Н. А. Гайдамакин

# АВТОМАТИЗИРОВАННЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ, БАЗЫ И БАНКИ ДАННЫХ

## Вводный курс

*Допущено УМО вузов по образованию в области информационной безопасности в качестве учебного пособия для студентов высших учебных заведений, обучающихся по специальностям «Компьютерная безопасность» и «Комплексное обеспечение информационной безопасности автоматизированных систем»*

Москва
«Гелиос АРВ»
2002

УДК 681.3.06
ББК 32.973.2
Г38

**Гайдамакин Н. А.**
Г38 Автоматизированные информационные системы, базы и банки данных. Вводный курс: Учебное пособие. — М.: Гелиос АРВ, 2002. — 368 с., ил.

ISBN 5-85438-035-8

Учебное пособие содержит сведения по автоматизированным информационным системам и лежащим в основе их создания и функционирования системам управления базами данных.

Рассматриваются структура и классификация автоматизированных информационных систем и СУБД, модели организации данных в фактографических СУБД, основы концептуального проектирования банков данных фактографических систем и схемно-структурного проектирования реляционных баз данных. Представлен материал по основам и технологиям построения распределенных и многопользовательских информационных систем.

Для студентов вузов и слушателей институтов повышения квалификации, изучающих курсы по теории информационных систем, баз и банков данных, информационного поиска, а также для специалистов, занимающихся информационным обеспечением деятельности предприятий и организаций.

УДК 681.3.06

ББК 32.973.2

**ISBN 5-85438-035-8**



## ПРЕДИСЛОВИЕ

Характерной чертой нашего времени являются интенсивно развивающиеся процессы информатизации практически во всех сферах человеческой деятельности. Они привели к формированию новой информационной инфраструктуры, которая связана с новым типом общественных отношений (информационные отношения), с новой реальностью (виртуальной реальностью), с новыми информационными технологиями различных видов деятельности. Сердцевиной современных информационных технологий являются автоматизированные информационные системы (АИС), создание, функционирование и использование которых привело к возникновению специфических понятий, категорий, приемов и навыков.

В настоящее время изучение дисциплин, связанных с автоматизированными информационными системами (АИС), является обязательным элементом подготовки специалистов в самых разнообразных областях деятельности.

Вместе с тем анализ отечественной и переводной учебной литературы по тематике АИС, баз и банков данных показывает ее расслоение по нескольким, независимо развивающимся, направлениям.

Первое направление, которое условно можно назвать «программистским», связано с системами управления базами данных (СУБД) фактографических АИС и представлено рядом классических учебных изданий, по которым на протяжении последних двадцати лет осуществляется подготовка специалистов по программированию, разработке и эксплуатации средств вычислительной техники. К числу подобных работ относится энциклопедическое издание одного из основоположников теории баз данных, известного американского специалиста, К. Дейта «Введение в системы баз данных», выдержавшее с момента своего первого выхода более шести изданий. В этом же ряду находится книга еще одного известного американского специалиста А. Саймона «Стратегические технологии баз данных».

В отечественной литературе данное направление представлено рядом фундаментальных изданий, среди которых можно упомянуть работы: Л. А. Овчарова, С. Н. Селеткова «Автоматизированные банки данных», В. В. Бойко, В. М. Савинкова «Проектирование баз данных информационных систем», а также более современные учебные курсы по системам управления базами данных известных отечественных специалистов С. Д. Кузнецова «Введение в СУБД» и Г. М. Ладыженского «Системы управления базами данных — коротко о главном».

Особенностью перечисленных выше изданий является их математико-программистская направленность, что формирует определенные требования к исходной подготовке обучаемых.

Второе направление — просветительское — обусловлено процессами так называемого реинжиниринга (переустройство бизнеспроцессов на предприятиях и в организациях на основе новых информационных технологий), активно развившегося в 90-е годы и ориентированного на различные категории управленческих работников (менеджеров). Особенностью работ данного направления является «популяризаторское» изложение материала в отношении основ, устройства и функционирования баз данных и СУБД. Наиболее интересной в этом плане можно назвать книгу Д. Васкевича «Стратегии Клиент/Сервер. Руководство по выживанию для специалистов по реорганизации бизнеса».

Третье направление — библиотечное — относится к области документальных информационных систем, являющихся современным инструментом документального информационного поиска, традиционно рассматриваемого в контексте библиотечно-информационной сферы. Здесь можно упомянуть ряд известных изданий, в частности работы: Дж. Солтона «Динамические библиотечно-информационные системы», Ф. У. Ланкастера «Информационно-поисковые системы», А. И. Черного «Введение в теорию информационного поиска», А. В. Соколова «Информационно-поисковые системы». Вероятно, ввиду гораздо более ранней, чем у компьютерных систем, предыстории библиотечно-информационной сферы, это направление развивалось практически до 90-х годов отдельно от фактографических АИС. В результате, документальные АИС, несмотря на общую с фактографическими системами природу практически никогда не рассматривались с ними в одной работе.

Последние тенденции в развитии АИС, баз данных и СУБД проявляют признаки конвергенции фактографического и документального направления, что предопределяет изучение предметной сферы АИС в рамках единой идеологии и единых подходов.

Еще одним мотивом для создания данного учебного пособия послужили наблюдения по контингенту специалистов, участвующих в создании и эксплуатации АИС.

Массовая компьютеризация и персонализация компьютерной техники во второй половине 80-х годов, внедрение АИС в деятельность не только крупных, средних, но и мелких предприятий потребовало большого количества специалистов, способных такие системы разрабатывать. В результате, в сферу создания и эксплуатации (администрирования) АИС пришел большой отряд специалистов из смежных областей, прежде всего разработчиков программного обеспечения так называемых прикладных программистов. Обладая навыками программирования и знанием языков программирования высокого уровня, во многих случаях специалисты данной категории слабо или вовсе не представляют системологических основ АИС, а также многих других аспектов предназначения и функций баз данных. Поэтому создание АИС зачастую рассматривается в узком смысле создания базы данных и разработки примитивного интерфейса для работы с ней. Подобный технократический подход не может обеспечить должный уровень, качество и эффективность разрабатываемых АИС.

Целью создания представляемого учебного пособия является системное изложение всех аспектов тематики АИС, которое соединяет, хотя бы в рамках вводного и краткого курса, основы учебного материала первого, второго и третьего направлений учебных изданий, касающихся АИС.

Пособие состоит из семи глав.

Первая глава посвящена основам информационного обеспечения различных процессов и систем, понятиям, функциям, структуре и классификации информационных систем, системам представления данных в фактографических АИС.

Во второй главе рассматриваются функции и внутреннее устройство систем управления базами данных фактографических АИС, классические модели организации данных (иерархическая, сетевая и реляционная). Изложение реляционной модели в части манипуляционной составляющей сопровождается примерами операций над данными. Большая часть второй главы посвящена внутренней схеме баз данных, структурам физической организации данных с рассмотрением также и таких основополагающих аспектов организации и функционирования баз данных, как индексирование и хеширование записей.

Третья глава посвящена основам создания АИС. Приводятся материалы нормативных документов (ГОСТов), регламентирующих организацию и содержание работ по созданию АИС, рассматриваются понятие и сущность технического задания. Излагаются также основы проектирования центрального компонента АИС — банка данных, с описанием этапов концептуального и схемно-структурного проектирования. Описывается сущность процессов нормализации таблиц из классической теории реляционной модели данных. Как и во всем пособии, все операции над данными иллюстрируются

примерами. В силу вводного характера пособия важная часть теории и практики проектирования баз данных, связанная с CASE-технологиями, затрагивается только в постановочном плане.

Четвертая глава, занимающая центральное место в работе, рассматривает комплекс вопросов, связанных с вводом, обработкой и выводом данных в фактографических АИС. Структура и направленность изложения материала обусловлены логикой и характером информационных задач, решаемых или обеспечиваемых процессами и процедурами обработки данных в отличие от программистского подхода в учебных работах первого направления. Подробно рассмотрены как простейшие вопросы, связанные с просмотром, поиском и фильтрацией табличных данных, так и вопросы классификации, содержания, в том числе программное выражение на языке SQL и оптимизация запросов по обработке данных в реляционных СУБД. Вводятся также другие смежные понятия и процедуры, связанные с событийной техникой обработки данных, формами и отчетами по результатам обработки и вывода данных. Приведены редко рассматриваемые в учебной литературе особенности обработки, и в частности навигации, по связанным записям в АИС с сетевой моделью организации данных.

Пятая глава посвящена теоретическим основам и технологиям построения распределенных АИС. Представлены технологии и модели «Клиент-серверных» систем, в том числе с рассмотрением основ реализации мониторов транзакций. Глава содержит также излагаемые практически только в руководствах по СУБД сведения по технологиям объектного связывания и репликации данных.

Одной из особенностей пособия является изложение в шестой главе в идеологии единого представления с фактографическими системами основ построения и функционирования документальных информационных систем (ИПС). Рассматриваются теоретические основы документального информационного поиска, классификация и особенности разновидностей информационно-поисковых документальных систем. Представлен материал по широко применяемым в настоящее время информационно-поисковым каталогам и тезаурусам, полнотекстовым и гипертекстовым ИПС, моделям организации и особенностям обработки данных, лежащим в основе полнотекстовых и гипертекстовых ИПС.

Заключительная седьмая глава охватывает вопросы администрирования и защиты данных в АИС. Рассматриваются основные функции и задачи, решаемые администраторами баз данных, политики и модели безопасности (разграничения доступа) в СУБД, технологические аспекты реализации моделей безопасности, в том числе технологии «представлений» и другие конструкции языка SQL, обеспечивающие построение и управление системой разграничения доступа в АИС. Приведены также необходимые в минимальном объеме сведения по классификации уровней защищенности информации в автоматизированных системах по Руководящим документам Гостехкомиссии России.

Автор выражает благодарность тем людям, без помощи и поддержки которых создание пособия со столь разноплановым материалом было бы вряд ли возможным. Прежде всего хотелось бы выразить признательность за советы и критические замечания известному специалисту в области документального информационного поиска Д. Я. Шараеву, а также коллегам по работе О. Н. Соболеву, С. А. Необутову, С. Н. Смирнову, А. П. Коваленко и отдельно О. В. Безусовой, принявшей на себя тяготы редактирования первоначального варианта рукописи. Автор также признателен заведующему кафедрой вычислительной техники УГТУ-УПИ С. Л. Гольдштейну и его сотрудникам, в частности Т. Я. Ткаченко, а также заведующему сектором информационных систем ИММ УрО РАН И. А. Хохлову за рецензирование первоначального варианта пособия.

# 1. ОСНОВЫ ИНФОРМАЦИОННОГО ОБЕСПЕЧЕНИЯ ПРОЦЕССОВ И СИСТЕМ

## 1.1. Понятие и содержание информационного обеспечения

Информационное обеспечение является составной частью более широкого понятия информационных процессов. В нормативно-правовой трактовке ***информационные процессы*** определяются как *«процессы создания, сбора, обработки, накопления, хранения, поиска, распространения и потребления информации** и охватывают тем самым все сферы человеческой деятельности.

* Закон РФ «Об участии в международном информационном обмене» от 04. 07. 1996 г.. № 85-03.

Информационное обеспечение чаще всего соотносится с организационно-управленческой и производственно-технологической сферой. Поэтому под ***информационным обеспечением*** будем

понимать *совокупность процессов сбора, обработки, хранения, анализа и выдачи информации, необходимой для обеспечения управленческой деятельности и технологических процессов.*

Основополагающим в определении информационного обеспечения является понятие информации.

Термин ***информация*** происходит от латинского *informatio —разъяснение, изложение.* До середины нашего столетия информация трактовалась как *сведения, передаваемые людьми устным, письменным или другим (знаками, техническими средствами) способом.* После 50-х годов на фоне бурного развития средств связи и телекоммуникаций, возникновения и внедрения в различные сферы жизни электронно-вычислительной техники появились новые, расширенные трактовки понятия информация. Информацию в *вероятностно-статистическом* (или *энтропийном)* подходе стали трактовать *как уменьшение степени неопределенности знания о каком-либо объекте, системе, процессе или явлении, или изменение неопределенности состояния самого объекта, системы, явления, процесса.* Такую трактовку по имени ее автора, американского математика К. Э. Шеннона еще называют информацией *но Шеннону.*

Известна также и широко используется философская, или точнее говоря, *общенаучная* трактовка понятия информации *как изменение объема и структуры знания воспринимающей системы.* При этом под воспринимающей системой понимается не только собственно сам человек или его производные (коллектив, общество), но и, вообще говоря, любая система, например биологическая клетка, воспринимающая при рождении генетическую информацию.

Существует еще и *нормативно-правовая* трактовка понятия информации, которая используется в законодательных актах, регламентирующих информационные процессы и технологии. Так, в частности, в законе РФ «Об информации, информатизации и защите информации» (от 20.02.95 № 24-ФЗ) дается следующее определение термина «информация» — *сведения о лицах, предметах, фактах, событиях и процессах независимо от способа их представления.* Добавим в связи с этим еще один важный нормативно-правовой аспект. Статья 128 Гражданского кодекса РФ информацию, наряду с вещами (включая деньги, ценные бумаги и иное имущество, в том числе имущественные права), работами и услугами, результатами интеллектуальной деятельности, нематериальными благами, определяет видом объектов гражданских прав, распространяя на нес тем самым весь институт гражданского права, включая права собственности и авторское право.

Как представляется, в контексте рассмотрения содержания информационно-аналитической сферы наиболее подходящим является объединение общенаучной и нормативно-правовой трактовки понятия информации. Поэтому в дальнейшем информацию будем понимать *как изменение объема и структуры знания о некоторой предметной области (лица, предметы, факты, события, явления, процессы) воспринимающей системой (человек, организационная структура, автоматизированная информационная система) независимо от формы и способа представления знания.*

При рассмотрении понятия информационного обеспечения в контексте обработки информации важное значение имеет понятие данных. От информации *данные* отличаются конкретной формой представления и являются некоторым ее подмножеством, определяемым целями и задачами сбора и обработки информации. К примеру, данные по сотрудникам какой-либо организации в виде формализованных учетных карточек кадрового подразделения содержат лишь некоторый перечень необходимых сведений (ФИО, год рождения, образование, семейное положение, должность и т. д.) в отличие от огромного количества сведений, характеризующих каждого конкретного человека. Поэтому определим ***данные*** как *информацию, отражающую определенное состояние некоторой предметной области в конкретной форме представления и содержащую лишь наиболее существенные с точки зрения целей и задач сбора и обработки информации элементы образа отражаемого фрагмента действительности.*

Таким образом, *информация на стадии данных* характеризуется определенной *формой представления* и дополнительной характеристикой, выражаемой термином *структура.*

Структура данных связана с понятием ***представления информации*** и определяется (функциональной, логической, технологической и т. п. структурой той предметной области, информацию о которой содержат данные. Вместе с тем данные могут быть представлены и в неструктурированной (форме, что предопределяет технологические особенности их накопления и обработки. Таким образом, можно выделить *неструктурированную* и *структурированную форму представления данных.*

В качестве примера ***неструктурированной*** формы можно привести:

• связный текст (т. е. документ на естественном языке — на литературном, официально-деловом и т. д.);

• графические данные в виде фотографий, картинок и прочих неструктурированных изображений.

Примерами ***структурированной*** формы данных являются:

• анкеты;

• таблицы;

• графические данные в виде чертежей, схем, диаграмм.

Способы сбора, анализа и обработки структурированных и неструктурированных данных существенно различаются. Наиболее развитыми в настоящее время, с точки зрения задач обработки и анализа информации, являются программные средства обработки структурированных данных, т. к. структуризацию можно считать первичной и наиболее трудно формализуемой и алгоритмизируемой обработкой.

В плане оперирования с информацией в процессах ее создания (порождения), сбора, выдачи и потребления важное значение имеет понятие ***документированной информации*** или просто ***документа.*** Можно сказать, что в большом количестве случаев информация предстает и фигурирует в образе документа, исключая ту часть информационных процессов, которые оперируют исключительно с данными, как, например, в автоматизированных системах управления технологическими процессами — АСУТП, где информация порождается в виде показаний датчиков (входные данные), обрабатывается, выдается и потребляется в виде управляющих сигналов (выходные данные) на технологическое оборудование.

Как и в случае с понятием самой информации, существует несколько трактовок термина документ — *историческая, организационно-управленческая* и *нормативно-правовая трактовка.*

*Исторически* документ понимался (и в определенных случаях понимается сейчас) как *объект, средство, способ для удостоверения личности, прав собственности* и т. д.

В *организационно-управленческом* смысле документ понимается как служебный или организационно-распорядительный документ, т. е. как *форма и способ выражения организационно-управленческих решений и воздействий.*

В *нормативно-правовом* аспекте документ определяется как *зафиксированная на материальном носителе информация с реквизитами, позволяющими ее идентифицировать.**

* Закон РФ «Об информации, информатизации и защите информации» от 20.02.1995г., №24-ФЗ.

Для традиционного «бумажного» документа совокупность реквизитов, идентифицирующих конкретный документ, определяется соответствующими ГОСТами* и руководящими документами по делопроизводству или отраслям технологической документации.** Не вдаваясь в детали, отметим, что важнейшим реквизитом, идентифицирующим традиционные документы, является подпись должностного лица. Подобный подход для компьютерной информации в настоящее время развит в виде техники «электронных цифровых подписей», основанных на криптографических методах, также закреплен соответствующими ГОСТами*** и применяется в телекоммуникационных системах передачи данных. Вместе с тем такие особенности компьютерной формы информации, как возможность ее эталонного копирования (т. с. практически мгновенного и в любых количествах порождения полностью идентичных копий, экземпляров), делают процесс идентификации документов в компьютерной форме и в более широком смысле аспект юридического статуса документов в вычислительной среде сложной и до конца еще нерешенной проблемой.

* ГОСТ Р 6.30-97. Унифицированная система организационно-распорядительной документации. Требования к оформлению документов—М.: Госстандарт России, 1997.

** Например, ЕСКД— Единая система конструкторской документации.

*** ГОСТ Р 34.10-94. Информационная технология. Криптографическая защита информации. Процедуры выработки и проверки электронной цифровой подписи на базе асимметричного криптографического алгоритма—М.: Госстандарт России, 1994.

Под *документированием* информации в широком смысле слова можно понимать выделение единичной смысловой части информации (данных) по некоторой предметной области в общей ее массе, обособление этой части с приданием ему самостоятельной роли (имя, статус, реквизиты и т. п.). *Процесс документирования* превращает информацию *в* ***информационные ресурсы.*** Нормативно-правовая трактовка информационных ресурсов определяет их как *«отдельные документы и отдельные массивы документов, документы и массивы документов в информационных системах (библиотеках, архивах, фондах, банках данных, других видах информационных систем)».**

* Закон РФ «Об участии в международном информационном обмене» от 04.07.1996 г., №85-ФЗ.

Таким образом, документирование информации подводит к одному из самых (фундаментальных понятий в сфере информационного обеспечения — информационным системам. Так же как и для понятий информации и документа, понятие информационной системы многогранно и имеет несколько определений и подходов. В нормативно-правовом смысле ***информационная система*** определяется как *«организационно упорядоченная совокупность документов (массивов документов) и информационных технологий, в том числе и с использованием средств вычислительной техники и связи, реализующих информационные процессы».**

* Закон РФ «Об информации, информатизации и защите информации» от 20.02.1995 г., №24-ФЗ.

В технологическом плане аспект использования средств вычислительной техники (СВТ) в информационных системах и обеспечение на этой основе автоматизации решения каких-либо задач проявляется в близком термине ***автоматизированная система*** *— «система, состоящая из персонала и комплекса средств автоматизации его деятельности, реализующая информационную технологию выполнения установленных функций».**

*' ГОСТ 34.003-90. Информационная технология. Комплекс стандартов на автоматизированные системы. Термины и определения —М.: Изд-во стандартов, 1991.

Опыт, практика создания и использования автоматизированных информационных систем в различных сферах деятельности позволяет дать более широкое и универсальное определение, которое полнее отражает все аспекты их сущности.

Под информационной системой в дальнейшем понимается *организованная совокупность программно-технических и других вспомогательных средств, технологических процессов и функционально-определенных групп работников, обеспечивающих сбор, представление и накопление информационных ресурсов в определенной предметной области, поиск и выдачу сведений, необходимых для удовлетворения информационных потребностей установленного контингента пользователей — абонентов системы.*

Исторически первыми видами информационных систем являются архивы и библиотеки. Им присущи все атрибуты информационной системы. Они обеспечивают в какой-либо предметной области сбор данных, их представление и хранение в определенной форме (книго-, архивохранилища, каталоги и т. д.), в них определяется порядок использования информационных (фондов (т. е. определены абоненты, режимы и способы выдачи информации—абонементы, читальные залы и т. п.).

***Информационные системы,*** *в которых представление, хранение и обработка информации осуществляются с помощью вычислительной техники,* называются ***автоматизированными,*** или сокращенно ***АИС.*** Автоматизированные информационные системы в настоящее время являются неотъемлемой частью современного инструментария информационного обеспечения различных видов деятельности и наиболее бурно развивающейся отраслью индустрии информационных технологий.

Таким образом, информационные системы являются основным средством, *инструментарием* решения задач информационного обеспечения, а соотношение понятий, связанных с информационным обеспечением, можно отобразить в виде схемы, приведенной на рис. 1.1.

*Технологическое и организационно-штатное воплощение информационного обеспечения* в большинстве случаев осуществляется в трех формах:

• служба документационного обеспечения управления (СлДОУ);
• информационная служба;
• экспертно-аналитическая служба.

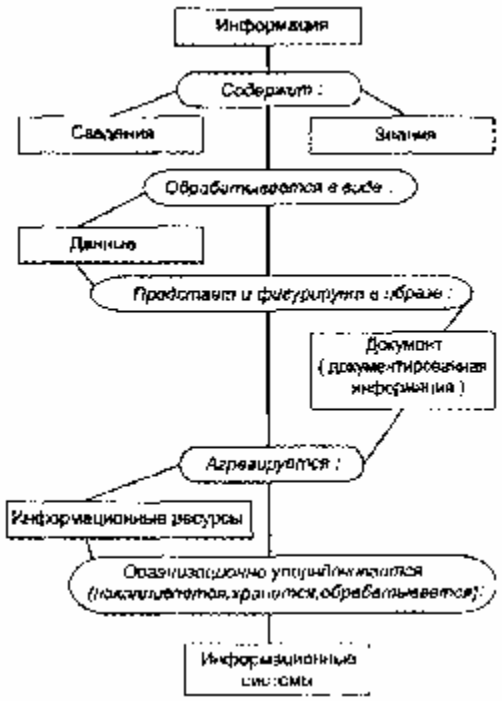



*Рис. 1.1. Схема понятии информационного обеспечения*

Традиционной организационно-штатной и технологической структурой является ***СлДОУ,*** которая в обобщенном виде реализует следующие ***функции*:**

• организация и обеспечение делопроизводства и документооборота;

• документационное обеспечение подготовки и осуществления управленческих решений через организацию и контроль разработки и согласования организационно-распорядительных и информационно-справочных документов;

• внутренний мониторинг (контроль реализации управленческих решений, оценка их результатов через контроль и отслеживание прохождения документов).

