А.И.МИШЕНИН

Теория экономических информационных систем

Издание четвертое,
дополненное и переработанное

Рекомендовано
Министерством образования
Российской Федерации
в качестве учебника
для студентов высших учебных заведений,
обучающихся по специальности
"Математические методы
и исследование операций в экономике"

Москва
"Финансы и статистика"
2002

УДК 002.53+681.3:33(075.8)
ББК 65с.я.73
М71

РЕЦЕНЗЕНТЫ:

кафедра экономической информатики
Московского государственного университета коммерции;
доктор экономических наук, профессор ***А.П. Иванов***

Мишенин А. И.

М71 Теория экономических информационных систем: Учебник. – 4-е изд., доп. и перераб. – М.: Финансы и статистика, 2002. – 240 с.: ил.

ISBN 5-279-01987-9.

Дается характеристика компонентов экономических информационных систем (ЭИС) – вычислительной системы, базы данных, программного обеспечения; рассматриваются этапы их жизненного цикла – проектирование, внедрение, эксплуатация, развитие. Моделирование представлений информации в ЭИС предполагает использование синтаксических моделей данных (реляционной, сетевой и иерархической) и семантических моделей (семантические сети, фреймы и др.). Моделирование процессов опирается на сети Петри. Теоретические методы проектирования иллюстрируются практическими задачами. Для иллюстрации методов обработки данных используются языки Паскаль, SQL, dBASE и Пролог (3-е изд. – 1993 г.).

Для студентов, обучающихся по специальностям «Математические методы и исследование операций в экономике», а также «Прикладная информатика».

М $\frac{0605010204 - 175}{010(01) - 2002}$ 241 – 2001

УДК 002.53 + 681.3 : 33 (075.8)
ББК 65с.я.73

ISBN 5-279-01987-9

ПРЕДИСЛОВИЕ

Развитие экономики и других сфер человеческой деятельности в наше время связано с применением вычислительной техники, созданием информационных систем различного назначения.

В книге представлена система понятий, описывающих экономическую информационную систему (ЭИС) в целом и ее компоненты. Рассматриваются основные проблемы, модели и методы создания и сопровождения ЭИС.

В дисциплине «Теория экономических информационных систем» (ТЭИС) основное внимание уделяется проблемам организации информации при решении задач на ЭВМ и технологии обработки информации.

Организация материала в учебнике связана с выделением компонентов ЭИС (база данных, метаинформация, программное обеспечение, вычислительная система), этапов их жизненного цикла (проектирование, внедрение, эксплуатация, модификация), уровней представления информации (внешний, концептуальный, внутренний).

Для единиц информации вводится ряд свойств, таких, как имя, значение, структура, операции над названными свойствами, ограничения и методы представления значений. Детальное изучение этих свойств производится в главах 2, 3 и 4.

Моделирование внешних представлений экономической информации опирается на анализ экономических показателей. Моделирование представлений информации в ЭИС на концептуальном уровне предполагает использование синтаксических

моделей данных (реляционной, сетевой, иерархической) и семантических моделей (семантические сети, фреймы и др.).

Модели данных анализируются с учетом применяемых информационных конструкций, операций и ограничений. Для применения синтаксических моделей данных характерно использование единого аппарата функциональных зависимостей. Он позволяет производить нормализацию реляционных баз данных, создавать корректные сетевые и иерархические базы данных. Исследуются также ациклические базы данных.

Семантические модели данных представлены как средство исследования предметной области и аппарат представления знаний о предметной области и самой ЭИС. База знаний трактуется как развитие базы данных, позволяющее получать сведения, явно не хранящиеся в ЭИС. Проанализированы возможности моделей фиксации и манипулирования знаниями.

При рассмотрении внутреннего уровня представления информации основное внимание уделяется анализу структур данных и алгоритмов, особенностям их реализации на современных ЭВМ.

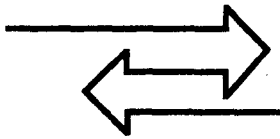
Исследование вычислительных процессов в ЭИС опирается на выделение параметров, характеризующих все структурные компоненты системы, рассмотрение формальных характеристик процессов, анализ вычислительной нагрузки ЭВМ при реализации процессов.

Для иллюстрации методов обработки данных в книге используются языки Паскаль, SQL, dBASE и Пролог.

Успешному усвоению материала учебника поможет разработанная автором обучающая система ВЕГА по дисциплине «Теория экономических информационных систем».

Материал книги соответствует учебной программе по дисциплине «Теория экономических информационных систем». Книга предназначена для специалистов, занятых разработкой и сопровождением ЭИС, а также студентов, обучающихся по специальностям «Прикладная информатика» и «Математические методы и исследование операций в экономике».

Параграф 4.3 написан совместно с Н. В. Андреевой.



Глава 1

ОСНОВНЫЕ ПОНЯТИЯ ЭКОНОМИЧЕСКИХ ИНФОРМАЦИОННЫХ СИСТЕМ

1.1 ИНФОРМАЦИОННАЯ СИСТЕМА В ОБЩЕМ ВИДЕ

Сегодня обработка экономической информации стала самостоятельным научно-техническим направлением с большим разнообразием идей и методов. Отдельные компоненты процесса обработки данных достигли высокой степени организации и взаимосвязи, что позволяет объединить все средства обработки информации на конкретном экономическом объекте понятием "экономическая информационная система" (ЭИС). Детальное изучение ЭИС опирается на понятия "информация" и "система", к которым мы и переходим.

Информация и система, возможно, являются простейшими фундаментальными категориями, не выражаемыми через более общие понятия. Поэтому приводимые далее определения всего лишь поясняют и уточняют эти категории.

Понятие информации

Существующие определения понятия "информация" после тщательного анализа обычно признаются неудовлетворительными. Чаще всего эти определения рассматривают информацию в сравнительно узком контексте. Попытки дать более

широкое определение вносят элемент неясности. Поэтому вряд ли возможно сформулировать одно точное определение этого понятия.

Довольно таки распространенным является взгляд на информацию как на ресурс, аналогичный материальным, трудовым и денежным ресурсам. Эта точка зрения отражается в следующем определении.

Информация – новые сведения, позволяющие улучшить процессы, связанные с преобразованием вещества, энергии и самой информации.

Информация не отделима от процесса информирования, поэтому необходимо рассматривать источник информации и потребителей информации. Роль потребителей информации очерчивается в таком определении.

Информация – новые сведения, принятые, понятые и оцененные конечным потребителем как полезные.

Информацией являются сведения, расширяющие запас знаний конечного потребителя.

Выделяются три фазы существования информации.

1. Ассимилированная информация – представление сообщений в сознании человека, наложенное на систему его понятий и оценок.

2. Документированная информация – сведения, зафиксированные в знаковой форме на каком-то физическом носителе.

3. Передаваемая информация – сведения, рассматриваемые в момент передачи информации от источника к приемнику.

В дальнейшем будем рассматривать только документированную или передаваемую информацию. Подавляющая масса информации собирается, передается и обрабатывается с помощью знаков. *Знаки* – это сигналы, которые могут передавать информацию при наличии соглашения об их смысловом содержании между источниками и приемниками информации. Набор знаков, для которых существует указанное соглашение, называется *знаковой системой*. Многие знаковые системы, естественно, нельзя четко ограничить, однако при обработке информации на электронных вычислительных машинах (ЭВМ) наличие точного перечня знаков обязательно.

Правильное восприятие информации конечным потребителем может быть затруднено из-за наличия различных помех, называемых информационным шумом.

Различаются три разновидности шума и соответственно три информационных фильтра, блокирующих этот шум.

1. Синтаксический фильтр. В последовательности знаков, хранимых на носителе или передаваемых, могут быть обнаружены участки, относительно которых отсутствует соглашение о придании им смысла. Эти участки составляют синтаксический шум, и они распознаются синтаксическим фильтром. Фильтр содержит набор решающих правил, позволяющих различать правильные (осмысленные) и неправильные (бессмысленные) последовательности знаков.

2. Семантический фильтр. Первый аспект семантического шума связан с отсутствием новизны в получаемом сообщении. Иначе говоря, сообщение не расширяет знаний потребителя. Второй аспект семантического шума связан с прохождением ложного сообщения через синтаксический фильтр. Допустим, что сообщение "Запас материала с кодом 141672 равен 956 тонн" дважды искажено так, что вместо цифры 7 воспринято 4 и вместо 9 воспринято 3. Первое искажение может быть зарегистрировано синтаксическим фильтром, только если материал с кодом 141642 вообще не должен храниться на предприятии, а второе искажение синтаксический фильтр не заметит. Такие искажения должен обнаруживать семантический фильтр. Он проверяет соответствие контролируемого сообщения с уже имеющейся информацией. Если на предприятии в нашем примере установлено, что запас любого материала должен превышать его месячную потребность, а для материала 141672 она составляет 720 тонн, то после исправления первой ошибки семантический фильтр обнаружит вторую ошибку. Существенные для семантического фильтра взаимосвязи устанавливаются также предметными науками, например бухгалтерским учетом, экономической статистикой и др.

3. Прагматический фильтр, говоря в общем, устанавливает степень ценности информации для потребителя. Элементы прагматической оценки обычно охватывают полноту информации

(исчерпывающее отражение явления), ее своевременность, компактность (возможна меньшая длина сообщения), употребимость (число потенциальных потребителей) и доступность.

Информация на пути от источника к потребителю проходит через ряд преобразователей – кодирующих и декодирующих устройств, вычислительную машину, обрабатывающую информацию по определенному алгоритму, и т. д. На промежуточных стадиях преобразования семантические и прагматические свойства сообщений отступают на второй план ввиду отдаленности потребителя, поэтому понятие информации заменяется на менее ограничительное понятие "данные".

Данные представляют собой набор утверждений, фактов и/или цифр, лексически и синтаксически взаимосвязанных между собой.

Лексические отношения (часто называемые парадигматическими) отражают постоянные связи в структуре языка, например, род – вид, целое – часть. Связи между отдельными частями сообщения отражаются *синтаксическими (синтагматическими) отношениями*. Они являются переменными, например, положение запятой в фразе "Казнить нельзя помиловать" определяет тот или иной ее смысл. Данные безразличны к семантическому и прагматическому фильтру. В тех случаях, когда различие между информацией и данными нет нужды подчеркивать, они употребляются как синонимы.

Чтобы определить понятие «экономическая информация», надо очертить рамки экономических процессов. В наиболее общей форме экономическими процессами являются производство, распределение, обмен и потребление материальных благ. Информация об указанных процессах называется *экономической информацией*.

Для обработки экономической информации характерны сравнительно простые алгоритмы, преобладание логических операций (упорядочение, выборка, корректировка) над арифметическими, табличная форма представления исходных и результатных данных.

К важнейшим признакам, по которым обычно осуществляется классификация циркулирующей экономической информации, относятся:

1. Отношение к данной управляющей системе. Этот признак позволяет разделить сообщения на входные, внутренние и выходные.

2. Признак времени. Относительно времени сообщения делятся на перспективные (о будущих событиях) и ретроспективные. К первому классу относится плановая и прогнозная информация, ко второму – учетные данные. По времени поступления разделяются периодические и непериодические сообщения.

3. Функциональные признаки. Формируется классификация по функциональным подсистемам экономического объекта. Например, информация о трудовых ресурсах, производственных процессах, финансах и т. п., в другом разрезе – на данные планирования, нормирования, контроля, учета и отчетности.

Надо констатировать, что не существует меры информации, равно применимой на всех стадиях обработки информации. Остается единственная возможность – учитывать количество обрабатываемых знаков, т. е. объем информации. Эта величина отражает, естественно, только внешнюю сторону информационных процессов.

Понятие системы

Системой называется любой объект, который, с одной стороны, рассматривается как единое целое, а с другой – как множество связанных между собой или взаимодействующих составных частей.

Понятие системы охватывает комплекс взаимосвязанных элементов, действующих как единое целое. В систему входят следующие компоненты.

1. Структура – множество элементов системы и взаимосвязей между ними. Математической моделью структуры является граф.

2. Входы и выходы – материальные потоки или потоки сообщений, поступающие в систему или выводимые ею. Каждый входной поток характеризуется набором параметров {x(i)};

значения этих параметров по всем входным потокам образуют вектор-функцию X. В простейшем случае X зависит только от времени t, а в практически важных случаях значение X в момент времени t+1 зависит от X(t) и t. Функция выхода системы Y определяется аналогично.

3. Закон поведения системы – функция, связывающая изменения входа и выхода системы Y = F(X).

4. Цель и ограничения. Качество функционирования системы описывается рядом переменных u1, u2,..., uN. Часть этих переменных (обычно всего одна переменная) должна поддерживаться в экстремальном значении, например, max u1. Функция u1 = f(X,Y,t,...) называется *целевой функцией,* или целью. Зачастую f не имеет аналитического и вообще явного выражения. На остальные переменные могут быть наложены (в общем случае двусторонние) ограничения

$$aK <= gK(uK) <= bK, \text{ где } 2 <= K <= N.$$

Среди известных свойств систем целесообразно рассмотреть следующие – относительность, делимость и целостность.

Свойство относительности устанавливает, что состав элементов, взаимосвязей, входов, выходов, целей и ограничений зависит от целей исследователя. Реальный мир богаче системы. Поэтому от исследователя и его целей зависит, какие стороны реального мира и с какой полнотой будет охватывать система. При выделении системы некоторые элементы, взаимосвязи, входы и выходы не включаются в нее из-за слабого влияния на остающиеся элементы, из-за наличия самостоятельных целей, плохо согласующихся с целью всей системы, и т.д. Они образуют внешнюю среду для рассматриваемой системы.

Делимость означает, что систему можно представить состоящей из относительно самостоятельных частей – подсистем, каждая из которых может рассматриваться как система. Возможность выделения подсистем (декомпозиция системы) упрощает ее анализ, так как число взаимосвязей между подсистемами и внутри подсистем обычно меньше, чем число свя-

зей непосредственно между всеми элементами системы. Выделение подсистем проводит исследователь, и оно условно.

Свойство целостности указывает на согласованность цели функционирования всей системы с целями функционирования ее подсистем и элементов.

Надо также иметь в виду, что система, как правило, имеет больше свойств, чем составляющие ее элементы. Так, предприятие обладает юридической самостоятельностью, а его подразделения – нет.

Экономическая информационная система представляет собой систему, функционирование которой во времени заключается в сборе, хранении, обработке и распространении информации о деятельности какого-то экономического объекта реального мира. Информационная система создается для конкретного экономического объекта и должна в определенной мере копировать взаимосвязи элементов объекта.

Экономические информационные системы предназначены для решения задач обработки данных, автоматизации конторских работ, выполнения поиска информации и отдельных задач, основанных на методах искусственного интеллекта.

Задачи обработки данных обеспечивают обычно рутинную обработку и хранение экономической информации с целью выдачи (регулярной или по запросам) сводной информации, которая может потребоваться для управления экономическим объектом.

Автоматизация конторских работ предполагает наличие в ЭИС системы ведения картотек, системы обработки текстовой информации, системы машинной графики, системы электронной почты и связи.

Поисковые задачи имеют свою специфику, и информационный поиск представляет собой интегральную задачу, которая рассматривается независимо от экономики или иных сфер использования найденной информации.

Алгоритмы искусственного интеллекта необходимы для задач принятия управленческих решений, основанных на моделировании действий специалистов предприятия при принятии решений.

Для ЭИС соблюдаются следующие принципы их построения и функционирования.

1. Соответствие. ЭИС должна обеспечивать функционирование объекта с заданной эффективностью. Критерий эффективности должен быть количественным.

2. Экономичность. Затраты на обработку информации в ЭИС должны быть меньше экономического выигрыша на объекте при использовании этой информации.

3. Регламентность. Большая часть информации в ЭИС поступает и обрабатывается по расписанию, со строгой периодичностью.

4. Самоконтроль. Непрерывная работа ЭИС по обнаружению и исправлению ошибок в данных и процессах их обработки.

5. Интегральность. Однократный ввод информации в ЭИС и ее многократное, многоцелевое использование.

6. Адаптивность. Способность ЭИС изменять свою структуру и закон поведения для достижения оптимального результата при изменяющихся внешних условиях.

Среди других особенностей ЭИС следует назвать обработку больших объемов информации по сравнительно простым алгоритмам, высокий удельный вес логической обработки данных (сортировка, группировка, поиск, корректировка) и представление подавляющей части информации в виде документов.

При создании информационной системы возникает задача объективной оценки качества ее функционирования. Такая оценка особенно актуальна потому, что современные информационные системы – это сложные и дорогостоящие проекты, на их создание расходуются значительные ресурсы.

Эффективность работы информационной системы выражается при помощи набора числовых характеристик, называемых *критериями эффективности*. Каждый критерий количественно определяет степень соответствия между результатами проектирования или функционирования ЭИС и поставленными перед ней целями.

Величина, выбранная в качестве критерия, должна удовлетворять ряду требований:

- должна прямо зависеть от процесса проектирования (функционирования) системы,
- давать наглядное представление об одной из целей системы,
- иметь сравнительно простой алгоритм расчёта,
- допускать приближенную оценку по экспериментальным данным.

ЭИС обычно оценивается по комплексу критериев. Оценке подлежат:

- система в целом,
- отдельные составляющие этапа проектирования системы, например проекты информационного, программного и технического обеспечения,
- важнейшие компоненты этапа эксплуатации системы, например подготовка информации, ее обработка, ведение информационных массивов.

Как правило, функционирование ЭИС направлено на успешную реализацию нескольких целей.

Типичный перечень может включать следующие цели.

1. Повышение эффективности управления объектом (цели С1 – С3):

С1 – максимальную полноту информации для обеспечения принимаемых решений;

С2 – представление информации с максимально возможной скоростью;

С3 – максимальное удобство взаимодействия информационной системы с потребителями.

2. Эффективное использование ресурсов ЭИС (цели С4 – С6):

С4 – сокращение расходов на создание, эксплуатацию и развитие ЭИС;

С5 – максимальное извлечение выходной информации из имеющегося объема данных;

С6 – сокращение избыточности в базе данных.

Критериями для названных целей будут:

К1 – отношение объема информации в базе данных к объему информации на объекте управления;

К2 – время обработки информации в ЭИС;

К3 – время, которое потребители расходуют на запрос необходимой информации и ее использование в управлении;

К4 – сумма капитальных вложений и текущих затрат на создание, эксплуатацию и развитие ЭИС;

К5 – отношение объемов входной и выходной информации;

К6 – доля избыточной информации в общем объеме данных.

Одновременное достижение указанных целей практически неосуществимо. Например, повышение эффективности ЭИС по критериям К1 и К3 вызывает увеличение К4 и достигнутое состояние системы противоречит С4.

Классификация ЭИС

ЭИС классифицируются по многим аспектам. Эта классификация является расширяющейся. Мы рассмотрим классификацию по функциональному признаку, по режимам работы ЭИС и по способу распределения вычислительных ресурсов. Другие аспекты классификации здесь рассматриваться не будут.

Среди ЭИС выделяются управляющие информационные системы (для управления технологическими процессами на предприятии) и системы административно-организационного типа для обслуживания коллектива специалистов, осуществляющих управление предприятием.

ЭИС, рассматриваемые в этой книге, следует отнести к административно-организационным системам.

С функциональной точки зрения можно выделить такие классы ЭИС, как системы обработки данных (СОД), автоматизированные системы управления (АСУ) и информационно-поисковые системы (ИПС). Структурные схемы названных систем показаны на рис. 1.1 и рис. 1.2. Многие реальные ЭИС обладают чертами нескольких из названных классов, а не какого-то одного.

Основными функциями ЭИС являются сбор, передача, хранение информации и такие операции обработки, как ввод, выборка, корректировка и выдача информации. Для операций

Рис. 1.1. Структурные схемы системы обработки данных и автоматизированной системы управления

преобразования входной информации в выходную, которые не обеспечиваются названными выше функциями, необходимо создание прикладных программ.

ЭИС, дополненная прикладными программами различного назначения, образует *систему обработки данных* – СОД. Для прикладных программ СОД характерно наличие математических соотношений, которые позволяют вычислять значения элементов выходной информации по известным значениям входной информации без применения методов оптимизации процессов управления экономическим объектом. В качестве примера можно назвать программы расчета заработной платы сотрудников предприятия, формирования статистической отчетности и т.п.

СОД производит информационное обслуживание специалистов органа управления объектом, принимающих управленческие решения. Решение, принятое на основе представленной информации, передается на управляемый объект, минуя СОД.

Можно трактовать СОД как систему, которая преобразует поток входной информации в поток выходной информации. Фактически используются следующие потоки информации:

- входная информация, которая поступает от управляемого объекта и из внешнего мира (от других предприятий и организаций),

- необрабатываемая информация, т. е. часть входной информации, которая непосредственно передается органу управления, минуя обработку,
- нормативно-справочная информация,
- выходная информация, т.е. информация, обработанная системой и представляемая органу управления и внешнему миру,
- промежуточная информация, т.е. часть выходной информации, которая необходима СОД для выполнения расчетов в последующие периоды времени.

Если СОД способна выполнять выбор управленческих решений (автономно или с участием специалистов), то она становится *автоматизированной системой управления* – АСУ. Принятие решений системой может производиться на основе экономико-математических методов либо путем моделирования действий специалиста по принятию управленческого решения. Прикладные программы АСУ, формирующие управленческое решение, как правило, используют экономико-математические методы для выбора оптимальных решений. Исходные данные для оптимизационной задачи (например, себестоимость продукции для расчета оптимальной производственной программы) рассчитываются в режиме системы обработки данных. Моделирование принятия решений специалистом реализуется в так называемых экспертных системах, которые построены на принципах искусственного интеллекта и баз знаний.

Типичными для АСУ являются задачи оптимального управления запасами материалов и полуфабрикатов на складах предприятия. АСУ прогнозирует поступление материалов и их расход на основное производство, а в случае несоблюдения норм запаса материалов формирует заявки предприятиям-поставщикам.

Информационно-поисковые системы (ИПС) предназначены для отыскания в каком-то множестве документов тех, которые посвящены указанной в информационном запросе теме или содержат необходимые сведения. Схема функционирования ИПС приведена на рис.1.2 . При вводе в ИПС каждый документ подвергается индексированию.

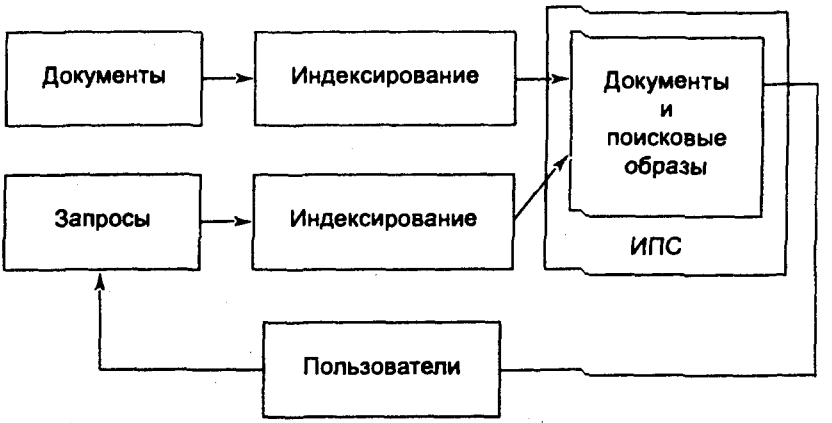


Рис. 1.2. Схема функционирования информационно-поисковой системы

Под *индексированием* понимается процесс, который состоит из двух этапов:

- определения тем, которые отражаются в данном документе,
- выражения этих тем на языке, принятом в информационно-поисковой системе, и записи в виде поисковых образов, которые связываются с документом.

Для того чтобы при помощи ИПС можно было отыскать документы, соответствующие некоторому информационному запросу, сам запрос также должен быть заиндексирован. Процесс поиска осуществляется путем сопоставления поисковых образов документов с поисковым образом запроса. При полном или частичном совпадении образов документ считается соответствующим запросу и выдается пользователю.

В ЭИС могут применяться два режима решения задач – пакетный и диалоговый.

При *пакетном режиме* обработки данные в системе накапливаются до тех пор, пока не наступит заданный момент времени, или объем данных не превысит некоторый предел. Затем имеющаяся информация обрабатывается несколькими последовательно запускаемыми программами. В качестве примера системы, работающей в пакетном режиме, можно назвать систему сбора и группировки статистической отчетности предприятий.

Пакетный режим обработки данных характеризуется рядом недостатков. Так, потребитель информации обособлен от процесса ее обработки, что в некоторых ситуациях вызывает у него сомнения в правильности исходных данных и результатов в применяемых алгоритмах. Пакетная система снижает также оперативность принятия решений на управляемом объекте.

При *диалоговом режиме* работы происходит обмен сообщениями между пользователем и системой. Роль «активного» элемента пользователь и система выполняют попеременно. ЭИС активна от момента завершения ввода информации и команд пользователем до завершения обработки команды (запроса). Пользователь обдумывает результат обработки запроса и вводит данные для следующего запроса. Следует отметить, что последовательность команд, выдаваемых пользователем в диалоговом режиме работы, не является фиксированной заранее, а существенно зависит от результатов ранее выполненных команд.

Необходимость диалога с системой возникает при решении экономических задач с многовариантной логикой, когда пользователь имеет возможность определять наиболее перспективные варианты решения задачи и постоянно следить за осмысленностью вычислений, производимых системой. Типичной диалоговой задачей можно считать расчеты по распределению ресурсов между несколькими потребителями.

По способу распределения вычислительных ресурсов выделяются локальные и распределенные ЭИС. *Локальная система* использует одну электронно-вычислительную машину, а в распределенной системе организуется взаимодействие нескольких ЭВМ, соединенных между собой каналами связи.

Распределенная информационная система представляет собой объединение информационных систем, выполняющих собственные, не зависимые друг от друга функции, с целью коллективного использования информационных фондов и вычислительных ресурсов этих систем. Отдельные информационные системы, как правило, территориально удалены друг от друга.

Каждый из процессов в распределенной информационной системе может независимо обрабатывать локальные данные и принимать соответствующие решения. Отдельные процессы обмениваются информацией через сеть связи с целью обработки данных, расположенных в других узлах сети для собственных или коллективных нужд.

Наиболее распространенными схемами связи в распределенных ЭИС являются системы с центральной ЭВМ и системы с кольцевой структурой связей ЭВМ.

В заключение можно отметить, что вопросы использования экономической информации, методы управления экономическими объектами и методы принятия управленческих решений не рассматриваются в теории экономических информационных систем. Предполагается, что к моменту создания ЭИС указанные вопросы так или иначе решены. В теории ЭИС изучаются проблемы организации информации в системе и эффективной эксплуатации ЭИС.

1.2 КОМПОНЕНТЫ ЭКОНОМИЧЕСКИХ ИНФОРМАЦИОННЫХ СИСТЕМ

При решении любых задач с использованием ЭВМ требуется наличие ряда компонентов:

- исходной и справочной информации для расчета,
- метода (алгоритма) решения задачи, записанного в виде программы, которая может быть выполнена на ЭВМ,
- самой ЭВМ как исполнителя алгоритмов,
- пользователей, т.е. лиц, которые используют результаты решения задачи в своей профессиональной деятельности.

Для функционирования ЭИС необходимы компоненты, аналогичные названным выше, но с более сложной организацией.

Компоненты информационной системы – это база данных, концептуальная схема и информационный процессор, образующие вместе систему хранения и манипулирования данными.

В окружающем нас мире выделяются материальные системы различного назначения. Все, что происходит в процессе функционирования материальных систем, может быть описано в форме сообщений. Появление сообщений о событиях, происходящих в материальной системе, представляет собой информационное отображение материальных процессов.

Так, выпуск продукции рабочими порождает сообщения о том, кто из рабочих изготовил определенные изделия, когда и на каком оборудовании, в каком количестве и т. д.

Сообщение может быть выражено на естественном языке, однако часто применяют форматированные сообщения, когда выделяются опорные свойства (параметры) происходящего события и в сообщении приводятся названия свойств и их значения.

Пример сообщения

На склад #2 1.02.98 поступили генераторы от завода «Динамо» в количестве 50 шт. по цене 200 руб.

ФОРМАТИРОВАННЫЙ ВАРИАНТ ЭТОГО СООБЩЕНИЯ:

Название параметра	**Значение параметра**
Получатель	Склад № 2
Отправитель	Завод "Динамо"
Изделие	Генератор
Дата	01.02.98
Цена	200 руб.
Количество	50 шт.

Таких сообщений о поступлении изделий на склады предприятия появляется достаточно много. Они совпадают по названиям параметров и различаются по значениям параметров. В этом случае удобно представление в виде таблицы, показанной на с. 21.

Многие сообщения легко разделяются на компоненты и представляются в форматированном виде. Форматированные сообщения – это наиболее массовый вид сообщений, храни-

T					
Получатель	Отправитель	Изделие	Дата	Цена	Количество
Склад № 2	з-д "Динамо"	Генератор	01.02.98	200	50
Склад № 8	з-д "Каучук"	Шина	09.02.98	80	100
...	...	...	...	...	...
Склад № 4	з-д АТЭ-2	Реле	23.05.98	10	160

мых и обрабатываемых в ЭИС. Вместе с тем существует экономическая информация, которую практически невозможно форматировать, например приказы по предприятию.

База данных (БД) – это набор сообщений, которые

- являются истинными для соответствующей материальной системы,
- непротиворечивы по отношению друг к другу и к концептуальной схеме.

Сообщения в БД обычно являются форматированными и хранятся в виде единиц информации.

Единицей информации называется набор символов, которому придается определенный смысл. Это понятие в основном относится к базе данных, хранящей форматированные сообщения.

> Если в сообщении «На склад № 2 01.02.98 поступили генераторы от завода «Динамо» в количестве 50 шт. по цене 200 руб.» названия параметров фиксированы, то набор символов «склад № 2, з-д «Динамо», генератор, 01.02.98, 200 руб., 50 шт.» является единицей информации. «01.02.98» также является единицей информации.

Минимально необходимы две единицы информации – атрибут и составная единица информации (СЕИ).

Атрибутом называется информационное отображение отдельного свойства некоторого объекта, процесса или явления.

Любое сообщение записывается в форматированном виде как указание свойств (параметров) предметов, о которых мы

говорим. Поэтому информационное отображение любого явления представляет собой набор соответствующим образом подобранных атрибутов.

Составная единица информации представляет собой набор из атрибутов и, возможно, других СЕИ.

Простейшими СЕИ являются таблицы, подобные приведенной выше. СЕИ позволяет создавать произвольные комбинации из атрибутов.

База данных ЭИС хранится в запоминающих устройствах вычислительной системы (ЭВМ). Хранимые представления данных очень часто не соответствуют первоначальному множеству форматированных сообщений. Однако сейчас при рассмотрении БД будем считать, что сообщения хранятся в виде таблиц.

Концептуальная схема (от слова concept – понятие) представляет собой описание структуры всех единиц информации, хранящихся в БД. Под *структурой* понимается вхождение одних единиц информации в состав других единиц информации.

В рамках нашего примера можно говорить о двух единицах информации – параметре (атрибуте) и таблице (СЕИ).

Предположим, что таблица Т соответствует всей базе данных. В концептуальной схеме должно быть указано, что БД состоит из Т, а Т содержит параметры Получатель, Отправитель, Изделие, Дата, Цена, Количество. Более содержательное представление о концептуальной схеме приводится в гл. 2 книги.

Информационный процессор – это механизм, который в ответ на получение команды выполняет операции с БД и концептуальной схемой. Информационный процессор состоит из вычислительной системы и системы управления базой данных – СУБД.

Под *вычислительной системой* будем понимать серийно выпускаемую электронно-вычислительную машину (ЭВМ) либо несколько ЭВМ, соединенных каналами связи в вычислительную сеть.

ЭВМ состоит из ряда устройств, каждое из которых способно выполнять свойственные ему операции.

Оперативное запоминающее устройство (ОЗУ) используется для хранения программ и данных. Скорость чтения и записи для ОЗУ, как правило, одинаковы, ОЗУ является наиболее быстродействующим устройством ЭВМ.

Процессор выполняет команды, находящиеся в программе. Выполнение команды обычно предполагает обращение к ОЗУ.

ЭВМ может содержать любое число внешних устройств, чаще всего это внешние запоминающие устройства и устройства печати информации.

База данных предполагает централизованное управление данными, что обеспечивает ряд преимуществ:

- сокращение избыточности хранимых данных благодаря однократному хранению каждого сообщения в базе данных,
- совместное использование хранимых данных всеми пользователями ЭИС,
- стандартизацию представления данных, упрощающую проблемы эксплуатации БД и обмена данными между ЭИС,
- обеспечение процедур проверки достоверности информации и процедур ограничения доступа к данным,
- совмещение требований к использованию БД со стороны различных пользователей ЭИС.

Системой управления базой данных называется комплекс программ, обеспечивающий централизованное хранение, накопление, модификацию и выдачу данных, входящих в БД.

Предполагается, что в управлении базой данных принимает участие специальное должностное лицо – администратор базы данных.

Предметная область

Любая экономическая система представляет собой совокупность связанных ресурсов и процессов. К ресурсам относятся, например, рабочие и служащие, сырье и материалы, станки, деньги, изделия и полуфабрикаты. Процесс – это преобразование одного набора ресурсов в другой набор ресур-

сов. Одновременно могут происходить многие процессы. Так, процесс производства изделий использует входные ресурсы – рабочую силу, материалы и оборудование, а на выходе процесса получаются готовые изделия или полуфабрикаты. Завершение процесса производства позволяет выполнить другие процессы, например передачу продукции на склад.

Взаимосвязанные ресурсы и процессы экономической системы можно описать в терминах предметной области.

Предметной областью называются элементы материальной системы, информация о которых хранится и обрабатывается в ЭИС.

Информационным отображением всей предметной области экономического объекта служит информационная база ЭИС. *Информационная база* состоит из одной или нескольких баз данных, определенных выше. Далее будем всюду рассматривать ЭИС с одной базой данных.

При рассмотрении объектов предметной области и их информационного отображения в БД сложился единый, не зависимый от СУБД понятийный аппарат. Для описания предметной области необходимы такие термины, как объект, свойство объекта, взаимодействие (связь) объектов, свойство взаимодействия.

Объектом называется любой элемент некоторой системы.

В экономических приложениях понятие объекта сужается до понятия физического объекта, под которым понимается любой предмет, занимающий место в пространстве. Следует различать отдельный физический объект (отдельный предмет) и объект – понятие, который охватывает множество физических объектов. *Отдельный предмет* часто называется *экземпляром объекта*, а *различные множества предметов*, образованные по заданному принципу, называются *типами объектов*. *Первоначальная группировка* экземпляров в некоторые множества-классы называется *классификацией*. Полученные классы объектов соответствуют приведенному выше определению типа. Типы объектов могут объединяться для формирования новых типов по принципу «множество, элементами которого являются другие множества».

Объекты экономической сферы группируются в три крупных типа, имеющих название *средств производства*, *предметов труда* и *исполнителей*.

Свойством объекта называется некоторая величина, которая характеризует состояние объекта в любой момент времени. Отдельный экземпляр объекта можно точно описать, если указать достаточное количество значений его свойств. Два экземпляра объектов являются различными, если они отличаются по значению хотя бы одного свойства.

Существенные упрощения в описании объектов связаны с установлением аналогий в структуре объектов, образующих класс. Объекты одного класса описываются одноименными свойствами. Объекты, входящие в некоторый тип, содержат ряд свойств, характерных для типа в целом. Этот принцип называется *наследованием свойств*. Так, все экземпляры объектов, образующих тип «основные фонды», характеризуются свойством *балансовая стоимость*, которое отсутствует у других типов, например у типа «исполнители».

Деятельность, которая развернута во времени, охватывается понятием *взаимодействие объектов*. Взаимодействием объектов называется факт участия нескольких объектов в каком-либо процессе, который протекает и во времени, и в пространстве.

Свойством взаимодействия называется такое свойство, которое характеризует совместное поведение объектов, но не относится ни к одному объекту в отдельности. Например, при производстве изделий взаимодействуют объекты Рабочий, Материал, Оборудование, Изделие. Количество изделий, произведенных за определенный день, является свойством взаимодействия, но никак не характеризует указанные выше объекты, взятые в отдельности.

Проблема полноты отображения объектов и процессов предметной области в хранимые данные решается в ЭИС следующим образом. Предполагается, что представление объекта или процесса сводится к указанию его свойств; информационным отображением свойств служат атрибуты и, следовательно, экземпляр объекта или экземпляр процесса представлен в базе данных как набор пар <Имя атрибута>,<Значение атрибута>, где имена

атрибутов различны и соответствуют названиям свойств объекта или процесса. Вопрос о выражении сущности объектов с помощью того или иного набора свойств решается путем расширения набора свойств, описывающих объект, чем достигается более полное представление о его сущности. Количество свойств должно быть таково, чтобы всегда можно было отличить объект одного класса от объекта другого класса, а также любые два объекта из одного и того же класса. Более глубокие представления философского порядка о соотношении сущности и явления, содержания и формы при анализе ЭИС обычно не привлекаются.

Среди свойств, описывающих объект, необходимо выделить *идентифицирующие свойства*, т.е. свойства, по значению которых можно однозначно отличить данный экземпляр объекта от любого другого (в том числе и в пределах класса объектов, содержащего этот экземпляр).

В ряде случаев установление идентифицирующего свойства не является простой задачей.

Рассмотрим, например, объект Личность. Простейшими идентифицирующими свойствами личности обычно считаются Фамилия, Имя, Отчество. Однако наличие однофамильцев с совпадающими именами и отчествами показывает, что этих трех свойств недостаточно для идентификации. Можно пойти по пути расширения списка идентифицирующих свойств, добавляя свойства Дата рождения, Национальность и т.д., пока не будет обеспечена однозначная идентификация требуемого множества людей, или предложить новое идентифицирующее свойство (возможно, вводимое искусственно). В нашем примере можно использовать два свойства – Номер паспорта и Серия паспорта, однако паспортная система охватывает не всех жителей страны (за рубежом в качестве идентификатора личности часто используется номер социального страхования). Если множество людей ограничено рамками некоторого предприятия (учреждения), то идентифицирующим свойством может служить Табельный номер.

Искусственный идентификатор, как правило, соответствует обычной нумерации экземпляров объектов, например Инвентарный номер.

Пренебрежение вопросами идентификации объектов является серьезной проектной ошибкой. Отсутствие идентифицирующих свойств объектов помешает расширению перечня решаемых задач, затруднит совместимость с другими ЭИС, сделает невозможными некоторые виды информационного поиска.

Детализация ЭИС

В качестве предметной области можно изучать не только материальные системы, но и саму ЭИС. Выделяемые в ЭИС объекты, свойства и взаимодействия служат понятийной основой для моделей создания и функционирования ИС. Такие компоненты ИС, как база данных и программное обеспечение, не являются физическими объектами, поэтому информационное отображение ИС осуществляется в метаинформацию. Метаинформацию следует представлять как информацию об информации.

Классификация компонентов ЭИС приводится на рис. 1.3 и в табл. 1.1.

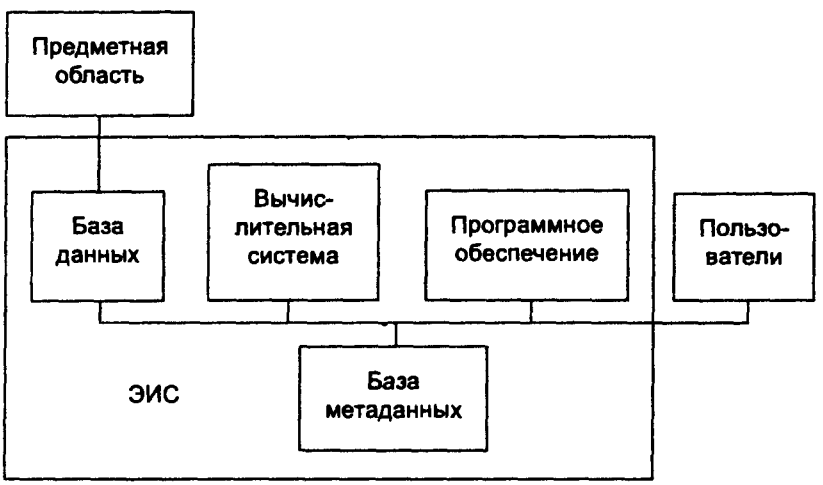


Рис. 1.3. Компоненты экономической информационной системы

Таблица 1.1. **Компоненты информационной системы**

Данные/ метаданные	Система/ процесс	Вычислительная система	Внешняя Среда
Атрибут Отношение Показатель База данных Ключ Схема Подсхема	Подсистема Задача Программа Задание Транзакция	Физическое устройство Терминал Линия(канал) Узел сети ЭВМ	Пользователь Администратор БД

Понятия, которые описывают данные и метаданные, рассматриваются в последующих разделах.

Элементарным процессом при пакетной обработке данных является *задание*, при диалоговой обработке – *транзакция* (взаимодействие). Задание содержит одну или несколько программ, выполняемых в определенной последовательности. Транзакция обычно представляет собой одну команду информационного процессора.

Задачу можно рассматривать с точки зрения ее экономического содержания и метода решения на ЭВМ. Определение содержательной стороны задачи связано с декомпозицией функций управления экономическим объектом. В этом контексте экономическая задача является элементарным процессом, реализующим некоторую функцию управления в конкретном подразделении системы управления. С точки зрения решения на ЭВМ задача представляет собой определенную последовательность программ, реализующих формирование фиксированного потока выходной информации.

Группировка задач в подсистемы соответствует принятой классификации основных функций управления экономическим объектом. Так, для промышленного предприятия крупными подсистемами обычно являются:

- управление сбытом и реализацией продукции,
- технико-экономическое планирование,

- управление материально-техническим снабжением,
- бухгалтерский учет,
- оперативное управление производством,
- управление технической подготовкой производства.

Пользователей экономической информационной системы можно подразделить на пять типов:

- случайные пользователи, взаимодействие которых с ЭИС не обусловлено их служебными обязанностями,
- параметрические пользователи, которые работают с ЭИС повседневно, в соответствии с четко определенной областью деятельности, по регламентированным процедурам,
- аналитики и исследователи, информационные потребности которых непредсказуемы (в отличие от параметрических пользователей),
- прикладные программисты, которые разрабатывают программы для реализации запросов к базе данных. Эти программы используются в основном параметрическими пользователями,
- системные программисты, которые разрабатывают служебные программы, расширяющие возможности операционной системы ЭВМ и СУБД, например программы разграничения доступа к данным, проверки достоверности данных, восстановления базы данных после сбоя в работе ЭВМ, программы печати документов и т.п.

Администратор базы данных – это специалист или группа специалистов, занятых обслуживанием пользователей базы данных. Администратор должен координировать процессы сбора информации, проектирования и эксплуатации базы данных, обеспечения защиты и целостности данных. Администратор обязан учитывать текущие и перспективные информационные потребности пользователей.

Описание хранимой и обрабатываемой информации в ЭИС делается с разной степенью детализации. Используются три уровня представления (рис.1.4):

1. Внешний уровень – описание информационных потребностей конечного пользователя.

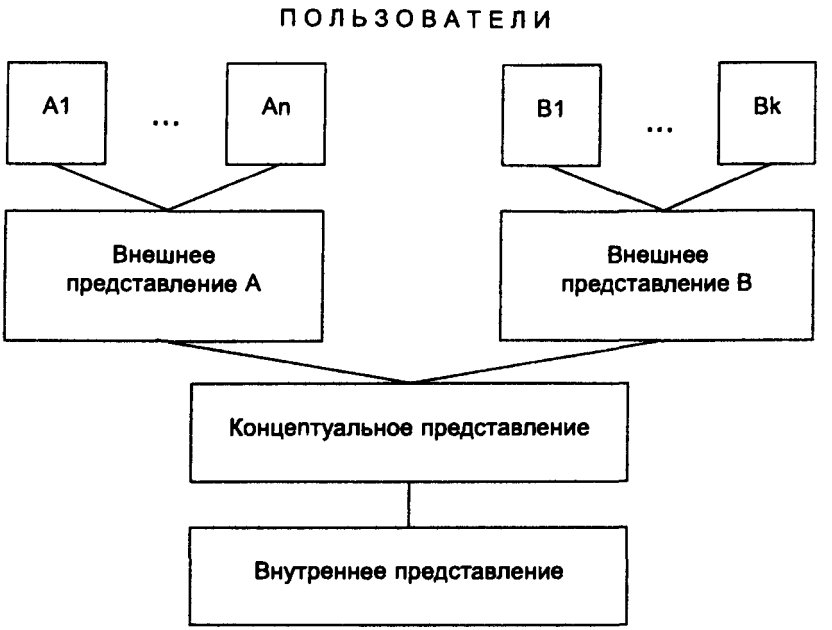


Рис. 1.4. Детализация представлений ЭИС

2. Концептуальный уровень – описание информационных потребностей на уровне понятий ЭИС.

3. Внутренний уровень – описание способа хранения информации в памяти ЭВМ и методов доступа к ней.

Внутренний уровень наиболее близок к физической памяти ЭВМ, внешний уровень наиболее близок к пользователям, а концептуальный уровень занимает промежуточное положение.

Информационные потребности отдельного пользователя относятся лишь к некоторой части базы данных, и описание этих потребностей может не совпадать с хранимыми в ЭИС представлениями данных.

Внешнее представление может пользоваться любым аппаратом понятий. Единственное требование состоит в возможности преобразования его в концептуальное представление. Цель концептуального уровня – создать такое формальное

представление о базе данных, чтобы любое внешнее представление являлось его подмножеством. В процессе интеграции внешних представлений устраняются двусмысленности и противоречия в информационных потребностях различных пользователей. Допускается много внешних описаний, каждое из которых отображается частью базы данных, и единственное концептуальное описание, представляющее всю БД.

