *null,* содержит *null.* Любой столбец, независимо от типа данных, может принимать значение *null.* Недопустимо присваивать

значение *null* тем столбцам, для которых определены ограничения целостности *NOT NULL* или *PRIMARY KEY*.

Любая скалярная функция возвращает велечину *null,* когда задан аргумент *null.* Групповые функции, такие как функция вычисления среднеарифметического, среднегеометрического, максимального значения и др., игнорируют значения *null.* Строки таблицы, которые содержат, или не содержат значение *null* в данном столбце, можно определить оператором сравнения *IS NULL* или *IS NOT NULL* соответственно. Когда значение *null* используется с другими операторами сравнения, результат сравнения - *FALSE*. Например, команда *SELECT* вывода имен студентов, имеющих семестр равным *null* не выведет строки, если задано условие *WHERE SEM=NULL,* независимо от того, что могут быть студенты, для которых пропущено ввод семестра. Чтобы вывести имена студентов, для которых пропущено ввод семестра, условие должно быть *SEM IS NULL*.

Любая таблица имеет псевдостолбцы *LEVEL, ROWID* и *ROWNUM* и любая последовательность дает два значения *NEXTVAL* и *CURRVAL*. Одна последовательность генерирует уникальные значения, которые можно использовать при записи значения первичного ключа или в столбце с ограничением *UNIQUE*. Последнее генерированное значение предоставляется *CURRVAL,* а следующее значение генерируется *NEXTVAL*. Использовать эти значения в SQL-командах можно, посредством конструкции *scquence.CURRVAL* и *sequence.NEXTVAL,* где *sequence* - имя последовательности, задаваемое командой *CREATE SEQUENCE.* Значения *NEXTVAL* и *CURRVAL* можно использовать в списке выводимых данных командой *SELECT.*

Псевдостолбец *ROWID* описан в разделе 2.1. первой части.

Псевдостолбец *ROWNUM* дает порядковый номер строки в совокупности строк, которые возвращаются в результате запроса команды *SELECT.* Дня первой строки *ROWNUM* равен 1, для второй 2 и т.д. Этот псевдостолбец может быть использован для нумерации выбранных строк или для ограничения количества строк, которые выводятся, как показано в следующем примере:

*SELECT * FROM chairs WHERE ROWNUM<10;*

Псевдостолбец *LEVEL* дает уровень строки запроса задавая иерархию данных. Иерархию данных можно определить предложениями *START WITH* и *CONNECT BY* команды *SELECT*. Первая строка имеет уровень 1 и называется родителем некоторых из следующих строк, которые называются детьми. Дети строк уровня 1 имеет уровень 2, дети строк уровня 2 имеют уровень 3 и т.д.

В командах SQL и PL/SQL можно записывать комментарии. Они особенно полезны в сохраненных в базе данных функциях, процедурах и пакетах. Комментарий может начинаться либо парой символов /* и заканчиваться */, либо начинаться символами — и продолжаться до конца строки. При первом способе задания, комментарий может занимать несколько строк, а при втором он должен занимать только одну строку. Комментарий может находиться между частями одной команды, независимо от того, каким образом он задан. Ниже приводится пример, в котором показаны способы включения комментариев в программу:

SELECT deptno, names, subject
/ Выбор студентов, которые имеют двойки. Выводятся их факультетские номера, имена и учебные дисциплины */*
FROM grades, students / Таблица grades содержит оценки. Таблица students используется для имен студентов */*
WHERE grade=2 and; — Условие получения двойки.
grades.deptno—students, deptno; — Связывает две таблицы.

2. Операторы, функции, выражении и условия

2.1. Операторы

Операторы используются для выполнения некоторых операций с данными таких как вычитание, деление и т.д. Данные, над которыми производятся операции называются операндами. Операторы бывают унарные и бинарные. Унарные операторы имеют один операнд, а бинарные - два. Унарные операторы - это *NOT, PRIOR,* бинарные - *OR, УМНОЖЕНИЕ* и др. Операторы + и - могут быть как унарными, так и бинарными. Множество операторов можно разделить на арифметические, знаковые, логические, операторы для сравнения и установки иерархии.

2.1.1. Арифметические операторы. Унарные арифметические операторы + и - записываются перед арифметическим выражением и имеют самый высокий приоритет выполнения. Оператор " - " меняет знак выражения, а "+" не меняет его. Бинарные операторы УМНОЖЕНИЕ (*) и ДЕЛЕНИЕ */)* производят умножение и деление двух арифметических выражений и занимают второе место по приоритету выполнения. Результат выполнения этих операторов - выражение арифметического типа. Бинарные операторы "+" и "-" выполняют суммирование и вычитание двух арифметических выражений и имеют самый низкий приоритет выполнения.

Результат операций - тоже выражение арифметического типа.

2.1.2. Знаковые операторы. Есть один знаковый оператор, который обозначается | | и объединяет две или больше знаковых строк (оператор конкатенации). Символ знакового оператора вставляется между ними. Например, выражение *'Сегодня ' | | '11.04.1997' | | 'года'* дает результат

Сегодня 11.04.1997 года

2.1.3. Операторы сравнения. Эти операторы, при заданном условии, используются для сравнения двух значений или одного значения со многими другими значениями. Сравниваемые величины могут быть арифметическими выражениями или знаковыми строками. Результатом действия операторов сравнения является либо ложь, которая записывается FALSE, либо истина (TRUE). Ниже перечисляются операторы сравнения:

= - определяет равенство двух значений переменных;

!=, ^=, <> -определяет различия (неравенства) двух значений;

> - определяет, является ли значение первого операнда большим, чем значение второго операнда;

< - определяет, является ли значение первого операнда меньшим, чем значение второго операнда;

>= - определяет, является ли первое значение большим или равным второму;

<= - определяет, является ли первое значение меньшим или равным второму;

IS NULL - проверяет равна ли данная величина значению *null.* Например, с помощью приведенной ниже команды, выводятся все студенты *(students),* для которых пропущен ввод в базу данных наименования специальности *(spec):*

*SELECT * FROM students WHERE spec IS NULL;*

NOT NULL - проверяет неравенство данной величины значению *null.*

IN - проверяет находиться ли данное значение в списке значений.

Например, следующая ниже команда выводит имена преподавателей *(readers),* занимающих должности *(scdegree)* профессора и доцента:

*SELECT*FROM readers WHERE scdegree IN ('доцент', 'профессор');*

NOT IN - проверяет отсутствие данного значения в списке значений. Например, следующая команда выводит факультетские номера *(deptno)* и имена *(names)* студентов, у которых нет балла *(grade)* плохо (2):

SELECT deptno, names FROM students WHERE deptno NOT IN (SELECT DISTINCT deptno FROM grades WHERE grade=2);

ANY, SOME - сравнивает находиться ли данное значение в каком-либо отношении с некоторым знаяением из списка значений. Используется вместе с одним из операторов =, !=, <, >, <= или >=. Оператор *IN* эквивалентен = *ANY.* Например, следующая команда выводит факультетские номера и имена студентов, которые имеют двойки:

SELECT deptno, names FROM students WHERE deptno= ANY (SELECT DISTINCT deptno FROM grades WHERE grade=2);

ALL - сравнивает находится ли данное значение в каком-то отношении со всеми значениями списка значений. Используется вместе с одним из операторов =, !=, <, >, <= или >=. Оператор *NOT IN* эквивалентен != *ALL.* Например, следующая команда выводит имена преподавателей, которые не являются как членами *(chairno)* кафедры информатики, так и кафедры математики:

*SELECT * FROM readers WHERE chairno!= ALL (10, 20);*

EXISTS - дает значение *TRUE,* если запрос возвращает хоть одну строку. Например, следующая команда выводит имена преподавателей, которые ставили балл 2.

SELECT FROM readers WHERE EXISTS*

*(SELECT * FROM grades WHERE grades, chairno=readers. chairno AND grades.readerno=grades.readerno);*

x [NOT] LIKE y [ESCAPE z] - сравнивает значения типа *CHAR* или *VARCHAR2.* Квадратные скобки в определении обозначают, что ключевое слово *NOT* и предложение *ESCAPE* не обязательны и их можно пропустить. При сравнении учитываются прописные и строчные буквы. В задаваемых значениях могут содержаться знаки % и _, которые имеют специальное предназначение. Знак % задает строку *x* произвольной длины, в том числе и нулевую строку (строка нулевой длины). Знак _ задает произвольный символ в *x.* Знаки % и _ могут быть включены в строку *y* как любой другой символ, используя предложение *ESCAPE.* Чтобы это произошло, перед знаком % или _ необходимо записать знак *z* , указанный в предложении *ESCAPE.*

Пример 1. Для вывода имен, которые начинаются на *Ни,* необходимо выполнить команду:

SELECT names FROM students WHERE names LIKE 'Ни%';

Пример 2. Для вывода имен, вторая буква у которых *и,* необходимо выполнить команду:

SELECT names FROM students WHERE names LIKE '_и%';

Пример 3. Для вывода имен кафедр, в которых содержится строка *'а_б*', необходимо выполнить команду:

SELECT name FROM chairs WHERE chairs LIKE '%а_б' ESCAPE' \';

В этом случае знак _ воспринимается как знаки *а* и *б*, и не воспринимается как специальный знак.

2.1.4. Логические операторы. Логические операторы включают унарный оператор *NOT* и бинарные *AND* и *OR.* Логические операторы, в качестве операндов (аргументов), используют выражения с операторами для сравнения. Результаты выполнения логических операторов, в зависимости от значений аргументов, приведены в следующих трех таблицах.

NOT	TRUE	FALSE	NULL
	FALSE	**TRUE**	**NULL**

AND	TRUE	FALSE	NULL
TRUE	**TRUE**	**FALSE**	**NULL**
FALSE	**FALSE**	**FALSE**	**FALSE**
NULL	**NULL**	**FALSE**	**NULL**

OR	TRUE	FALSE	NULL
TRUE	**TRUE**	**TRUE**	**TRUE**
FALSE	**TRUE**	**FALSE**	**FALSE**
NULL	**TRUE**	**FALSE**	**NULL**

Каждый аргумент может иметь три значения: true, false и null, Значения функции в зависимости от комбинации аргументов приведены в таблицах жирным шрифтом.

2.1.5. Операторы установки. Операторы установки используются для объединения строк двух запросов. Есть четыре оператора установки, действие которых следующее:

UNION - объединяет все *различные* строки двух запросов. Результатом является объединение множеств *различных* строк в двух запросах;

UNION ALL - объединяет все строки двух запросов, в том числе и одинаковые. Результатом является объединение множеств всех строк в двух запросах;

INTERSECT - объединяет общее в различных строках двух запросов. Результат - пересечение множеств различных строк в двух запросах;

MINUS - объединяет все различные строки первого запроса, которые отсутствуют во втором запросе. Результатом является вычитание строк двух запросов.

Пример. Пусть имеются два запроса о получении информации о студентах с двойками. Первый запрос по предметам *(subject)* информатике и математике, а второй - по физике и химии. Тогда:

SELECT deptno, subject FROM grades WHERE grade=2 AND subject IN ('матаматика', 'информатика');

deptno	subject
12023	информатика
21164	математика
21164	информатика
18975	математика

SELECT deptno, subject FROM grades WHERE grade=2 AND subject IN ('физика','химия');

deptno	subject
32896	физика
21164	химия
31753	физика
31753	химия

Следующие примеры показывают действие операторов установки:

Пример 1. *UNION*

SELECT deptno FROM grades WHERE grade=2 AND subject IN ('математика', 'информатика')
UNION
SELECT deptno FROM grades WHERE graded AND subject IN ('физика', 'химия');

deptno
12023
21164
18975
32896
31753

Пример 2. *UNION ALL*

SELECT deptno FROM grades WHERE grade=2 AND subject IN ('математика', 'информатика')
UNION ALL
SELECT deptno FROM grades WHERE graded AND subject IN ('физика', 'химия');

deptno
12023
21164
21164
18975
32896
21164
31753

Пример 3. *INTERSECT*

SELECT deptno FROM grades WHERE grade=2 AND subject IN ('математика','информатика')
INTERSECT
SELECT deptno FROM grades WHERE grade=2 AND subject IN ('физика','химия');

deptno
21164

Пример 4. *MINUS*

SELECT deptno FROM grades WHERE grade=2 AND subject IN ('математика', 'информатика')
MINUS
SELECT deptno FROM grades WHERE grade=2 AND subject IN ('физика', 'химия');

deptno
12023
18975

2.2. Функции

Функции, как и операторы, возвращают один результат (одно значение) при заданных аргументах, но отличаются от операторов по форме их использования. Функции могут иметь один, два или больше аргументов, или вообще не иметь аргументов и задаются следующим форматом:

function(argument1, argument2,...)

Функции могут использоваться и с аргументами, тип которых отличается от ожидаемого типа функции. Тогда ORACLE преобразует значение в требуемый функцией тип. Описанные здесь функции используются только в SQL-командах и не используются в командах процедурного языка PL/SQL при записи PL/SQL-процедур и функций.

Функции в SQL разделяются на два основные типа: функции для одной строки (или константы) и групповые функции для множества строк (или ряда констант). Функции одной строки возвращают на каждую обработанную строку по одному значению, тогда как

групповые функции возвращают для всех обработанных строк только одно значение. Функции одной строки могут использоваться в списке выводимых значений команды *SELECT,* которые не содержат предложения *GROUP BY* и в предложениях *WHERE, START WITH* и *CONNECT BY.* Групповые функции могут использоваться в списке выводимых значений команды *SELECT,* которая содержит предложение *GROUP BY* и в предложении *HAVING.* В этом случае ORACLE разделяет строки таблицы по группам, причём строки одной группы имеют одинаковые значения в столбцах, включенных в предложение *GROUP BY.*

Функции, в зависимости от типа результата и их аргументов, могут быть объединены в следующие группы: арифметические, знаковые, функции даты, преобразования типов и другие функции.

2.2.1. Арифметические функции.

♦ *ABS(n)* - возвращает абсолютное значение *n.*

Пример: *SELECT ABS(-12) "Absolute" FROM DUAL;*

Absolute
12

♦ *CEIL(n)* - возвращает самое маленькое целое число, которое больше или равно *n.*

Пример: *SELECT CEIL(112.3) "Ceiling" FROM DUAL;*

Ceiling
113

♦ *COS(n)* - возвращает косинус *n,* где *n* в радианах.

Пример: *SELECT COS(60* 3.14159265359/180) "Cosine of 60 degree" FROM DUAL;*

Cosine of 60 degree
.5

♦ *COSH(n)* - возвращает косинус гиперболический *n.*

Пример: *SELECT'COSH(O) "Hyperbolic cosine of0" FROM DUAL;*

Hipebolic cosine of 0
1

♦ *EXP* - возвращает число *e* в степени *n.*

Пример: *SELECTEXP(4) "e to the 4th power" FROM DUAL;*

e to the 4th power
54.59815

♦ *FLOOR/n)* - возвращает наибольшее целое число, которое меньше или равно *n.*

Пример: *SELECT FLOOR(12.7) "Floor" FROM DUAL;*

Floor
12

♦ *LN(n)* - возвращает натуральный логарифм *n.*

Пример: *SELECT LN(95) "Natural log of 95" FROM DUAL;*

Natural log of 95
4.55.387689

♦ *LOG(m,n)* -возвращает логарифм *n* при основе *m.*

Пример: *SELECT LOG(10,100) "Los base 10 of 100" FROM DUAL:*

Log base 10 of 100
2

♦ *MOD(m.n)* - возвращает остаток деления *m на n.*

Пример: *SELECTM0D(12,5) "Modulus" FROM DUAL;*

Modulus
2

♦ *POWER(m, n)* - возвращает *m* в степени *n.*

Пример: *SELECT POWER(2,3) "Raised" FROM DUAL:*

Raised
8

♦ *ROUND(n[,m])* - округляет значение *n* до *m* знаков после десятичной точки. Если *m* пропущено, *n* округляется до целого числа. Можно задать отрицательное значение *m*-для округления слева от десятичной точки.

Пример: *SELECT ROUND(12.194,1) "Round" FROM DUAL;*

Round
12.2

♦ *SIGN(n)* - возвращает знак *n.* Если *n*<0 возвращает -1, если *n*=0 возвращает 0, если *n*>0 возвращает 1.

Пример: *SELECT SIGN(-9.5) "Sien" FROM DUAL:*

Sign
-1

♦ *SIN(n)* - возвращает синус и, где *n* в радианах.

Пример: *SELECTSIN(30*3.14159265359/180) "Sine of 30 degree" FROM DUAL;*

Sine of 30 degree
.5

♦ *SINH(n)* - возвращает синус гиперболический *n.*

Пример: *SELECT SINH(1) "Hyperbolic sine of 1" FROM DUAL;*

Hiperbolic sine of 1
1.17520119

♦ *SQRT(n)* - возвращает корень квадратный из *n.*

Пример: *SELECT SQRT(4) "Square root" FROM DUAL;*

Square root
2

♦ *TAN(n)* - возвращает тангенс *n*, где *n* в радианах.

Пример: *SELECTTAN(45*3.14159265359/180) "Tangent of 45 degree" FROM DUAL;*

Tangent of 45 degree
1

♦ *TANH(n)* - возвращает тангенс гиперболический *n.*

Пример: *SELECT TANH(.5) "Hyperbolic tangent of.5" FROM DUAL;*

Hyperbolic tangent of ,5
.462117157

♦ *TRUNC(n[,m])* - возвращает срезанное до *m* знаков после десятичной запятой значение *n*. Если *m* пропущено, возвращает целую часть числа *n.* Можно задать отрицательное значение *m* для среза слева от десятичной запятой.

Пример: *SELECT TRUNC(127.94,-1) "Truncate" FROM DUAL;*

Truncate
120

2.2.2. Функции знаковых строк

♦ *CHR(n)* - по заданным десятичным кодам *n* возвращает символ.

Пример: *SELECT CHR(129) "Character" FROM DUAL;*

Character
Б

♦ *CONCAT(char1, char2)* - возвращает объединение знаков для *char1* и *char2.* Эта функция эквивалентна знаковой операции объединения | |.

Пример: *SELECTCONCAT(CONCAT(NAMES, 'e '), SCDEGREE) "Научное звание" FROM readers WHERE chairno=20 AND readerno=10;*

Научное звание
Петров доцент

♦ *INlTCAP(char)* - возвращает знаковую строку любого слова. Результат имеет прописную первую букву и строчные буквы для остальных знаков. Слова должны быть составлены только буквами и цифрами. Знак различен от буквы или цифры считается разделителем между словами.

Пример: *SELECTINITCAP('the soap') "Capitalized" FROM DUAL;*

Capitalized
The Soap

♦ *LOWER(char)* - возвращает строку *char*, преобразуя все ее буквы в строчные.

Пример:*SELECT LOWER('ПЕТРОВ ДОЦЕНТ) "Lowercase" FROM DUAL;*

Lowercase

Петров доцент

♦ *LPAD(char1, n [,char2])* - возвращает строку *char1* заполненную слева до *n* знаков знаками *char2*. По умолчанию *char2 "* - пробел.