***Информационная служба*** (отдел автоматизации, вычислительный центр и т. д.) в общем виде обеспечивает решение следующих задач:

• сбор недокументированной и документированной информации из внешних источников, необходимой для поддержки управленческих и технологических процессов;

• внешний мониторинг (выявление и анализ управленческих проблем, возникающих по внешним причинам);

• информационное оповещение и удовлетворение информационных потребностей управленческих и технологических структур;

• внутренний мониторинг (статистика, отчетность, оценка результатов деятельности).

***Экспортно-аналитическая группа*** (группы советников, консультантов, так называемых аналитиков) привлекается для:

• анализа различных управленческих, производственных ситуаций;

• выработки альтернатив управленческих решений;

• прогнозирования последствий управленческих решений.

*Структура информационного обеспечения* определяется структурой (циклом) функционирования информационных систем. В общем плане можно выделить следующие элементы ***цикла функционирования информационных систем.***

• сбор информации;

• комплектование информацией;

• поиск и выдача сведений для абонентов системы;

• поддержание целостности, актуальности и сохранности информации.

***Сбор информации*** представляет собой *специальным образом организованный порядок и процесс получения и отбора информации, имеющей отношение к предметной области сведений информационной системы,* и включает:

• получение информации;

• оценку относимости информации;

• организационные схемы, порядок отбора и фиксации информации.

*Получение информации* осуществляется через организацию и использование системы источников и каналов получения информации.

*Оценка относимости* информации к предметной области сведений информационной системы в некоторых случаях осуществляется автоматически (информация с датчиков в автоматизированных системах управления технологическими процессами — АСУ ТП), а в других случаях (системы обеспечения аналитических исследований, мониторинга социально-экономических или экологических процессов и т. п.) представляет сложную, многокритериальную классификационную задачу, слабо поддающуюся автоматизации и выполняемую специальной категорией работников.

*Организационные схемы, порядок отбора и фиксации информации* определяют организационную основу подсистемы сбора информации и обусловливаются характером и другими параметрами источников и каналов получения информации.

***Комплектование*** информационной базы в различных ее формах в общем плане включает *предварительную обработку (рубрикацию, структуризацию) и занесение информации.*

Характер *предварительной обработки* информации определяется формой представления входных данных (структурированная, неструктурированная), особенностями представления данных в информационной системе и может включать *классификацию* сведений по определенным рубрикам (делам), разделам и т. п. предметной области с целью накопления материалов определенного характера, или объединенных каким-либо признаком, фактором. *Структуризация* информации представляет процесс преобразования документированной информации (т. е. информации на естественном неформализованном языке — обычно это текст документа) в семантику АИС, т. е. в информационный язык представления данных, используемый в конкретной информационной системе.

*Занесение* данных в информационную систему заключается в добавлении новых сведений и, при необходимости, их отождествлении, слиянии и установлении взаимосвязи новых данных с ранее накопленными. Принципиальное значение при этом имеет вопрос идентификации новых данных с возможно уже имеющимися в системе.

Важным моментом при занесении новой информации является также установление ее логической взаимосвязи с ранее введенными данными. В некоторых видах АИС (информационно-поисковые) такая задача является одной из основных, так как позволяет искать и устанавливать не всегда очевидные связи между информационными объектами и категориями сведений информационной системы. Комплектование информацией в таких случаях неотделимо от обработки и выдачи информации.

В информационных службах, обеспечивающих создание и эксплуатацию информационных систем, сбор и комплектование информации осуществляют работники ***группы отбора*** («отборщики», «индексаторы»), квалификация которых помимо информационной должна включать также знание конкретной предметной области информационных систем.

***Поиск и выдача данных*** включают *установление специального организационно-технологического порядка удовлетворения информационных потребностей абонентов информационной системы в управленческой деятельности и технологических процессах.*

Удовлетворение информационных потребностей осуществляется через *периодический плановый поиск и выдачу сведений, оповещение* и *обработку запросов,* выполняемую организационными структурами (СлДОУ, информационная служба), организующими и эксплуатирующими информационную систему.

*Периодический плановый поиск и выдача сведений* осуществляются в целях обеспечения процессов организации, планирования и осуществления конкретного вида деятельности, т. е. в основном для информационного обеспечения организационно-управленческой деятельности. Данного рода задачи включаются в функции и обязанности информационных, информационно-аналитических и других информационно-обеспечивающих служб и заключаются, как правило, в формировании и выдаче статистических и сводных данных по периодически повторяющимся ситуациям в управленческой и производственной сфере.

*Оповещение и обработка запросов* представляют собой формы *информационного обслуживания* управленческих и производственно-технологических структур. *Оповещение* может осуществляться в форме *инициативно-сигнального* оповещения, *объектового* и *планово-периодического* оповещения.

*Объектовое оповещение* обычно осуществляется через выдачу абонентам информационной системы любых новых данных по определенному объекту, тематике, событию и т. п., появляющихся в АИС из любых источников.

*Планово-периодическое оповещение* производится через выдачу абонентам всех новых данных, поступивших к определенному плановому сроку из всех источников в информационную систему по определенному объекту, тематике, событию, проблеме.

*Обработка запросов и выдача по ним сведений* является одной из основных функций информационных служб. Данная деятельность регламентируется по вопросам инициирования, санкционирования и формы подачи запросов, форм и способов выдачи информации по запросам, учета запросов и т. д.

***Поддержание целостности и сохранности информации,*** пересмотр, ревизия и отсеивание утратившей актуальность информации являются неотъемлемой функцией информационных подразделений, создающих и поддерживающих информационные системы. Данные задачи решаются категорией работников, называемых ***администраторами АИС.*** Администраторы обеспечивают создание и поддержание банков данных АИС, организацию разграничения доступа к ним, защиту информации от несанкционированного доступа (НСД), ее резервирование и восстановление при разрушении или утрате ее целостности вследствие преднамеренных и непреднамеренных воздействий или ситуаций. Подобного рода задачи требуют высокой квалификации персонала и выполняются наиболее подготовленными информационными работниками.

Периодическая ревизия информации в банках данных АИС призвана проверить целостность (не нарушены ли внутренние взаимосвязи информационных объектов) и сохранность данных, а также удалить из АИС информацию, потерявшую свою актуальность. Удаление информации из АИС, как и ее занесение в АИС, регламентируется специальными нормативно-инструктивными документами.

## 1.2. Структура и классификация информационных систем

В составе информационной системы можно выделить три подсистемы, представленные на рис. 1.2.


*Рис. 1.2. Состав и функциональные группы информационной системы*

***Организационно-технологическая подсистема сбора информации*** обеспечивает отбор и накопление данных в информационную систему и включает совокупность источников информации, организационно-технологические цепочки отбора информации для накопления в системе. Без правильно организованной, оперативно и эффективно действующей организационно-технологической подсистемы сбора информации невозможна эффективная организация функционирования всей информационной системы в целом.

***Подсистема представления и обработки информации*** составляет ядро информационной системы и является отражением представления разработчиками и абонентами системы структуры и картины предметной области, сведения о которой должна отражать информационная система. Подсистема представления и обработки информации является одним из наиболее сложных компонентов при разработке информационной системы.

***Нормативно-функциональная подсистема выдачи информации*** определяет пользователей, или иначе *абонентов,* системы, реализует целевой аспект назначения и выполнения задач информационной системы.

*Информационным ядром* (информационным фондом) подсистемы представления и обработки информации АИС, или, говоря иначе, внутренним носителем знаний о предметной области является *база данных* (БД). Понятие базы данных является центральным в сфере технологий автоматизированных информационных систем. В справочной литературе по информатике приводится следующее определение базы данных — *«совокупность данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимая от прикладных программ».** Нормативно-правовая трактовка понятия базы данных представлена в законе «О правовой охране программ для ЭВМ и баз данных», согласно которому *«база данных — это объективная форма представления и организации совокупности данных (например, статей, расчетов), систематизированных таким образом, чтобы эти данные могли быть найдены и обработаны с помощью ЭВМ».*

* Толковый словарь по информатике.—М.: Финансы и статистика, 1991.

Другим фундаментальным понятием, непосредственно связанным с АИС, является ***система управления базами данных*** (***СУБД***), которая по ГОСТу определяется как «*совокупность программ и языковых средств, предназначенных для управления данными в базе данных, ведения базы данных и обеспечения взаимодействия ее с прикладными программами*». В настоящее время развитие СУБД как специального вида программного обеспечения для создания и эксплуатации АИС приводит к более широким санкциям СУБД. Ввиду этого в расширенном толковании СУБД можно определить как *комплекс программных средств, реализующих создание баз данных, их поддержание в актуальном состоянии, а также обеспечивающих различным категориям пользователей возможность получать из БД необходимую информацию*.

Совокупность конкретной базы данных, СУБД, прикладных компонентов АИС (набор входных и выходных форм, типовых запросов для решения информационно-технологических задач в конкретной предметной области), а также комплекса технических средств, на которых они реализованы, образуют ***банк данных*** (БнД), или иначе ***автоматизированный банк данных*** (АБД). Таким образом, соотношение понятий БнД, СУБД и БД можно проиллюстрировать схемой, приведенной на рис. 1.3.

По характеру представления и логической организации хранимой информации АИС разделяются на ***фактографические, документальные и геоинформационные.***

***Фактографические*** АИС накапливают и хранят данные в виде множества экземпляров одного или нескольких типов структурных элементов *(информационных объектов).* Каждый из таких экземпляров структурных элементов или некоторая их совокупность отражают сведения по какому-либо факту, событию и т. д., отделенному (вычлененному) от всех прочих сведений и фактов.* Структура каждого типа информационного объекта состоит из конечного набора реквизитов, отражающих основные аспекты и характеристики сведений для объектов данной предметной области. К примеру, фактографическая АИС, накапливающая сведения по лицам, каждому конкретному лицу в базе данных ставит в соответствие запись, состоящую из определенного набора таких реквизитов, как фамилия, имя, отчество, год рождения, место работы, образование и т. д. Комплектование информационной базы в фактографических АИС включает, как правило, обязательный процесс структуризации входной информации из документального источника. Структуризация при этом осуществляется через определение (выделение, вычленение) экземпляров информационных объектов определенного типа, информация о которых имеется в документе, и заполнение их реквизитов.

* Отсюда и название—«фактографические системы».

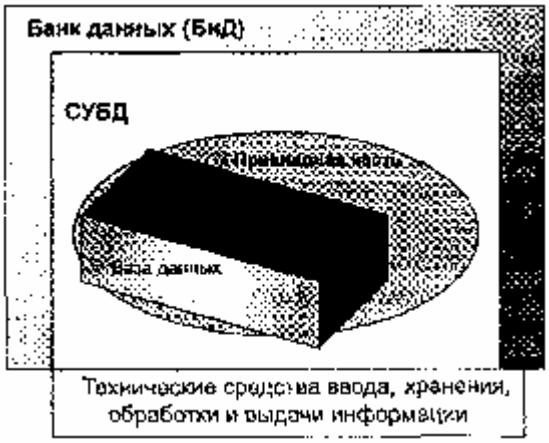



*Рис. 1.3. Соотношение понятий БнД, СУ БД и БД*

В ***документальных*** АИС единичным элементом информации является нерасчлененный на более мелкие элементы документ и информация при вводе (входной документ), как правило, не структурируется, или структурируется в ограниченном виде. Для вводимого документа могут устанавливаться некоторые формализованные позиции — дата изготовления, исполнитель, тематика и т. д. Некоторые виды документальных АИС обеспечивают установление логической взаимосвязи вводимых документов — соподчиненность по смысловому содержанию, взаимные отсылки по каким-либо критериям и т. п. Определение и установление такой взаимосвязи представляет собой сложную многокритериальную и многоаспектную аналитическую задачу, которая не может в полной мере быть формализована.

В ***геоинформационных*** АИС данные организованы в виде отдельных информационных объектов (с определенным набором реквизитов), привязанных к общей электронной топографической основе (электронной карте). Геоинформационные системы применяются для информационного обеспечения в тех предметных областях, структура информационных объектов и процессов в которых имеет пространственно-географический компонент, например маршруты транспорта, коммунальное хозяйство и т. п.

Разработка и проектирование информационной системы начинаются с построения концептуальной модели ее использования. ***Концептуальная модель использования*** информационной системы определяет, прежде всего, круг *конкретных задет и функций,* обеспечиваемых созданием и эксплуатацией информационной системы, а также систему сбора, накопления и выдачи информации.

Поэтому другим критерием классификации АИС являются ***функции и решаемые задачи,*** основными из которых могут являться:

• справочные;
• поисковые;
• расчетные;
• технологические.

***Справочные*** функции являются наиболее распространенным типом функций информационных систем и заключаются в предоставлении абонентам системы возможностей получения установочных данных на определенные классы объектов (Лица, Организации, Телефоны, Адреса и т. п.) с жестко или произвольно заданным набором сведений. *Видами* информационных систем, реализующих чисто справочные функции, являются всевозможные *электронные справочники, картотеки, программные* или *аппаратные «электронные записные книжки»* и их более развитые аналоги в виде т.н. *персональных информационных систем.*

Системы, реализующие ***поисковые*** функции, являются наиболее широко распространенным классом информационных систем, которые чаще всего называют *информационно-поисковыми* системами (ИПС). ИПС в общем виде можно рассматривать как некое *информационное пространство,* задаваемое в терминах информационно-логического описания предметной области — «информационные объекты», «информационные связи». Пользователям ИПС предоставляется возможность поиска и получения сведений по различным *поисковым образам* в таком информационном пространстве.

***Расчетные*** функции информационных систем заключаются в обработке информации, находящейся в системе, по определенным расчетным алгоритмам для различных целей. К числу подобных задач относится вычисление определенных статистических характеристик и показателей по экземплярам

различных типов объектов и отношений, данные по которым накапливаются в системе. Широко применяющейся разновидностью расчетных информационных систем являются различные системы автоматического проектирования, всевозможные бухгалтерские и финансово-экономические системы.

***Технологические*** функции информационных систем заключаются в автоматизации всего технологического цикла или отдельных его компонент, какой-либо производственной или организационной структуры. К системам, обеспечивающим подобные задачи, относится широкий класс автоматизированных систем управления (АСУ, АСУ ТП). Другой разновидностью технологических информационных систем являются системы автоматизации документооборота.

Рассмотренная классификация автоматизированных информационных систем, как и всякая классификация, условна и на практике конкретная АИС может характеризоваться комплексным характером представления информации (например, являться фактографически-документальной системой) и решать комплекс справочных, поисковых, расчетных и технологических задач.

## 1.3. Система представления и обработки данных фактографических АИС

В архитектуре подсистемы представления и обработки информации фактографических АИС можно выделить различные уровни представления информации, отображенные на рис. 1.4.

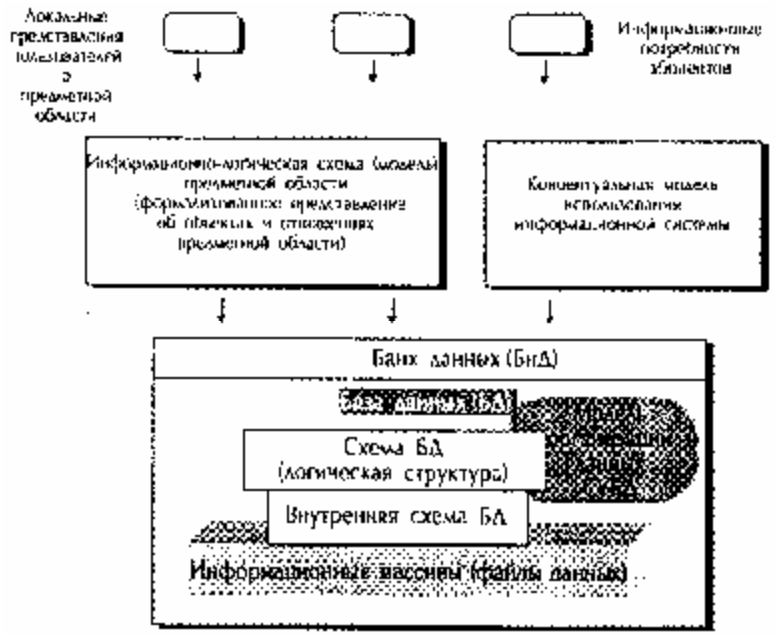



*Рис. 1.4. Уровни представления информации в АИС*

*Начальный уровень* определяется локальными представлениями о предметной области пользователей-абонентов информационной системы и их представлениями о своих информационных потребностях. На основе анализа этих представлений определяется ***информационно-логическая*** или сокращенно *инфологическая* ***схема*** предметной области, подлежащей отображению информационной системой, и концептуальная *модель использования* информационной системы. Инфологическая схема представляет собой формализованное представление (описание) объектов и отношений фрагмента действительности.

Наиболее часто формализация представлений о предметной области осуществляется в рамках модели «объекты-связи» (так называемая ***ER-людель*** — от англ. *Entity Relationship).* При этом под ***информационным объектом*** в общем плане понимается некоторая сущность фрагмента действительности, например организация, документ, сотрудник, место, событие и т. д. В предметной области выделяются различные *типы* объектов, представляемые в информационной системе в каждый момент времени конечным набором *экземпляров* данного типа. Каждый тип объекта включает (идентифицируется) присущий ему набор *атрибутов* (свойств, характерных признаков, параметров). ***Атрибут*** представляет логически неделимый элемент структуры информации, характеризующийся множеством атомарных значений. Для примера можно привести атрибут «Имя» объекта типа «Лицо», который характеризуется множеством всех возможных имен, и атрибут «Текст» объекта типа «Документ», который характеризуется множеством средств смыслового выражения в определенном национальном языке.

Экземпляр объекта образуется совокупностью конкретных значений атрибутов данного типа объекта. Один или некоторая группа атрибутов объекта данного типа могут исполнять роль ***ключевого атрибута,*** по которому идентифицируются (различаются) конкретные экземпляры объектов. К примеру, для объектов типа «Лицо» ключом может являться совокупность атрибутов «Фамилия», «Имя», «Отчество» или один атрибут, выражающий номер паспорта (удостоверения личности).

Различные типы объектов и различные экземпляры одного типа объекта могут быть охвачены определенными отношениями, которые в рамках ER-модели выражаются т. н. связями. Так, например, объекты «Сотрудник» и «Организация» могут быть охвачены отношением «Работа», т. е. связаны этим отношением. При этом ***связи*** могут быть двух типов — ***иерархические,*** или, иначе говоря, структурные (владелец-подчиненный) и ***одноуровневые,*** например, родственная связь «Брат-сестра» между двумя экземплярами объекта типа «Лицо» (в отличие от иерархической родственной связи—«Отец-сын»). Объекты-владельцы иерархических связей-отношений иногда называют *структурными* объектами, в противовес *простым* объектам, которые таковыми не являются (не являются владельцами).

Структурные и одноуровневые связи (отношения), в свою очередь, по признаку множественности могут быть трех типов — *«один-к-одному»* (например, отношение «Лицо-Паспорт», имея в виду под «Паспортом» не атрибут объекта Лицо, а самостоятельный объект, состоящий из атрибутов «Номер», «Вид паспорта», «Владелец», «Место выдачи», «Дата выдачи» и т. д.), *«один-ко-многим»* (например, отношение «Подразделение-Сотрудник», имея в виду, что в одном подразделении может работать много сотрудников, но каждый сотрудник работает только в одном подразделении) и *«многие-ко-многим»* (например, отношение «Лицо-Документ», имея в виду, что один человек может быть автором, или иметь какое-либо другое отношение ко многим документам, и, в свою очередь, один документ может иметь много авторов.

Помимо этого информационные потребности абонентов информационной системы могут включать также и оперирование ***опосредованными*** (т. е. косвенными, непрямыми, ассоциативными) связями. Примерами таких непрямых связей является совместная работа нескольких человек на одном предприятии (подразделении). Прямая непосредственная связь в данном случае, как правило, устанавливается только между объектами «Лицо» и «Организация», но не между различными экземплярами объекта «Лицо».

Одним из способов *представления* формализованного описания предметной области информационной системы в рамках модели «объекты-связи» является использование техники специальных диаграмм, которая была предложена известным американским специалистом в области баз данных Ч. Бахманом. В ***диаграммах Бахмана*** объекты (сущности) представляются вершинами некоторого математического графа, а связи —дугами графа. Виды и свойства связей-отношений объектов отображаются направленностью, специальным оформлением дуг и расположением вершин графа.

В качестве примера можно привести инфологическую схему предметной области сведений информационной системы, предназначенной для накопления данных о научной работе в каком-либо учебном или исследовательском учреждении (см. рис. 1.5).


*Рис. 1.5. Мифологическая схема предметной области информационной системы со сведениями о научной работе*

На приведенном рисунке однонаправленность дуг означает структурность связи «владелец-подчиненный», двунаправленность дуг означает одноуровневые связи, двойные стрелки означают

множественность отношения «один-ко-многим», двунаправленность двойных стрелок означает одноуровневые отношения «многие-ко-многим».

Одним из недостатков использования ER-диаграмм Бахмана для описания формализованных схем (моделей) предметных областей информационных систем является их статичность, не позволяющая наглядно и непосредственно отображать *процессы,* в которые вовлечены сущности и которым подвержены отношения (связи). Отчасти подобные проблемы преодолеваются введением дополнительных сущностей, выражающих собственно процессы и ситуации — событие, действие, момент времени. Аналогичным образом в некоторых случаях вводятся пространственные сущности для адекватного представления сущностей и отношений предметной области—маршрут, место, населенный пункт, здание, элемент здания, зона и т. д.

*Вторым уровном* представления информации в информационной системе (см. рис. 1.4) является ***схема базы дачных,*** (называемая еще *логической структурой данных),* представляющая описание средствами конкретной СУБД инфологической схемы предметной области (информационные объекты, реквизиты, связи).

Совокупность средств и способов реализации схемы базы данных в конкретной СУБД составляет ***модель организации данных.***

Схема базы данных содержит также ***ограничения целостности данных.*** Ограничения целостности представляют собой набор установок и правил по типам, диапазонам, соотношениям (и т. д.) значений атрибутов объектов, характеристик и особенностей связей между объектами. К примеру, диапазон значения атрибута «Дата рождения» объекта лицо не может выходить за рамки текущей даты, значение атрибута «Дата приобретения» объекта «Имущество» не может быть позднее значения атрибута «Дата продажи», значение атрибута «Количество» объекта «Материал» не должно быть меньше минимально необходимого на складе и т. п. Ограничения целостности данных лежат в основе контроля корректности информации при ее вводе в систему и периодического контроля наличия смысловых и других ошибок в базе данных после проведения операций добавления, удаления и изменения данных.