Внешнее представление оказывается достаточным для применения ряда прикладных программ, которые можно охарактеризовать как генераторы отчетов. Генерация отчетов предполагает преобразование потока входной информации в выходной поток. Само преобразование включает группировку информации, подведение итогов и т. п. Результат оформляется в виде отчетов, удобных для использования специалистами. Необходимые для генератора отчетов описания структуры входной информации и отчетов, а также вычислений при формировании отчета легко могут быть выполнены конечными пользователями (специалистами предприятия или организации).

Концептуальное представление описывает полное информационное содержание базы данных в более абстрактной форме по сравнению со способом физического хранения данных. Оно может полностью отличаться от описания информационных потребностей отдельных пользователей, в частности использовать другую систему понятий, обозначений и правил описания. В концептуальном описании необходимы не только сведения о структуре обрабатываемой информации, но и сведения о технологии ее обработки – применяемые методы контроля информации, описание использования потоков информации в подразделениях предприятия, описание ограничений на доступ к информации и ряд других.

Концептуальный уровень описания оказывается достаточным для использования программной поддержки в виде систем управления базами данных. Концептуальное описание при этом необходимо адаптировать к требованиям конкретной СУБД. После этого появляется возможность использования

всех средств обработки данных, предоставляемых этой СУБД, значительно упрощаются вопросы разработки программного обеспечения системы, сокращаются сроки разработки ЭИС.

К концептуальному представлению предъявляется требование устойчивости. Это означает, что ряд изменений в предметной области не должен приводить к обязательной корректировке концептуального представления. Концептуальное представление должно быть достаточно абстрактным, т. е. не содержать ограничений, вытекающих из программной реализации требуемых методов обработки данных.

Как известно, в естественном языке различаются правила написания текстов (синтаксис языка) и сами тексты (книги, статьи и т.д.). В обработке данных *правила описания данных* содержатся в моделях данных, а *описание информации* для конкретной ЭИС называется *представлением*, *схемой* или *структурой*.

Принципиальными различиями обладают три модели данных – реляционная, сетевая и иерархическая, у которых разные множества допустимых информационных конструкций.

Существующие СУБД обеспечивают реализацию возможностей этих моделей данных с теми или иными ограничениями и уточнениями, что дает повод говорить о наличии самостоятельной модели данных у каждой СУБД. Однако при создании СУБД происходит модификация модели данных исходя из удобства программной реализации системы.

Организация данных в ЭИС рассматривается с позиций той или иной модели данных, и обычно за пределами рассмотрения остаются методы представления звуковых сигналов, изображений и т. п.

Внутреннее описание данных определяет организацию данных в памяти ЭВМ и методы доступа к данным. Это наиболее детальное описание процессов обработки данных в ЭИС. Если ЭИС разработана с применением СУБД, то требуемые параметры внутреннего описания довольно немногочисленны.

В ряде случаев применение СУБД не позволяет реализовать все требования к ЭИС (например, высокое быстродей-

ствие программ). Тогда для поддержки внутреннего уровня описания системы требуется разработка уникальных программ доступа к данным.

Если структура хранимой базы данных меняется, то должны обеспечиваться все требования концептуального описания системы, существовавшие до начала изменений.

Многоуровневая детализация представлений ЭИС обусловлена разницей между способом описания предметной области и теми спецификациями, которые могут быть эффективно обработаны современными СУБД и пакетами прикладных программ. Наличие нескольких уровней детализации позволяет расчленить процесс проектирования ЭИС на несколько более простых шагов, а также предоставить возможность участия в разработке ЭИС пользователям и специалистам, не имеющим профессиональной подготовки в области обработки данных.

1.3
КЛАССИФИКАЦИЯ И ОСНОВНЫЕ СВОЙСТВА ЕДИНИЦ ИНФОРМАЦИИ

Существуют две основные единицы информации – атрибут и составная единица информации. Определение атрибута было дано в п. 1.2. *Атрибут* соответствует понятию переменной в языках программирования и понятию реквизита в бухгалтерском учете.

Атрибут характеризуется именем и значением. Именем атрибута называется его условное обозначение в процессах обработки данных.

Значением атрибута называется величина, характеризующая некоторое свойство объекта, явления, процесса в конкретных обстоятельствах. Все *допустимые значения атрибута* образуют множество, называемое *доменом этого атрибута*.

Формально атрибут с именем X представляет собой пару (X,z), где z – элемент Z. Множество Z называется *доменом зна-*

чений (областью определения атрибута X), величина z является значением атрибута X в заданный момент времени.

Определение домена предполагает указание его имени и списка значений. Если число значений в домене невелико, то их список можно указать при объявлении данных в программе. Например, в языке программирования Паскаль это выглядит как

```
type
day = 1..31;
month = (январь, февраль, март, апрель, май, июнь, июль,
август, сентябрь, октябрь, ноябрь, декабрь);
year = 1900..1999;
{в описании типа атрибутов перечисляются допустимые
значения атрибутов День, Месяц и Год соответственно}
```

Зачастую невозможно перечислить все элементы домена, поэтому для домена указываются тип и длина значения. Наиболее употребительны текстовые (символьные), числовые, логические значения, а также значения дат и другие специальные типы значений.

Пример

Домен фамилий – FAM. Перечислить фамилии невозможно, поэтому ограничим FAM значениями текстового типа длиной до 20 символов. Для языка Паскаль получим:

```
var FAM: string[20];
```

В домене, определенном таким образом, могут оказаться элементы, заведомо не являющиеся фамилией, например 'MMMM', но такие случаи при определении домена не учитываются.

Для ряда доменов множество входящих в них значений задается с помощью перечисления допустимых значений. Если в домене необходимо перечислить обозначения объектов из некоторого класса, то разрабатывается классификатор, содержащий условные обозначения (коды) отдельных объектов и классов, к которым эти объекты отнесены.

Рассмотрим простейшие системы классификации и кодирования, применяемые для обозначения объектов в базе данных вместо их полных названий.

В первую очередь, если классификация объектов вообще не требуется, производится их нумерация, и кодом каждого объекта служит его порядковый номер. Такая система кодирования называется *порядковой*.

Если все множество объектов классифицируется по одному признаку, то коды объектов целесообразно разделить на несколько частей (серий) по количеству значений этого признака и в пределах каждой серии использовать последовательные номера.

Когда используется несколько классификационных признаков и их взаимная подчиненность соответствует выделению классов объектов, подклассов внутри каждого класса и т.д., удобно использовать разрядную систему кодирования.

В качестве примера рассмотрим различные системы кодирования значений атрибута Код студента. Порядковый код студента – это просто его номер в списке всех студентов. Предположим, что необходимо различать студентов-дневников, вечерников и заочников с использованием серийной системы кодирования. Для этого последовательные номера от 1 до 5999 будем использовать при кодировании дневников, номера от 6000 до 7999 – при кодировании вечерников, от 8000 до 9999 – при кодировании заочников. Если в этих же условиях применить разрядный код, то первый знак кода будет принимать три значения (1 – дневное отделение, 2 – вечернее, 3 – заочное), а следующие 4 знака отводятся для нумерации студентов каждого отделения. В разрядном коде можно учесть больше признаков, например, первый знак – код отделения, второй – код факультета, третий – код курса, четвертый – код группы, пятый и шестой – порядковый номер студента в группе. Обратите внимание, что, увеличивая число различимых признаков в коде, мы вынуждены увеличивать и длину значения атрибута Код студента.

Разрядная система кодирования применяется для кодирования объектов, определяемых несколькими соподчиненными признаками. Кодируемые объекты систематизируются по классификационным признакам на каждой ступени классификации. Каждому признаку классификации отводится определенное число разрядов, в пределах которого кодирование начинается с единицы. Классификационные группировки по младшим признакам кодируются в зависимости от кода более старшего признака.

Если значения нескольких атрибутов определены на одном и том же домене, то такие атрибуты называются ролевыми.

На домене FAM могут быть определены атрибуты с именами: Студент, Преподаватель, Автор. Все это ролевые атрибуты.

Атрибуты Фамилия рабочего и Табельный номер рабочего неролевые, хотя описывают одних и тех же людей.

Домен значений, как правило, не хранится в базе данных как самостоятельный информационный объект. Однако среди ролевых атрибутов домена в базе данных, безусловно, существует атрибут с наиболее полным перечнем значений, и этот атрибут необходимо использовать для контроля достоверности вновь вводимой информации. Например, на предприятии наиболее полный список сотрудников должен присутствовать в базе данных отдела кадров.

Составной единицей информации (СЕИ) называется набор из атрибутов и, возможно, других СЕИ. Определение СЕИ построено рекурсивно (т. е. в определении понятия участвует само понятие), но противоречия здесь нет, поскольку «другие СЕИ» когда-нибудь будут состоять только из атрибутов (ввиду конечности сообщений).

Атрибут и отношение образуют минимально возможный набор единиц информации. На практике удобно использовать большее число единиц информации, как это показано, например, в табл. 1.2. Следует отметить, что БД в целом также является единицей информации. Если рассматривать единицы

информации как информационные объекты, то можно говорить об их свойствах, как это делается в табл. 1.2. В то же время единицы информации – это нефизические объекты, так как они не занимают место в пространстве.

Свойства единиц информации представлены в табл.1.2.

Таблица 1.2. **Свойства единиц информации**

<table>
<tr><td rowspan="3">Название свойства</td><td rowspan="3">Атрибут</td><td colspan="4">Составная единица информации</td></tr>
<tr><td colspan="2">Представления пользователя</td><td colspan="2">Представления проектировщика</td></tr>
<tr><td>Документ</td><td>Показатель</td><td>Отношение</td><td>Вещное отношение</td></tr>
<tr><td>Имя
Значение
Структура</td><td>+
+
-</td><td>+
+
+</td><td>+
+
+</td><td>+
+
+</td><td>+
+
+</td></tr>
<tr><td>Операции над именем</td><td colspan="5">Переименование, объявление синонима</td></tr>
<tr><td rowspan="2">над значением</td><td rowspan="2">Перекодирование</td><td colspan="4">Выборка, корректировка</td></tr>
<tr><td></td><td>Арифметические операции</td><td colspan="2"></td></tr>
<tr><td>над структурой</td><td>–</td><td>Декомпозиция, композиция, нормализация, свертка</td><td>–</td><td>Проекция, соединение, добавление атрибутов</td><td>Добавление/ изъятие атрибутов</td></tr>
<tr><td>Ограничения</td><td>Принадлежность домену</td><td colspan="4">Функциональные зависимости</td></tr>
<tr><td>Методы организации значений</td><td>–</td><td colspan="2">Последовательный</td><td colspan="2">Последовательный, индексный, прямой, цепной, древовидный</td></tr>
</table>

Множество атрибутов объединяется в одну СЕИ по следующим принципам:

- соответствующие атрибуты описывают один и тот же факт или экономический процесс,

- значения атрибутов, входящих в СЕИ, возникают одновременно, связаны логическими или арифметическими соотношениями.

Простейшими характеристиками СЕИ являются имя, структура и значение. *Имя* СЕИ – это ее условное обозначение в процессах обработки информации. *Структурой* СЕИ называется вхождение одних единиц информации в состав других единиц информации.

Аппарат СЕИ рассчитан на описание структуры экономических документов. *Документом* называется материальный носитель информации (обычно бланк бумаги), содержащий оформленные в установленном порядке сообщения и имеющий юридическую силу.

Существует сравнительно много способов описания структуры СЕИ. Для описания, не зависимого от конкретных языков программирования и СУБД, достаточно указывать после имени СЕИ список имен входящих в нее атрибутов и СЕИ. Будем помещать этот список в круглые скобки, а имена внутри скобок перечислять через запятую. Имя СЕИ может сопровождаться размерностью, т.е. указанием на количество одинаковых по структуре значений этой СЕИ. Размерность, если она не равна 1, указывается в скобках после имени СЕИ.

Рассмотрим в качестве примера документа «Приходный ордер» с сокращенным составом атрибутов (рис.1.5). СЕИ приходного ордера, названная Прих, содержит атрибуты Дата (дата поступления материалов), Пост (код поставщика материалов), Склад и таблицу, также включающую ряд атрибутов. Эта таблица является «другой» СЕИ в составе СЕИ Прих и названа Табл. Обратите внимание, что в экономических документах таблицы не имеют названий, названия всех элементов документа требуются при его машинной обработке.

В СЕИ Табл содержатся атрибуты Ннм (номенклатурный номер материала), Кво-док (количество материала, принятое по товарно-транспортной накладной), Кво-пр (количество материала, принятого на склад), Цена (цена материала), Сумма (результат перемножения значений Кво-пр и Цена). Размерность 3 у СЕИ

Табл соответствует трем строкам в таблице приходного ордера, а размерность 2 у СЕИ Прих определяется наличием двух документов в нашем примере.

Окончательно структура СЕИ приходного ордера имеет вид:

Прих(2).(Дата,Пост,Склад,Табл(3).(Ннм,Кво-док,Кво-пр, Цена,Сумма))

Прих

Дата	Пост	Склад
01.10.95	1728	02

Ннм	Кво-док	Кво-пр	Цена	Сумма
26 114	16	16	5,00	80,00
49 712	10	8	6,00	48,00

Прих

Дата	Пост	Склад
07.10.95	3476	02

Ннм	Кво-док	Кво-пр	Цена	Сумма
49 712	12	10	6,00	60,00
72 426	8	8	8,00	64,00
26 114	5	5	5,00	25,00

Рис. 1.5. Бланки документа «Приходный ордер»

Определение значения СЕИ можно дать, опираясь на значения единиц информации, входящих в структуру СЕИ. Значение атрибута, входящего в СЕИ, определяется непосредственно. Значение СЕИ, входящей в другую СЕИ, можно определить рекурсивно, однако надо учесть размерность СЕИ. Кроме того, для множества значений СЕИ в составе другой СЕИ необходимо самостоятельное понятие. Назовем собранием СЕИ множество ее значений в составе СЕИ более высокого уровня. Количество значений в собрании СЕИ равно ее размерности.

Значением СЕИ называется набор значений непосредственно входящих в нее атрибутов и набор собраний непосредственно входящих в нее СЕИ. Одно значение СЕИ приходного ордера содержит по одному значению атрибутов Дата, Пост, Склад и собрание СЕИ Табл. Собрание Табл включает в себя три значения, в каждое значение Табл входит по одному значению атрибутов Ннм, Кво-док, Кво-пр, Цена и Сумма. Всего в нашем примере определены два значения СЕИ Прих.

Одно значение СЕИ при хранении ее в памяти ЭВМ часто называется *записью*.

Все языки программирования содержат средства описания структуры СЕИ.

Пример

Рассмотрим аппарат описания языка Паскаль, в котором структура СЕИ соответствует понятию «тип записи».

Определение типа записи начинается зарезервированным словом record (запись), за ним следует список разделов записи. В конце списка ставится зарезервированное слово end (конец). Каждый раздел записи определяет тип одного или более атрибутов. Так как компоненты записи могут быть любого типа, то допускаются конструкции СЕИ с произвольной структурой:

```
Date = record
Day: [1..31];
Month: (Jan,Feb,Mar,Apr,May,Jun,July,Aug,Sep,Oct,Nov,Dec);
Year: 1900..1999;
end;
type rec = record
tn : 1..1000; {Табельный номер}
fio : string[30]; {Фио рабочего}
dr : Date; {Дата рождения}
nc : 1..9; {Номер цеха}
end;
Var zap: rec;
```

Переименованием единицы информации называется присвоение ей нового имени, объявление синонима – это установление второго, третьего и т.д. равноценного имени для единицы информации.

Операция над значением атрибута всего одна – это перекодирование, т.е. замена существующего кода значения на новый для всех значений.

Выборка – операция выделения подмножества значений СЕИ, которые удовлетворяют заранее поставленным условиям выборки.

Корректировка означает выполнение одной из операций:

- добавление нового значения СЕИ,
- исключение существующего значения СЕИ,
- замена некоторого значения СЕИ на новое значение.

Декомпозиция – операция преобразования исходной СЕИ в несколько СЕИ с различными структурами. Декомпозиция приходного ордера может привести, например, к двум СЕИ

Цены(Ннм,Цена)
Приход(Дата,Пост,Склад,Ннм,Кво-док,Кво-пр,Сумма)

Декомпозиция, как и все операции над структурой СЕИ, одновременно производит преобразование множества значений, в частности нельзя однозначно определить размерность СЕИ Цены и Приход.

Композиция – операция преобразования нескольких СЕИ с различными структурами в одну СЕИ. Декомпозиция и композиция являются взаимообратными операциями, в частности, композиция Цены и Приход дает Прих.

Нормализация – это операция перехода от СЕИ с произвольной структурой к СЕИ с двухуровневой структурой. Одновременно происходит перекомпоновка значений СЕИ.

Нормализация приходного ордера приводит к следующему результату:

Дата	Пост	Склад	Ннм	Кво-док	Кво-пр	Цена	Сумма
01.10.95	1728	02	26114	16	16	5.00	80.00
01.10.95	1728	02	49712	10	8	6.00	48.00
07.10.95	3476	02	49712	12	10	6.00	60.00
07.10.95	3476	02	72426	8	8	8.00	64.00
07.10.95	3476	02	26114	5	5	5.00	25.00

Свертка – операция преобразования СЕИ с двухуровневой структурой в СЕИ с произвольной многоуровневой структурой. Свертка нормализованного приходного ордера может быть произведена в исходную структуру, а также в другие ненормализованные документы, имеющие экономический смысл, например карточку складского учета (рис.1.6).

Карт

Склад	Ннм	Цена
02	26 114	5,00

Пост	Дата	Кво-док	Кво-пр	Сумма
1 728	01.10.95	16	16	80,00
3 476	07.10.95	5	5	25,00

Карт

Склад	Ннм	Цена
02	49 712	6,00

Пост	Дата	Кво-док	Кво-пр	Сумма
1 728	01.10.95	10	8	48,00
3 476	07.10.95	12	10	60,00

Карт

Склад	Ннм	Цена
02	72 426	8,00

Пост	Дата	Кво-док	Кво-пр	Сумма
3 476	07.10.95	8	8	64,00

Рис. 1.6. Карточка складского учета

При анализе экономических документов ставится задача разделения документа на элементарные осмысленные фрагменты, называемые *показателями*. Это позволяет установить смысловые взаимосвязи между различными документами, обеспечить одинаковое понимание всеми пользователями применяемых единиц информации и их единое обозначение, использовать полученные результаты для определения структуры базы данных.

Показатель представляет собой полное описание количественного параметра, характеризующего некоторый объект или процесс. Соответствующее описание произвольного свойства (необязательно количественного) называется атомарным фактом и рассматривается в п. 4.1.

Чтобы точнее характеризовать атрибуты, образующие показатель, необходимо отметить существенные различия свойств, которые отображаются атрибутами. Материальные процессы, как известно, имеют качественную характеристику и количественную характеристику. Соответственно и атрибуты должны разделяться на два класса, которые называются «атрибуты-признаки» и «атрибуты-основания».

Атрибут-признак представляет собой информационное отображение качественного свойства некоторого объекта, предмета, процесса, а *основание* является отображением их количественного свойства.

В состав показателя должны входить один атрибут-основание и несколько атрибутов-признаков, однозначно характеризующих условия существования основания.

Как единица информации показатель является разновидностью СЕИ. Схематично структура показателя П представляется выражением

П(P1,P2,...,Pk,Q),

где P1,P2,...,Pk – атрибуты-признаки, Q – атрибут-основание.

Если представить себе показатель с двумя, например, атрибутами-основаниями, то его можно разделить на две части, в каждой из которых будет один атрибут-основание и характеризующие его признаки. Полученные части содержат меньше атрибутов и поэтому соответствуют определению показателя.

Таким образом, в показателях отображаются количественные свойства объектов и процессов. Вместе с тем существуют документы, не содержащие атрибутов-оснований, например анкеты кадрового учета, сведения о структуре подразделений предприятия и т. д. Следовательно, не вся экономическая информация может быть представлена в форме показателей.

Минимальный набор атрибутов показателя должен содержать:

- атрибуты, отображающие идентификаторы объектов,
- атрибуты, отображающие признак времени,
- атрибут, отображающий некоторое количественное свойство объекта или взаимодействия.

Для установления признаков и оснований в конкретных документах можно использовать следующие закономерности:

1. Если значение атрибута является исходным данным или результатом арифметической операции – это основание.

2. Если значение текстовое – это признак.

3. Если атрибут обозначает предмет – это признак.

4. Если атрибут в некотором показателе является признаком (основанием), – он будет играть эту роль и в других показателях.

5. Если показатели описывают сходные процессы – их признаковые части совпадают.

6. Если основание показателя вычисляется по значениям других оснований, то набор признаков такого показателя есть объединение признаков, связанных с этими основаниями.

Критерием качества создания базы данных может служить минимальная избыточность хранимой информации. Обычно минимальная избыточность выражается принципом: *каждое сообщение хранится в БД один раз*. Соблюдение этого принципа дает ряд преимуществ:

- сокращается объем памяти ЭВМ, требуемой для хранения базы данных,
- сокращается трудоемкость ввода данных в ЭВМ и упрощаются проблемы контроля достоверности вводимой информации,
- упрощаются алгоритмы корректировки данных, так как корректировка сообщения может быть проведена за одно обращение к базе данных.

Использование аппарата экономических показателей позволяет создать структуру БД с минимальной избыточностью, если сначала расчленить все сведения, циркулирующие в ЭИС, на показатели, а потом объединить атрибуты родственных показателей по принципу: *в один файл включается группа экономических показателей с одинаковым составом атрибутов-признаков.*

Пример

АТРИБУТЫ ДОКУМЕНТА «ПРИХОДНЫЙ ОРДЕР».

Дата	Кмат – код материала	Цена
Склад	Кво-док – количество по документу	Сумма
Пост – код поставщика	Кво-пр – количество принято	

Атрибутами-основаниями являются Кво-док, Кво-пр, Цена и Сумма, которые представляют количественную характеристику процесса оприходования материала на складе. Можно сделать вывод о наличии в нашем документе четырех показателей, по одному на каждое основание. Выяснение структуры каждого показателя связано с определением атрибутов-признаков для соответствующих оснований.

У основания Кво-док необходимыми признаками будут Кмат (имеется в виду количество материала), Склад и Пост (склад принимает материалы от конкретного поставщика) и Дата (необходимо указание времени). В результате структура показателя (назовем его П1) принимает вид:

П1 (Кмат, Склад, Пост, Дата, Кво-док)

При рассмотрении показателя П2 с основанием Кво-пр можно использовать правило 5 (основания Кво-док и Кво-пр описывают сходные процессы), после чего

П2 (Кмат, Склад, Пост, Дата, Кво-пр)

Для показателя П3 с основанием Цена необходимо установить, зависят ли цены материалов от предприятия-поставщика или они постоянны. Если допустить последнее, то получаем

П3 (Кмат, Цена)

Сумма в показателе П4 является результатом вычисления:

Сумма = Кво-пр * Цена,

поэтому согласно правилу 6 признаки показателя П4 получаются в результате объединения признаков из показателей П2 и П3, т.е.

П4 (Кмат, Склад, Пост, Дата, Сумма)

Указанные показатели образуют в базе данных 2 файла

F1 с атрибутами Кмат, Цена

F2 с атрибутами Кмат, Склад, Пост, Дата, Кво-док, Кво-пр, Сумма.

Одна из причин выделения показателей в особую разновидность единиц информации заключается в том, что показатель является минимальной группой атрибутов, сохраняющей информативность (осмысленность) и поэтому достаточной для образования самостоятельного документа.

Для показателей, описывающих экономические процессы (взаимодействие объектов), можно классифицировать их составные части:

- формальную характеристику, указывающую на алгоритм получения атрибута-основания в показателе,
- перечень объектов, участвующих в процессе,
- название процесса,
- единицу измерения атрибута-основания,
- определение момента времени или периода времени,
- название функции управления,
- название экономической системы, в которой происходит описываемый процесс.

Указание всех названных частей необходимо для точного обозначения показателя. Атрибуты-признаки показателя должны отображать в обязательном порядке лишь перечень объектов, участвующих в процессе, и период (момент) времени. Очень часто включается признак, отмечающий единицу измерения, а остальные характеристики показателя обычно указываются в его названии, а не в хранимых значениях.

Показатель удобно применять как обобщающую единицу измерения объема данных.

Существует аналогия между экономическими показателями и переменными с индексами, которые рассматриваются, например, в линейной алгебре. Так, показатель П3(Кмат, Цена) соответствует величине C(i), где C – цена материала с i-м кодом материала Кмат.

Переменная C соответствует атрибуту-основанию Цена, индекс i – атрибуту-признаку Кмат. В общем случае переменная всегда отображает атрибут-основание, а индексы этой переменной – значения соответствующих атрибутов-признаков показателя.

Естественное отличие состоит в том, что индекс i переменной C обычно изменяется от 1 до некоторого фиксированного значения, а номенклатурные номера материалов (и вообще любые значения атрибутов-признаков) могут кодироваться многими способами, необязательно порядковыми кодами.

Остальные показатели приходного ордера соответствуют таким переменным с индексами, как

K (i, j, m, n) для П1
P (i, j, m, n) для П2
S (i, j, m, n) для П4

где:
j – номер склада,
m – код поставщика,
n – дата.

Расчетные соотношения для показателей соответствуют выражениям для переменных с индексами, например,

$$S(i, j, m, n) = P(i, j, m, n) * C(i).$$

Закономерности, установленные в математике для арифметических операций над переменными с индексами, естественно, трансформируются в правила арифметических действий над показателями.

1. Рассмотрим показатель с числовым значением x и множеством индексов X и показатель с числовым значением y и множеством индексов Y. Пусть @ обозначает одно из четырех арифметических действий. Тогда множество индексов Z у величины z=x@y равно объединению множеств X и Y. Если множества X и Y содержат общие индексы, то необходимым условием корректности вычисления z является совпадение значений таких индексов у переменных x и y.

2. Для очень распространенных операций суммирования и умножения заимствуются правила линейной алгебры. В частности:

- если суммирование производится по двум различным индексам, каждый из которых меняется независимо от другого, то порядок суммирования безразличен,
- если пределы изменения одного индекса зависят от другого индекса суммирования, то при перемене порядка суммирования пределы изменения каждого из индексов становятся другими.

Если индексы суммирования не указаны, то суммирование производится по всем индексам, которые под знаком суммы встречаются два раза. *Индексы*, по которым ведется суммирование, называются *заглушенными*, индексы, по которым суммирование не ведется, называются *свободными*.

Представление экономической информации в форме показателей не является универсальным, так как существуют значительные массивы осмысленной экономической информации, не содержащие атрибутов-оснований (например, описания структуры экономических объектов – подразделений предприятия и т.п.).

Модель арифметических вычислений

Модель арифметических вычислений в ЭИС основывается на графе взаимосвязи показателей (или файлов). В графе G(S,U) множество вершин S = {s(i)} представляет собой все показатели (файлы), хранящиеся в базе данных. Дуга u(i,j) от s(i) к s(j) существует в том случае, если существует расчетное соотношение для показателя s(j) и в правой части этого соотношения присутствует показатель s(i).

Граф взаимосвязи показателей, дополненный параметрами потоков данных и запросов, служит основой для решения следующих задач:

- разделения промежуточных показателей на хранимые и динамически вычисляемые,
- распределения файлов по узлам вычислительной сети.

Соответствующие модели должны просчитываться заново при расширении или сокращении состава решаемых экономических задач, изменении структуры вычислительной сети.

Пример

Рассмотрим модель вычислений для задачи формирования программы поставок готовой продукции.

Поток входной информации содержит следующие показатели:

Пг(i,j) – программа поставок i- изделия j-потребителю на внутренний рынок в год,

П(i,j,k) – аналогичная программа с разбивкой по кварталам (k = 1,2,3,4),

Эг(i,j) – программа поставок i- изделия j-потребителю на экспорт в год,

Э(i,j,k) – аналогичная программа с разбивкой по кварталам.

Показатели Пг и П находятся в файле прикреплений на поставку продукции П, показатели Эг и Э – в файле нарядов-заказов Э. В файле цен находятся цены изделий Ц(i).

Выходные показатели:

Рг(i), Р(i,k) – программы поставок i-изделия на год и на квартал в натуральном выражении (файл Р).

Сг(i), С(i,k) – программы поставок i-изделия на год и на квартал в стомостном выражении (файл С).

Основные расчетные соотношения имеют вид:

$Pr(i)=\Sigma Пr(i,j)+\Sigma Эr(i,j)$,

$P(i,k)=\Sigma П(i,j,k)+\Sigma Э(i,j,k)$,

$Cr(i)=Ц(i)Pr(i)$; $C(i,k)=Ц(i)P(i,k)$.

Граф взаимосвязи файлов для указанной задачи приводится на рис. 1.7,а. В качестве новой задачи, которая расширяет состав графа, введем задачу оперативного контроля за формированием портфеля заказов с новыми файлами R (программа производства изделий на год и квартал) и Q (соответствие программы производства портфелю заказов). Расширенный граф взаимосвязи файлов показан на рис. 1.7,б.

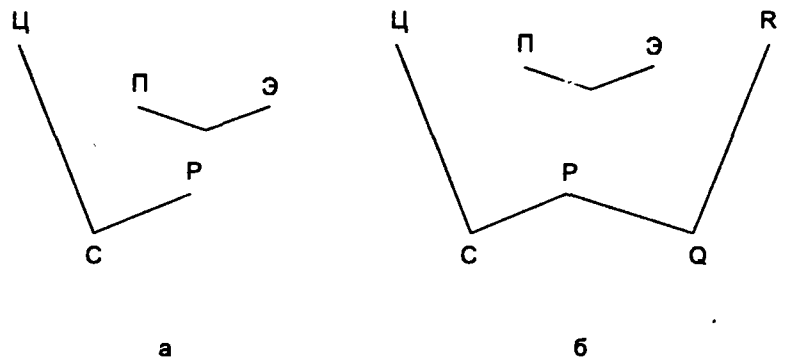


Рис. 1.7. Граф взаимосвязи файлов для задачи формирования плана поставок готовой продукции:
а — исходный граф; б — расширенный граф

Постановка экономической задачи содержит описания структуры исходных, нормативно-справочных, выходных и производных показателей, а также расчетные соотношения для вычисления выходных и промежуточных показателей, дополненные графом взаимосвязи показателей.

Материал последующих глав книги в значительной мере связан с детализацией понятий, введенных в этом пункте. Операции над единицами информации и ограничения рассматриваются в гл. 2 , методы организации значений – в гл. 3.

1.4
ЖИЗНЕННЫЙ ЦИКЛ ЭКОНОМИЧЕСКОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

В жизненном цикле ЭИС можно укрупненно выделить несколько этапов, относящихся к ее разработке, и период эксплуатации системы. *Разработкой* (проектированием) ЭИС называется процесс составления описания еще не существующей системы на разных языках и с различной степенью детализации, в ходе которого осуществляется оптимизация проектных решений. В процессе детализации описаний наступает момент, когда имеющиеся описания позволяют создать действующую систему (изготовление изделия по имеющимся чертежам) и наступает период эксплуатации ЭИС.

Проектирование разделяется на проектные операции. *Проектная операция* включает выбор проектных решений и позволяет определить значения параметров, характеризующих БД, вычислительную систему и программное обеспечение.

Этапами проектирования являются: обоснование создания ЭИС, разработка технического задания, техническое и рабочее проектирование, ввод ЭИС в действие. Процесс эксплуатации обычно через некоторые периоды времени прерывается стадиями модификации системы.

Стадию эксплуатации можно охарактеризовать как период стабильного функционирования ЭИС, не требующий изменения ранее принятых проектных решений.

Под *стадией модификации* будем понимать процесс корректировки проектных решений по отдельным компонентам ЭИС.

Более детальное описание работ на стадии проектирования включает в себя следующие действия.

1. Обследование предметной области:
- границы предметной области и возможности ее расширения,
- перечень объектов предметной области,
- информационные потребности пользователей,

- необходимые процессы обработки данных с указанием их периодичности,
- ЭВМ, на которой предполагается реализовать ЭИС,
- требования к функционированию ЭИС, частота поступления и корректировки информации, методы обеспечения ее достоверности.

Результатом обследования предметной области должно быть техническое задание на разработку системы.

2. Определение объектов и их атрибутов.

Для каждого объекта и процесса необходимо:

- выделить идентифицирующие свойства и провести нормализацию,
- определить количество экземпляров каждого объекта и рост этой величины во времени,
- определить методы вычислений производных показателей на основе значений исходных показателей.

3. Установление всех структурных связей между объектами и процессами и вычислимости на этой основе всех запросов. Разработка структуры базы данных, проверка ее корректности и полноты.

4. Определение технологии работы ЭИС, т.е. определение порядка сбора, контроля и хранения данных, определение форматов ввода-вывода информации, установление объемных и временных характеристик выдачи информации, установление правил работы всех групп пользователей.

5. Выбор ЭВМ и программных средств для реализации ЭИС. Среди программных средств в первую очередь необходимо выбрать операционную систему и СУБД. Оценка требуемых объемов памяти и трудоемкости разработки программ.

6. Проверка корректности проекта и определение сроков его реализации.

Итогом перечисленных выше действий становится технический проект ЭИС.

7. На стадии рабочего проектирования необходимо:

- создать описания всех компонентов базы данных,
- разработать экранные формы и системы меню для всех групп пользователей,

- разработать программы для всех приложений,
- заполнить ЭИС отладочными данными и оттестировать ее,
- составить инструкции по работе с ЭИС и обучить пользователей.

Стадия эксплуатации начинается с заполнения ЭИС реальными данными.

Этапы эксплуатации и модификации ЭИС поочередно меняют друг друга до тех пор, пока не наступит момент морального старения ЭИС и будет принято решение о ее ликвидации и разработке принципиально новой системы (рис. 1.8).

<table>
<tr><td colspan="3">Проектирование</td><td rowspan="2">Эксплуатация. Модификация</td><td>Утилизация</td></tr>
<tr><td>ТЗ и ТП</td><td>РП</td><td>Ввод</td><td rowspan="2">Использование старого проекта в новом
Использование содержимого БД
Продажа компонентов ЭИС</td></tr>
<tr><td>Обследование ПО
Идентификация
Структура БД
Технология
Выбор ЭВМ и СУБД
Проверка корректности</td><td>Описание БД
Интерфейсы
Программирование
Тестирование
Обучение</td><td>Актуализация БД
Опытная эксплуатация
Авторский надзор</td><td></td></tr>
</table>

Рис. 1.8. Жизненный цикл ЭИС:
ТЗ – техническое задание; ТП – технический проект;
РП – рабочий проект; Э – эксплуатация; М – модификация

На стадии эксплуатации ЭИС требуется обеспечить реорганизацию БД, рестарт и восстановление, копирование БД, контроль непротиворечивости БД.

Сопровождение программного обеспечения на стадии эксплуатации ЭИС осуществляет прикладной программист. Сопровождение базы данных реализует администратор базы данных. Сопровождение вычислительной системы выполняют операторы и сменные инженеры.

Важность исследования процессов модернизации ЭИС можно пояснить такими данными: стоимостные затраты на модернизацию ЭИС достигают примерно трети объема эксплуатационных расходов, за год в ЭИС обычно меняется 10–40% первичных документов и 20–50% выходных документов.

Экономическим объектам свойственны динамичность и развитие, что непосредственно влияет на состояние ИС. Поэтому на стадии эксплуатации ИС усиливаются факторы, доказывающие необходимость последующей модернизации. Среди них:

- изменения на объекте управления и во внешней среде (дрейф параметров предметной области),
- изменение состава рабочей нагрузки вычислительной системы, замена оборудования, рост объема файлов,
- накопление опыта работы с ЭИС,
- обнаружение проектных ошибок.

Задачи модификации ЭИС обычно рассматриваются как неперспективные и нежелательные. Эта работа считается очень простой в сравнении с проектированием ЭИС, она ассоциируется с исправлением проектных ошибок, сделанных другими. Вместе с тем модифицируемая система обычно плохо документирована, попытки улучшения прикладных программ иногда кончаются ничем. В итоге довольно быстро наступает момент, когда интерес к системе теряется и начинается новая разработка. Однако планомерная модификация базы данных и других компонентов ЭИС позволяет поддерживать в требуемых границах ее технические и эксплуатационные характеристики, отсрочить момент морального старения системы.

На стадии эксплуатации системы в отсутствие специальных мероприятий по модернизации ИС ухудшаются ее эксплуатационные показатели, например, снижается пропускная способность. Происходит также ухудшение соответствия между параметрами предметной области и параметрами БД.

В процессе эксплуатации ЭИС производится слежение за изменением параметров ЭИС и предметной области. Для этого используются, например:

- информация об изменениях в системе документооборота и структуре отдельных документов,
- данные об изменениях в составе решаемых экономических задач, системе экономических показателей и методах их расчета,
- характеристики потока запросов к БД,
- оценки пользователей о качестве получаемой информации,
- информация системной мониторной программы или аналогичных средств, работающих в составе применяемых операционных систем и СУБД, сбор статистики о выполненных заданиях.

Должны также фиксироваться изменения количественных и качественных характеристик предметной области. В этой сфере могут происходить изменения в организационной структуре экономического объекта, составе параметров, характеризующих объект, методах их расчета. Изменения зачастую связаны с реконструкцией производства, выпуском новых изделий, освоением новых технологий, совершенствованием конструкторской документации. Может меняться состав организационных и технологических ограничений на объекте.

Сравнение результатов измерений с аналогичной информацией за прошлые периоды времени и отклонение текущих параметров функционирования ЭИС от нормативных могут дать основание для проведения модификации ЭИС. Анализ результатов наблюдений должен быть различным в зависимости от целей, которые предполагается достичь после проведения модификации. Первоначально должна быть поставлена цель модификации ЭИС и определено множество методов, ведущих к достижению требуемой цели. Собираемая и анализируемая информация должна лишь доказать (или опровергнуть) целесообразность применения конкретного метода модификации и позволить выработать его спецификацию (рис. 1.9).

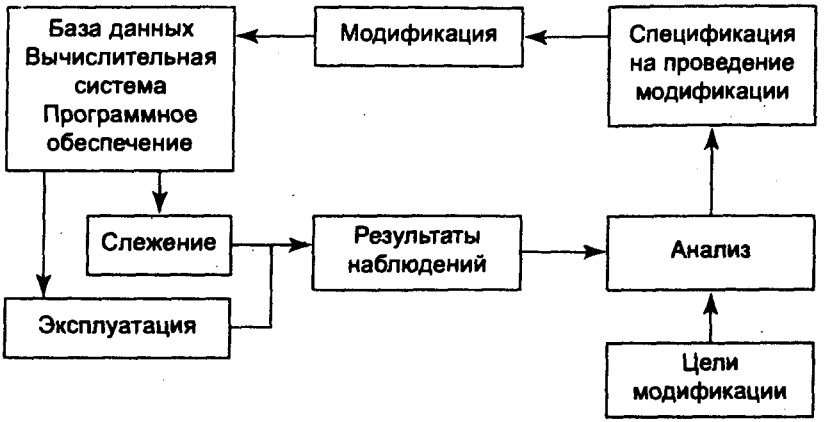


Рис. 1.9. Взаимозависимость действий на стадиях эксплуатации и модификации ЭИС

Цели модификации ЭИС можно разделить на шесть больших групп:

- исправление проектных ошибок,
- улучшение эксплуатационных характеристик ЭИС,
- адаптация к изменениям в предметной области,
- разработка нового приложения,
- обеспечение совместимости с другими ИС,
- перенос БД в новую аппаратно-программную среду.

Конкретные методы модификации ЭИС группируются по четырем направлениям (см. табл. 1.3):

- реструктуризация БД,
- перепрограммирование прикладных задач,
- реорганизация БД,
- настройка вычислительной системы.

Данные таблицы показывают, что автономные операции над базой данных (реструктуризация и реорганизация) не могут быть эффективно использованы для целей модификации без одновременного совершенствования вычислительной системы и программного обеспечения.

Таблица 1.3. **Соответствие целей и методов модификации ЭИС**

Цели модификации	Методы модификации ЭИС			
	Реструктуризация	Перепрограммирование прикладных задач	Реорганизация	Настройка вычислительной системы
Исправление проектных ошибок	+	+		+
Улучшение эксплуатационных характеристик	+	+	+	+
Адаптация к изменениям в предметной области	+	+		
Разработка нового приложения	+	+		+
Совместимость с другими ЭИС	+	+		+

Большинство процедур модификации ЭИС могут производиться без прекращения стадии эксплуатации. Однако необходим контроль всех компонентов ЭИС (базы данных, вычислительной системы, программных средств) после проведения каких-либо усовершенствований.

ВОПРОСЫ И ЗАДАНИЯ

1. *Изобразите* бланк документа, который соответствует описанию записи zap на языке программирования Паскаль.

2. *Представьте* результат нормализации СЕИ со структурой C1(10).(C2(5).(P1,P2,P3),C3(7).(P4,P5,P6))?

3. *Определите* состав показателей в приводимых ниже документах. Имена атрибутов выберите самостоятельно. Определите количество и атрибутный состав файлов в базе данных для представления каждого документа.

1. Атрибуты документа "Карточка водителя":

Табельный номер	Разряд работы	Часы работы
ФИО водителя	Номер путевого листа	Оплата по тарифу
Номер автомашины	Дата	Надбавка за ремонт

2. Атрибуты документа "Кассовый отчет кинотеатра":

Кинотеатр	Фильм	Число проданных мест
Число мест	Режиссер	Выручка от фильма
Дата		

3. Атрибуты документа "Акт о ликвидации основных средств":

Название объекта	Код подразделения	Сумма износа
Год изготовления	Дата ликвидации	Остаточная стоимость
Инвентарный номер	Первоначальная стоимость	

4. Для документов, используемых в задаче 3, укажите атрибуты, которые обозначают:

- объект;
- свойство объекта (со ссылкой на название объекта);
- взаимодействие объектов (с указанием объектов, участвующих во взаимодействии).

Глава 2

МОДЕЛИ ДАННЫХ

2.1 РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ

Определение модели данных предусматривает указание множества допустимых информационных конструкций, множества допустимых операций над данными и множества ограничений для хранимых значений данных.

Модель данных, с одной стороны, представляет собой формальный аппарат для описания информационных потребностей пользователей, а с другой – большинство СУБД ориентируются на конкретную модель данных, и, таким образом, если информационные потребности удается точно выразить средствами одной из моделей данных, то соответствующая СУБД позволяет относительно быстро создать работоспособный фрагмент ЭИС.

Информационные конструкции, операции и ограничения моделей данных выбираются из достаточно небольшого множества вариантов, характеризующего "крупные" информационные объекты и операции. В частности, не допускается рассмотрение отдельных символов данных, операций сложения атрибутов, ограничения на соответствие типов данных и т. п., что характерно для языков программирования.

Классификация информационных конструкций (информационных объектов) тесно связана с областью их использования в ЭИС.

1. Объекты для технологии баз данных – отношения и веерные отношения.

2. Объекты для технологии искусственного интеллекта – предикаты, фреймы и семантические сети.

3. Объекты для технологии мультимедиа – тексты, графические изображения, фонограммы и видеофрагменты.

Информационные объекты послужили основой для объектно-ориентированного проектирования систем, когда фиксируется множество информационных объектов и действий над объектами. Типичный список действий включает в себя создание/уничтожение объекта, редактирование объекта, фиксацию одного объекта в качестве части другого объекта, связывание объектов, синхронизацию действий над объектами.

Довольно-таки часто все названные объекты встраиваются в структуру отношений, которые можно считать простейшими универсальными объектами.

Количество существенно различных моделей данных определяется наличием различных множеств информационных конструкций. С этой точки зрения принципиальными различиями обладают три модели данных – реляционная, сетевая и иерархическая.

Реляционная модель данных характеризуется следующими компонентами:

- информационной конструкцией – отношением с двухуровневой структурой,
- допустимыми операциями – проекцией, выборкой, соединением и некоторыми другими,
- ограничениями – функциональными зависимостями между атрибутами отношения.

Каждому классу объектов P материального мира ставится в соответствие некоторое множество атрибутов, например A1, A2,...,An. Отдельный объект класса P описывается строкой величин (a1, a2, ..., an), где ai – значение атрибута Ai.

Строка (a1, a2, ..., an) называется *кортежем*. Всему классу объектов соответствует множество кортежей, называемое *отношением*. Обозначим отношение, описывающее класс объектов P, также через P.

Выражение P(A1, A2,...,An) называется *схемой отношения* P.

Для каждого компонента кортежа должна быть указана ее связь с соответствующим атрибутом. В реляционной модели данных для обеспечения этой связи порядок компонентов кортежа совпадает с порядком следования атрибутов в схеме отношения.

Каждое отношение представляет состояние класса объектов в некоторый момент времени. Следовательно, одной схеме отношения в разные моменты времени могут соответствовать разные отношения.

Множество значений отношения можно представить в виде таблицы, в которой соблюдаются следующие соответствия:

- название таблицы и перечень названий граф соответствуют схеме отношения,
- строке таблицы соответствует кортеж отношения,
- все строки таблицы (и соответственно все кортежи) различны,
- порядок строк и столбцов произвольный (в частности, реляционная модель данных не предполагает специальную сортировку строк).

Отношение реляционной БД может быть описано в терминах теории множеств. Рассмотрим множество доменов

$$D = \{ D1, D2, D3, ..., Dk \}$$

и создадим декартово произведение доменов

$$U = D1 \cdot D2 \cdot D3 \cdot ... \cdot Dk.$$

Каждый элемент U имеет вид (d1, d2, d3,...,dk), где d1 – элемент домена D1, d2 – элемент D2 и т.д. В множестве U представлены всевозможные сочетания значений доменов, полученные по названному принципу. Среди элементов из U содержатся такие, которые не соответствуют реально имеющимся сообщениям и не могут храниться в БД. Отношение R, включающее истинные сообщения, является подмножеством U. Атрибуты отношения R используют домены из D в качестве своих областей определения.