Пример: *SELECT LPAD('1.2',12) "LPAD пример" FROM DUAL;*

LPAD пример

1.2

♦ *REPLACE(char1, search_string [,replacement_string])* - возвращает строку *char1*, в которой каждая подстрока *search_string* заменена строкой *replacement_string*. Если *replacemant string* пропущена или *null*, каждая подстрока *search_string* стирается.

Пример: *SELECT REPLACE('JACK AND JUE','J','BL') "Changes" FROM DUAL;*

Changes

BLACK AND BLUE

♦ *RPAD(char1, n [,char2])* - возвращает строку *char1* заполненную с правой стороны до *n* знака знаками *char2*. По умолчанию *char2 "* - пробел.

Пример: *SELECT RPAD('1.2', 12, '0') "RPAD пример" FROM DUAL;*

RPAD пример

1.2000000000

♦ *SUBSTR(char, m [,n])* - возвращает подстроку для *char с m-го* знака от начала *char* и с длиной *n* знаков. Если *m* отрицательно, то с *m*-го знака от конца *char*. Если *n* пропущено, функция возвращает подстроку до конца *char*. Не допускается, чтобы *m* и *n* были равны нулю, или чтобы *n* было отрицательное.

Примеры: *SELECT SUBSTR('ABCDEFG'. 3.2) "Substring" FROM DUAL:*

Substring

CD

SELECT SUBSTR('ABCDEFG'.-3,2) "Reversed Substring" FROM DUAL;

Reversed Substring

EF

♦ *TRANSLATE(char, from, to)* - возвращает *char* с подмененными знаками. *From*-строка задает список знаков для подмены, а *to*-строка задает конктерные знаки, которыми заменяют знаки *char*. Знаки *char*, которые не встречаются в *from*, не меняются. Строки *from* и *to* должны иметь одинаковую длину.

Пример: Если хотим, чтобы в тексте *'ИВАНОВ : ИНФОРМАТИКА - 4.5'* все буквы сменились на *X*, и все цифры сменились на *9*, то следует выполнить следующую команду:

SELECT TRANSLATE('ИВАНОВ : ИНФОРМАТИКА -4.5' '0123456789АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЮЯ' '9999999999XXXXKXXXXXXXXXXXXXXXXXXXXXXX') "Translate" FROM DUAL;

Translate
XXXXX : XXXXXXXXXXX – 9.9

♦ *UPPER(char)* - возвращает строку *char*, преобразуя все ее буквы в прописные.

Пример: *SELECT UPPER('Петров доцент') "Uppercase" FROM DUAL;*

Uppercase
ПЕТРОВ ДОЦЕНТ

2.2.3. Функции знаковых строк, возвращающие числовое значение

♦ *ASCII(char)* - возвращает десятичный код первого знака строки *char*.

Пример: *SELECT ASCII('Б') FROM DUAL;*

ASCII('Б')
129

♦ *INSTR(char1, char2 [,n [,m]])* - возвращает номер позиции, в которой *char2* встречается *m*-й раз в *char1*, начиная с позиции *n* строки *char1*. Если *n* отрицательно, то поиск начинает с конца *char1*. Значение *m* должно быть положительным. Если *n* и *m* пропущены, их подразумевающиеся значения равны 1. Если поиск неуспешен, возвращается 0.

Пример: *SELECT INSTR('CORPORATE FLOOR', 'OR',3,2) "Instring" FROM DUAL;*

Instring
14

♦ *LENGTH(char)* - возвращает длину строки *char* в количестве знаков, включая пробелы в начале или конце строки.

Пример: *SELECT LENGTH ('Кандидат') "Length in characters" FROM DUAL;*

Length in characters
8

2.2.4. Функции данных типа DATE

♦ *ADD_MONTHS(d,n)* - возвращает дату *d* плюс *n месяцев.* Если *d* - последний день месяца и получаемый месяц не имеет такого

количества дней, то в качестве результата возвращается дата последнего дня получаемого месяца.

Пример: *SELECT SYSDATE, ADD_MONTHS(SYSDATE,1) "Next month"*
FROM DUAL;

SYSDATE	Next month
10-JAN-97	10-FEB-97

♦ *LAST DAY(d)* - возвращает последний день месяца, заданного датой *d*.

Пример: *SELECT SYSDATE, LAST_DAY(SYSDATE) "Last"*
LAST_DAY(SYSDATE) - SYSDATE "Days Left" FROM DUAL;

SYSDATE	Last	Days Left
10-JAN-97	31-JAN-97	21

♦ *MONTHS BETWEEN(d1,d2)* - возвращает количество месяцев между датами *d1* и *d2*. Если *d1* после *d2*, результат положительный, а если наоборот, то результат отрицательный. Если *d1* и *d2* - содержат один и тот же день месяца, результат - целое число. В противном случае, результат - дробное число, где дробная часть - оставшееся количество дней до целого месяца разделенное на 31.
Пример: *SELECT MOUNTHS_BETWEEN('11-FEB-97','10-JAN-97') "Mounths" FROM DUAL;*

Mounths
1.03235806

♦ *SYSDATE* - возвращает текущую системную дату и время. Эту функцию нельзя применять в условиях ограничений, заданных командой *CHECK*.

Пример: *SELECT TO_CHAR(SYSDATE, 'DD-MM-YYYY HH24:MM:SS') "Сейчас" FROM DUAL;*

Сейчас
14-01-1997 10:55:12

2.2.5. Функции преобразования данных

♦ *TO_CHAR(d [,fmt])* - преобразует дату *d* в символьную строку типа *VARCHAR2*. Поле *fmt* определяет формат. Элементы формата приведены ниже в таблице. Если формат пропущен, подразумевается *'DD-MON-YY'*.

Элемент	Значение
YYYY	Год с 4 цифрами.
YYY, YY или Y	Год с 3, 2 или 1 цифрой.
Q	Четверть года.
MM	Месяц с двумя цифрами от 01 до 12.
RM	Месяц с римскими цифрами от I до XII.
MONTH	Имя месяца на английском до 9 знаков с пробелами.
MON	Три знака для имени месяца на английском языке.
WW	Неделя года (1-53), где первая неделя начинается с первого дня года и продолжается до седьмого дня года.
IW	Стандартная неделя года.
W	Неделя месяца (1-5), где первая неделя начинается с первого дня месяца и продолжается до седьмого дня месяца.
DDD	День года с 1 до 366.
DD	День месяца с 1 до 31.
D	День недели с 1 до 7.
DAY	Наименование дня на английском языке с пробелами до 9 знаков.
DY	Аббревиатура имени дня.
AM или PM	Индикатор времени (до или после обеда).
HH или HH1	Час дня от 1 до 12.
HH24	Час дня от 0 до 23.
MI	Минуты часа от 0 до 59.
SS	Секунды от 0 до 59.
-/,.;:'text'	Знаки пунктуации или текст, который вставляется в соответственные позиции результата.

Пример: *SELECT TO_CHAR(SYSDATE, 'DAY, DD MONTH YYYY') "Сейчас" FROM DUAL;*

Сейчас

Monday, 13 January 1997

TOCHAR(n [fmt]) - преобразует число *n* типа *NUMBER* в строку типа *VARCHAR2.* Формат преобразования определяется полем *fmt.* Элементы формата приведены ниже в таблице. Если формат пропущен, подразумевается '99.99', притом цифр до и после десятичной точки столько, сколько нужно для записи числа.

Элемент	Пример	Описание
9	9999	Цифра от 0 до 9. Ведущие нули и нули в конце числа, после десятичной точки, возвращаются как пробелы.
0	0999 99.90	Ведущие нули и нули в конце числа, после десятичной запятой, возвращаются как 0.
$	$9999	Возвращает знак доллара перед числом.
MI	999MI	Возвращает минус после отрицательного значения.
S	S9999	Возвращает плюс при положительном значении, и минус при отрицательном.
,	9,999	Возвращает запятую в данную позицию.
.	999.9	Возвращает десятичную точку в данную позицию.
V	9999V99	Умножает значение на 10 в степени *n*, где *n* - число после V.
EEEE	.9EEEE	Возвращает число в экспоненциальном формате.
RN rn	RN	Возвращает число с прописными или строчными римскими цифрами. Значение должна быть между 1 и 3999.

Пример:

SELECT TO_CHAR(0.00001123, '9.999EEEE') "Char" FROM DUAL;

Ck

1.123E-05

♦ *TO_DATE(char [fmt])* - преобразует строку *char* типа *CHAR* или *VARCHAR2* в значение типа *DATE* по заданному формату *fmt,* который имеет те же элементы, что и элементы функции *TOCHAR.* Если формат пропущен, подразумевается *'DD-MON-YY'.*

Пример: *SELECT TO_CHAR(TO_DATE('14-01-97','DD-MM-YY'), 'DD-MON YYYY') "Convertion " FROM DUAL;*

Convertion

14-JAN-1997

Для задания формата преобразования даты и чисел можно использоват модификаторы *FM* и *FX,* которые контролируют использование пробелов и точное соответствие между данными и форматом. Модификаторы можно использовать больше чем один раз в один формат. Первое их задание включает их действие в первой части знаков, второе задание исключает их действие во второй части знаков, третье задание включает их действие в третьей части знаков и т.д.

Модификатор *FM* не позволяет вывод пробелов при преобразовании данных с функцией *TO_CHAR.*

Примеры: *SELECT TO_CHAR(SYSDATE, 'DD MONTH, YYYY') "Witt blanks" FROM DUAL;*

With blanks
14 January, 1997

SELECT TO_CHAR(SYSDATE, 'FMDD MONTH, YYYY') "Without blanks" FROM DUAL;

Without blanks
14 January, 1997

При преобразовании числовых данных функция *TO_CHAR(123.456,'9.9EEEE')* дает результат ' *1.2E+02',* в то время как *TO_CHAR(123.456, 'FM9.9EEEE')* возвращает результат '*1.2E+02'.* Модификатор *FX* требует точного соблюдения формата при задании данных в функции *TO_DATE.* Если модификатор не задан, можно использовать и другие разделители между форматными элементами и между ними может находиться произвольное количество пробелов. Например, возможно следующее задание данных для преобразования: *TO _DATE('15/JAN/1997', 'DD-MON-YYYY'),* в то время как *TO _DATE('15/JAN/1997', 'FXDD-MON-YYYY')* вызовет ошибку.

Чтобы правильно задать данные, необходимо записать:

TO _DATE('15-JAN-1997', 'FXDD-MON-YYYY').

♦ *TO_NUMBER(char [fmt])* - преобразует строку *char* типа *CHAR* или *VARCHAR2* в значение типа *NUMBER* по формату *fmt,* который имеет те же элементы как и при функции *TO_CHAR.* Если формат пропущен, *char* должна быть действительной числовой *ORACLE* - константой.

Пример: *SELECT TO_CHAR(TO_NUMBER('0.00001123'), '9.999EEEE') "Char" FROM DUAL;*

Char
1.123E-05

♦ *CHARTOROWID(char)* - преобразует строку *char* типа *CHAR* или *VARCHAR2* в значение типа *ROWID.*

Пример: *SELECT name FROM chairs*
WHERE ROWID=CHARTOROWID('0000000F. 0001.0002');

NAME
Математика

♦ *ROWIDTOCHAR(rowid)* - преобразует значение аргумента типа *ROWID* в значение типа *VARCHAR2.*

Пример: *SELECT ROWID FROM chairs WHERE name= 'Математика';*

ROWID
0000000F.0001.0002

2.2.6. Групповые функции. Групповые функции возвращают одно значение в результате обработки многих строк, которые отвечают

одному и тому же условию. Вместе с аргументом любой групповой функции используется одна из опций *DISTINCT* или *ALL.* Перная опция определяет включение одной строки в обработку при повторяющихся значениях, а вторая - включение всех строк во время обработки при повторяющихся значениях. Например, при опции *DISTINCT* количество строк, содержащихся в данном столбце 1, 1, 1 и 3 равно 2, а при опции *ALL* - 4. Опция *ALL* является подразумевающейся.

♦ *AVG([DISTINCT+ALL] n)* - возвращает среднее арифметическое значение данных в столбце типа *NUMBER.*

Пример: вывод факультетских номеров и успеваемости студентов:

SELECTdeptno, AVG(grade) "Средняя успеваемость" FROM grades GROUP BY deptno;

DEPTNO	Средняя успеваемость
11123	4.57
21756	4.14

♦ *COUNT([DISTINCT+ALL] expr)* - возвращает количество строк удовлетворяющих данному условию.

Пример: вывод количества студентов с двойками:

SELECT COUNT(D1STINCT deptno) "Общее количество студентов с двойками"FROM grades WHERE grade=2;

Общее количество студентов с двойками
35

♦ *MAX([DISTINCT+ALL] expr)* - возвращает максимальное значение выражения.

Пример: вывод факультетского номера студента с самой высокой успеваемостью и значения его среднего балла успеваемости:

SELECT deptno, MAX(AVG (grade)) "Успеваемость" FROM grades GROUP BY deptno;

DEPTNO	Успеваемость
23612	4.97
12624	4.97

♦ *MIN([DISTINCT+ALL] expr)* - возвращает минимальное значение.

Пример: вывод факультетского номера студента с самой низкой успеваемостью и его средний балл успеваемости:

SELECT deptno, MIN(A VG(grade)) "Успеваемость " FROM grades GROUP BY deptno;

DEPTNO	Успеваемость
31245	3.52

♦ *STDDEV([D1STINCT+ALL] expr)* - возвращает среднее квадратическое отклонение выражения.

Пример: квадратный корень дисперсии средней успеваемости студентов:

SELECT STDDEV((AVG(grade)) "Standart Deviation"
FROM grades GROUP BY deptno;

Standart Deviation
2.53406

♦ *SUM([DISTINCT+ALL] expr)* - возвращает сумму значений выражения.

Пример: вывод суммы оценок студента с факультетским номером 23186:

SELECT SUM(grade) "Общая сумма" FROM grades WHERE deptno=23186;

Общая сумма
204.83

♦ *VARIANCE([D1STINCT+ALL] expr)* - возвращает дисперсию выражения.

Пример: дисперсия всех оценок всех студентов:

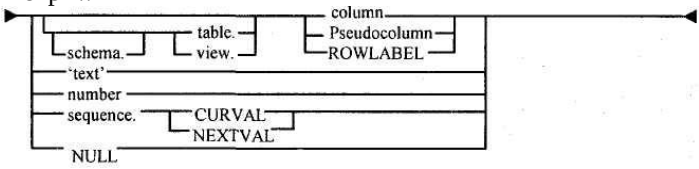
SELECT VARIANCE(grade) "Variance" FROM grades;

Variance
84725.21

2.3. Выражения и условия

Выражения используются для составления условий, аргументов функции или SQL - команд. При описании команд задаются ограничения на выражения, которые могут в них использоваться. Выражения имеют различные формы. Не допускается использование выражений всех форм во всех командах SQL. Выражение формы 1 описывается следующей синтаксической диаграммой:

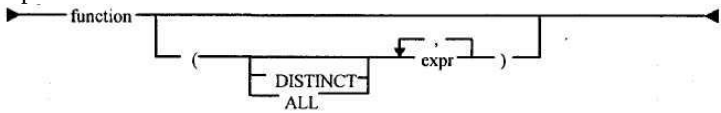
exprl::=



Параметр *pseudocolumn* может быть *ROWID* или *ROWNUM.* Этот параметр может быть использован только с таблицей, его нельзя использовать с видом.

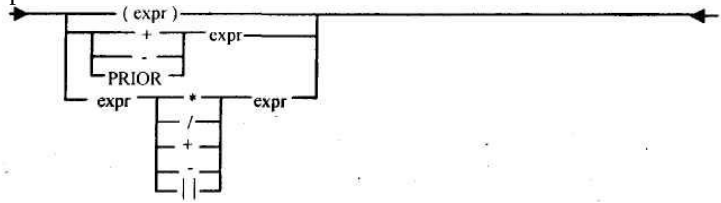
Для вызова функций используется выражение формы 2 со следующей синтаксической диаграммой:

expr2::=

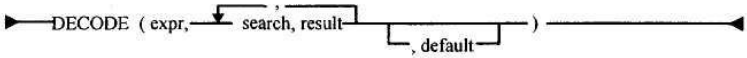


Для комбинирования выражений формы 1 и формы 2 используется выражение 3 со следующей синтаксической диаграммой:

expr3::=



Для преобразования данных используется специальное выражение *DECODE* со следующей синтаксической диаграммой: DECODE_expr::=



При выполнении этого выражения, ORACLE сравнивает значение *expr* с каждым значением *search* и если найдет совпадение возвращает как результат соответствующее значение *result.* Если в процессе сравнения не произойдет совпадения, в качестве результата возвращается значение *default,* или *null,* если параметр *default* пропущен. При сравнении, ORACLE автоматически преобразует *expr* и каждый *search* в тип первого *search.* Результат всегда преобразовывается в тип первого *result.* Если первый *result* типа *CHAR* или *null,* то возвращается значение типа *VARCHAR2.* Максимальное количество всех параметров, включая: *expr, search, result* и *default,* - 255.

Пример: вывод имени кафедры, если ее номер 10 или 20 и вывод "Другие" в остальных случаях:

SELECT chairno, DECODE(chairno, 10, 'Информатика', 20, 'Математика', 'Другие') FROM chairs;

CHAIRNO	CHAIRNO
10	Информатика
20	Математика
30	Другие

В некоторых SQL командах можно использовать список выражений, которые задаются следующей синтаксической диаграммой:

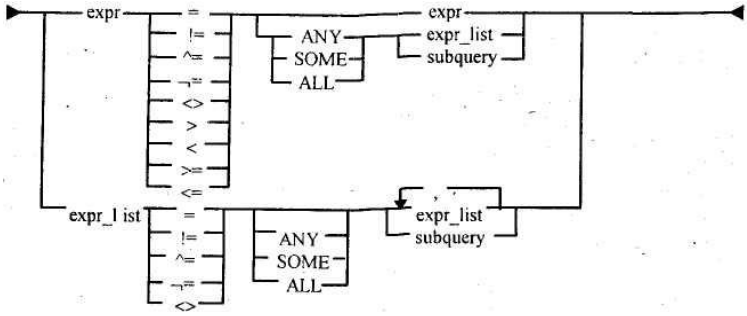
expr_list::=

Примером списка выражений является список: (10, 20, 30).