*Третий и самый «низкий» уровень* представления информации в фактографических информационных системах выражается ***внутренней схемой базы данных,*** определяющей структуру организации и особенности хранения информационных массивов, в которых и находятся собственно сами данные (см. рис. 1.4).

Более конкретные особенности представления и организации данных определяются конкретным типом и особенностями СУБД, используемой для создания фактографической информационной системы.

### Вопросы и упражнения

1. Разъясните соотношение и взаимосвязь понятий «информация», «знания», «сведения» и «данные».
2. Охарактеризуйте в терминах информационного обеспечения листок по учету кадров и автобиографию, заполняемые и составляемые сотрудником при приеме на работу.
3. Бланк заказа заполняется клиентом, подписывается работником организации, регистрируется, исполняется и подшивается в соответствующее дело. Охарактеризуйте в терминах информационного обеспечения указанные операции и процессы.
4. Является ли обязательным использование СВТ в информационных системах? Каковы главные признаки информационной системы и чем она отличается от простой совокупности информационных ресурсов?
5. Одной из задач работника информационно-аналитической службы является поиск, сбор и систематизация всех публикаций по определенной проблеме. Как можно охарактеризовать эти функции с точки зрения структуры и этапов информационного обеспечения?
6. К какому этапу цикла функционирования информационных систем относится извещение абонента читального зала об исполнении заказа на интересующую его книгу, журнал?
7. Чем отличается база данных от информационного массива?
8. Каково соотношение понятий банка данных и базы данных?
9. К какому типу информационных систем можно отнести картотеку личных дел сотрудников организации?
10. Какого типа информационную систему наиболее целесообразно создавать для информационного обеспечения снабжения товарами сети магазинов торговой компании?

11. Чем отличается инфологическая схема предметной области информационной системы от схемы ее базы данных?
12. Какие атрибуты или совокупности атрибутов объектов «Образование» (наименование учебного заведения, год поступления, год окончания, квалификация, специальность, № диплома), «Имущество» (инв. №, наименование, тип, дата приобретения) могут исполнять роль ключей?
13. Каковы типы отношений-связей между объектами (сущностями) «Счет»—«Банк», «Товары»—«Поставщики», «Студенты»—«Преподаватели» и «Автомобиль»—«Паспорт транспортного средства (ПТС)»?

# 2. СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ ФАКТОГРАФИЧЕСКИХ ИНФОРМАЦИОННЫХ СИСТЕМ

История СУБД как особого вида программного обеспечения неразрывно связана с историей начала использования электронно-вычислительных машин для организации хранения и обработки информации. Именно в то время (конец 60-х, начало 70-х годов) были разработаны основы программного обеспечения для создания и эксплуатации фактографических информационных систем. В конце 70-х, начале 80-х годов направление программного обеспечения под общим названием «СУБД» превратилось в одну из наиболее бурно развивающихся отраслей программной индустрии.

При этом основные программно-математические и технологические решения по СУБД были разработаны в 70-х годах в ряде крупных исследовательских проектов. Наиболее известными из них являются проект «Рабочей группы по базам данных» КОДАСИЛ (DBTG CODASY*L*) с участием уже упоминавшегося Ч. Бахмана, пионерские работы основателя теории реляционных баз данных Е. Кодда, проект разработки системы управления реляционными базами данных «System R» фирмы IBM (1975-1979 гг.) и проект разработки СУБД «Ingres» (Interactive Graphics and Retrieval System) в университете Беркли (1975-1980 гг.) под руководством известного специалиста в области баз данных М. Стоунбрейкера.

## 2.1. Функции, классификация и структура СУБД

С начала своего возникновения в конце 60-х годов автоматизированные информационные системы ориентировались на хранение и обработку больших объемов данных, которые не могли быть одновременно и полностью размещены в оперативной памяти ЭВМ.

В структуре программного обеспечения ЭВМ, как в то время, так и сейчас, за организацию, размещение и оперирование данными во внешней (долговременной) памяти отвечает операционная система ЭВМ, соответствующий компонент которой чаще всего называется «файловой системой». Данные во внешней памяти компьютера представлены именованными совокупностями, называемыми файлами. В большинстве случаев операционная (файловая) система не «знает» внутренней смысловой логики организации данных в файлах и оперирует с ними как с однородной совокупностью байтов или строк символов.

С точки зрения смысла и назначения АИС файлы данных имеют структуру, отражающую информационно-логическую схему предметной области АИС. Эта структура данных в файлах должна обязательно учитываться в операциях обработки (собственно, в этом и заключается одна из основных функций АИС). Вместе с тем, в силу невозможности в большинстве случаев размещения файлов баз данных сразу целиком в оперативной памяти компьютера, структуру данных в файлах баз данных приходится учитывать при организации операций обращения к файлам во внешней памяти.

Отсюда вытекает основная особенность *СУБД* как вида программного обеспечения. Будучи по природе *прикладным программным обеспечением,* т. е. предназначенным для решения конкретных прикладных задач, СУБД изначально выполняли и *системные функции* — расширяли возможности файловых систем *системного программного обеспечения.*

В общем плане можно выделить следующие ***функции,*** реализуемые ***СУБД:***

• организация и поддержание логической структуры данных (схемы базы данных);
• организация и поддержание физической структуры данных во внешней памяти;
• организация доступа к данным и их обработка в оперативной и внешней памяти.

***Организация и поддержание логической структуры данных*** (схемы базы данных) обеспечивается средствами *модели организации данных.** ***Модель данных*** определяется способом организации данных, ограничениями целостности и множеством операций, допустимых над объектами

организации данных. Соответственно модель данных разделяют на три составляющие — *структурную, целостную* и *манипуляционную.*

* В обиходе просто «модель данных».

Известны три основные модели организации данных:
• иерархическая;
• сетевая;
• реляционная.

Модель данных, реализуемая СУБД, является одной из основных компонент, определяющих функциональные возможности СУБД по отражению в базах данных информационно-логических схем предметных областей АИС. Модель организации данных, по сути, определяет *внутренний информационный язык* автоматизированного банка данных, реализующего автоматизированную информационную систему.

Модели данных, поддерживаемые СУБД, довольно часто используются в качестве критерия для классификации СУБД. Исходя из этого, различают ***иерархические СУБД, сетевые СУБД*** и ***реляционные СУБД***.

Другой важной функцией СУБД является ***организация и поддержание физической структуры данных во внешней памяти.*** Эта функция включает организацию и поддержание внутренней структуры файлов базы данных, иногда называемой ***форматом файлов базы дачных,*** а также создание и поддержание специальных структур (индексы, страницы) для эффективного и упорядоченного доступа к данным. В этом плане эта функция тесно связана с третьей функцией СУБД — организацией доступа к данным.

Организация и поддержание физической структуры данных во внешней памяти может производиться как на основе штатных средств файловых систем, так и на уровне непосредственного управления СУБД устройствами внешней памяти.

***Организация доступа к данным и их обработка в оперативной и внешней памяти*** осуществляется через реализацию процессов, получивших название транзакций. ***Транзакцией*** *называют последовательную совокупность операций, имеющую отдельное смысловое значение по отношению к текущему состоянию базы данных*. Так, например, транзакция по удалению отдельной записи в базе данных последовательно включает определение страницы файла данных, содержащей указанную запись, считывание и пересылку соответствующей страницы в буфер оперативной памяти, собственно удаление записи в буфере ОЗУ, проверку ограничений целостности по связям и другим параметрам после удаления и, наконец, «выталкивание» и фиксацию в файле базы данных нового состояния соответствующей страницы данных.

Транзакции принято разделять на две разновидности — изменяющие состояние базы данных после завершения транзакции и изменяющие состояние БД лишь временно, с восстановлением исходного состояния данных после завершения транзакции. Совокупность функций СУБД по организации и управлению транзакциями называют ***монитором транзакций.***

Транзакции в теории и практике СУБД по отношению к базе данных выступают внешними процессами, отождествляемыми с действиями пользователей банка данных. При этом источником, инициатором транзакций может быть как один пользователь, так и несколько пользователей сразу. По этому критерию СУБД классифицируются на ***однопользовательские*** (или так называемые «настольные») и ***многопользовательские*** («тяжелые», «промышленные») СУБД. Соответственно в многопользовательских СУБД главной функцией монитора транзакций является обеспечение эффективного совместного выполнения транзакций над общими данными сразу от нескольких пользователей.

Непосредственная обработка и доступ к данным в большинстве СУБД осуществляется через организацию в оперативной памяти штатными средствами операционной системы или собственными средствами системы *буферов оперативной памяти,* куда на время обработки и доступа помещаются отдельные компоненты файла базы данных (страницы). Поэтому другой составной частью функций СУБД по организации доступа и обработки данных является ***управление буферами оперативной памяти.***

Еще одной важной функцией СУБД с точки зрения организации доступа и обработки данных является так называемая журнализация всех текущих изменений базы данных. ***Журнализация*** представляет собой основное средство обеспечения сохранности данных при всевозможных сбоях и разрушениях данных. Во многих СУБД для нейтрализации подобных угроз создается журнал изменений базы

данных с особым режимом хранения и размещения. Вместе с установкой режима периодического сохранения резервной копии БД журнал изменений* при сбоях и разрушениях данных позволяет восстанавливать данные по произведенным изменениям с момента последнего резервирования до момента сбоя. Во многих предметных областях АИС (например, БД с финансово-хозяйственными данными) такие ситуации сбоя и порчи данных являются критическими и возможности восстановления данных обязательны для используемой СУБД.

* Резервная копия БД и журнал изменений, как правило, размещаются на отдельных от основного файла БД носителях.

Исходя из рассмотренных функций, в ***структуре СУБД*** в современном представлении можно выделить следующие ***функциональные блоки:***

- процессор описания и поддержания структуры базы данных;
- процессор запросов к базе данных;
- монитор транзакций;*
- интерфейс ввода данных;
- интерфейс запросов;
- интерфейс выдачи сведений;
- генератор отчетов.

* Как правило, в однопользовательских СУБД монитортранзакций в виде отдельного функционального элемента СУБД не реализуется и не выделяется.

Схематично взаимодействие компонент СУБД представлено на рис. 2.1.

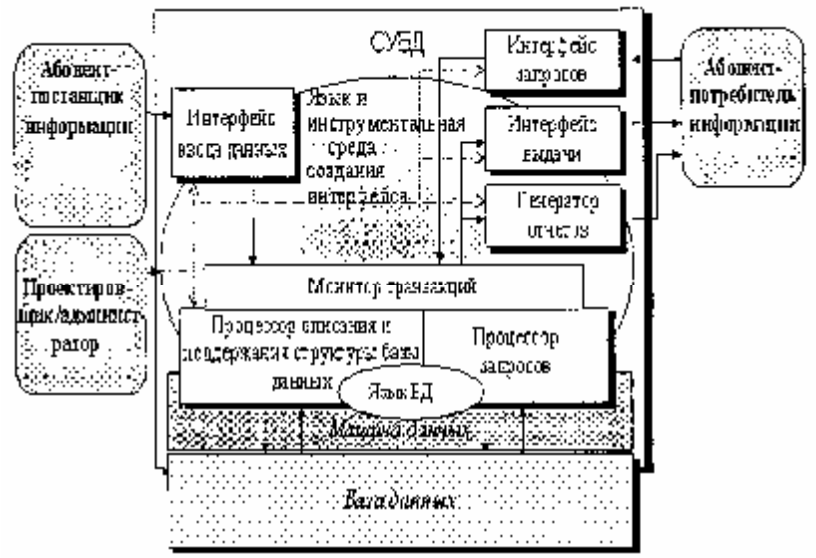



*Рис. 2.1. Структура и взаимодействие компонент СУБД*

Ядром СУБД является ***процессор описания и поддержания структуры базы данных.*** Он реализует модель организации данных, средствами которой проектировщик строит *логическую структуру (схему) базы данных,* соответствующую инфологической схеме предметной области АИС, и обеспечивает построение и поддержание *внутренней схемы базы данных.*

Процессором описания и поддержания структуры данных в терминах используемой модели данных (иерархическая, сетевая, реляционная) обеспечиваются установки заданной логической структуры базы данных, а также трансляция (перевод) структуры базы данных во внутреннюю схему базы данных (в физические структуры данных). В АИС на базе реляционных СУБД процессор описания и поддержания структуры базы данных реализуется на основе ***языка базы данных,*** являющегося составной частью ***языка структурированных запросов (SQL).***

***Интерфейс ввода данных СУБД*** реализует *входной информационный язык банка данных,* обеспечивая абонентам-поставщикам информации средства описания и ввода данных в информационную систему. Одной из современных тенденций развития СУБД является стремление приблизить входные информационные языки и интерфейс ввода к естественному языку общения с пользователем в целях упрощения эксплуатации информационных систем так называемых «неподготовленными» пользователями. Данная проблема решается через применение диалоговых методов организации интерфейса и использование ***входных форм.*** Входные формы, по сути, представляют собой

электронные аналоги различного рода анкет, стандартизованных бланков и таблиц, широко используемых в делопроизводстве и интуитивно понятных большинству людей (неподготовленных пользователей). Интерфейс ввода при этом обеспечивает средства создания, хранения входных форм и их интерпретацию в терминах описания логической структуры базы данных для передачи вводимых через формы сведений процессору описания и поддержания структуры базы данных.

***Интерфейс запросов*** совместно с процессором запросов обеспечивает концептуальную модель использования информационной системы в части стандартных типовых запросов, отражающих информационные потребности пользователей-абонентов системы. Интерфейс запросов предоставляет пользователю средства выражения своих информационных потребностей. Современной тенденцией развития СУБД является использование диалогово-наглядных средств в виде специальных «конструкторов» или пошаговых «мастеров» формирования запросов.

***Процессор запросов*** интерпретирует сформированные запросы в терминах ***языка манипулирования данными*** и совместно с процессором описания и поддержания структуры базы данных собственно и исполняет запросы. В реляционных СУБД основу процессора запросов составляет язык манипулирования данными, являющийся основной частью языка SQ*L*. Тем самым на базе процессора запросов и процессора описания и поддержания структуры базы данных образуется низший уровень оперирования данными в СУБД, который иногда называют ***машиной данных.*** Стандартные функции и возможности машины данных используют компоненты СУБД более высокого порядка (см. рис. 2.1), что позволяет разделить и стандартизировать компоненты СУБД и банка данных на три уровня —логический уровень, машина данных и собственно сами данные.

Функции ***монитора транзакции,*** как уже отмечалось, заключаются в организации совместного выполнения транзакций от нескольких пользователей над общими данными. При этом дополнительной функцией, неразрывно связанной, в том числе и с основной функцией, является обеспечение целостности данных и ограничений над данными, определяемыми правилами предметной области АИС.

***Интерфейс выдачи*** СУБД получает от процессора запросов результаты исполнения запросов (обращений к базе данных) и переводит эти результаты в форму, удобную для восприятия и выдачи пользователю-абоненту информационной системы. Для отображения результатов исполнения запросов в современных СУБД используются различные приемы, позволяющие «визуализировать» данные в привычной и интуитивно понятной неподготовленному пользователю форме. Обычно для этого применяются табличные способы представления структурированных данных, а также специальные ***формы выдачи*** данных, представляющие также, как и ***формы ввода,*** электронные аналоги различных стандартизованных бланков и отчетов в делопроизводстве.

Формы выдачи лежат также и в основе формирования так называемых ***«отчетов»,*** выдающих результаты поиска и отбора информации из БД в письменной форме для с формализованного создания соответствующих текстовых документов, т. е. для документирования выводимых данных. Для подобных целей в состав современных СУБД включаются ***генераторы отчетов.***

В заключение по структуре и составу СУБД следует также добавить, что современные программные средства, реализующие те или иные СУБД, представляют собой совокупность ***инструментальной среды создания и использования баз данных*** в рамках определенной модели данных (реляционной, сетевой, иерархической или смешанной) и ***языка СУБД*** (язык описания данных, язык манипулирования данными, язык и средства создания интерфейса). На основе программных средств СУБД проектировщики строят в целях реализации конкретной информационной системы (инфологическая схема предметной области, задачи и модель использования, категории пользователей и т. д.) автоматизированный банк данных, функционирование которого в дальнейшем поддерживают администраторы системы и услугами которого пользуются абоненты системы.

## 2.2. Модели организации данных

### 2.2.1. Иерархическая и сетевая модели организации данных

В ***иерархической*** модели объекты-сущности и отношения предметной области представляются наборами данных, которые имеют строго *древовидную структуру,* т. е. допускают только иерархические (структурные) связи-отношения. Иерархическая модель данных была исторически первой, на основе которой в конце 60-х-начале 70-х годов были разработаны первые профессиональные СУБД-СУБД IMS (Information Management System) фирмы IBM, СУБД Total для

компьютеров HP3000. К иерархическим СУБД также относятся отечественные промышленные СУБД 70-80-х годов «ОКА» и «ИНЭС».

База данных с иерархической моделью данных состоит из упорядоченного набора экземпляров структуры типа «дерево», что иллюстрируется примером на рис. 2.2.

В приведенном примере информационный объект «Отделы» является предком информационного объекта «Подразделения», который, в свою очередь, является предком информационного объекта «Сотрудники». Объект «Подразделения» является потомком объекта «Отделы», а объект «Сотрудники» потомком объекта «Подразделения». Экземпляры потомка с общим предком называются близнецами.

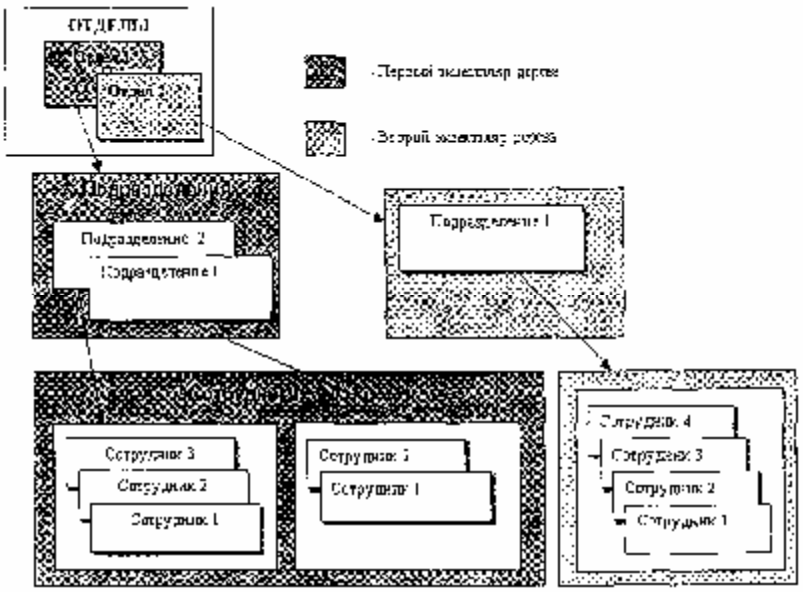


*Рис. 2.2. Пример иерархической организации данных*

В иерархической модели устанавливается *строгий порядок обхода* дерева (сверху-вниз, слева-направо) и следующие *операции над данными:*

• найти указанное дерево (например, отдел № 3);
• перейти от одного дерева к другому;
• перейти от одной записи к другой (например, от отдела к первому подразделению);
• перейти от одной записи к другой в порядке обхода иерархии;
• удалить текущую запись.

Основное внимание в *ограничениях целостности* в иерархической модели уделяется целостности ссылок между предками и потомками с учетом основного правила: никакой потомок не может существовать без родителя.

***Сетевая модель*** является расширением иерархической и широко применялась в 70-е годы в первых СУБД, использовавшихся крупными корпорациями для создания информационных систем (СУБД IDMS — Integrated Database Management System компании Cullinet Software Inc., СУБД IDS, отечественные СУБД «СЕТЬ», «БАНК», «СЕТОР»). Одним из идеологов концепции сетевой модели являлся Ч. Бахман. Эталонный вариант сетевой модели данных, разработанный с участием Бахмана, был описан в проекте «Рабочей группы по базам данных» КОДАСИЛ (DBTG CODASY*L*).

В отличие от иерархической, в сетевой модели объект-потомок может иметь не одного, а вообще говоря, любое количество объектов-предков. Тем самым допускаются *любые связи-отношения,* в том числе и *одноуровневые.* В результате сущности и отношения предметной области АИС представляются графом любого (не только древовидного) типа. Пример такой организации данных приведен на рис. 2.3.

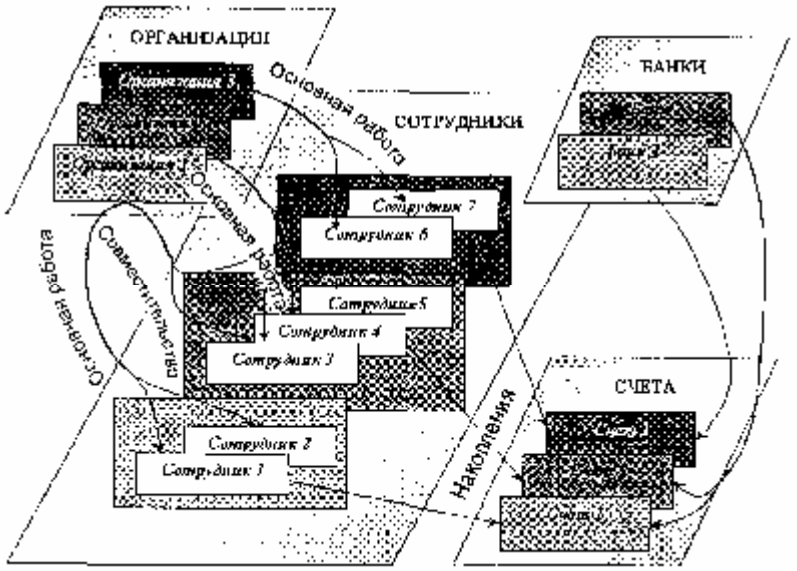



*Рис. 2.3. Пример сетевой организации данных*

Сетевая СУБД состоит из одного или нескольких *типов записей* (типов информационных объектов) и набора *типов связей* между ними. Каждый тип записей представлен в БД набором экземпляров записей данного типа. Аналогично каждый тип связи представлен набором экземпляров связей данного типа между конкретными экземплярами типов записей. В приведенном на рис. 2.3 примере типами записей являются «Организация», «Сотрудник», «Банк», «Счет», а типами связей — «Совместительство», «Основная работа», «Вклады», «Накопления». При этом тип записи «Счет» имеет двух предков — «Сотрудник» и «Банк», экземпляр типа записи «Сотрудник» может иметь два предка (по связям «Основная работа» и «Совместительство»), являющихся различными экземплярами типа записи «Организация».