Реляционная база данных представляет собой множество отношений.

Схема реляционной БД содержит следующие компоненты

$$S(rel) = <A,R,Dom,Rel,V(s)>,$$

где A – множество имен атрибутов,
R – множество имен отношений,
Dom – вхождение атрибутов в домены,
Rel – вхождение атрибутов в отношения,
V(s) – множество ограничений (в том числе функциональных зависимостей).

Описание процессов обработки отношений может быть выполнено двумя способами:

- указанием перечня операций, выполнение которых приводит к требуемому результату (процедурный подход),
- описанием свойств, которым должно удовлетворять результирующее отношение (декларативный подход).

Приводимые далее операции над отношениями ориентированы на процедурное описание процессов обработки данных.

Множество отношений и операций над ними образует реляционную алгебру.

При рассмотрении операций над отношениями будет использоваться как алгебраическая форма записи операций, так и запись на языке реляционных СУБД семейства dBASE (для определенности – dBASE 3). Далее будут рассмотрены аналогичные конструкции языка SQL.

Как правило, список операций содержит проекцию, выборку, объединение, пересечение, вычитание, соединение и деление.

Проекцией называется операция, которая переносит в результирующее отношение те столбцы исходного отношения, которые указаны в условии операции. Алгебраическая запись проекции имеет вид:

$$T = R[X],$$

где R – исходное отношение,
T – результирующее отношение,
X – список атрибутов в структуре отношения T (условие проекции).

Пример

Рассмотрим два отношения: W1, содержащее сведения о продаже продукции в 1998 г., и W2, в котором указаны цены на продукцию и комплектующие изделия для соответствующих видов продукции.

W1			
Магазин	Продукция	План	Факт
Динамо	Эдв-12	120	140
Динамо	ЗВИ	200	200
АТЭ	Эдв-12	80	170
АТЭ	Эдв-30	150	100

W2		
Продукция	Цена	Компл
Эдв-12	40	Вк-15
Эдв-12	40	Р-20
Эдв-30	20	Вк-15
Эдв-30	20	Р-20
ЗВИ	120	Р-20

Если требуется отношение X1, содержащее сведения только о фактическом выпуске продукции, то оно получается в результате выполнения проекции

X1 = W1[Магазин, Продукция, Факт]

и имеет вид:

X1		
Магазин	Продукция	Факт
Динамо	Эдв-12	140
Динамо	ЗВИ	200
АТЭ	Эдв-12	170
АТЭ	Эдв-30	100

Столбцы результирующего отношения могут указываться в любом порядке:

X2 = W1[Продукция, Магазин, Факт].

В СУБД DBASE проекция реализуется двумя командами и этот же пример выглядит как

use W1 && открыть отношение W1

copy to X1 field Магазин, Продукция, Факт && проекция

Знак && означает комментарий.

Следующий пример проекции – это получение справочника X3 цен на продукцию:

use W2

copy to X3 field Продукция, Цена

Значения отношения X3 показаны ниже:

X3	
Продукция	**Цена**
Эдв-12	40
Эдв-12	40
Эдв-30	20
Эдв-30	20
ЗВИ	120

Значительным неудобством в X3 является наличие одинаковых строк. В реляционной алгебре требуется, чтобы результат проекции не содержал одинаковых строк, однако практически такие СУБД, как dBASE, не обеспечивают этого. Правильным результатом проекции X4 = W2[Продукция, Цена] является отношение

X4	
Продукция	**Цена**
Эдв-12	40
Эдв-30	20
ЗВИ	120

Отношение с заданной структурой зачастую можно получить из различных исходных отношений БД. В этом случае, естественно, не гарантируются одинаковые результаты.

Пусть нам необходим список отделов учреждения. В базе данных имеются отношения Служащий (Фамилия, Отдел,...) и Технолог (Фамилия,Отдел,...). Проекция Служащий[Отдел] формирует полный список отделов, а Технолог[Отдел] может содержать меньше значений, если в некоторых отделах не работают технологи.

Выборкой называется операция, которая переносит в результирующее отношение те строки из исходного отношения, которые удовлетворяют условию выборки. Условие выборки проверяется в каждой строке отношения по отдельности и не может охватывать информацию из нескольких строк. Существуют две простейшие разновидности условия выборки:

1. Условие вида Имя_атрибута <знак сравнения> Значение, где допускаются знаки сравнения =, #, >, =>, <, <=. Например, Цена > 100.

2. Условие вида Имя_атрибута_1 <знак сравнения> Имя_ _атрибута_2. Например, Факт > План.

Имена атрибутов должны содержаться в структуре исходного отношения. Алгебраическая запись выборки имеет вид

$$T = R[p],$$

где R – исходное отношение;
T – результирующее отношение;
p – условие выборки.

В качестве **примера** получим значения

X5 = W1[Продукция = "Эдв-12"]

X5			
Магазин	Продукция	План	Факт
Динамо	Эдв-12	120	140
АТЭ	Эдв-12	80	170

В СУБД dBASE выборка реализуется командой copy с опцией for, например,

```
use W1
copy to X6 for Факт => План
```

X6			
Магазин	Продукция	План	Факт
Динамо	Эдв-12	120	140
Динамо	ЗВИ	200	200
АТЭ	Эдв-12	80	170

В команде copy условия выборки и проекции могут присутствовать одновременно, но атрибуты условия выборки должны включаться в условие проекции.

Операции объединения, пересечения и *вычитания* производятся над двумя исходными отношениями с одинаковой структурой. Точных аналогов этих операций в dBASE нет.

Обозначим исходные отношения через R1 и R2, а результирующее – T.

Объединение T = U(R1, R2) содержит строки, присутствующие либо в отношении R1, либо в R2.

Пересечение T = I(R1, R2) содержит строки, присутствующие в отношениях R1 и R2 одновременно.

Вычитание T = M(R1, R2) содержит те строки из R1, которые отсутствуют в R2.

Если справочник цен на продукцию X4 необходимо дополнить новыми сведениями из отношения W3, то надо выполнить объединение

$$X7 = U\ (X4,\ W3).$$

W3	
Продукция	Цена
Эдв-42	40
Тэн-10	15

X7	
Продукция	Цена
Эдв-12	40
Эдв-30	20
ЗВИ	120
Эдв-42	40
Тэн-10	15

В dBASE предусмотрена операция добавления append, которая для нашего примера реализуется командами

```
use X4
append from W3,
```

что приводит к добавлению в отношение X4 всех строк из отношения W3. Отличия добавления от объединения – создание

результирующего отношения на месте исходного отношения и возможность появления одинаковых строк в результирующем отношении.

Операция соединения отношений выполняется над двумя исходными отношениями и создает одно результирующее. Каждая строка первого исходного отношения сопоставляется по очереди со всеми строками второго отношения, и если для этой пары строк соблюдается условие соединения, то они сцепляются и образуют очередную строку в результирующем отношении. Условие соединения имеет вид:

Имя_атрибута_1 <знак сравнения> Имя_атрибута_2,

где Имя_атрибута_1 находится в одном исходном отношении, а Имя_атрибута_2 – в другом. Будем использовать следующее обозначение операции соединения:

$$T = R1[p]R2,$$

где R1 и R2 – исходные отношения,
T – результирующее отношение,
p – условие соединения.

Практически наиболее важный частный случай соединения называется натуральным соединением и имеет следующие особенности:

- знаком сравнения в условии соединения является "=",
- Имя_атрибута_1 и Имя_атрибута_2 должны совпадать, а точнее, содержать пересечение списков атрибутов исходных отношений,
- список атрибутов результирующего отношения образуется в результате объединения списков атрибутов исходных отношений.

Обозначение натурального соединения не содержит условия соединения и имеет вид T = R1 * R2.

Если требуется сведения о продаже продукции из отношения W1 дополнить данными о ценах на продукцию из отношения X7, то задача решается с помощью соединения

X8 = W1[Продукция = Продукция]X7

X8					
Магазин	Продукция	План	Факт	Продукция	Цена
Динамо	Эдв-12	120	140	Эдв-12	40
Динамо	ЗВИ	200	200	ЗВИ	120
АТЭ	Эдв-12	80	170	Эдв-12	40
АТЭ	Эдв-30	150	100	Эдв-30	20

Первая строка из W1 и первая строка из X7 удовлетворяют условию Продукция = Продукция, поэтому сцепляются. Остальные строки из X7 не будут сцепляться с первой строкой из W1 (условие не соблюдается). Вторая строка из W1 при сравнении со всеми строками из X7 сцепится только с третьей строкой и т. д. Если применять операцию натурального соединения, то в отношении X8 будет отсутствовать второй столбец с именем Продукция.

В СУБД DBASE для проведения соединения необходимо использовать две области для исходных отношений:

```
SELECT 1
USE W1 && открытие отношения W1
SELECT 2
USE X7 && открытие отношения X7
SELECT 1
JOIN WITH X7 TO X9 FOR Продукция=X7->Продукция
        FIELDS Магазин, Продукция, X7->Цена, План, Факт
```

Опция with называет второе исходное отношение, опция for содержит условие соединения, опция fields перечисляет имена атрибутов результирующего отношения. В приведенном примере выполнено натуральное соединение и передвинут столбец Цена.

В ряде случаев соединение дает некорректные результаты, например:

X10 = W1 * W2

содержит неоднократно одинаковые сообщения о выпуске продукции только потому, что эти виды продукции используют несколько комплектующих изделий.

X10					
Магазин	Продукция	План	Факт	Цена	Компл
Динамо	Эдв-12	120	140	40	Вк-15
Динамо	Эдв-12	120	140	40	Р-20
Динамо	ЗВИ	200	200	120	Р-20
АТЭ	Эдв-12	80	170	40	Вк-15
АТЭ	Эдв-12	80	170	40	Р-20
АТЭ	Эдв-30	150	100	20	Вк-15
АТЭ	Эдв-30	150	100	20	Р-20

Вообще говоря, ограничения, которые присутствуют в исходных отношениях, определенным образом трансформируются в ограничения для отношений, полученных в результате применения операций реляционной алгебры. Соответствующие закономерности будут рассмотрены в п. 2.2 в связи с анализом ограничений, присущих реляционной модели данных.

Натуральное соединение определено и в тех случаях, когда соединяемые отношения совпадают по структуре или не содержат общих атрибутов. Если структура отношений R и S одинакова, то R * S выполняет фактически пересечение I(R,S).

Когда отношения R и S не содержат общих атрибутов, считается, что условие соединения выполнено для любой пары сопоставляемых строк отношений, и R * S сцепляет каждую строку из R со всеми строками из S.

Операция натурального соединения имеет ряд свойств, например коммутативность и ассоциативность.

Свойство коммутативности означает, что операции

$$X1=R * S \text{ и } X2=S * R$$

порождают, в сущности, одно и то же отношение.

Свойство ассоциативности означает, что операция

$$Y1=(R*S)*T \text{ и операция } Y2=R*(S*T)$$

дают одинаковый результат. Различия, безусловно, состоят в неодинаковом порядке строк X1 и X2 (Y1 и Y2 соответственно). Кроме того, промежуточные отношения при вычислении Y1 и Y2 могут иметь резко различающиеся размеры (число строк), хотя Y1 и Y2 содержат равное число строк.

Описание *операции деления отношений* начнем с **примера**.

Пусть существует отношение Y(ФИО, ЯП), где для каждого программиста с фамилией ФИО указываются языки программирования ЯП, которые он знает.

Y	
ФИО	ЯП
Иванов	Си
Иванов	Фортран
Иванов	Паскаль
Петров	Си
Петров	Паскаль
Семин	Си
Семин	Фортран
Яшин	Фортран
Яшин	Паскаль

Необходимо выделить фамилии программистов, знающих языки Си и Фортран одновременно. Попытка воспользоваться операцией выборки

$$X11 = Y[\text{ ЯП = “Си” AND ЯП = “Фортран”}]$$

(через AND обозначена логическая операция “и”) будет безуспешной, так как в одной строке отношения нет информации о двух языках программирования сразу и отношение X11 будет пустое.

Определим операцию, называемую “образ”. В отношении T(A,B) образом значения a атрибута A является множество значений атрибута B, и каждый элемент b этого множества образует вместе с a некоторую строку (или часть строки) отношения T.

$$\text{im } B(a) = \{b1, b2, \ldots, bk\},$$

где im – знак операции “образ”,
a – значение, образ которого вычисляется,
B – имя атрибута для образа значения a,
b1, b2,...,bk – значения атрибута B.

Стоящая перед нами задача решается путем вычисления образов значений “Си” и “Фортран” и последующего пересечения найденных образов.

im ФИО("Си") = {"Иванов","Петров","Семин"}

im ФИО("Фортран") = {"Иванов","Семин","Яшин"}

im ФИО("Си") ∩ im ФИО("Фортран") = {"Иванов","Семин"}

Такая связка операций взятия образа и пересечения полученных множеств (количество значений, для которых вычисляется образ, может быть произвольным, а не 2, как в нашем примере) требуется достаточно часто, поэтому вводится специальная операция – *деление*.

Условимся, что существует отношение-делимое W(A,B) (в нашем примере это Y) и отношение-делитель V(A). Для необходимого нам запроса отношение-делитель имеет вид:

Z
ЯП
Си Фортран

Результатом операции деления является отношение Q(B), *содержащее пересечение образов всех строк отношения-делителя* V(A), *вычисленных на основе отношения-делимого* W(A,B),

$$Q = D\,(W,V),$$

где D – знак операции деления.

Результат деления X12 = D (Y,Z) содержит следующие значения:

X12
ФИО
Иванов Семин

В СУБД семейства DBASE отсутствует операция деления отношений, и для реализации деления необходима специальная подпрограмма.

Декларативный подход к обработке реляционных баз данных основан на интерпретации понятий и методов математической логики. В частности, реляционное исчисление базируется на исчислении предикатов. Перечислим необходимые для реляционного исчисления понятия математической логики.

1. Символы переменных и констант. В языковых конструкциях реляционного исчисления им соответствуют имена атрибутов и переменных, а также константы.

2. Логические связки "и", "или", "не" и знаки сравнения =, # (не равно), >, <, >=, <=.

3. Термы, т. е. любые константы и переменные, а также функции, аргументами которых служат термы.

4. Элементарные формулы – предикаты, аргументами которых являются термы. Предикаты, связанные операциями "и", "или", "не", также являются элементарными формулами. Элементарными формулами служат, например, выражения Фамилия = "Мишенин" и Сумма <= Итог.

5. Формулы, т. е. результат применения кванторов общности или существования к элементарным формулам. Формула соответствует запросу к реляционной базе данных, выраженному средствами реляционного исчисления.

В применяемых ниже языковых конструкциях используется синтаксис языков ALPHA и QUEL. Объявление переменной и ее привязка к определенному отношению производятся с помощью оператора Диапазон (range) следующего вида

range of <переменная> is <отношение><квантор>

Квантор принимает одно из значений:

some, когда переменная обозначает одну из строк отношения (квантор существования);

all, когда переменная обозначает все строки отношения (квантор общности).

Основной формой оператора запроса Получить (get) является выражение

get <отношение> (<целевой список>) where <условие>.

Условие соответствует формуле исчисления кортежей. В условиях можно использовать все знаки сравнения и логические связки "и", "или", "не".

Языковая конструкция для кванторов может быть такой:

range of Z is Перевозка some.

Переменная Z при вычислении означает одну из строк (some – квантор существования) отношения Перевозка.

range of W is Перевозка all.

Переменная W означает все строки (all – квантор общности) отношения Перевозка.

Реализацией запроса к базе данных из "Получить даты поставок из города Киева" является запись средствами реляционного исчисления:

range of Z is Перевозка some
get W (Дата) where Поставщик.Код_поставщика=Z.Код_поставщика
and Поставщик.Город="Киев".

В последнее время наиболее распространенным декларативным языком запросов является SQL (структурированный язык запросов). Центральным средством доступа к БД в SQL являются команда Select и ее параметры – From, Where, Group by, Having, Order by.

В команде Select указываются имена выводимых атрибутов или знак *, если надо выводить все атрибуты. Параметр From является обязательным и содержит имена требуемых для выполнения запроса отношений. Если здесь указано несколько отношений, то они будут преобразованы командами натурального соединения в одно отношение. Параметр Where оп-

ределяет условия, которым должны удовлетворять выводимые данные. В записи условий применяются знаки сравнения (=,> и т.д.), опции All, Any, Between, Exists, Like, In и логические операторы. Параметр Group by объединяет записи с одинаковым значением некоторого атрибута-ключа. Параметр Having при необходимости проверяет условия внутри группы записей, выделенных с помощью Group by. Параметр Order by определяет имена атрибутов, по которым должен быть отсортирован результат.

Ранее упомянутый запрос "Получить даты поставок из города Киева" выполняется командой

```
Select Дата
From Перевозка, Поставщик
Where Поставщик.Город="Киев"
```

Другой подход к декларативному представлению запросов реализован в языке Пролог. Значения отношения в Прологе хранятся в виде множества фактов. Например, для отношения Изделие (Код, Поставщик, Цена) факты могут иметь вид

```
Изделие (513,"Динамо",800)
Изделие (155,"Динамо",600)
Изделие (247,"АТЭ",600)
```

Простейший запрос для отношения Пролога сводится к указанию в структуре отношения имени переменной на месте атрибута–цели запроса, указанию требуемых значений на месте атрибутов, для которых задано условие выборки, и указанию знака _ на месте атрибутов, значение которых безразлично.

Например, запрос "Получить список изделий с ценой 600" реализуется выражением goal (цель)

```
Goal: Изделие (x,_,600)
```

Ответ Пролога имеет вид

```
x=155
x=247.
```

2.2
НОРМАЛИЗАЦИЯ ОТНОШЕНИЙ

Центральная задача проектирования базы данных ЭИС – определение количества отношений (или иных составных единиц информации) и их атрибутного состава.

Задача группировки атрибутов в отношения, набор которых заранее не фиксирован, допускает множество различных вариантов решений. Рациональные варианты группировки должны учитывать следующие требования:

- множество отношений должно обеспечивать минимальную избыточность представления информации,
- корректировка отношений не должна приводить к двусмысленности или потере информации,
- перестройка набора отношений при добавлении в базу данных новых атрибутов должна быть минимальной.

Нормализация представляет собой один из наиболее изученных способов преобразования отношений, позволяющих улучшить характеристики БД по перечисленным критериям.

Ограничения на значения, хранимые в реляционной базе данных, достаточно многочисленны. Соблюдение этих ограничений в конкретных отношениях связано с наличием так называемых нормальных форм. Процесс преобразования отношений базы данных к той или иной нормальной форме называется *нормализацией отношений*. Нормальные формы нумеруются последовательно от 1 по возрастанию, и чем больше номер нормальной формы, тем больше ограничений на хранимые значения должно соблюдаться в соответствующем отношении.

Ограничения, типичные для реляционной модели данных, – это функциональные и многозначные зависимости, а также их обобщения. В принципе, множество дополнительных ограничений может расти и соответственно будет увеличиваться число нормальных форм. Применяемые ограничения ориентированы на сокращение избыточной информации в реляционной базе данных.

Отношение в первой нормальной форме (сокращенно 1НФ) – это обычное отношение с двухуровневой структурой. Недопустимость в структуре отношения третьего и последующих уровней является ограничением, определяющим 1НФ отношения.

Преобразование ненормализованного отношения в представление, соответствующее 1НФ, – это операция нормализации, рассмотренная выше. Следует отметить определенное терминологическое несоответствие – нормализация СЕИ приводит к 1НФ, а нормализация отношений реляционной БД обычно производится до 3НФ или 4НФ.

Реляционная база данных в целом характеризуется 1НФ, если все ее отношения соответствуют 1НФ.

Следующие нормальные формы (вторая и третья) используют ограничения, связанные с понятием функциональной зависимости, которое необходимо предварительно рассмотреть.

2.2.1 Функциональные зависимости и ключи

Функциональные зависимости определяются для атрибутов, находящихся в одном и том же отношении, удовлетворяющем 1НФ.

Простейший случай функциональной зависимости охватывает 2 атрибута. *В отношении* R(A,B,...) *атрибут* A *функционально определяет атрибут* B, *если в любой момент времени каждому значению* A *соответствует единственное значение* B (обозначается A → B).

Иначе говорят, что B функционально зависит от A (обозначается B = f(A)). Первое обозначение оказывается более удобным, когда число функциональных зависимостей растет и их взаимосвязи становятся труднообозримыми; оно и будет использоваться в дальнейшем. Отсутствие функциональной зависимости обозначается A —/→ B.

Можно определить ситуацию A → B с помощью операции "образ", сказав что множество im B(a) должно содержать один элемент для любого значения a атрибута A.

Рассмотрим простой **пример** с атрибутами ФИО и ГР (год рождения) в отношении R1.

R1	
ФИО	**ГР**
Иванов	1960
Зуев	1963
Смирнов	1960
Яшина	1961

Предположим, что в столбце ФИО представлены сведения о разных людях и соответствующие значения в столбце не повторяются. Тогда можно утверждать наличие функциональной зависимости ФИО → ГР, поскольку каждому значению атрибута ФИО в отношении R1 соответствует единственное значение атрибута ГР. Можно утверждать, что это ограничение будет соблюдаться и далее, так как оно перефразируется в утверждение: *у каждого человека единственный год рождения*, которое справедливо.

Практически каждое ограничение для проверки функциональной зависимости можно преобразовать в утверждение о свойствах объектов предметной области, которое можно проверить, не анализируя множество значений соответствующего отношения. Именно так мы и будем поступать в дальнейшем. Наличие в столбце ГР повторяющихся годов (1960) не опровергает установленной нами зависимости, но это означает ГР —/→ ФИО.

Одновременное соблюдение двух зависимостей вида A → B и B → A называется *взаимно-однозначным соответствием* и обозначается A ↔ B.

В качестве **примера** рассмотрим отношение R2 с атрибутами Магазин и Расч (номер расчетного счета).

R2	
Магазин	**Расч**
ММЗ	704098
Динамо	122095
АТЭ	440162

Можно утверждать, что у каждого магазина единственный номер расчетного счета и каждый расчетный счет принадлежит единственному магазину. Это доказывает справедливость функциональных зависимостей Магазин → Расч и Расч → Магазин, т.е. Магазин ↔ Расч.

Наконец, самыми распространенными являются случаи отсутствия функциональных зависимостей, например, ФИО —/→ Дисциплина и Дисциплина —/→ ФИО в отношении R3, описывающем экзамены студентов. Здесь каждый студент сдает экзамены по нескольким дисциплинам, и по каждой дисциплине экзамен сдается многими студентами.

R3	
ФИО	Дисциплина
Петров	Физика
Федин	Химия
Алешин	Физика
Петров	Химия

Таким образом, для атрибутов A и B некоторого отношения возможны следующие ситуации:

- отсутствиефункциональнойзависимости,
- наличие A → B (или B → A), но не обе зависимости вместе,
- наличиевзаимно-однозначногосоответствияA↔B.

Понятие функциональной зависимости распространяется на ситуацию с тремя и более атрибутами в следующей форме. Группа атрибутов (для определенности A,B,C) функционально определяет атрибут D в отношении T(A,B,C,D,....), если каждому сочетанию значений <a,b,c> соответствует единственное значение d (a – значение A, b – значение B, c – значение C, d – значение D). Наличие такой функциональной зависимости будем обозначать A,B,C → D. Случай, когда в правой части функциональной зависимости присутствует несколько атрибутов, не нуждается в специальном рассмотрении. Взаимно-однозначные соответствия для трех и более атрибутов также не имеют самостоятельного значения.

Пусть в отношении T1 представлены сведения о закончившихся экзаменах.

T1				
ФИО	Дисциплина	Дата	Преподаватель	Оценка
Петров	Физика	01.06.98	Иванов	4
Федин	Химия	01.06.98	Смирнов	5
Алешин	Физика	01.06.98	Иванов	5
Петров	Химия	07.06.98	Смирнов	5

Ограничение, состоящее в том, что студент не может в один день сдать два и более экзаменов, означает справедливость ряда функциональных зависимостей:

ФИО, Дата → Дисциплина,

ФИО, Дата → Преподаватель,

ФИО, Дата → ОЦЕНКА.

Существование функциональных зависимостей связано с применяемыми способами кодирования атрибутов. Так, для множества учреждений можно утверждать, что каждый отдел (как объект предметной области) относится к единственному учреждению. Однако этого недостаточно для доказательства функциональной зависимости Отдел → Учреждение. Если в каждом учреждении отделы нумеруются последовательно, начиная с 1, то функциональная зависимость неверна. Если же код отдела, кроме номера, содержит и код учреждения (или уникальность кодов обеспечивается каким-то другим способом), то функциональная зависимость Отдел →Учреждение справедлива. Зависимость ФИО → ГР в R1 соблюдается, если ФИО является атрибутом- идентификатором для каждого человека, что может быть справедливо только для небольших множеств людей. Невнимание к способам кодирования атрибутов может привести к несоответствию функциональных зависимостей и хранящихся данных, что является серьезной проектной ошибкой.

Для показателя со множеством атрибутов-признаков P={P1,P2,...,Pn} и атрибутом-основанием Q справедлива функциональная зависимость P → Q, хотя нельзя утверждать, что это единственная зависимость на указанных атрибутах.

С помощью функциональных зависимостей определяется понятие ключа отношения, точнее ряд разновидностей ключей – вероятные, первичные и вторичные.

Вероятным ключом отношения называется такое множество атрибутов, что каждое сочетание их значений встречается только в одной строке отношения, и никакое подмножество атрибутов этим свойством не обладает. Вероятных ключей в отношении может быть несколько.

Рассмотрим в качестве **примера** отношение T3 (имена и значения атрибутов – условные):

T3				
ZEN	RAM	AST	SPIM	BIG
1A	31	dwa	wii	73
3B	21	bun	cup	40
3D	30	mun	lam	58
4D	31	sab	wii	40
7B	30	sab	irt	38

Можно утверждать, что вероятным ключом отношения T3 является атрибут ZEN (значения в столбце ZEN не повторяются). Кроме того, еще один вероятный ключ представлен парой атрибутов RAM, AST.

Важность вероятных ключей при обработке данных определяется тем, что выборка по известному значению вероятного ключа дает в результате одну строку отношения либо ни одной.

На практике атрибуты вероятного ключа отношения связываются со свойствами тех объектов и событий, информация о которых хранится в отношении. Если в результате корректировки отношения изменились имена атрибутов, образующих ключ, то это свидетельствует о серьезном иска-

жении информации. Следовательно, систематическая проверка свойств вероятного ключа позволяет следить за достоверностью информации в отношении.

Когда в отношении присутствует несколько вероятных ключей, одновременное слежение за ними очень затруднено и целесообразно выбрать один из них в качестве основного (первичного).

Первичным ключом отношения называется такой вероятный ключ, по значениям которого производится контроль достоверности информации в отношении.

Применительно к экономической информации в подавляющем большинстве случаев отношения, полученные из существующих экономических документов, содержат единственный вероятный ключ, который является и первичным ключом. Это объясняется тем, что содержимое экономических документов понимается всеми пользователями одинаково. Далее будем иметь в виду только такие отношения. Наличие двух и более вероятных ключей в отношениях с осмысленной информацией можно объяснить наличием нескольких возможных способов интерпретации одних и тех же данных. Первичный ключ часто называется просто *ключ*.

В отношениях с большим числом строк нахождение первичного ключа путем непосредственного применения определения достаточно затруднено. Кроме того, на стадии проектирования ЭИС значения многих отношений просто неизвестны. Поэтому практически первичный ключ отношения определяется по известным функциональным зависимостям.

Каждое значение первичного ключа встречается только в одной строке отношения. Значение любого атрибута в этой строке также единственное. Если через К обозначить атрибуты первичного ключа в отношении R(A,B,C, ... ,J), то справедливы следующие функциональные зависимости К → А, К → В, К → С, ... , К → J. *Набор атрибутов первичного ключа функционально определяет любой атрибут отношения.* Обратное также верно: *если найдена группа атрибутов, которая функционально определяет все атрибуты отношения по отдельности, и эту группу нельзя сократить, то найден первичный ключ отношения.*

Вернемся к отношению T1 с функциональными зависимостями:

ФИО, Дата → Дисциплина,

ФИО, Дата → Преподаватель,

ФИО, Дата → Оценка.

Нетрудно установить, что

ФИО, Дата → ФИО,

ФИО, Дата → Дата

(в каждом сочетании значений ФИО, Дата значение ФИО встречается один раз). Следовательно, первичный ключ в отношении T1 составляют атрибуты ФИО, Дата и при поиске ключа не потребовались конкретные значения T1.

Знание ключа отношения позволяет устанавливать ряд функциональных зависимостей, например, в T3 ZEN → BIG, RAM, AST → BIG.

Для множества функциональных зависимостей существует ряд закономерностей, которые выражаются теоремами. Знание теорем позволяет из исходного множества функциональных зависимостей получать производные зависимости.

Отметим ряд известных теорем о функциональных зависимостях. Атрибуты, фигурирующие в каждой теореме, должны находиться в одном и том же отношении.

Теорема 1

A,B → A и A,B → B.

Доказательство основано на том, что в строке <a,b> для атрибутов A и B значение a (как и значение b) присутствует один раз.

Теорема 2

A → B и A → C тогда и только тогда, когда A → BC.

Рассмотрим произвольное значение a атрибута A. Если A → B и A → C, то im B(a) и im C(a) содержат по одному элементу. Предположим, что зависимость A → BC неверна и im BC(a) состоит из 2 или более элементов. Тогда либо im B(a), либо im C(a) должны содержать более одного элемента. Полученное противоречие доказывает зависимость A → BC.

Обратно, если A → BC, то im BC(a) содержит один элемент вида <b,c> для любого a. Зафиксируем некоторое значение a1. Значение b (как и значение c) встречается в сочетании с a1 только один раз, следовательно, справедливо A → B и A → C.

Теорема 3

Если A → B и B → C, то A → C.

Предположим, что зависимость A → C неверна и множество im C(a) содержит более одного элемента. Каждому значению a соответствует единственное значение b (в силу A → B), поэтому im C(b) содержит более одного элемента. Получилось противоречие с условием B → C, что и доказывает теорему.

Примечательно, что доказательства остальных теорем опираются на первые 3 теоремы, а не доказываются от противного.

Теорема 4

Если A → B, то AC → B (C произвольно).

Доказательство

AC → A (теорема 1), A → B (условие), следовательно, AC → B по теореме 3.

Теорема 5

Если A → B, то AC → BC (C произвольно).

Доказательство

AC → B (теорема 4), AC → C (теорема 1), следовательно, AC → BC по теореме 2.

Теорема 6

Если A → B и BC → D, то AC → D.

Доказательство

Из A → B следует AC → BC (теорема 5). BC → D (условие), поэтому AC → D по теореме 3.

Количество теорем, которые можно доказать в таком стиле, достаточно велико, но мы ограничимся указанными шестью.

Каждой функциональной зависимости вида A → B можно поставить в соответствие булевскую функцию вида AB′ (знак логического умножения пропущен, через ′ обозначено логи-

ческое отрицание). Множеству функциональных зависимостей соответствует дизъюнкция выражений, соответствующих каждой зависимости в отдельности.

Введем сокращенные обозначения:

A – ФИО, B – Дата, C – Дисциплина,

D – Преподаватель, E – Оценка.

Тогда для рассмотренных выше функциональных зависимостей

ФИО, Дата → Дисциплина,

ФИО, Дата → Преподаватель,

ФИО, Дата → Оценка

булевская функция будет выглядеть как

$ABC' + ABD' + ABE'$

(знаком + обозначено логическое сложение).

Для некоторого множества функциональных зависимостей F введем множество F~, называемое покрытием. Покрытие F~ содержит все функциональные зависимости, которые можно получить из множества F в результате применения теорем 1– 6 (включая и содержимое F). Одно и то же покрытие F~ может быть получено из различных множеств функциональных зависимостей. Среди таких множеств выделим множество с минимальным числом зависимостей и назовем его *минимальным покрытием* (базисом) множества зависимостей F. Иначе можно сказать, что минимальным покрытием называется множество функциональных зависимостей, из которого удалены все зависимости, являющиеся следствиями оставшихся зависимостей и теорем 1 – 6.

Непосредственное применение теорем к множеству F целесообразно только при небольшом количестве зависимостей. Теоретической основой методов поиска минимального покрытия служит аппарат булевских функций.

Можно рекомендовать следующие теоремы:

- если A → BC, то A →B и A →C,
- если A → B, то AD → B,
- если A → B и B → C, то A → C.

Зависимости, указанные в условии той или иной теоремы, остаются в списке функциональных зависимостей, а зависимости, указанные в заключении теоремы, удаляются.

Если заранее известно, что вероятный ключ в отношении только один, то его можно найти простым способом. Вероятный ключ (если он единственный, т.е. совпадает с первичным ключом) – это набор атрибутов, которые не встречаются в правых частях всех функциональных зависимостей. Иными словами, из полного списка атрибутов отношения необходимо вычеркнуть атрибуты, встречающиеся в правых частях всех функциональных зависимостей. Оставшиеся атрибуты образуют первичный ключ.

Последний вопрос, нуждающийся в рассмотрении, это существование в отношениях реляционной базы данных неопределенных (нулевых) значений. Безопасной является ситуация, когда неопределенные значения отсутствуют у атрибутов, входящих в левые части каких-либо функциональных зависимостей. Если это требование не соблюдается, то операция выборки может создавать результирующие отношения с неопределенными значениями в тех позициях, где их по смыслу ответа быть не должно, а операция соединения может давать полностью неправильные результаты.

2.2.2 Вторая и третья нормальные формы отношений

Отношение имеет вторую нормальную форму (2НФ), *если оно соответствует* 1НФ *и не содержит неполных функциональных зависимостей.*

Неполная функциональная зависимость – это две зависимости:

- вероятный ключ отношения функционально определяет некоторый неключевой атрибут,
- часть вероятного ключа функционально определяет этот же неключевой атрибут.

Отношение, не соответствующее 2НФ, характеризуется избыточностью хранимых данных.

Например:

T4			
Магазин	Изделие	Цена	План_1999_г.
Салют	М22	50	200
Салют	К14	40	100
АТЭ	М22	50	300
АТЭ	Т62	60	100

Функциональные зависимости отношения Т4:

(1) Магазин, Изделие → План_1999_г.,

(2) Изделие → Цена.

Вероятным ключом в Т4 являются атрибуты Магазин, Изделие. Для доказательства можно сослаться на функциональные зависимости:

Магазин, Изделие → План_1999_г.,

(3) Магазин, Изделие → Цена (теорема 4),

(4) Магазин, Изделие → Магазин (теорема 1),

(5) Магазин, Изделие → Изделие (теорема 1).

Зависимости (3) и (2) вместе образуют неполную функциональную зависимость, по этой причине отношение Т4 находится лишь в 1НФ, а не во 2НФ.

Избыточность иллюстрируется тем фактом, что цена изделия указывается столько раз, сколько магазинов продают это изделие (изделие М22 в Т4). Переход к 2НФ и соответственно устранение отмеченной избыточности данных связано с созданием двух отношений вместо исходного отношения Т4.

Т41=Т4[Магазин,Изделие, План_1999_г.],

Т42=Т4[Изделие,Цена].

Т41		
Магазин	Изделие	План_1999_г.
Салют	М22	200
Салют	К14	100
АТЭ	М22	300
АТЭ	Т62	100

Т42	
Изделие	Цена
М22	50
К14	40
Т62	60

Ключом в Т41 служат атрибуты Магазин, Изделие, в Т42 – Изделие, и легко определить, что оба отношения соответствуют требованиям 2НФ.

База данных находится в 2НФ, если все ее отношения находятся в 2НФ.

Отношение соответствует 3НФ, *если оно соответствует* 2НФ *и среди его атрибутов отсутствуют транзитивные функциональные зависимости* (ФЗ).

Транзитивная ФЗ – это две ФЗ:

- вероятный ключ отношения функционально определяет неключевой атрибут,
- этот атрибут функционально определяет другой неключевой атрибут.

Если К – ключ отношения, А, В – не ключевые атрибуты и К → А, А → В – справедливые ФЗ, то они являются транзитивными. Частный случай транзитивной ФЗ – неполная ФЗ, когда К = С,Е и К → Е, Е → А.

Рассмотрим **пример.**

Т5		
ФИО	Группа	Факультет
Петров	101	ЭИ
Федин	101	ЭИ
Алешин	102	ЭИ
Яшина	102	ЭИ

Функциональные зависимости:

(6) ФИО → Группа,

(7) Группа → Факультет,

(8) ФИО → Факультет.

Ключ отношения Т5 – ФИО. Зависимости (6) и (7) вместе образуют транзитивную ФЗ, поэтому Т5 находится в 2НФ, но не в 3НФ.

Избыточность данных в Т5 связана с тем, что принадлежность группы к факультету указывается столько раз, сколько студентов обучается в этой группе.

Переход от Т5 к отношениям в 3НФ дает следующие результаты:
Т51=Т5[ФИО,Группа],
Т52=Т5[Группа,Факультет].

Т51	
ФИО	Группа
Петров	101
Федин	101
Алешин	102
Яшина	102

Т52	
Группа	Факультет
101	ЭИ
102	ЭИ

Отношения Т51, Т52 получились двухатрибутными, поэтому нарушение требований 3НФ в них невозможно.

База данных находится в 3НФ, если все ее отношения находятся в 3НФ.

Приведенные примеры показывают, что отношения, в которых соблюдается одна ФЗ либо ни одной, будут соответствовать условиям 2НФ и 3НФ, так как неполная и транзитивная ФЗ представляют собой две зависимости. На этом принципе основан алгоритм получения отношений в 3НФ.

Исходными данными для алгоритма служит некоторый список атрибутов, охватывающий одно отношение, базу данных или ее часть. В любом случае предполагается (хотя бы теоретически) существование одного отношения с заданным списком атрибутов. В противном случае нельзя применять некоторые теоремы о ФЗ и нельзя гарантировать, что одна и та же ФЗ, например $A \rightarrow B$, справедливая в двух различных отношениях R и S, соответствует равным отношениям R[A,B] и S[A,B].

Алгоритм получения отношений в 3НФ обладает следующими свойствами:

- сохраняет все первоначальные функциональные зависимости, т.е. зависимость, справедливая в R, справедлива и в одном из производных отношений. Это гарантирует получение осмысленных отношений с легко интерпретируемой структурой,

- обеспечивает соединение без потерь, т.е. значения исходного отношения R могут быть восстановлены из проекций отношения R с помощью операции соединения,
- результат декомпозиции в 3НФ обычно содержит меньше значений атрибутов, чем исходное отношение R (происходит уменьшение избыточности).

Алгоритм нормализации (к 3НФ)

1. Получить исходное множество функциональных зависимостей для атрибутов рассматриваемой БД.

Если исходные функциональные зависимости не удается определить путем анализа смысловых характеристик атрибутов, приходится использовать перечисление и отбраковку допустимых вариантов функциональных зависимостей.

Рассматриваются все сочетания по два атрибута, и в каждом случае доказывается или отвергается функциональная зависимость. Затем рассматриваются сочетания:

- по три атрибута, где первые два могут функционально определять третий,
- по четыре атрибута, где первые три могут функционально определять четвертый и т.д.

Применение теорем о функциональных зависимостях позволяет сократить количество рассматриваемых вариантов. Практически перечисление вариантов заканчивается, когда сочетания атрибутов станут содержать первичный ключ.

2. Получить минимальное покрытие множества функциональных зависимостей. В минимальном покрытии должны отсутствовать зависимости, которые являются следствием оставшихся зависимостей по теоремам 1 – 6. В частности, требуется объединить функциональные зависимости с одинаковой левой частью в одну зависимость. Обозначим полученное минимальное покрытие функциональных зависимостей через

$$F = \{f1,...,fi,...,fk\}.$$

3. Определить первичный ключ отношения.

4. Для каждой функциональной зависимости fi создать проекцию исходного отношения Ri = R[Xi], где Xi – объединение атрибутов из левой и правой частей fi.

5. Если первичный ключ исходного отношения не вошел полностью ни в одну проекцию, то создать отдельное отношение из атрибутов ключа.

Для практического применения алгоритма нормализации до 3НФ существенны два вопроса:

- как учесть наличие взаимно-однозначных соответствий,
- как сократить объем перебора вариантов при первоначальном определении множества функциональных зависимостей.

Для взаимно-однозначных соответствий принято выделение старшего (по объему представляемого понятия) атрибута, который затем представляет все атрибуты взаимно-однозначного соответствия.

Для ускорения шага 1 алгоритма нормализации не рассматриваются такие зависимости, которые являются следствием уже найденных зависимостей и теорем 1 – 6, и используется ряд закономерностей об отсутствии функциональных зависимостей. Среди них – теорема:

Если $A \rightarrow B$ и $D \not\rightarrow C$, то $ABD \not\rightarrow C$

и правило, согласно которому функциональные зависимости с атрибутом-основанием в левой части не требуется рассматривать (в противном случае алгоритм станет создавать отношения, состоящие только из атрибутов-оснований и не имеющие экономического смысла).

Пример

Рассмотрим отношение со сведениями о научно-исследовательских работах. Список атрибутов: НИИ, Директор, Адрес, Отдел, Ксотр, Тема, Датанач, Датакон, Приор, Заказ, Обфин, Работа, Прод ФИО.

На шаге 1 будет выделен блок взаимно-однозначных соответствий на атрибутах: НИИ, Директор, Адрес. Далее атрибутом-представителем будем считать НИИ. Список функциональных зависимостей приводится ниже.

Список функциональных зависимостей

Отдел → НИИ,

Отдел → Директор,

Отдел → Ксотр,

Тема → Датанач,

Тема → Датакон,

Тема → Приор,

ФИО → НИИ,

ФИО → Директор,

ФИО → Отдел,

Тема, Заказ → Обфин,

Тема, Работа, ФИО → Прод.

Структура реляционной базы данных

R1(НИИ,Директор,Адрес),

R2(НИИ,Отдел,Ксотр),

R3(Тема,Датанач,Датакон,Приор),

R4(ФИО,Отдел),

R5(Тема,Заказ,Обфин),

R6(Тема,Работа,ФИО,Прод),

R7(Тема,Заказ,Работа,ФИО).

Отношение R7 содержит первичный ключ для исходного множества атрибутов, а отношения R1–R6 построены на основе зависимостей из минимального покрытия.

Вычислительная сложность каждого этапа приведения отношений к 3НФ различна, но наиболее длительным является получение исходного списка функциональных зависимостей. Количество проверяемых ФЗ является экспоненциальной функцией от количества рассматриваемых атрибутов.

Хотя на основе функциональных зависимостей можно установить более сильную нормальную форму Бойса–Кодда (БКНФ), практически она применяется редко. Во-первых, база данных в БКНФ не всегда существует, во-вторых, ее отличие от 3НФ обычно связано с наличием нескольких ключей в отношении, что в экономических приложениях встречается исключительно редко.

2.2.3
Ациклические базы данных

Ряд ограничений в предметной области и БД не может быть описан с помощью функциональных зависимостей, что приводит к необходимости рассмотрения новых типов зависимостей – многозначных и взаимных.

В отношении R(A,B,C) существует многозначная зависимость (МЗ) A →→ B, если для любого a, являющегося значением атрибута A

$$im(BC)a = im(B)a \circ im(C)a,$$

где ° – знак декартова произведения множеств.

Положение атрибутов B и C равноценно, поэтому одновременно справедливо A →→ C, т.е. многозначные зависимости всегда встречаются парами.

Отношение R(A,B,C) с многозначной зависимостью A →→ B содержит избыточную информацию, хотя и несколько другого рода, чем отношение в 1НФ. Оказывается, что отношения R1=R[A,B] и R2=R[A,C] вместе представляют всю информацию из R, хотя и имеют обычно меньший объем. Справедливо соотношение R=R1*R2, которое можно считать равноценным определению многозначной зависимости.

Пример

Рассмотрим отношение Z(Завод, Изделие, Компл, План), где Компл – название комплектующего изделия для данного Изделия, а План – план выпуска Изделий. Справедлива функциональная зависимость Завод, Изделие → План, и мы имеем следующие отношения в 3НФ:

Z1(Завод, Изделие, План),

Z2(Завод, Изделие, Компл).

Отношение Z1 содержит двухатрибутный ключ, и в нем не может быть многозначной зависимости. В Z2 существует МЗ вида Изделие →→ Завод (и, следовательно, Изделие →→ Компл),

поскольку каждое изделие комплектуется одним и тем же набором комплектующих изделий, на каком бы заводе оно ни производилось. Поэтому Z2 необходимо разделить на два отношения Z21(Завод, Изделие) и Z22(Изделие, Компл). Вся информация из Z21 содержится в Z1, поэтому окончательный список отношений состоит из Z1 и Z22.

Операция соединения позволяет получить Z=Z1*Z22.

В ряде случаев декомпозиция реляционной БД на основе многозначных зависимостей дает различные результаты при перестановках в списке многозначных зависимостей, что является серьезным недостатком. Поэтому мы рассмотрим специальный класс реляционных баз данных, названных *ациклическими*, для которых характерна однозначная декомпозиция на основе многозначных зависимостей и ряд других полезных (с точки зрения удобства обработки данных) свойств.

Для определения понятия ациклической схемы БД введем граф соединений на множестве отношений {S1,S2,...,Sk}. Вершинами графа соединений являются имена существующих отношений S1, S2,...,Sk. Дуга графа <Si,Sj> существует, если в структуре отношений Si и Sj имеются общие атрибуты. Обозначим их для определенности через A(i,j) и назовем *весом дуги*. Путь на графе соединений называется A-путем, если атрибут A содержится в структуре каждого отношения, лежащего на пути. В графе соединений требуется, чтобы для каждой пары отношений Si, Sj с общим атрибутом A(i,j) существовал A(i,j)-путь между Si и Sj.