Выражения могут использоваться в списке выводимых значений командой *SELECT,* с предложениями *WHERE* и *HAVING,* с предложениями *CONNECT BY, START WITH* и *ORDER BY* команды *SELECT,* с предложением *VALUES* команды *INSERT* и с предложением *SET* команды *UPDATE.*

Когда в SQL командах есть параметр *condition,* необходимо использовать условие с одним из операторов *IN, BETWEEN, IS NULL, EXISTS* или *LIKE,* или использовать условие, имеющее одну из следующих двух форм:

condition 1::=



condition2::=

Условия могут быть использованы в условии *WHERE* команд *DELETE, SELECT* и *UPDATE.* Кроме того, могут быть использованы с предложениями *START WITH, CONNECT BY* и *HAV1NG* команды *SELECT.*

3. Команды SQL

3.1. Команды определения, изменения и стирания объектов

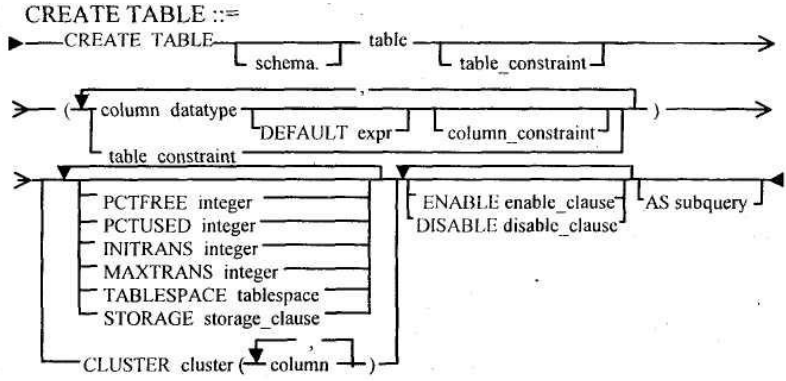
3.1.1. Команда определения таблицы. Создание таблицы - основная структура, которая используется для сохранения данных пользователей, связано с определением следующей информации:

⇒ определение столбцов таблицы;

⇒ ограничения целостности;
⇒ табличное пространство, в котором содержится таблица;
⇒ характеристики сохраняемых данных;
⇒ кластер сохраняемых данных;
⇒ данные для записи в таблицу произвольного запроса.

Чтобы создать таблицу в собственной схеме, пользователь должен иметь привилегию *CREATE TABLE*. Для создания таблицы в схеме другого пользователя, он должен иметь привилегию *CREATE ANY TABLE*. Пользователь также должен иметь свободное табличное пространство в квоте табличного пространства, которое будет содержать таблицу или иметь привилегию *UNLIMITED TABLE SPACE*.

Синтаксис команды создания таблицы приведен на следующей диаграмме:



Значение ключевых слов и параметров следующее:

- *schema* - определяет схему, в которой будет создаваться таблица. Если параметр пропущен, таблица создается в схеме пользователя, который выполнил данную команду;
- *table* - определяет имя таблицы, которая создается;
- *column* - определяет имя столбца таблицы. Количество столбцов может быть до 254;
- *datatype* - задает тип данных столбца;
- *DEFAULT* - задает значение, которое должно присваиваться столбцу, если в команде *INSERT*, которая добавляет строки, пропущено значение для этого столбца. Тип выражения должен соответствовать типу столбца. Выражение не может содержать обращения к другим столбцам, к псевдостолбцам *CURRVAL*, *NEXTVAL* и *ROWNUM* или к константам для дат, которые не определены полностью;

- *column constraint* - определяет ограничение целостности как часть определения столбца. Описание ограничения целостности столбца задается после описания параметров команды;
- *table constraint* - определяет ограничение целостности как часть определения таблицы. Описание ограничения для целостности таблицы приводится после описания параметров команды;
- *PCTFREE* - определяет процент объема каждого табличного блока данных, который сохраняется для будущих коррекций табличных строк. Задаваемый процент должен быть числом в диапазоне от 1 до 99. Значение 0 является флагом того, что весь блок был заполнен вставкой новых строк. Подразумевающееся значение 10%;
- *PCTUSED* - определяет минимальный процент используемого объема, который ORACLE поддерживает для каждого блока данных. В одном блоке можно вставить строки, когда его используемый объем меньше *PCTUSED* процентов. Подразумевающийся процент для *PCTUSED* - 40. Сумма процентов *PCTFREE* и *PCTUSED* должна быть меньше 100;
- *INI TRANS* - определяет количество транзакций делающих запись, которые распределены для каждого блока данных таблицы. Это количество может находиться в пределах от 1 до 255, а подразумевающееся значение-1. Рекомендуется не менять подразумевающееся значение. Любая транзакция, которая корригирует блок, требует записи в блок. Этот параметр определяет минимальное количество конкурентных транзакций, которые могут быть корригированы блоком и помогает устранить накапливание динамически распределенных транзакций записи;
- *MAXTRANS* - определяет максимальное количество транзакций, производящих коррекции, которые распределены для каждого блока данных таблицы. Это количество находится в интервале от 1 до 255, а подразумевающееся значение зависит от размера блока. Лучше, чтобы подразумевающееся значение не менялась.
- *TABLE SPACE* - определяет табличное пространство, в котором создается таблица. Если параметр пропущен, таблица создается в подразумевающемся табличном пространстве пользователя;
- *STORAGE* - определяет характеристики сохранения данных таблицы. Описание этого ключевого слова находится после описания параметров команды. Это ключевое слово определяет значения для больших таблиц;

- *CLUSTER* - определяет, что таблица будет частью кластера. Столбцы в предложении - это табличные столбцы, которые включены в столбцах кластерного ключа. Вообще столбцы кластерного ключа - это столбцы первичного ключа или часть столбцов первичного ключа. Столбцы должны быть заданы в той же строке, что и столбцы кластерного ключа. Если используется это предложение, нельзя задавать параметры *PCTFREE, PCTUSED, INITRANS, MAXTRANS, TABLESPACE* и предложение *STORAGE;*
- *ENABLE* - это ключевое слово определяет ограничения целостности и разрешает их. Описание приводится после описания параметров команды;
- *DISABLE* - это ключевое слово определяет ограничения целостности и запрещает их. Описание приводится после описания параметров команды.

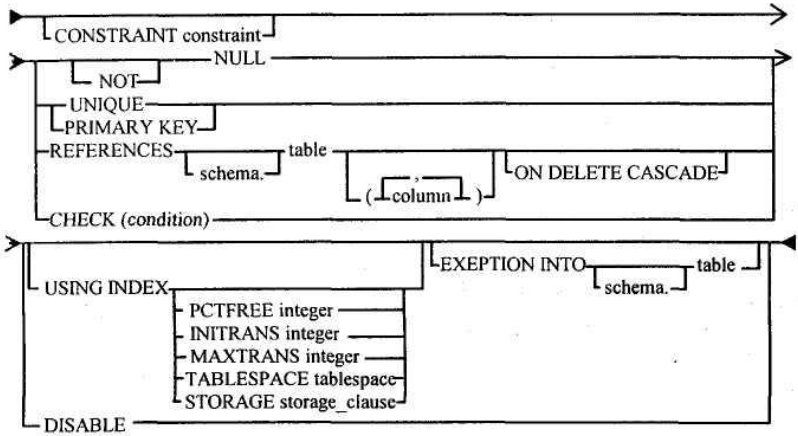
Ограничения целостности можно определить и в параметрах *table constraint* и *columnconstraint,* где они могут быть разрешены или запрещены ключевыми словами *ENABLE* или *DISABLE.* Если соответствующее ключевое слово пропущено, подразумевается, что ограничение разрешено. В команде *CREATE TABLE* нельзя использовать предложение для разрешения или запрещения триггеров. После того как таблица создана, ограничения целостности и триггеры могут разрешать или запрещать команду *ALTER TABLE;*

- *AS subquery* - служит для вставки строк в созданной таблице, полученных в результате запроса из существующих таблиц или видов. Если включить это предложение, определения столбцов определяют только их имена, подразумевающиеся значения и ограничения целостности, но не тип и длину данных. ORACLE определяет тип и длину столбцов на основе данных запроса. Так же автоматически задаётся ограничение *NOT NULL* в столбце новой таблицы, если оно существует в соответствующем столбце запрашиваемой таблицы и данные не меняются запросом SQL-функций или операторов. В этом случае команда *CREATE TABLE* не может содержать одновременно предложение *AS subquery* и соответствующее определение ограничения целостности. Количество столбцов в таблице должно быть равно количеству выражений в запросе. Если все выражения находятся в запросе столбца, имена столбцов в определенной таблице можно пропустить. В этом случае имена столбцов таблицы те же самые, что и имена столбцов опроса.

3.1.1.1. Ключевое слово *CONSTRAINT* служит для определения ограничения целостности таблицы и столбцов. Оно может быть использовано в командах *CREATE TABLE* и *ALTER TABLE.*

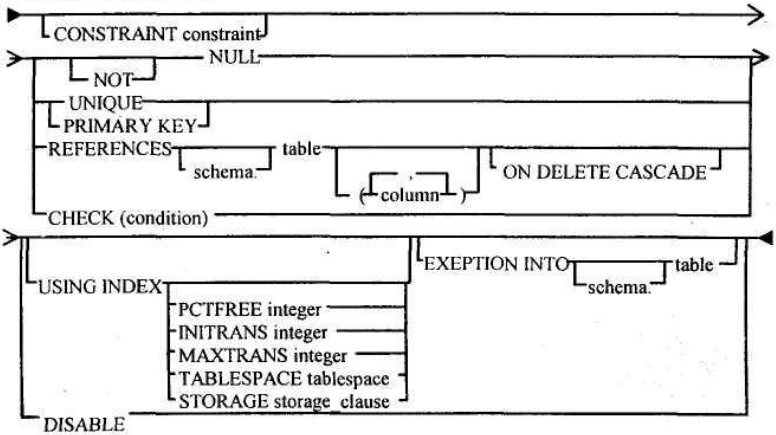
Синтаксическая диаграмма ограничения целостности таблицы следующая:

table constraint: :=



Синтаксическая диаграмма ограничения целостности столбцов:

column_constraint::=



Описание параметров и ключевых слов следующее:

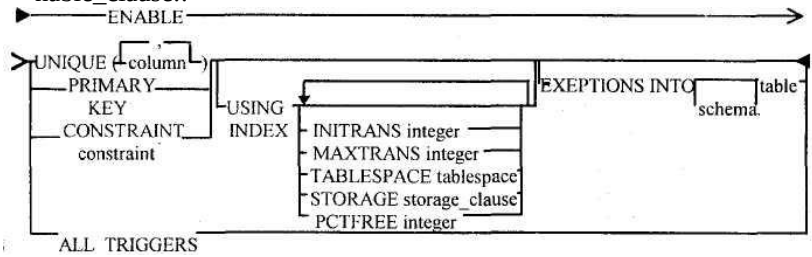
- *CONSTRAINT* - идентифицирует ограничение с именем *constraint.* ORACLE сохраняет это имя в словаре данных вместе с определением ограничений. Если пропустить имя, ORACLE генерирует его в формате *SYS_Cn,* где *n* целое число, которое делает имя уникальным в базе данных. Имена и определения ограничения целостности можно вывести запросом к словарю данных;

- *NULL* - определяет, что в столбце могут содержаться значения типа *null;*
- *NOT NULL* - определяет, что в столбце нельзя иметь значения типа *null.* Если *NULL* и *NOT NULL* пропущены, подразумевается *NULL;*
- *UNIQUE* - определяет определенный столбец или группу столбцов как уникальный ключ;
- *PRIMARY KEY* - определяет столбец или группу столбцов как первичный ключ таблицы;
- *FOREIGN KEY* - определяет столбец или столбцы как внешний ключ;
- *REFERENCES* - определяет первичный или уникальный ключ связной таблицы, который связан с внешним ключом созданной таблицы;
- *ON DELETE CASCADE* - определяет, что ORACLE поддерживает связанную целостность автоматическим стиранием соответствующих значений внешнего ключа, если стереть связанные с ними значения первичного или уникального ключа;
- *CHECK* - определяет условие, которому должна отвечать каждая строка таблицы;
- *USING INDEX* - определяет параметры индекса, который ORACLE создаст, чтобы поддерживать ограничения уникального или первичного ключа. Параметры такие же, как и в команде *CREATE TABLE.* Это выражение допустимо только тогда, когда используется *UNIQUE* или *PRIMARY KEY;*
- *EXCEPTIONS INTO* - определяет таблицу, в которой должны быть записаны данные для строк, которые нарушают ограничения целостности. Таблица должна существовать, прежде чем использовать это предложение;
- *DISABLE* - запрещает ограничение целостности. Если данное ограничение запрещено, то ORACLE его не применяет. В слуае, если этот параметр пропущен, ограничение автоматически применяется. Ограничения могут быть разрешены или запрещены предложениями *ENABLE* или *DISABLE* команд *CREATE TABLE* или *ALTER TABLE.*

3.1.1.2. Предложение *ENABLE* используется для разрешения одного ограничения целостности или для разрешения всех триггеров связанных с данной таблицей. Для разрешения всех триггеров необходимо использовать команду *ALTER TABLE.* Для разрешения одного триггера, *ENABLE* должно использоваться в команде *ALTER TRIGGER.*

Синтаксическая диаграмма предложения *ENABLE* следующая

nable_clause::=



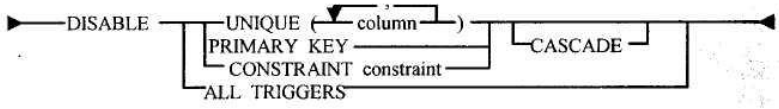
Описание параметров и ключевых слов следующее:

- *UNIQUE* - разрешает ограничение *UNIQUE,* дефинированное для определения столбца или столбцов;
- *PRIMARY KEY* - разрешает ограничение *PRIMARY KEY,* которое было задано во время создания таблицы;
- *CONSTRAINT* - разрешает ограничение с именем *constraint*;
- *USING INDEX* - определяет параметры индекса, который ORACLE создаст, чтобы поддержать ограничения уникального или первичного ключа. Параметры такие же, как и в команде *CREATE TABLE.* Это предложение допустимо только при использовании *UNIQUE* или *PRIMARY KEY;*
- *EXCEPTIONS INTO* - определяет таблицу, в которую заносятся данные для строк, нарушающих ограничения целостности. Таблица должна существовать, прежде чем это выражение будет использовано;
- *ALL TRIGGERS* - разрешает все триггеры связанные с таблицей. Этот параметр может быть использован в предложении *ENABLE* только с командой *ALTER TABLE.* Когда одно ограничение целостности разрешается командой *ALTER TABLE,* ORACLE проверяет каждую строку в таблице, выполняется ли для нее ограничение. Если все строки удовлетворяют ограничению, оно разрешается. Если какая - то строка не удовлетворяет ограничению, ORACLE возвращает сообщение о том, что ограничение не разрешено. Если ограничение уже разрешено, ORACLE проверяет всегда ли данные команд *INSERT, UPDATE* и *DELETE,* которые изменяют содержание таблицы, удовлетворяют ограничению. Если новые данные удовлетворяют ограничению, ORACLE выполняет команду. Если новые данные нарушают ограничение, ORACLE не выполняет команду и выводит сообщение о нарушении ограничения. Когда данная строка нарушает ограничение целостности, информация об этом записывается в таблице заданной предложением *EXEPTIONS INTO.* Эта таблица должна

содержать столбцы указанного типа со следующими данными: тип *ROWID* для *ROWID* строки, нарушающей ограничение; тип *VARCHAR2* для имени собственника таблицы; тип *VARCHAR2* для имени таблицы; тип *VARCHAR2* для имени ограничения, заданного предложением *CONSTAINT.*

3.1.1.3. Предложение *DISABLE* используется для запрета одного ограничения целостности или для запрета всех триггеров, связанных с данной таблицей. Для запрета всех триггеров следует использовать команду *ALTER TABLE.*

Синтаксическая диаграмма для DISABLE следующая: disable_clause::=

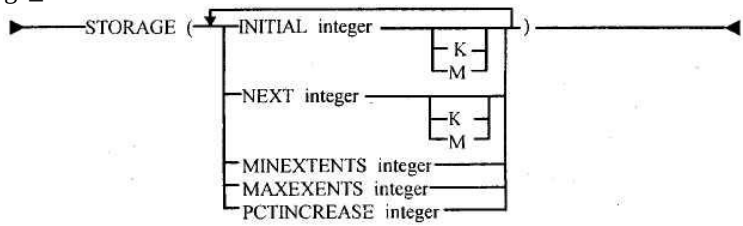


Описание параметров и ключевых слов:

- *UNIQUE* - запрещает ограничение UNIQUE заданное для определенного столбца или столбцов;
- *PRIMARY KEY* - запрещает ограничение *PRIMARY KEY,* которое было задано во время создания таблицы;
- *CONSTRAINT* - запрещает ограничение с именем *constraint*;
- *CASCADE* - запрещает любое ограничение целостности, которое зависит от заданного в предложении ограничения;
- *ALL TRIGGERS* - запрещает все триггеры связанные с таблицей. Этот параметр может быть использован с предложением *DISABLE* только с командой *ALTER TABLE.*

3.1.1.4. Предложение *STORAGE* можно использовать для задания характеристик сохранения данных в различных схемных объектах и в табличных пространствах. Предложение можно использовать во время создания и изменения следующих схемных объектов: таблицы, индексы, кластеры, rollback сегменты и табличные пространства.

Синтаксическая диаграмма предложения *STORAGE* следующая: storage_clause::=



Описание параметров и ключевых слов следующее:

- *INITIAL* - определяет размер первого экстента объекта в байтах. Использование *K* или *M* определяет объем памяти, соответственно в килобайтах или мегабайтах. Подразумевающийся объем имеет размер 5 блоков для данных. Минимальный объем это 2 блока для данных, а максимальный размер зависит от операционной системы (ОС);
- *NEXT* - определяет объем любого следующего экстента объекта в байтах. Использование *K* или *M* определяет объем, соответственно в килобайтах или мегабайтах. Подразумевающийся объем - 5 блоков данных, минимальный объем -1 блок данных, а максимальный зависит от ОС;
- *PCTINCREASE* - определяет процент, с которым объем любого экстента после второго будет нарастать в сравнении с объемом предыдущего экстента. Подразумевающееся значение составляет 50, что означает, что каждый новый экстент будет на 50% больше предыдущего. Минимальное значение - 0, это означает, что все экстенты, после второго, будут иметь одинаковый объем. Максимальный размер зависит от ОС;
- *MINEXTENTS* - определяет количество экстентов, которые будут распределяться для объекта, когда он будет создан. Подразумевающееся минимальное значение равно 1, за исключением rollback сегмента, для которого это значение равно 2. Максимальное количество зависит от ОС;
- *MAXEXTENTS* - определяет максимальное количество экстентов, которые будут распределяться для объекта. Минимальное количество равно 1, а максимальное и подразумевающееся зависит от размера блока данных.