Для данного типа связи *L* между типом записи предка P и типом записи потомка C выполняются следующие условия:

• каждый экземпляр типа P является предком только в одном экземпляре *L,*

• каждый экземпляр C является потомком не более чем в одном экземпляре *L.*

В рамках сетевой модели возможны следующие ситуации:

• тип записи потомка в одном типе связи *L*1 может быть типом записи предка в другом типе связи *L*2 (как в иерархической модели);

• данный тип записи P может быть типом записи потомка в любом числе типов связи;

• может существовать любое число типов связи с одним и тем же типом записи предка и одним и тем же типом записи потомка;

• если *L*1 и *L*2 —два типа связи с одним и тем же типом записи предка P и одним и тем же типом записи потомка C, то правила, по которым образуется родство, в разных связях могут различаться;

• типы записей X и Y могут быть предком и потомком одной связи и потомком и предком в другой; предок и потомок могут быть одного типа записи (связь типа «петля»).

В сетевой модели устанавливаются следующие операции над данными:

• найти конкретную запись (экземпляр) в наборе однотипных записей;

• перейти от предка к первому потомку по некоторой связи;

• перейти к следующему потомку по некоторой связи;

• создать новую запись;

• уничтожить запись;

• модифицировать запись;

• включить в связь;

• исключить из связи;

• переставить в другую связь.

Реализация связей и сведений по ним в виде отдельных записей в БД обеспечивает одну важную отличительную особенность сетевых СУБД - навигацию по связанным данным. Сетевые СУБД обеспечивают возможность непосредственной ***«навигации»*** (перехода) от просмотра реквизитов экземпляра одного типа записи (например, «Организация») к просмотру реквизитов экземпляра связанного типа записей (например, «Сотрудник»). Тем самым пользователю предоставляется

возможность многокритериального анализа базы данных без непосредственной формализации своих информационных потребностей через формирование запросов на специальном языке, встроенном в СУБД. Поэтому СУБД с сетевой организацией данных иногда еще называют СУБД с навигацией.

Другой сильной стороной сетевой модели в немногих примерах современной реализации сетевых СУБД является также использование *множественных типов данных* для описания атрибутов информационных объектов (записей), что позволяет создавать информационные структуры, которые хорошо отражают традиционную табличную форму представления структурированных данных. К примеру, при описании типа записи «Сотрудник» в сетевой модели можно ввести реквизит «Имена детей», характер значений которого является множественным.

Сетевая модель позволяет наиболее адекватно отражать инфологические схемы сложных предметных областей. Вместе с тем, несмотря на появление в конце 70-х годов стандарта по сетевой модели данных КОДАСИЛ, не получила широкого распространения ни одна из попыток создания языковых программных средств, которые позволили бы в разных прикладных информационных системах одинаковым образом описывать данные с сетевой организацией. В результате в сетевых СУБД данные оставались жестко связанными как с самой СУБД, так и с прикладными компонентами АИС, что затрудняло специализацию в развитии программных компонент СУБД сетевого типа и объективно затормаживало процесс их развития.

### 2.2.2. Реляционная модель организации данных

В ***реляционной модели*** объекты-сущности инфологической схемы предметной области АИС представляются плоскими таблицами данных. Столбцы таблицы, называемые ***полями*** базы данных, соответствуют атрибутам объектов-сущностей инфологической схемы предметной области. Множество атомарных значений атрибута называется ***доменом.*** Так доменом для поля «Имя» является множество всех возможных имен. Различные атрибуты могут быть определены на одном и том же домене — например, атрибуты «Год поступления» (в вуз) и «Год окончания» определены на одном и том же домене, являющемся перечнем дат определенного диапазона.

Строки таблицы, представляющие собой различные сочетания значений полей из доменов, называются ***кортежами*** (в обиходе просто ***записями***) базы данных и соответствуют *экземплярам объектов-сущностей* инфологической схемы предметной области.

Считается, что сильной стороной реляционных баз данных является развитая математическая теория, лежащая в их основе—реляционная алгебра. Само слово «реляционная» происходит от англ. *relation* — отношение. Но в случае реляционных баз слово «отношение» выражает не взаимосвязь между таблицами-сущностями, а определение самой таблицы как математического отношения доменов.*

* Для любителей математики—*отношением* называется подмножество декартова произведения множеств, роль которых в данном случае играют домены, таким образом *таблицы* — это отношение доменов, а *строки таблицы (кортежи)* — элементы отношения доменов.

*Ключевому атрибуту* объекта-сущности, который идентифицирует (определяет, отличает от других) конкретный экземпляр объекта, в таблице соответствует ***ключевое поле*** (так называемый *ключ* таблицы). Примером ключа в таблице «Отделы» может быть поле «Номер отдела» или поле «Наименование отдела». В тех случаях, когда конкретную запись таблицы идентифицирует значение не одного поля, а совокупность значений нескольких полей, тогда все эти поля считаются ключевыми, а ключ таблицы является ***составным.*** Примером такой ситуации может служить таблица «Сотрудники», роль составного ключа в которой может играть совокупность полей «Фамилия», «Имя», «Отчество».*

* В той ситуации, когда исключается полное совпадение фамилии, имени и отчества у разных сотрудников.

Ключевое поле для созданной записи (заполненной строки таблицы) впоследствии обновиться (изменить значение) уже *не может.*

В некоторых таблицах роль ключа могут играть сразу несколько полей или групп полей. Например, в той же таблице «Сотрудники» может быть определен второй ключ «Номер паспорта», который также может идентифицировать конкретную запись (экземпляр) объекта «Сотрудник». В этих случаях один из ключей объявляется ***первичным.*** Значения непервичных ***ключей,*** которые называются ***возможными,*** в отличие от первичных ключей могут *обновляться.*

Совокупность определенных для таблицы-отношения полей, их свойства (ключи и пр.) составляют ***схему таблицы-отношения.*** Две отдельные таблицы-отношения с одинаковой схемой называются ***односхемными.***

Как уже отмечалось, таблица в реляционной модели отражает определенный объект-сущность из мифологической схемы предметной области АИС. *Отношения-связи* объектов-сущностей в реляционной модели устанавливаются через введение в таблицах дополнительных полей, которые *дублируют ключевые поля* связанной таблицы. К примеру, связь между таблицей «Сотрудники» и таблицей «Отделы» устанавливается через введение копии ключевого поля «Номер отдела» из таблицы «Отделы» в таблицу «Сотрудники» — см. рис. 2.4. Такие поля, дублирующие ключи связанной таблицы, называются ***внешними ключами***.

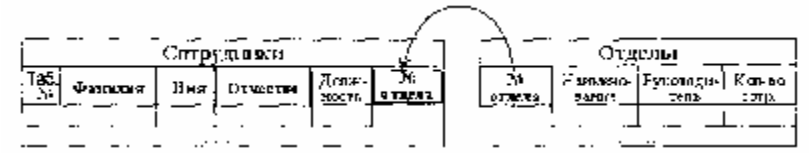



*Рис. 2.4. Пример связи в реляционных таблицах*

На приведенном рисунке ключевые поля таблиц обведены жирными рамками, а поле с внешним ключом — двойной рамкой.

Так как значения первичного ключа уникальны, т. е. не могут повторяться (в таблице «Отделы» может быть только один кортеж по, к примеру, 710-му отделу), а значения других полей и, в частности, внешнего ключа могут повторяться (в таблице «Сотрудники» может быть несколько строк-кортежей сотрудников по 710-му отделу), то такой механизм автоматически обеспечивает связь типа «Один-ко-многим». Отсюда также можно заключить, что связи между таблицами типа «Один-к-одному» в реляционной модели автоматически обеспечиваются при одинаковых первичных ключах, например между таблицей «Сотрудник» с ключом «Таб_№» и таблицей «Паспорт» с таким же ключом.

Другой вывод, который следует из анализа данного механизма реализации связей, заключается в том, что реляционная модель не может непосредственно отражать связи типа «Многие-ко-многим», что объективно снижает возможности реляционной модели данных при отражении сложных предметных областей.

Таким образом, *структурная составляющая реляционной модели* определяется небольшим набором базовых понятий — таблица-отношение, схема таблицы-отношения, домен, поле-атрибут, кортеж-запись (строка), ключ, первичный ключ, вторичный ключ, внешний ключ (отсылка). Данный набор понятий позволяет описывать естественным образом, близким к понятийному аппарату диаграмм Бахмана, большинство инфологических схем не слишком сложных предметных областей. Это обстоятельство как раз и способствовало интенсивному развитию реляционных СУБД в 80-х-90-х годах.

***Ограничения целостности*** (*целостная составляющая*) реляционной модели можно разделить на две группы — требование *целостности сущностей* и требование *целостности ссылок*.

Требование ***целостности сущностей*** в общем плане заключается в требовании *уникальности экземпляров* объектов инфологической схемы, отображаемых средствами реляционной модели. Экземплярам объектов инфологической схемы в реляционной модели соответствуют кортежи-записи таблиц-отношений. Поэтому требование *целостности сущностей* заключается в требовании уникальности каждого кортежа.

Отсюда вытекают следующие ограничения:

- отсутствие кортежей дубликатов (данное требование реализуется не через требование отсутствия совпадения значений одновременно по всем полям, а лишь по полям первичных ключей, что обеспечивает определенную гибкость в описании конкретных ситуаций в предметных областях АИС);
- отсутствие полей с множественным характером значений атрибута (данное ограничение по отношению к весьма типичным ситуациям при описании реальных предметных областей в реляционной модели обеспечивается так называемой *нормализацией* таблиц-отношений, т.е. разбиением исходной таблицы на две или более связанные таблицы с единичным характером значений полей-атрибутов).

Требование ***целостности ссылок*** заключается в том, что для любого кортежа-записи с конкретным значением внешнего ключа (отсылки) должен обязательно существовать кортеж связанной таблицы-

отношения с соответствующим значением первичного ключа. Простым примером этого очевидного требования является таблица-отношение «Сотрудники» с внешним ключом «№ отдела» и отсылаемая (связанная) таблица «Отделы» с первичным ключом «№ отдела» (см. рис. 2.4). Если существует кортеж-запись «Иванов», работающий в отделе № 710, то в таблице «Отделы» обязательно должен быть кортеж-запись с соответствующим номером отдела (каждый сотрудник должен обязательно в каком-либо отделе хотя бы числиться).

Теоретико-множественный характер реляционных таблиц-отношений требует также *отсутствия упорядоченности кортежей* и *отсутствия упорядоченности полей-атрибутов.* Отсутствие упорядоченности записей-кортежей в таблицах-отношениях усложняет поиск нужных кортежей при обработке таблиц. На практике с целью создания условий для быстрого нахождения нужной записи таблицы без постоянного упорядочения (переупорядочения) записей при любых изменениях данных вводят ***индексирование полей*** (обычно ключевых). Индексирование полей, или лучше сказать создание индексных массивов, является типовой распространенной операцией практически во всех СУБД, поддерживающих и другие, не реляционные модели данных, и заключается в построении дополнительной упорядоченной информационной структуры для быстрого доступа к записям-кортежам.

Все *операции* над данными в реляционной модели (манипуляционная составляющая) можно разделить на две труппы — ***операции обновления*** таблиц-отношений и операции ***обработки таблиц-отношений.***

К *операциям обновления* относятся:

• операция ВКЛЮЧИТЬ —добавляет новый кортеж (строку-запись) в таблицу-отношение. Требует задания имени таблицы и обязательного значения ключей. Выполняется при условии уникальности значения ключа. Добавить новую строку-запись со значением ключа, которое уже есть в таблице, невозможно;

• операция УДАЛИТЬ — удаляет одну или группу кортежей (строк-записей). Требует задания имени таблицы, имени поля (группы полей) и параметров значений полей, кортежи с которыми должны быть удалены;

• операция ОБНОВИТЬ — изменяет значение не ключевых полей у одного или группы кортежей. Требует задания имени таблицы-отношения, имен полей и их значений для выбора кортежей и имен изменяемых полей.

Особенностью операций обработки в реляционной модели по сравнению с иерархической и сетевой моделью является то, что в качестве единичного элемента обработки выступает не запись (экземпляр объекта), а *таблица в целом,* т. е. множество кортежей (строк-записей). Поэтому все операции обновления являются операциями над множествами и их *результатом* является также множество, т. е. *новая таблица-отношение.* К *операциям обновления* относятся следующие операции:

• операция ОБЪЕДИНЕНИЕ — выполняется над двумя односхемными таблицами-отношениями. Результатом объединения является построенная по той же схеме таблица-отношение, содержащая все кортежи первой таблицы-отношения и все кортежи второй таблицы-отношения. При этом кортежи-дубликаты в итоговой таблице устраняются;

• операция ПЕРЕСЕЧЕНИЕ — выполняется также над двумя односхемными таблицами-отношениями. Результатом является таблица-отношение, построенная по той же схеме и содержащая только те кортежи первой таблицы-отношения, которые входят также в состав кортежей второй таблицы-отношения. Для примера рассмотрим таблицу «Сотрудники» со схемой (полями) «ФИО», «Год рождения», «Национальность» и таблицу «Вкладчики банка «НАДЕЖНЫЙ» с той же схемой. Результатом их пересечения будет новая таблица с той же схемой, содержащая кортежи только тех сотрудников, которые имеют вклады в банке «НАДЕЖНЫЙ», что иллюстрируется примером, приведенным на рис. 2.5;

• операция ВЫЧИТАНИЕ — выполняется также над двумя односхемными таблицами-отношениями. Результатом является таблица-отношение, построенная по той же схеме и содержащая только те кортежи первой таблицы-отношения, которых нет в составе кортежей второй таблицы-отношения. Результатом операции вычитания над таблицами в предыдущем примере будет новая таблица стой же схемой, содержащая кортежи только тех сотрудников, которые не имеют вкладов в банке «НАДЕЖНЫЙ»;

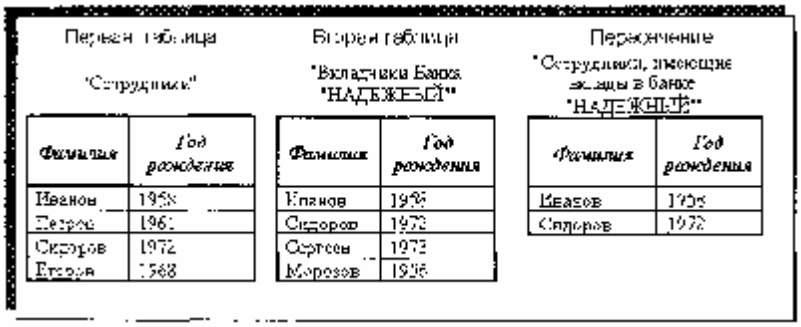



*Рис. 2.5. Пример операции ПЕРЕСЕЧЕНИЕ*

- операция ПРОИЗВЕДЕНИЕ (ДЕКАРТОВО) — выполняется над таблицами-отношениями с разными схемами. Результатом является таблица-отношение, схема которой включает все поля первой и все поля второй таблицы. Кортежи (строки-записи) результирующей таблицы образуются путем последовательного сцепления каждого кортежа первой таблицы-отношения к каждому кортежу второй таблицы-отношения. Количество кортежей результирующей таблицы соответственно равно произведению количества кортежей первой таблицы на количество кортежей второй таблицы. На рис. 2.6 иллюстрируется пример операции произведения;
- операция ВЫБОРКА (горизонтальное подмножество) — выполняется над одной таблицей-отношением. Результатом является таблица-отношение той же схемы, содержащая подмножество кортежей исходной таблицы-отношения, удовлетворяющих условию выборки;


*Рис. 2.6. Пример операции произведения*

- операция ПРОЕКЦИЯ (ВЕРТИКАЛЬНОЕ ПОДМНОЖЕСТВО) —также выполняется над одной таблицей-отношением. Результатом является новая таблица-отношение, схема которой содержит только некоторое подмножество полей исходной таблицы-отношения. Каждому кортежу исходной таблицы соответствует кортеж итоговой таблицы, образованный соответствующими значениями по полям, вошедшим в итоговую таблицу-отношение. При этом в итоговой таблице кортежи-дубликаты устраняются и поэтому мощность итоговой таблицы (количество кортежей) может быть равна или меньше исходной. На рис. 2.7 приведен пример операции проекции;


*Рис. 2.7. Пример операции проекции*

- операция СОЕДИНЕНИЕ — выполняется над таблицами-отношениями с разными схемами. В каждой таблице-отношении выделяется поле, по которому будет осуществляться соединение. При этом оба поля должны быть определены на одном и том же домене. Схема итоговой таблицы-отношения включает все поля первой таблицы и все поля второй таблицы (как в произведении). Кортежи итоговой таблицы-отношения образуются путем сцепления каждого кортежа из первой таблицы с теми кортежами второй таблицы, значения которых по полю сцепления одинаковы. На рис. 2.8 приведен пример операции соединения;


*Рис. 2.8. Пример операции соединения*

- операция ДЕЛЕНИЕ — выполняется над двумя таблицами-отношениями, первая из которых называется делимым, а вторая делителем. При этом схема таблицы-делителя должна состоять из подмножества полей таблицы делимого. Схема итоговой таблицы-отношения содержит только те поля таблицы-делимого, которых нет во второй таблице-делителе. Кортежи итоговой таблицы-отношения образуются на основе кортежей первой таблицы (делимого) по значениям полей, вошедших в итоговую таблицу при условии того, что если взять произведение (декартово) итоговой таблицы-отношения и второй таблицы-отношения (делителя), то образуются соответствующие кортежи первой таблицы (делимого). На рис. 2.9 приведен пример операции деления.

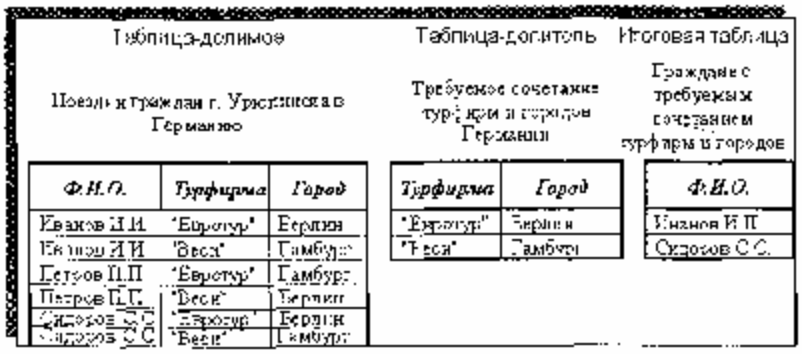



*Рис. 2.9. Пример операции деления*

В своем исходном виде манипуляционная составляющая реляционной модели не предусматривает операции над полями-атрибутами и схемами таблиц-отношений. Однако на практике применение рассмотренных выше операций манипулирования данными может приводить к временным нарушениям требований ограничения целостностей, которые преодолеваются специальными операциями переименования, удаления, добавления полей-атрибутов.

Реляционная модель организации данных сыграла трудно переоценимую роль в развитии программного обеспечения АИС. Именно реляционные СУБД были тем программным инструментарием, на основе которого происходила массовая информатизация малых и средних предприятий и организаций в 80-х годах. В начале 90-х годов реляционные СУБД стали фактическим стандартом для построения самых разнообразных информационных систем. Вместе с тем проявились и определенные ограничения реляционной модели, которые не позволяют адекватно описывать такие сложные предметные области, как конструирование, производственные технологические процессы и др. Поэтому в 90-х годах были предприняты попытки создания новых усовершенствованных моделей организации данных в виде *постреляционных СУБД* и *объектно-ориентированных СУБД*. К сожалению, до сей поры ни одна из попыток создания и описания новых моделей описания данных не стандартизована, и количество коммерческих СУБД, основанных на новых моделях данных, исчисляется единицами.

## 2.3. Внутренняя схема баз данных фактографических АИС

Изначально и по сей день программное обеспечение АИС (СУБД) в качестве места физического размещения данных ориентировано на внешнюю (дисковую) память. Как уже отмечалось, размещение данных во внешней памяти, точнее эффективность доступа к ним во внешней памяти, существенно влияет на эффективность обработки данных. В результате важным аспектом АИС является внутренняя схема базы данных, которую организует и поддерживает СУБД.

В общем плане внутренняя схема базы данных включает три основных компонента, представленные на рис. 2.10.

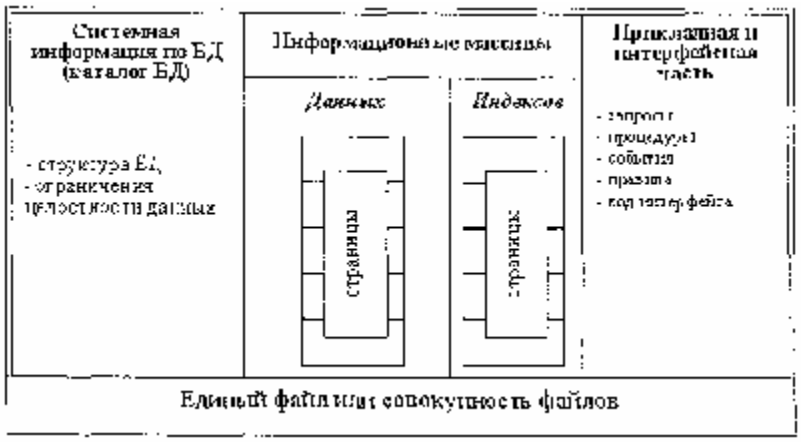



*Рис. 2.10. Состав внутренней схемы базы данных*

Центральным компонентом внутренней схемы являются ***информационные массивы,*** включающие собственно *данные* (информационных объектов логической схемы БД, т.е. в реляционных СУБД таблиц), и массивы *индексов,* являющихся специальными дополнительными конструкциями для ускорения доступа к данным основных информационных объектов. Информационные массивы в большинстве СУБД состоят из одной или нескольких так называемых *страниц,* каждая из которых содержит совокупность некоторых единичных элементов, называемых *физическими записями.* В результате, единичным элементом внутренней схемы баз данных АИС является физическая запись, в большинстве случаев совпадающая по смыслу с логической записью, т. е. в реляционных СУБД с табличной строкой.

Способы организации записей в страницах (расположение, добавления, корректировка, удаление) составляют ***физические структуры*** данных, которые образуют третий (низший) уровень представления информации в информационной системе (см. рис. 1.4).

Важным компонентом внутренней структуры является ***каталог БД,*** в котором размещается системная информация по логической структуре БД, включающая описание основных информационных объектов (имена, структура, параметры, связи) и ограничения целостности данных. Организация системной информации БД определяется особенностями конкретной СУБД, а сам каталог может входить непосредственно в файлы данных (область описателей данных) или составлять отдельный информационный массив.

Как уже отмечалось, в состав автоматизированного банка данных АИС помимо самой базы данных входит и *прикладной компонент,* образуемый совокупностью интерфейсных элементов представления, ввода и обработки данных, типовых запросов и процедур обработки данных, а также «событий» и «правил», отражающих правила и специфику предметной области АИС (так называемые «правила бизнеса»). Соответственно во внутренней схеме БД выделяется специальная область, в которой размещается информация по прикладному компоненту АИС.