Если граф можно превратить в дерево с помощью исключения некоторых дуг при сохранении названного требования, то база данных с отношениями {S1,S2,...,Sk} является ациклической.

Например, в графе соединений на рис. 2.1,а можно разорвать любую из дуг и превратить граф в дерево. По этой причине база данных, состоящая из отношений S1, S2, S3, является ациклической.

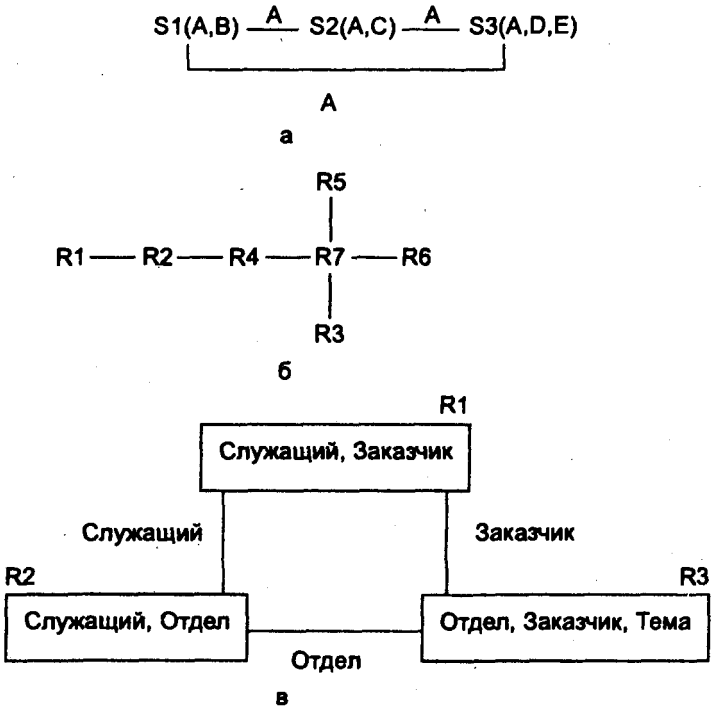


Рис. 2.1. Пример графа соединений:
а, б – ациклический; в – циклический

Рассмотрим алгоритм проверки структуры БД на ацикличность, использующий обычные сведения об атрибутном составе отношений БД.

Алгоритм проверки структуры БД на ацикличность

Исходные данные – список отношений с указанием атрибутного состава каждого отношения.

Метод

Шаг 1. Если некоторый атрибут встречается только в одном отношении, вычеркнуть данный атрибут из этого отношения.

Шаг 2. Если все атрибуты некоторого отношения находятся среди атрибутов другого отношения, то первое отношение вычеркивается из списка.

Шаги 1 и 2 можно применять в любой последовательности.

Если в результате будут вычеркнуты все отношения, то БД является ациклической. В обратном случае – БД циклическая.

Достаточно широкий класс ациклических реляционных БД можно охарактеризовать следующей теоремой.

База данных в третьей нормальной форме с единственным ключом БД и условием: *если ключ отношения* Si *не содержится в ключе* БД, *то это одноатрибутный ключ – является ациклической базой данных*.

Рассмотрим структуру реляционной базы данных в 3НФ, полученную в предыдущем пункте.

R1(НИИ#, Директор, Адрес),
R2(НИИ, Отдел#, Ксотр),
R3(Тема#, Датанач, Датакон, Приор),
R4(ФИО#, Отдел),
R5(Тема#, Заказ#, Обфин),
R6(Тема#, Работа#, ФИО#, Прод),
R7(Тема#, Заказ#, Работа#, ФИО#).

В схемах отношений атрибуты ключа отмечены знаком #. Отношения R1 и R2 имеют одноатрибутный ключ, ключи остальных отношений являются частью ключа БД из R7, следовательно, эта БД ациклическая. Дерево соединений приводится на рис. 2.1,б.

База данных с циклическим графом соединений может давать некорректные ответы на запросы из-за существования нескольких неравноценных путей доступа при реализации запросов. В частности, отношение с атрибутами, полученными путем объединения весов дуг, образующих цикл, может быть вычислено несколькими способами и давать разный результат. Для доказательства можно продемонстрировать наличие различающихся функциональных зависимостей в получаемых отношениях.

Рассмотрим простой **пример** графа соединений на рис.2.1.в.

Существование цикла очевидно. Отношение T(Служащий, Отдел, Заказчик) может быть вычислено либо как T=R1*R2, либо в виде:

T=(R1*R3)[Служащий, Отдел, Заказчик].

В первом случае будет справедлива функциональная зависимость Служащий → Отдел, а во втором случае – нет, так как с заказчиками могут работать служащие, не являющиеся сотрудниками каких-то отделов.

Восстановление свойств ацикличности БД может быть произведено двумя способами.

1. Добавление в БД нового отношения с атрибутами, равными объединению весов дуг, образующих цикл. В этом случае придется допустить существование неопределенных значений в новом отношении.

2. Добавление новых атрибутов, переименование и разделение атрибутов. Такое решение не требует дополнительных соглашений при интерпретации запросов и не создает дополнительные неопределенные значения.

Приведем типичные варианты разделения и переименования атрибутов.

1. При структуре функциональных зависимостей вида A → B → C, A → E → C необходимо установить две различные роли для атрибута C (например, C1 и C2).

Рассмотрим БД с атрибутами Студент, Группа, Куратор, Кафедра и зависимостями Студент → Группа → Кафедра, Студент → Куратор → Кафедра. В первом случае подразумевается скорее всего выпускающая кафедра для группы студентов, а во втором – кафедра, на которой работает куратор. После разделения атрибута Кафедра получаем структуру БД в виде:

P1(Студент, Группа),

P2(Группа, Выпускающая_кафедра),

P3(Студент, Куратор),

P4(Куратор, Кафедра_куратора).

2. Каждое применение теоремы псевдотранзитивности Т6 свидетельствует о цикличности БД.

> Рассмотрим, например, зависимости Служащий → Отдел и Отдел, Заказчик → Тема. Это ситуация, показанная на рис.2.1,в. Для преодоления цикличности необходимо разделить роли атрибута Отдел, например, ввести атрибут Отдел_служащего.

3. Структура функциональных зависимостей вида АВ → С, С → В часто получается, если действия, совершаемые объектом, приписываются классу объектов. Необходимо в таком случае добавить в структуру БД атрибут, обозначающий этот объект.

> Например, служащий фирмы покупает билет в аэропорту и счет направляется в отделение фирмы. Поэтому справедливы зависимости Аэропорт, Фирма → Отделение и Отделение → Фирма. Добавив атрибут Служащий, мы разрушаем нежелательную структуру функциональных зависимостей и получаем Аэропорт, Служащий → Отделение и Отделение → Фирма.

Следует отметить, что циклическая база данных может содержать два, три и более циклов. Описанный выше алгоритм фиксирует наличие циклов вообще. Поэтому когда циклическая БД преобразована рекомендуемыми выше способами, необходимо заново проверить ее на ацикличность.

Учитывая безусловно высокую трудоемкость приведения отношений к 3НФ при достаточно большом числе атрибутов (в качестве границы можно назвать 12–14 атрибутов), можно рекомендовать следующие действия при проектировании структуры реляционной БД:

- каждый входной документ привести к 3НФ и установить первичный ключ в каждом случае,
- для полученного на шаге 1 множества отношений построить граф соединений. Если граф соединений можно преобразовать в дерево соединений (или алгоритм проверки ацикличности, приведенный выше, дал положительный результат), то база

данных в целом является ациклической и соответствует 3НФ. В обратном случае для достижения ацикличности требуется выполнить преобразования, рекомендованные выше.

Критерии, которым соответствует база данных в 3НФ, и ациклическая БД, безусловно, не совпадают. В первую очередь ациклическая БД не гарантирует минимальную избыточность представления информации. Гарантии единственного пути доступа в ациклической БД, вероятно, следует признать более существенными для пользователей-непрофессионалов. Надо также учитывать элементарность метода проверки ацикличности БД в сравнении с необходимостью формального анализа функциональных зависимостей, требуемых при создании БД в 3НФ.

2.2.4 Доступ к реляционной базе данных

Реляционная база данных, в целом соответствующая третьей нормальной форме или обладающая ацикличностью, характеризуется рядом свойств, знание которых облегчает и упорядочивает процессы обработки данных.

В качестве типичных операций рассмотрим здесь выборку и корректировку информации.

Выборка позволяет получить некоторое подмножество значений атрибутов, хранящихся в БД.

Для моделирования процесса выборки данных из БД требуется провести классификацию возможных запросов. Обозначим через A – имя неключевого атрибута, v – одно из его значений, e – соответствующее значение первичного ключа и @ – один из знаков сравнения =,#,<,>,<=,>=. Когда величины e или v должны быть найдены, они обозначаются знаком "?". Мы приходим к трем классам запросов:

1. A(e) = ? – найти значение атрибута по известному значению ключа;

2. A(?) @ v – найти записи с заданным значением неключевого атрибута.

3. A(?) = ? – перечислить значения данного атрибута для каждой записи.

Условия запросов второго типа могут комбинироваться с помощью логических связок "не", "и", "или".

Введем понятия оболочки и входа запроса. *Оболочкой запроса* называется множество имен атрибутов, упоминаемых в формулировке запроса. *Входом запроса* называется множество имен атрибутов, для которых заданы условия вида A(?) @ v.

Рассмотрим пример запроса: какие предприятия, расположенные в Киеве, поставляют свою продукцию на завод ММЗ по железной дороге.

Оболочку запроса составляют атрибуты Предприятие (поставщик), Город (поставщика), Продукция, Завод (потребитель), Вид _транспорта. Входом запроса являются атрибуты Город, Завод, Вид _транспорта. Допустим, что запрос реализуется на отношении

R(Предприятие, Город, Продукция, Завод, Вид_транспорта)

с ключом Предприятие, Продукция, Завод. Атрибут Завод служит частью первичного ключа, следовательно, данный запрос относится к типу 2. В нашем примере выход запроса – Предприятие.

Формально будем записывать запрос в виде:

Выбрать Q, где T @ t.

Через T обозначены имена атрибутов входа запроса, t – их значения, Q – множество атрибутов на выходе запроса (часть оболочки).

Реализацию запросов к базе данных с помощью операторов реляционной алгебры можно описать следующими правилами.

1. В словесной формулировке запроса выделить имена атрибутов, составляющие оболочку, вход и выход запроса, а также условия выборки.

2. Зафиксировать множество атрибутов оболочки. Если все необходимые атрибуты находятся в каком-то одном отношении, то последующие операции выборки и проекции проводятся только с ним. Если требуемые атрибуты распределены по нескольким отношениям, то эти отношения необходимо

соединить. Каждая пара отношений соединяется по условию равенства атрибутов с совпадающими именами (или определенных на общем домене).После каждого соединения с помощью проекции можно отсечь ненужные для последующих операций атрибуты.

3. Полученное единственное отношение далее обрабатывается операциями выборки и проекции. Выборка по значениям атрибута должна предшествовать проекции, в которой этот атрибут выводится из отношения.

4. Если запрос можно разделить на части (подзапросы), то его реализация также разделяется на части, где результатом каждого подзапроса является отдельное отношение.

5. Указанная последовательность действий является стандартной, но, возможно, создает промежуточные отношения слишком большого размера. Этот недостаток можно компенсировать, выполняя некоторые выборки и проекции над исходными отношениями (до проведения соединения) и меняя взаимный порядок требуемых соединений.

Рассмотрим свойства соединений в базе данных с 3НФ и в ациклической БД.

База данных в 3НФ обладает свойством соединения без потерь, иными словами, отношения, полученные алгоритмом приведения БД к 3НФ, можно соединить в любом порядке и получить единственное корректное отношение, представляющее всю базу данных. Однако оболочка запроса обычно содержит небольшое число атрибутов, и такое большое отношение требуется крайне редко. Для соединения (по условию равенства атрибутов) двух произвольных отношений из базы данных в 3НФ справедлива следующая закономерность:

Соединение является корректным, если атрибут (или группа атрибутов), по которым производится соединение, служит первичным ключом хотя бы в одном из двух соединяемых отношений.

Для двух произвольных отношений R1,R2 рассмотрим X-значения из R1 и X-значения из R2, обозначив их R1.X и R2.X соответственно.

Введем 4 ролевые зависимости:

1. Равенство. Множество X-значений из R1 и множество X-значений из R2 совпадают в каждый момент времени.

2. Включение. Все значения R1.X содержатся в множестве значений R2.X.

3. Частичное включение. Множества значений R1.X и R2.X содержат общие элементы.

4. Непересечение. Множества R1.X и R2.X не содержат общих элементов.

Если соединяемые отношения R1 и R2 независимо друг от друга корректировались, то для корректности соединения необходимо, чтобы соблюдалось равенство или включение для R2.X, представляющего первичный ключ в названной выше закономерности.

В ациклической базе данных, кроме того, гарантируется, что при последовательных соединениях нескольких отношений число строк в промежуточных результатах соединений будет или монотонно убывать, или монотонно возрастать в зависимости от смысла запроса, в рамках которого производятся эти соединения.

Для оптимизации запроса к реляционной базе данных используются следующие преобразования:

- объединение операций выборки по нескольким условиям в одну операцию выборки с составным условием,
- если операция проекции применяется к результату выполнения пересечения, объединения, вычитания или натурального соединения двух отношений, то она может быть предварительно применена к каждому исходному отношению,
- если операция выборки применяется к результату выполнения пересечения, объединения, вычитания или натурального соединения двух отношений, то она может быть предварительно применена к каждому исходному отношению.

Пример

Рассмотрим систему отношений:

Деталь (Код_детали, Название, Вес),

Поставщик (Код_поставщика, Вид_транспорта, Город),

Перевозка (Код_поставщика, Код_потребителя, Код_детали, Дата, Количество).

Примерами запросов, иллюстрирующих названные выше правила, могут служить:

1. Получить коды всех поставляемых деталей.

Атрибут Код_детали встречается в отношениях Деталь и Перевозка. По смыслу запроса необходимо выполнить проекцию отношения Перевозка:

```
use Перевозка
copy to R1 field Код_детали
```

В отношении R1 значения кодов могут повторяться, и перед выводом ответа необходимо исключить эти повторы.

2. Выделить коды всех поставщиков из Киева, перевозящих детали автотранспортом.

Список атрибутов в формулировке запроса содержит Код_поставщика, Город и Вид_транспорта, находящихся в одном отношении Поставщик. Поэтому запрос реализуется следующими командами:

```
use Поставщик
copy to R2 for Город="Киев".and.Вид_транспорта="автотранспорт";
field Код_поставщика
```

3. Получить названия деталей, поставляемых заводом Динамо (код завода "174432").

Атрибуты запроса Название и Код_поставщика находятся в разных отношениях (Деталь и Перевозка), следовательно, потребуется их соединение по атрибуту Код_детали.

```
use Перевозка
set filter to Код_поставщика="174432"
select 2
use Деталь
join with Перевозка to R4;
for Код_детали = Перевозка->Код_детали field Название
```

4. Получить коды всех поставщиков, поставляющих детали с кодом "1425" или "7252".

Необходимые атрибуты Код_поставщика и Код_детали находятся в отношении Перевозка:

```
use Перевозка
copy to R3 for Код_детали="1425".or.Код_детали="7252";
field Код_поставщика
```

Если список деталей достаточно длинный, то целесообразно хранить его в некотором отношении, например, Z1(Код_детали) и применять команды:

```
select 1
use Перевозка
select 2
use Z1
join with Перевозка to R4;
for Код_детали = Перевозка->Код_детали field Код_поставщика
```

5. Получить коды всех поставщиков, поставляющих детали с кодом "1425" и "7252" одновременно.

Примечательно, что команды:

```
use Перевозка
copy to R5 for Код_детали="1425".and.Код_детали="7252";
field Код_поставщика
```

не формируют ответ на запрос (R5 будет пустым отношением). Можно рекомендовать, например, такие команды:

```
select 1
use Перевозка
copy to W1 for Код_детали="1425" field Код_поставщика
copy to W2 for Код_детали="7252" field Код_поставщика
use W1
select 2
use W2
join with W1 to R5 for Код_поставщика = W1->Код_поставщика
```

Если коды деталей находятся в некотором отношении, например в Z1, определенном в предыдущем примере, то необходимо применить операцию деления:

```
use Перевозка
copy to W1 field Код_поставщика, Код_детали
file1 = "W1"
file2 = "Z1"
file3 = "Y1"
attr = "Код_детали"
do divide with file1, file2, file3, attr
```

6. Получить коды поставщиков, поставляющих все детали.

Запрос разделяется на две части – получить список всех деталей (это результат первого запроса R1) и получить коды поставщиков (требуется деление):

```
use Перевозка
copy to W1 field Код_поставщика, Код_детали
file1 = "W1"
file2 = "R1"
file3 = "Y2"
attr = "Код_детали"
do divide with file1, file2, file3, attr
```

7. Для каждой поставляемой детали выбрать ее название и местонахождение поставщика.

Атрибуты запроса Название и Город находятся в отношениях Деталь и Поставщик. Соединить их невозможно (отсутствуют общие атрибуты), но при использовании отношения Перевозка задача решается:

```
select 1
use Перевозка
select 2
use Деталь
join with Перевозка to R4 for Код_детали = Перевозка->Код_
детали;
```

field Код_поставщика, Код_детали, Название

use Поставщик

select 1

use R4

join with Поставщик to R10 for Код_поставщика = Поставщик ->Код_поставщика;

field Код_детали, Название, Город

Определенные особенности реализации характерны для запросов, содержащих отрицание в условии запроса.

Пусть необходимо выделить фамилии программистов, не знающих языка Фортран (см. пример из п. 2.1). Попытка воспользоваться операцией выборки:

Y3	
ФИО	ЯП
Иванов	Си
Иванов	Паскаль
Петров	Си
Петров	Паскаль
Семин	Си
Яшин	Паскаль

use Y

copy to Y3 for ЯП # "Фортран"

приводят к отношению Y3, показанному выше,

т. е. в ответ попали все программисты. Задача решается путем разделения запроса на части – из списка всех программистов вычесть список программистов, знающих язык Фортран.

Для реализации запросов к реляционной БД необходимо хорошо представлять, как трансформируются функциональные зависимости исходных отношений в ФЗ для результирующих отношений.

Известен общий метод вывода функциональных зависимостей в отношении, полученном в результате соединения или выборки. Атрибуты исходных отношений последовательно нумеруются и функциональные зависимости, справедливые для них, переписываются с использованием числовых обозначений атрибутов. К этим зависимостям добавляется "числовой" эквивалент зависимости $X \leftrightarrow X$ (для операции соединения) или зависимость $0 \rightarrow B$, когда производится выборка по равенству значений атрибута B. В итоге мы получаем все функциональные зависимости, справедливые в результирующем отношении.

Пример

Обозначим в R1: первичный ключ X через 1, неключевые атрибуты – через 2, в R2: X через 3, первичный ключ как 3,4, неключевые атрибуты – 5. Для R1*R2 будут справедливы функциональные зависимости

$1 \rightarrow 2$; $3,4 \rightarrow 5$; $3 \rightarrow 1$; $1 \rightarrow 3$,

откуда следует $3,4 \rightarrow 2$. Таким образом, атрибуты 3,4 являются первичным ключом в R1*R2.

Многие СУБД содержат средства проверки уникальности первичного ключа, а не соблюдения функциональных зависимостей. СУБД семейства DBASE не проверяют даже уникальность значений ключа. Таким образом, контроль за соблюдением функциональных зависимостей в отношении после его корректировки – обязанность прикладных программистов или администратора БД.

При включении новой строки в отношение, соответствующее 3НФ, необходимо проверять отсутствие в отношении строк с тем же значением первичного ключа, как в новой строке. Иначе может произойти изменение списка атрибутов ключа в этом отношении, и база данных в целом потеряет свойства 3НФ. При исключении строки из отношения в 3НФ потери свойств 3НФ произойти не может.

2.3
СЕТЕВАЯ И ИЕРАРХИЧЕСКАЯ МОДЕЛИ ДАННЫХ

Информационными конструкциями в сетевой модели данных являются отношения и веерные отношения. Понятие "отношения" уже рассматривалось применительно к реляционной модели данных и будет использоваться здесь без изменений, хотя в некоторых сетевых СУБД допускаются отношения с многоуровневой (три и более) структурой.

Сетевая БД представляется как множество отношений и веерных отношений. Отношения разделяются на основные и зависимые.

Веерным отношением W(R,S) *называется пара отношений, состоящая из одного основного* R, *одного зависимого отношения* S *и связи между ними при условии, что каждое значение зависимого отношения связано с единственным значением основного отношения.*

Названное условие является ограничением, характерным для сетевой модели данных в целом. Способ реализации этого ограничения в памяти ЭВМ неодинаков у различных сетевых СУБД.

Допустимые в сетевой модели данных операции представляют собой различные варианты выборки.

Сетевые базы данных в зависимости от ограничений на вхождение отношений в веерные отношения разделяются на многоуровневые сети и двухуровневые сети.

Ограничение двухуровневых сетей состоит в том, что каждое отношение может существовать в одной из перечисленных ниже ролей:

- вне каких-либо веерных отношений,
- в качестве основного отношения в любом количестве веерных отношений,
- в качестве зависимого отношения в любом количестве веерных отношений.

Запрещается существование отношения в качестве основного в одном контексте и одновременно в качестве зависимого в другом контексте.

Многоуровневые сети не предусматривают никаких ограничений на взаимосвязь веерных отношений, в некоторых сетевых СУБД разрешены даже циклические структуры сети.

Среди существующих в настоящее время сетевых СУБД наиболее распространены системы, поддерживающие двухуровневую сеть. Операция связывания отношений в реляционных СУБД также приводит к двухуровневым системам отношений. Двухуровневые сети обладают свойством ацикличности, о котором будет сказано ниже, и по этой причине очень часто применяются разработчиками ЭИС и прикладными программистами.

Для двухуровневых сетевых СУБД вводятся еще два ограничения (с теоретической точки зрения необязательные):

- первичный ключ основного отношения может быть только одноатрибутным,
- веерное отношение существует, если первичный ключ основного отношения является частью первичного ключа зависимого отношения.

Организация веерного отношения в памяти ЭВМ

В структуру основного и зависимого отношений вводится дополнительный атрибут, называемый *адресом связи*. Значения адресов связи совместно обеспечивают в веерном отношении соответствие каждого значения зависимого отношения S с единственным значением основного отношения R.

Значение отношения при хранении в памяти ЭВМ часто называется *записью*. *Адресом связи* называется атрибут в составе записи, в котором хранится начальный адрес или номер следующей обрабатываемой записи.

Связь значений зависимого отношения с единственным значением основного отношения в простейшем случае обеспечивается следующим образом. Адрес связи некоторой запи-

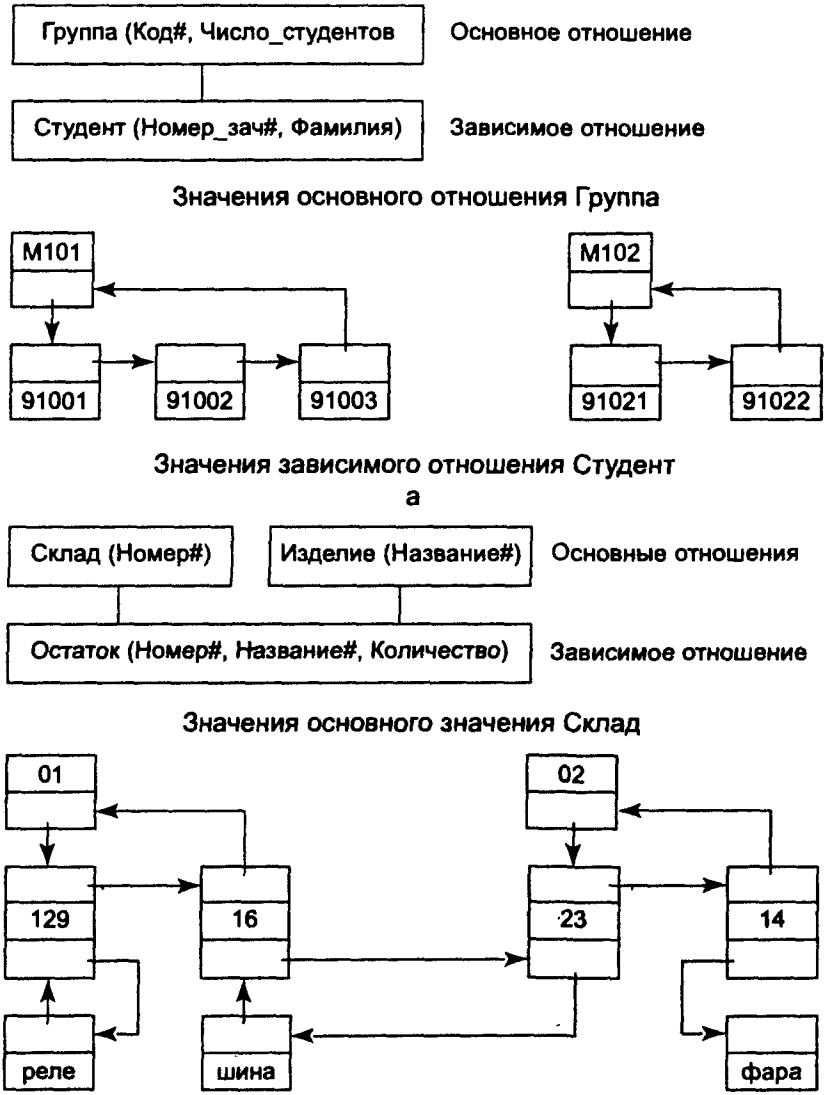


Рис. 2.2. Структуры и значения веерных отношений

си основного отношения указывает на одну из записей зависимого отношения (значением адреса связи основного отношения является начальный адрес этой записи зависимого отношения), адрес связи указанной записи зависимого отношения – на следующую запись зависимого отношения, связанную с той же записью основного отношения и т.д. Последняя запись зависимого отношения в этой цепочке адресует названную выше запись основного отношения.

Получается кольцевая структура адресов связи, называемая *веером*, где роль "ручки" веера играет запись основного отношения. На графических иллюстрациях адрес связи изображается стрелкой, направленной от адреса связи данной записи к той записи, чей начальный адрес (номер) служит значением этого адреса связи. На рис. 2.2 показаны структуры и значения веерных отношений двух простых сетевых двухуровневых БД. Атрибуты первичного ключа во всех случаях помечены #.

Схема сетевой БД содержит следующие компоненты:

S(net) = <A,R,WW,Dom,Rel,Net,V(s)>,

где WW – множество веерных отношений,
Net – вхождение отношений в веерные отношения.

Остальные элементы схемы аналогичны тем, которые введены выше для реляционных баз данных.

Существуют стандартные соглашения о способах включения и исключения данных в веерном отношении. Способ включения может характеризоваться как автоматический и неавтоматический.

Способ автоматический указывает, что при появлении нового значения основного отношения оно сразу же ставится в соответствие некоторому значению зависимого отношения и образует новый элемент веерного отношения. Несоблюдение этого правила характерно для способа неавтоматического.

Способ исключения может быть обязательный и необязательный. Способ обязательный означает, что после того, как значение включено в основное отношение, оно становится его

постоянным членом. Его можно обновлять, но нельзя удалять из отношения. Способ необязательный означает, что любое значение основного отношения можно удалить.

Из аналогии определений веерного отношения и функциональной зависимости следует утверждение: *если существует веерное отношение, то ключ зависимого отношения функционально определяет ключ основного отношения,* и наоборот, *если ключ одного отношения функционально определяет ключ второго отношения, то первое отношение может быть зависимым, а второе – основным в некотором веерном отношении.*

Для доказательства достаточно заметить, что в формулировке: *каждое значение зависимого отношения связано с единственным значением основного отношения* точным представителем значения отношения является значение его первичного ключа, и отсюда следует приведенная выше формулировка о функциональных зависимостях между ключами.

Указанный факт обычно используется для того, чтобы при наличии функциональной зависимости между первичными ключами двух отношений доказать корректность связывания этих отношений в веерное отношение.

В схеме сетевой БД отношения и веерные отношения часто трактуются как файлы и связи, что позволяет рассматривать сетевую структуру как множество файлов

F = {F1(X1),F2(X2),...,Fi(Xi),...,Fn(Xn)},

где Xi – атрибуты ключа в файле Fi.

Дополнительно вводится граф сетевой структуры B с вершинами {X1,X2,...,Xi,...,Xn}. Дуга <Xi,Xj> в графе B существует, если Xi является частью Xj и Fj[Xi] является подмножеством Fi. Последнее условие имеет тот же смысл, что и синтаксическое включение отношений в реляционной модели данных. Здесь предполагается, что ключ основного файла содержится в зависимом файле. Граф B аналогичен графу соединений для реляционной БД.

Введем определение сетевой ациклической базы данных DBA. *База данных* DBA *называется ациклической, если между любыми*

двумя вершинами на графе B *существует не более одного пути. Двухуровневые сети всегда ациклические.*

Для множества файлов F ациклической базы данных DBA вполне применима операция

m(DBA) = F1 & F2 & ... & Fi & ...& Fn,

называемая *максимальным пересечением*. Ее аналогом может служить последовательность соединений в реляционной БД.

Рассмотрим алгоритм формирования структуры двухуровневой сетевой БД на основе известного множества атрибутов и функциональных зависимостей.

Исходное множество функциональных зависимостей и атрибуты первичного ключа получаются так же, как при формировании множества отношений в 3НФ. Алгоритм иллюстрируется тем же примером, что и в п. 2.2.2.

Алгоритм получения двухуровневой структуры сети

1. Для каждой функциональной зависимости вида A → B создается файл Fi(A,B). Каждый блок взаимно-однозначных соответствий также порождает файл с ключом, равным старшему по объему понятия атрибуту.

В нашем **примере** будут созданы следующие файлы (ключи помечены знаком #):

F1(НИИ #, Директор, Адрес),
F2(Отдел #, НИИ, Ксотр),
F3(Тема #, Датанач, Датакон, Приор),
F4(ФИО #, Отдел),
F5(Тема #, Работа #, ФИО #, Прод),
F6(Тема #, Заказ #, Обфин).

2. У всех пар файлов, полученных на шаге 1, проверяется условие для ключей (Ki является частью Kj). Если оно соблю-

дается, то из соответствующих файлов создается веерное отношение Wij(Fi,Fj).

В нашем **примере** получим

W35(F3,F5), W45(F4,F5), W36(F3,F6).

3. Если на шаге 2 будут получены два веерных отношения Wij и Wjk, то все атрибуты файла Fi передаются в файл Fj, и Fi вместе с Wij уничтожаются.

В нашем примере таких веерных отношений нет.

4. Атрибуты, не вошедшие в состав веерных отношений на шаге 2, добавляются в те файлы Fn (и содержащие Fn веерные отношения), где они будут неключевыми. При наличии нескольких подходящих файлов предпочтение отдается основным файлам. Если требуемые Fn отсутствуют, то создается новый файл из атрибутов первичного ключа, и повторяются шаги 2, 3, 4.

В нашем **примере** F4 расширяется атрибутами НИИ, Директор, Адрес, Ксотр.

На рис.2.3 показана структура соответствующей двухуровневой БД.

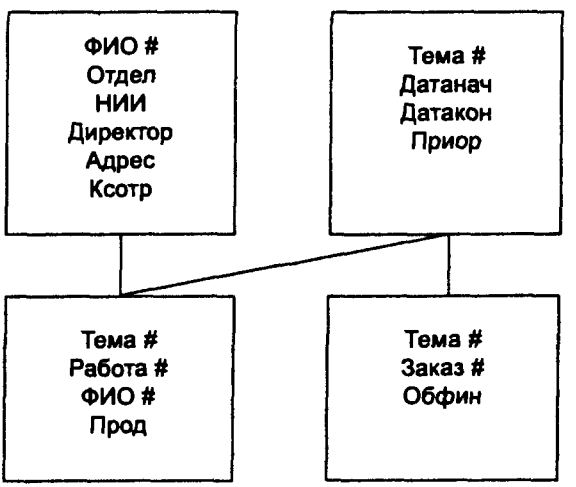


Рис. 2.3. Сетевая БД со сведениями о научно-исследовательских работах

Структуры основных отношений показаны в верхней части рисунка, а структуры зависимых отношений – внизу.

Перед рассмотрением операций в сетевой базе данных следует отметить, что существуют 2 различных подхода к обработке данных средствами СУБД.

1. Применение базового языка программирования. При этом подходе для манипулирования данными в СУБД разрабатывается универсальный язык программирования, обеспечивающий обращения к базе данных, работу с переменными и остальные возможности. Примером СУБД с базовым языком является dBASE.

2. Применение включающего языка программирования. Включающий язык – это обычный язык программирования (например, Паскаль), в котором обращения к базе данных реализуются с помощью подпрограмм. Среди параметров, передаваемых подпрограмме, указываются название операции, имена обрабатываемых отношений и др. Включающий язык используется практически во всех известных сетевых СУБД. Это объясняется принципом доступа к данным в сетевой базе данных, который может быть назван *навигационным*.

Центральным для навигационного принципа доступа является понятие "текущая запись" в отношениях базы данных. Текущей записью в отношениях после выполнения некоторой операции является значение отношения, на котором операция завершилась. Следующая операция начинается с этой текущей записи, а в результате выполнения операции положение текущей записи изменяется (завершение операции может изменить положение текущей записи и в других отношениях).

Рассмотрим операции выборки для двухуровневой сетевой базы данных. Чтобы не пользоваться синтаксисом включающего языка, условимся записывать лишь название операции и условие выборки. Примеры выборки относятся к сетевой структуре, изображенной на рис.2.4. В этой базе данных на основном отношении Сотрудник и зависимом Зарплата установлены два веерных отношения Осн – основная зарплата и Доп – дополнительная зарплата.

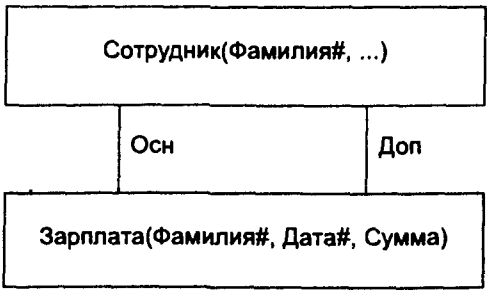


Рис. 2.4. Пример сетевой базы данных для операций выборки

1. Операция OBTNM – получить запись в основном отношении.

OBTNM (Сотрудник * "Иванов")

После выполнения указанной выборки в отношении Сотрудник в качестве текущей будет установлена запись со значением первичного ключа "Иванов". Затем атрибуты этой текущей записи обрабатываются средствами включающего языка (становятся значениями переменных, печатаются и т.п.).

2. Операция OBTNF – получить запись в зависимом отношении.

OBTNF(Сотрудник * "Иванов", Зарпл*Осн)

При выборке в зависимом отношении текущей записью становится следующая запись зависимого отношения относительно той, которая раньше была текущей в зависимом отношении. Условие выборки содержит указание на текущую запись в основном отношении, а также на имя зависимого отношения и имя веерного отношения.

В результате выполнения двух операторов
OBTNM (Сотрудник * "Иванов")
OBTNF(Сотрудник * "Иванов", Зарпл*Осн)

при условии, что обращения к отношению Зарпл ранее не производились, будут получены сведения о первой (с момента поступления на работу) основной зарплате Иванова.

Средствами включающего языка может быть организован циклический процесс, и тогда возможны более сложные варианты доступа.

Рассмотрим, например, последовательность операций

```
OBTNM (Сотрудник * "Иванов")
M1: OBTNF(Сотрудник * "Иванов", Зарпл*Осн)
print Зарпл
goto M1
```

Оператор print печатает все атрибуты текущей записи отношения Зарпл. На первый взгляд использование безусловного перехода создает зацикливание при выполнении. Однако операторы выборки в сетевых СУБД по окончании выборки вырабатывают код возврата, и выход за последнюю запись в отношении Зарпл сопровождается специальным значением этого кода в команде OBTNF, означающим неудачное завершение выборки. Значение кода возврата всегда проверяется средствами включающего языка, и этим обеспечивается выход из цикла. Практически приведенная выше последовательность операций напечатает все сведения о получении Ивановым основной заработной платы.

В сетевых СУБД количество операций выборки достаточно велико. Мы рассмотрели минимально необходимое множество вариантов выборки. Остальные варианты выборки создают более удобные для прикладного программиста возможности реализации запросов.

Рассмотрим реализацию в сетевых СУБД функций, аналогичных операциям проекции и соединения для реляционных систем. Оказывается, что аналог операции проекции для сетевой СУБД не нужен, так как соответствующие функции выполняет описание подсхемы сетевой базы данных. *Схемой сетевой* БД называется описание всех отношений с указанием атрибутного состава и ключей каждого отношения, а также

веерных отношений. В прикладной программе имеется возможность объявить часть отношений сетевой базы данных, в каждом отношении – некоторое подмножество атрибутов (с обязательным оставлением атрибутов-ключей) и лишь некоторые веерные отношения. Соответствующее описание данных называется *подсхемой*. Отношения, веерные отношения и атрибуты, не указанные в подсхеме, становятся недоступными прикладной программе. В отличие от операции проекции база данных, соответствующая подсхеме, не создается физически, а происходит ограничение доступа к исходной БД, которая определена в схеме.

Аналог операции соединения в сетевой СУБД также не нужен, но по другой причине. Дело в том, что результаты допустимых соединений фактически зафиксированы в сетевой СУБД с помощью цепочек адресов связи. Доступ к результатам возможного соединения начинается от некоторого основного отношения к вееру значений в соответствующем зависимом отношении, достигаемые при этом значения ключей в зависимом отношении запоминаются и используются для поиска в каком-то другом основном отношении, от этого основного отношения возможен переход к новому зависимому и т.д.

Иерархическая модель данных

Иерархическая модель данных имеет много общих черт с сетевой моделью данных, хронологически она появилась даже раньше, чем сетевая. *Допустимыми информационными конструкциями* в иерархической модели данных являются *отношение*, *веерное отношение* и *иерархическая база данных*. В отличие от ранее рассмотренных моделей данных, где предполагалось, что информационным отображением одной предметной области является одна база данных, в иерархической модели данных допускается отображение одной предметной области в несколько иерархических баз данных.

Понятия отношения и веерного отношения в иерархической модели данных не изменяются.

Иерархической базой данных называется множество отношений и веерных отношений, для которых соблюдаются два ограничения.

1. Существует единственное отношение, называемое корневым, которое не является зависимым ни в одном веерном отношении.

2. Все остальные отношения (за исключением корневого) являются зависимыми отношениями только в одном веерном отношении.

Схема иерархической БД по составу компонентов идентична сетевой базе данных. Названные выше ограничения поддерживаются иерархическими СУБД.

На рис. 2.5 изображена структура иерархической базы данных о студентах и преподавателях вуза, удовлетворяющая всем ограничениям, указанным в определении.

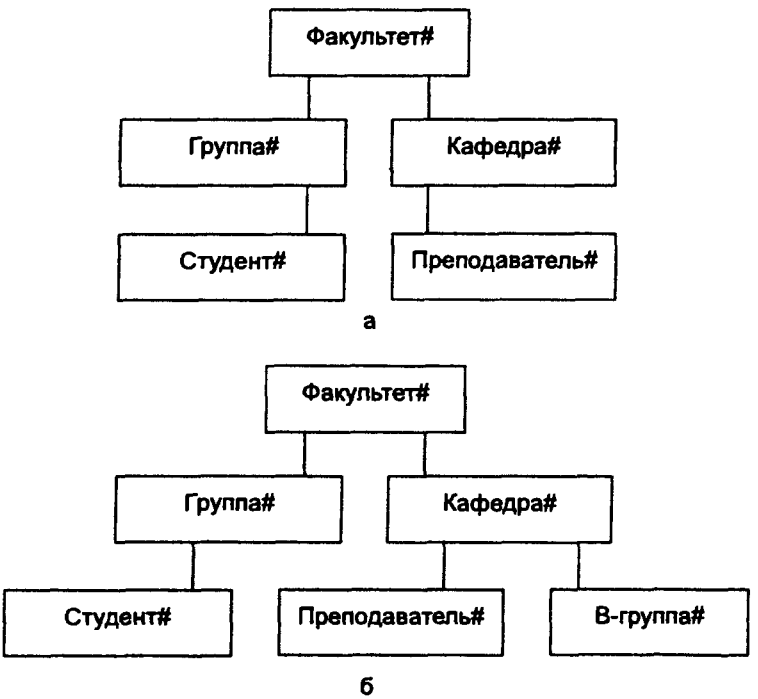


Рис. 2.5. Иерархическая база данных для вуза:
а — исходная структура;
б — с добавленными сведениями о группах дипломников

В графических иллюстрациях структуры приводятся ключи соответствующих отношений.

Если понадобится в рамках данной иерархической структуры указать для групп, выполняющих дипломное проектирование, связь с соответствующей выпускающей кафедрой, то установить веерное отношение Р(Кафедра,Группа) невозможно, так как Группа не может быть зависимым отношением дважды (она уже является зависимой для отношения Факультет). Зафиксировать связь студенческих групп с выпускающей кафедрой можно путем выделения соответствующих групп в отдельное отношение с ключом В-группа, что приводит к появлению избыточной информации.

Ограничение, которое поддерживается в иерархической модели данных, состоит в невозможности нарушения требований, фигурирующих в определении иерархической базы данных. Это ограничение обеспечивается специальной укладкой значений отношений в памяти ЭВМ. Ниже мы рассмотрим одну из простейших реализаций укладки иерархической БД.

На рис. 2.6 приводится связь значений отношений из иерархической базы данных, структура которой показана на рис. 2.5,а. Каждое значение представляется соответствующей величиной первичного ключа.

Далее эта информация организуется в линейную последовательность значений, как это показано на рис. 2.6,б.

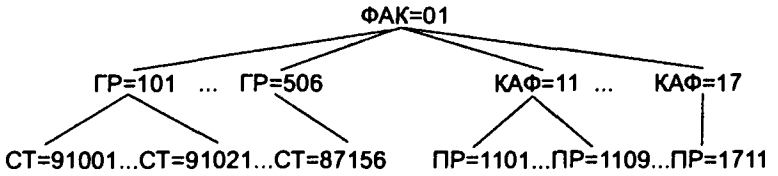
Необходимо отметить, что существуют различные возможности прохождения иерархически организованных значений в линейной последовательности. Принцип, применяемый для иерархических баз данных, называется концевым прохождением.

Правила концевого прохождения

1. Начиная с первого значения корневого отношения, перечисляются первые значения соответствующих отношений на каждом уровне вплоть до последнего.

2.Перечисляются все значения в том веерном отношении, на котором остановился шаг 1.

3. Перечисляются значения всех вееров этого веерного отношения.



а

Запись 1

ФАК=01	ГР=101	СТ=91001	СТ=91002	...	СТ=91021	ГР=102	СТ=91022

...	ГР=506	...	СТ=87156	КАФ=11	ПР=1101	ПР=1102	...	ПР=1109

КАФ=12	ПР=1201	...	КАФ=17	...	ПР=1711

Запись 2

ФАК=02	ГР=110	СТ=91188	СТ=91021	ГР=111	...

б

Рис. 2.6. Линейное представление значений в иерархической базе данных:
а — иерархическая взаимосвязь значений;
б — линейное представление данных

4. От достигнутого уровня происходит подъем на предыдущий уровень, и если возможно применить шаг 1, то процесс повторяется.

Записью иерархической базы данных называется множество значений, содержащих одно значение корневого отношения и все вееры, доступные от него в соответствии со структурой иерархической базы данных. В нашем примере одну запись образуют данные, относящиеся к одному факультету.

Для веерных отношений в составе иерархической базы данных справедлива уже известная закономерность: *если существует веерное отношение, то ключ зависимого отношения функционально определяет ключ основного отношения, и наоборот, если*

ключ одного отношения функционально определяет ключ второго отношения, то первое отношение может быть зависимым, а второе – основным в некотором веерном отношении.

Кроме того, ограничение на существование единственного корневого отношения в иерархической базе данных трансформируется в требование: *первичный ключ каждого некорневого отношения должен функционально определять первичный ключ корневого отношения.*

Рассмотрим алгоритм формирования иерархической БД на основе известного множества атрибутов и функциональных зависимостей. Исходное множество функциональных зависимостей и атрибуты первичного ключа получаются так же, как при формировании множества отношений в 3НФ. Алгоритм иллюстрируется тем же примером, что и в п. 2.2.2.

Алгоритм получения структуры иерархической БД

1. Для каждой функциональной зависимости вида A → B создается отношение Si(A,B). Каждый блок взаимно-однозначных соответствий также порождает отношение с ключом, равным старшему по объему понятия атрибуту.

В нашем примере будут созданы следующие отношения (ключи помечены знаком #):

S1(НИИ #, Директор, Адрес),
S2(Отдел #, НИИ, Ксотр),
S3(Тема #, Датанач, Датакон, Приор),
S4(ФИО #, Отдел),
S5(Тема #, Работа #, ФИО #, Прод),
S6(Тема #, Заказ #, Обфин).

2. Разделить отношения на группы по признаку: два отношения находятся в общей группе, если их ключи функционально определяют хотя бы один общий атрибут.

Для отношений S1 – S6 получаем две группы:

- S1, S2, S4, S5 (все ключи функционально определяют атрибут НИИ);
- S3, S6, S5 (все ключи функционально определяют атрибут ТЕМА).

Далее шаги 3, 4, 5 выполняются раздельно для каждой группы. Количество групп определяет количество иерархических БД.

3. У всех пар отношений группы проверяется условие для ключей Kj → Ki. Если оно соблюдается, то из соответствующих отношений создается веерное отношение Wij(Si,Sj).

4. Найти в группе цепи веерных отношений и сцепить их в дерево. Элемент цепи образуется по условию Wij – Wjk.