Пример1. Следующая команда создает таблицу кафедр *(chairs),* которая содержит следующие столбцы: номер кафедры *(chairno),* имя кафедры *(name)* и имя факультета *(depart).*

```
CREA TE TABLE chairs
 ( chairno NUMBER(3) CONSTRAINT pk_chairs PRIMARY KEY,
 name VARCHAR(20) CONSTRAINT nn__name NOT NULL,
CONSTRAINT upper_name CHECK (name=UPPER(name)),
depart VARCHAR(20))
 TABLESPACE small;
```

2. Следующая команда создает таблицу преподавателей *(readers),* которая содержит следующие столбцы: номер кафедры *(chairno),* номер преподавателя *(readerno)* кафедры, научная степень преподавателя *(scdegree)* и его имя *(names).*

```
CREATE TABLE readers
( chairno NUMBER(3) CONSTRAINT fk_creaders
```

REFERENCES chairs (chairno),
readerno NUMBER(2),
CONSTRAINTpk readers PRIMARY KEY (chairno, readerno),
scdegree VARCHAR2(8),
names VARCHAR2(30) CONSTRAINT nn_name)
TABLESPACE middle;

Ограничение *nn_name* определено в команде создания таблицы кафедр.

3. Следующая команда создает таблицу студентов *(students),* которая содержит следующие столбцы: факультетский номер *(deptno)* студента, его имя *(names),* специальность *(spec)* и семестр *(sem).*

CREATE TABLE students
(deptno NUMBER(5) CONSTRAINT pk_students PRIMARY KEY,
names VARCHAR2(30) CONSTRAINT nn_name,
spec VARCHAR2(20) CONSTRAINT nn_name,
sem NUMBER(2) DEFAULT 1
CONSTRAINT ck__sem CHECK (sem BETWEEN 1 AND 10)) TABLESPACE middle;

Как и в предыдущей команде, ограничение *nn_name* определено в команде создания таблицы кафедр.

4. Следующая команда создает таблицу балов *(grades)* студентов, которая содержит следующие столбцы: факультетский номер *(deptno)* студента, номер кафедры *(chairno),* номер преподавателя *(readerno)* кафедры, учебная дисциплина *(subject)* и бал *(grade).*

CREATE TABLE grades
(deptno NUMBER(5) CONSTRAINT fk_grades 1
REFEFENCESstudents (deptno),
chairno NVMBER(3),
readerno NUMBER(2),
CONSTRAINT fk_grades2 FOREIGN KEY (chairno, readerno)
REFERENCES readers (chairno. readerno),
subject VARCHAR(40),
grade NUMBER(3,2) CONSTRAINT valid_grade
CHECK (grade BETWEEN 2 AND 5)
sem NUMBER(2) CONSTRAINT ch_sem)
TABLESPACE big;

3.1.2. Команда изменения определения таблицы. Команду *ALTER TABLE* можно использовать, чтобы изменить определение таблицы:

⇒ добавляя столбец;

⇒ добавляя ограничение целостности;

⇒ переопределяя столбец по типу, размеру, подразумевающемуся значению;

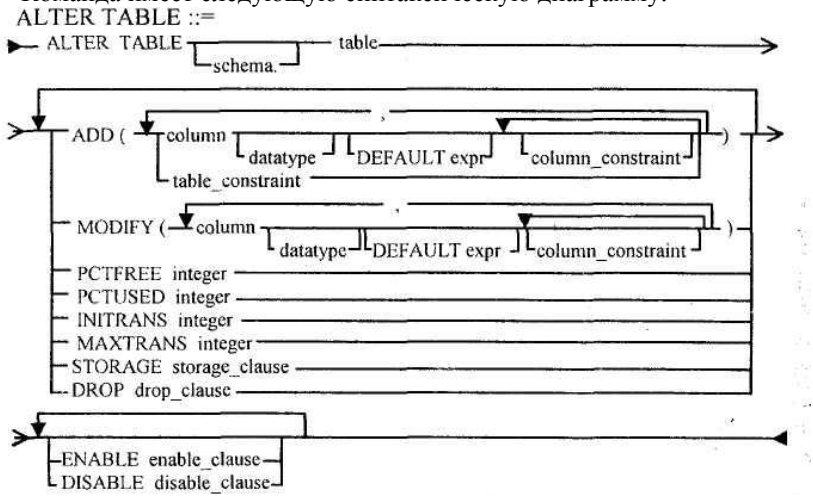
⇒ модифицируя характеристики сохранения или других параметров;

⇒ разрешая, запрещая или устраняя ограничение целостности или триггера;

⇒ распределяя заданный экстент.

Пользователь должен иметь привилегию *ALTER* или *ALTER ANY TABLE,* чтобы выполнитm эту команду.

Команда имеет следующую синтаксическую диаграмму:



Описание ключевых слов и параметров:

- *schema* - определяет схему, в которой содержится таблица;
- *table* - определяет имя таблицы, которая будет изменена;
- *ADD* - определяет добавление столбца или ограничение целостности;
- *MODIFY* - определяет изменение определения столбца;
- *column* - определяет имя столбца, который добавляется или изменяется;
- *datatype* - определяет тип нового столбца или новый тип существующего столбца;
- *DEFAULT* - определяет подразумевающееся значение нового столбца или новое подразумевающееся значение существующего столбца;

- *column constraint* - добавляет ограничение или устраняет *NOT NULL*-ограничение для столбца;
- *table constraint* - добавляет табличное ограничение целостности;
- *PCTFREE, PCTUSED, INITRANS, MAXTRANS* - определяют новые значения соответствующих параметров таблицы. Значение параметров то же самое, что и в команде *CREATE TABLE;*
- *STORAGE* - изменяет характеристики сохранения таблицы;
- *DROP* - устраняет ограничение целостности. Описание этого предложения дано после описания параметров команды;
- *ENABLE* - разрешает одно ограничение или все триггеры, связанные с таблицей. Описание этого предложения дано после описания команды *CREATE TABLE;*
- *DISABLE* - запрещает одно ограничение или все триггеры, связанные с таблицей. Описание этого предложения дано после описания команды *CREATE TABLE;*

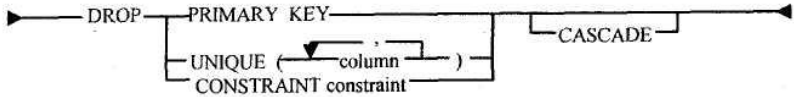
Если добавляется новый столбец, то все строки в этом столбце содержат *null.* Если есть вид, который базируется на таблице с добавленным столбцом и в запросе определяющего вида используется звездочка (*) для выбора всех столбцов базовой таблицы, ORACLE не добавит автоматически новый столбец в вид. Чтобы включить новый столбец в вид, надо выполнить команду *CREATE VIEW* с параметром *OR REPLACE.*

Предложение *MODIFY* можно использовать, учитывая некоторые ограничения. Тип данных в столбце может изменяться и размер столбца может уменьшаться, только если все строки в этом столбце имеют значение *null.* Размер столбца может увеличиваться во всех случаях до максимально допустимого размера. При смене подразумевающегося значения для данного столбца, оно не изменяется для существующих строк, а присваивается новым вставленным строкам. Можно добавить новое ограничение, связанное со столбцом только тогда, когда *NOT NULL* и все строки имеют значение отличное от *null* в этом столбце. Другие ограничения целостности, которые связаны с данным столбцом, могут добавляться с использованием предложения *ADD* как табличные ограничения.

Предложение *DROP* устраняет ограничение целостности БД. Это предложение может быть использован только в команде *ALTER TABLE.*

Предложение имеет следующую синтаксическую диаграмму:

drop_clause ::=



Описание ключевых слов и параметров следующее:

- *PRIMARY KEY* - стирает табличное ограничение *PRIMARY KEY;*
- *UNIQUE* - стирает ограничение *UNIQUE* для заданного столбца;
- *CONSTRAINT* - стирает ограничение целостности заданное с именем *constraint;*
- *CASCADE* - стирает все другие ограничения целостности, которые связаны с данным ограничением.

Примеры: 1. Следующая команда изменяет определение таблицы преподавателей, добавляя столбец для зарплаты *(sal)* и столбец для руководителя *(mngr)* факультета. В столбце руководителя факультета значение равно *null.* В том же столбце руководители кафедр имееют кодовый номер декана, который со своей стороны, состоит из номера кафедры соединенного с номером на кафедре. В столбце руководителя преподаватели имеют кодовый номер руководителя кафедры, который состоит из номера кафедры соединенного с номером на кафедре. *ALTER TABLE readers ADD (sal NUMBER(6), mngr NUMBER(5));*

2. Следующий пример показывает использование предложения *STORAGE* для задания параметров сохранения индекса, который используется для поддерживания первичного ключа таблицы студентов.

ALTER TABLE students PRIMARY KEY USING INDEX
STORAGE (INITIAL 5K NEXT 10K);

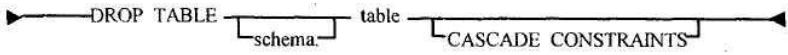
3. Следующая команда показывает использование предложения *DROP* для стирания ограничения таблицы кафедр для ввода названий кафедр прописными буквами.

ALTER TABLE chairs DROP CONSTRAINT upper_name;

3.1.3. Команда стирания таблицы. Команда *DROP TABLE* устраняет (стирает) таблицу и все ее данные из базы данных. Чтобы выполнит эту команду, пользователь должен иметь привилегию *DROP ANY TABLE.*

Команда имеет следующую синтаксическую диаграмму:

DROP TABLE ::=



Описание ключевых слов и параметров следующее:

- *schema* - определяет схему, в которой содержится подлежащая удалению таблица. Если параметр пропущен, имеется ввиду таблица из схемы пользователя, который задал команду;

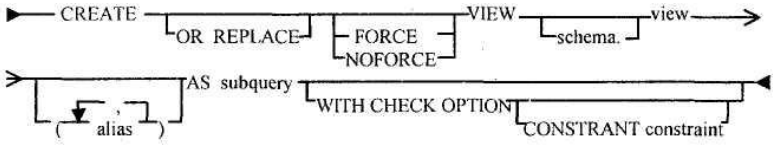
- *table* - определяет имя таблицы, которая стирается;
- *CASCADE CONSTRAINT* - стирает все связанные с таблицей ограничения целостности, которые базируются на первичном и уникальном ключе, заданном в удаленной таблице. Если этот параметр пропущен и такие ограничения существуют, ORACLE возвращает сообщение об ошибке и не удаляет таблицу. Вместе с таблицей, стираются все табличные индексы, независимо от способа их создания. Если таблица не является частью кластера, ORACLE возвращает все блоки данных табличных пространств, содержащие таблицу и ее индексы для нового использования. Если таблица является базовой для вида или используется в сохраненных функциях, процедурах или пакетах, соответственные объекты становятся недействительными, но не стираются. Эти объекты можно стереть или использовать после нового создания таблицы. Таблицу можно создать снова и связанные с ней объекты можно будет использовать опять. Для этого она должна иметь предыдущее имя и содержать столбцы с прежними именами и типами данных, которые используются в объектах, связанных с таблицей.

Пример: Следующая команда стирает таблицу кафедр и связанные с ней ограничения.

DROP TABLE chairs CASCADE CONSTRAINT;

3.1.4. Команда определения вида. Команда *CREATE VIEW* определяет вид, который базируется на одной или более таблицах или видах. Чтобы использовать эту команду, пользователь должен иметь привилегию *CREATE VIEW* или *CREATE ANY VIEW*. Собственник вида должен иметь привилегии для выбора, вставки, исправления и стирания строк во всех таблицах и видах, на которых базируется созданный вид. Команда имеет следующую синтаксическую диаграмму:

CREATE VIEW ::=



Описание ключевых слов и параметров:

- *OR REPLACE* - создает снова вид, если он уже существует. Этот параметр используется для изменил определения существующих видов;

- *FORCE* - определяет создание вида независимо от того, существуют ли базовые таблицы вида, или собственник вида имеет привилегию их использовать;
- *NOFORCE* - определяет создание вида, только если базовые таблицы вида существуют и собственник вида имеет привилегию их использовать.
- *schema* - определяет схему, в которой будет создаваться вид. Если параметр пропущен, подразумевается схема, содержащая данную команду;
- *view* - определяет имя вида, который создается;
- *alias* - имена выбранных выражений запроса, определяющих вид. Эти имена становятся именами столбцов вида. Они могут быть пропущены только тогда, когда в запросе список выражений содержит только имена столбцов или символ звездочки (*), что означает все столбцы. В этом случае имена столбцов запроса становятся именами столбцов вида;
- *AS subquery* - запрос, который определяет столбцы и строки таблицы (таблиц), которая задает вид. Запросом может быть любая команда *SELECT* без предложений *ORDER BY* и *FOR UPDATE.* Список выражений запроса может включать до 254 выражений. Больше информации можно получить из описания команды *SELECT;*
- *WITH CHECK OPTIONS* - определяет, что вставка и корригированные данные, должны быть в строках, которые выбираются из запроса вида. Этот параметр не может быть использован, если вид базируется на запросе другого вида. Параметр *CONSTRAINT* определяет имя, которое присвоено ограничению *CHECK OPTION.* Если имя пропущено, ORACLE автоматически присваивает ограничению имя *SYSCn,* где *n* целое число, которое обеспечивает уникальность имени в базе данных. Когда задается запрос, определяющий вид, необходимо соблюдать следующие ограничения. Запрос не может содержать псевдостолбцы *CURRVAL* и *NEXTVAL.* Если в запросе включены *ROWID, ROWNUM* или *LEVEL,* то они должны иметь псевдоимена, определенные в запросе. Если в запросе используется звездочка (*) для задания всех столбцов базовой таблицы, запрос трансформируется в содержащиеся имена всех столбцов базовой таблицы при создании вида. Поэтому, когда добавляется новый столбец в базовой таблице, вид должен создаваться снова с параметром *OR REPLACE.* С видом можно использовать команды *DELETE, INSERT, UPDATE* и *SELECT* для работы как с данными, так и с таблицами. Команды *DELETE, INSERT* и *UPDATE* нельзя использовать, если запрос вида содержит соединения данных

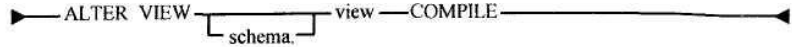
нескольких таблиц или видов, устанавливающих операторы объединения выбранных значений нескольких запросов, групповые функции, предложения *GROUP BY* или *START WITH* и оператор *DISTINCT.*

Пример: Следующая команда создает вид, который имеет столбцы с именами студентов и все столбцы таблицы с баллами студентов.
CREATE VIEW n grades

*SELECT names, grade. * FROM students, grades*

WHERE students, deptno = grades, deptno;

3.1.5. Команда перекомпилирования вида. Команда *ALTER VIEW* используется для перекомпилирования вида. Это необходимо производить тогда, когда ORACLE сделал данный вид недействительным, по причине изменения таблицы из его базовых таблиц. Команда имеет следующую синтаксическую диаграмму:
ALTER VIEW: :=



Описание ключевых слов и параметров:

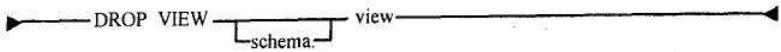
- *schema* - определяет схему, в которой содержится вид. Если параметр пропущен, то подразумевается вид из схемы пользователя.
- *view* - определяет имя вида, который будет перекомпилироваться.

Ключевое слово *COMPILE* определяет перекомпилирование вида и обязательно должно быть использовано.

Во время выполнения команды *ALTER VIEW* определение вида не изменяется. Для изменения определения вида, необходимо выполнить команду *CREATE VIEW* с параметром *OR REPLACE.*

3.1.6. Команда стирания вида. Команда *DROP VIEW* устраняет (стирает) вид из базы данных. Чтобы выполнить эту команду, пользователь должен иметь привилегию *DROP ANY VIEW.*

Команда имеет следующую синтаксическую диаграмму:
DROP VIEW ::=



Описание ключевых слов и параметров:

- *schema* - определяет схему, в которой содержится стираемый вид. Если параметр пропущен, то подразумевается вид из схемы пользователя, который задал команду;
- *view* - определяет имя вида, который стирается.

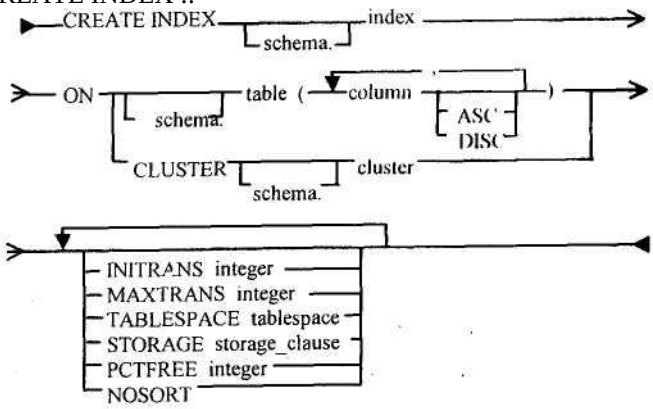
Если вид является базовым для другого вида или используются в сохранен- I ных функциях, процедурах или пакетах, соответствующие объекты становятся неактуальными, но не стираются. Определение вида может быть изменено - вид стирается и создается снова. Пример:

Следующая команда | стирает вид с именами и баллами студентов.
DROP VIEW ngrades;

3.1.7. Команда определения индекса. Команда *CREATE INDEX* создает индекс по одному или более столбцам таблицы. При использовании индекса столбцов, выбор строк ускоряется. Чтобы выполнить эту команду, пользователь должен иметь привилегию *CREA TE ANY INDEX.*

Команда имеет следующую синтаксическую диаграмму:

CREATE INDEX ::=



Описание ключевых слов и параметров:

- *schema* - определяет схему, в которой будет содержаться созданный индекс. Если параметр пропущен, то имеется ввиду индекс из схемы пользователя, который задал команду;
- *index* - определяет имя индекса, который создается;
- *table* - определяет таблицу, для которой создается индекс. Если пропущена схема таблицы, то таблица выбирается из схемы пользователя;
- *ASC* или *DESC* - определяет упорядочивание индекса. *ACS* определяет

возрастание индекса, а *DESC* - убывание. Если этот параметр пропустить, то индекс будет изменяться в возрастающем порядке;

- *CLUSTER* - определяет кластер, для которого будет создан кластерный индекс. Если в схеме пропущен кластер, то он берётся из схемы пользователя, выполнившего команду;
- *1NITRANS, MAXTRANS, TABLESPACE, STORAGE* и *PCTFREE* - значение этих параметров описано в описании команды *CREATE TABLE;*

• *NOSORT* - показывает ORACLE, что строки сохранены в возрастающем порядке в базе данных и поэтому ORACLE не должен их сортировать при создании индекса.