Все три части внутренней структуры и их составные элементы (например, информационные массивы отдельных информационных объектов БД) могут размещаться в одном едином файле базы данных или в разных файлах. Во втором случае внутренняя схема БД определяется совокупностью и порядком расположения данных файлов.

### 2.3.1. Физические структуры данных

Страничная организация информационных массивов БД определяется общей спецификой доступа к данным больших объемов. Доступ к физическим записям во внешней памяти в большинстве СУБД осуществляется через считывание в оперативную память страниц файла данных, содержащих соответствующие записи. Непосредственная обработка записей производится в оперативной памяти, для чего СУБД образует и поддерживает, как уже отмечалось, специальные ***буферы,*** в которых временно размещаются страницы, содержащие обрабатываемые записи. После завершения обработки страница с соответствующими записями «выталкивается» из буфера и фиксируется в дисковом файле.

Аналогичным образом осуществляется размещение и доступ к индексным массивам. В итоге общий принцип организации доступа к данным во внешней памяти можно проиллюстрировать схемой, приведенной на рис. 2.11.

*Физические структуры* организации файлов данных подразделяются *на линейные* и *нелинейные.*

В ***линейных структурах*** в одну страницу файла базы данных объединяются записи-кортежи одной таблицы (информационного объекта), которые располагаются в *последовательном* (линейном) *порядке* друг за другом. Каких-либо ссылок, указателей на связи между записями не предусматривается.

Если не применяется специального порядка размещения записей в страницах (так называемая «расстановка» записей), то при *добавлении* записей в большинстве случаев в линейных структурах каждая новая запись помещается непосредственно за последней записью. Если страница файла данных заполняется, то для соответствующей таблицы выделяется дополнительная страница.

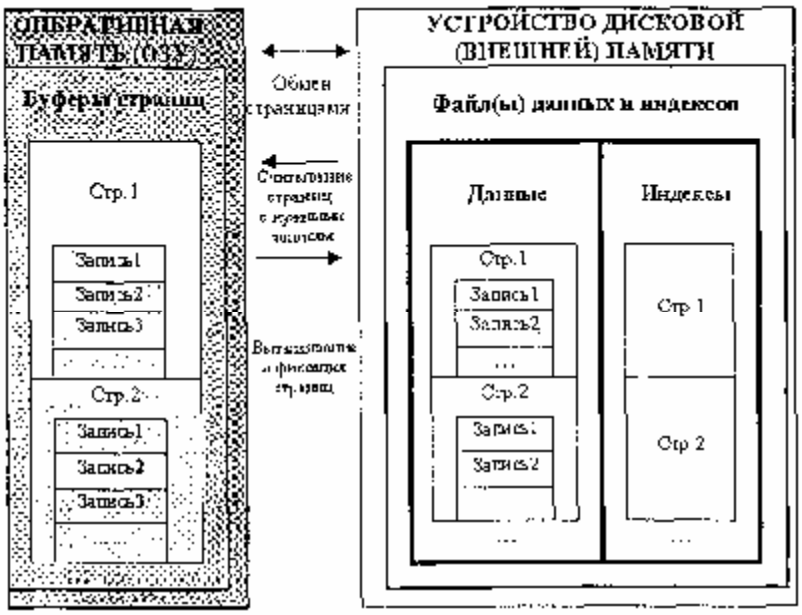



*Рис. 2.11. Общий принцип организации внутренней схемы базы данных*

*Удаление* записей в линейных структурах может производиться двумя способами. В *первом способе* при удалении записи сразу же осуществляется автоматическое перезаписывание на новых позициях всех строк-записей, лежащих за удаляемой, с целью уплотнения и ликвидации появляющихся пустых мест. Подобный подход обеспечивает максимальную эффективность использования дискового пространства, но вызывает существенные накладные расходы при любых операциях по удалению записей. Поэтому *другим,* достаточно распространенным *способом* является простое «вычеркивание» удаляемой записи без перезаписывания всех записей, лежащих за удаляемой, с соответствующим появлением пустых мест в страницах файла данных. Ведение базы данных в этом случае может быть организовано так, чтобы при превышении общего объема пустых мест в странице выше определенного значения (скажем, больше 30%) специальный компонент СУБД автоматически производил дефрагментацию страниц, устраняя пустые места по ранее удаленным записям. В некоторых СУБД запуск данной процедуры предоставляется непосредственно самому пользователю (администратору) для периодического уплотнения (сжатия) файла базы данных.

При *корректировках* записей могут возникать случаи, когда новое значение изменяемого поля корректируемой записи может потребовать больше (меньше) дискового пространства, ранее занимаемого под старое значение данного поля. Решение этой проблемы приводит к двум разновидностям линейных структур файлов баз данных.

*Первая разновидность* основана на подходе, позаимствованном из *структуры текстовых файлов.* Текстовый файл состоит из последовательно расположенных строк символов (набора байтов, определяющих номера символов строки в соответствии с кодовой таблицей). Строки имеют различную длину и отделяются друг от друга символом возврата каретки. Строка в данном случае является физической записью, а доступ к ней осуществляется по ее номеру k путем последовательного считывания (продвижения) k-1 предшествующих строк-записей (см. рис. 2.12).

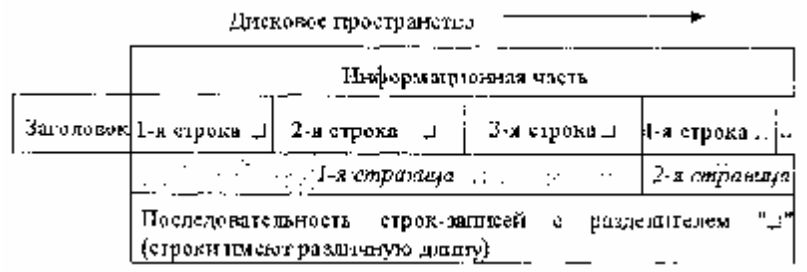



*Рис. 2.12. Линейная структура текстового файла*

Если при корректировке какого-либо поля требуется больше (меньше) дискового пространства, то файл расширяется (уплотняется) с автоматическим перезаписыванием на новых позициях всех строк, лежащих за корректируемой. Такой подход, так же, как и первый подход при удалениях записей, обеспечивает максимальную эффективность использования дискового пространства, но не дает возможности быстрого прямого доступа к нужной строке, так как местоположения записей постоянно меняются.

* Точнее говоря, перезаписываются все записи соответствующей страницы. Если она переполняется, то перезаписываются записи и следующей страницы, и т. д.

*Другим подходом к организации линейных структур* для решения проблем корректировки данных является выделение для каждой записи одинакового дискового пространства, исходя из максимально возможного заполнения строк по установленным типам полей.

Такой подход применяется в широко используемых для создания «настольных информационных систем» (системы «рабочего стола») СУБД куста dBASE (dBase, FoxPro, C*l*ipper), которые создают и оперируют базами данных в формате так называемых ***dbf-файлов.*** Структура dbf-файла состоит из трех частей* — заголовка, блока описания структуры базы и информационной части (см. рис. 2.13). В заголовке последовательно представлены поля, которые определяют тип файла базы данных (с memo-полями или без них), дату последнего изменения, номер последней записи, смещение, с которого начинается информационная часть (записи), размер каждой записи. Блок описания структуры размещается после заголовка до информационной части и состоит из последовательности элементов, каждый из которых описывает определенное поле логической структуры (схемы) базы данных. Структура описания поля содержит последовательное описание имени поля, типа поля (числовое, текстовое, дата и т. д.), длины поля и заканчивается специальным символом для отделения описания одного поля от другого. Информационная часть состоит из последовательности групп байтов одинаковой длины без специальных разделителей, каждая из которых собственно и выражает содержимое конкретной физической записи.

* Спенс Р. Clipper. Руководство по программированию. Версия 5.01 / Пер. с англ — Мн.: Тивали, 1994-480 с (с.428).

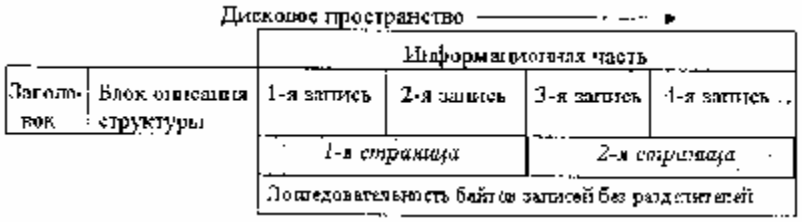



*Рис. 2.13. Линейная структура dbf-файла*

Такой способ организации данных обеспечивает *прямой доступ* к любой записи, так как ее положение (смещение) однозначно вычисляется по ее номеру и параметрам полей. Вместе с тем эффективность использования дискового пространства при таком подходе невысокая, так как в полях записей хранятся и пустые значения (т. е. физически занимают место). Тем не менее простота и эффективность доступа в таких линейных структурах файлов баз данных обусловили их популярность в тех случаях, когда объем данных невелик и вопросы эффективности использования дискового пространства не существенны.

В ***нелинейных структурах*** записи одного информационного объекта необязательно располагаются друг за другом на одной странице файла данных, но обязательно содержат *специальные указатели* на следующую запись объекта *(односвязные списки)* или на связанные записи других информационных объектов *(многосвязные списки, древовидные структуры)*. Соответственно физические записи в

нелинейных структурах включают помимо информационных полей одно или несколько *полей указателей,* где размещаются адреса связанных записей (см. рис. 2.14).

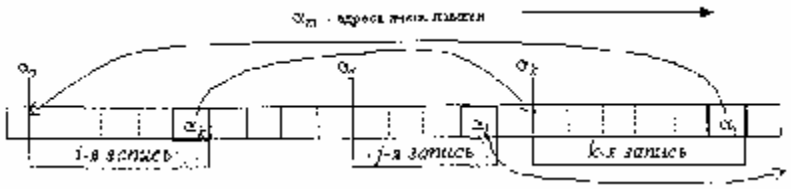



*Рис. 2.14. Нелинейная структура данных ни примере односвязного списка (поля указателей выделены жирными рамками)*

Непосредственная адресация* связанных записей обеспечивает в большинстве случаев *более эффективный,* чем в линейных структурах, *доступ к данным.* Однако расплатой за это являются *существенно большие* и *сложные* по сравнению с линейными структурами *затраты* и *процедуры преобразования* (перетряски) файла базы данных при любых *операциях добавления, удаления и корректировки записей,* так как помимо проблем определения мест размещения записей и появления пустых мест в файле данных добавляется проблема перенастройки указателей на связанные записи после изменения данных.

* Реализуется в виде прямой или косвенной адресации. При прямой адресации в указателях размещаются физические адреса начала связанных записей. При косвенной адресации в указателях находятся номера связанных записей, физические адреса которых отыскиваются по специальному справочнику, в который ставятся на учет физические адреса всех новых записей.

Образование страниц физических записей файлов данных осуществляется на основе той или иной стратегии минимизации расходов на доступ к записям и расходов на операции их добавления, удаления и корректировки, и существенным образом зависит от типа конкретной нелинейной структуры. Одной из таких стратегий является минимизация расходов на доступ к записям, связанным на логическом уровне отношением «Один-ко-многим», что обусловлено широкой распространенностью таких отношений в огромном количестве предметных областей АИС. Данная стратегия основывается на использовании *иерархических древовидных структур.*

Общая схема этой стратегии такова. Для записей информационного объекта на стороне «Один» образуется страница файла данных, в которую последовательно (как в линейных структурах) помещаются соответствующие записи. При появлении в базе данных связанных записей для каждой записи из страницы объекта на стороне «Один» образуются «подчиненные страницы» для размещения соответствующих связанных записей объекта на стороне «Многие». В свою очередь, подчиненные записи сами могут иметь свои подчиненные записи, для которых создаются соответствующие страницы, и т. д. (см. рис. 2.15, а.


*Рис. 2.15. а. Пример нелинейной древовидной иерархической структуры данных*

Если в процессе ведения базы данных какая-либо страница переполняется, то для нее образуется связанная страница-продолжение на основе техники односвязного списка.*

* В конце каждой страницы выделяется специальное поле-указатель на возможное продолжение страницы (т.н. цепной список) помимо того, что каждая запись страницы сама содержит поле-указатель на страницу подчиненных записей.

При таком подходе действительно обеспечивается эффективная обработка связанных записей, так как осуществляется не единичное, а сразу страничное их считывание в буферы оперативной памяти. Временные затраты на поиск и обработку записей в буферах оперативной памяти существенно меньше затрат на считывание данных из «медленной» внешней памяти и блоковое (страничное)

считывание обеспечивает существенный выигрыш в расходах на доступ и обработку связанных записей.

Однако во многих предметных областях АИС отношения между информационными объектами могут порождать ситуации наличия у определенных записей на стороне «Многие» сразу нескольких разных предков на стороне «Один» (см., например, рис. 2.3), что не вписывается в рамки подобных древовидных иерархических структур. Решение таких проблем обычно осуществляется через введение избыточности (дублирования) данных по связанным записям или другими способами.

Для формализованного описания нелинейных структур применяют аппарат теории графов, в рамках которого подобные иерархические древовидные структуры называются *деревьями.** На рис. 2.15, б приведено условное изображение *корневого дерева,* соответствующего иерархической нелинейной структуре данных на рис. 2.15, а, а также термины и понятия, связанные с ним. Вершины (узлы) дерева по отношению друг к другу могут быть *предками* или *потомками.* Предок может иметь несколько потомков, но каждый потомок имеет только одного предка. Вершины, имеющие общего предка, называются *братьями.* Вершины, имеющие потомков, называются *внутренними.* Внутренняя вершина, не имеющая предков, называется *корнем дерева.* Вершины, не имеющие потомков называются *листьями.* Все внутренние вершины корневого дерева упорядочиваются по *уровням* иерархии. Уровень узла определяется количеством предков до корня дерева. Количество уровней называется *высотой* дерева.

---

* *Деревом* называется связный неориентированный граф без циклов (см., например **Лекции по теории графов** / Емелнчев В. А., Мельников О. И., Сарванов В.И., Тышкевич Р.И.—М.: Наука, 1990, или Асанов М.О. Дискретная оптимизация: Учебное пособие. —Екатеринбург: УралНАУКА, 1998.

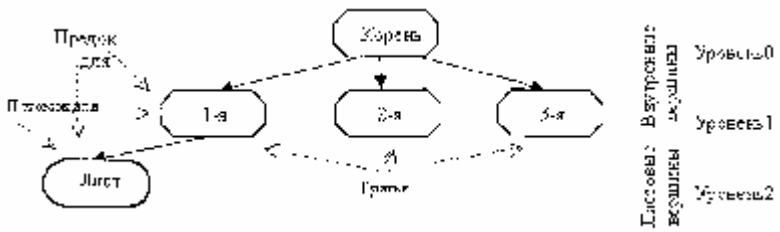



*Рис. 2.15, 6. Условное изображение средствами теории графов нелинейной древовидной иерархической структуры*

Еще одним важным параметром деревьев является максимально возможное количество потомков у одного предка. Данный параметр называется *степенью* дерева. Деревья степени больше двух называются *сильноветвистыми.* И наконец, важным в практическом плане для нелинейных структур данных является понятие *сбалансированности* дерева. Различают *сбалансированность по степени вершин,* когда каждая внутренняя вершина может иметь количество потомков, ограниченное определенной и одинаковой для всех вершин величиной *{арность* дерева) и *сбалансированность по высоте,* когда путь (количество вершин) от корня до любого листа дерева одинаков или различается не более чем на единицу.

Развитый математический аппарат теории графов позволил разработать для древовидных структур данных оптимальные по определенным критериям *операции обхода* (поиска нужной записи), *включения* (добавления новой записи) и *исключения* (удаления записи) с целью минимизации расходов как по доступу, так и по изменению данных. Хорошая алгоритмизируемость этих операций предопределила широкое использование нелинейных древовидных структур в схемах физической организации данных, обеспечиваемых СУБД.

### 2.3.2. Индексирование данных

Как уже отмечалось, стандартным приемом повышения эффективности доступа к записям в базах данных является создание индексных массивов по отдельным, обычно ключевым полям. Идея индексов основана на том факте, что если для повышения эффективности доступа к конкретному кортежу-записи использовать линейное упорядочение кортежей в таблице (скажем, по алфавиту для текстовых ключевых полей или по возрастанию для числовых ключевых полей), то накладные расходы по «перетряске» всей таблицы после добавления/удаления строк-кортежей превысят выигрыш во времени доступа. Структура индексов (индексных массивов) строится так, чтобы на

основе некоторого критерия можно было бы быстро находить по значению индексируемого поля указатель на нужную запись (строку) таблицы и получать к ней доступ. При этом совокупность записей-кортежей базовой таблицы не обязательно упорядочивать, а при их изменении необходимо изменить только лишь индексный массив.

Как и для информационных массивов самих данных (таблиц), так и для индексных массивов применяются линейные и нелинейные структуры. В качестве ***линейных структур индексов*** в большинстве случаев выступают ***инвертированные списки.*** Инвертированный список строится по схеме таблицы с двумя колонками — «Значение индексируемого поля» и «Номера строк»* (см. рис. 2.16). Инвертированные списки чаще всего применяются для индексации полей, значения которых в разных строках-записях могут повторяться, к примеру, поле «Год рождения» таблицы «Сотрудники» (в реляционных базах данных такие поля не могут быть ключевыми).

* На практике не номера строк базовой таблицы, а номера страниц файла БД, где находится соответствующая строка.

| ***Значение индексируемого поля ("Год рождения")*** | ***Номера, строк*** |
|---|---|
| 1958 | 3 |
| 1959 | 5, 17, 123, 256 |
| 1960 | 31, 32, 77 |
| 1961 | 11, 45, 58, 167, 231 |
| 1962 | 7, 8, 9, 10, 234, 235, 236 |

*Рис. 2.16. Пример инвертированного списка*

Строки инвертированного списка упорядочиваются по значению индексируемого поля. Для *доступа* к нужной строке исходной таблицы сначала в упорядоченном инвертированном списке отыскивается строка с требуемым значением поля,* затем считывается номер соответствующей строки (строк) в исходной таблице и далее по нему уже осуществляется непосредственный доступ к искомой строке базовой таблицы.

* Способы поиска в линейно упорядоченных структурах см., например, в работе: ***Вирт Н.***. Алгоритмы и структуры данных: Пер. с англ.—М.: Мир, 1989.

При *добавлении* новой строки в базовую таблицу ее значение по индексируемому полю ищется в ранее составленном индексе. Если соответствующая строка инвертированного списка отыскивается (т.е. подобное значение индексируемого поля среди строк таблицы уже встречалось и было поставлено на учет), то в ячейку второго столбца соответствующей строки индекса дописывается номер страницы, куда была помещена соответствующая строка базовой таблицы. Если такого значения в индексе нет, то создается новая строка индекса и осуществляется переупорядочение нового состояния индексного массива.

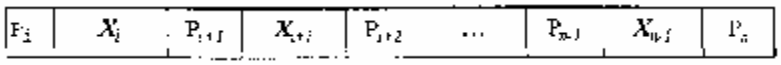
При *удалении* строки из базовой таблицы также осуществляется поиск соответствующей строки в индексном массиве и осуществляется вычеркивание в индексе соответствующего номера отсылаемой строки базовой таблицы. Если при этом других строк в базовой таблице с таким же значением индексируемого поля не осталось (соответствующая ячейка индекса стала пустой), то удаляется и вся данная строка индекса с последующим переупорядочением всего индексного массива. При этом за счет того, что индекс в виде инвертированного списка содержит лишь один столбец значений, затраты на «перетряску» при добавлении или удалении записей существенно меньше по сравнению с тем, если бы переупорядочение происходило непосредственно в самой базовой таблице.*

* Кроме того, строки базовой таблицы можно упорядочивать только лишь по какому-либо одному полю, а индексные массивы можно создавать сразу по нескольким полям.

Индексы в виде инвертированных списков являются особенно эффективными в том случае, когда значения индексируемого поля часто повторяются, образуя равномерные по мощности группы. В этом случае количество ситуаций, при которых требуется добавление или удаление строк индекса, невелико, и затраты на переупорядочение индекса при изменениях данных в базовой таблице незначительны. В результате выигрыш по затратам на доступ существенно превышает накладные расходы по переупорядочению индекса в процессе ведения базы данных.

*Нелинейные структуры индексов* применяются для создания индексных массивов ключевых полей или тех полей, значения по которым не повторяются. При организации индексов в таких случаях чаще всего используются уже упоминавшиеся древовидные иерархические структуры в виде ***Б-деревьев***. Б-дерево представляет собой корневое сбалансированное сильно ветвистое дерево.

Каждая внутренняя вершина, если она полностью заполнена, содержит информацию о n–1 различных последовательно возрастающих значениях индексируемого поля по следующей схеме, приведенной на рис. 2.17.

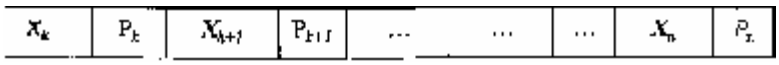


где: $X_i$ – *i-е* значение индексируемого поля, при этом $X_i$**J**$X_{i+1}$ – указатель на вершину, содержащую значения индексируемого поля, меньшие или равные $X_i$.

*Рис. 2.17. Структура внутренней вершины Б-дерева*

Число n называется *порядком* Б-дерева.

Листовая вершина, в отличие от внутренней, содержит информацию о нахождении страницы в файле базы данных с записями-кортежами, имеющими соответствующие значения индексируемого поля. Схема листовой страницы представлена на рис.2.18.



где: $\mathbf{P}_k$ – указатель на страницу файла данных, содержащую строку (строки) со значением индексируемого поля, равным $X_k$.

*Рис. 2.18. Структура листовой вершины Б-дерева*

*Сбалансированность* Б-дерева означает одинаковое количество потомков до листовой вершины по любым разветвлениям от корневой вершины. В качестве примера на рис. 2.19 приведено Б-дерево 3-го порядка уровня 2, построенное для поля «Год рождения» таблицы «Сотрудники».