В нашем примере получим:

группа 1: цепь W12(S1,S2), W24(S2,S4), W45(S4,S5) образует дерево;

группа 2: цепей нет, но W35(S3,S5) и W36(S3,S6) образуют дерево.

5. Атрибуты, оставшиеся вне цепей на шаге 4, добавить в структуру тех отношений, где они будут неключевыми, либо в структуру отношений, соответствующих висячим вершинам дерева.

6. Если группы, полученные на шаге 2, содержат общие отношения, то решить вопрос о целесообразности установления логических связей между иерархическими БД.

7. Сократить список атрибутов в сегментах за счет удаления атрибутов зависимого отношения, общих в паре "основной – зависимый".

Итоговая иерархическая структура содержит две иерархические базы данных. В некоторых иерархических СУБД не допускается логическая связь баз данных, определенная в п.6 алгоритма, так как формально это является нарушением ограничения 2 иерархической модели данных.

Манипулирование иерархической базой данных происходит с использованием включающего языка. Подпрограмма обращения к базе данных содержит ряд параметров, из которых мы будем использовать только код требуемой операции

и одно или несколько условий выбора. По причинам, которые уже были названы при изучении сетевой модели данных, для иерархической модели данных необходимы только операции выборки.

Минимальное множество вариантов выборки соответствует трем операциям.

1. GU – получить уникальную запись по известным значениям первичного ключа на каждом уровне дерева иерархической базы данных.

2. GN – получить следующую запись на том уровне дерева, где находится текущая запись после выполнения оператора GU.

3. GNP – получить следующую запись на расположенном непосредственно ниже уровня дерева относительно позиции, где находится текущая запись после выполнения оператора GU или GN.

Например, для запроса – получить информацию о преподавателе с кодом 1103 кафедры 11 факультета 01 – потребуется оператор

```
GU Факультет (Факультет="01")
  Кафедра (Кафедра="11")
  Преподаватель (Преподаватель="1103")
print Преподаватель
```

В этом примере названия отношений совпадают с названиями соответствующих первичных ключей.

Запрос – получить список всех студентов группы 102 – реализуется следующей последовательностью операторов

```
GU Факультет (Факультет="01")
  Группа (Группа="102")
  Студент
M1: GN Студент
print Студент
goto M1
```

Поскольку в операторе GU не указано условие выборки в отношении Студент, текущей записью станет первая запись этого отношения, и далее циклическое повторение оператора GN обеспечит требуемое извлечение всех записей о студентах группы 102. Выход из цикла произойдет в результате получения кода возврата "конец отношения".

Сравнение моделей данных

При сравнении моделей данных очень трудно отделить факторы, характеризующие принципиальные особенности модели, от факторов, связанных с реализацией этих моделей данных средствами конкретных СУБД.

Рассматривая преимущества и недостатки известных моделей данных, следует отметить ряд несомненных *достоинств реляционного подхода*:

- Простота. В реляционной модели всего одна информационная конструкция, которая формализует табличное представление данных, привычное для пользователей-экономистов.
- Теоретическое обоснование. Наличие теоретически обоснованных методов нормализации отношений и проверки ацикличности структуры позволяет получать базы данных с заданными характеристиками.
- Независимость данных. Когда необходимо изменить структуру реляционной БД, это, как правило, приводит к минимальным изменениям в прикладных программах.

Среди *недостатков реляционной модели* данных необходимо назвать следующие.

- Низкая скорость при выполнении операции соединения.
- Большой расход памяти для представления реляционной БД. Хотя проектирование в 3НФ рассчитано на минимальную избыточность (каждый факт представляется в БД один раз), другие модели данных обеспечивают меньший расход памяти для представления тех же фактов. Например, длина адреса связи обычно намного меньше, чем длина значения атрибута.

Достоинствами иерархической модели данных являются следующие.

- Простота. Хотя модель использует три информационные конструкции, иерархический принцип соподчиненности понятий является естественным для многих экономических задач (например, организация статистической отчетности).
- Минимальный расход памяти. Для задач, допускающих реализацию с помощью любой из трех моделей данных, иерархическая модель позволяет получить представление с минимально требуемой памятью.

Недостатки иерархической модели.

- Неуниверсальность. Многие важные варианты взаимосвязи данных невозможно реализовать средствами иерархической модели, или реализация связана с повышением избыточности в базе данных.
- Допустимость только навигационного принципа доступа к данным.
- Доступ к данным производится только через корневое отношение.

Необходимо отметить следующие *преимущества сетевой модели данных.*

- Универсальность. Выразительные возможности сетевой модели данных являются наиболее обширными в сравнении с остальными моделями.
- Возможность доступа к данным через значения нескольких отношений (например, через любые основные отношения).

В качестве *недостатков сетевой модели* данных можно назвать.

- Сложность, т.е. обилие понятий, вариантов их взаимосвязей и особенностей реализации.
- Допустимость только навигационного принципа доступа к данным.

Результаты, полученные для ациклических баз данных, позволяют говорить о равноценных возможностях представления информации у ациклических реляционных БД, двухуровневых сетевых БД и иерархической БД без логических связей.

При анализе моделей данных не затрагивалась проблема упорядоченности значений в отношениях баз данных. Для реляционной модели данных эта упорядоченность с теоретической точки зрения необязательна, а в двух других моделях она широко используется для повышения эффективности реализации запросов.

На окончательный выбор модели данных влияют многие дополнительные факторы, например, наличие хорошо зарекомендовавших себя СУБД, квалификация прикладных программистов, размер базы данных и др.

В последнее время реляционные СУБД заняли преимущественное положение как средство разработки ЭИС. Недостатки реляционной модели компенсируются ростом быстродействия и ресурсов памяти современных ЭВМ. Вследствие процессов децентрализации управления в экономике многие базы данных ЭИС имеют простую структуру, которая легко трансформируется в понятные системы таблиц (отношений).

2.4
МОДЕЛЬ ИНВЕРТИРОВАННЫХ ФАЙЛОВ И ИНФОРМАЦИОННО-ПОИСКОВЫЕ СИСТЕМЫ

Модель инвертированных файлов можно рассматривать как частный случай сетевой двухуровневой модели данных. Произведенные упрощения двухуровневой сети позволили создать еще более понятную прикладным программистам и пользователям модель данных.

Основными информационными конструкциями в модели инвертированных файлов являются основной файл, который соответствует ранее введенному понятию "отношения", "инвертированный файл" и "список связи".

Множество основных файлов базы данных обозначим через {F1,F2,...,Fn}. Все записи файлов F1...Fn получают в пределах базы данных единую нумерацию.

В основном файле Fi разрешается выделить один или несколько атрибутов, по значениям которых затем будут формироваться инвертированные файлы и списки связи. С точки зрения ранее рассмотренной реляционной модели данных выделяемый атрибут может быть как первичным, так и вторичным ключом в основном файле Fi.

Естественно, что выделенный атрибут (обозначим его A) может принимать в Fi несколько различных значений {a(1),a(2),...,a(k)}.

Поставим в соответствие каждому значению a(j) множество номеров записей файла Fi, в которых это значение связано с именем атрибута A.

{a(1),n(t),n(p),...}
{a(2),n(g),n(h),...}
.....
{a(j),n(x),n(y),...}
.....

Через n с соответствующим индексом обозначены номера записей из Fi.

Определенная таким образом последовательность значений атрибута A и номеров записей основного файла Fi является инвертированным файлом, который далее будем обозначать через A(Fi).

Чтобы определить понятие списка связи, необходимо отметить, что единая нумерация всех записей базы данных приводит к тому, что номер записи становится первичным ключом во всех основных файлах базы данных независимо от того, какие атрибуты образуют ключ в каждом из этих файлов.

Рассмотрим два файла Fi и Fm, в структуре которых имеется общий атрибут A. В этой ситуации существуют два списка связи (Fi,Fm) и (Fm,Fi). В списке (Fi,Fm) для каждого номера записи из файла Fi указываются номера записей из файла Fm, имеющие то же самое значение атрибута A. Аналогично определяется содержимое списка связи (Fm,Fi).

Аналогия с двухуровневой сетью заключается в следующем. Связь инвертированного файла A(Fi) и файла Fi соответствует типу "основной–зависимый". Отличия сводятся к тому, что

атрибут A не имеет никакого отношения к первичному ключу Fi (в двухуровневой сети он должен быть частью первичного ключа), и, кроме того, вместе с атрибутом A в инвертированном файле запрещено хранить значения других атрибутов.

Пример

База данных содержит основные файлы Сотрудники и Зарплата, показанные на рис. 2.7. Естественно, что списки связи установлены по атрибуту Фамилия, а инвертированных списков в нашем примере максимально может быть пять (по числу атрибутов в основных файлах).

Сотрудники	
Фамилия	Должность
01 Котов	инженер
02 Яшина	технолог
03 Седов	технолог
04 Рогов	инженер

Зарплата		
Фамилия	Дата	Зарплата
05 Яшина	10.01.98	500
06 Седов	20.03.98	400
07 Яшина	20.03.98	500
08 Котов	20.03.98	600
09 Рогов	10.04.98	400
10 Котов	10.04.98	300
11 Яшина	10.05.98	400

Инвертированный список Должность (Сотрудники)
инженер — 01, 04
технолог — 02, 03

Список связи (Сотрудники, Зарплата)
01 — 08, 10
02 — 05, 07, 11
03 — 06
04 — 09

Список связи (Зарплата, Сотрудники)
05 — 02
06 — 03
07 — 02
08 — 01
09 — 04
10 — 01
11 — 02

Рис. 2.7. База данных

Преимущества модели инвертированных файлов особенно проявляются при реализации выборки с большим количеством условий. Каждое условие выборки соответствует множеству номеров записей, и комбинация условий выборки означает манипулирование ранее полученными из инвертированных файлов множествами номеров записей.

В информационно-поисковых системах ключевые атрибуты соответствуют ключевым словам, определяющим тематику документа. Количество ключевых слов для документа может быть любым. Связь основного и инвертированного файла в этом случае выглядит иначе и показана на рис. 2.8.

Пусть дан запрос: найти все документы, содержащие ключевые слова A и C. Система обратится к инвертированному файлу и найдет группы ключей A и C. Совпадающие значения номеров укажут в нашем примере на искомую запись с номером 140.

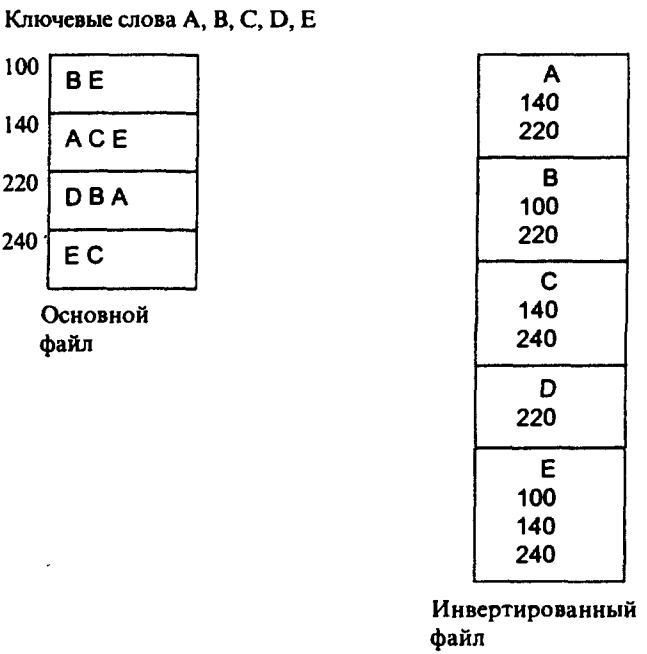


Рис. 2.8. Связь основного и инвертированного файла

Логические связки в запросах могут быть любыми, и с математической точки зрения требуемые поисковые операции есть операции пересечения, объединения, вычитания над множествами номеров записей, которые хранятся в инвертированных массивах для атрибутов, названных в запросе.

Так, при обработке запроса – найти все записи, содержащие ключи E или C, кроме A, которому в терминах теории множеств соответствует запись (E ∪ C)\ A, производится последовательное вычисление

$$(\{ 100, 140, 240\} \cup \{ 140, 240\}) \setminus \{ 140, 220\} = \{ 100, 240\}.$$

Результат означает, что запросу удовлетворяют записи с номерами 100 и 240.

Следует отметить, что поиск по инвертированному файлу обнаруживает только номера записей и плохо приспособлен для указания всех ключей, связанных с найденной записью. Между тем эта информация часто запрашивается. В одном из наших примеров запись с адресом 140 была найдена по значениям ключей A и C очень быстро, но определить, есть ли в этой записи третий ключ E, используя только инвертированный файл, очень трудно.

Модель инвертированных файлов служит основой для ряда современных информационно-поисковых систем. Одна база данных создается обычно для одного класса документов, которые объединены общей тематикой, например справочная информация о предприятиях и организациях, сведения о производимой продукции, информация о происходящих выставках.

С учетом реляционного подхода одна база данных в таком случае соответствует одному отношению. Однако компоненты этого отношения имеют ряд важных особенностей.

1. Значением атрибута может быть текст произвольных размеров, причем разбиение этого текста на строки может варьироваться и не должно влиять на реализацию поисковых запросов.

2. Документ может сопровождаться графической иллюстрацией, и в таком случае ее предоставление определяется средствами компьютерной графики. Изображение не является значением какого-то атрибута в общепринятом смысле слова, поскольку как значение не может участвовать в операциях выборки.

Из всего многообразия реализаций информационно-поисковых языков модели инвертированных файлов соответствуют дескрипторные языки.

Дескриптором, или ключевым словом, называется слово или словосочетание, используемое для краткого обозначения темы документа, хранящегося в базе данных информационно-поисковой системы. Конкретный документ может сопровождаться несколькими дескрипторами в зависимости от количества характеризующих его терминов.

Получение списка дескрипторов для каждого конкретного документа является достаточно сложной и трудоемкой задачей, которую обычно решают специалисты в той области знаний, которой посвящена информационно-поисковая система. Один из более простых подходов к определению списка дескрипторов для всех документов в базе данных заключается в том, что из всех атрибутов документа выбирается несколько наиболее информативных, и все слова, составляющие значения таких атрибутов, переносятся в список дескрипторов. Разумеется, при таком методе получения дескрипторов должна быть исключена ситуация попадания в дескрипторы явно неинформативных частей речи (предлогов, местоимений и некоторых других). Второй проблемой является необходимость отбрасывать в словах-дескрипторах окончания слов, чтобы употребление одного и того же термина в разных словосочетаниях не приводило к появлению множества дескрипторов, различных по написанию, но обозначающих одно и то же понятие.

В каждой информационно-поисковой системе должна присутствовать административная подсистема и поисковая подсистема.

Административная подсистема предназначена для организации новых баз данных, определения структуры вводимых в них записей, ввода подготовленных документов в базы данных в соответствии с определенными структурами, а также для создания главного инвертированного файла – основного средства ускорения поиска требуемой информации в ИПС с помощью ключевых слов.

Рассмотрим пример базы данных ИПС с инвертированным файлом для поиска сведений об экспонатах выставок. Документы, хранящиеся в базе данных, представляют собой описания выставочных экспонатов. Среди атрибутов документа источниками дескрипторов могут быть следующие: Название экспоната, Описание экспоната, Ключевые слова, Разработчик экспоната.

Атрибуты Название экспоната, Описание экспоната, Разработчик экспоната являются текстовыми величинами, содержащими произвольное количество строк и слов. Наиболее информативными с точки зрения выделения дескрипторов являются Название экспоната и Разработчик экспоната. При автоматическом получении множества дескрипторов слова, содержащиеся в атрибуте Описание экспоната, содержат слишком много слов из общей лексики языка, и наличие их в инвертированном файле терминов создаст файл слишком большого размера, в котором значительная доля слов не характеризует выставочные экспонаты и является информационным шумом. Надо отметить, что содержимое главного инвертированного файла предоставляется пользователю при работе с информационно-поисковой системой на экране дисплея, чтобы выбрать конкретные значения дескрипторов для команд выборки. С этой точки зрения администратор информационно-поисковой системы должен отбирать для автоматического индексирования те атрибуты, в которых содержится мало слов, составляющих информационный шум, а Описание экспоната этим требованиям не удовлетворяет.

В названии экспоната информативность слов, образующих значение этого атрибута, безусловно, высокая, однако существует возможность, что в название не попадут термины, характеризующие область техники, к которой относится экспонат, область его применения и т. п. С этой целью в базе данных выставочных экспонатов предусмотрен дополнительный атрибут Ключевые слова, в котором разработчики экспоната или эксперты выставки должны указать дополнительные дескрипторы, отсутствующие среди слов названия экспоната.

Если дескриптор содержит несколько слов, которые нельзя разделять при автоматическом индексировании, то в этой группе слов вместо пробелов должны использоваться знаки подчеркивания.

Процесс создания базы данных информационно-поисковой системы завершается формированием главного инвертированного файла, в котором для каждого значения дескриптора, полученного при автоматическом индексировании, указываются номера записей, среди значений атрибутов которых есть слова или словосочетания, совпадающие с этим дескриптором. Списки связи в этом случае не требуются, поскольку отдельные базы данных представляют собой тематически различные множества документов и не имеют общих дескрипторов. Кроме того, возможно формирование дополнительных инвертированных файлов по значениям тех атрибутов, которые не подключались к процессу автоматического индексирования.

Команда поиска Найти может использовать в качестве условий выборки значения из главного инвертированного файла, а также из дополнительных инвертированных файлов. Кроме того, пользователь может набрать условие выборки на клавиатуре.

Рассмотрим основные возможности реализации поиска, характерные для большинства информационно-поисковых систем, на простых примерах.

Пример 1

Поиск записей, содержащих слово подшипник. При поиске используется главный инвертированный файл.

Текст команды и ответ информационно-поисковой системы приводятся ниже.

>Найти подшипник

1 27 Найти подшипник,

где 1 – порядковый номер запроса,
27 – *количество найденных записей.*

Найти подшипник – текст команды запроса, который система воспроизводит заново после окончания поиска.

Теперь найденные записи (документы) доступны для просмотра, они полностью или частично могут быть напечатаны, перенесены во внешнюю память компьютера. С найденным множеством записей могут производиться и другие операции по формированию производной информации.

Следует отметить, что слово подшипник в этом примере найдено с помощью главного инвертированного файла во всех атрибутах 27 записей, подключенных к процессу автоматического индексирования (в нашем примере – это атрибуты Название экспоната, Ключевые слова, Разработчик экспоната).

Для того чтобы найти записи, содержащие слово подшипник как значение конкретного атрибута (например, Название экспоната), надо использовать дополнение названия атрибута к искомому слову.

Пример 2

Поиск записей, содержащих в атрибуте Название экспоната слово подшипник.

>Найти подшипник/Название

2 19 Найти подшипник/Название

Для просмотра предоставляются 19 записей, в которых слово подшипник встречается в значениях атрибута Название экспоната.

В большинстве поисковых систем реализованы три логические операции: И (конъюнкция), ИЛИ (дизъюнкция), НЕ (конъюнкция плюс отрицание).

Пример 3

>Найти подшипник НЕ роликовый

3 10 Найти подшипник НЕ роликовый

Каждая из 10 записей, выбранных по этому запросу, содержит термин подшипник, но не содержит термин роликовый.

В одном запросе возможно использование нескольких логических операций. В этом случае для многих поисковых систем принимается условие, что все логические операции имеют одинаковый приоритет и выполняются слева направо.

С помощью скобок можно изменить порядок применения логических операций к результатам выполнения элементарных поисковых операций по отдельным терминам.

Пример 4

Использование скобок. Команда

>Найти подшипник ИЛИ ротор И электродвигатель

аналогична команде

>Найти (подшипник ИЛИ ротор) И электродвигатель

Измененный порядок скобок приведет к другой команде поиска с другим результатом.

Если необходимо найти записи, в которых есть термины, начинающиеся с одинаковой последовательности букв, следует использовать усеченные термины. Для указания усеченных терминов служит вопросительный знак "?", расположенный в конце поискового термина-корня. После вопросительного знака может непосредственно следовать число, обозначающее максимальное количество усекаемых знаков.

Для указания произвольного символа в поисковом термине используется восклицательный знак "!". Этот знак не может замещать первый символ термина.

>Найти т!л

проводится поиск терминов типа тол, тыл и так далее.

>Найти маши?

проводится поиск терминов типа машина, машинный, машиностроение и так далее.

Возможности информационно-поисковых систем по обработке множества отобранных в результате выполнения запроса документов здесь не рассматриваются.

Кроме поисковых операций, для современных информационных систем очень важна реализация возможности "движения" по базе данных "вверх-вниз-направо-налево" и переход к смежным разделам базы данных. В этих случаях указание

условий поиска необязательно, и решение о переходе в ту или иную область базы данных принимает пользователь.

Общеупотребительной в таких случаях является *иерархическая модель данных с наличием ссылок* на тематически смежные записи данных в тех случаях, когда это необходимо.

Пример

Рассмотрим структуру ВУЗа. Из всех видов деятельности ВУЗа рассматривается только научно-исследовательская работа (рис. 2.9).

На первом уровне иерархической структуры должны располагаться:

- главная запись (сведения о руководстве ВУЗа);
- отношение Person с данными о преподавателях. Сведения о преподавателях нумеруются – Person(1), Person(2), Person(3) и т. д. Сведения о факультетах являются самостоятельными иерархическими базами данных. Их количество равно числу факультетов.

На первом уровне базы данных о факультете располагаются две главные записи:

- сведения о руководстве факультета;
- список преподавателей факультета.

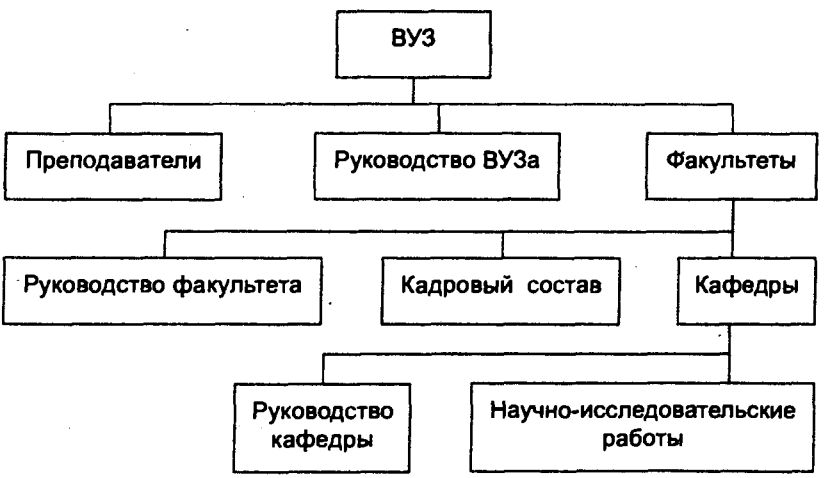


Рис. 2.9. Логическая структура информационно-поисковой системы вуза

Затем представлены иерархические базы данных о каждой кафедре факультета.

В базе данных о кафедре есть главная запись (руководство кафедры) и отношение Research, в котором отдельные темы исследований занумерованы как Research(1), Research(2), Research(3) и т. д.

Ссылки на другие разделы базы данных реализуются через номера строк в отношении Person.

Например, *содержимое главной записи* о ВУЗе:

<H2>Руководство</H2>
<DL>
<DT><B>Ректор</B><DD>Person(1)
<DT><B>Проректор по учебной работе</B><DD>Person(2)
<DT><B>Проректор по научной работе</B><DD>Person(3)
<DT><B>Проректор по социально-экономическим вопросам</B><DD>Person(4)
<DT><B>Проректор по административно-хозяйственной работе</B><DD>Person(5)
<DT><B>Ученый секретарь Совета</B><DD>Person(6)
</DL>

Список преподавателей факультета 01.

Person(1)
Person(4)
Person(2011)
Person(1001)
Person(1002)
...
Person(1043)
Запись Research(3):
Name(Региональные проблемы рыночной экономики)
Chief(1001)
BEGIN

Исследованы проблемы развития производительных сил России до 2000 года. Рассмотрены вопросы региональной экономической политики, и сформулированы основные направления развития отраслей и народнохозяйственных комплексов.

END

Ссылка Chief(1001) означает, что научным руководителем темы является преподаватель, информация о котором хранится как Person(1001). Этот преподаватель работает на факультете 01.

Структура базы данных разрешает фиксировать сведения о преподавателях, не работающих ни на одном факультете или работающих на нескольких факультетах.

ВОПРОСЫ И ЗАДАНИЯ

1. Сведения об учебном процессе зафиксированы в четырех отношениях:

Студ(Гр,Зач,ФИО)
Оценка(Гр,Зач,Дисц,Дата,Пр,Оц)
Расп(Дата,Гр,Дисц,Пр)
Преп(Дисц,Пр,Каф)
В задании используются следующие обозначения:
Студ – студент
Гр – номер группы
Зач – номер зачетной книжки
ФИО – фамилия студента
Дисц – дисциплина
Пр – фамилия преподавателя
Оц – оценка
Расп – расписание
Преп – преподаватель
Каф – название кафедры

Запишите с помощью операторов реляционной алгебры следующие запросы. В тех случаях, когда это возможно, запишите запросы на языках dBASE и SQL.

1) Найдите фамилии преподавателей, ведущих занятия в группах 305 и 307 одновременно.
2) Какие оценки получил студент Федоров?
3) У каких студентов преподает Иванов?
4) Какие студенты сдали те же экзамены, что и Федоров?
5) Какие преподаватели работают 10.10.98?
6) Какие преподаватели ведут занятия в тех же группах, что и Иванов?
7) По каким предметам сдается зачет, а не экзамен?

8) Какие студенты изучают дисциплину ВМ 10.10.98?

9) Какие дисциплины преподаются на кафедре ВМ?

10) Какие преподаватели преподают дисциплину ВМ?

11) Какие преподаватели поставили удовлетворительные оценки в группе 305?

12) Какие экзамены сданы у всех студентов группы 305?

13) Какие кафедры ведут занятия в группе 305?

14) Какие преподаватели работают в те же дни, что и Иванов?

15) Какие преподаватели поставили отличные оценки студенту Федорову?

16) По каким дисциплинам студент Федоров получил отличные оценки?

17) Какие студенты учатся в той же группе, что и Федоров?

2. *Разработайте* программы реализации операций вычитания и деления отношений для СУБД семейства dBASE. Требуемые имена файлов и атрибутов передавайте как параметры.

3. В п. 2.2.1 приведен метод поиска вероятного ключа отношения (если он единственный). Из полного списка атрибутов отношения *вычеркните* те, которые встречаются в правых частях функциональных зависимостей. Оставшиеся атрибуты образуют вероятный ключ. *Докажите* корректность этого метода. Какое множество функциональных зависимостей необходимо использовать?

4. Какой нормальной форме соответствует база данных, приведенная в задании 1?

5. *Определите* отношения в 3НФ для известного списка атрибутов БД и функциональных зависимостей.

Табельный номер (таб. N)	Участок → Цех
Фамилия рабочего (ФИО)	Таб. N → Цех
Цех	Таб. N → Участок
Участок	Таб. N, Дата → Сумма
Дата	Таб. N → ФИО
Сумма зарплаты (сумма)	Таб. N → ФИО, Участок

6. *Определите* отношения в 3НФ для известного списка атрибутов БД и функциональных зависимостей.

ФИО служащего (ФИО)	ФИО, Дата → Должность
Должность	ФИО → Должность
Дата	ФИО, Дата → Зарплата
Зарплата	ФИО, Имя_ребенка → Возраст
Имя_ребенка	
Возраст ребенка	

7. *Разработайте* структуру двухуровневой сетевой базы данных для заданного множества атрибутов и отношений.

ФИО студента	W(ФИО, Группа)
Группа	S(Преподаватель, Кафедра)
Дисциплина	Z(ФИО, Дисциплина, Преподаватель, Оценка)
Преподаватель	Y(Группа, Преподаватель, Дисциплина, День занятий)
Кафедра	
Оценка экзамена	F(Дисциплина, Кафедра)
День занятий	

Студенты и преподаватели-однофамильцы отсутствуют.

8. *Найдите* все вероятные ключи в отношении Т3 из п. 2.2.1.

9. Для приведенной ниже иерархической структуры базы данных укажите минимально возможный набор атрибутов в отношениях:

Атрибуты: Музей, Город, Экспонат, Год, Выставка, ФИО реставратора.

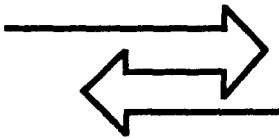
Отношения: W(Музей, Город), C(Экспонат, Год поступления), T(Экспонат, Год реставрации, ФИО), S(Выставка, Экспонат, Год выставки).

Веерные отношения: (W,C), (C, S), (C, T).

Названия музеев и выставок не повторяются.

10. Для баз данных в 3НФ из заданий 5 и 6 проверьте их ацикличность.

11. Из каких атрибутов состоит первичный ключ отношения, в котором нет справедливых функциональных зависимостей?



ГЛАВА 3

МЕТОДЫ ОРГАНИЗАЦИИ ДАННЫХ

3.1 АНАЛИЗ АЛГОРИТМОВ И СТРУКТУР ДАННЫХ

Методы хранения данных в памяти ЭВМ обычно предполагают раздельное хранение значений каждой составной единицы информации. Отдельное значение СЕИ, находящееся в памяти ЭВМ, называется *записью*. Запись состоит из значений атрибутов, входящих в структуру СЕИ. Множество записей образует массив, или файл. Термин *массив* обычно используется при рассмотрении данных в оперативной памяти ЭВМ, а термин *файл* применяется для данных, хранимых на внешних запоминающих устройствах. Как правило, файл содержит записи, принадлежащие одной и той же СЕИ, хотя в общем случае это не является обязательным.

Под *организацией значений данных* понимают относительно устойчивый порядок расположения записей данных в памяти ЭВМ и способ обеспечения взаимосвязи между записями.

Организация значений данных (далее называемая просто организацией данных) может быть линейной и нелинейной. При линейной организации данных каждая запись, кроме первой и последней, связана с одной предыдущей и одной последующей записями. У записей, соответствующих нелинейной организации данных, количество предыдущих и последующих записей может быть произвольным.

Линейные методы организации данных различаются только способами указания предыдущей и последующей записи по отношению к данной записи. Но это приводит к тому, что ал-

горитмы, эффективные для одних методов организации данных, становятся неприемлемыми для других методов.

Среди линейных методов выделяются последовательная и цепная организации данных. При *последовательной организации данных* записи располагаются в памяти строго одна за другой, без промежутков, в той последовательности, в которой они обрабатываются. Последовательная организация данных обычно и соответствует понятию массив (файл).

Записи, составляющие массив, с точки зрения способа указания их длины делятся на записи фиксированной, переменной и неопределенной длины. *Записи фиксированной (постоянной) длины* имеют одинаковую, заранее известную длину. Если длины записей неодинаковы, то длина указывается в самой записи. Такие записи называют *записями переменной длины*. Вместо явного указания длины записи можно отмечать окончание записи специальным символом-разделителем, который не должен встречаться среди информационных символов значения записи. Записи, заканчивающиеся разделителем, называются *записями неопределенной длины*.

Адреса промежуточных записей фиксированной длины в массиве задаются формулой

$$A(i) = A(1)+(i-1)*L,$$

где A(1) – начальный адрес первой записи;
A(i) – начальный адрес i-й записи;
L – длина одной записи.

Для массива записей переменной и неопределенной длины подобной простой формулы не существует. Они занимают меньший объем памяти, но их обработка ведется с меньшей скоростью, поскольку затруднено обнаружение следующей записи.

В структуре записей последовательного массива обычно выделяется один или несколько ключевых атрибутов, по значениям которых происходит доступ к остальным значениям атрибутов той или иной записи. Состав ключевых атрибутов необязательно соответствует понятию первичного ключа.

Будем считать, что последовательный массив состоит из записей фиксированной длины, а среди атрибутов записи будем выделять только один ключевой атрибут. Наличие в записях других (неключевых) атрибутов специально не оговаривается. Иллюстрацией такого последовательного массива служит рис. 3.1.

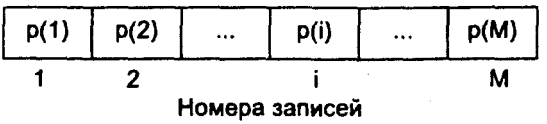


Рис. 3.1. Последовательная организация данных

Ключевые атрибуты в записях обозначаются через p(i), где i – номер записи, общее число записей в массиве обозначается через M.

Записи массива могут быть упорядоченными или неупорядоченными по значениям ключевого атрибута (ключа), имя которого одинаково во всех записях. Ключевой атрибут обычно является *атрибутом-признаком*. Часто требуется поддерживать упорядоченность записей по нескольким именам ключевых признаков. В этом случае среди признаков устанавливается старшинство. Условие упорядоченности записей в массиве (и вообще для линейной организации данных) выглядит следующим образом:

p (i) < = p (i + 1) – упорядоченность по возрастанию;

p(i) >= p(i+1) – упорядоченность по убыванию.

Наиболее важными и часто применяемыми алгоритмами обработки данных являются формирование данных, их поиск и корректировка, а также последовательная обработка. Эти алгоритмы могут быть реализованы с использованием достаточно большого количества методов организации данных. Здесь мы рассмотрим выбор наилучшего метода организации данных для названных алгоритмов. Сами методы организации данных будут представлены в их простейшей форме, а уточнения, рассчитанные на данные большого объема на вне-

шних запоминающих устройствах, как это характерно для СУБД и пакетов прикладных программ, приводятся в п. 3.3.

Данные обычно возникают в неупорядоченной форме. Перед обработкой, как правило, целесообразно упорядочить их по значениям ключевых атрибутов, что составляет одну из основных работ по формированию (подготовке) данных. Процедуру упорядочения файла часто называют *сортировкой*.

Упорядоченные данные эффективны для организации быстрого поиска информации. Выходные документы, выводимые на печать, полученные на основе отсортированных данных, удобны для дальнейшего использования человеком. Многие алгоритмы задач управления вообще рассчитаны на использование только упорядоченных данных. Отсортированные данные позволяют организовать быструю обработку нескольких массивов. Далее будем считать все массивы упорядоченными по возрастанию значений одного атрибута, когда для ключа i-й записи p(i) справедливо условие p(i)<=p(i+1).

Критерии эффективности алгоритмов

Естественной характеристикой эффективности того или иного алгоритма служит время его выполнения в зависимости от ряда параметров хранимой информации. Поэтому для каждого метода организации данных требуется анализировать следующие величины:

- время формирования данных, т. е. время создания в памяти ЭВМ так или иначе упорядоченного представления данных (упорядочение способно ускорить выполнение алгоритмов поиска данных);
- время поиска данных. Как известно, условия поиска (выборки) на практике могут быть достаточно разнообразные. Анализируется обычно простейший и наиболее распространенный случай (поиск по совпадению) – найти записи, у которых значение ключевого атрибута равно заранее известной величине q;

- время корректировки данных. Из всех возможных вариантов корректировки учитывается включение или исключение одной записи;
- объем дополнительной памяти, расходуемой под служебную информацию (например, адреса связи).

На время выполнения алгоритмов влияет, однако, ряд факторов, которые не хотелось бы рассматривать при теоретическом анализе алгоритмов. Среди них – быстродействие конкретной ЭВМ, применяемый язык программирования, стиль программирования конкретного программиста. Чтобы можно было не принимать во внимание подобные факторы, целесообразно анализировать не время, а количество выполняемых элементарных операций, в частности операций сравнения значений ключевых атрибутов и искомых величин. Время выполнения алгоритма обычно прямо пропорционально числу требуемых сравнений, и подобная замена критерия эффективности алгоритма вполне законна.

При анализе алгоритмов необходим еще ряд допущений, обеспечивающих использование равномерного распределения вероятностей для всех случайных величин, описывающих работу алгоритма, в том числе:

- распределение значений ключевых атрибутов в массиве из M записей – равномерное;
- значение q при поиске по совпадению выбрано случайно: это означает, что поиск с одинаковой вероятностью 1/M может закончиться на любой записи массива;
- положение включаемой (исключаемой) записи при корректировке определяется теми же вероятностями, что и при поиске.

Минимальное число сравнений при упорядочении последовательного файла можно определить, не рассматривая ни одного конкретного алгоритма с помощью рассуждений, характерных для теории информации. Процедура упорядочения состоит из серии сравнений ключевых атрибутов и тех или иных перераспределений записей по результатам сравнения. Массив, обладающий наибольшей неопределенностью своего

состояния, будем называть *случайным массивом*. Все его M! состояний равновозможны. Через M! обозначено произведение 1·2·...·M.

Таким образом, минимальное число сравнений, необходимое для упорядочения массива из M записей, определяется как

$$C = \log M!,$$

что соответствует минимально возможному числу вопросов о состоянии массива с возможными ответами типа: да – нет. Условимся, что функция log обозначает логарифм по основанию 2.

После преобразований по формуле Стирлинга получаем:

$$C=M \cdot (\log M - 1.43).$$

В анализе алгоритмов принято учитывать в подобных формулах лишь те слагаемые, которые ощутимо влияют на общую сумму (разность).

Соответствующее преобразование выглядит как $C \sim M \cdot \log M$ (читается: C пропорционально M · logM). Значением коэффициента пропорциональности в этом контексте мы интересоваться не будем. Разумеется, справедлива запись выражения для времени сортировки $T \sim M \cdot \log M$.

Поиск в последовательном массиве

Поиском называется процедура выделения из некоторого множества записей определенного подмножества, записи которого удовлетворяют некоторому заранее поставленному условию. Условие поиска часто называют запросом на поиск.

Простейшим условием поиска является *поиск по совпадению*, т. е. равенство значения ключевого атрибута i-й записи p(i) и некоторого заранее заданного значения q. Алгоритмы всех разновидностей поиска можно получить из алгоритмов поиска по совпадению, которые и рассматриваются в дальнейшем.

Базовым методом доступа к массиву является *ступенчатый поиск*. Этот метод предполагает упорядоченность обрабатываемых записей, причем безразлично, по возрастанию или по убыванию. Для определенности будем считать, что массив отсортирован по возрастанию значений ключевого атрибута p(i).

Простейшим вариантом ступенчатого поиска (его можно назвать одноступенчатым) является *последовательный поиск*. Искомое значение q сравнивается с ключом первой записи, если значения не совпадают, с ключом второй записи и т. д. до тех пор, пока q не станет больше ключа очередной записи. Алгоритм последовательного поиска может быть представлен следующей программой на языке Паскаль:

```
program poisk;
const M=20;
var i,q:integer;
p: array[1..M] of integer;
begin
write('ввод q');
readln(q);
for i:=1 to M do
begin
write('ввод', i, ' элемента массива'); readln(p[i])
end;
i:=1;
while q < p[i] do i:= i+1;
if q <> p[i] then writeln('в массиве нет значения', q)
else writeln('номер ',i,' значение', q)
end.
```

Число сравнений в цикле while может быть выражено как $C= \Sigma i*r(i)$, где суммирование по i ведется в пределах от 1 до M.

Поиск с одинаковой вероятностью $r(i)=1/M$ может окончиться на любой записи, поэтому

$$C = (1/M)\cdot \Sigma i = (M + 1)/2 \text{ или } C \sim M.$$

Рассмотрим двухступенчатый поиск в массиве, состоящем из M записей. Для заданного M выбирается константа d1, называемая *шагом поиска*. Если необходимо отыскать запись со значением ключевого атрибута, равным q, производятся следующие действия.

Значение q последовательно сравнивается с рядом величин p(1), p(1+d1), p(1+2*d1), ..., p(1+k*d1) до тех пор, пока будет впервые достигнуто неравенство p(1+m*d1)=>q. Здесь заканчивается первая ступень поиска. На второй ступени q последовательно сравнивается со всеми ключами, которые имеют номер 1+m*d1 и больше, до тех пор, пока в процессе сравнений будет достигнут ключ, больший, чем q. Извлеченные при этом записи с ключом q образуют *результат поиска*.

Эффективность поиска измеряется количеством произведенных сравнений. Для двухступенчатого поиска среднее число сравнений примерно составит:

$$C = M/(2 * d1) + d1/2.$$

Параметр d1 выбираемый и естественно выбрать его так, чтобы минимизировалось C. Будем считать d1 непрерывной переменной и вычислим производную

$$C' = -M/(2*d1\hat{}2)+1/2.$$

Из условия C'= 0 получаем d1=SQR(M). Вторая производная C" в точке x = d1 положительна, следовательно, достигнуто минимальное значение C.

При n-ступенчатом поиске заранее выбираются константы n и S. На первом этапе ключевые атрибуты для сравнения с искомым ключом q выбираются из массива по закону арифметической прогрессии, начиная с p(1) и шагом d1=M/S (округление в меньшую сторону). Когда будет впервые достигнут ключ p(k) > q, выбирается шаг d2 = d1/S и организуются сравнения с этим шагом, начиная с p(k-d1). Описанные действия повторяются n раз, причем шаг на последней ступени поиска dn=1.

Минимальное число сравнений достигается при S=M^(1/n), и, кроме того, существует оптимальное n=ln(M).

Ступенчатый поиск имеет важный частный вариант – *бинарный поиск*, когда S=2.

Для бинарного поиска вводятся левая граница интервала поиска A и правая граница B. Первоначально интервал охватывает весь массив, т. е. A=0, B=M+1. Вычисляется середина интервала i по формуле i=(A+B)/2 с округлением в меньшую сторону. Ключ i-й записи p(i) сравнивается с искомым значением q. Если p(i) =q, то поиск заканчивается. В случае p(i)>q записи с номерами i+1, i+2, ..., M заведомо не содержат ключа q, и надо сократить интервал поиска, приняв B=i. Аналогично при p(i)<q надо взять A=i. Далее середина интервала вычисляется заново, и все действия повторяются. Если будет достигнут нулевой интервал, то требуемой записи в массиве нет.

Алгоритм бинарного поиска может быть представлен следующей программой на языке Паскаль:

```
program bin;
const M=10;
var i,A,B,q: integer;
p:array[1..M] of integer;
begin
write('ввод q');
readln(q);
for i:=1 to M do
begin
write('ввод', i, ' элемента массива'); readln(p[i])
end;
A:=0; B:=M+1;
i:=(A+B) div 2; {определение середины интервала}
{цикл до встречи p[i]=q или до нулевого интервала A=i}
while (p[i]<>q) and (A<i) do
begin
if q>p[i] then A:=i else B:=i;
i:=(A+B) div 2; {середина нового интервала}
end;
```

```
if A=i then writeln('в массиве нет значения', q)
else writeln('номер ',i,' значение',q)
end.
```

Достаточно легко оценить максимальное число сравнений Cm при поиске данных бинарным методом. Сокращение интервала поиска на каждом шаге в худшем случае приведет к интервалу нулевой длины, что соответствует отсутствию в массиве искомого значения ключевого атрибута. После первого сравнения интервал поиска составит M/2 записей, после второго – M/4 и т. д. Когда интервал поиска впервые станет меньше 1, применяемая схема округления результата деления даст нулевой интервал и поиск закончится. Это соответствует неравенству

$$M/(2^{Cm})<=1$$

(знак ^ обозначает возведение в степень), откуда Cm~log M.

Среднее число сравнений при бинарном поиске составляет C=log(M) -1. Для обоснования представим все возможные разветвления алгоритма бинарного поиска в виде дерева. Число уровней дерева соответствует Cm. Половина возможных сравнений расположена на последнем уровне и половина – на первых (log(M)-1) уровнях.

Определение лучшего метода поиска (из рассмотренных выше последовательного, двухступенчатого и бинарного) опирается на следующие рассуждения. Во всех трех случаях время поиска является функцией от числа записей M. Конкретные выражения составляют:

- для последовательного поиска

$$T1\sim M \text{ или } T1=k1*M;$$

- для двухступенчатого поиска

$$T2\sim SQR(M) \text{ или } T2=k2(SQR(M));$$

- для бинарного поиска

$$T3\sim logM \text{ или } T3=k3*logM.$$

Коэффициенты пропорциональности в каждом случае неизвестны (они определяются в зависимости от быстродействия ЭВМ и применяемого языка программирования). Однако очевидно, что функция логарифма растет с увеличением M медленнее, чем две другие функции (для T1 и T2), как это показано на рис. 3.2.

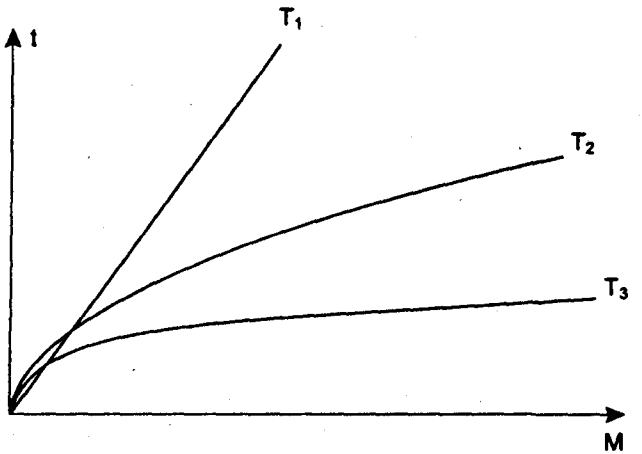


Рис. 3.2. Сравнение методов поиска данных в массиве

Поэтому можно утверждать даже при неизвестных коэффициентах k1, k2, k3, что при достаточно большом числе записей M бинарный метод поиска выполняется, безусловно, быстрее двух остальных. Значения k1, k2, k3 влияют на граничную величину M, выше которой преимущество бинарного поиска безусловно.

Корректировка последовательного массива

Корректировка массива может касаться одной его записи или группы записей. Отдельная запись может включаться в массив и исключаться из него. Кроме того, в записи могут быть измене-

ны значения отдельных атрибутов (как правило, неключевых). В любом случае перед непосредственно корректировкой выполняется поиск местонахождения корректируемой записи.