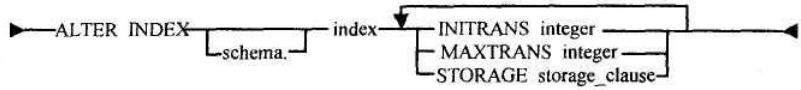
Индекс ускоряет поиск данных в таблице, но замедляет вставки и корректирование строк, так как вместе с таблицей надо корректировать и индекс. При первоначальной записи данных в таблицу, работа выполнится быстрее, если сначала записываются строки, а затем создается индекс. Примеры: 1. Следующая команда создает индекс по столбцам с факультетскими номерами для таблицы баллов студентов. *CREA TE INDEX grades deptno ON grades (deptno);*

2. Приведённая ниже команда создает индекс по столбцам с номерами кафедры и номерами преподавателей на кафедре для таблицы баллов студентов. *CREA TE INDEX grades chaimoreaderno*

ON grades (chairno, readerno);

3.1.8. Команда изменения определения индекса. Командой *ALTER INDEX* можно изменять только характеристики сохранения индекса в базе данных. Чтобы выполнить эту команду, пользователь должен иметь привилегию *ALTER ANY INDEX.* Команда имеет следующую синтаксическую диаграмму:

ALTER INDEX::=



Описание ключевых слов и параметров:

• *schema* - определяет схему, содержащую изменяемый индекс. Если параметр пропущен, то имеется ввиду индекс из схемы пользователя, задавшего команду;

• *index* - определяет имя индекса, который изменяется.

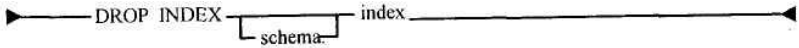
• *INITRANS, MAXTRANS* и *STORAGE* - значение этих параметров описано ;; при описании команды *CREATE TABLE.*

Пример: Следующая команда изменяет определение индекса *grades deptno, ' ~i* который создается командой (пример 1) определения индекса, задавая па-' \ раметры для сохранения индекса.

ALTER INDEX grades Jeptno IN1TRANS 5 STORAGE (NEXT 100);

3.1.9. Команда стирания индекса. Команда *DROP INDEX* устраняет индекс базы данных. Чтобы выполнить эту команду, пользователь должен иметь привилегию *DROP ANY INDEX.*

Команда имеет следующую синтаксическую диаграмму: DROP INDEX ::=



Описание ключевых слов и параметров:

• *schema* - определяет схему, в которой содержится индекс. Если параметр *schema* пропущен, то индекс содержится в схеме пользователя, который задал команду;

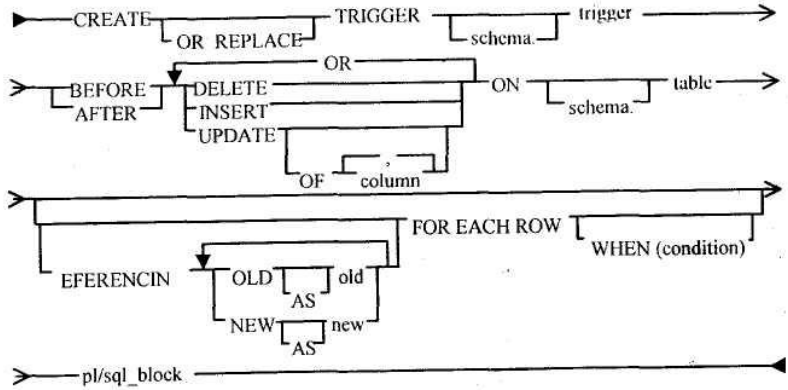
• *index* - определяет имя индекса, который будет удаляться;

Пример: Приведённая ниже команда стирает индекс *grades deptno,* созданный командой в примере 1 для определения индекса. *DROP INDEX grades deptno;*

3.1.10. Команда определения триггера. Команда *CREATE TRIGGER* создает и разрешает триггер базы данных. Триггер представляет PL/SQL-блок, который сохраняется в базе данных, и связан с некоторой таблицей. ORACLE автоматически выполняет триггер в процессе реализации определенной SQL команды над связанными таблицами. Чтобы выполнить команду, пользователь должен иметь привилегию *CREATE TRIGGER* или *CREATE ANY TRIGGER.* Если триггер включает SQL-команды или вызывает сохраненные процедуры или функции, пользователь должен иметь привилегии работы с ними.

Команда имеет следующую синтаксическую диаграмму:

CREATE TRIGGER ::=



Описание ключевых слов и параметров:

• *OR REPLACE* - создает заново триггер, если он уже существует. Этот параметр можно использовать, чтобы изменить определение существующего триггера не стирая его;

• *schema* - определяет схему, в которой будет содержаться созданный триггер, при его отсутствии подразумевается схема пользователя.

• *trigger* - определяет имя триггера, который создается;

• *BEFORE* - определяет, что триггер реализуется до выполнения контролированных команд;

- *AFTER* - определяет, что триггер реализуется после выполнения контролированных команд;
- *DELETE* - определяет, что ORACLE выполнит триггер при стирании строки таблицы;
- *INSERT* - определяет, что ORACLE выполнит триггер при вставке строки в таблицу;
- *UPDATE* - определяет, что ORACLE выполнит триггер при исправлении строки в таблице;
- *OF* - определяет, при корректировке каких строк выполняется триггер. По умолчанию, триггер выполнится, при исправлении любого столбца;
- *ON* - определяет схему и таблицу, с которой свяжется создающийся триггер. По умолчанию берется схема пользователя;
- *REFERENCING* - определяет связанные имена. Связанные имена можно использовать в PL/SQL - блоке и с условием *WHEN* триггера строки, чтобы указать новое и старое значение в текущей строке. Подразумевающиеся связанные имена *OLD* и *NEW*.
- *FOR EACH ROW* - ORACLE задействует триггер каждый раз, когда реализуется соответствующая команда, выполняя PL/SQL - блок для каждой обрабатываемой командой строки. Если параметр не указан, то выполняется триггер команды (или команд). ORACLE выполняет такой триггер каждый раз, когда выполнится соответствующая команда;
- *WHEN* - определяет ограничение триггера. Оно содержит условие, которое должно выполнится, чтобы выполнился триггер. Условие может быть задано только для триггера строк;
- *pl/sqlblok* - определяет одну или несколько команд SQL и его расширения PL/SQL, которые реализуются при выполнении триггера.

Триггер - это сохраненная в базе данных процедура, связанная с некоторой таблицей. ORACLE автоматически выполняет процедуру, когда выполнится соответствующая команда. Триггер используется в следующих случаях: обеспечение сложного доступа и ясной обработки событий; генерирование автоматических значений для столбцов; обеспечение выполнения сложных ограничений.

Синтаксис команды создания триггера имеет следующие части:

- контролированные команды - это команды SQL, при выполнении которых включается триггер. Сюда относятся команды *DELETE*, *INSERT* и *UPDATE*. Команда создания триггера должна включать хотя бы одну из этих команд;
- параметр *ON* - определяет таблицу, которая связана с триггером;

• ограничение триггера - определяет дополнительное условие, которое должно быть удовлетворено, чтобы выполнился триггер.

Действие триггера определяется командами, которые включены в PL/SQL - блоке. Во время создания триггера ORACLE его разрешает. Триггер можно запретить или разрешить командами *ALTER TRIGGER* или *ALTER TABLE.*

Пример: Следующая команда определяет триггер, связанный с таблицей баллов. Он преобразует введенные с произвольными буквами (прописные и строчные) наименования учебных дисциплин в имена с прописными буквами. Такое же ограничение задано и для имен кафедр с помощью *CHECK.* Надо отметить, что когда не соблюдается ограничение, заданное командой *CHECK,* ORACLE выводит сообщение об ошибке и не записывает данные. До тех пор, пока такое ограничение поддерживается триггером, он преобразует имена с прописными буквами и данные всегда записываются. *CREATE TRIGGER upper_grades*

BEFORE INSERT OR UP DA TE

ON grades

BEGIN IF :new.subject<>UPPER(:new.subject)

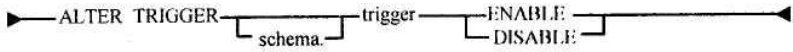
THEN .new.subject: = UPPER(:new.subject); END IF;

END;

3.1.11. Команда изменения определения триггера. Команда *ALTER TRIGGER* используется для разрешения или запрещения триггера. Чтобы выполнилась команда, пользователь должен имен, привилегию *ALTER ANY TRIGGER.*

Команда имеет следующую синтаксическую диаграмму:

ALTER TRIGGER ::=



Описание ключевых слов и параметров:

• *schema* - определяет схему, в которой будет содержаться триггер. Если параметр пропущен, то триггер выбирается из схемы пользователя;

• *trigger* - определяет имя триггера.

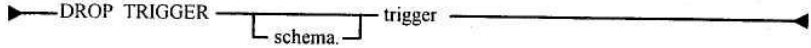
Пример: Данная ниже команда разрешает триггер *upper_grades.* Он пре-

образует имена учебных дисциплин записанных произвольными буквами в имена с прописными буквами. Эта команда должна выполниться после команды изменения определения таблиц студентов, которая приведена в примере 3 команд изменения определения таблицы, после выполнения которой триггер становится недействительным. *ALTER TRIGGER upper grades ENABLE;*

3.1.12. Команда стирания триггера. Команда *DROP TRIGGER* устраняет триггер из базы данных. Чтобы выполнилась эта команда, пользователь должен иметь привилегию *DROP ANY TRIGGER.*

Команда имеет следующую синтаксическую диаграмму:

DROP TRIGGER ::=



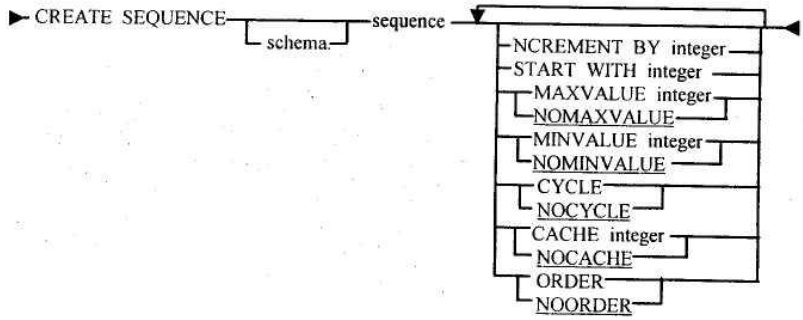
Описание ключевых слов и параметров:

- *schema* - определяет схему, в которой содержится подлежащий удалению триггер. Если параметр пропущен, то устраняется триггер из схемы пользователя, который задал команду;
- *trigger* - определяет имя триггера, который должен быть удален.

Пример: Следующая команда стирает триггер, который обеспечивает наличие прописных букв, имен учебных дисциплин в таблице балов. *DROP TRIGGER upper^grades;*

3.1.13. Команда создания последовательностей. Команда *CREATE SEQUENCE* определяет последовательность, которая генерирует ряд значений. Чтобы создать последовательность в схеме пользователя, он должен иметь привилегию *CREATE SEQUENCE,* а для определения последовательностей в схеме другого пользователя, должен иметь привилегию *CREATE ANY SEQUENCE.* .

Команда имеет следующую синтаксическую диаграмму: CREATE SEQUENCE ::=



Описание ключевых слов и параметров:

- *schema* - определяет схему, в которой будет содержаться создаваемая последовательность. Если параметр пропущен, последовательность относится к схеме пользователя, который задал команду;
- *sequence* - определяет имя последовательности, которая создается;

• *INCREMENT BY* - определяет интервал между двумя соседними по значению членами последовательностей. Этим значением может быть любое положительное или отрицательное число, но оно не может быть нулём. Если значение положительное, то последовательность возрастающая, а если отрицательное - убывающая. Если пропустить это значение, то подразумевающееся число будет равно 1; ■

• *MINVALUE* - определяет минимальное значение в последовательности;

• *NOM1NVALUE* - определяет минимальное значение I для возрастающей последовательности или -($10^{Л}$26) для убывающей последовательности;

• *MAXVALUE* - определяет максимальное значение в последовательности;

• *NOMAXVALUE* - максимальное значение ($10^{Л}$27) для возрастающей последовательности или -1 для убывающей последовательности;

• *START WITH* - определяет значение первого члена последовательности. Это значение должно быть меньше максимального для возрастающей последовательности или больше минимального для убывающей последовательности. Если это значение не задано, то подразумевающееся значение -это минимальное для возрастающей последовательности или максимальное для убывающей последовательности;

• *CYCLE* - определяет, что последовательность продолжит генерировать, члены, циклическим образом, после того как достигнет минимального или максимального значения. После достижения возрастающая последовательностью максимального значения, она начинает новый цикл генерации с минимального значения. После достижения убывающей последовательностью минимального значения, она продолжает генерации с максимального значения;

• *NOCYCLE* - определяет, что последовательность не будет генерировать члены, после того как достигнет минимального или максимального значения. Этот параметр подразумевается;

• *CACHE* - определяет, какое количество значений из последовательности задано предварительно с целью сохранения в памяти для более позднего использования. Минимальная величина этого параметра равна 2;

• *NOCACHE* - отмена ражима *CACHE* ;

Если пропустить *CACHE* и *NOCACHE,* подразумевается *CACHE 20.*

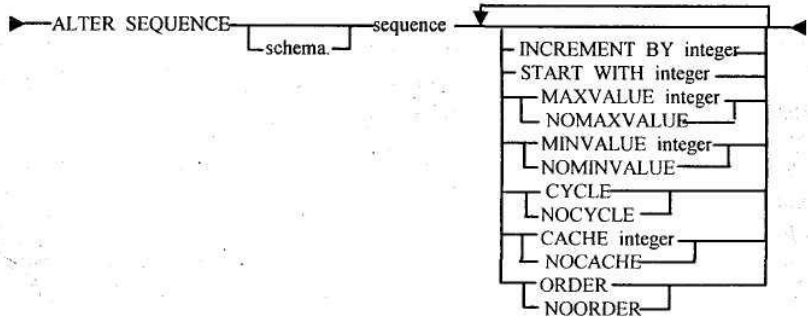
• *ORDER* - гарантирует, что последовательность генерирует свои члены упорядоченным образом;

• *NOORDER* - не гарантирует, что последовательность генерирует свои

члены упорядоченным образом. Этот параметр подразумевается.

Последовательность может быть использована псевдостолбцами *CLJRVAL* и *NEXTVAL.* Псевдостолбец *CURVAL* возвращает текущее значение последовательности, а *NEXTVAL* возвращает новое (следующее) значение. Эти значения можно использовать при формировании уникального или первичного ключа. Пример: Приведённая ниже команда определяет последовательность, которая может быть использована для задания факультетских номеров студентов при их записи в таблицу студентов. Использование этой последовательности можно увидеть в примере добавления строки в таблицу студентов командой *INSERT. CREATE SEQUENCE sseq START WITH 10000 INCREMENT 10; 3.1.14. Команда изменения определения последовательности.* Команда *ALTER SEQUENCE* используется, чтобы изменить некоторые параметры последовательности. Она имеет следующую синтаксическую диаграмму:

ALTER SEQUENCE ::=

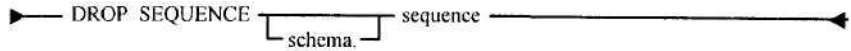


Ключевые слова и параметры имеют те же самые значения, что и в команде *CREATE SEQUENCE.* Изменение определения сказывается только на тех членах последовательности, которые генерируются после выполнения этой команды, а сгенерированные ранее члены не меняются.

Пример: Следующая команда изменяет определение последовательности *sseq,* задавая ей максимальное значение. *ALTER SEQUENCE sseq MAXVALUE 50000;*

3.1.15. Команда удаления последовательности. Команда *DROP SEQUENCE* удаляет последовательность. Для этого она должна быть из схемы пользователя или сам пользователь должен иметь привилегию *DROP ANY SEQUENCE.* Команда имеет следующую синтаксическую диаграмму:

DROP SEQUENCE ::=



Описание ключевых слов и параметров:

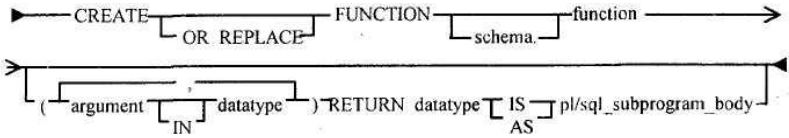
- *schema* - определяет схему, в которой содержится подлежащая удалению последовательность. Если параметр пропущен, то подразумевается последовательность из схемы пользователя;
- *sequence* - определяет имя последовательности, которая удаляется.

Пример: Следующая команда стирает последовательность *sseq.*

DROP SEQUENCE sseq;

3.1.16. Команда создания функции. Команда *CREATE FUNCTION* создает самостоятельно сохраненную в базе данных функцию. Чтобы выполнить эту команду, пользователь должен иметь привилегию *CREATE PROCEDURE* или *CREATE ANY PROCEDURE.* Она имеет следующую синтаксическую диаграмму:

CREATE FUNCTION ::=



Описание ключевых слов и параметров:

- *OR REPLACE* - создает снова функцию, если она уже существует. Этот параметр может быть использован, чтобы изменить определение существующей функции не стирая ее.
- *schema* - определяет схему, в которой будет создаваться функция. Если параметр пропущен, то имеется ввиду функция из схемы пользователя;
- *function* - определяет имя функции, которая создастся;
- *argument* - задает имя аргумента функции. Гхли функция не имеет аргументов, скобки после имени функции следует упустить;
- *datatype* - определяет, какого типа данных должен быть аргумент. В процессе определения типа не нужно задавать длину строки, или формат, и точность числа. Они извлекаются из программной среды, которая вызывает функцию;
- *RETURN datatype* - определяет тип значения, которое возвращается функцией. В процессе определения типа не нужно задавать длину строки, размер и точность числа. Они извлекается из программной среды, которая вызывает функцию;

• *pl/sql_subprogram_body* - определяет тело функции. Тело функции задается на языке PL/SQL, который здесь не рассматривается.

Пример: Следующая команда определяет функцию, которая определяет средний бал по данной дисциплине, для данной специальности данного семестра, которые являются параметрами функции.

CREATE FUNCTION avg_grade (subj VARCHAR2(20), spec NUMBER, sem NUMBER)

RETURN NUMBER IS avg_grNUMBER(3,2); BEGIN , SELECT A VG(grade) INTO avg_gr FROM grades

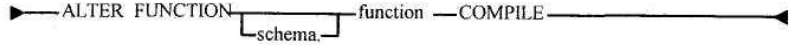
WHERE subject=sebj AND grades, sem =avg_grade. sem and \ deplno IN (SELECT deptno FROM students - '

WHERE students.spec-avg_grade.spec); [f] *RETURN (avg_gr);*

END;

3.1.17. Команда перекомпилирования функции. Команда *ALTER FUNCTION* используется для перекомпиляции функции. Это необходимо производить тогда, когда ORACLE сделал функцию недействительной, так как изменились объекты связанные с нею. Она имеет следующую синтаксическую диаграмму:

ALTER FUNCTION:=



Описание ключевых слов и параметров:

• *schema* - определяет схему, в которой содержится функция. Если параметр пропущен, то имеется ввиду функция из схемы пользователя ;

• *function* - определяет имя функции, которая перекомпилируется.

Ключевое слово *COMPILE* задает перекомпиляцию функции и оно является обязательным.