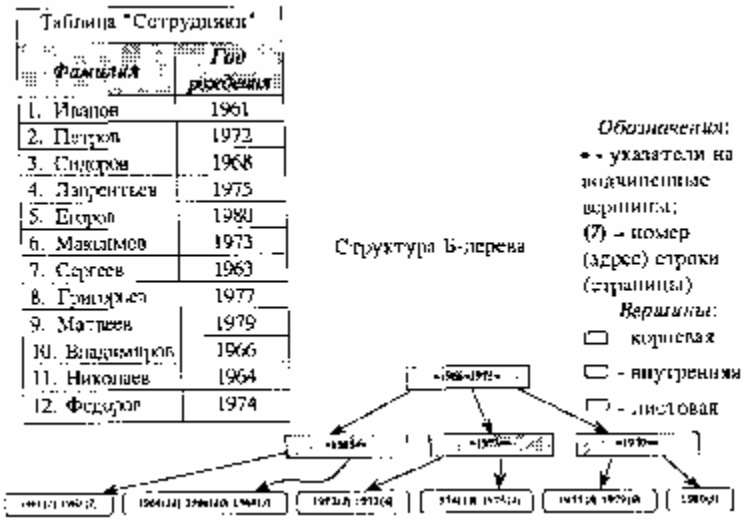



*Рис. 2.19. Пример Б-дерева 3-го порядки уровня 2*

Как правило, порядок Б-дерева выбирается таким, чтобы информация одной полностью заполненной вершины соответствовала странице файла данных. В силу того что объем одной страницы файлов данных существенно больше размера полей, то в большинстве случаев в одну полностью заполненную страницу вместе с указателями помещается большое количество значений индексируемого поля, исчисляемое сотнями и даже тысячами значений (n >> 100...1000). Это обстоятельство обусловливает сравнительно небольшой уровень Б-деревьев на практике (порядка 2...4) даже в тех случаях, когда количество строк в базовой таблице исчисляется десятками тысяч. В результате доступ к нужной записи по определенному значению индексируемого поля через Б-дерево осуществляется за небольшое количество страничных обменов между внутренней и внешней памятью по следующему алгоритму:

— в оперативную память из внешней считывается страница с корневой вершиной и последовательно просматривается до первого значения, превышающего значение индексируемого поля нужной записи. При этом определяется ссылка (номер) страницы-потомка (внутренней или листовой);

— в оперативную память считывается страница-потомок. Если она внутренняя, то ее обработка производится аналогично корневой. Если страница-потомок является листовой, то она

последовательно просматривается до нахождения нужного значения индексируемого поля. При этом определяется номер страницы файла данных, которая содержит нужную запись;

— если при просмотре листовой страницы нужное значение индексируемого поля не находится, то поиск считается отрицательным, т. е. принимается решение о том, что строки-кортежа с требуемым значением индексируемого поля в таблице-отношении (файле данных) нет.

Анализ данного алгоритма показывает, что при поиске нужного значения индексируемого поля по индексу в виде Б-дерева требуется такое количество страничных обменов между внешней и внутренней памятью, которое равно уровню Б-дерева плюс единица. При достаточно большом значении n, а это, как уже отмечалось, наиболее распространенная ситуация, уровень m Б-дерева невелик (m >> 2..3), и, следовательно, количество страничных индексных обменов с памятью также невелико. При этом сбалансированность Б-дерева обусловливает одинаковые затраты по доступу к любой записи с любым значением индексируемого поля. *Сбалансированность* Б-деревьев обеспечивается легко алгоритмизируемым правилом их построения (роста) при включении нового значения индексируемого поля.

*Включение* нового значения индексируемого поля осуществляется следующим образом.

• Производится поиск требуемого значения в существующем Б-дереве. При его отсутствии поиск, естественно, заканчивается отрицательным результатом, но в оперативной памяти остается листовая страница, куда, следовательно, и необходимо поместить новое значение индексируемого поля.

• Если листовая страница не заполнена полностью, в нее записывается новое значение индексируемого поля и указатель на страницу файла данных, содержащую соответствующую запись (строку таблицы).

• Если листовая страница уже заполнена, то она делится «пополам», и тем самым порождается новая листовая страница. Среднее значение исходной листовой страницы записывается в вершину-предок на соответствующее место.* При этом после вставленного значения образуется указатель-ссылка на новую листовую страницу, образованную остатком от ополовиненной исходной листовой страницы.

* Так, чтобы в странице обеспечивался последовательный порядок расположения -значений индексируемого поля по схеме, приведенной на рис. 2.17.

• Если при занесении среднего значения переполненной листовой страницы в страницу-предок происходит переполнение самой страницы-предка, то она аналогично делится пополам, «посылая» свое среднее значение своему предку. Если при этом происходит переполнение корневой страницы, то она также делится на две новые внутренние страницы, а ее среднее значение образует новую корневую страницу, повышая уровень дерева на единицу и т. д. (говорят, что Б-дерево «растет» вверх).

*Удаление* определенного значения индексируемого поля производится следующим образом:

• Производится поиск требуемого значения индексируемого поля и в результате в оперативную память считывается нужная листовая страница.

• Из нее удаляется соответствующее значение индексируемого поля и указатель-адрес соответствующей страницы файла данных.

• Если после удаления размер занятого пространства памяти текущей листовой страницы в сумме с размером занятого пространства левого (правого) брата больше, чем размер памяти одной страницы, то процесс заканчивается.

• В противном случае текущая листовая страница объединяется с левым (правым) братом. Тем самым одна из листовых страниц (левая или правая) опустошается и уничтожается. В вершине-предке удаляется значение индексируемого поля, указатель перед которым или после которого ссылался на опустошенную листовую страницу. При этом может, в свою очередь, возникнуть необходимость слияния и опустошения уже внутренних страниц, которое осуществляется аналогичным образом.

Описанный алгоритм удаления значений индексируемого поля соответствует одной из наиболее широко применяемых разновидностей Б-деревьев—***Б*-деревьев,*** которые позволяют использовать в среднем до 2/3 пространства всех страниц (вершин) дерева.

Структура Б-деревьев является эффективным способом индексации больших массивов данных и широко применяется в современных СУБД.

### 2.3.3. Расстановка (хеширование) записей

В некотором смысле альтернативным подходом к организации физических структур данных является *расстановка записей.* Как и при использовании линейных и нелинейных структур, основной задачей расстановки записей является минимизация расходов на доступ и изменения данных во внутренней и внешней памяти.

Идея расстановки, известной в англоязычной литературе также под термином ***хеширование,**** заключается в том, чтобы при выделении под размещение данных определенного участка памяти так организовать порядок расположения записей в нем, чтобы место для новых записей и поиск старых записей можно было осуществлять на основе некоторого преобразования их ключевых полей.** При образовании *(добавлении)* новой записи к значению ее ключевого поля применяется специальная *хеш-функция* (или иначе хеш-свертка), которая ставит в соответствие значению ключевого поля, и, следовательно, всей записи, некоторое числовое значение, обычно являющееся номером (адресом) местоположения, куда в итоге и помещается соответствующая запись (см. рис. 2.20).

* От англ. *to hash* – нарезать, крошить, делить месиво, т.е. равномерно перемалывать ключи нолей в адреса (номера) записей.

** Отсюда еще одно название данного подхода—«преобразование ключей», впервые введенное в русской литературе еще в 1956 г. будущим академиком А. П. Ершовым.

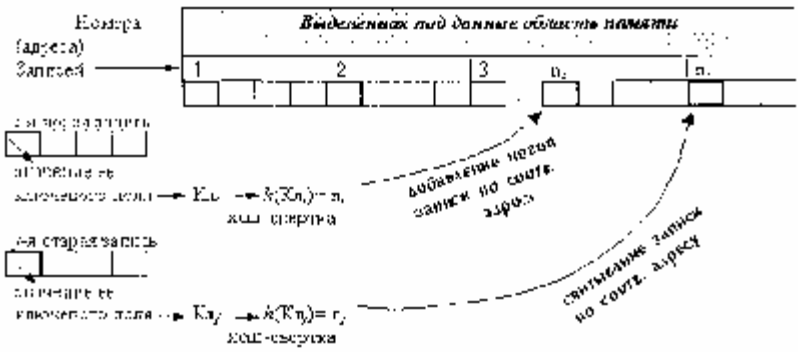



*Рис. 2.20. Принцип расстановки записей по значению ключей*

При *доступе* (поиске) нужной записи над значением ее ключевого поля также осуществляется хеш-свертка, что сразу же даст возможность определить местоположение искомой записи и получить к ней доступ.

Таким образом, в идеале хеширование обеспечивает доступ к нужным записям за одно (!!!) обращение к области размещения данных.

Функция преобразования $h(\text{Кл}_j)$ выбирается на основе двух требований: 1) ее результат для возможного диапазона значений ключевого поля должен находиться в пределах диапазона адресов (номеров) области памяти, выделяемой подданные, и 2) значения функции в пределах выделенного диапазона адресов должны быть равномерными. На практике наиболее широкое распространение нашли хеш-функции, основанные на операциях деления по модулю:

$$h(\text{Кл}_j)=(\text{Кл}_j \bmod \text{M}) + 1,$$

где ($\text{Кл}_j$ mod M) означает операцию деления по модулю M,* а число M выбирается исходя из необходимости попадания значений хеш-функции в требуемый диапазон.

* Значением операции деления по модулю M является остаток от обычного деления числа на M, например результат деления 213 по модулю 20 равен 13.

Если в качестве M выбрать число, являющееся степенью двух, то подобная хеш-функция эффективно вычисляется процедурами выделения нескольких двоичных цифр.

Для текстовых ключевых полей обычно применяют подобные подходы, основанные на использовании операции сложения по модулю значений кодов первых нескольких символов ключа.

Основной проблемой хеширования с прямой адресацией записей является возможность появления *одинаковых значений* хеш-сверток при разных значениях полей (так называемые синонимы). Такие ситуации называются *коллизиями,* так как требуют определенного порядка, правила их разрешения.

Применяются два подхода к *разрешению* коллизий. *Первый подход* основывается на использовании технологии *цепных списков* и заключается в присоединении к записям, по которым возникают коллизии, специальных дополнительных указателей, по которым размещаются новые записи, конфликтующие по месту расположения с ранее введенными. Как правило, для таких записей

отводится специально выделенный участок памяти, называемый *областью переполнения.* В алгоритм доступа добавляются операции перехода по цепному списку для нахождения искомой записи. Вместе с тем при большом количестве переполнении теряется главное достоинство хеширования — доступ к записи за одно обращение к памяти.

*Второй подход* к разрешению коллизий основывается на использовании *дополнительного преобразования ключей* по схеме:

$$\mathrm{n}_i^{(доп)} = h(\mathrm{Кл}_i) + \mathrm{g}(\mathrm{Кл}_i) = \mathrm{n}_i^{(0)} + \mathrm{g}(\mathrm{Кл}_i)$$

где $\mathrm{n}_i^{(0)}$ – исходное конфликтующее значение адреса записи; $\mathrm{g}(\mathrm{Кл}_i)$ – дополнительное преобразование ключа; $\mathrm{n}_i^{(доп)}$ – дополнительное значение адреса.

При этом могут встречаться ситуации, когда и такое дополнительное преобразование приводит к коллизии, т.е. новое значение $\mathrm{n}_i^{(доп)}$ также оказывается уже занятым. В таких случаях дополнительное преобразование $\mathrm{g}(\mathrm{Кл}_i)$ чаще всего организуют на основе рекуррентной процедуры по следующей схеме:

$$\mathrm{n}_i^{(k)} = \mathrm{n}_i^{(k-1)} + f(k)$$

где *f(k)* – некоторая функция над номером итерации (пробы).

В зависимости от вида функции *f(k)* такие подходы называют *линейными* или *квадратичными пробами.*

Основным недостатком второго подхода к разрешению коллизий является во многих случаях существенно больший объем вычислений, а также появление для линейных проб эффектов группирования номеров (адресов) текстовых ключевых полей, т. е. неравномерность номеров, если значения ключей отличаются друг от друга всего несколькими символами.

Вместе с тем *коллизии* (одинаковые значения хеш-сверток ключей) при использовании *страничной организации структуры файлов баз данных* во внешней памяти получили свое положительное и логичное *разрешение* и применение. Значением хеш-свертки ключа в этом случае является номер страницы, куда помещается или где находится соответствующая запись (см. рис. 2.21).

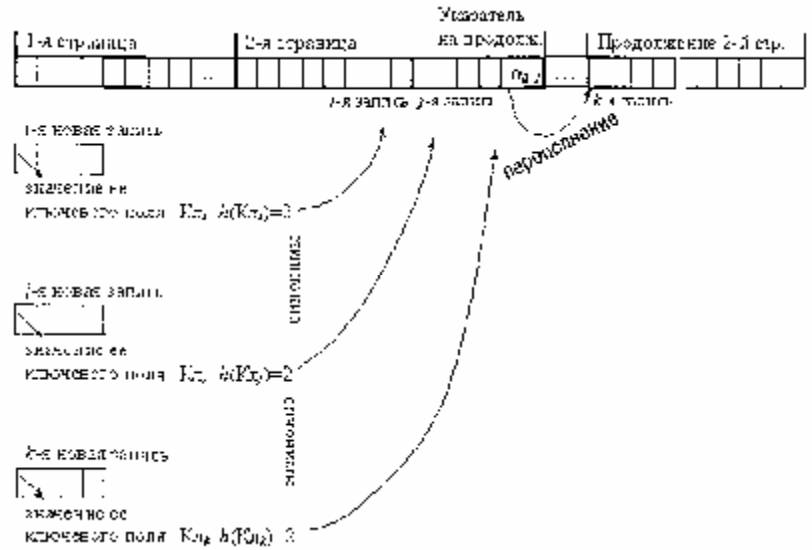



*Рис. 2.21. Принцип хеширования записей но страницам файла данных*

Для *доступа* к нужной записи сначала производится хэш-свертка значения его ключевого поля и тем самым определяется номер страницы файла базы данных, содержащей нужную запись. Страница файла данных из дисковой пересылается в оперативную память, где путем последовательного перебора отыскивается требуемая запись. Таким образом, при отсутствии переполнении доступ к записям осуществляется за одно страничное считывание данных из внешней памяти.

С учетом того что количество записей, помещающихся в одной странице файла данных, в большинстве случаев достаточно велико, случаи переполнений на практике не так часты. Вместе с тем, при большом количестве записей в таблицах файла данных, количество переполнении может существенно возрасти и так же, как и в классическом подходе, теряется главное достоинство хэширования — доступ к записям за одну пересылку страницы в оперативную память. Для смягчения подобных ситуаций могут применяться комбинации хэширования и структуры Б-деревьев для размещения записей по страницам в случаях переполнения.

Внутренняя схема базы данных обычно скрыта от пользователей информационной системы, за исключением возможности установления и использования индексации полей. Вместе с тем, особенности физической структуры файлов данных и индексных массивов, принципы организации и использования дискового пространства и внутренней памяти, реализуемые конкретной СУБД, должны учитываться проектировщиками банков данных, так как эти «прозрачные» для пользователей-абонентов особенности СУБД критично влияют на эффективность обработки данных в информационной системе.

**Вопросы и упражнения**

1. Перечислите основные функции, реализуемые СУБД, и охарактеризуйте их с точки зрения системного или прикладного характера решаемых задач.
2. Объясните соотношение понятии «операции», «транзакции» и «работа (действия) пользователя» в базе данных.
3. Перечислите (функциональные компоненты СУБД и охарактеризуйте системный или прикладной характер решаемых ими задач.
4. Дайте определение «модели организации данных» и перечислите ее составляющие. Охарактеризуйте особенности сетевой модели данных по отношению к иерархической модели.
5. Перечислите основные понятия структурной составляющей реляционной модели данных. Каким образом строятся связи между таблицами-отношениями, какие типы связей и почему обеспечиваются при этом?
6. В чем заключается н каким образом обеспечивается целостность в реляционной модели данных?
7. Какое основное различие между операциями обновления данных и операциями обработки таблиц-отношений в реляционной модели?
8. Дайте определение операции ОБЪЕДИНЕНИЯ таблиц-отношений. Каково наименьшее и наибольшее количество строк может быть в результате объединения таблиц?
9. Дайте определение операции ПЕРЕСЕЧЕНИЯ таблиц-отношений. Каково наименьшее и наибольшее количество строк может быть в результате пересечения таблиц?
10. Дайте определение операции ВЫЧИТАНИЯ таблиц-отношений. Каково наименьшее н наибольшее количество строк может быть в результате вычитания таблиц?
11. Какие нарушения целостности данных могут происходить в результате операции ПРОЕКЦИЯ (ВЕРТИКАЛЬНОЕ ПОДМНОЖЕСТВО)?
12. Дайте определение операции СОЕДИНЕНИЯ таблиц-отношений. Каково наименьшее n наибольшее количество строк может быть в результате соединения таблиц?
13. Что является единичным элементом в физической структуре данных? В какие структуры более высокого порядка объединяются единичные элементы данных?
14. Дайте сравнительную характеристику преимуществ и недостатков разновидностей линейных структур физической организации данных.
15. Охарактеризуйте общий принцип нелинейных структур физической организации данных и перечислите их основные разновидности.
16. Выделите и поясните главную идею использования древовидных иерархических структур физической организации данных. В терминологии теории графов дайте определения основных понятий «деревьев».
17. Постройте средствами теории графов структуру физической организации данных, которая могла бы соответствовать на логическом уровне следующим трем таблицам:

**Здания**

| №№ | Улица | № дома |
|---|---|---|
| 1 | Бикулова | 5 |
| 2 | Рахова | 113/б |
| 3 | Попова | 47 |
| 4 | Рязина | 53 |

**Квартиры**

| №№ | №№ здания | № кв. | Площадь | Кол-во комнат |
|---|---|---|---|---|
| 1 | 1 | 5 | 115м² | 3 |
| 2 | 1 | 7 | 80м² | 2 |
| 3 | 3 | 11-я | 33м² | 1 |
| 4 | 1 | 73 | 150м² | 4 |
| 5 | 2 | 42 | 62м² | 3 |
| 6 | 3 | 6 | 29м² | 2 |
| 7 | 1 | 87/1 | 31м² | 2 |

**Жильцы**

| №№ | № кв. | ФИО | Отв. |
|---|---|---|---|
| 1 | 1 | Иванов | |
| 2 | 3 | Петров | Да |
| 3 | 3 | Сидоров | Да |
| 4 | 1 | Сергеев | Да |
| 5 | 4 | Егоров | |
| 6 | 4 | Андреев | Да |
| 7 | 4 | Гаврилов | |
| 8 | 5 | Николаев | |
| 9 | 5 | Васильев | Да |
| 10 | 6 | Власов | Да |
| 11 | 7 | Федоров | Да |

Определите тип и параметры построенной структуры физической организации данных. Укажите, сколько страниц файла базы данных понадобится для данной структуры.

18. Обоснуйте выбор типа индексов для полей таблицы «Расписание занятий» со следующей схемой: №№, Дата, Время (1-я пара, 2-я пара, 3-пара, 4-я пара), Аудитория (30 аудиторий), Вид (Лекция, Семинар, Пр. занятие. Лабораторная работа. Зачет, Экзамен), Дисциплина (100 уч. дисциплин). Преподаватель (80 преподавателей), Учебная группа (15 уч.групп).
19. Произведите преобразование индекса в виде Б-дерева, представленного на рис. 2.18,при:

а) добавлении 13-й записи «Данилов, 1976г. р.»;

б) добавлении 14-й записи «Никаноров, 1967 г. р.»;

в) удалении 9-й записи «Матвеев, 1979 г. р.».

20. Проиллюстрируйте (по шагам добавления записей) процесс построения индекса в виде Б-дерева 3-го порядка по полю «Таб_№» таблицы:

| № строки | Таб_№ | «Фамилия» |
|---|---|---|
| 1 | 343 | Иванов |
| 2 | 97 | Петров |
| 3 | 113 | Сидоров |
| 4 | 705 | Лаврентьев |
| 5 | 453 | Егоров |
| 6 | 605 | Максимов |
| 7 | 177 | Сергеев |
| 8 | 806 | Григорьев |
| 9 | 337 | Матвеев |
| 10 | 620 | Владимиров |
| 11 | 359 | Николаев |
| 12 | 803 | Федоров |

21. В чем преимущества и недостатки с точки зрения эффективности операций доступа к данным и преобразования данных между индексированием и хешированием?3. Основы создания автоматизированных информационных систем

# 3. ОСНОВЫ СОЗДАНИЯ АВТОМАТИЗИРОВАННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

## 3.1. Общие положения по созданию автоматизированных систем

Создание автоматизированных информационных систем в нашей стране регламентируется «Комплексом стандартов и руководящих документов на автоматизированные системы» (ГОСТ 34.201-89, ГОСТ 34.602-89, РД 50-682, РД 50-680-88, ГОСТ 34.601-90, ГОСТ 34.401-90, РД 50-34.698-90, ГОСТ 34.003-90, Р 50-34.119-90). В частности, ГОСТ 34.601-90 определяет следующие ***стадии и этапы создания*** АС (см. табл. 3.1).

*Таблица 3.1*

| *Стадии* | *Этапы работ* |
|---|---|
| 1. Формирование требований к АС | 1.1. Обследование объекта и обоснование необходимости создания АС<br>1.2. Формирование требований пользователя к АС<br>1.3. Оформление отчета о выполненной работе и заявки на разработку АС (тактико-технического задания) |
| 2. Разработка концепции АС | 2.1. Изучение объекта<br>2.2. Проведение необходимых научно-исследовательских работ<br>2.3. Разработка вариантов концепции АС и выбор варианта концепции АС, удовлетворяющего требованиям пользователя<br>2.4. Оформление отчета о выполненной работе |
| 3. Техническое задание | 3.1. Разработка и утверждение технического задания на создание АС |
| 4. Эскизный проект | 4.1. Разработка предварительных проектных решений по системе и ее частям<br>4.2. Разработка документации на АС и ее части |
| 5. Технический проект | 5.1. Разработка проектных решений по системе и ее частям<br>5.2. Разработка документации на АС и ее части<br>5.3. Разработка и оформление документации на поставку изделий для комплектования АС и (или) технических требований (технических заданий) на их разработку<br>5.4. Разработка заданий на проектирование в смежных частях проекта объекта автоматизации |
| 6. Рабочая документация | 6.1. Разработка рабочей документации на систему и ее части<br>6.2. Разработка или адаптация программ |
| 7. Ввод в действие | 7.1. Подготовка объекта автоматизации к вводу АС в действие<br>7.2. Подготовка персонала<br>7.3. Комплектация АС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями)<br>7.4. Строительно-монтажные работы<br>7.5. Пусконаладочные работы<br>7.6. Проведение предварительных испытаний<br>7.7. Проведение опытной эксплуатации<br>7.8. Проведение приемочных испытаний |
| 8. Сопровождение АС | 8.1. Выполнение работ в соответствии с гарантийными обязательствами<br>8.2. Послегарантийное обслуживание |

Одним из центральных элементов всего процесса создания АС является разработка ***технического задания,*** структура которого согласно ГОСТ 34.602-89 содержит следующие разделы:

1) общие сведения;
2) назначение и цели создания (развития) системы;
3) характеристика объектов автоматизации;
4) требования к системе;
5) состав и содержание работ по созданию системы;
6) порядок контроля и приемки системы;
7) требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие;

8) требования к документированию;
9) источники разработки.

Суть технического задания как основного документа в процессе создания АС заключается в проработке, выборе и утверждении основных технических, организационных, программных, информационно-логических и лингвистических решений, которые устанавливаются в разделе «Требования к системе». Данный раздел, в свою очередь, состоит из трех подразделов:

1) требования к системе в целом;
2) требования к функциям (задачам), выполняемым системой;
3) требования к видам обеспечения.