Включение новой записи (например, со значением ключевого атрибута w) в последовательный упорядоченный массив не должно нарушать его упорядоченность. Поэтому сначала необходимо найти положение новой записи относительно имеющихся в массиве записей, т. е. выполнить поиск ключа новой записи w. Если значения w в массиве нет, то поиск остановится там, где должна находиться запись с ключом w при сохранении общей упорядоченности массива. Новая запись не может сразу занять место, где остановился поиск, необходимо выполнить пересылку записей, чтобы освободить его. Пересылка начинается с последней записи (она после пересылки занимает следующую позицию по направлению к концу массива), затем предпоследняя запись пересылается на место последней и т. д. В результате освобождается место для новой записи.

Итак, время включения записи складывается из времени поиска и времени пересылки записей. В среднем пересылка затрагивает примерно половину записей массива. Время пересылки одной записи пропорционально ее длине.

$$Tk=\log M+M\cdot L,$$

где L – длина одной записи массива.

В формуле для Tk второе слагаемое по величине всегда значительно превышает первое, поэтому можно считать:

$$Tk\sim M\cdot L.$$

При исключении записи из массива также необходимо сначала найти в массиве ключ удаляемой записи, а затем выполнить пересылки записей в направлении к началу массива для уничтожения исключаемой записи в памяти. Очевидно, что время исключения записи описывается той же формулой, что и время включения записи.

Можно прийти к выводу, что включение-исключение записей поодиночке является очень длительной процедурой. Поэтому в ряде случаев удобно накапливать корректирующие записи в специальном массиве изменений, а не вносить их по мере появления.

Массив записей, подвергающихся изменениям, называется *основным*. Изменения, которые необходимо внести в основной массив, накапливаются в специальном упорядоченном массиве изменений, рассчитанном на 1<=m<=M записей. Обычно m составляет несколько процентов от M. При необходимости обработки основного массива он объединяется с массивом изменений.

При объединении основного массива с массивом изменений выполняются следующие операции (алгоритм слияния):

- ключ очередной записи основного массива сравнивается с ключом очередной записи массива изменений, и запись с меньшим значением ключа добавляется в новый массив (результат слияния);
- когда все записи одного из массивов будут перезаписаны полностью, оставшиеся записи другого массива добавляются в результирующий массив, и алгоритм заканчивается.

Цепная (списковая) организация данных

Решение целого ряда задач обработки данных требует применения таких методов организации данных, которые позволили бы связать физически разнесенные в памяти данные в логическую последовательность, определяющую порядок их обработки. Простейшим методом, применяемым для этих целей, является *списковая (цепная) организация данных*.

Списком называется множество записей, занимающих произвольные участки памяти, последовательность обработки которых задается с помощью адресов связи. *Адресом связи* некоторой записи называется атрибут, в котором хранится начальный адрес или номер записи, обрабатываемой после этой записи. Обычная последовательность обработки записей в списке определяется возрастанием значений ключа в записях.

В списке выделяется собственная информация (записи с содержательными сведениями) и ассоциативная информация, т. е. все адреса связи.

Описание записей списка на языке программирования (например, Паскаль) может быть произведено двумя способами.

1. Определение адресов связи как начальных адресов записей:

```
type ref=^node;
node=record
    key: integer; {ключевой атрибут записи}
    other: string[30]; {остальные атрибуты}
    next: ref {адрес связи}
end;
```

2. Определение адресов связи как номеров записей:

```
const M=12;
type
node=record
    key: integer; {ключевой атрибут записи}
    other: string[30]; {остальные атрибуты}
    next: integer {адрес связи}
end;
var t:array[1..M] of node;
```

Второй вариант является более практичным, особенно если требуется хранить список на внешнем запоминающем устройстве.

Возможны два способа организации списка – совместное размещение собственной и ассоциативной информации, когда запись и ее адрес связи образуют одно целое (рис. 3.3,а), и раздельное, когда имеется списковая организация адресов связи и последовательное хранение собственной информации (рис. 3.3,б).

При списковой организации данных необходим специальный атрибут, называемый *указателем списка*, который содержит начальный адрес или номер первой в порядке обработки

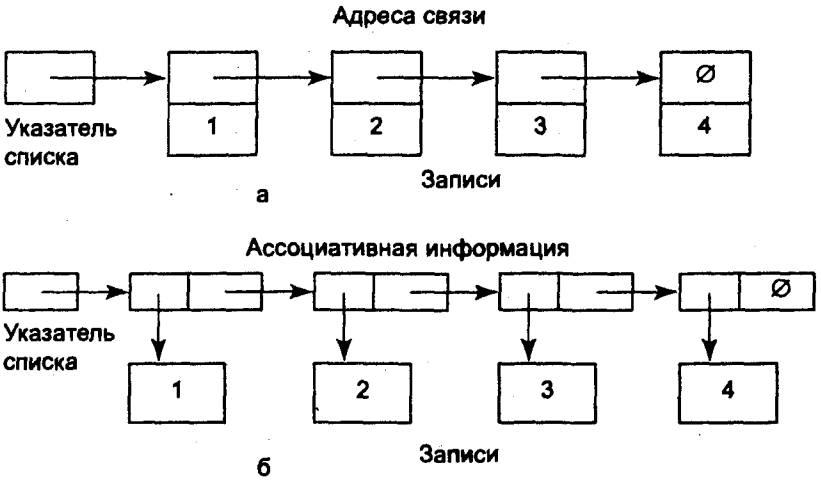


Рис. 3.3. Варианты списковой организации данных:
а – совместное хранение записей и адресов связи;
б – раздельное хранение записей и адресов связи (∅ – конец списка)

записи списка. Кроме того, адрес связи последней записи списка должен содержать специальное значение, называемое *концом списка* и отмечающее, что последующих записей у данной записи нет. Обычно конец списка отмечается нулем.

На рисунках адрес связи изображается прямоугольником со стрелкой, стрелка указывает на запись, адрес хранения которой содержится в адресе связи.

При формировании упорядоченного списка записей возможны два варианта:

- вновь поступающие записи вставлять так, чтобы не нарушать упорядоченность по ключу;
- создать сначала неупорядоченный список, а затем отсортировать его.

Учитывая, что для сортировки можно использовать метод слияния, второй вариант следует признать более целесообразным.

В итоге время формирования упорядоченного списка пропорционально $T \sim M \cdot \log M$.

Для поиска в упорядоченном списке можно использовать те же методы, что и в последовательном массиве, однако эффективность этих методов иная, поскольку адреса связи создают возможность быстрого доступа только к следующей записи списка.

Для *поиска данных в однонаправленном списке* используется единственный метод – *последовательный поиск*. Ключевой атрибут первой записи (ее адрес извлекается из указателя списка) сравнивается с искомым значением q, затем такое же сравнение выполняется для ключа второй записи, которая извлекается по адресу связи первой записи, и т. д. Время поиска, естественно, пропорционально T~M.

Неэффективность бинарного поиска для списковой организации данных объясняется тем обстоятельством, что для достижения середины интервала требуется последовательное движение в соответствии с адресами связи и суммарное количество переходов от записи к записи достаточно велико. Число переходов от записи к записи при доступе к серединам интервалов представляется величиной M/2+M/4+M/8+..., что практически составляет M.

Для ускорения доступа к списку могут быть рекомендованы такие варианты использования адресов связи, как двунаправленный и кольцевой список (рис. 3.4):

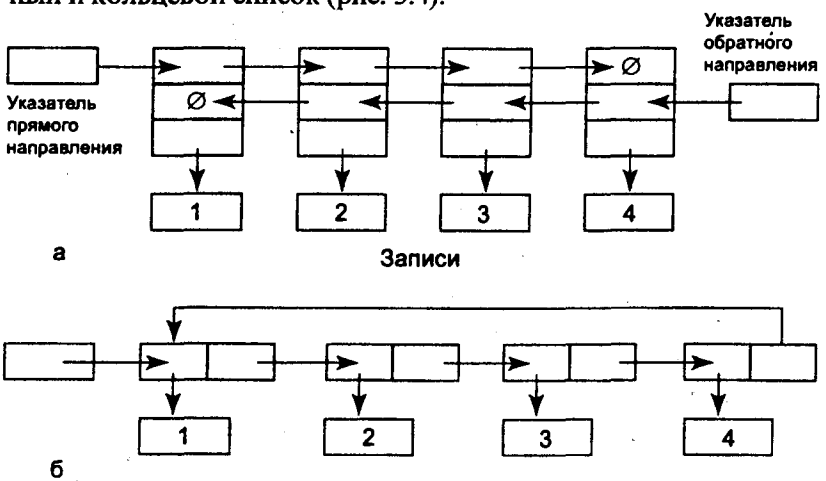


Рис. 3.4. Организация списков:
а – двунаправленного; б – кольцевого

- двунаправленный список образован двумя цепочками адресов связи – от первой записи к последней и от последней записи к первой;
- в кольцевом списке последний адрес связи указывает на первую запись.

Цепной каталог

Цепным каталогом называется сплошной участок памяти (или несколько таких участков), в котором одновременно размещаются список обрабатываемых записей и список свободных позиций памяти. Адрес связи, отмечающий первую обрабатываемую запись, называется *указателем списка*. Адрес связи, отмечающий первую свободную позицию памяти, называется *указателем свободной памяти*. Адрес связи последней записи (или последней позиции свободной памяти) в списке называется *концом списка*, и здесь отмечается нулевым значением.

Рассмотрим **пример** цепного каталога, в котором адреса связи представлены номерами соответствующих записей. Описание каталога на языке Паскаль имеет вид:

```
const M=9;
type
node=record
key: integer; {ключевой атрибут записи}
next: integer {адрес связи}
end;
var t:array[1..M] of node;
```

Первоначальное состояние каталога показано на рис. 3.5,а.

Включение и исключение записей в цепном каталоге предполагает поиск местоположения включаемой (исключаемой) записи и замену значений адресов связи для установления новой последовательности записей основного списка и списка свободной памяти.

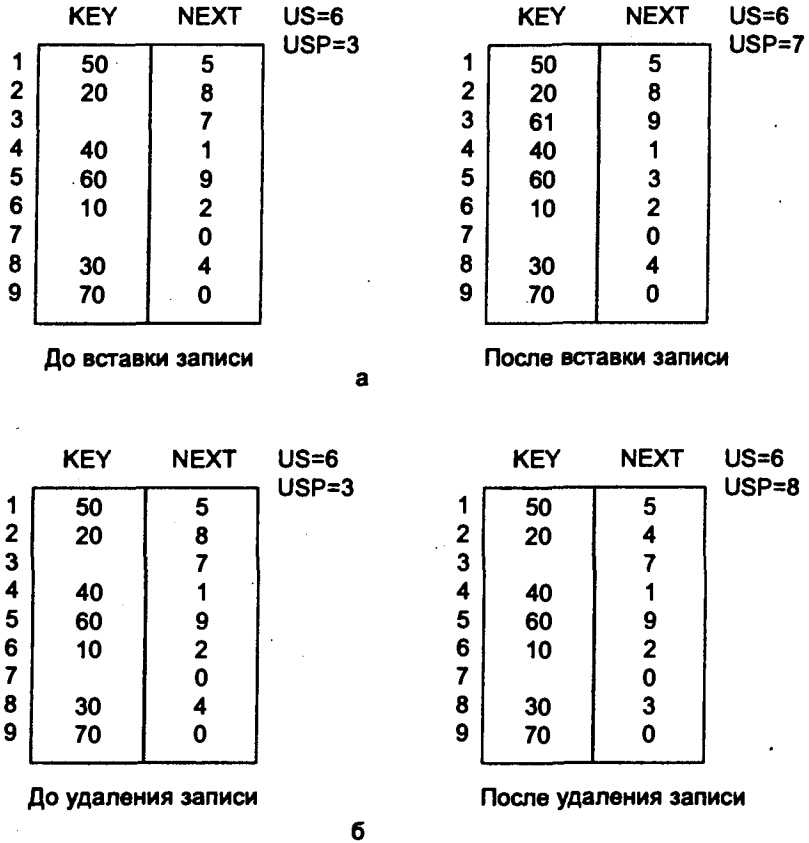


Рис. 3.5. Операции корректировки в цепном каталоге:
а – ставка записи с ключом 61; б – удаление записи с ключом 30

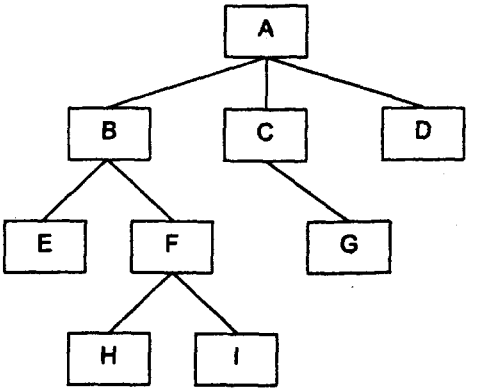
Алгоритм вставки записи с ключом F в цепной каталог.

1. Найти в каталоге запись с ключом непосредственно меньше, чем F (предшествующая запись).

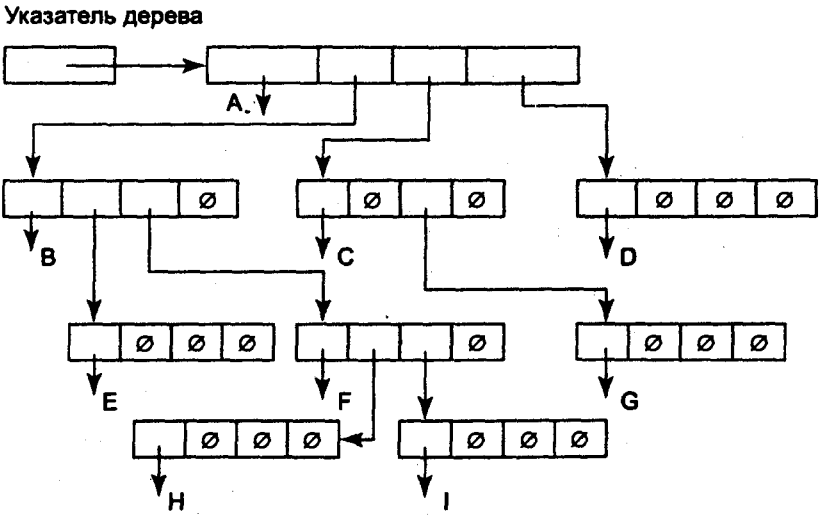
2. Поместить запись с ключом F в первую позицию свободной памяти.

3. Заменить указатель свободной памяти (УСП) на адрес связи новой записи, этот адрес – на адрес связи предшествующей записи, а последний – на первоначальное значение УСП.

На рис. 3.5,а вставляется запись с ключом F=61.



а



б

Рис. 3.6. Пример древовидной организации данных (а) и представление его в памяти ЭВМ (б)

Алгоритм удаления записи с ключом W из каталога.

1. Найти в каталоге запись с ключом непосредственно меньше, чем W (предшествующая запись).

2. Заменить УСП на адрес связи предшествующей записи, этот адрес – на адрес связи исключаемой записи, а последний – на первоначальное значение УСП.

На рис. 3.5,б удаляется запись с ключом W=30.

Оценка времени корректировки складывается из времени реализации поиска и времени на замену значений адресов связи. В последнем случае число пересылок адресов связи всегда одинаково и не зависит от числа записей в цепном каталоге, поэтому затраты времени на поиск при корректировке являются доминирующими и время корректировки пропорционально T~M.

Древовидная организация данных

Древовидной организацией данных (деревом) называется множество записей, расположенных по уровням следующим образом:

- на 1-м уровне расположена только одна запись (корень дерева),
- к любой записи i-го уровня ведет адрес связи только от одной записи уровня i-1.

В данном определении понятия "дерево" и "уровень" вводятся одновременно. Если записи получат номера уровней, соответствующие определению, то они получат и древовидную организацию.

Количество уровней в дереве называется *рангом*. Записи дерева, которые адресуются от общей записи (i-1)-го уровня, образуют *группу*. Максимальное число элементов в группе называется *порядком дерева*. В дереве на рис. 3.6,а порядок равен 3 и ранг составляет 4 (записи дерева обозначены заглав-

ными латинскими буквами). Деревья обычно формируются двунаправленными, адрес связи от записи уровня i+1 к записи i-го уровня называется *обратным*.

При размещении дерева в памяти ЭВМ каждая запись может занимать произвольное место.

Назовем *звеном связи* набор адресов связи, принадлежащих одной записи. Если порядок дерева равен p, то звено связи состоит из p+1 адресов (один адрес обратный, определяющий связь с записью непосредственно более высокого уровня). Корень дерева адресуется из специального указателя дерева. Незанятые адреса связи содержат признак конца списка. В качестве примера размещения дерева в памяти ЭВМ на рис. 3.6,б показан один из возможных вариантов представления дерева с рис. 3.6,а, обратные адреса связи не показаны.

Рассмотрим деревья порядка 2 (бинарные). Они интересны тем, что составляющие их записи могут быть упорядочены. Для этого один из атрибутов записи должен быть объявлен ключевым.

Чтобы определить понятие упорядоченности бинарных деревьев, требуется ввести ряд новых понятий. В качестве примера рассмотрим бинарное дерево на рис. 3.7 (внутри показаны значения ключевого атрибута). Запись A – корень дерева. Записи, у которых заполнены два адреса связи, называются *полными*, записи с одним заполненным адресом – *неполными*, записи с двумя незаполненными адресами – *концевыми*. На рис. 3.7 записи A, B, E, F – полные, C – неполная, D, H, I, J, K – концевые. Адреса связи делятся на левые и правые. Так, адрес от E к H – *левый*, от E к I – *правый*. Каждая запись имеет левую и правую ветви. Правую (левую) ветвь записи образует поддерево, адресованное из этой записи через правый (левый) адрес связи. У записи C правая ветвь состоит из записей F, I, K, левая ветвь пустая.

В упорядоченном бинарном дереве значение ключевого атрибута каждой записи должно быть больше, чем значение ключа

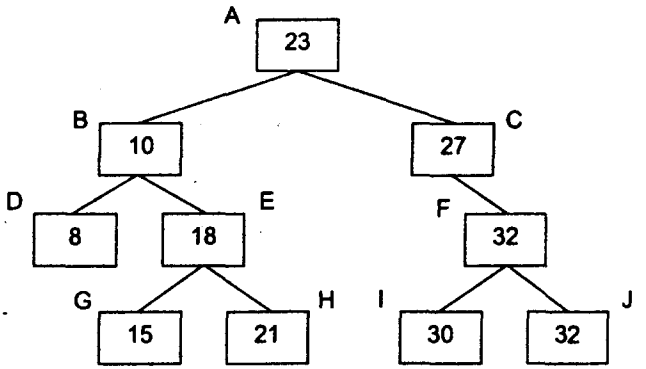


Рис. 3.7. Пример бинарного дерева

у любой записи на ее левой ветви, и не меньше, чем ключ любой записи на ее правой ветви. Это определение позволяет также различать левые и правые адреса (ветви).

Упорядоченное бинарное дерево формируется из неупорядоченного массива записей по специальному алгоритму. Этот алгоритм создает дерево из первой записи массива, затем – дерево из первых двух записей, из первых трех записей и так далее до исчерпания всех записей массива.

Алгоритм построения упорядоченного бинарного дерева

1. Первая запись массива с ключом p(1) становится корнем дерева.

2. Значение ключа второй записи p(2) сравнивается с p(1), находящимся в корне дерева. Если p(2)<p(1), то вторая запись помещается на левой от корня ветви, в противном случае – на правой ветви. Сейчас получено упорядоченное дерево из первых двух записей, далее на каждом шаге создается упорядоченное дерево из первых i записей.

3. Выбор места i-й записи массива производится следующим образом. Ключ p(i) сравнивается с корневым значением, и выполняется переход по левому адресу (если p(1)>p(i)), а при p(1)<=p(i) – по правому адресу. Ключ достигнутой записи также сравнивается с p(i), и снова организуется переход по левому или правому адресу и т. д. Когда будет достигнут незаполненный адрес связи, то он должен адресовать запись с ключом p(i). Указанные действия повторяются до исчерпания всех записей исходного массива.

Например, упорядоченное дерево на рис. 3.7 получается из массива записей с ключевыми атрибутами 23, 10, 18, 27, 15,32, 8,30,32,21.

В процессе поиска данных по совпадению в упорядоченном бинарном дереве просматривается некоторый путь по дереву, начинающийся всегда в его корне. Искомое значение ключа q сравнивается со значением корня p(1). Если p(1)>q, просмотр дерева продолжается по левой ветви корня, если p(1)<=q – по правой. Для произвольной записи дерева с ключом p(i) результаты сравнения означают:

- p(i)=q – запись, удовлетворяющая условию поиска, найдена, и поиск продолжается по правой ветви p(i);
- p(i)>q – производится переход к записи, расположенной на левой ветви p(i);
- p(i)<q – производится переход к записи, расположенной на правой ветви p(i).

Поиск заканчивается, когда у какой-либо записи отсутствует адрес связи, необходимый для дальнейшего продолжения поиска.

Для определения среднего числа сравнений при поиске записи в упорядоченном бинарном дереве рассмотрим ситуацию, когда требуемая запись не найдена, что равноценно вставке записи с искомым значением в дерево. Ненайденный ключ может с одинаковой вероятностью попасть в один из M+1 интервалов, образованных ключами, находящимися в дереве. Неудачный поиск закончится всегда на нулевом адресе связи.

Если обозначить через E(M) сумму длин всех ветвей дерева с учетом выхода на нулевые адреса связи, то среднее число сравнений при поиске в упорядоченном бинарном дереве C'(M) составит

$$C'(M)=E(M)/(M+1).$$

Дерево с M–1 вершиной отличается от дерева с M вершинами, полученного из него, отсутствием одной концевой записи или (применительно к величине E) двумя нулевыми адресами связи. Поэтому E (M) – E (M – 1) =2. Вычислим разность

$$dL = C'(M) - C'(M-1)= \frac{E(M) - E(M-1)}{M+1} = 2/(M+1).$$

Математическое ожидание средней длины пути при поиске C'(M) равно сумме математических ожиданий dL от 1 до M. Приближенно заменяя суммирование интегрированием, получаем:

$$C'(M) = 2\ln(M+1).$$

При замене натурального логарифма на двоичный ($\ln 2 = 0{,}7$) получаем оценку числа сравнений при поиске в упорядоченном бинарном дереве:

$$C = 1{,}4 \log M \text{ или } C \sim \log M.$$

При формировании упорядоченного бинарного дерева в среднем производится

$$C = 1{,}4M*\log M$$

сравнений пар ключевых атрибутов, где M – число записей, для которых строится дерево. Это соответствует затратам на поиск местоположения очередной записи в упорядоченном бинарном дереве из двух, трех и т. д. (до M) записей.

Включение новой записи при корректировке упорядоченного бинарного дерева означает выполнение одного шага алгоритма формирования дерева с включаемой записью на входе.

Сложность исключения зависит от того, какая запись исключается – концевая, неполная или полная. Первые два случая аналогичны корректировке при списковой организации данных. Адрес связи на исключаемую концевую запись заменяется на признак конца строки, адрес связи на исключаемую неполную запись заменяется на ее собственный адрес связи.

При исключении полной записи решается задача о подстановке на ее место другой записи, такой, что ее ключ не нарушает общей упорядоченности бинарного дерева – такие записи называются соседними. Соседняя слева запись – это запись с ключом, который непосредственно меньше ключа данной записи, а ключ соседней справа записи равен или непосредственно больше, чем ключ данной записи.

Способ нахождения соседней справа записи очень простой. Если исключаемая запись имеет ключ q, то от нее происходит переход по правой ветви дерева и производится поиск от достигнутой записи значения q. Запись, на которой остановится поиск, будет соседней. Она пересылается на место исключаемой записи, а сама соседняя запись исключается. Это уже простая задача, так как соседняя запись не может быть полной.

При поиске соседней слева записи надо выполнить переход по левой ветви от данной записи (с ключом q), а дальнейшие действия такие же, как и для поиска соседней справа записи.

Списки

В древовидной организации данных связь какой-то записи с N записями, составляющими ее группу, реализуется с помощью N адресов связи. Возможно, однако, связать все записи группы в цепочку и адресовать с предшествующего уровня первую запись группы. Таким образом получается новая нелинейная организация данных – список.

Списком называется множество элементов, каждый из которых может быть либо записью, либо списком. Структуру списка выражают формулой, в которой записи помечаются

буквами, а списки заключаются в круглые скобки. Список, включенный в другой список, называется подсписком. Перечисление всех списков из записей a1, a2,...,aN, образующих множество L0, сводится к следующему.

Обозначим через Li' множество всех кортежей с элементами из Li. Введем последовательность множеств:

L1=L0'+L0 (+ здесь означает знак объединения множеств)
L2=L1'+L1
L3=L2'+L2
.....

Все списки содержатся в множестве L=L0+L1+...Li+...

Пример

Пусть L0={a, b}. Тогда
L1={(a, b), (a), (b). a, b}
L2={((a, b), (a), (b), a, b),((a, b), (a), (b), a), ...)
и т. д.

Введенный аппарат списков намного шире, чем требуется для представления деревьев. Он используется самостоятельно также в задачах искусственного интеллекта и анализа текстовой информации.

Все линейные списки содержатся в множестве L0'.

Записи, входящие в список, могут занимать произвольные участки в памяти ЭВМ. Адреса связи, принадлежащие каждой записи, образуют *звено связи*. В звене связи однонаправленного списка два адреса: первый указывает на следующий элемент списка, второй – на подсписок или запись. В звене связи двунаправленного списка четыре адреса связи: два из них обеспечивают прямое и обратное направление в списке, третий и четвертый адресуют начало н конец подсписка. Однонаправленный список ((a,b),b,((c),c)) показан на рис. 3.8.

Дерево тоже имеет формулу, так как существует эквивалентный ему список,

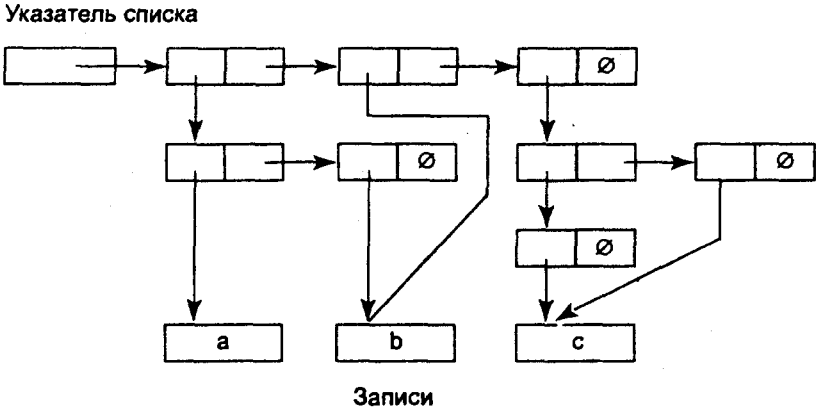


Рис. 3.8. Представление списка ((a, b), b, ((c,) c)) в памяти ЭВМ

Последовательная, цепная и бинарная древовидная организации данных предназначены для решения общей задачи – обработки записей с одним ключевым атрибутом. Поскольку они взаимозаменяемы, имеет смысл задача выбора лучшей организации данных.

В табл. 3.1 собраны все оценки методов организации данных, что позволяет сделать ряд выводов.

По критерию времени формирования данных цепной каталог и бинарное дерево имеют определенные преимущества перед последовательным массивом, несмотря на то, что процессы формирования описываются одинаковыми формулами.

Таблица 3.1. **Сравнение методов организации данных**

Критерии оценки	Методы организации данных			*Лучший метод*
	последовательный	*цепной*	*бинарное дерево*	
Время формирования	~ M logM	~ M logM	~ M logM	цепной, бинарное дерево
Время поиска	~ logM	~ M	~ logM	последовательный, бинарное дерево
Время корректировки	~ M	~ M	~ logM	бинарное дерево
Объем дополнительной памяти	0	~ M	~ M	последовательный

Это объясняется необходимостью пересылки записей в процессе сортировки последовательного массива, а в цепном каталоге и бинарном дереве при формировании пересылаются адреса связи, а не целые записи.

По времени поиска последовательный массив и бинарное дерево предпочтительнее цепного каталога. Минимальное время корректировки характерно для бинарного дерева, а минимальный объем дополнительной памяти – для последовательного массива.

Мы приходим к окончательному выводу, что абсолютно безупречного метода организации данных не существует. Однако минимальное время обычно считается более важным критерием, чем минимальная дополнительная память, и тогда лучшим методом организации данных в оперативной памяти ЭВМ необходимо признать упорядоченное бинарное дерево.

Следует отметить, что для последовательной и цепной организации данных разработаны методы ускорения поиска, представленные в следующем параграфе, которые неприменимы к деревьям. Это в ряде случаев создает преимущества для последовательных массивов перед деревьями.

3.2
МЕТОДЫ УСКОРЕНИЯ ДОСТУПА К ДАННЫМ

Ускорение доступа к данным достигается применением принципиально иных методов размещения информации и ее поиска либо путем создания массивов вспомогательной информации о хранимых данных.

Эти же методы необходимы при организации доступа к информации по нескольким ключевым атрибутам одновременно.

Доступ к требуемым записям может осуществляться не только путем сравнения искомого значения ключа с ключами записей, извлекаемых из массива по определенному алгоритму (как это было в рассмотренных методах обработки данных), но и

в результате вычисления местоположения требуемой записи. Сами записи могут быть упорядочены алгоритмом сортировки либо используется специальная расстановка записей.

Адресная функция

Расстановка записей происходит в соответствии с так называемой адресной функцией (другие общеупотребительные ее названия – "рандомизирующая функция" и "хэш-функция"). Применяемые при этом методы организации данных часто называются *методами рандомизации*.

Адресной функцией называется зависимость

$$i=f(p),$$

где i – номер (адрес) записи;
p – значение ключевого атрибута в записи.

Адресная функция может вырабатывать одинаковое значение i для значений p, принадлежащих разным записям, которые в этом случае называются синонимами. К функции f предъявляются следующие требования:

- она должна быть задана аналитически и вычисляться достаточно быстро;
- ключевые атрибуты, подчиняющиеся произвольному распределению, функция должна переработать в равномерно распределенные номера записей; это условие обычно соблюдается приближенно;
- число записей-синонимов должно составлять 10–20% от общего числа записей.

Известно достаточно много адресных функций, хорошо соответствующих этим требованиям. Простейшая адресная функция имеет вид:

$$i = p - a,$$

где a – константа.

Пусть известны минимальное значение ключевого атрибута в массиве p′ и максимальное значение p″ . Тогда a = p′ – 1. Необходимый участок памяти для данных оценивается в p″ – p′ +1 запись. Записи-синонимы связываются в цепочки с помощью адресов связи, они занимают дополнительную (резервную) память.

Пример размещения записей с ключами 13, 11, 14, 11, 15,18, 14, 16 согласно адресной функции i = p – a показан на рис. 3.9.

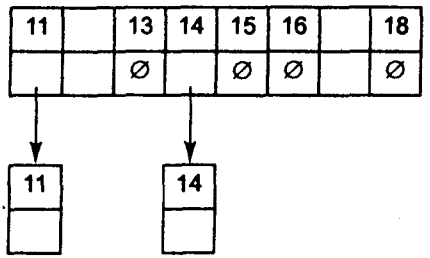


Рис. 3.9. Организация записей в соответствии с адресной функцией i = p – a

При доступе к записи с ключом q вычисляется i=f(q) и производится обращение к i-й записи. При необходимости с помощью адресов связи извлекаются все синонимы.

Недостаток адресной функции вида i = p – a – большой объем неиспользуемой памяти, если p″ – p′ много больше, чем количество записей M.

Указанного недостатка лишена функция вида

$$i = \text{ОСТ}(p/m).$$

Здесь m – целое число; ОСТ – остаток от деления p на m.

Вычисление i на языке Паскаль производится с помощью оператора i: =p mod m.

Значение m принимается равным простому числу, которое непосредственно больше либо непосредственно меньше числа записей M.

Выделяются 2 зоны памяти – основная и резервная. *Основная зона* содержит m записей. *Резервная зона* предназначена для размещения записей-синонимов.

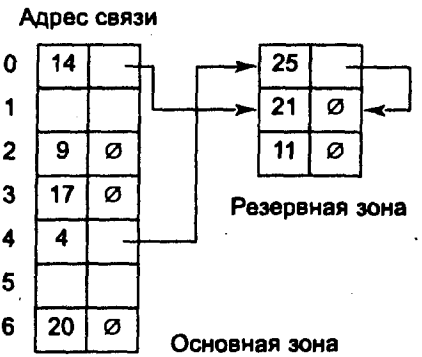


Рис. 3.10. Организация записей в соответствии с адресной функцией i = ОСТ (p/m)

При формировании данных согласно адресной функции сначала производится заполнение основной зоны. Если при этом позиция основной зоны, полученная при вычислении, уже занята, то текущая запись помещается в резервную зону и адресуется из этой позиции основной зоны. В дальнейшем при такой ситуации наращивается цепочка записей в резервной зоне.

Например, для массива ключей со значениями 17, 9, 4, 14, 25, 21, 20, 11 необходимо выбрать m=7, поскольку M=8 (возможно также m=19). Содержимое основной и резервной зон иллюстрирует рис.3.10. Резервная зона заполняется последовательно. При поиске значения, например q=11, вычисляется i = ОСТ(11/7)=4, и далее последовательно сравниваются 4 и 11, 25 и 11и т. д. В рассматриваемом примере число записей-синонимов составляет 3/8.

Индексы

Для ускорения поиска записей в массиве используется дополнительная информация, организованная в виде массива индексов.

Индексом называется набор ключей и адресов записей, которые выбираются из основного массива по определенному

закону. Отдельный элемент набора индексов также называется индексом, хотя это не соответствует значению слова index – список.

Имеются три важные разновидности индексов:

- информация о каждой записи основного массива попадает в индекс (сплошная индексация);
- номера записей, информация о которых выносится в индекс, образуют арифметическую прогрессию с шагом d>1. Основной массив, дополненный таким индексом, обычно называется индексно-последовательным;
- ключи записей, информация о которых выносится в индекс, приближенно образуют арифметическую прогрессию.

Сплошной индекс связан с созданием инвертированного массива ключевых атрибутов к основному массиву. Этот случай уже рассмотрен в п. 2.4.

Индексно-последовательный массив представляет собой последовательный массив, отсортированный по значениям ключевого атрибута, к которому создается дополнительный массив индексов.

В индекс выносится информация о записях, номера которых образуют арифметическую прогрессию с шагом d. На рис. 3:11 показаны ключевые атрибуты основного массива и состояние массива индексов для d=3.

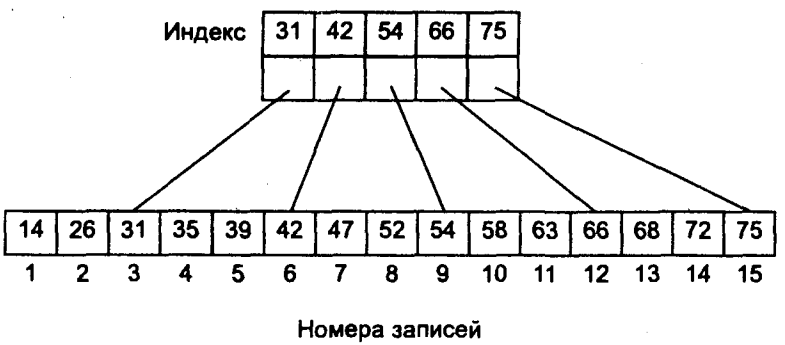


Рис. 3.11. Индексно-последовательная организация данных

Поиск значения q в индексно-последовательном массиве происходит в две стадии:

- в массиве индексов (который отсортирован в силу упорядоченности основного массива);
- среди записей основного массива, расположенных между двумя соседними индексами, найденными на первой стадии.

Применение индексов приводит к ускорению доступа, если основной массив располагается на внешнем запоминающем устройстве, а массив индекса может быть полностью размещен в оперативной памяти ЭВМ.

Если ключи записей, информация о которых выносится в индекс, приближенно образуют арифметическую прогрессию, мы получаем ситуацию с адресной функцией для индекса (рандомизация индекса).

Точное описание рандомизированного индекса состоит в следующем. Индекс с номером n хранит адрес записи основного массива, ключ которой равен или непосредственно больше значения

$$p(1)+z(n-1),$$

где z – константа (шаг арифметической прогрессии);
p(1) – значение ключа первой записи основного массива.

На рис. 3.12 показаны ключевые атрибуты основного массива (идентичные с предыдущим примером) и состояние массива рандомизированных индексов для z=10. Примечательно, что значения ключей в таком индексе хранить не нужно.

Рассмотрим поиск с использованием рандомизированных индексов. На первой стадии номер требуемого далее индекса определяется по формуле

$$n = \frac{q - p(1)}{z} + 1.$$

Результат деления округляется в меньшую сторону. Интервал записей основного массива на второй стадии поиска определяется адресами, указанными в n-м и (n+1)-м индексах.

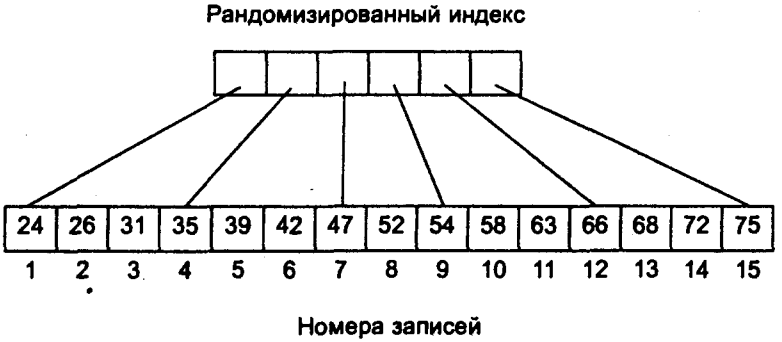


Рис. 3.12. Рандомизация индекса

Важной *разновидностью доступа к данным*, которая требует специальных методов ускорения доступа, является *обработка информации по нескольким ключевым признакам*. В структуре записи массива определяется несколько ключевых атрибутов, причем в различных прикладных программах требуется доступ к записям по различным сочетаниям этих атрибутов и, возможно, требуется последовательная обработка всего массива.

Среди ключевых атрибутов записи устанавливается порядок старшинства. Извлекаемые на обработку записи должны быть упорядочены в пределах всего массива по самому старшему ключу. В пределах группы записей с одинаковым значением старшего ключа должна соблюдаться упорядоченность по значениям следующего по старшинству ключа и т. д. Проще всего представить упорядоченность по нескольким ключам с помощью следующей схемы.

Пример

Рассмотрим записи с четырьмя атрибутами в порядке старшинства слева направо A1, A2, A3, A4. Значения A1 упорядочены (для примера) по возрастанию. На каждом множестве записей, которые соответствуют одинаковому значению A1, реализована упорядоченность по возрастанию значений A2 для записей, у которых значение A1 одинаково. Далее, на каждом множестве записей с одинаковыми парами значений атрибутов A1 и A2 соблюдается

упорядоченность значений по атрибуту А3. И, наконец, на каждом множестве записей с одинаковыми значениями атрибутов А1, А2, А3 должна соблюдаться упорядоченность по возрастанию для атрибута А4.

Последовательный массив с такой упорядоченностью может обеспечивать быстрый доступ к данным по следующим сочетаниям ключевых атрибутов a1+a2+a3+a4, a1+a2+a3, a1+a2 и a1.

Количество сочетаний атрибутов, необходимых для реализации максимально широкого круга запросов, в нашем примере составляет 15. Хранение нескольких по-разному рассортированных дублей массива или систематическая сортировка единственного массива в соответствии с поступающими запросами не является хорошим решением проблемы.

Рассмотрим возможности создания нескольких массивов индексов в этой ситуации. Индекс удобно формировать не для одного ключевого атрибута, а для набора атрибутов. Естественно, что индекс ключевых атрибутов a1+a2+a3+a4 может также использоваться для быстрого доступа по атрибутам a1+a2+a3, a1+a2 и a1. Поэтому в нашем примере максимально необходимо создание 4 индексов с упорядоченностью атрибутов a1+a2+a3+a4, a1+a2+a4, a1+a3+a4, a2+a3+a4.

Для доступа к данным по нескольким ключевым атрибутам используется также мультисписковая организация данных. *Мультисписком* называется множество списков, организованных на общем множестве записей. Если требуется доступ к записям по t ключам, то формируется t списков для каждого ключевого атрибута в отдельности.

Рассмотрим особенности организации мультисписков, которые предназначены для обработки записей по нескольким ключевым атрибутам.

Пример

В табл. 3.2 показаны 14 записей с ключевыми атрибутами Фамилия и Профессия (остальные атрибуты в данном случае несущественны). На пересечении строки с некоторой фамилией и столбца с некоторой профессией указан номер записи, которая содержит именно эти значения в качестве ключей.

В простейшем случае мультисписок будет содержать два списка – с указателем Фамилия – (A(1), A(2), A(3), ... , A(13), A(14)) и с указателем Профессия– (A(3), A(6), A(12), A(1), A(7), A(10), A(13), A(2), A(4), A(8), A(14), A(5), A(9), A(11)).

Таблица 3.2. **Мультисписок для двух ключевых атрибутов**

Фамилия	Профессия			
	слесарь	токарь	штамповщик	электрик
Бардин		A(1)	A(2)	
Басов	A(3)		A(4)	A(5)
Батов	A(6)	A(7)		
Белов			A(8)	A(9)
Иванов		A(10)		A(11)
Исаев	A(12)	A(13)	A(14)	

При размещении мультисписка во внешней памяти необходимо размещать каждый список в небольшом числе рядом расположенных участков, что позволит уменьшить время доступа.

Эффективная организация мультисписка предполагает выполнение следующих условий:

- число записей в каждом списке должно быть небольшим,
- адреса хранения записей должны монотонно возрастать.

Для сокращения длины списков в мультисписке необходимо детализировать содержимое указателей. Например, указатель Фамилия = "Ба" определяет список (A(1), A(2), A(3), A(4), A(5), A(6), A(7)), указатель Фамилия = "Бе" – список (A(8), A(9)), указатель Фамилия = "И" – список (A(10), A(11), A(12), A(13), A(14)). Поскольку атрибут Профессия содержит четыре значения, возможно существование следующих четырех списков: (A(3),A(6),A(12)); (A(1),A(7),A(10),A(13)); (A(2), A(4),A(8), A(14)); (A(5),A(9),A(11)).

При поиске в сокращенных списках необходимо сначала проанализировать все указатели, чтобы выбрать одну строку, заведомо содержащую требуемую информацию.

Рассмотрим запрос с условием

Фамилия ="Иванов" и Профессия = "электрик"

Потребуются три обращения к памяти для выбора списка (A(10), A,(11), A(12)),A(13), A(14)) и четыре обращения для выбора списка (A(5),A(9), A(11)). В указателях хранится длина каждого списка. Вторая строка короче, поэтому она просматривается полностью до извлечения нужной записи A(11).

3.3 ОРГАНИЗАЦИЯ ДАННЫХ ВО ВНЕШНЕЙ ПАМЯТИ ЭВМ

В качестве внешней памяти ЭВМ используются в основном устройства электромагнитной записи сигналов, для которых характерно примерное равенство затрат времени на чтение и запись информации, – *магнитные диски*. В отличие от оперативной памяти ЭВМ для них перед непосредственно чтением/записью требуется подвод необходимого участка магнитного носителя к механизму чтения/записи (в реальных запоминающих устройствах могут двигаться и носитель данных, и механизм чтения/записи). Поэтому время доступа к данным на внешнем запоминающем устройстве зависит от места расположения данных на диске или ленте, что существенно отличает их от оперативной памяти и определяет специфику организации данных во внешней памяти ЭВМ (рис. 3.13).

Данные на внешнем запоминающем устройстве хранятся в виде файлов. *Файл* представляет собой множество логически связанных записей. Запись обычно соответствует одному значению некоторой составной единицы информации. Каждый файл имеет уникальное *имя файла*. В простейшем случае файл представляет последовательный массив записей на внешнем запоминающем устройстве.

Анализ методов организации данных остается в основном справедливым и для данных во внешней памяти ЭВМ; однако

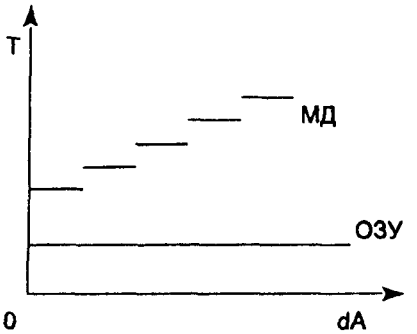


Рис. 3.13. *График зависимости времени доступа к данным от адресного расстояния*

серьезным фактором, влияющим на время доступа, становится взаимное расположение файлов и записей на магнитном носителе.

Определим адресное расстояние dA как разность адресов предыдущего и текущего обращения к запоминающему устройству, взятую со знаком "плюс".

$$dA = |A(i-1) - A(i)| .$$

Чтобы применять адресное расстояние ко всем типам запоминающих устройств, отметим, что с магнитного диска читается (записывается) не отдельный символ (байт), а сектор или блок данных размером, например, 512 байт.

Организация внешней памяти персональных ЭВМ имеет ряд отличий от принципов, используемых в мини-ЭВМ и средних ЭВМ. Вся внешняя память разделена на физические блоки (секторы), имеющие фиксированный размер (обычно 512 байт), который не зависит от желания проектировщика системы. Обмен с оперативной памятью происходит только целыми секторами.

Когда производится только последовательная обработка файла, оптимальный (с точки зрения минимального времени доступа) размер блока должен быть наиболее крупным из возможных; когда происходит только выборка одиночных записей, оптимальными являются блоки размером в одну запись.