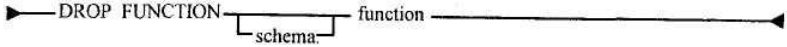
В процессе выполнения команды *ALTER FUNCTION*, функция перекомпилируется независимо от того действительна ли она, или недействительна. Ее определение, однако, не изменяется. Если необходимо изменить определение функции, необходимо выполнить команду *CREATE FUNCTION* с параметром *OR REPLACE*. Пример: Следующая команда перекомпилирует функцию для получения средней успеваемости. Эта команда должна быть выполнена после команды изменения определения таблицы студентов, приведенной в примере 3 команд изменения определения таблицы. После выполнения функция становится недействительной. *ALTER FUNCTION avgjgrade COMPILE;*

3.1.18. Команда стирания функции. Команда *DROP FUNCTION* стирает функцию. Чтобы это произошло функция должна быть из

схемы пользователя или сам пользователь должен иметь привилегию *DROP ANY PROCEDURE.*

Команда имеет следующую синтаксическую диаграмму:

DROP FUNCTION ::=



Описание ключевых слов и параметров:

- *schema* - определяет схему, в которой содержится подлежащая стиранию функция. Если параметр пропущен, то функция берется из схемы пользователя, который задал команду;
- *function* - определяет имя функции, которая будет удаляться;

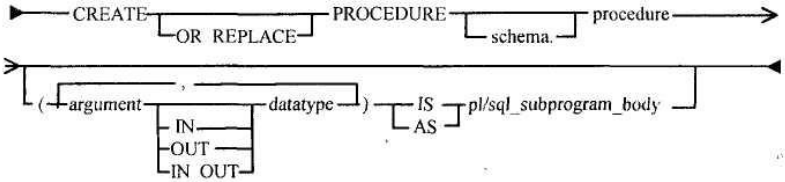
Пример: Следующая команда стирает функцию получения среднего успеха *(avg_grade).*

DROP FUNCTION avg_grade;

3.1.19. Команда создания процедуры. Команда *CREATE PROCEDURE* создает самостоятельно сохраняемую в базе данных процедуру. Чтобы выполнить эту команду, пользователь должен иметь привилегию *CREATE PROCEDURE* или *CREATE ANY PROCEDURE.*

Команда имеет следующую синтаксическую диаграмму:

CREATE PROCEDURE ::=



Описание ключевых слов и параметров:

- *OR REPLACE* - создает заново процедуру, если она уже существует. Этот параметр можно использовать для изменения определения существующих процедур не стирая их;
- *schema* - определяет схему, в которой будет создаваться процедура. По умолчанию, имеется ввиду процедура из схемы пользователя;
- *procedure* - определяет имя процедуры, которая создается.
- *argument* - задает имя аргумента процедуры.
- *IN* - определят необходимость задания значения для соответствующих входных аргументов при вызове процедуры;
- *OUT* - определяет, что процедура вернет значении ныходных аргументов окружению, которое вызвало процедуру;
- *IN OUT* - определяет необходимость задания значения как для входных, так и для выходных аргументов процедуры;

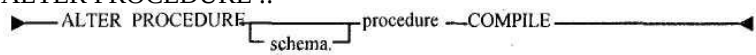
- Если пропустить *IN, OUT* или *IN OUT,* подразумевается *IN.*
- *datatype* - определяет, какого типа данных должен быть аргумент. В процессе определения типа нет нужды задавать длину строки, формат и точность числа. Они определяются из вызывающей программной среды;
- *pl/sql_subprogram_body* - определяет тело процедуры. Тело процедуры представляется на языке PL/SQL, который здесь не рассматривается.

Пример: Следующая команда определяет процедуру, которая выполняется по окончанию учебного года. Она создает таблицу студентов выпускников, производя запись в ней, если нет плохих отметок, стирает выпускников с таблицы студентов, создает таблицу балов выпускников, записывая их в таблицу и стирает отметки выпускников из таблицы баллов. Считается, что срок обучения 10 семестров.

```
CREA TE PROCEDURE new_school_year AS
BEGIN
CREA TE TABLE students 97 AS SELECT * FROM students
 WHERE sem=10 AND deptno NOT IN
(SELECT DISTINCT deptno FROM grades WHERE grade=2);
DELETE students WHERE sem=10;
CREATE TABLE grades 97 AS SELECT * FROM grades
WHERE deptno IN (SELECT deptno FROM students 97);
 DELETE grades
WHERE deptno IN (SELECT deptno FROM students 97);
END;
```

3.1.20. Команда перекомпилирования процедуры. Команда *ALTER PROCEDURE* используется для перекомпилирования процедур. Такая необходимость возникает тогда, когда ORACLE сделал процедуру недействительной из-за изменения объектов, которые с ней связаны. Команда имеет следующую синтаксическую диаграмму:

ALTER PROCEDURE ::=



Описание ключевых слов и параметров:

- *schema* - определяет схему, в которой содержится процедура. Если параметр пропущен, то имеется ввиду процедура из схемы пользователя;
- *procedure* - определяет имя процедуры.

Ключевое слово *COMPILE* определяет перекомпилирование процедуры и его присутствие обязательно.

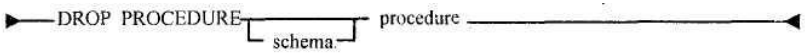
В процессе выполнения команды *ALTER PROCEDURE,* процедура перекомпилируется независимо от того является ли она действительной или недействительной. При этом ее определение не меняется. Если необходимо изменить определение процедуры, следует выполнить команду *CREATE PROCEDURE* с параметром *OR REPLACE.*

Пример: Следующая команда перекомпилирует процедуру окончания учебного года. Эта команда должна выполниться после команды изменения определения таблицы студентов, которая была приведена в примере 3 команд изменения определения таблицы. С выполнением команды процедура становится недействительной. *ALTER PROCEDURE new_school_year COMPILE;*

3.1.21. Команда стирания процедуры. Команда *DROP PROCEDURE* стирает процедуру. Для этого процедура должна быть из схемы пользователя или сам пользователь должен иметь привилегию *DROP ANY PROCEDURE.*

Команда имеет следующую синтаксическую диаграмму:

DROP PROCEDURE ::=



Описание ключевых слов и параметров:

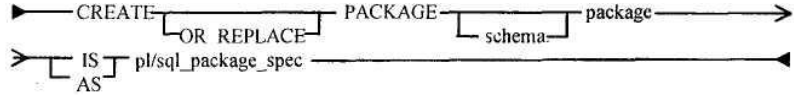
- *schema* - определяет схему, в которой содержится подлежащая стиранию процедура. Если параметр пропущен, подразумевается процедура из схемы пользователя, выполнившего команду;
- *procedure* - определяет имя процедуры, которая будет удаляться.

Пример: Следующая команда удаляет процедуру окончания учебного года.

DROP PROCEDURE new_school_year;

3.1.22. Команда создания пакета. Команда *CREATE PACKAGE* используется для создания спецификации пакета. Пакет - это совокупность процедур, функций и других объектов, сохраняемых как один объект в базе данных. Чтобы выполнить команду, пользователь должен иметь привилегию *CREATE PROCEDURE* или *CREATE ANY PROCEDURE.* Команда имеет следующую синтаксическую диаграмму:

CREATE PACKAGE ::=



Описание ключевых слов и параметров:

- *OR REPLACE* - создает заново пакет, если он уже существует. Этот параметр можно использовать, чтобы изменить определение существующих пакетов не стирая их;

• *schema* - определяет схему, в которой будет создан пакет. Если параметр пропущен, пакет берется из схемы пользователя;

• *package* - определяет имя пакета, который создастся;

• *pl/sql_package_spec* - задает спецификацию пакета, в которой объявляются его объекты. Спецификация пакета представляется на языке PL/SQL. Программные объекты, которые могут быть включены в пакет - это процедуры, функции, переменные, константы, курсоры и исключения. Курсор -это объект, который обеспечивает получение данных с помощью команды *SELECT.* Исключением является таблица, в которой записывается информация о нарушении ограничений целостности. Включенные в спецификацию пакета объекты определяются в теле пакета, что производится командой *CREATE PACKAGE BODY.* Для полного создания пакета необходимо выполнить последовательно команды *CREATE PACKAGE* и *CREATE PACKAGE BODY.*

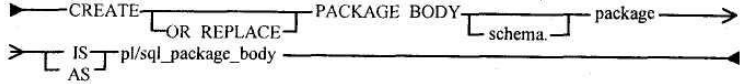
Пример: Приведенная ниже команда определяет спецификацию пакета, который содержит функцию записи студента, процедуру стирания студента и процедуру увеличения зарплат преподавателей, после получения нового научного звания.

```
CREATE PACKAGE st_sal AS
 BEGIN
FUNCTION insert_st (names VARCHAR2(30), spec VARCHAR2(20),
          sem NUMBER(2) ) RETURN NUMBER;
PROCEDURE remov_st ( deptno NUMBER(5) );
PROCEDURE increase_sal ( chairno NUMBER, readerno NUMBER,
          incr_sal NUMBER);
END;
```

3.1.23. Команда создания тела пакета. Команда *CREATE PACKAGE BODY* используется для создания тела пакета. Чтобы выполнить команду, пользователь должен иметь привилегию *CREATE PROCEDURE* или *CREATE ANY PROCEDURE.*

Команда имеет следующую синтаксическую диаграмму:

CREATE PACKAGE BODY ::=



Описание ключевых слов и параметров:

• *OR REPLACE* - создает заново тело пакета, если оно уже существует. Этот параметр может быть использован, чтобы изменить определение существующего тела пакета не стирая его;

• *schema* - определяет схему, в которой будет создано тело пакета. Если параметр пропущен, то имеется ввиду тело пакета из схемы пользователя;

• *package* - определяет имя тела пакета, которое создается;

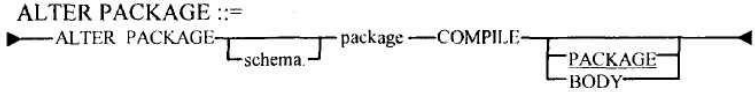
• *pl/sql_package_body* - задает тело пакета, в котором определяются объекты, включенные в пакет. Тело пакета задается на языке PL/SQL.

Пример: Приведенная ниже команда определяет тело пакета, которое содержит функцию записи студента, процедуру стирания студента и процедуру увеличения зарплат преподавателей после получения нового научного звания.

```
CREATE PACKAGE BODY st_sal AS
tot_stds NUMBER; /* определяет переменную пакета */
BEGIN
FUNCTION insert_st ( names VARCHAR2(30), spec VARCHAR2(20),
sem NUMBER(2)) RETURN NUMBER IS
new_deptno NUMBER(5);
BEGIN
SELECTsseq.NEXTVAL INTO new_deptno FROM DUAL;
 INSERT INTO sstudents VALUES (new_deptno, spec, sem);
 tot_stds : = tot_stds + 1;
 RETURN (new_deptno);
 END;
PROCEDURE remove_st ( deptno NUMBER(5) ) IS
BEGIN
DELETE students WHERE students.deptno=remove_st.deptno;
DELETE grades WHERE grades, deptno=remove_st. deptno;
END;
PROCEDURE increase sal ( chairno NUMBER, readerno NUMBER,
          incr_sal NUMBER ) IS
 curr_sal NUMBER(6);
BEGIN
SELECT curr_sal INTO curr_sal FROM readers
 WHERE readers.chairno=increase_sal.chairno AND
 readers.chairno=increase_sal.chairno;
IF curr_sal IS NULL
THEN RAISE no_sal;
ELSE UPDATE readers SET sal:=sal+incr_sal
WHERE readers. chaimo=increase_sal. chairno
AND readers.chaimo=increase_sal.chairno;
 END;
END;
```

3.1.24. Команда перекомпилирования пакета. Команда *ALTER PACKAGE* используется для перекомпилирования пакета. Это необходимо выполнять тогда, когда пакет стал недействительным из-за изменения связанных с ним объектов. Команда имеет синтаксическую диаграмму:

ALTER PACKAGE ::=



Описание ключевых слов и параметров:

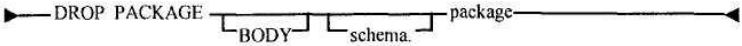
- *schema* - определяет схему, в которой содержится пакет. Если параметр пропущен, то подразумевается пакет из схемы пользователя;
- *package* - определяет имя пакета, который будет перекомпилироваться;
- *COMPILE* - определяет перекомпилирование пакета и обязательно используется.
- *PACKAGE* - определяет спецификации перекомпилируемого тела пакета;
- *BODY*- определяет перекомпилирование только тела процедуры.

Если пропустить *PACKAGE* и *BODY,* подразумевается *PACKAGE.* При выполнении команды *ALTER PACKAGE,* пакет перекомпилируется независимо от того действителен он или недействителен, но его спецификация и определение тела при этом не изменяются. Если необходимо изменить спецификацию пакета или определение тела, надо выполнить команду *CREATE PACKAGE* или *CREATE PACKAGE BODY,* имеющую параметр *OR REPLACE.*

Пример: Следующая команда перекомпилирует пакет *st_sal.* Эту команду надо выполнять после команды изменения определения таблицы студентов, которая приводилась в примере 3 команды изменения определения таблицы. После выполнения команды переопределения таблицы, пакет *st_sal* становится недействительным. *ALTER PACKAGE st_sal COMPILE;*

3.1.25. Команда стирания пакета. Команда *DROP PACKAGE* стирает пакет. Чтобы это сделать, пакет должен быть из схемы пользователя или пользователь должен иметь привилегию *DROP ANY PROCEDURE.* Команда имеет следующую синтаксическую диаграмму:

DROP PACKAGE::=



Описание ключевых слов и параметров:

- *BODY* - стирает только тело пакета. Если пропустить параметр, ORACLE стирает как тело, так и спецификацию пакета;

- *schema* - определяет схему, в которой находится пакет. Если параметр пропущен, то имеется ввиду пакет из схемы пользователя, который задал команду;
- *package* - определяет имя пакета, который будет удаляться.

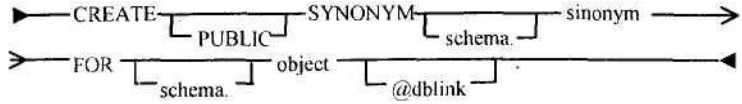
Пример: Данная команда стирает пакет *st_sal.*

DROP PACKAGE st_sal;

3.1.26. Команда создания синонима. Команда *CREATE SYNONYM* создает альтернативное имя таблицы, вида, последовательности, процедуры, функции, пакета или другого объекта. Чтобы выполнить команду, синоним должен быть из схемы пользователя или пользователь должен иметь привилегию *CREATE ANY SYNONYM.*

Команда имеет следующую синтаксическую диаграмму:

CREATE SYNONYM ::=



Описание ключевых слов и параметров:

- *PUBLIC* - определяет создание общего синонима, который доступен всем потребителям. Если пропустить параметр, синоним доступен только потребителю, который его создает;
- *schema* - определяет схему, в которой будет создаваться синоним. Если параметр пропущен, то синоним берется из схемы пользователя;
- *synonym* - определяет имя синонима, который создается;
- *FOR object* - определяет объект, для которого создается синоним. Если пропустить параметр *schema,* определяющий *object,* то подразумевается, что объект из схемы пользователя выполнившего команду.

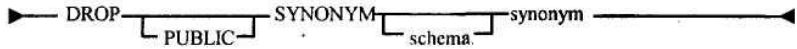
Синоним может быть использован в командах *SELECT, INSERT, UPDATE, DELETE* и *LOCK TABLE.* Когда используется общий синоним, пользователь может не задавать схему, которая содержит синоним, если в его схеме нет объекта с именем синонима.

Пример. Следующая команда определяет синоним таблицы с оценками, который доступен всем пользователям.

CREATE PUBLIC SYNONYM pgrades FOR grades;

3.1.27. Команда стирания синонима. Команда *DROP SYNONYM* стирает синоним. Чтобы это сделать, синоним должен быть из схемы пользователя или пользователь должен иметь привилегию *DROP ANY SYNONYM.* Она имеет следующую синтаксическую диаграмму:

DROP SYNONYM ::=



Описание ключевых слов и параметров:

- *schema* - определяет схему, в которой находится подлежащей стиранию синоним. Если параметр пропущен, то имеется ввиду синоним из схемы пользователя, который задал команду;
- *synonym* - определяет имя удаляемого синонима.

Пример: Данная команда стирает синоним таблицы балов.

DROP PUBLIC SYNONYM pgrades;

3.1.28. Команда переименования объекта. Команда *RENAME* используется для переименования таблицы, вида, последовательности или личного синонима. Чтобы переименовать объект, он должен быть из схемы пользователя, выполнившего команду.

Команда имеет следующий синтаксис:

RENAME old TO new

Описание параметров следующее:

- *old* - задает текущее имя существующей таблицы, вида, последовательности или личного синонима;
- *new* - задает новое имя объекта.

Во время выполнения этой команды, новое имя переименованного объекта заменяет старое имя во всех объектах базы данных, связанных с переименованным объектом.

Пример: Следующая команда переименовывает таблицу с данными кафедр.

RENAME chairs TO dept_chairs;

3.1.29. Команда вывода описания таблицы или вида. Команда *DESCRABE* выводит описание таблицы или вида, показывая имена столбцов, наличие ограничения *NOT NULL* и тип столбцов. Пользователь может выполнить команду для любой таблицы или вида, для которых есть привилегия *SELECT TABLE.* Она имеет следующую синтаксическую диаграмму:

DESCRIBE: :=

Описание ключевых слов и параметров:

- *schema* - определят схему, в которой содержится таблица или вид. Если параметр пропущен, то подразумевается таблица или вид из схемы пользователя, который задал команду;
- *table* или *view* - определяет имя таблицы или вида, для которых выводится описание;
- *dblink* - задает полное или частичное имя связи базы данных с распределенной базой данных, в которой находится таблица или вид. Если этот параметр пропустить, то ORACLE подразумевает, что таблица находится в локальной базе данных.