*Требования к системе в целом* отражают концептуальные параметры и характеристики создаваемой системы, среди которых указываются требования к структуре и функционированию системы, к надежности и безопасности, к численности и квалификации персонала и т. д.

*Требования к функциям (задачам)* содержат перечень функций, задач или их комплексов; временной регламент каждой функции, задачи или комплекса задач; требования к качеству реализации каждой функции; к форме представления выходной информации; характеристики необходимой точности и времени выполнения, требования одновременности выполнения группы функций; достоверности выдачи результатов.

*Требования по видам обеспечения* в зависимости от вида системы содержат требования по математическому, информационному, лингвистическому, программному, техническому, метрологическому, организационному, методическому и другим видам обеспечения системы.

Для большинства разновидностей АС, в частности для всевозможных видов АИС, особое значение имеют решения и ***требования по информационному обеспечению.*** В данном подразделе, в частности, определяются требования:

• к составу, структуре и способам организации данных в системе *(информационно-логическая схема)',*
• к информационному обмену между компонентами системы;
• к информационной совместимости со смежными системами;
• по использованию общероссийских и других классификаторов, унифицированных документов;
• по применению систем управления базами данных;
• к структуре процесса сбора, обработки, передачи данных в системе и представлению данных;
• к защите данных от разрушений при авариях и сбоях в электропитании системы;
• к контролю, хранению, обновлению и восстановлению данных;
• к процедуре придания юридической силы документам, продуцируемым техническими средствами АС.

На основе установленных в техническом задании основных требований и технических решений на последующих этапах конкретизируются и непосредственно разрабатываются компоненты и элементы системы.

В частности, на этапе 4.1 *«Разработки предварительных проектных решений по системе и ее частям»* определяются:

• функции АС;
• функции подсистем;
• концепция информационной базы и ее укрупненная структура;
• функции системы управления базой данных;
• состав вычислительной системы;
• функции и параметры основных программных средств.

На этапе 5.1 *«Разработка проектных решений но системе и ее частям»* осуществляется разработка общих решений по системе и ее частям:

• по функционально-алгоритмической структуре системы;
• по функциям персонала и организационной структуре;
• по структуре технических средств;
• по алгоритмам решения задач и применяемым языкам;
• по организации и ведению информационной базы ***(структура базы данных);***
• по системе классификации и кодирования информации ***(словарно-классификационная база);***
• по программному обеспечению.

Разработка и документация программного обеспечения в процессе создания или комплектования автоматизированных систем (п. 6.2) регламентируются комплексом стандартов, объединенных в группу «Единая система программной документации (ЕСПД)».

Таким образом, создание автоматизированных информационных систем представляет собой сложный многоэтапный процесс, требующий привлечения различных категорий специалистов — программистов, инженеров, управленческих и других работников.

## 3.2. Проектирование банков данных фактографических АИС

Одной из наиболее трудоемких и сложных задач при создании АИС является проектирование банка данных как основы подсистемы представления и обработки информации. Логическая и физическая структуры банка данных отражают представление разработчиками и пользователями информационной системы той предметной области, сведения о которой предполагается отражать и использовать в АИС.

Проектирование банков данных фактографических информационных систем осуществляется на основе формализации структуры и процессов предметной области АИС, и, в соответствии с уровнями представления информации в АИС (см. рис. 1.3), включает ***концептуальное*** (пп. 3.1 и 4.1) и ***схемно-структурное*** проектирование (п. 5.1).

В организационном плане в группе разработчиков банка данных выделяют специалистов по формализации предметной области, специалистов по программному обеспечению СУБД, а также технических дизайнеров и специалистов по эргономике. *Специалисты по формализации предметной области* (их еще называют формализаторами или постановщиками задач), как правило, возглавляют весь проект создания АИС и обеспечивают (функции взаимодействия с заказчиком. К данной категории специалистов предъявляются наиболее сложные профессиональные требования. С одной стороны, такие работники должны быть специалистами в севере программного обеспечения АИС (операционные системы, СУБД и т. д.), а с другой стороны, они должны хорошо представлять (или освоить) конкретную предметную область АИС, т. е. быть (временно стать) бухгалтерами, экономистами, делопроизводителями и т.п. *Специалисты по программному обеспечению СУБД* относятся к категории профессиональных программистов, определяют выбор СУБД и обеспечивают построение ее средствами автоматизированного банка данных по разработанной постановщиком задачи (формализатором) концептуальной схеме. *Технические дизайнеры* и *специалисты по эргономике* обеспечивают эстетичную и эргономичную сторону интерфейса с пользователем в АИС при вводе, обработке и поиске данных.

### 3.2.1. Концептуальное проектирование

Концептуальное проектирование банков данных АИС является в значительной степени *эвристическим* процессом, и адекватность построенной в его рамках инфологической схемы предметной области проверяется в большинстве случаев эмпирически по анализу и проверке удовлетворения информационных потребностей пользователей для решения задач АИС.

В процедуре концептуального проектирования можно выделить следующие *этапы:*

- обзор и изучение области использования АИС для формирования общего представления о предметной области;
- формирование и анализ круга функций и задач АИС;
- определение основных объектов-сущностей предметной области и отношений между ними;
- формализованное описание предметной области. ***Обзор и изучение области использования АИС для формирования общего представления о предметной области*** осуществляется разработчиком в непосредственном взаимодействии с заказчиком. Разработчиком при этом изучается также и необходимая организационно-распорядительная документация — положения, уставы, инструкции, функциональные обязанности и т.п. На этой основе определяются основные процессы, участники и информационные потоки в предметной области АИС. Принципиальным моментом для фактографических АИС является *фрагментирование предметной области,* т. с. ее разделение на организационные, технологические, функциональные или иные фрагменты. При этом формализатору необходимо прояснить ряд вопросов и решить следующие задачи:
- выделить перечень фрагментов (лица, принимающие решения на различных уровнях организационной иерархии, функционально-технологические структуры, подразделения и т. п.), подлежащих охвату, т. е. информационному отражению в АИС;
- определить информационные потребности и информационные результаты деятельности каждого фрагмента (какая информация, в каком виде, в какие сроки и т. п.);

• определить общие характеристики и содержание процессов потребления и обработки информации в каждом фрагменте (содержание информации, технология ее обработки, передачи, использования и т.д.).

Ответы на эти вопросы помогут сформировать представление о существующей *(«как есть»)* *технологии формирования, накопления, обработки и использования информации* в рамках предметной области АИС и проанализировать совместно с заказчиком *«узкие места»* и *недостатки* в существующей технологии.

Проиллюстрируем данный этап проектирования на примере создания банка данных фактографической АИС по учету, контролю, исполнению и прохождению организационно-распорядительных и информационно-справочных документов. Общее знакомство с предметной областью можно получить в беседе с руководителем и работниками службы документационного обеспечения управления (СлДОУ — секретариат, делопроизводство, канцелярия и т. п.) о системе и порядке документооборота в организации. Дополнительно целесообразно также ознакомиться с регламентирующими данный участок работы нормативными документами.*

---

* В данном случае:

Типовая инструкция по делопроизводству в министерствах и ведомствах Российской Федерации. — М.: Изд-е Комитета по делам архивов при Правительстве Российской Федерации, 1992;

Примерное положение о службе документационного обеспечения управления» (приложение к «Типовой инструкции по делопроизводству...»). — М.: Изд-е Комитета по делам архивов при Правительстве Российской Федерации, 1992;

ГОСТ Р 6.30-97 «Унифицированная Система Организационно-Распорядительной Документации. Требования к оформлению документов».

В результате такого знакомства можно выделить следующие фрагменты предметной области:

• руководители организации; подразделения организации; их руководители; сотрудники, исполняющие документы; мероприятия; документы, обработка которых или подготовка которых реализует управленческие решения и мероприятия;

• служба документационного обеспечения управления; его руководители и работники, ведущие регистрацию, учет, обработку и хранение документов.

Информационные потребности первого фрагмента сводятся к своевременному получению и рассмотрению входящих документов, своевременному получению проектов готовящихся документов для согласования и визирования или принятых внутренних документов для исполнения или использования при организации и проведении различных мероприятий. Кроме того, для первого фрагмента важным является также и получение справочной информации по каким-либо конкретным документам, хранящимся в СлДОУ, поиск нужных документов по реквизитам, тематике, содержанию и т. п.

Информационные потребности второго фрагмента в целом можно охарактеризовать необходимостью организации и контроля всех этапов документооборота в организации (где в данный момент находится конкретный документ, кем завизирован, подписан, утвержден, зарегистрирован, поставлен ли на контроль, исполнен ли, в какое дело приобщен и т. д.).

Характеристики и процессы по документообороту кратко можно выразить следующим образом. Инициирование, подготовка и реализация большинства управленческих решений на различных уровнях организационной иерархии осуществляются на основе использования (руководства), подготовки, принятия и исполнения организационно-распорядительных и информационно-справочных документов. Входящие документы докладываются на решение руководителям организации, которые через резолюции на документах организуют принятие и исполнение необходимых мероприятий. Резолюции на документах доводятся СлДОУ до исполнителей (руководителей) подразделений. Исполнение мероприятия по документу ставится СлДОУ на контроль. Документ после исполнения по решению руководителя может быть уничтожен или приобщен к определенному номенклатурному делу. Внутренние организационно-распорядительные документы (приказы, решения, планы, графики) готовятся в плановом или в инициативном порядке исполнителями из соответствующих подразделений. При этом документ проходит стадию проекта, согласования, утверждения, доведения до исполнителей и исполнения. Для подготовки, принятия и исполнения организационно-распорядительных документов и при проведении различных мероприятий может потребоваться подготовка необходимых информационно-справочных документов (справок, протоколов, отчетов, писем, запросов и т. п.), стадии которых в общем плане могут исключать некоторые стадии организационно-распорядительных документов.

После формирования общего представления о предметной области производится ***определение круга функций и задач,*** решение которых предполагается обеспечивать с помощью АИС. Круг функций и задач АИС определяется на основе *декомпозиции основной цели создания АИС* так называемого «лозунга») *путем формирования последовательно детализируемых способов их решения* с учетом существующей технологии накопления и обработки информации и преодоления ее узких мест с помощью АИС. При этом определяется предварительный *перечень пользователей* системы и уточняются *их информационные потребности.*

В рассмотренном примере с АИС по делопроизводству, очевидно, основной целью (лозунгом) является повышение эффективности управленческих процессов в организации. Решение этой задачи может быть достигнуто через повышение эффективности документооборота путем уменьшения сроков подготовки и прохождения документов, улучшение контроля за исполнением документов, создание эффективной информационно-справочной поддержки для подготовки исполнителями проектов служебных документов. Пользователями АИС, очевидно, должны являться работники СлДОУ, руководители и исполнители документов в подразделениях.

Главным итоговым результатом концептуального проектирования является ***определение основных объектов-сущностей предметной области и отношений между ними.*** В большинстве случаев организационные, технологические и прочие отношения предметной области имеют документальное выражение в различного рода *организационно-распорядительных, информационно-справочных и других нормативно-служебных документах.* Поэтому выделение основных информационных объектов-сущностей предметной области начинается с анализа таких документов и регламентации (положения, инструкции, бланки, формализованные карточки и отчеты, формы, журналы и т. п.).

Анализ «бумажной» документации позволяет сформировать *перечень атрибутов,* характеризующих те или иные объекты-сущности и отношения предметной области. При этом в одном нормативном или служебном документе могут быть отражены атрибуты различных объектов сущностей и отношений. Поэтому можно выделить *два подхода* формирования перечня сущностных объектов предметной области и их атрибутов — *дедуктивный* и *индуктивный.*

В *первом подходе* на основе формирования общего представления о предметной области АИС, функций АИС, а также информационных потребностей основных абонентов-пользователей выделяются *основные понятия и категории,* которыми *оперируют* (которыми выражаются) *фрагменты предметной области.* Данные понятия и категории принимаются за первоначальную основу *списка объектов-сущностей предметной области.* Далее на основе анализа служебной и технологической документации, а также дополнительного взаимодействия с заказчиком формируются *атрибуты,* характеризующие выделенные объекты-сущности.

При определении перечня атрибутов каждого объекта предметной области, как и самого перечня объектов сущностей, руководствуются соображениями ***минимальной достаточности,*** соблюдая знаменитый принцип «бритвы Оккама»* известного английского философа Уильяма Оккама (1285-1349). Иначе говоря, и перечень самих объектов-сущностей и набор их атрибутов должен *быть достаточным* для решения всех частных задач системы и удовлетворять информационным потребностям абонентов-пользователей системы, но он также *не должен быть избыточным,* чтобы минимизировать расходы по накоплению информации и эксплуатации АИС.

---

* «Не умножай число сущностей без необходимости». См., например, с. 317 в работе: Философский словарь / Под ред. М.Т.Тимофеева, 6-е изд., перераб. и доп.— М.: Политиздат, 1991.

Во *втором подходе* на основе анализа служебной и технологической документации выделяются все необходимые для решения частных задач АИС сведения, их характеристики и параметры, и на этой основе формируется *общий перечень атрибутов предметной области.* Далее на основе эвристического анализа производится агрегация (группирование) атрибутов в отдельные группы, образующие объекты-сущности предметной области.

Часть атрибутов и понятий предметной области выражают ***процессы-отношения*** между объектами-сущностями. Такие атрибуты выделяются, и анализируются ***параметры и характер связей,*** которые они выражают — *структурность, направленность, множественность, обязательность* наличия для экземпляров объектов.

Чаще всего выделение объектов-сущностей, их атрибутов и отношений-связей осуществляется *комбинированным способом на итерационной основе,* с многократным уточнением исходного списка объектов, агрегацией атрибутов в группы и т. д. Распространенным приемом в этом случае является

«***обобщение***» некоторых понятий и атрибутов. Суть обобщения заключается в объединении в одну сущность близких или однотипных понятий, категорий, атрибутов на основе анализа их частных проявлений и вариантов. К примеру, совокупность понятий «холодильник», «стиральная машина», «телевизор», «пылесос» и т. п. обобщается сущностью «Бытовые электроприборы» с атрибутом «Тип», имеющим соответствующий список значений.

В примере с АИС по делопроизводству на основе предварительного формирования общего представления о предметной области, а также дополнительного изучения документации СлДОУ (структура журналов регистрации и учета документов, порядок ведения номенклатурных дел и т. д.) можно выделить следующие понятия и категории — *документ, реквизиты, исполнитель. подготовка, согласование, руководитель, утверждение, подписание, регистрация, доклад документа, резолюция, мероприятия, подразделения, доведение до исполнителей, исполнение документа, контроль исполнения, хранение, уничтожение, выдача, дело в производстве, архивное дело.* Часть этих понятий и категорий прямо выражают объекты-сущности (документ), часть выражает атрибуты сущностей (реквизиты документа). Часть понятий (исполнитель, руководитель) можно обобщить одной сущностью (сотрудник). Часть понятий (исполнение, согласование, подписание, утверждение) выражает отношения между сущностями (между документом и сотрудником).

В итоге перечень объектов сущностей предметной области АИС делопроизводства и их атрибутов может быть следующим:*

- Документ (Рег. №, Дата, Название вида, Заголовок к тексту, Гриф, Текст);
- Сотрудник (Таб. №, ФИО, Подразделение, Должность, Кабинет, Телефон);
- Подразделение (№, Наименование);
- Мероприятие (Наименование, Дата начала, Дата окончания, Завершенность);
- Дело (№№, Наименование, Дата начала, Дата окончания, Гриф).

* Данный вариант является исключительно иллюстративно-учебным.

Отношения, которыми охвачены объекты-сущности, можно отобразить следующей таблицей:

*Таблица 3.2*

Отношения объектов-сущностей предметной области АИС по делопроизводству

| Наименование отношения | Между какими объектами-сущностями | | Тип отношений | Обязательность отношений | |
|---|---|---|---|---|---|
| | Первый объект | Второй объект | | С первым объектом | С вторым объектом |
| "Утверждает" | Сотрудник | Документ | "Один-ко-многим" | Нет | Нет |
| "Исполняет" | Сотрудник | Документ | "Один-ко-многим" | Нет | Да |
| "Подписал" | Сотрудник | Документ | "Много-ко-многим" | Нет | Да |
| "Написал" | Сотрудник | Документ | "Один-ко-многим" | Нет | Да |
| "Согласовал" | Сотрудник | Документ | "Много-ко-многим" | Нет | Нет |
| "Зарегистрировал" | Сотрудник | Документ | "Много-ко-многим" | Нет | Нет |
| "Находится" | Документ | Подразделение | "Один-ко-многим" | Нет | Нет |
| "Подготовил" | Подразделение | Документ | "Один-ко-многим" | Нет | Нет |
| "Руководит" | Сотрудник | Подразделение | "Один-ко-одному" | Нет | Да |
| "Работает" | Подразделение | Сотрудник | "Один-ко-многим" | Да | Да |
| "Отправил" | Документ | Мероприятие | "Много-ко-многим" | Нет | Нет |
| "Назначение" | Сотрудник | Мероприятие | "Много-ко-многим" | Нет | Да |
| Ответственный | Сотрудник | Мероприятие | "Один-ко-многим" | Нет | Да |
| Проводит | Подразделение | Мероприятие | "Один-ко-многим" | Нет | Да |
| Участвует | Подразделение | Мероприятие | "Много-ко-многим" | Нет | Нет |
| Приобщен | Дело | Документы | "Один-ко-многим" | Да | Да |
| Ведет | Сотрудник | Дело | "Один-ко-многим" | Нет | Да |

***Формализованное описание концептуальной схемы банка данных*** осуществляется средствами одной из *семантических моделей данных.* Семантические модели данных возникли в противовес ограниченности средств и способов выражения смысловой организации связей между таблицами-сущностями в реляционной модели данных. При этом в большинстве случаев семантические модели применяются на стадии концептуального проектирования с последующим преобразованием концептуальной схемы банка данных в структуру соответствующей реляционной базы данных. В этом смысле разработку концептуальной схемы банка данных называют ***семантическим моделированием данных.***

Наиболее популярными являются разновидности уже упоминавшейся ER-модели, использующие для графического представления структуры данных аппарат диаграмм Бахмана. Формализованное описание ***ER-модели*** было предложено в 1976 году Петером Пин-Шен Ченом.* Основными компонентами структурной составляющей семантической модели Чена являются *сущности, наборы сущностей, атрибуты сущностей, наборы значений атрибутов, ключевые атрибуты сущностей. связи, виды связей, атрибуты связей, наборы связей, ключевые атрибуты связей.***

* Перевод оригинальной статьи П. Чена «Модель «Сущность-Связь» — шаг к единому представлению данных» представлен в журнале СУБД.—№3 — 1995 г. С. 137-157.

** Легко заметить, что семантическая модель Чена является агрегацией и обобщением сетевой и реляционных моделей.

Оригинальные предложения П. Чена по графическому обозначению в диаграммах Бахмана сущностей и связей претерпели изменения, и далее мы будем придерживаться современных вариантов графического изображения концептуальных схем, а именно — объекты-сущности изображать прямоугольниками, при необходимости вставляя в них перечень их атрибутов, связи типа «Один-ко-многим» будем обозначать линиями с парой символов (1 ∞) на концах соответствующих объектов, связи типа «Миогие-ко-многим» линиями с парой символов (∞ ∞) и связи типа «Один-к-одному» линиями с парой символов (1 1). Обязательный характер связи будем обозначать черным квадратиком на конце соответствующей связи, необязательный характер — пустым квадратиком.

В качестве примера на рис. 3.1 приведена концептуальная схема банка данных АИС по делопроизводству.

Формализованное описание ***концептуальной схемы банка данных*** в большинстве случаев осуществляется на бумаге и служит ***основой эскизного проекта создания банка данных*** информационной системы. Следующим шагом в проектировании является построение средствами СУБД схемы банка (базы) данных, которое в большинстве случаев производится «вручную». Иначе говоря, средствами СУБД, поддерживающей ту или иную модель данных, скажем реляционную, создается структура банка данных, соответствующая концептуальной схеме. При этом при переходе от концептуального к схемно-структурному проектированию может иметься разрыв в семантических средствах выражения сущностей, атрибутов, связей и т. д. *Адекватность* реализации концептуальной схемы банка данных определяется, как уже отмечалось, *эвристически* и *эмпирически* в ходе *отладки* и дальнейшей *эксплуатации* банка данных.

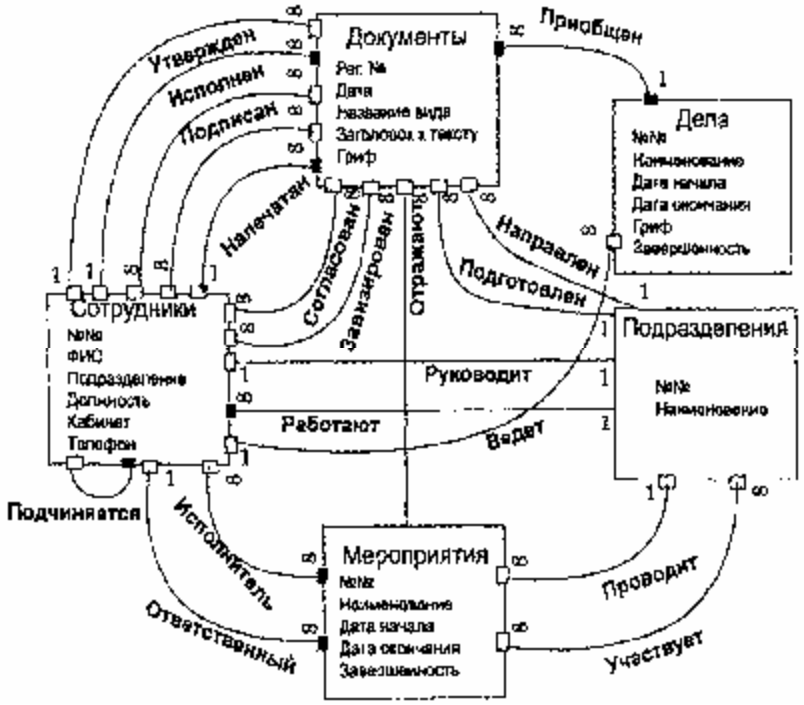



*Рис. 3.1. Пример концептуальной схемы банка данных АИС по делопроизводству*

### 3.2.2. Проектирование схем реляционных баз данных

В реляционных СУБД при ***проектировании схемы реляционной базы*** данных можно выделить следующую ***последовательность*** процедур:

*а*) определение перечня таблиц и их связей;

*б)* определение перечня полей, типов полей, ключевых полей каждой таблицы (разработка схем таблиц-отношений), установление связей между таблицами через внешние ключи;

*в)* определение и установление индексов (индексирования) для полей в таблицах;

*г)* разработка списков (словарей) для полей с перечислительным характером значений данных;

*д)* установление ограничений целостности по полям таблиц и связям;

*е)* нормализация таблиц, доработка перечня таблиц и их связей.