Существует ряд стандартных методов организации файлов на магнитном диске и соответственно методов доступа к этим файлам. Среди них – последовательная, индексно-последовательная, индексно-произвольная и прямая организация файлов. Во всех случаях необходимо выделение в записях файла одного ключевого атрибута.

При *последовательной организации файла* на магнитном диске возможен доступ от только что обработанной записи к последующей записи (по направлению к концу файла). Переход в обратном направлении невозможен, единственный путь состоит в закрытии файла, повторном его открытии и движении к нужной записи в прямом направлении.

Индексно-последовательный файл представляет собой последовательный файл, снабженный индексами. На магнитном диске выделяются три области – первичная, индексная и область переполнения. В первичной области помещаются упорядоченные по значениям ключевого атрибута записи, когда файл впервые создается.

В зависимости от размера первичной области могут создаваться один, два или три уровня индексов:

- индекс первого уровня отмечает последнюю запись каждой дорожки магнитного диска;
- индекс второго уровня отмечает последнюю запись каждого цилиндра магнитного диска.

Если файл индекса второго уровня достаточно велик по размеру, то для него допускается создание индекса третьего уровня. *Область переполнения* предназначена для размещения записей, включаемых в индексно-последовательный файл. Новые записи связываются в цепочку и размещаются на том цилиндре, при котором ключи новых записей соответствуют интервалу значений ключей в первичной области этого цилиндра.

Перечислим *итоговые характеристики индексно-последовательного доступа*:

- значения ключей записей должны быть отсортированы;
- в индекс заносится наибольший ключ для всех записей блока (дорожки);

- наличие повторяющихся значений ключа недопустимо;
- эффективность доступа зависит от числа уровней индексации, распределения памяти для размещения индекса, числа записей в файле и размера области переполнения.

Если в индекс попадает информация о ключе каждой записи, то получаем *индексно-произвольный доступ*. Записи файла могут быть при этом не упорядочены по значению ключа. Индекс для индексно-произвольного метода доступа практически всегда формируется как многоуровневый. Типичная организация многоуровневого индекса соответствует понятию B-дерева. Нижний уровень B-дерева образуют индексы со ссылкой на каждую запись основного массива. Благодаря использованию адресных ссылок упорядоченность основного массива не обязательна. Индексы нижнего уровня разделены на страницы, и в конце каждой страницы оставляется резервная память. Последний индекс каждой страницы поступает на страницу предпоследнего уровня B-дерева. Когда эта страница будет почти заполнена индексами, последний из них поступит на страницу более высокого уровня и т. д.

Пользуясь аналогией между многоступенчатым поиском и многоуровневым индексом, можно утверждать, что минимальное время доступа обеспечивает B-дерево с числом уровней, примерно равным lnM (M – число записей в основном массиве).

Корень B-дерева обычно имеет два разветвления, так как при большем числе разветвлений происходит незначительное ускорение доступа. Пример B-дерева для 14 записей основного массива приводится на рис. 3.14.

Поиск данных (например, значения q) начинается с корня B-дерева. Предположим, что k8<q<k14. В этом случае выполняются переход по правой ветви на второй уровень и поиск на соответствующей странице. Будем считать, что k10<q<k13. Следовательно, надо спуститься на третий уровень по адресу, записанному в k13, и завершить поиск. Достоинство B-дерева состоит в его простом расширении. При переполнении какой-либо страницы половина ее содержимого переходит в новую страницу, а на вышестоящем уровне появляется новый индекс.

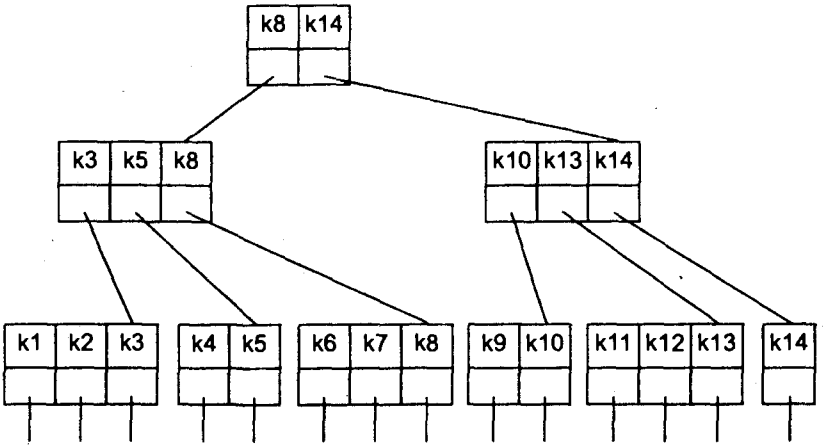


Рис. 3.14. Многоуровневый индекс в виде B-дерева

Прямой метод доступа соответствует файлу, который использует адресную функцию вида i=p–a, рассмотренную в п. 3.2. Для прямого доступа характерны следующие особенности:

- не требуется упорядоченность записей файла;
- наличие повторяющихся значений ключа недопустимо;
- значениям нескольких ключей может соответствовать один и тот же адрес (блок).

На выбор между названными выше методами организации файлов существенное влияние оказывает количество записей, которое должно быть обработано в процессе реализации запроса. Этот параметр называется долей выборки и равен отношению числа требуемых при выборке записей файла к общему числу записей в файле.

Информация о соответствии доли выборки записей и методах организации файла приводится ниже.

Доля выборки записей	**Наилучшая организация файла**
1-я запись	Прямая
0..10%	Прямая, индексная
10..100%	Последовательная

Блок данных на внешнем запоминающем устройстве обычно не заполняется полностью, т. е. оставляется резервная память (обычно 10–15% размера блока). Если этого не делать, то включение новых записей потребует создания для них новых блоков практически при каждой корректировке. Эти блоки будут содержать довольно мало записей, отчего резко возрастет объем дополнительной памяти, необходимый для массива.

Когда резервная память блока будет исчерпана и в него потребуется включить новую запись, наступает переполнение блока.

Частота переполнений описывается формулой

$$K = (V+1)/(2r - 1),$$

где K – ожидаемое число корректирующих обращений (включений и исключений записей) к одному блоку до наступления переполнения;

V – объем свободной памяти блока, выраженный в количестве записей;

r > 0,5 – вероятность того, что корректирующее обращение является включением.

Если r <= 0,5, то блок, как правило, никогда не переполнится. После переполнения блока вслед за ним в память включается новый блок, в который переписывается половина записей из переполненного блока.

Оптимальное вторичное индексирование

Многие СУБД и ППП предоставляют возможность создания и поддержания вторичных индексов для файлов, хранящихся на магнитном диске. *Увеличение числа вторичных индексов позволяет существенно сократить время реализации запросов, но параллельно возрастают затраты на корректировку индексов при внесении изменений в основной файл.*

Будем выделять в структуре файла первичный ключ и атрибуты, которые могут служить для построения вторичного индекса (вторичные ключи). Если ключ является многоатрибутным, то по его отдельным атрибутам может существовать вторичный индекс.

Нам известны типы запросов к файлу (определяемые атрибутами-входами запроса) и допустимые для файла стратегии поиска (определяемые именами вторичных ключей). Особое положение среди стратегий поиска занимает последовательный доступ к файлу, который не использует никаких индексных путей.

Затраты времени на реализацию поиска и корректировки данных в файле будем выражать количеством прочитанных или записанных блоков информации, предполагая, что длина блока одинакова и в основном файле, и в файлах-индексах. Обозначим через c(i,j) количество прочитанных блоков для удовлетворения j-го типа запроса при помощи i-й стратегии поиска. Очевидно, что стратегия не способна удовлетворить запрос, если атрибуты-входы запроса и вторичные ключи не содержат общих имен.

В этом случае c(i,j) равно количеству блоков информации в основном файле. Формулы для расчета числа блоков обычно приводятся в технической документации на СУБД и чаще всего имеют вид

$$ML/B,$$

где M,L – число записей в основном массиве и длина записи в байтах,
B – размер блока в байтах.

Когда атрибуты-входы запроса и вторичные ключи содержат общие имена, формулы для c(i,j) сильно зависят от состава общих имен. Пусть общим является атрибут A(x) с количеством различных значений m(x) и длиной значения l(x). Количество блоков в файле-индексе, построенном по атрибуту A(x), обозначим через n(x).

Формулы для расчета n(x) сильно различаются у разных СУБД из-за многообразия способов реализации индекса. Здесь будет использоваться формула

$$n(x) = (m(x)l(x) + M*l)/B,$$

где l – длина адресной ссылки на запись основного файла.

Величина c(i,j) определяется количеством чтений индекса (log n(x) или (n(x) + 1)/2 в зависимости от способа доступа) и количеством чтений из основного файла. Пренебрегая возможностью нахождения нескольких записей-целей в одном блоке, получаем выражение для числа считываемых блоков в виде r(j,x)*M/m(x). Если в условии j-запроса указано точное значение атрибута искомых записей, то r(j,x) = 1, если задается диапазон значений, то r(j,x) равно количеству значений в этом диапазоне. Итак,

$$c(i,j) = \log n(x) + r(j,x)*M/m(x).$$

Корректировка данных в файле разделяется на включение/исключение записи и замену значения атрибута в записи. Количество обрабатываемых блоков при включении c'(i) = log n(i) + 1, при исключении c"(i) = c'(i).

При рассмотрении корректировок в файле с индексами надо различать внесение изменений в запись с известным значением первичного ключа (собственно корректировка) и по известному значению вторичного ключа (доступ через индекс). В первом случае необходимо ввести дополнительный параметр c'(i,k). Он представляет собой количество обрабатываемых блоков для k-го типа обновления при помощи i-й стратегии поиска. Если обновление не затрагивает соответствующий индекс, то c'(i,k) = 0. Когда c'(i,k) # 0, то c'(i,k) = 2(log n(i) + 1).

Во всех указанных формулах округление производится после каждого действия в большую сторону.

Для существующих типов запросов вводятся соответствующие вероятности q(j), аналогичные вероятности u(k)

k = 1...s описывают варианты обновлений. Вероятность включения записи обозначается через I, вероятность исключения – через D.

Критерий эффективности применяемой системы вторичных индексов минимизирует затраты на проведение произвольного запроса и произвольного корректирующего обращения.

$$\min F(i) = \Sigma\, q(j)c(i,j) + \Sigma\, u(k)c'(i,k) + Ic'(i) + Dc''(i).$$

Время внесения изменений в основной файл предполагается постоянным и поэтому в F(i) не входит.

Дополнительно должно соблюдаться соотношение

$$q(i) + u(k) + I + D = 1.$$

Переменная i описывает возможные стратегии поиска. Максимальное число стратегий составляет 2^N–1, где N – число атрибутов, которые могут быть использованы при вторичном индексировании.

Если запросы можно разделить на t групп, так что множества атрибутов из каждой группы не пересекаются, то исходную задачу можно разделить на t задач. Для снижения размерности задачи можно также исключать из рассмотрения индексные пути, которые используются в запросах с суммарной частотой менее 0,05. Запросы, для которых количество считываемых из основного файла блоков всегда превышает 0,1 от их общего количества, также не должны анализироваться, так как они быстрее реализуются в режиме последовательного чтения файла.

Если производятся только запросы и очень мала доля времени на корректировку, то все возможные индексные пути должны включаться в оптимальное решение. Если количество запросов незначительно (преобладает последовательная обработка) и достаточно много корректирующих обращений, то индексные пути вообще не нужны.

ВОПРОСЫ И ЗАДАНИЯ

1. Как можно использовать упорядоченные бинарные деревья для подсчета частоты встречаемости слов в тексте?

2. Какими способами можно объединить два упорядоченных бинарных дерева в одно? Выберите из них лучший способ и представьте соответствующий алгоритм.

3. Во многих задачах, где взаимосвязь данных соответствует понятию сети, числовые значения приписываются не вершинам сети, а ее дугам. Как представить эти данные в виде таблицы? Можно ли значения дуг передать вершинам и какие неудобства это вызовет?

4. Какова вероятность того, что массив из M записей случайно окажется упорядоченным?

5. Изобразите однонаправленные и двунаправленные представления в памяти ЭВМ следующих списков:

а) ((a,b),(c,a),d)

б) ((a,(b,c)),(d))

6. Для последовательного массива и упорядоченного бинарного дерева известен алгоритм поиска по совпадению. Как использовать этот алгоритм для поиска по условию p(i)>q?

ГЛАВА 4

МОДЕЛИРОВАНИЕ ПРЕДМЕТНЫХ ОБЛАСТЕЙ В ЭКОНОМИКЕ

4.1 СЕМАНТИЧЕСКИЕ МОДЕЛИ ДАННЫХ

Известные средства описания данных ориентируются на формы представления информации (синтаксические модели данных) или смысловые характеристики информации (семантические модели). Синтаксическими являются модели, рассмотренные в гл. 2.

Семантические модели данных представляют собой средство представления структуры предметной области. Такие модели имеют много общего с иерархическими и сетевыми моделями данных, они могут использоваться как средство построения структуры соответствующих баз данных.

Семантические модели должны отвечать следующим требованиям:

- обеспечить интегрированное представление о предметной области;
- понятийный аппарат модели должен быть понятен как специалисту предметной области, так и администратору БД;
- модель должна содержать информацию, достаточную для дальнейшего проектирования ЭИС.

Семантические модели данных используют общий набор понятий и отличаются конструкциями, применяемыми для их выражения, полнотой отражения понятий в модели, удоб-

ством использования при разработке ЭИС. Как эталон семантической полноты рассматривается естественный язык, а для формализации языковых конструкций в моделях применяется аппарат математической лингвистики.

Рассмотрим *конструкции естественного языка*, декомпозиция которых невозможна без утраты смысла, т. е. высказывания. Структура высказываний оказывается достаточной для выражения закономерностей, присутствующих в предметной области и ЭИС.

Элементами высказываний служат атомарные факты. Способ представления атомарного факта состоит в указании объектов, их взаимодействий и свойств, которые описывают событие, соответствующее атомарному факту, а также указании времени наступления этого события.

Объекты могут быть атомарными и составными. *Атомарный объект* – это любой объект, разложение которого на другие объекты в рамках данной предметной области не производится. Составные объекты содержат так или иначе организованные множества объектов. Рекурсивно применяя это определение, можно получить произвольную структуру объектов и фактов и рассматривать ее как составной объект. Информация о том, что объект имеет некоторое свойство или несколько объектов взаимосвязаны, представляется в виде высказывания об объекте (или группе объектов).

Существуют правила вывода новых свойств и связей из ранее определенных свойств и связей. Конъюнкция двух свойств является новым свойством. Свойства могут образовывать комбинации и наследоваться через связи.

Объект может существовать независимо от того, определены или нет свойства и связи, относящиеся к этому объекту. Обязательное свойство, необходимое для определения существующего объекта, – это время его появления и время его исчезновения (как элемента информационных потребностей пользователей ЭИС).

Атомарный факт представляется тремя компонентами:

(x, y, t),

где x – множество объектов O1, O2,..., Ok;
y – свойство или связь объектов;
t – время.

Объект может быть составным, т. е. построенным как множество других объектов и, возможно, атомарных фактов.

Объекты могут вступать в отношения двух типов – обобщения, когда один объект определяется в виде множества других объектов, и агрегации, когда объект соотносится с именем действия, в котором он может участвовать. Например, объект Личность обобщает такие объекты, как Рабочий, Служащий, Студент; объект Транспорт агрегируется с действием Перевозка. Обобщения и агрегации могут образовывать иерархические структуры.

Семантические модели данных обычно предполагают два уровня интерпретации, уровень объектов предметной области и уровень атрибутов базы данных. Оба уровня при необходимости можно совместить в одном представлении.

Известно достаточно большое число семантических моделей данных (например, модель "сущность-связь", модель семантических сетей и др.); однако используемые в них понятия, идеи и методы характеризуются большим сходством, что облегчает их совместное рассмотрение.

Модель сущностей и связей

Наиболее распространенной семантической моделью является модель, названная "сущность-связь". Эта модель использует графическое представление всех компонентов. Базовыми элементами в модели "сущность-связь" служат типы сущностей, обозначаемые далее прямоугольниками, и типы связей, обозначаемые двойными прямоугольниками. Многие сущности, рассматриваемые в этой модели, соответствуют физическим объектам предметной области.

Структура предметной области в модели "сущность-связь" изображается в форме диаграммы. Дуги на диаграмме соединяют тип сущности с типом связи. На дугах указывается 1 или m в соответствии с тем, сколько раз идентификатор объекта может возникнуть в строках отношений, представляющих связи объектов (1 – один раз, m – несколько раз).

Диаграмма может представлять только объекты и связи или дополнительно содержать атрибуты, описывающие их свойства.

В структуре связей объектов допускаются следующие типы связей:

- N-арные связи (рис. 4.1,а), приводится пример тернарной связи;
- рекурсивные связи (рис. 4.1,б);
- несколько связей для одной и той же пары объектов (рис. 4.1,в).

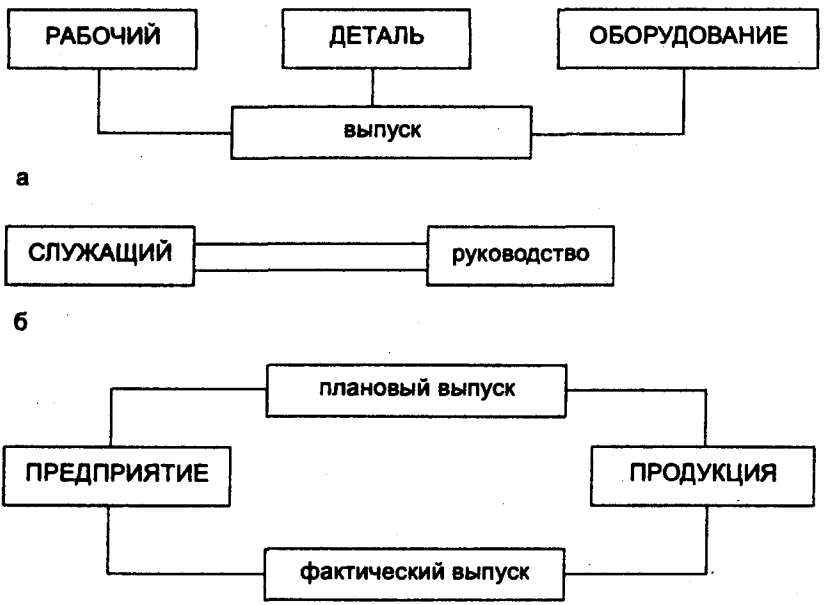


Рис. 4.1. Варианты соответствий между сущностями и связями:
а – N-арные связи; б – рекурсивные связи;
в – несколько связей для одной и той же пары объектов

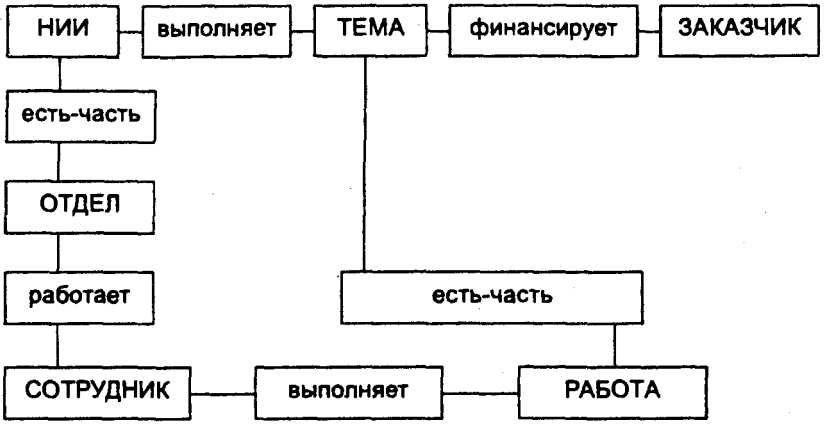


Рис. 4.2. Модель "сущность-связь" для базы данных, рассмотренной в п. 2.2.2

В качестве примера списания предметной области средствами модели "сущность-связь" на рис. 4.2 показано представление, использованное в примерах к гл. 2 (метки дуг 1 и m не показаны).

Представление предметной области с помощью модели "сущность-связь" позволяет:

- однозначно разработать структуру многоуровневой сетевой базы данных;
- обеспечить одинаковое понимание всеми пользователями содержимого базы данных.

Модель "сущность-связь" характеризуется рядом недостатков:

- не содержит изобразительных средств для фиксации организационной иерархии процессов управления и агрегации данных по уровням управления;
- принятое в модели правило формирования множества отношений базы данных создает слишком много отношений для объектов и связей. В результате диаграмма объектов и связей реальной задачи быстро становится громоздкой и необозримой.

Возможности выражения семантики в терминах сущностей и связей весьма ограничены. Повышение выразительной силы изобразительных средств достигается в модели семантической сети. В определенном смысле возможности семантической сети являются наиболее универсальными из известных к настоящему времени.

Поэтому приложения аппарата семантических сетей целесообразно разделить на две ветви – для обеспечения осмысленности информации, хранимой в базе данных (что и рассматривается далее в этом параграфе), и для представления знаний любой природы (рассматривается в п. 4.2).

Семантические сети применительно к задачам проектирования структуры базы данных ЭИС используются в сравнительно узком диапазоне – для представления структуры понятий и структуры событий.

Семантические сети представляют собой ориентированные графы с помеченными дугами.

Они позволяют структурировать имеющуюся информацию и знания. Аппарат семантических сетей является естественной формализацией ассоциативных связей, которыми пользуется человек при извлечении каких-то новых фактов из имеющихся. Построение сети способствует осмыслению информации и знаний, поскольку позволяет установить противоречивые ситуации, недостаточность имеющейся информации и т. п.

Обычно в семантической сети предусматриваются четыре категории вершин:

- понятия (объекты),
- события,
- свойства,
- значения.

Понятия представляют собой константы или параметры, которые определяют физические или абстрактные объекты. *События* представляют действия, происходящие в реальном мире, и определяются указанием типа действия и ролей, кото-

рые играют объекты в этом действии. *Свойства* используются для представления состояния или для модификации понятий и событий.

Сведения семантической сети образуют сценарий, который является набором понятий, событий, причинно-следственных связей. Применительно к базе данных сценарий может рассматриваться как шаблон, которому должна соответствовать хранимая информация, чтобы обеспечивалась ее осмысленность.

Необходимо различать *вершины*, обозначающие экземпляры объектов, и *вершины*, представляющие классы объектов. Например, Новиков – экземпляр типа Студент. В семантической сети экземпляр может принадлежать более чем одному классу (Новиков – и Студент, и Спортсмен). Различные роли Новикова отображаются его принадлежностью к различным классам. Новиков – студент в своих связях с преподавателями и дисциплинами, а в отношениях с тренером и командой он – спортсмен.

В других моделях в отличие от семантической сети типы объектов указаны в схеме, а экземпляры объектов представлены значениями в базе данных. В семантической сети один и тот же экземпляр объекта может быть соотнесен с несколькими типами.

В синтаксических моделях (реляционной, сетевой или иерархической) для обеспечения такой связи потребуется дублирование информации об объекте.

Различие между вершинами сети (представление экземпляра и представление класса) приводит к существованию трех типов дуг:

- дуга, соединяющая два экземпляра, соответствует утверждению,
- дуга между классом и экземпляром показывает пример элемента класса,
- дуга, связывающая два класса, определяет бинарное отношение классов.

Все семантические отношения предметной области можно разделить на следующие:

- лингвистические,
- логические,
- теоретико-множественные,
- квантификационные.

Лингвистические отношения бывают глагольные (время, вид, род, число, залог, наклонение) и атрибутивные (модификация, размер, форма). *Логические отношения* подразделяются на конъюнкцию, дизъюнкцию, отрицание и импликацию.

Теоретико-множественные отношения – это отношение подмножества, отношение части и целого, отношение множества и элемента. Эти отношения обладают свойством транзитивности.

Квантификационные отношения делятся на логические кванторы общности и существования, нелогические кванторы ("много", "несколько") и числовые характеристики.

Основой для определения того или иного понятия является множество его отношений с другими понятиями.

Обязательными отношениями являются:

- класс, к которому принадлежит данное понятие,
- свойства, выделяющие понятие из всех понятий данного класса,
- примеры данного понятия.

Поскольку термины, использованные в определении понятия, сами служат понятиями, то их определение организуется по той же схеме. В итоге связи понятий образуют структуру, в общем случае сетевую.

Существуют две обязательные связи при установлении структуры понятий:

- связь "есть-нек" (от слов "есть некоторый"). Направлена от частного понятия к более общему и показывает принадлежность элемента к классу,
- связь "есть-часть". Показывает, что объект содержит в своем составе разнородные компоненты (объекты), не подобные данному объекту.

Пример семантической сети для описания структуры понятия "юридическое лицо" приведен на рис. 4.3. Одинарными

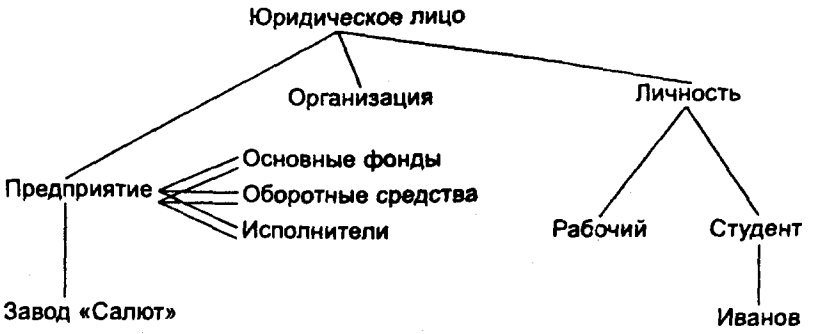


Рис. 4.3. Пример семантической сети для отображения связи понятий

линиями показаны связи "есть-нек", двойными линиями – связи "есть- часть". В семантической сети с помощью связи "есть-нек" можно показывать ссылку на экземпляр объекта.

Рассмотрим теперь представление событий и действий с помощью семантической сети. Предварительно выделяются простые отношения, которые характеризуют основные компоненты события. В первую очередь из события выделяется действие, которое обычно описывается глаголом. Далее необходимо определить объекты, которые действуют, объекты, над которыми эти действия производятся, и т. д. Все эти связи предметов, событий и качеств с глаголом называются падежами. Обычно рассматривают следующие падежи:

- агент – предмет, являющийся инициатором действия;
- объект – предмет, подвергающийся действию;
- источник – размещение предмета перед действием;
- приемник – размещение предмета после действия;
- время – указание на то, когда происходит событие;
- место – указание на то, где происходит событие;
- цель – указание на цель действия.

На рис. 4.4 приводится семантическая сеть, описывающая структуру события "Директор завода "САЛЮТ" остановил 25.03.96 цех № 4, чтобы заменить оборудование".

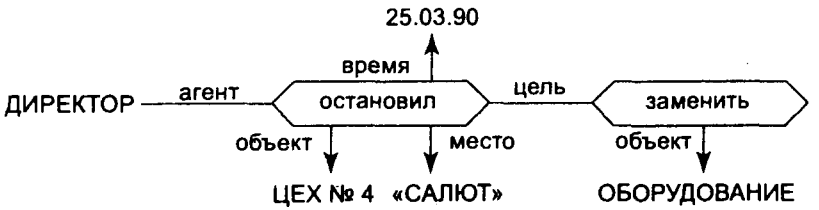


Рис. 4.4. Пример семантической сети для отображения связи событий

Необходимо отметить ряд преимуществ семантических сетей:

• описание объектов и событий производится на уровне, очень близком к естественному языку;

• обеспечивается возможность сцепления различных фрагментов сети;

• в семантической сети возможные отношения между понятиями и событиями образуют достаточно небольшое и хорошо формализованное множество;

• для каждой операции над данными и знаниями можно выделить из полной сети, представляющей всю семантику (или все знания), некоторый участок семантической сети, который охватывает необходимые в данном запросе смысловые характеристики.

В настоящее время ведутся теоретические исследования семантических моделей данных для обеспечения их совместимости с программными спецификациями запросов к базе данных. В итоге описания структуры информации и алгоритмов используют общий понятийный аппарат.

4.2 БАЗЫ ЗНАНИЙ

В современных системах управления вопрос о принятии решений информационной системой требует фиксации знаний об управляемом объекте и реализации моделей принятия решений, характерных для человека-специалиста (инженера, технолога, экономиста, бухгалтера). Способность человека на-

капливать и использовать знания, принимать решения можно назвать естественным интеллектом, соответствующие возможности информационной системы получили название *искусственный интеллект*.

Система понятий для представления знаний существенно отличается от понятий для представления данных, поэтому отображение знаний производится в базу знаний. Вместе с тем база знаний способна хранить данные как простую разновидность знаний.

Запросы, которые формулируются пользователями информационной системы, реализуются одним из двух возможных способов:

- сообщения, являющиеся ответом на запрос, хранятся в явном виде в БД, и процесс получения ответа представляет собой выделение подмножества значений из файлов БД, удовлетворяющих запросу;
- ответ не существует в явном виде в БД и формируется в процессе логического вывода на основании имеющихся данных.

Последний случай принципиально отличается от рассмотренной ранее технологии использования баз данных и рассматривается в рамках представления знаний, т. е. информации, необходимой в процессе вывода новых фактов.

База знаний содержит:

- сведения, которые отражают существующие в предметной области закономерности и позволяют выводить новые факты, справедливые в данном состоянии предметной области, но отсутствующие в БД, а также прогнозировать потенциально возможные состояния предметной области;
- сведения о структуре ЭИС и БД (метаинформация);
- сведения, обеспечивающие понимание входного языка, т. е. перевод входных запросов во внутренний язык.

Принято говорить не о "знаниях вообще", а о знаниях, зафиксированных с помощью той или иной модели знаний.

Принципиальными различиями обладают три модели представления знаний – продукционная модель, модель фреймов и модель семантических сетей.

Продукционная модель состоит из трех основных компонентов:

- набора правил, представляющего собой в продукционной системе базу знаний;
- рабочей памяти, в которой хранятся исходные факты и результаты выводов, полученных из этих фактов;
- механизма логического вывода, использующего правила в соответствии с содержимым рабочей памяти и формирующего новые факты.

Каждое правило содержит условную и заключительную части. В *условной части правила* находится одиночный факт либо несколько фактов (условий), соединенных логической операцией "И".

В *заключительной части правила* находятся факты, которые необходимо дополнительно сформировать в рабочей памяти, если условная часть правила является истинной.

Пример

Предположим, что в рабочей памяти хранятся следующие факты:

- доля выборки записей равна 0,09;
- ЭВМ – PC XT.

Правила логического вывода имеют вид:

1. Если метод доступа индексный, то СУБД – dBASE 3.
2. Если метод доступа последовательный, то СУБД – dBASE 3.
3. Если доля выборки записей <0,1, то метод доступа – индексный.
4. Если СУБД – dBASE 3 и ЭВМ – PC XT, то программист – Иванов.

Механизм вывода сопоставляет факты из условной части каждого правила с фактами, хранящимися в рабочей памяти. В данном примере сопоставление условия правила 3 с фактами из рабочей памяти приводит к добавлению нового факта "Метод доступа – индексный" и исключению правила 3 из списка применяемых правил.

С учетом нового факта становится справедливой условная часть правила 1, и в рабочей памяти появляется факт "СУБД – dBASE 3". Далее становится применимым правило 4, что приво-

дит к фиксации в рабочей памяти факта "Программист – Иванов". В этот момент дальнейшее применение правил невозможно, и процесс вывода останавливается. Наш пример показывает, что применимость каждого правила из базы знаний в процессе вывода вовсе не обязательна.

Новые факты, полученные механизмом вывода:

- метод доступа – индексный,
- СУБД – dBASE 3,
- программист – Иванов.

В приведенном примере для получения вывода правила применялись к фактам, записанным в рабочей памяти, и в результате применения правил добавлялись новые факты. Такой способ действий называется *прямым выводом*. Возможен также обратный вывод целей. В качестве цели выступает подтверждение истинности факта, отсутствующего в рабочей памяти. При обратном выводе исследуется возможность применения правил, подтверждающих цель, необходимые для этого дополнительные факты становятся новыми целями и процесс повторяется.

Предположим, что в нашем примере запрос цели имеет вид:

? "программист – Иванов".

Эта цель подтверждается правилом 4. Необходимые для правила 4 факты – "ЭВМ – PC XT" и "СУБД – dBASE 3". Первый из них присутствует в рабочей памяти, а второй становится новой целью. Для этой цели требуется подтверждение правила 1 или правила 2. Факт-условие правила 2 не содержится в рабочей памяти и не является заключением существующих правил. Поэтому данная ветвь обратного вывода обрывается. Для применения правила 1 необходим факт "Метод доступа – индексный", он является заключением правила 3, а условие правила 3 соблюдается (в рабочей памяти хранится факт "Доля выборки записей равна 0.09").

В итоге первоначальная цель "программист – Иванов" признается истинной.

В случае обратного вывода условием останова системы является окончание списка правил, которые относятся к доказываемым целям. При прямом выводе останов происходит по окончании списка применимых правил. Следует отметить, что на каждом шаге вывода количество одновременно примени-

мых правил может быть любым (в отличие от примеров, приведенных выше). Последовательность выбора подходящих правил не влияет на однозначность получаемого ответа; однако может существенно увеличить требуемое число шагов вывода. В реальных базах знаний с большим числом правил это может существенно снизить быстродействие системы. В системах с обратным выводом есть возможность исключить из рассмотрения правила, не имеющие отношения к выводу требуемых целей, и тем самым несколько ослабить указанный отрицательный эффект. По этой причине системы с обратным выводом целей получили большее распространение.

Рассмотрим **пример** представления фактов, правил и запросов на языке Пролог.

Компонентами программы на Прологе являются предикаты (predicates), цель (goal) и условия (clauses). *Предикатами* называются высказывания, составные части которых являются переменными (в нашем примере – символическими именами). Цель – это последовательность фактов, истинность которых должен доказать механизм логического вывода. В Прологе используется обратный способ вывода цели. Условия могут быть либо предикатами с конкретными значениями символических имен, либо правилами (заключение в Прологе пишется слева от знака :-, условия, связанные логическими операциями "И", – через запятую справа).

```
predicates
ruk(symbol,symbol)
nach(symbol,symbol)
print1(symbol,symbol)
goal
write("Введите фамилию "), nl, readln(S), write("Список начальников"), nl, write("для ",S), nl, print1(Q,S).
clauses
ruk("Петров","Иванов").
ruk("Петров","Смирнов").
ruk("Яшин","Петров").
nach(X,Y) :- ruk(X,Y).
nach(X,Y) :- ruk(X,Z),nach(Z,Y).
print1(Q,S) :- nach(Q,S),write(Q),nl,fail.
```

В этом примере введены предикаты:

ruk(X,Y) – для представления в базе знаний факта, что X является непосредственным руководителем для Y,

nach(X,Y) – для представления в базе знаний факта, что X является руководителем для Y,

print1(Q,S) – для печати всех руководителей сотрудника S.

Утверждения содержат факты о конкретных непосредственных руководителях и правила для предикатов. Раздел цели позволяет ввести фамилию сотрудника и затем напечатать список всех его начальников.

Сведения о всех начальниках в нашем примере не хранятся в явном виде в базе знаний и являются результатом логического вывода.

Представление знаний в виде набора правил имеет следующие преимущества:

- простота создания и понимания отдельных правил;
- простота механизма логического вывода.

К недостаткам этого способа организации базы знаний относятся неясность взаимных отношений правил и отличие от человеческой структуры знаний.

Фреймы

В основе теории фреймов лежит фиксация знаний путем сопоставления новых фактов с рамками, определенными для каждого объекта в сознании человека. Структура в памяти ЭВМ, представляющая эти рамки, называется *фреймом*. С помощью фреймов мы пытаемся представить процесс систематизации знаний в форме, максимально близкой к принципам систематизации знаний человеком.

Фрейм представляет собой таблицу, структура и принципы организации которой являются развитием понятия отношения в реляционной модели данных. Новизна фреймов определяется двумя условиями:

- имя атрибута может в ряде случаев занимать в фрейме позицию значения,
- значением атрибута может служить имя другого фрейма или имя программно реализованной процедуры.

Структура фрейма показана ниже. *Слотом фрейма* называется элемент данных, предназначенный для фиксации знаний об объекте, которому отведен данный фрейм.

Перечислим параметры слотов.

Имя слота. Каждый слот должен иметь уникальное имя во фрейме, к которому он принадлежит. Имя слота в некоторых случаях может быть служебным. Среди служебных имен отметим имя пользователя, определяющего фрейм; дату определения или модификации фрейма; комментарий.

Указатель наследования. Он показывает, какую информацию об атрибутах слотов во фрейме верхнего уровня наследуют слоты с теми же именами во фрейме нижнего уровня. Приведем типичные указатели наследования:

S (тот же). Слот наследуется с теми же значениями данных;

U (уникальный). Слот наследуется, но данные в каждом фрейме могут принимать любые значения;

I (независимый). Слот не наследуется.

Указатель типа данных. К типам данных относятся:

FRAME (указатель) – указывает имя фрейма верхнего уровня;

ATOM (переменная),

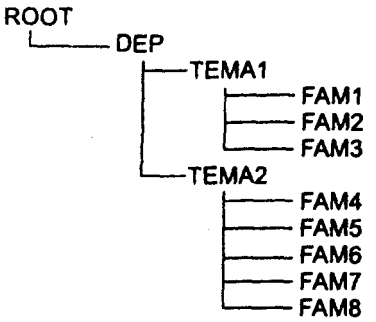
TEXT (текстовая информация),

LIST (список),

LISP (присоединенная процедура).

С помощью механизма управления наследованием по отношениям "есть-нек" осуществляются автоматический поиск и определение значений слотов фрейма верхнего уровня и присоединенных процедур.

Рассмотрим **пример** использования системы фреймов. Иерархия фреймов, показанная на рис. 4.5, отображает организационную структуру и работы, выполняемые в некотором отделе конструкторского бюро. Она предназначена для фиксации факта оконча-



а

Имя слота	Указатель наследования		Указатель типа	Значение слота
FRAME-NAME: DEP				
AKO	(U)	ROOT	FRAME	ROOT
DESINF	(U)	ROOT	TEXT	(ОТДЕЛ 23)
DATE	(U)	ROOT	LIST	
TEMA	(I)	•TOP•	LIST	(TEMA1 TEMA2)
TEMA1	(I)	•TOP•	LIST	NIL
TEMA2	(I)	•TOP•	LIST	NIL
FLAG1	(I)	•TOP•	ATOM	
FLAG2	(I)	•TOP•	ATOM	
LOGIC	(U)	•TOP•	LISP	MAIN
FRAME-NAME: TEMA1				
AKO	(U)	ROOT	FRAME	DEP
DESINF	(U)	ROOT	TEXT	(КОНСТРУИРОВАНИЕ ПЛЕЕРА)
DATE	(U)	ROOT	LIST	
FAM	(I)	•TOP•	LIST	(FAM1 FAM2 FAM3)
FAM1	(I)	•TOP•	LIST	NIL
FAM2	(I)	•TOP•	LIST	NIL
FAM3	(I)	•TOP•	LIST	NIL
FLAG1	(I)	•TOP•	ATOM	
FLAG2	(I)	•TOP•	ATOM	
FLAG3	(I)	•TOP•	ATOM	
LOGIC	(U)	•TOP•	LISP	COMP1
FRAME-NAME: FAM1				
AKO	(U)	ROOT	FRAME	TEMA1
DESINF	(U)	ROOT	TEXT	(ЛЕНТОПРОТЯЖНЫЙ МЕХАНИЗМ)
DATE	(U)	ROOT	LIST	
TODAY	(I)	•TOP•	ATOM	
ENDDATE	(I)	•TOP•	ATOM	02.04.91
LOGIC	(U)	•TOP•	LISP	COMPDATE

б

Рис. 4.5. Пример базы знаний фреймового типа:
а – иерархия фреймов; б – значения слотов

ния отдельных работ исполнителями, группами и отделом в целом. Фрейм ROOT является стандартным фреймом, все другие фреймы должны быть подчинены ему. Слот AKO используется для установления иерархии фреймов.

Работа начинается посредством передачи сообщения в слот LOGIC фрейма DEP. При этом запускается присоединенная процедура MAIN, которая передает в фреймы нижнего уровня значение текущей даты. Когда происходит заполнение какого-то слота в фрейме, делается попытка дать значения всем слотам этого фрейма, в том числе попытка выполнения присоединенной процедуры. Если запускается присоединенная процедура COMPDATE фрейма FAM1, она сравнивает текущую дату TODAY и дату окончания работы ENDDATE; в случае наступления срока окончания работы слот FLAG в вышестоящем фрейме TEMA1 получает значение 1. Поскольку текущая дата была передана во все фреймы нижнего уровня, присоединенные процедуры этих фреймов заполнят слоты FLAG в фреймах предшествующего уровня, что дает возможность запуска присоединенных процедур фреймов этого уровня. Каждая присоединенная процедура (например, процедура COMP1 фрейма TEMA1) анализирует атомы FLAG своего фрейма и, если все они равны 1, передает в фрейм DEP значение FLAG, равное 1. Обращение к системе фреймов ROOT позволяет определить наступление сроков окончания работ сотрудниками отдела.

Фреймовые системы обеспечивают *ряд преимуществ* по сравнению с продукционной моделью представления знаний:

- знания организованы на основе концептуальных объектов;
- допускается комбинация представления декларативных (как устроен объект) и процедурных (как взаимодействует объект) знаний;
- иерархия фреймов вполне соответствует классификации понятий, привычной для восприятия человеком;
- система фреймов легко расширяется и модифицируется.

Трудности применения фреймовой модели знаний в основном связаны с программированием присоединенных процедур.

Семантические сети для представления знаний

Особенность семантической сети как модели знаний состоит в единстве базы знаний и механизма вывода новых фактов. На основании вопроса к базе знаний строится семантическая сеть, отображающая структуру вопроса, и ответ получается в результате сопоставления общей сети для базы знаний в целом и сети для вопроса.

Рассмотрим **пример** семантической сети, отображающий подчиненность сотрудников в отделе учреждения, приведенный на рис. 4.6,а. Приводятся связи, показывающие подчиненность первого сотрудника. Остальные сотрудники отдела связываются через вершины сети связями типа "руководит 2", "руководит 3" и т. д.

Вопрос "Кто руководит Серовым?" представляется в виде подсети, показанной на рис. 4.6,б. Сопоставление общей сети с сетью запроса начинается с фиксации вершины "руководит", имеющей ветвь "объект", направленную к вершине "Серов". Затем производится переход по ветви "руководит", что и приводит к ответу "Петров".

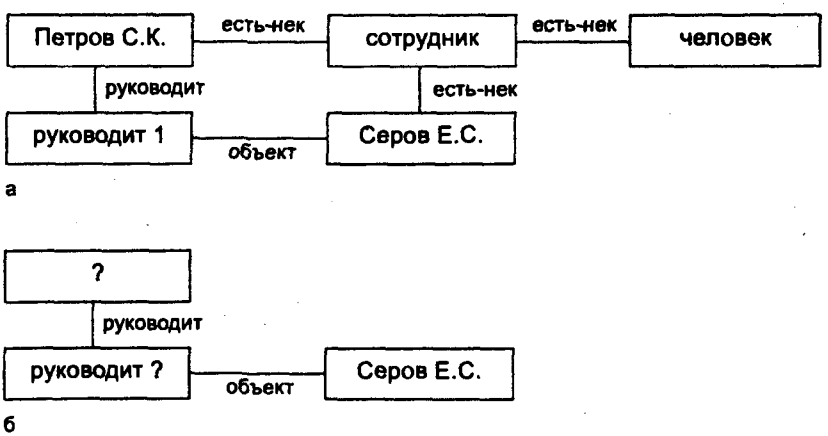


Рис. 4.6. Примеры: а – семантической сети;
б – сети логического вывода для запроса

Преимущества семантических сетей состоят в том, что это достаточно понятный способ представления знаний на основе отношений между вершинами и дугами сети. Однако с увеличением размеров сети ухудшается ее обозримость и увеличивается время вывода новых фактов с помощью механизма сопоставления.

Модели знаний – продукционная, фреймовая и модель семантических сетей – обладают практически равными возможностями представления знаний, использующих отношения "есть-нек" и "есть-часть". Дополнительно каждая модель знаний содержит средства усиления этой "базовой" конфигурации:

- продукционная модель позволяет легко расширять и усложнять множество правил вывода;
- фреймовая модель позволяет усилить вычислительные аспекты обработки знаний за счет расширения множества присоединенных процедур;
- модель семантических сетей позволяет расширять список отношений между вершинами и дугами сети, приближая выразительные возможности сети к уровню естественного языка.

4.3 ТЕЗАУРУСЫ ЭКОНОМИЧЕСКОЙ ИНФОРМАЦИИ

Первоначально идея разработки тезаурусов возникла в словарной практике в связи с составлением толковых словарей. Тезаурусы использовались как средство описания семантической структуры естественного языка. Затем они были применены в практике автоматизированных информационно-поисковых систем.

Термин "тезаурус" является общеупотребительным и общепринятым термином как элемент информационного языка, позволяющего фиксировать парадигматические отношения и отношения синонимии между понятиями предметной области.

Для обеспечения возможно большей полноты задачи при информационном поиске, основанном на дескрипторных информационно-поисковых языках (ИПЯ), необходимо избыточное индексирование документов и информационных запросов.

Под *избыточным индексированием* понимается дополнение поискового образа документов или поискового предписания дополнительными дескрипторами, которые связаны по смыслу с основными дескрипторами. При этом более предпочтительным считается избыточное индексирование не документов, а информационных запросов.

Для того чтобы можно было производить избыточное индексирование, необходимо преобразование алфавитного словаря дескрипторов в нормативный словарь-справочник, в котором были бы наглядно выражены важнейшие парадигматические связи между дескрипторами.