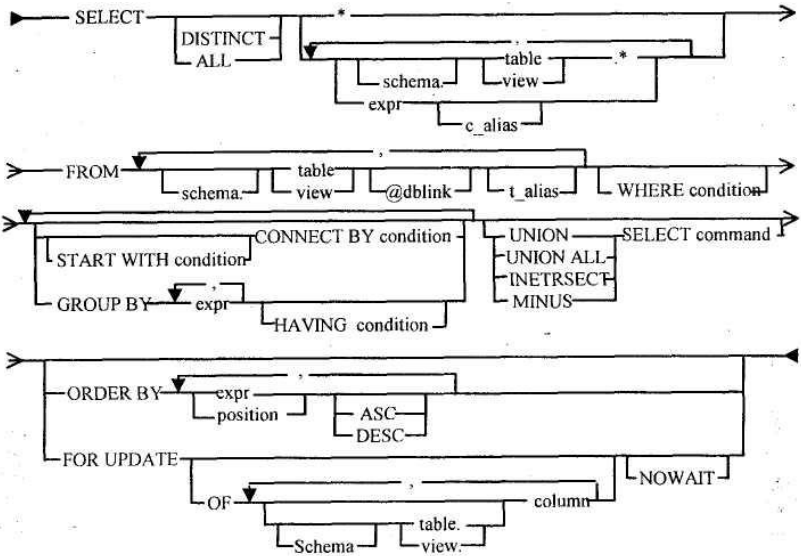
Пример. Следующая команда выводит описание таблицы кафедр *CHAIRS:* *DESC chairs;*

Name	*Null*	*Type*
CHAIRNO	*NOT NULL*	*NUMBER(3)*
NAME	*NOT NULL*	*VARCHAR2(20)*
DEPART		*VARCHAR2(20)*

3.2. Команды работы с данными базы данных

3.2.1. Команда выборки данных из таблицы и вида. Команда *SELECT* возвращает данные из одной или больше таблиц. С помощью этой команды формируется запрос к базе данных для получения определенной информации. Можно выбирать данные из таблицы пользователя. Чтобы получать данные из таблицы другого пользователя, нужно иметь привилегию *SELECT* для этих таблиц. Чтобы выбирать данные из видов, базовые таблицы должны принадлежать пользователю или пользователь должен иметь для них привилегию *SELECT*. Привилегия *SELECT ANY TABLE* позволяет выбирать данные из любой таблицы или вида. Команда имеет следующую синтаксическую диаграмму:

SELECT::=



Описание ключевых слов и параметров:

- *DISTINCT* - определяет режим получения одного экземпляра из любой совокупности дублирующихся строк. (Дублирующиеся строки -

такие строки, которые имеют одинаковые значения для каждого выражения из списка выбранных данных);

- *ALL* - определяет режим получения значений всех выбранных строк, включая дублирующиеся. Параметр *ALL* подразумевается;
- * - задает выбор всех столбцов всех таблиц и видов, заданных предложением *FROM;*
- *table.* * и *view.* * - определяет выбор всех столбцов таблицы и вида. Для определения схемы, отличающейся от схемы пользователя, можно использовать определитель *schema;*
- *expr* - выбор выражения, обычно на основе значения в столбце таблицы или вида, заданных предложением FROM. Синтаксис выражения был приведен выше. В простом случае это имя столбца. Последовательность выражений и столбцов определяет список выбранных данных. В простейшем случае этот список является списком выбранных столбцов. Если в двух или более таблицах есть столбец с одним и тем же именем, то этот столбец должен задаваться именем таблицы и именем столбца, что является его полным именем. Соответственно, полное имя таблицы или вида включает и имя схемы. Иногда ORACLE работает медленнее, если использовать полные имена таблиц и столбцов. Имя столбца в этом списке может включать и определитель *schema,* только если таблица или вид, содержащий столбец, имеет определитель *schema* в предложении *FROM;*
- *c_alias* - задает другое имя выражения, в котором участвует столбец, и определяет имя, которое будет использоваться для заголовка столбца. Это имя не влияет на действительное имя столбца и его нельзя использовать в другом месте запроса;
- *schema* - определяет заданную предложением *FROM* схему таблицы или вида, из которых выбираются данные. Если параметр пропущен, то подразумевается, что таблица или выражения принадлежат пользователю, задавшему команду;
- *table* и *view* - задает имя таблицы или вида из которых выбираются данные;
- *dblink* - задает полное или частичное имя связи БД с распределенной БД, в которой находится таблица или вид. Если этот параметр пропущен, ORACLE подразумевает, что таблица находится в локальной базе данных;
- *t_alias* - задает другое имя, псевдоимя таблицы или вида, которое может быть использовано в других частях той же команды *SELECT.* Другие ссылки на таблицу или вид, чаще всего в других запросах той же команды, нужно делать с использованием псевдоимени;

• *WHERE* - определяет выбранные строки. Выбранными считаются те строки, для которых логическое условие имеет значение TRUE. Если это предложение пропущено, ORACLE выводит все строки таблиц и видов, которые заданы предложением *FROM;*

• *START WITH ... CONNECT BY* - определяет получение содержимого строк в иерархическом порядке;

• *GROUP BY* - группирует выбранные строки по признаку основного значения выражения любой строки и возвращает одну строку с суммарной информацией для каждой группы;

• *HAVING* - определяет ограничение строк из группы, возвращая по группам те строки, для которых условие истинно, т.е. имеет значение *TRUE.* Если пропустить это предложение, ORACLE возвращает все строки в группе;

• *UNION, UNION ALL, INTERSECT* и *MINUS* - определяет объединение строк двух команд *SELECT* с использованием соответствующего оператора установки;

• *ORDER BY* - задает упорядочивание возвращаемых строк. Параметры *expr* задают упорядочивание строк в зависимости от значения выражения. В выражение включен столбец, или столбцы из списка выбранных данных, или другие столбцы из таблицы или вида, заданных предложением *FROM.*

Параметр *position* определяет порядковый номер столбца из списка выбранных столбцов, по значению которого производится упорядочивание. Параметр *ASC* или *DESC* определяет упорядочивание соответственно в возрастающем или убывающем порядке. Подразумевается возрастающий;

• *FOR UPDATE* - запрещает выбранные строки и делает их недоступными для изменения или стирания другими пользователями до окончания транзакции. Обычно, после команды *SELECT,* включающей параметр *FOR UPDATE,* выполняется одна или несколько команд *UPDATE* для изменения данных в запрещенных строках;

• *NOWAIT* - возвращает контроль пользователю если команда *SELECT* пытается запретить строку, которая уже запрещена другим пользователем. Если этот параметр пропущен, то ORACLE ждет пока строка станет доступной и тогда возвращает выбранные командой строки.

Примеры команды *SELECT:*

1. Следующая команда выводит преподавателей с кафедры информатики, которые имеют научное звание профессора.

```
SELECT names FROM readers WHERE chaimo=20 AND
        scdegree='профессор ';
```

2. Следующая команда выводит иерархическую структуру преподавателей, где преподаватели первого уровня - это деканы, второго уровня - руководители кафедр и третьего уровня - все остальные преподаватели.

SELECTLPADC ',2(LEVEL-l))++names OGRjCHART, cairno, readerno, mngr, scdegree FROM readers*
START WITH mngr IS NULL
*CONNECT BY PRIOR chairno * 100+readerno=mngr;*

ORG_CHART	CHAIRNO	READERNO	MNGR	SCDEGREE
Александров	20	10		профессор
Иванов	10	10	2010	доцент
Петров	10	20	1010	доцент
Дмитриев	10	30	1010	ассистент
…	…	…	…	…
Сергеев	20	20	2010	доцент
Алексеев	20	30	2020	ассистент

В этой команде использован оператор *PRIOR.* Он определят, что заданное после него значение для данной строки, определят подчиненные ей строки.

3. Следующая команда выводит иерархическую структуру преподавателей первого и второго уровня.

SELECTLPADf ',2(LEVEL-l))++names OGRjCHART,*
cairno, readerno, mngr, scdegree FROM readers
START WITH mngr IS NULL
CONNECT BY PRIOR chairno 100+readerno=mngr AND LAVEL<=2;*

ORG_CHART	CHAIRNO	READERNO	MNGR	SCDEGREE
Александров	20	10		профессор
Иванов	10	10	2010	доцент
Сергеев	20	20	2010	доцент
…	…	…	…	…

4. Следующая команда иллюстрирует простое соединение данных из двух таблиц. Команда *SELECT* простого соединения имеет следующий синтаксис:

SELECT table1.column1, table2.column2, ...
FROM table1, table2
WHERE table 1.column3=table2.column4...

В примере выводятся имена студентов специальности информатики, первый курс из таблицы *students,* соединенные с отметками из таблицы *grades.*

SELECT names, subject, grade FROM students, grades
WHERE spec='Информатика' AND students.sem=1 AND

students. deptno=grades.deptno AND students.sem=grades.sem;

5. Следующая команда иллюстрирует внешнее соединение данных двух таблиц. Команда *SELECT* для внешнего соединения имеет следующий синтаксис:

SELECT table1. column1, table2.column2, ...
FROM table1, table2
WHERE table1. column3 (+) = table2. column4... или
WHERE table1 .column3 = table2.column4 (+)...

В столбце, после которого записан оператор (+), могут и не быть значения, определенные столбцом без оператора (+). В простом соединении такие данные не выводятся. В случае внешнего соединения ORACLE генерирует строки со значением *null* для столбца, после которого есть (+) и нет соответствующего значения столбца без (+).

Пусть необходимо написать команду вывода отметок по математике для студентов первого курса. Тогда, при простом соединении не выводятся имена студентов, которые не имеют такой оценки, в то время как для внешнего соединения они выводятся.

В следующей команде, которая иллюстрирует простое соединение выводятся только имена студентов, которые имеют отметки.

SELECT names, grade, spec FROM students, grades
WHERE students.sem=1 AND subject= 'Математика'
AND students, deptno = grades, deptno;

NAMES	GRADE	SPEC
Иванов	4.50	Математика
Петров	3.00	Математика
…	…	…

В приведённой ниже команде, при помощи внешнего соединения выведутся имена всех студентов, независимо от того имеют ли они отметки.

SELECT names, grade, spec FROM students, grades
WHERE students.sem=1 AND subject='Математика'
AND students, deptno = grades, deptno (+);

NAMES	GRADE	SPEC
Иванов	4.50	Математика
Алексеев		История
Петров	3.00	Математика
…	…	…

6. Следующая команда иллюстрирует предложение *GROUP BY*. Она выводит средний балл студентов по специальностям и курсам.

SELECT spec, students.sem, AVG (grade)

FROM students, grades
WHERE students, deptno =grades. deptno AND
students.sem=grades.sem
GROUP BY spec, students.sem;

7. Следующая команда иллюстрирует предложение *HAVING.* Она выводит средний балл студентов по специальностям и курсам. Плохие отметки не обрабатываются.

SELECT spec, students.sem, AVG (grade)
FROM students, grades
WHERE students. deptno=grades.deptno AND
students.sem=grades.sem
GROUP BY spec, students.sem HAVING grade> 2;

8. Следующая команда иллюстрирует предложение *ORDER BY.* Она выводит имена студентов специальности математики, упорядоченные по курсам и факультетским номерам.

SELECT sem, deptno, names FROM students
WHERE spec-'Математика' ORDER BY sem, deptno;

SEM	DEPTNO	NAMES.
1	57120	Алексеев
1	57121	Иванов
…	…	…
2	40624	Петров
2	40627	Александров

9. Следующая команда иллюстрирует применение подзапроса. Подзапрос - это использование команды *SELECT* без предложения *OREDR BY FOR UPDATE* с условием *WHERE* другой SQL-команды. В команде *SELECT* подзапрос имеет следующий синтаксис:

WHERE expr operator (SELECT list FROM table1,...
WHERE table1. column1 = table2.colunin2);

В командах *UPDATE* и *DELETE* подзапрос имеет следующий синтаксис:

WHERE column operator (SELECT list FROM table1, ...
WHERE table1.column1 = table2.column2);

В примере выводятся имена, семестр и специальность студентов, которые имеют больше чем 2 двойки. Это является также иллюстрацией использования псевдоимени таблицы и двойного вложения подзапроса. Второй подзапрос возвращает количество двоек студента с факультетским номером, определенный первым подзапросом и задан *al_grades. deptno.* Первый подзапрос возвращает факультетские номера студентов, которые имеют больше чем 2 двойки.

SELECT names, sem, spec FROM students

WHERE deptno IN (SELECT deptno FROM grades al_grades
WHERE 2 <(SELECT COUNT(grade) FROM grades
WHERE grade=2 AND grades. deptno=al_grades. deptno));

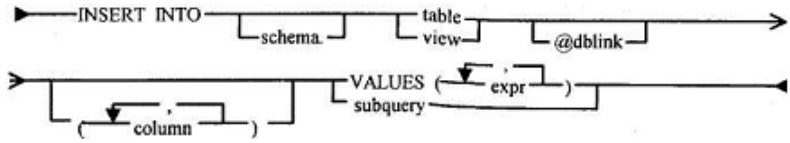
10. Использование таблицы *DUAL.* Это таблица схемы пользователя *SYS,* но доступна для каждого пользователя. Она имеет только один столб *DUMMY* типа *VARCHAR(1)* и содержит только одну строку со значением *X'.* Таблица используется для вычисления выражений командой *SELECT.* С помощью следующей команды выводится текущая дата.

SELECT SYSDA TE FROM DUAL;

3.2.2. Команда вставки строк в таблицу или вид. Команда *INSERT* вставляет строки в таблицу или вид, который основан только на одной таблице. Когда вставляются новые строки в вид, они добавляются, в действительности, в базовую таблицу вида. Чтобы данная команда выполнилась пользователь должен иметь привилегию *INSERT* для таблицы или привилегию *INSERT ANY TABLE.*

Команда имеет следующую синтаксическую диаграмму:

INSERT : :=



Описание ключевых слов и параметров:

• *schema* - определяет схему таблицы или вида, в которую добавляются строки. Если пропустить параметр, то подразумевается, что таблица или вид принадлежат пользователю, задавшему команду;

• *table* и *view* - задает имя таблицы или вида.

• *dblink* - задает полное или частичное имя связи с базой данных распределенной базы данных, в которой находится таблица или вид. Если этот параметр пропустить, то ORACLE подразумевает, что таблица находится в локальной базе данных;

• *column* - задает имя столбца. В предложении *VALUES* должно быть значение для каждого столбца, который включен в список столбцов. Если данный столбец таблицы не задан в этом списке, ему присваивается подразумевающееся значение, которое было задано командой *CREATE TABLE.* Если в команде *CREATE TABLE* не определено подразумевающееся значение, то столбцу будет присвоено значение *null.* Если в команде не задается список столбцов, то в предложении *VALUES* необходимо задать значение для каждого столбца таблицы;

- *VALUES* - определяет строку значений, которую добавляют в таблицу;
- *subquery* - это команда *SELECT,* которая возвращает строки, которые добавляются в таблицу. Список выбранных значений в команде *SELECT* должен иметь столько же элементов, сколько столбцов в команде *INSERT.*

Команду *INSERT* нельзя использовать для добавления строк в вид, в определении которого есть оператор *DISTINCT,* предложение *GROUP BY* или групповые функции.

Примеры:

1. Следующая команда добавляет строку в таблице оценок.

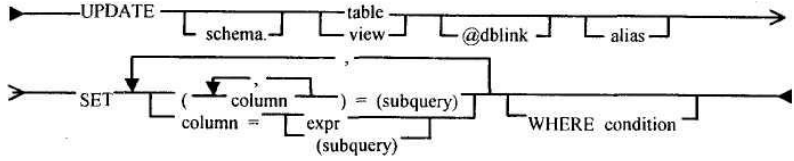
INSERT INTO grades VALUES (12583, 20, 10, 'Информатика', 4.5);

2. Приведённая ниже команда добавляет студентов выпускников, закончивших после исправительной сессии, в таблицу выпускников.

*INSERT INTO students_97 SELECT * FROM students WHERE sem=10;*

3.2.3. Команда актуализации данных в таблице и виде. Команда *UPDATE* изменяет существующие значения в столбцах таблицы или вида, которые были созданы только на основе одной таблицы. Когда строки в виде модифицируются, они, в действительности, модифицируют базовую таблицу вида. Чтобы выполнить команду, пользователь должен иметь привилегию *UPDATE* таблицы. Команда имеет следующую синтаксическую диаграмму:

UPDATE ::=



Описание ключевых слов и параметров:

- *schema* - определяет схему таблицы или вида, в которой данные будут актуализироваться. Если параметр пропущен, то подразумевается, что таблица или вид принадлежат пользователю, задавшему команду;
- *table* и *view* - задает имя таблицы или вида, в которой данные будут изменяться;
- *dblink* - задает полное или частичное имя связи с базой данных распределенной базы данных, в которой находится таблица или вид. Если этот параметр пропустить, то ORACLE подразумевает, что таблица находится в локальной базе данных;

- *alias* - задает альтернативное имя, псевдоимя таблицы или вида, которое можно использовать в других частях той же команды *UPDATE;*
- *column* - определяет имя столбца, значения которого будут изменяться. Столбцы, имена которых не заданы в *SET*, не меняются;
- *expr* - выражение, задающее новое значение, которое будет присвоено столбцу;
- *subquery* - это команда *SELECT,* возвращающая новые значения, которые будут присвоены столбцу.
- *WHERE* - определяет строки, в которых заданные в команде столбцы будут изменять свои значения. Если это предложение пропустить, все заданные столбцы всех строк меняют свое значение.

Команду *UPDATE* нельзя использовать для изменения значений вида, в определении которого есть оператор *DISTINCT,* предложение *GROUP BY* или групповые функции.

Если в команде используется подзапрос, то он должен возвращать точно одну строку значений и это должна быть корректируемая строка. Если подзапрос не вернет строку - столбцам присваивается значение *null.*

Примеры:

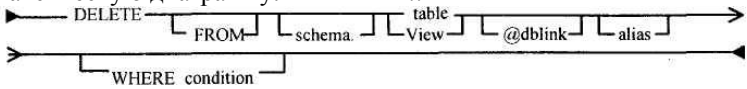
1. Следующая команда актуализирует значение семестра в таблице студентов после его окончания.

UPDATE students SETsem = sem + 1;

2. Следующая команда изменяет имена и специальность студента, которые введены ошибочно.

UPDATE students SET names='Иван Петров Иванов',
spec='Математика' WHERE deptno=21856;

3.2.4. Команда стирания строки таблицы и вида. Команда *DELETE* стирает строки из таблицы или вида, который основан только на одной таблице. Когда стираются строки в виде, то в действительности они стираются в базовой таблице вида. Чтобы выполнить команду, таблица должна быть из схемы пользователя или он должен иметь привилегию *DELETE ANY TABLE.* Команда имеет синтаксическую диаграмму: DELETE ::=



Описание ключевых слов и параметров:

- *schema* - определяет схему таблицы или вида, в которой стираются строки. Если параметр пропущен, то подразумевается таблица или вид пользователя, задавшего команду;

- *table* и *view* - задает имя таблицы или вида, где будут удаляться строки;
- *dblink* - задает полное или частично имя связи с распределенной БД, в которой находится таблица или вид. Если этот параметр пропущен, то ORACLE подразумевает, что таблица находится в локальной БД;
- *alias* - задает альтернативное имя, псевдоимя таблицы или вида, которое может быть использовано в других частях той же команды *DELETE;*
- *WHERE* - определят строки, которые будут стираться. Это строки, для которых условие принимает значение истина, и имеет значение TRUE. Если это предложение пропустить, то стираются все строки таблицы.

Команду *DELETE* нельзя использовать для стирания строк вида, в определении которого есть оператор *DISTINCT,* предложение *GROUP BY* или групповые функции.

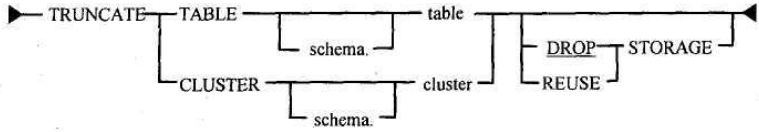
Пример:

Следующая команда из таблицы студентов стирает все строки исключенных выпускников и тех, которые имеют отметку "плохо" после экзаменационной сессии.

DELETE students WHERE sem=10 AND deptno IN
(SELECT DISTINCT deptno FROM grades WHERE grade=2);

3.2.5. Команда стирания всех строк таблицы. Команда *TRUNCATE* стирает безусловно все строки таблицы или кластера. Чтобы выполнить команду, таблица или кластер должны быть из схемы пользователя или он должен иметь привилегию *DELETE ANY TABLE.* Команда имеет следующую синтаксическую диаграмму:

TRUNCATE ::=



Описание ключевых слов и параметров:

- *TABLE* - задает стирание всех строк таблицы;
- *CLUSTER* - задает стирание всех строк кластера;
- *schema* - определяет схему таблицы или кластера, строки которых стираются. Если параметр пропущен, то подразумевается, что таблица или кластер принадлежат пользователю, задавшему команду;
- *table* или *cluster* - задает имя таблицы или кластера;

• *DROP STORAGE* - освобождает пространство удаленных строк таблицы или кластера. Это пространство может позже использоваться другими объектами табличного пространства;

• *REUSE STORAGE* - оставляет пространство стертых строк для той же самой таблицы или кластера. Это пространство позже может быть использовано только повыми данными, которые добавляются в таблицу или кластер. Если пропустить опции *REUSE STORAGE* и *DROP STORAGE,* ORACLE использует опцию *DROP STORAGE* по умолчанию.

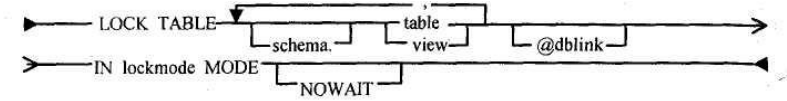
Для стирания всех строк таблицы можно использовать и команду *DELETE,* но ныполнение *TRUNCATE* отличается от выполнения *DELETE* по двум пунктам. Во-первых, командой *TRUNCATE* можно освободить пространство стертых срок для пользования другими объектами, в то время как команда *DELETE* не освобождает пространство удаленных строк. Во-вторых, команда *TRUNCATE* выполняется быстрее команды *DELETE,* потому что она не генерирует *rollback* информацию, и не выполняет связанные с таблицей триггеры.

Пример:

Следующая команда стирает все строки таблицы студентов и оставляет табличное пространство для ввода новых строк в таблицу. *TRUNCATE students REUSE STORAGE;*

3.2.6. Команда закрытия таблицы. Команда *LOCK TABLE* закрывает одну или больше таблиц определенным ею методом. Это закрытие не мешает автоматическому закрытию и разрешает или запрещает доступ к таблице или виду для других пользователей во время операций со стороны пользователя, выполнившего команду. Каждый пользователь может закрывать свои таблицы, а для закрытия других таблиц он должен иметь привилегию *LOCK ANY TABLE.* Команда имеет следующую синтаксическую диаграмму:

LOCK TABLE ::=



Описание ключевых слов и параметров:

• *schema* - определяет схему таблицы или вида, которая будет закрываться. Если пропустить параметр, то подразумевается, что таблица или вид принадлежат пользователю, задавшему команду;

• *table* и *view* - задает имя таблицы или вида, которые закрываются;

• *dblink* - задет полное или частичное имя связи с распределенной БД, в которой находится таблица или вид. Если этот параметр

пропустить, то ORACLE подразумевает, что таблица находится в локальной БД;

• *lockmode* - определяет один из следующих методов закрытия:

• *EXCLUSIVE* - разрешает запросы к закрытой таблице, запрещая, при этом, остальные операции с ней;

• *SHARE* - разрешает конкурирующие запросы, но запрещает изменения в закрытой таблице;

• *ROW SHARE* - разрешает конкурентный доступ к таблице и запрещает другим пользователям закрывать всю таблицу для исключительного доступа командой *LOCK TABLE* параметром *EXCLUSIVE;*

• *ROW EXCLUSIVE* - разрешает конкурентный доступ к таблице и запрещает другим пользователям закрывать всю таблицу для исключительного доступа командой *LOCK TABLE* с параметром *EXCLUSIVE* или *SHARE.* Это закрытие производится автоматически при выполнении команды *INSERT, UPDATE* или *DELETE;*

• *SHARE ROW EXCLUSIVE* - позволяет рассмотреть всю таблицу, но запрещает другим пользователям закрывать таблицу командой *LOCK TABLE* с параметром *SHARE* и изменять строки таблицы;

• *NOWAIT* - определяет, что ORACLE немедленно окончит выполнение команды не закрывая таблицу, если заданная таблица была уже закрыта другим пользователем. В этом случае ORACLE выводит сообщение, что таблица закрыта другим пользователем. Если этот параметр пропустить, то ORACLE ждет пока таблица станет доступной, запрашивает ее и передает управление пользователю.

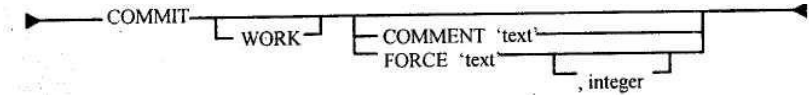
Пример: Следующая команда закрывает таблицу с оценками для последующего корректирования двоек.

LOCK TABLE grades IN SHARE NOWAIT;

3.3. Команды управления транзакциями

3.3.1. Команда окончания транзакции. Команда *COMMIT* заканчивает текущую транзакцию и делает постоянными все перемены сделанные в ней. Кроме того, эта команда стирает все точки сохранения *rollback* информации и освобождает все закрытия, которые были произведены в транзакции. Команда имеет следующую синтаксическую диаграмму.

COMMIT ::=



Описание ключевых слов и параметров:

• *WORK* - включается в команду только для схожести со стандартом SQL. Команды *COMMIT* и *COMMIT WORK* эквивалентны;

• *COMMENT* - задает комментарий, который должен быть связан с текущей транзакцией. Длина текста *'text'* должна быть меньше 50 знаков, который сохраняется в словаре базы данных, в виде *DBA_2PC_PENDING* как идентификатор транзакции, если она станет сомнительной из-за нарушения связи или машинного сбоя;

• *FORCE* - завершает данную сомнительно окончившуюся распределенную транзакцию. Транзакция идентифицируется *'text'*, который сохраняется в словаре базы данных. Можно использовать *integer*, чтобы присвоить транзакции специфический помер изменения системы (SCN - system change number). Если не задать *integer*, транзакция заканчивается, используя текущий SCN.

3.3.2. Команда определения точки сохранения. Команда *SAVEPOINT* идентифицирует точку в ходе выполнения транзакции, после которой, при необходимости, можно отменить сделанные действия. Она имеет следующий синтаксис:

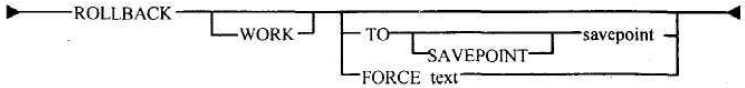
SAVEPOINT savepoint;

где *savepoint* - это имя точки сохранения создаваемой *rollback* информации.

Точки сохранения *rollback* информации используются во время выполнения команды *ROLLBACK.*

3.3.3. Команда отмены действий, которые произведены транзакцией. Команда *ROLLBACK* отменяет произведенные текущей транзакцией действия. Чтобы отменить одну сомнительно окончившуюся распределенную транзакцию, пользователь должен иметь привилегию *FORCE TRANSACTION,* а чтобы отменить сомнительно окончившую распределенную транзакцию другого пользователя, надо иметь привилегию *FORCE ANY TRANSACTION.*

Команда имеет следующую синтаксическую диаграмму: ROLLBACK ::=



Описание ключевых слов и параметров:

• *WORK*- включается в команду только для схожести со стандартом SQL;

• *TO savepoint* - отменяет текущую транзакцию до определенной точки. Если пропустить это предложение, транзакция отменяется вообще;

• *FORCE* - отменяет сомнительно окончившуюся распределенную транзакцию, которая идентифицируется *'text'*.

Когда команда выполнится без предложения *TO savepoint,* производятся следующие действия: транзакция оканчивается; не производятся изменения в базе данных; стираются точки сохранения *rollback* информации; освобождаются закрытие таблицы и виды.

Когда команда выполнится с предложением *TO savepoint,* производятся следующие действия: не производятся изменения в базе данных сделанные после точки сохранения, заданной *savepoint;* стираются точки сохранения *rollback* информации, которые созданы после заданной точки *savepoint;* освобождаются закрытие таблицы и виды командами выполненными после точки сохранения, заданной *savepoint.*

Пример:

Следующей транзакцией корректируются последовательно зарплаты преподавателей (профессора, доценты и ассистенты), создавая точки сохранения. После последней коррекции устанавливается, что сумма увеличения зарплаты доцента ошибочна и транзакция отменяется до первой точки сохранения. После правильной коррекции зарплат, транзакция оканчивается нормально.

UPDATE readers SETsal=sal+1500 WHERE scdegree='профессор'; SAVEPOINT sp1;

UPDATE readers SET sal=sal+1200 WHERE scdegree= 'доцент'; SAVEPOINT sp2;

UPDATE readers SET sal=sal+800 WHERE scdegree='асистент'; ROLLBACK TO sp1;

UPDATE readers SET sal=sal+1100 WHERE scdegree='доцент'; SAVEPOINT sp2;

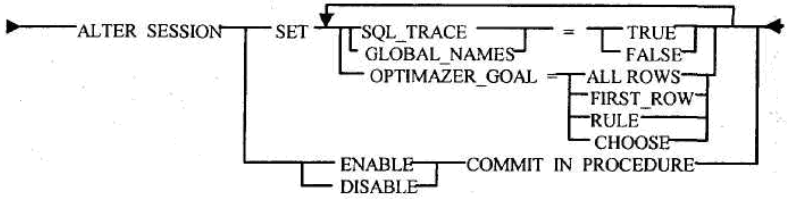
UPDATE readers SET sal=sal+800 WHERE scdegree='асистент'; COMMIT;

3.4. Команды управления ORACLE-сесии

3.4.1. Команда изменения опции текущей сессии. Команда *ALTER SESION* может быть использована для изменения следующих опций текущей сессии: разрешить или запретить запись статистики выполнения SQL-команд; определить подход оптимизации выполнения SQL-команд; разрешить или запретить управление транзакциями в процедурах и функциях.

Команда имеет следующую синтаксическую диаграмму:

ALTER SESSION ::=



Описание ключевых слов и параметров:

• *SQL TRACE* - разрешает (при TRUE), или запрещает (при FALSE), запись статистики выполнения SQL-команд;

• *OPTIMIZER GOAL* - определяет подход оптимизирования выполнения SQL команд в зависимости от используемого параметра, а именно:

* *RULE* - оптимизация производится на базе индексов и кластеров, связанных с обрабатываемой таблицей и синтаксисом конструкции использованной команды;

* *ALL ROWS* - оптимизация производится с учетом статистических данных, описывающих доступ к таблицам, индексам и кластерам во время выполнения SQL-команд, с тем, чтобы минимизировать время получения всех строк обработки;

* *FIRST ROW* - оптимизация производится с учетом статистических данных, описывающих доступ к таблицам, индексам и кластерам во время пополнения SQL-команд, с тем, чтобы минимизировать время получения первой строки обработки;

* CHOOSE - заставляет оптимизатор выбрать подход оптимизации базы статистических данных, записанных в словаре данных;

* (*COMMIT IN PROCEDURE* - разрешает, если используется *ENABLE,* или запрещает, если используется *DISABLE,* применение команд для управлении транзакциями *COMMIT* и *ROLLBACK* в процедурах и функциях.

Примеры:

1. Данная команда включает запись статистических данных ныполнения SQL-команд.

ALTER SESION SET SQLJRACE;

2. Следующая команда разрешает использование команды *COMMIT* в процедурах.

ALTER SESION ENABLE COMMIT IN PROCEDURE;.

ЗАКЛЮЧЕНИЕ.

Команды SQL можно использовать в следующих программных средствах ORACLE:

- *SQL*FROMS* - программа создания выходных форм, с помощью которых происходит манипулирование данными в базе данных: вводятся строки, изменяются значения в столбцах введенных строк и стираются строки;
- *SQL *REPORTWR1TER* - программа вывода отчетов из базы данных.

Команды SQL могут быть использованы также в расширении языка PL/SQL и в программах написанных на языках Pascal, C, Fortran и Cobol. С точки зрения характера работы, которая производится над базой данных, субъекты работающие в ORACLE могут быть разделены на три группы: *администраторы* базы данных, *программисты* - разработчики приложений и *пользователи.* Нельзя четко разграничить команды SQL, которые используются различными группами, так как каждый может работать с теми командами, для которых имеет привилегии.

Здесь мы рассмотрели команды, которые наиболее часто используются программистами и пользователями и не рассмотрели команды, которые наиболее характерны для работы администратора.

Литература

1. European Directory of Management Consultants. 1995. London: FEACO-AP Information services, 1995.

2. Guidelines for the Use of Consultants by World Bank Borrowers and by the World Bank as Executing Agency. Washington (D.C.): The World Bank, 1992.

3. Hurley N. Management Consultancy Manual: Operating a Successful Management Consultancy Assignment. Ankara: SMIDO, 1990.

4. Kubr M. How to select and use consultants: A client's guide. Geneva: ILO, 1993.

5. Maister D. Professional Service Firm Management. Boston: Maister Associates. Inc., 1990.

6. Кудинов А. О рынке консалтинговых услуг. // www.bcg.ru.

7. Монахова Е. Управленческое консультирование конца XX века. // www.pcweek.ru/kis.

8. Посадский А.П., Хайниш С.В. Консультационные услуги в России. - М.: Финстатинформ, 1995, - 171 с.

9. Уткин Э.А. Консалтинг. – М.: ЭКМОС, 1998, - 256 с.

10. Интернет-сервер Гарвардской школы бизнеса: www.hbs.edu

11. BS7799: Информационная безопасность начинается с менеджмента. // Банковские технологии. №8, 1998, - С. 73-75.

12. Международный стандарт ИСО 9000. Системы менеджмента качества. Основные положения и словарь. 2-е изд. 2000-12-15. ISO - 2000.

13 Международный стандарт ИСО 9001. Системы менеджмента качества. Требования. 3-е изд. 2000-12-15. ISO – 2000.

14. Международный стандарт ИСО 9004. Системы менеджмента качества. Руководство по улучшению деятельности. 2-е изд. ISO – 2000.

15. ISO 9000 Introduction and Support Package: Guidelines on the Process Approach to quality management systems. ISO/TC 176/SC 2/N 544R. 17 May, 2001.

16. ISO 9000 Introduction and Support Package: Guidance on the Documentation Requirements of ISO 9001:2000. ISO/TC 176/SC 2/N 544R. 13 March, 2001.

17. Давид Марка, Клемент МакГоуэн. Методология структурного анализа и проектирования. Пер . с англ . М .:1993, 240 с ., ISBN 5-7395-0007-9

18. INTEGRATION DEFINITION FOR FUNCTION MODELING (IDEF0). Draft Federal Information Processing Standards Publication 183, 1993, December 2. www.idef.com

19. Р50.1.028-2001. Методология функционального моделирования. М.: Госстандарт России, 2000. www.cals.ru

20. Менеджмент качества и международные стандарты ИСО 9000 версии 2000 г. Материалы семинара в рамках Программы ИСО для развивающихся стран. Минск, Июль 2001 г. 79 с.

21. Framework for Managing Process Improvement. Vol.1. Electronic College of Process Innovation. DoD USA. May, 1994.

22. С. Бобровски - ORACLE 7 и вычисления клиент - сервер, Москва, "ЛОРИ", 1996.

23. М. Ричарде и др. - Oracle 7.3, Энциклопедия пользователя, Киев, "DiaSoft", 1997.

24. ORACLE7 SERVER - CONCEPT MANUAL.

25. ORACLE7 SERVER - SQL LANGUAGE REFERENCE MANUAL -Part Number 778-70-1292, December 1992.

26. С. Бемер, Г.Фратер - MS Access...для пользователя, Киев, "BHV", 1994.

27. Р. Дженнингс - Access 95 в подлинике, 2 тома, СПб, "BHV", 1997.

28. Р. Дженнингс - Microsoft Access 97 в подлинике, 2 тома, Санкт-Петербург, "BHV", 1997.

29. Керри Н. Праг, Мишель Р. Ирвин - Access 97, Библия пользователя, Киев-Москва, "Диалектика", 1997.

30. А. Горев, С. Макашаринов - Microsoft Visual FoxPro 3.0, Новые возможности для программиста, Санкт-Петербург, "Питер Пресс", 1995.

31. Б. Сосински - Разработка приложений в среде Visual FoxPro 5, Киев-Москва, "Диалектика", 1997.

32. Цыпкин Я. З. Основы теории автоматических систем. — М.: Наука. Гл. ред. физ.-мат. лит., 1977. — 560 с.

33. Лукашин Ю. 77. Адаптивные методы краткосрочного прогнозирования. — М.: Статистика, 1979. — 254 с.

34. Марпл-мл. С. Л. Цифровой спектральный анализ и его приложения: Пер. с англ. — М.: Мир, 1990. — 584 с.

35. Дрейпер Н., Смит Г. Прикладной регрессионный анализ: В 2-х книгах: Пер. с англ. —М.: Финансы и статистика, 1986. —Кн. 1. — 366 с.

36. Дрейпер Н., Смит Г. Прикладной регрессионный анализ: В 2-х книгах: Пер. с англ. — М.: Финансы и статистика, 1987. — Кн. 2. — 351 с.

37. Бриллинджер Д. Временные ряды. Обработка данных и теория: Пер. с англ. — М.: Мир, 1980. — 536 с.

38. Ахметшин А. М., Киргизов И. А. Прогнозирование кратковременных экономических рядов как задача адаптивной экстраполяции функций с финитным спектром // Пращ М1жнар. конф. з индуктивного моделювання «МКІМ — 2002». — Львщ 2002. — С. 29—33.

39. *Клих Ю. А., Плотникова Л. И.* Спектральный анализ функций. — К.:УМКВО, 1992.— 108 с.

40. *Kasabov N. K.* Foundations of neural networks, fuzzy systems, and knowledge engineering. — Cambridge, Mass.: MIT Press, 1996. — 550 p.

41. *Беллман Р.* Введение в теорию матриц: Пер. с англ. — М.: Наука. Гл. ред. физ.-мат. лит., 1976. — 352 с.

Научно-практическое издание

Кононюк Анатолий Ефимович

Консалтинтология
Общая теория консалтинга

Книга 3

Авторская редакция

Подписано в печать 21.10.2010 г.
Формат 60х84/16.
Усл. печ. л. 16,5. Тираж 300 экз.

Издатель и изготовитель:
Издательство «Освита Украины»
04214, г. Киев, ул. Героев Днепра, 63, к. 40
Свидетельство о внесении в Государственный реестр
издателей ДК №1957 от 23.04.2009 г.
Тел./факс (044) 411-4397; 237-5992
E-mail: osvita2005@ukr.net, www.rambook.ru