Тезаурус – это словарь-справочник, в котором перечислены все лексические единицы ИМЯ с синонимичными им словами, а также выражены все важнейшие смысловые отношения между лексическими единицами.

Традиционно на тезаурус, как на элемент информационного языка, возлагаются следующие функции:

- быть средством формализации лексики;
- быть средством терминологического контроля, с помощью которого сводится воедино и приводится к общему знаменателю лексическая многозначность, синонимия;
- быть средством избыточного индексирования информационных запросов;
- быть средством выражения парадигматических отношений языка.

Основные этапы разработки тезауруса можно свести к следующим:

1. Выбор источников лексики и отбор терминов.
2. Составление терминологического словаря.
3. Группировка терминов в тематические классы.
4. Формирование классов условной эквивалентности.
5. Установление парадигматических отношений.
6. Определение структуры тезауруса.

При выборе источников лексического материала для тезауруса необходимо учитывать:

- наиболее точное соответствие документов тематической направленности информационной системы;
- технологическую насыщенность документов и важность содержащейся в них информации;
- связь документов со смежными подсистемами.

Для *отбора лексического материала* необходимо использовать экономические документы, отражающие количественные и качественные характеристики экономического объекта. Кроме того, надо пользоваться такими вспомогательными средствами, как толковые и терминологические словари, справочники по исследуемой тематике, общесоюзные классификаторы.

Одним из важных источников для отбора лексического материала является изучение информационных потребностей пользователей, которое можно проводить с применением специально разработанных карт обследования.

Отбор терминов определяется специальными правилами. Перечислим некоторые из них:

- узкие термины следует применять, если в словаре отсутствуют подходящие более общие термины;
- многословный термин вводить в том случае, если обозначаемое им понятие встречается довольно часто;
- использовать вместо словосочетания два или более отдельных термина следует в том случае, если обозначаемое этим словосочетанием понятие является членом того же класса, что и каждый из этих отдельных терминов;
- прилагательное следует употреблять в сочетании с существительным.

В терминологическом словаре каждому отобранному термину дается определение, соответствующее его экономическому смыслу. Терминологический словарь служит в качестве пособия при формировании запросов конечными пользователями.

Все термины классифицируются в зависимости от функционального назначения в тематические классы. Тематический класс представляет собой группу терминов, несущих целевую

направленность. Ниже приведен перечень тематических классов экономической лексики, и каждому из них дано соответствующее толкование.

Наименование тематического класса	Толкование тематического класса
1. Экономические категории, действия, события	Термины, характеризующие экономические сущности и процессы
2. Субъекты действия	Термины, обозначающие название подразделений
3. Объекты действия	Термины, описывающие предметы, являющиеся объектами первого тематического класса
4. Назначение действия	Термины, описывающие целевое назначение или виды целевого использования каких-либо ресурсов
5. Место действия	
6. Источник поступления (финансирования)	Термины, конкретизирующие источник денежных средств или материальных ресурсов
7. Время действия	
8. Функция управления	Термины, характеризующие тип реализации (по плану, фактически, ожидаемое выполнение)
9. Единица измерения	
10. Атрибутивные характеристики действия	Термины, описывающие отличительные свойства элементов первого и третьего классов (способ исполнения, разряд работ и т. п.)
11. Обоснование действия	Термины, описывающие распоряжения и приказы на выполнение какого-либо действия
12. Причина отклонения	Термины, описывающие источники, вызывающие нарушения нормальной работы объекта

Дальнейшая работа по составлению тезауруса заключается в устранении многозначности (омонимия, полисемия) и синонимии тех терминов, которые включены в терминологический словарь.

Омонимия – это совпадение в звучании и написании разных слов, которые не имеют ничего общего ни по происхождению, ни по функционированию. Например: лук (растение) и лук (оружие). Снятие омонимии при обработке тезауруса осуществляется лексикографически, т.е. путем указания в скобках слов, уточняющих смысл омонима.

Полисемия – это перенос названия одного предмета на другие предметы на основании сходства (по форме, цвету, внутренним свойствам, функциям). Например: звезда (геометрическая фигура) и звезда (небесное тело), матрица(математическая) и матрица (техническая).

Омонимия и полисемия устраняются лексикографически при редактировании терминологического словаря.

Синонимия как явление естественного языка заключается в том, что одному "означаемому" (предмету, явлению) соответствует одно и более "означающих" (слов, словосочетаний). Например: алфавит – азбука, студенты – студенчество.

При построении тезауруса устранение синонимии производится путем группировки терминов в классы условной эквивалентности (КУЭ).

В КУЭ объединяются термины, между объемами понятий которых существуют отношения:

- равнозначности, когда объемы понятий полностью совпадают (геомагнетизм – земной магнетизм);
- перекрещивания, когда лишь часть объема одного понятия входит в объем другого понятия и, в свою очередь, лишь часть объема второго понятия входит в объем первого понятия (книга – монография);
- подчинения, когда объем одного понятия составляет часть объема другого понятия (стол – мебель);
- внеположенности, когда объемы понятий полностью исключают друг друга и при этом не исчерпывают области предметов, о которых ведется рассуждение (стол – стул: общий класс – мебель).

Объем понятий – это множество (класс) предметов, каждый из которых обладает всей совокупностью свойств и признаков, отражающих содержание этого понятия.

В качестве основания для группировки в классы условной эквивалентности может быть использовано любое из этих отношений. В результате формирования классов условной эквивалентности термины тезауруса группируются в синонимические ряды.

В каждом синонимическом ряду выделяется доминанта, то есть такой термин, который может заменить любое слово класса. Доминанту принято называть *дескриптором*. Однако фактически дескриптором является не имя КУЭ, а сам этот класс.

Основными критериями при выборе термина являются:

- полнота выражения смыслового назначения данного класса условной эквивалентности;
- значимость термина для поиска информации;
- научно-техническая точность термина и возможность его использования для индексирования;
- распространенность термина;
- краткость термина и понятность его для неспециалиста;
- частота появления термина в различных источниках;
- частота первичного использования термина при индексировании и поиске.

В качестве дополнительного условия при выборе дескриптора из КУЭ экономической лексики следует учитывать необходимость приведения дескрипторов к единой грамматической форме, то есть выбирать в качестве дескриптора термин, имеющий форму существительного.

Отметим, что парадигматические отношения в тезаурусах могут выражаться четырьмя способами:

- лексикографически;
- при помощи таблиц;
- аналитически;
- графически.

Лексикографический способ предполагает наличие специальных помет, которые указывают, в каких парадигматических отношениях находится данный дескриптор с заглавным.

Табличный способ заключается в том, что под заглавным дескриптором записываются со сдвигом на несколько знаков

вправо дескрипторы, находящиеся с заглавным дескриптором в определенном (и только одном) отношении. Такой способ чаще всего применяется в библиотечно-библиографических классификациях.

При *аналитическом способе* парадигматические отношения выражаются при помощи структуры кодов дескрипторов.

Примером применения аналитического способа может служить универсальная десятичная классификация.

Графический способ предполагает применение различных графических схем.

Терминам экономических документов присуще то свойство, что среди всех парадигматических отношений определяющими являются отношения "род-вид" ("вид-род"), обозначающие смысловую соподчиненность терминов по объему понятий, "целое-часть" ("часть-целое"), дающие математическую взаимосвязь экономических категорий.

Поэтому наиболее удобным будет сочетание лексикографического и табличного способов выражения парадигматических отношений. При этом для экспликации парадигматических отношений и отношений синонимии используются условные обозначения, рекомендуемые ГОСТом, а именно:

н – нижестоящий видовой дескриптор по отношению к заглавному дескриптору;

в – вышестоящий родовой дескриптор по отношению к заглавному дескриптору;

ц – дескриптор находится в отношении – целое к заглавному дескриптору;

ч – дескриптор находится в отношении – часть к заглавному дескриптору;

с – ключевое слово находится в отношении синонимии к заглавному дескриптору;

см – отсылка от ключевого слова к дескриптору.

Структура тезауруса влияет на результаты поиска и соответственно на эффективность работы всей системы.

Как правило, тезаурус представляется в виде совокупности расположенных в алфавитном порядке дескрипторных ста-

тей (семантических сегментов). Семантический сегмент представляет собой совокупность заглавного дескриптора и всех дескрипторов, связанных с ним какими-либо парадигматическими отношениями, а также ключевых слов, находящихся с заглавным дескриптором в отношении синонимии. Например:

Выпуск
с выпущено
с выработано
с произведено
с производство
Кольца уплотнительные круглого сечения
в Кольца уплотнительные
н Кольца резиновые уплотнительные круглого сечения, резина группы 0
н Кольца резиновые уплотнительные круглого сечения, резина группы 1

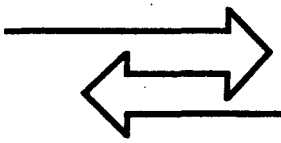
ВОПРОСЫ И ЗАДАНИЯ

1. С какой целью правило, примененное механизмом вывода, исключается из дальнейшего рассмотрения?

2. Как зафиксировать в базе знаний правило, условная часть которого образована путем соединения фактов с помощью логической операции "или"?

3. *Распределите* указанные ниже термины по тематическим классам. Укажите в пределах класса, там где это возможно, условные обозначения парадигматических отношений. Список терминов: предприятие, организация, склад, стеллаж, ячейка, марка, сорт, профиль, размер, тонна, цена, норма запаса, материал, дюраль, поставщик, получатель, приход, расход, остаток, лимит, выдано с учетом возврата, остаток лимита.

4. *Трансформируйте* программу на Прологе, чтобы напечатать список подчиненных для конкретного сотрудника.



ГЛАВА 5

МОДЕЛИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ В ЭКОНОМИЧЕСКИХ ИНФОРМАЦИОННЫХ СИСТЕМАХ

5.1 ПАРАМЕТРИЗАЦИЯ ЭКОНОМИЧЕСКИХ ИНФОРМАЦИОННЫХ СИСТЕМ

Рассмотрение ЭИС как предметной области, естественно, приводит к выделению компонентов ЭИС, их свойств и взаимосвязей между ними, что кратко охарактеризовано в п. 1.2.

Полная реализация этого подхода предполагает:

- определение количественных и качественных параметров объектов, входящих в ЭИС, и процессов их взаимодействия на различных стадиях жизненного цикла системы;
- создание систем хранения и обработки метаинформации, которые получили название баз данных проектировщика ЭИС и словарей-справочников данных;
- использование системы параметров ЭИС для моделирования процессов выбора проектных решений при создании ЭИС, процессов ее эксплуатации и развития.

Параметры ЭИС группируются в следующие классы:

1. Структура базы данных.
2. Структура программного обеспечения ЭИС.
3. Ограничения на доступ пользователей к компонентам базы данных и программного обеспечения.
4. Поток данных и запросов.
5. Вычислительная система ЭИС.

Существующие словари-справочники в основном ориентированы на хранение параметров структуры базы данных, структуры программного обеспечения ЭИС и ограничений на доступ пользователей к компонентам базы данных и программного обеспечения. В базах данных проектировщика дополнительно хранятся некоторые семантические характеристики информационного отображения предметной области в БД. Содержательная обработка и анализ метаинформации требуют создания методов и программных средств, которые первоначально не обеспечиваются программами словаря-справочника.

Параметры структуры базы данных рассматривались в гл. 2.

Параметры программного обеспечения показывают вхождение программ в задачи и подсистемы. Данные о размерах файлов, хранимых в базе данных, и размерах файлов, содержащих программы, представляют отдельную группу параметров.

На рис. 5.1 показаны связи метаобъектов ЭИС, информация о которых обычно хранится в словаре данных. Связи, которые могут быть дополнены объемно-временными параметрами, отмечены на рисунке номерами дуг.

Параметры потоков данных и запросов характеризуют технологические аспекты функционирования базы данных ЭИС, использование данных различными процессами обработки данных, связь процессов обработки данных с требуемым оборудованием, причинно-следственные и временные связи.

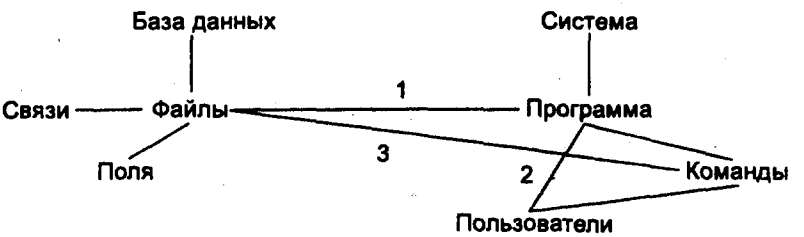


Рис. 5.1. Связи метаобъектов в словаре данных:
1 – среднее количество обращений к файлу из программы;
2 – частота и среднее время выполнения программы пользователей;
3 – частота и среднее время выполнения команд поиска и корректировки файлов

Модель потоков данных содержит объемные характеристики данных, циркулирующих в ЭИС, и динамику изменения этих характеристик во времени. Параметрами потока данных являются количество отношений (файлов), их тип, объем (количество записей и их длина). С помощью типов различается входная, выходная, промежуточная и нормативно-справочная информация. Для каждого файла отмечается количество корректирующих обращений, подразделяемых на включение, исключение и обновление записей.

Модель потока запросов содержит параметры потока пакетных задач и параметры потока интерактивных запросов.

При пакетном режиме обработки данные в файлах накапливаются до тех пор, пока не наступит заданный момент времени или объем данных не превысит некоторый предел. Затем имеющаяся информация обрабатывается несколькими последовательно запускаемыми программами. Среди параметров потока пакетных задач следует назвать: общее количество задач (за некоторый период времени), среднее количество задач в пакете, типы й приоритеты задач, объемы требуемых для них ресурсов.

При *интерактивном режиме* работы происходит обмен сообщениями между пользователем и ЭВМ. Роль активного элемента пользователь и ЭВМ выполняют попеременно. ЭВМ активна от момента завершения ввода информации и команд пользователем до завершения обработки команды (запроса). Пользователь обдумывает результат обработки запроса и вводит данные для следующего запроса. Следует отметить, что последовательность команд, выдаваемых пользователем в интерактивном режиме работы, не является фиксированной заранее, а зависит от результатов ранее выполненных команд.

Параметрами потока интерактивных запросов являются:

- количество пользователей, количество терминалов,
- среднее количество активных терминалов,
- интервалы между сеансами одного пользователя,
- время реакции пользователя,
- типы и приоритеты поступающих запросов,
- объемы ресурсов, необходимых для реализации запросов.

5.2
ФОРМАЛИЗАЦИЯ ПРОЦЕССОВ

Процессы управления экономическими объектами характеризуются ярко выраженной иерархической структурой. По этой причине ЭИС, которая обслуживает процесс принятия управленческих решений на экономическом объекте, должна подразделяться на иерархически соподчиненные компоненты.

Как правило, *в составе ЭИС как системы выделяются подсистемы, подсистемы подразделяются на задачи, в состав задач входят отдельные программные модули.*

Решение задач на ЭВМ в современных условиях чаще всего происходит в диалоговом режиме, а это в большинстве случаев означает, что выбор действий по решению задачи производит пользователь ЭИС из иерархически соподчиненных списков возможных действий, называемых меню действий или функций программы. Поэтому отдельные пункты меню, имеющиеся в программе, естественно, считаются составными частями программы. Кроме того, при реализации пункта меню необходимые последовательности действий могут состоять из программно-самостоятельных компонентов.

Естественно, что начатый таким образом процесс деления системы на составные части по признаку выполняемых в каждом случае действий можно довести до отдельного оператора, применяемого в системе языка программирования. Однако в качестве элементарного действия, выполняемого в ЭИС, целесообразно выбрать пункт меню того или иного программного модуля, реализуемого в системе в диалоговом режиме.

Выделение подсистем в ЭИС тесно связано с административным и функциональным распределением обязанностей среди коллективов специалистов, занятых производственной и управленческой деятельностью на экономическом объекте.

В соответствии с этим принципом подсистемами в ЭИС будут:

- подсистема бухгалтерского учета,

• подсистема статистической отчетности,
• подсистема отдела кадров и т. д.

Выполнение управленческих функций внутри подсистемы связано с решением экономических задач. Выделение экономической задачи связано с получением такой совокупности экономической информации, которая достаточна для реализации некоторой функции управления экономическим объектом или группы взаимосвязанных функций управления.

Задачи, программные модули, пункты меню при выполнении программ являются вариантами реализации вычислительных процессов в ЭИС. Далее определение процесса приводится применительно к задаче.

Процесс представляет собой некоторую последовательность действий, образующих задачу. Процесс определяется соответствующей программой, содержимым рабочей области памяти и дескриптором процесса.

Программа процесса – это упорядоченный набор машинных команд, реализующих действия, которые должны предприниматься процессом. Команды программы должны быть представлены в вычислительной системе как единое целое.

В *рабочую область памяти* входят наборы данных, которые процесс может считывать, записывать и использовать.

Дескриптор процесса определяет состояние ресурсов, предоставленных процессу. Самыми употребительными ресурсами для процесса являются ресурсы внешней памяти и ресурсы других ЭВМ, если вычислительная система ЭИС реализована как сеть ЭВМ.

Дескриптор процесса содержит *индикатор готовности*, который показывает, может ли этот процесс выполняться в данный момент времени или он должен ожидать завершения других процессов.

Процесс характеризуется *входом*, т.е. наборами данных, которые являются исходными для процесса, и *выходом* – наборами данных, образующимися в результате завершения процесса.

Действие: <название действия>
Вход: <список элементов входа>
Выход: <список элементов выхода>
Механизм: <используемые информационные и вычислительные ресурсы>
Например,
Действие: поиск следующей записи
Вход: предыдущая запись, условие поиска
Выход: найденная запись
Механизм: файл

Принципиальным является вопрос о взаимосвязи процессов.

Теоретической основой для решения этого вопроса является теория конечных систем, а точнее, теория машин со входом.

Математическое описание системы П с конечным числом состояний включает:

- множество допустимых входов U;
- множество допустимых выходов Y;
- множество состояний Q;
- функцию перехода $L: Q \times U \rightarrow Q$;
- функцию выхода $V: Q \times U \rightarrow Y$.

Применительно к процессам справедлив ряд упрощений. Так, состояния и выходы системы отождествляются и нет необходимости определять функцию выхода. Входы представляют собой определенные параметры процесса – характеристики входных данных, состояние индикатора готовности, сведения о ресурсах, доступных процессу. Состояние системы описывает характеристики выходных данных процесса. В общем случае сведения о входе и состоянии представляют собой векторные величины, которые далее будут обозначаться малыми латинскими буквами.

Пример

Рассмотрим процесс P с множеством состояний {i,j,k,l} и функцией перехода:

i	j	k	l
k	i	i	j

В этом случае предполагается, что у процесса P одно возможное состояние входа и состояние системы i после выполнения процесса перейдет в состояние системы k и т.д.

Процесс R связан с процессом P, в результате чего состояния, достигнутые после выполнения процесса P, определяют входные параметры для процесса R.

Необходимо существование перехода Z от состояний процесса P ко входам процесса R в виде соответствия, например:

i	j	k	l
y	x	z	y

Поскольку процесс R имеет три возможных состояния входа {x,y,z}, функция перехода для R с состояниями {a,b,c,d} может иметь вид:

	a	b	c	d
x:	d	c	d	b
y:	c	a	b	d
z:	b	c	d	a

Теперь, если процесс P первоначально характеризовался параметром состояния j и процесс R – соответственно параметром c, в результате последовательного выполнения процессов P и R произойдет следующее. Выполнение процесса P создаст состояние i, в соответствии с функцией перехода Z параметром входа для процесса R станет y. Поэтому на основании функции перехода для R состояние c для R сменится на b.

На практике входы и состояния процесса соответствуют логическим величинам (файл данных создан/не создан, ресурс для выполнения процесса свободен/занят, процесс готов/не готов к выполнению). Поэтому вместо алгебраических моделей взаимосвязи процессов, показанных выше, широкое распространение получили графические модели, элементами которых являются обозначения процессов и данных, а взаимосвязи между ними характеризуют причинно-следственные отношения.

Достаточно широкое распространение получили системы автоматизации проектирования и сопровождения ЭИС, основанные на двух теоретико-графовых моделях:

- SADT (Structured Analysis and Design Technique) – структурный системный анализ и технология разработки системы;

- IDEF (Integrated Definition) – интегрированное определение системы.

Применительно к процессам, реализуемым в ЭИС, модель должна иметь:

- описание последовательности процессов,
- указание входных и выходных данных относительно каждого процесса,
- фиксацию условий, при которых выполняется процесс,
- разделение процесса на составляющие его части (которые, в свою очередь, также являются процессами).

Основным элементом моделирования процесса является диаграмма. Диаграммы объединяются в иерархические структуры, причем чем выше уровень диаграммы, тем менее она детализирована. В состав диаграммы входят *блоки*, изображающие допустимые действия в системе, и *дуги*, изображающие взаимосвязь действий.

Таким образом, мы приходим к следующему множеству понятий, необходимых для определения процессов:

- процесс;
- данные;
- использует;
- формирует;
- содержит;
- управляется;
- получены;
- предназначены для.

Пример

Применим этот аппарат описания процессов и данных к следующему примеру. Он представляет собой фрагмент последовательности процессов в ЭИС и графически представлен на рис.5.2.

Описание процесса X выглядит следующим образом:

процесс X;
использует Y1,Y2;
формирует Y3, Y4, Y5;
содержит X1,X2;
процесс X1;

использует Y1;
формирует Y3,Y4,Y12;
содержит X11,X12;
процесс X2;
использует Y2,Y12;
управляется Y4;
формирует Y5;
процесс X11;
использует Y1;
формирует Y3,Y4,Y11;
процесс X12;
использует Y11;
формирует Y12.

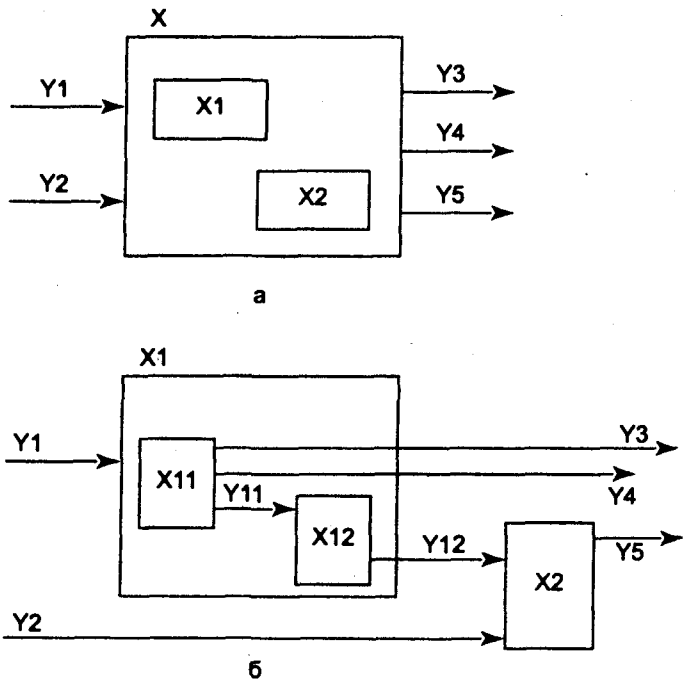


Рис. 5.2. Графическая иллюстрация процесса X:
а – процесс X; б – детализация процесса X

Аналогичные описания могут быть получены для связей данных относительно процессов с использованием термина *получены* для процесса, который сформировал эти данные, и термина *предназначены для*, чтобы назвать процесс, который будет использовать эти данные. В нашем примере эти описания не приводятся, поскольку не указаны предшествующие и последующие процессы для процесса X.

Сети Петри

Для анализа взаимосвязей процессов целесообразно использование сетей Петри.

Сети Петри первоначально были предназначены для описания взаимодействующих компонентов аппаратуры различного назначения, в частности вычислительных систем, однако впоследствии выяснилось, что они позволяют анализировать вычислительные процессы произвольной природы.

Сеть Петри состоит из четырех элементов: множества позиций P, множества переходов T, входной функции I и выходной функции O. Входная функция I отображает переход t в множество позиций I(t), называемых *входными позициями перехода*. Выходная функция O отображает переход t в множество позиций O(t), называемых *выходными позициями перехода*.

Практически более удобно представление сети Петри в виде графа с двумя типами вершин – кружки на графе обозначают позиции, а планки – переходы. Дуги от кружков к планке t обеспечивают задание входной функции, а дуги от планки к кружкам – выходной функции.

Применительно к формализации процессов получаем следующие соответствия:

- планки соответствуют вычислительным процессам,
- кружки соответствуют данным, событиям и условиям.

Если от кружка к планке проведена дуга, то кружок обозначает входное данное, событие или условие для соответствующего процесса. Если от планки к кружку проведена дуга, то кружок обозначает выходное данное, событие или условие.

Маркировка сети Петри представляет собой присвоение фишек позициям сети. Фишки являются метками позиций и на графе обозначаются точками внутри кружков. Количество фишек в каждой позиции может быть произвольным.

Процесс перераспределения фишек в сети называется *выполнением сети Петри*. Фишки находятся в кружках и управляют запуском переходов в сети. Переход запускается удалением фишек из всех его входных позиций и образованием новых фишек, помещаемых во все его выходные позиции.

Введем понятие *вектора маркировки*, в котором число элементов равно числу позиций в сети, а *значением элемента* является количество фишек в соответствующей позиции.

Одной из центральных аналитических задач для процессов, описываемых сетью Петри, является задача определения достижимости маркировки, когда для исходного вектора маркировки требуется установить существование последовательности переходов, после выполнения которой достигается некоторый заданный выходной вектор маркировки.

Применительно к вычислительным процессам в ЭИС с помощью анализа достижимости маркировки устанавливается последовательность действий, позволяющая получить требуемые данные, условия или события.

Один из наиболее элементарных *методов анализа достижимости маркировки* заключается в следующем. Структура сети Петри описывается двумя матрицами D′ и D″, число строк в которых равно числу переходов в сети, а число столбцов равно числу позиций.

Матрица D′ называется *матрицей входов* и содержит 1 на пересечении i-й строки и j-го столбца, если j-я позиция является входной для i-го перехода (в обратном случае элемент равен 0).

Матрица D″ называется *матрицей выходов* и содержит 1 на пересечении i-й строки и j-го столбца, если j-я позиция является выходной для i-го перехода.

Вектор x называется *вектором запуска переходов*. Число элементов в x равно числу переходов, а значение каждого элемента определяет количество запусков данного перехода в процессе выполнения сети Петри. Если исходный вектор мар-

кировки обозначить m0, а результирующую маркировку – через m1, то достижимость маркировки m1 равнозначна существованию вектора x с неотрицательными целыми элементами, который служит решением уравнения

$$m1 = m0 + x(D'' - D').$$

Пример

Рассмотрим фрагмент сети Петри на рис. 5.3. Исходная маркировка m0=(1,1,0,0,0,0,0) соответствует наличию в системе двух корректно вычисленных файлов данных. Определим достижимость маркировки m1=(0,0,1,0,0,1,1) для файлов с результирующей ин-

$$D' = \begin{vmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{vmatrix}$$

$$D'' = \begin{vmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{vmatrix}$$

формацией. Состояния матриц D′ и D″ показаны ниже.

Решением уравнения служит вектор запуска x=(1,0,1,1). Это означает, что требуемые выходные файлы можно получить в результате выполнения процессов с номерами 1, 3, 4.

Следует отметить, что применяемый метод не позволяет определить взаимный порядок выполнения этих процессов.

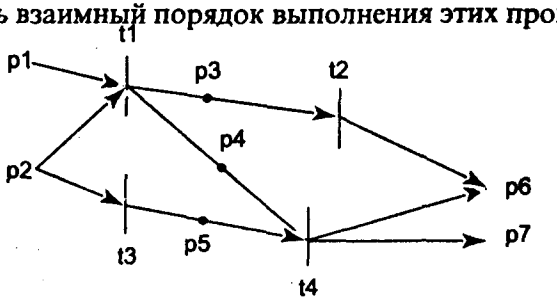


Рис. 5.3. Анализ достижимости для сетей Петри

5.3
МОДЕЛИРОВАНИЕ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

Рассмотрение параметров вычислительной системы (ВС) позволяет анализировать производительность ЭВМ и ее отдельных устройств.

Чтобы количественно определить меру производительности, надо установить элементарную единицу работы для ВС. В режиме пакетной обработки данных единицей работы является задание, которое пользователь представляет операционной системе как единое целое.

При интерактивной обработке единицей работы является взаимодействие, состоящее из двух частей – системной и терминальной. *Системная часть взаимодействия* – это обработка команды, поданной пользователем с терминала. *Терминальная часть* соответствует действиям пользователя начиная от получения результата выполнения предыдущей команды до подачи следующей команды. Время терминальной части взаимодействия называется *временем обдумывания*.

Рабочей нагрузкой ВС называется совокупность поступающих на обработку программ, данных и терминальных команд за некоторый период времени. Обычно предполагается, что рабочая нагрузка нечувствительна к изменениям производительности вычислительной системы, однако ускорение выполнения заданий и терминальных команд иногда вызывает их более частое применение пользователями.

Исходные данные для расчета производительности ВС и ее отдельных устройств получаются экспериментально с помощью аппаратных или программных измерительных систем.

Параметры, которые обычно фиксируются программными измерителями, приводятся в табл. 5.1.

Расчет производительности работы вычислительной системы и ее отдельных устройств будем производить методами операционного анализа.

Таблица 5.1. **Эксплуатационные параметры ВС**

Параметр	Обозначение
Количество выполненных заданий	C
Полезное время	T
Время выполнения заданий	B
Время работы процессора	B(1)
Число обращений к магнитным дискам	d
Количество напечатанных строк	n
Число сеансов работы в диалоге	m
Суммарное время сеансов	V

Зафиксируем некоторый период наблюдений за ВС и обозначим его через T. Время, в течение которого система обрабатывала задания, обозначим через B(0) (B(0) <= T), а количество обработанных заданий – через C(0) . Индекс 0 описывает параметры вычислительной системы как единого целого. В ряде случаев он опускается, чтобы не усложнять формулы.

Определим основные параметры.

1. Коэффициент использования U(0) = B(0) / T.
2. Среднее время выполнения одного задания S(0) = B(0) / C(0).
3. Интенсивность выходного потока заданий X(0) = C(0) / T.

Величина X иначе называется *пропускной способностью* ВС и часто принимается в качестве критерия производительности ВС, который необходимо максимизировать.

Отметим очевидное соотношение X(0) = U(0)/ S(0) и учтем возможность представления S(0) в виде

$$S(0) = Ft,$$

где F – среднее количество команд в задании,
t – время выполнения одной команды.

Величина F зависит как от аппаратуры, так и от программного обеспечения, поэтому настройка ВС часто предполагает и изменение конфигурации ВС, и усовершенствование применяемых программ.

Реальная ВС состоит из нескольких устройств, и задание в процессе выполнения захватывает в определенной последовательности одно из них.

Для отдельных устройств целесообразно рассмотреть следующие величины:

C(i) – количество заданий, покинувших i-устройство за период времени T;

C(i,j) – количество заданий, покинувших i-устройство и поступивших в j-устройство за период времени T;

B(i) – время занятости i-устройства за период времени T.

Условимся о стандартных номерах устройств:

1 – центральный процессор,

2 – магнитный диск-сервер,

3 – принтер-сервер.

Окончание обработки задания системой будем рассматривать как переход к нулевому устройству.

Принятые обозначения удобны для замкнутой системы, в которой каждое выполненное задание немедленно заменяется новым заданием с идентичными статистическими характеристиками.

Для отдельных устройств определяются коэффициент использования U(i), среднее время занятости устройства заданием S(i), интенсивность выходного потока заданий с устройства X(i) по формулам:

$$U(i) = B(i)/T,\ S(i) = B(i)/C(i),\ X(i) = C(i)/T = U(i)/S(i).$$

Введем дополнительные параметры устройств.

4. Вероятность перехода задания от устройства i к устройству j – q(i,j):

$$q(i,j) = C(i,j)/C(i),\ \Sigma q(i,j) = 1.$$

Вероятности вида q(0,j) характеризуют поступление новых заданий. Обычно $q(0,1) = 1$, а $q(0,j) = 0$ для $j > 1$. Практически всегда справедливо соотношение $q(i,i) = 0$.

5. Коэффициент посещения устройства i – V(i)

$$V(i) = X(i)/X(0) = C(i)/C(0).$$

Из определения q(i,j) следует соотношение

$$C(j) = C(i)q(i,j).$$

Делим обе части на T и получаем

$$X(j) = \Sigma X(i)q(i,j).$$

После деления на X(0) имеем с учетом V(0) = 1

$$V(j) = q(0,j) + \Sigma V(i)q(i,j).$$

Данное уравнение имеет особенно простые решения, если лишь K из величин q(i,j) находятся в интервале $0 < q(i,j) < 1$ ($j > 1$). Этому условию соответствуют, например, системы с одним центральным устройством (процессором), когда от остальных устройств возможен переход только к процессору. Второй важный случай – локальные вычислительные сети с одной ЭВМ-сервером и соединением типа "звезда".

6. Среднее время ответа устройства i – R(i)

$$R(i) = W(i)/C(i),$$

где W(i) – сумма времени ожидания и выполнения заданий.

Справедливы соотношения W(i) >= B(i) и R(i) >= S(i).

Средняя длина очереди к устройству составляет n(i) = W(i)/T, поэтому

$$R(i) = W(i)/C(i) = n(i)T/C(i) = n(i)/X(i).$$

Это равенство называется *законом Литтла*. Сумма средних длин очередей к устройствам представляет собой коэффициент мультипрограммирования N

$$N = \Sigma n(i) = \Sigma W(i)/T = W/T,$$

где W – суммарное старт-стопное время выполнения заданий при условии, что период наблюдения T не содержит простоев ВС.

Среднее время ответа вычислительной системы R(0) определяется как

$$R(0) = N/X(0) = \Sigma n(i)/X(0) = \Sigma V(i)R(i).$$

Функциональные связи устройств ВС удобно описывать в виде графа с вершинами, которые обозначаются номерами устройств i = 1...K и дугами (i,j) при условии, что q(i,j) > 0. Мы детально рассмотрим два класса ВС, показанных на рис. 5.4. Они являются системами с центральным устройством, для которых уравнения для коэффициентов посещения преобразуются к виду:

$$X(0) = X(1)q(1,0)$$
$$X(1) = X(0) + X(2) + X(3) +...+ X(K)$$
$$X(i) = X(1)q(1,i)$$
$$V(1) = 1/q(1,0)$$
$$V(i) = q(1,i)/q(1,0)$$

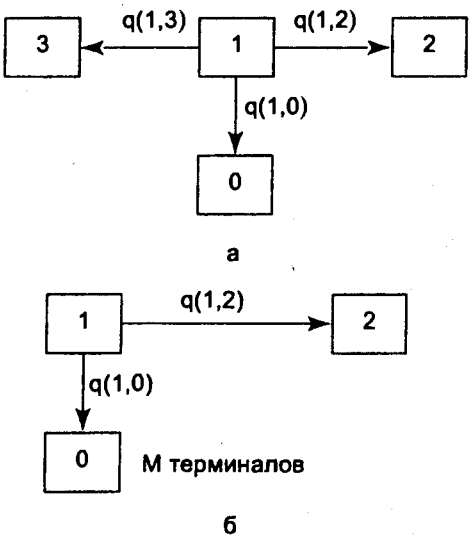


Рис. 5.4. Простейшие функциональные схемы ВС (обратные дуги с q (i, 1) = 1 не показаны):
а – система А. Центральный процессор с двумя каналами ввода-вывода и запуском заданий в пакетном режиме;
б – система В. Интерактивная нагрузка создается М-терминалами

Системой с центральным устройством является как локальная ЭВМ, так и сеть ЭВМ с соединением типа "звезда". В последнем случае "устройствами" считаются отдельные ЭВМ.

Вычислительные сети с соединениями ЭВМ по типу "звезда" относятся к ВС типа В. Компьютер-сервер содержит процессор-сервер (устройство 1), диск-сервер (устройство 2), принтер-сервер (устройство 3) и т.д. Компьютеры-хосты соответствуют отдельным терминалам ВС типа В. Когда компьютер-хост работает автономно от сервера, мы имеем терминальную часть взаимодействия (обдумывание). В случае приема/передачи данных и захвата при этом ресурсов сервера мы имеем системную часть взаимодействия.

Под настройкой действующей вычислительной системы понимают процессы изменения ее конфигурации и регулирования ее параметров, направленные на увеличение ее производительности. Методы настройки ВС, рассматриваемые здесь, можно применять как для автономно работающей ЭВМ, так и для ЭВМ в составе вычислительной сети. В первом случае необходима разработка специальных программ, фиксирующих время выполнения заданий, а во втором – необходимые измерения входят в состав статистики транзакций, фиксируемой сетевым программным обеспечением.

Выполняемые задания создают различную нагрузку для отдельных устройств ВС. С увеличением числа одновременно выполняемых заданий (т.е. коэффициента мультипрограммирования N) у всех устройств ЭВМ будут расти значения коэффициентов использования U(i). Устройство с номером d, которое первым достигнет значения U(d), практически равного 1, станет создавать основные задержки для выполняемых заданий; оно называется *насыщенным устройством*. Для увеличения производительности ВС можно заменить насыщенное устройство на более быстродействующее либо снизить нагрузку на него путем изменения структуры БД и модификации программ пользователей.

Для характеристики насыщенного устройства воспользуемся соотношением

$$U(i)/U(j) = (V(i)S(i))/(V(j)S(j)),$$

которое следует из определения коэффициентов посещения V(i).

У насыщенного устройства d U(d) = max{U(i)} и, следовательно,

$$V(d)S(d) = \max\{V(i)S(i)\}.$$

При возрастании коэффициента мультипрограммирования N имеем U(d) → 1 и получаем X(d) = 1/S(d). Поскольку V(0) = 1, справедливо равенство X(0)/X(d) = 1/V(d), и, следовательно, интенсивность выходного потока заданий не может превышать величины

$$X' = 1/(V(d)S(d)).$$

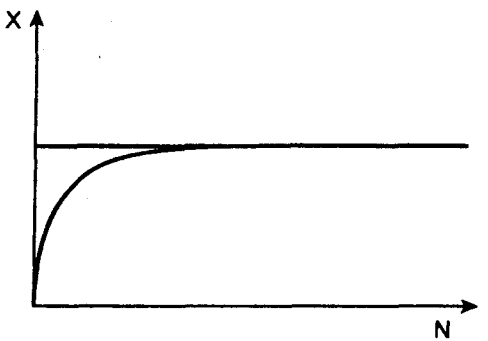


Рис. 5.5. Зависимость производительности вычислительной системы от коэффициента мультипрограммирования N

Если в выражении для среднего времени ответа учесть неравенства S(i) <= R(i), то получается минимальное среднее время ответа R '' = V(i)S(i), характерное для коэффициента мультипрограммирования N = 1.

Интенсивность выходного потока заданий X(0) в зависимости от N ограничена асимптотами, показанными на рис. 5.5.

Для увеличения интенсивности выходного потока заданий существуют следующие возможности настройки ВС.

1. Заменить насыщенное устройство на более быстродействующее. Это приведет к уменьшению S(d) и росту X'.

2. Заменить хранение производной информации на ее динамическое вычисление или, наоборот, в зависимости от того, какое устройство является насыщенным – магнитный диск или процессор. В этом случае уменьшится V(d).

3. Модифицировать прикладную программу, чтобы сократить число команд, выдаваемых на насыщенное устройство.

4. Если насыщенным является внешнее запоминающее устройство, то в качестве временной меры можно осуществить реорганизацию файлов на этом устройстве.

Насыщенным устройством на практике обычно оказывается центральный процессор или система магнитных дисков.

Основное соотношение, описывающее вычислительную сеть типа В, получается на основе закона Литтла. Среднее время одного терминального взаимодействия складывается из системного времени ответа R и времени Z обдумывания ответа пользователем. По закону Литтла выражение (R + Z)X(0) *определяет среднее число терминальных взаимодействий, т.е.* число включенных терминалов M. Отсюда

$$R = (M/X(0)) - Z.$$

Когда число терминалов M = 1, среднее время ответа R = R(0). С увеличением M выходной поток заданий X(0) будет возрастать, но не более величины X(0) = 1/V(d)S(d). Таким образом,

$$R >= (MV(d)S(d)) - Z.$$

Для больших M среднее время ответа R (рис. 5.6) имеет асимптоту, определяемую выражением MV(d)S(d) – Z. Ориентировочно величина

$$M(d) = (MV(d)S(d)) - Z$$

считается максимально допустимым количеством терминалов. При M >= M(d) отмечается резкий рост среднего времени ответа.

С течением времени возрастает опыт пользователей, работающих с ВС. С точки зрения производительности сети типа В это означает в первую очередь уменьшение времени обдумывания Z и увеличение среднего времени ответа R. При неизменном числе

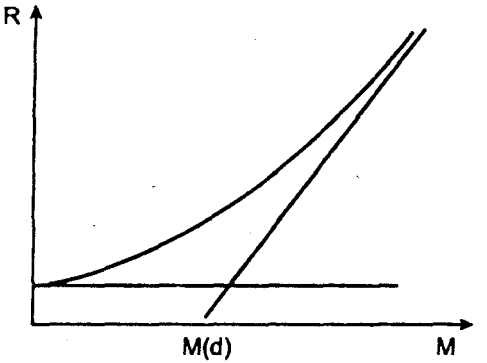


Рис. 5.6. Зависимость среднего времени ответа интерактивной вычислительной системы от числа терминалов M

терминалов возрастает коэффициент использования U(d) насыщенного устройства, что приближает момент проведения настройки ВС или других мероприятий по модернизации ВС. Одним из допустимых решений является сокращение числа активных терминалов путем перехода к иерархической сетевой архитектуре, когда "избыточные" терминалы получают доступ к собственному серверу, соединенному с основным сервером сети.

Выбор терминалов для нижних уровней иерархии сети из числа уже действующих терминалов производится методами кластерного анализа на основе измерения близости информационных потребностей у пользователей действующих терминалов.

ВОПРОСЫ И ЗАДАНИЯ

1. *Изобразите* сеть Петри для следующих матриц входов и выходов.

$$D'=\begin{vmatrix}1&1&0&0&0&0&0\\0&1&1&0&0&0&0\\0&0&1&0&0&0&0\\0&0&0&1&1&0&0\end{vmatrix} \qquad D''=\begin{vmatrix}0&0&1&1&1&0&0\\0&0&0&0&0&1&0\\0&0&0&0&0&0&1\\0&0&0&0&0&1&1\end{vmatrix}$$

2. По результатам измерений среднее время выполнения одного запроса на ЭВМ составляет 2 мин. Интенсивность потока запросов – 20 запросов в час. Чему равен коэффициент использования ЭВМ?

ЛИТЕРАТУРА

1. *Атре Ш.* Структурный подход к организации баз данных: Пер. с англ. / Под ред. В.И. Будзко. – М.: Финансы и статистика, 1983. – 317 с.

2. *Бусленко Н.П.* Моделирование сложных систем. – 3-е изд. – М.: Наука, 1978. – 399 с.

3. *Жеребин В.М., Мальцев В.Н., Совалов М.С.* Экономические информационные системы. – М.: Наука, 1978. – 200 с.

4. *Королев М.А., Мишенин А.И., Хотяшов Э.Н.* Теория экономических информационных систем. – М.: Финансы и статистика, 1984. – 223 с.

5. *Кузин Л.Т.* Основы кибернетики. – М.: Энергия, 1979. – Т. 2. – 584 с.

6. *Леон-Хонг Б., Плагман Б.* Системы словарей-справочников данных: Пер. с англ. – М.: Финансы и статистика, 1986. – 311 с.

7. *Литвин В.Г.* и др. Анализ производительности мультипрограммных ЭВМ. – М.: Финансы и статистика, 1984. – 159 с.

8. *Мартин Дж.* Организация баз данных в вычислительных системах. – М.: Мир, 1980. – 662 с.

9. *Мейер Д.* Теория реляционных баз даиных: Пер. с англ. – М.: Мир, 1987. – 608 с.

10. *Тиори Т., Фрай Дж.* Проектирование структур баз данных: Пер. с англ. / Под ред. В.И. Скворцова. – М.: Мир, 1985. – Т. 1. – 287 с.; Т. 2. – 320 с.

11. *Уэно Х.* и др. Представление и использование знаний: Пер. с яп. – М.: Мир, 1989. – 220 с.

12. *Цикритзис Д., Лоховски Ф.* Модели данных: Пер. с англ. – М.: Финансы и статистика, 1985. – 344 с.

ОГЛАВЛЕНИЕ

Учебное издание

Мишенин Александр Иванович

ТЕОРИЯ ЭКОНОМИЧЕСКИХ ИНФОРМАЦИОННЫХ СИСТЕМ

Ведущий редактор *Л.Д. Григорьева*
Редактор *Л.В. Сергеева*
Художественный редактор *Ю.И. Артюхов*
Технический редактор *И.В. Белюсенко*
Корректоры *Т.М. Колпакова, Г.В. Хлопцева*
Обложка художника *Н.М. Биксентеева*
Компьютерная верстка *Е.Ф. Тимохиной*

ИБ № 3884

Подписано в печать 23.07.2002. Формат 60×88/16
Гарнитура «Таймс». Печать офсетная
Усл. п. л. 14,7 Уч.-изд. л. 12,2
Тираж 3000 экз. Заказ 2393. «С» 175

Издательство «Финансы и статистика»
101000, Москва, ул. Покровка, 7
Телефон (095) 925-35-02. Факс (095) 925-09-57
E-mail: mail@finstat.ru http://www.finstat.ru

ГУП «Великолукская городская типография»
Комитета по средствам массовой информации Псковской области
182100, Великие Луки, ул. Полиграфистов, 78/12
Тел./факс: (811-53) 3-62-95
E-mail: VTL@MART.RU