Технологически процесс проектирования разделяют на процесс *предварительного проектирования таблиц и связен* между ними: *а), б), в), г)* и *д)* и последующую *нормализацию таблиц е).*

### 3.2.2.1. Проектирование и создание таблиц

В качестве первоначальной основы при определении перечня таблиц банка данных и связей между ними используется *перечень выделенных объектов-сущностей и отношении концептуальной схемы* банка данных. Иначе говоря, *для каждого объекта-сущности* в реляционных СУБД осуществляется *проектирование соответствующей таблицы*.

***Поля таблиц*** определяются на основе первоначально отработанных *атрибутов* информационных объектов концептуальной схемы банка данных. При этом дополнительно к основным базисным характеристикам (домен, поле-атрибут, кортеж, отношение, ключ, внешний ключ) в СУБД используется ***тип поля***. По своему смыслу тип поля совпадает с понятием типа данных в языках программирования. Традиционные СУБД поддерживают лишь ограниченный набор простых типов полей — числовые, символьные, темпоральные (время, дата), булевы (логические). Современные СУБД оперируют также и со специализированными типами полей (денежные величины), а также со сложными типами полей, заимствованными из языков программирования высокого уровня.

Домены полей таблиц в реляционных СУБД определяют некоторый базисный тип данных для поля, но вместе с тем не являются тождественным понятием типу поля (данных). В некотором смысле домен можно трактовать как некоторое подмножество базисного типа данных с определенной смысловой нагрузкой, — например, множество всех имен из множества всевозможных значений символьного типа данных.

Определение и установление ***ключевых полей*** таблиц в реляционных СУБД является следствием основополагающего требования по ограничениям целостности таблиц-отношений — требования уникальности каждого кортежа-строки. Иначе говоря, одно из полей таблицы, или определенная совокупность полей, в обязательном порядке должно быть определено как ключ. Определение ключевого поля осуществляется на основе смыслового эвристического анализа тематики таблицы при соблюдении принципа *минимальной достаточности*, т.е. количество полей, образующих ключ таблицы должно быть минимальным.

Правильность определения ключа таблицы проверяется эмпирически по возможным ситуациям совпадения у различных кортежей значений ключа. Во многих случаях выбор ключа является нетривиальной задачей. Какое поле, к примеру, выбрать ключевым для таблицы «Сотрудники»? Напрашивается составной ключ из полей «Фамилия», «Имя», «Отчество», однако в конкретных жизненных ситуациях имеется вероятность их совпадения. Можно добавить в состав ключа еще поле «Год рождения», но и при этом все равно сохранится, хотя и несколько снизится, вероятность совпадения. Альтернативным вариантом ключа может быть «№ паспорта», если ситуации с наличием у одного лица нескольких паспортов полностью исключаются. Если в банке данных ограничиться только сотрудниками данной организации, то отработанным вариантом ключа может быть табельный номер сотрудника — «Таб.№».* На практике распространенным приемом при проектировании таблиц является искусственное введение в качестве ключа параметра, являющегося аналогом табельного номера — внутреннего учетного номера экземпляра (записи) соответствующего объекта.

---

* Табельный номер кик раз и является примером уникального параметра для каждого сотрудника в платежных ведомостях (таблицах) для преодоления ситуации с совпадением фамилии, имей и т. д. сотрудников.

В некоторых СУБД для создания полей с уникальными идентификационными номерами кортежей-записей введен дополнительный тип поля, называемый «Счетчиком» или полем типа «AUTOINC». В отличие от обычных числовых (или порядкового типа) полей, значения счетчика генерируются СУБД автоматически при образовании новой записи н только в возрастающем порядке, считая все ранее созданные, в том числе и удаленные записи.

Как уже отмечалось, реляционная модель организации данных по признаку множественности обеспечивает лишь *два типа связей-отношений* между таблицами, отражающими объекты-сущности предметной области, — *«Один-ко-многим»* и *«Один-к-одному»*.

Связь типа ***«Один-ко-многим»*** реализует, вероятно, наиболее распространенный тип отношений между таблицами, когда одной записи в таблице на стороне «один» может соответствовать несколько записей в таблице па стороне «многие». ***Создание связей*** происходит в два этапа. *На первом этапе* в схеме таблицы, находящейся по создаваемой связи на стороне «многие», определяется поле с теми же параметрами (и, как правило, с тем же именем), что и ключевое поле таблицы на стороне «один», т. е. создается поле *внешнего ключа. На втором этапе* с помощью специальных средств СУБД собственно и определяется связь между таблицами путем установления (через специальные внутренние системные таблицы *факта* соответствия ключевого поля таблицы на стороне «один» полю внешнего ключа в таблице на стороне «многие».

В связях типа ***«Один-к-одному»*** каждой связанной записи одной таблицы соответствует в точности одна связанная запись в другой таблице. Как уже отмечалось, данный тип связи образуется путем связывания таблиц по одноименным и однотипным ключевым полям, т. е. когда связываемые таблицы имеют одинаковые ключевые поля. Эта ситуация по сути соответствует разбиению одной большой по количеству столбцов таблицы на две таблицы. Необходимость такого разбиения может быть обусловлена соображениями разграничения доступа к данным по определенным полям, либо целесообразностью выделения некоторого подмножества записей в исходной таблице. Так, например, из исходной таблицы «Студенты» можно выделить все записи по студентам, которые живут в общежитии, и образовать две таблицы, связанные отношением «Один-к-одному». Пример такой связи приведен на рис. 3.2.

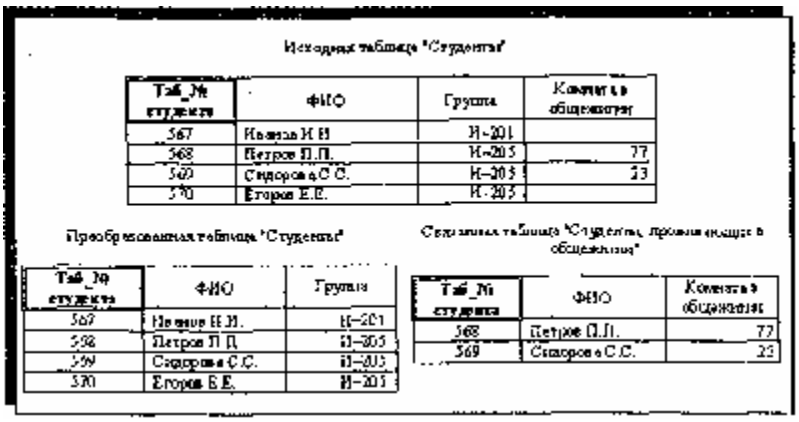



*Рис. 3.2. Пример реализации связи «Один-к-одному» в реляционных СУБД*

Связи типа *«Многие-ко-многим»* в реляционных СУБД в большинстве случаев реализуются через создание двух связей *«Один-ко-многим»,* которые связывают исходные таблицы с третьей общей (связной) таблицей. Ключ связной таблицы состоит, по крайней мере, из двух полей, которые являются полями внешнего ключа для связываемых отношением «Многие-ко-многим» исходных таблиц. Пример реализации связи типа «Многие-ко-многим», выражающей отношение «Согласование» между таблицами «Документы» и «Сотрудники», приведен на рис. 3.3.

Еще одним важным параметром при проектировании таблиц является определение необходимости ***индексирования*** тех или иных *полей* таблиц. Определение и установление индексов полей таблиц базы данных является, как уже отмечалось, важным средством создания условий эффективной обработки данных. Индексирование полей (создание индексных массивов) существенно повышает скорость поиска и доступа к записям базы данных. Однако при этом соответственно замедляется ввод и добавление данных из-за необходимости переупорядочения индексных массивов при любом обновлении, удалении или добавлении записей. Поэтому при проектировании таблиц следует тщательно проанализировать, насколько часто при последующей эксплуатации банка данных потребуется поиск или выборка строк-записей таблицы по значениям тех или иных полей, исходя из функций и задач АИС. На основе такого анализа и определяются те поля таблицы, для которых необходимо создать индексы. К примеру, в таблице «Сотрудники» базы данных по документообороту поле «ФИО» целесообразно определить индексируемым, так как, очевидно, довольно часто будет требоваться доступ к записям таблицы именно по значению этого поля. А вот поле «№ кабинета» вряд ли целесообразно индексировать, так как исходя из задач и функций АИС

по документообороту, можно прогнозировать, что задачи отбора, группирования и прочей обработки записей сточки зрения размещения сотрудников по кабинетам будут достаточно редкими.

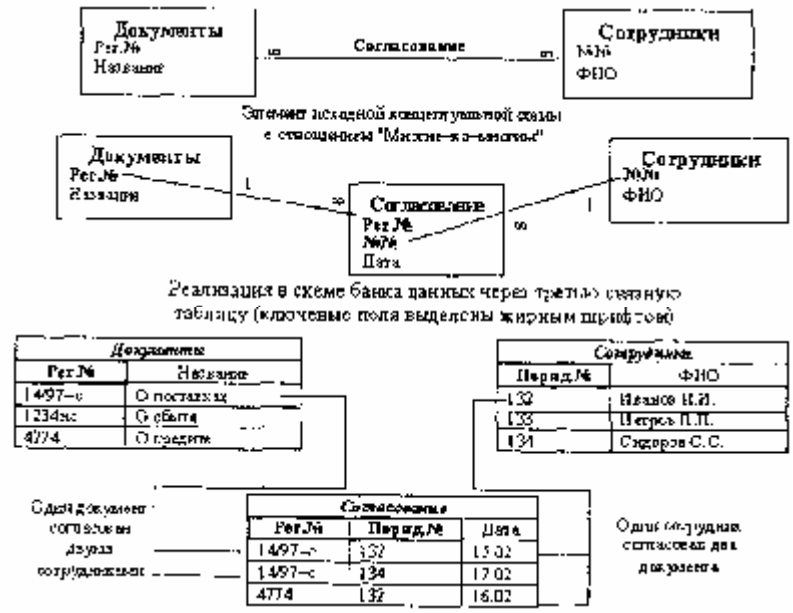



*Рис. 3.3. Реализация связей «Многие-ко-многим» в реляционных СУБД*

Анализ практики использования индексов в базах данных позволяет сделать вывод, что если в одной таблице установлено более 10 индексов, то либо недостаточно продумана структура базы данных (таблицы), либо не совсем обоснованно определены вопросы обработки данных исходя из задач АИС.

Ключевые поля в большинстве случаев являются индексируемыми автоматически, т. к. поиск и доступ к записям в базе данных производится прежде всего по значениям ключевых полей.

В большинстве СУБД вопросы внутреннего устройства индексных массивов остаются скрытыми и недоступными как для конечных пользователей, так и для проектировщиков. Допускается только лишь качественное определение режима индексирования — без повторов значений индексируемых полей* и с возможностью таких повторов, что, очевидно, определяет разные типы индексных массивов — Б-деревья или инвертированные списки.**

---

* Так называемый уникальный индекс. Автоматически устанавливается для ключевых полей.

** Как правило, данные вопросы в документации по СУБД не отражаются.

При определении параметров полей таблиц важное значение имеет также выделение полей с *перечислительным (словарным, списковым) характером значений.* Значения таких полей определяются из некоторого унифицированного ***списка-словаря.*** К примеру, поле «Образование» является не просто текстовым, а по сути текстово-списочным, так как набор его всевозможных значений составляет унифицированный список—«Начальное», «Среднее», «Среднеспециальное», «Среднетехническое», «Высшее». Некоторые СУБД обеспечивают возможность при проектировании таблиц построения и привязки к соответствующим полям таких списков-словарей.

Установление списков значений или, иначе говоря, словарей позволяет упростить в дальнейшем ввод данных в записях по таким полям путем выбора соответствующего значения из словаря и, кроме того, унифицировать ввод одинаковых значений в различных записях. К примеру, при ручном наборе значения «Среднее» помимо различий из-за возможных орфографических ошибок, могут быть также и регистровые различия («Среднее», «среднее», «СРЕДНЕЕ»), что в дальнейшем приведет к ошибкам поиска, фильтрации и выборки записей.

*Списки (словари)* значений могут быть *фиксированными,* не изменяемыми в процессе эксплуатации банка данных, или *динамическими.*

Фиксированные списки «привязываются» к соответствующим полям через специальные механизмы конкретной СУБД и размещаются в системных таблицах (каталоге) базы данных, доступа к которым пользователи-абоненты системы не имеют.

Динамические словари в большинстве случаев реализуются через создание дополнительных одностолбцовых таблиц, строки которых являются источником списка значений для полей других таблиц. Привязка подобных словарных таблиц в качестве источника значений для полей других

таблиц осуществляется также через специальные механизмы конкретной СУБД. Такие таблицы в дальнейшем доступны пользователям банка данных. Соответственно обновление, добавление или удаление записей в таблицах-словарях позволяет изменять словарный базис для полей соответствующих таблиц. В некоторых СУБД дополнительно может также устанавливаться режим *ограничения значений* словарно-списочных полей только установленным *списком* значений. Установление такого режима целесообразно в тех случаях, когда нужно исключить, в принципе, даже случайный (ошибочный) выход значений поля за пределы списка. Так, например, в случае поля «Оценка» значения могут быть только из списка «Неудовлетворительно», «Удовлетворительно», «Хорошо», «Отлично».

Одним из важных в практическом плане этапов проектирования таблиц является ***установление ограничений целостности по полям и связям.*** Как уже указывалось, в исходном виде в реляционной модели данных основным ограничением по значению полей является требование уникальности табличных строк-кортежей, что проявляется в требовании уникальности значений ключевых полей. Дополнительно в реляционных СУБД могут устанавливаться требования уникальности значений и по другим (не ключевым) полям через создание для них индексов в режиме без повторов (*UNIQUE),* а также установления режима обязательного заполнения в строках-кортежах определенных полей (режим *NOT NULL)*.

Вместе с тем современные СУБД могут предоставлять и более развитые возможности установления ограничений целостности данных. Можно определять допустимые диапазоны значений полей (например, значение поля «Оклад» не может быть меньше величины минимального размера оплаты труда), а также относительные соотношения значений по определенным полям таблицы (например, значение поля «Количество» в таблице «Товары» не может быть меньшим значения поля «Мин_запас», значение поля «Дата_исполнения» в таблице «Заказы» не может быть позднее поля «Дата_размещения» плюс определенное количество дней, скажем 7 дней, исходя из того требования, что любой заказ клиента должен быть исполнен в течение максимум 7 дней). Такие ограничения целостности данных отражают ту часть *правил и особенностей предметной области АИС,* которая не формализуется в рамках реляционной модели данных.*

* Ввиду данного обстоятельства в некоторых англоязычных источниках такие ограничения целостности называют «правилами бизнеса».

Другая часть ограничений целостности данных касается уже упоминавшегося требования ***целостности ссылок.***

Поддержание очевидного требования целостности ссылок встречает определенные трудности на этапе эксплуатации банка данных. Такой типичной ситуацией является, к примеру, удаление записи по отделу № 710 в таблице «Отделы». Если оставить без соответствующего изменения все связанные кортежи в таблице «Сотрудники» (кортежи со значением «710» внешнего ключа «№ отдела»), то как раз и произойдет нарушение целостности ссылок.

На практике в реляционных СУБД существует *три подхода* реализации требования целостности по ссылкам. В *первом подходе* запрещается удалять кортеж-запись какой-либо таблицы, если на него существуют ссылки из связанных таблиц. Иначе говоря, запись по отделу № 710 можно удалить лишь в том случае, если перед этим удалены (или переадресованы по ссылкам на другие записи) все сотрудники со значением «710» внешнего ключа «№ отдела». Во *втором подходе* ***при удалении записи*** отдела № 710 значения внешних ключей всех связанных кортежей таблицы «Сотрудники» автоматически становятся ***неопределенными.**** При *третьем подходе* осуществляется ***каскадное удаление*** всех кортежей, связанных с удаляемым кортежем, т. е. при удалении записи по отделу № 710 автоматически удаляются записи из таблицы «Сотрудники» с соответствующим значением внешнего ключа «№ отдела». Выбор того или иного режима целостности ссылок определяется на основе эвристического анализа правил и возможных ситуаций в предметной области АИС.

* Соответственно в этом случае СУБД различает «пустые» (N*ULL*) и «неопределенные» значения полей.

В заключение отметим, что разделение процесса проектирования таблиц на этапы а, б, в, г и д является условным, а сам процесс предварительного проектирования (создания) таблиц, как будет показано в параграфе 4.1, реализуется специальными инструкциями языка SQ*L*.

### 3.2.2.2. Нормализация таблиц

Нормализация реляционных таблиц-отношений определяется требованиями атомарности значений полей, а также более общим требованием рациональности группировки полей-атрибутов по различным таблицам.

С содержательной точки зрения нормализацию таблиц можно рассматривать как некоторую доработку концептуальной схемы базы данных в тех случаях, когда концептуальное проектирование банка данных произведено без достаточной проработки, и в ходе предварительного проектирования созданы таблицы, отражающие данные сразу нескольких объектов-сущностей предметной области АИС.

С формальной точки зрения нормализацию можно представить как последовательный процесс разбиения и преобразования некоторого небольшого исходного набора таблиц для построения набора взаимосвязанных таблиц в *нормальных формах.* Основатель реляционной модели данных Е. Кодд выделял три нормальные формы — *первую, вторую* и *третью.* Этот набор в дальнейшем был дополнен нормальной формой *Бойса-Кодда,* и далее *четвертой* и *пятой* нормальными формами.

Наиболее простой ***нормальной формой*** является ***первая,*** суть которой определяется уже упоминавшимся требованием атомарности (неделимости) полей и единственности значений по полям в реляционной модели данных. На рис. 3.4 приведен пример ненормализованной таблицы «Сотрудники», имеющей составное (делимое) поле «Мероприятия...» с множественными значениями по полям «Условное наименование» и «Награда».

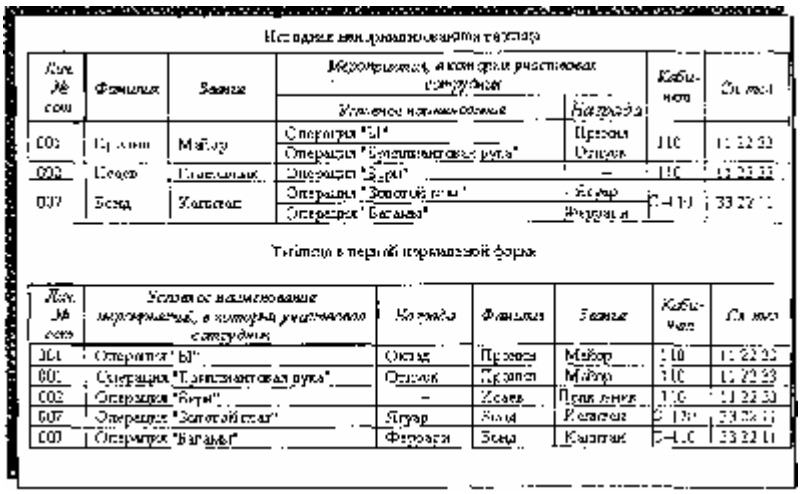



*Рис. 3.4. Пример приведения таблицы к первой нормальной форме*

Приведение таких таблиц к первой нормальной форме осуществляется путем образования составных ключей, при которых устраняются ситуации с множественными значениями полей. Данный процесс иллюстрируется на рис. 3.4 (жирной рамкой выделены ключевые поля).

Из приведенного примера видно, что таблицы *в первой нормальной форме* могут содержать многочисленные *ситуации дублирования* данных (в приведенном примере по полям «Фамилия», «Звание», «Кабинет», «Сл.тел.»). Кроме того, в таблице, находящейся в первой нормальной форме, могут встречаться и другие *аномалии схемы таблиц-отношений.* В частности, в приведенном примере нельзя образовать запись для сотрудника, не участвовавшего ни в одной операции. Удаляя запись об участии определенного сотрудника в определенной операции, можно удалить информацию о том, что он вообще работает в определенном подразделении. При переводе сотрудника в другое подразделение или при его перемещении в другой кабинет приходится изменять все записи-кортежи с данным сотрудником по различным операциям. Поэтому Е. Коддом был разработан специальный механизм разбиения таблиц для приведения к более совершенным нормальным формам. Этот механизм основан на понятии ***функциональной зависимости полей-атрибутов.***

Поле-атрибут Y *функционально зависит* от поля-атрибута X, если любому значению X всегда соответствует в точности одно значение Y. К примеру, атрибут «ФИО» функционально зависит от атрибута «Таб.№», т. е. каждому значению атрибута «Таб.№» соответствует только одно значение атрибута «ФИО». Другим примером является функциональная зависимость поля «Кабинет» от поля «Фамилия», так как обычно один сотрудник имеет рабочее место только в одном кабинете. Легко убедиться, что в таблице, находящейся в первой нормальной форме, все неключевые атрибуты функционально зависят от ключа таблицы.

***Вторая нормальная форма*** основывается на понятии ***полной функциональной зависимости.*** Функциональная зависимость неключевого атрибута от составного ключа таблицы называется *полной,* если он функционально зависит в целом от составного ключа, но не зависит отдельно от любой части составного ключа. В примере, приведенном на рис. 3.4, значение поля-атрибута «Фамилия» определяется только значением поля «Лич.№ сотр.», которое является частью составного ключа таблицы, и, следовательно, функционально полной зависимости неключевого поля-атрибута «Фамилия» от составного ключа нет. В полной функциональной зависимости от составного ключа находится поле-атрибут «Награда», так как только комбинация значений полей «Лич.№ сотр.» и «Условное наименование мероприятий...» определяет конкретное значение поля «Награда».

Таблица-отношение находится во ***второй нормальной форме,*** если она находится в первой нормальной форме и все ее неключевые атрибуты функционально полно зависят от составного ключа. Для перевода таблицы из первой нормальной формы во вторую необходимо:

- образовать проекцию (вертикальное подмножество) исходной таблицы на составной ключ и на поля, находящиеся в полной функциональной зависимости от составного ключа;
- построить еще одну или несколько проекций на часть составного ключа с полями, функционально зависящими от этой части ключа.

На рис. 3.5 показан пример приведения таблицы из первой нормальной формы во вторую.


*Рчс. 3.5. Пример приведения таблицы из первой во вторую нормальную форму*

В таблицах, находящихся во второй нормальной форме, большинство аномалий, присущих первой скорме, устранено. Вместе с тем по определенным атрибутам также могут сохраняться многочисленные ситуации дублирования данных. Так, например, в приведенном на рис. 3.5 примере происходит неоправданное дублирование информации о служебном телефоне «11 22 33», так как атрибут «Сл. тел.» фактически зависит не от атрибута «Лич.№ сотр.»», а от атрибута «Кабинет».* Иначе говоря, наблюдается цепочка функциональной зависимости атрибутов «Лич. № сотр.» — «Кабинет» — «Сл. тел.», а функциональная зависимость атрибута «Сл.тел.» от атрибута «Лич. № сотр.» является лишь логическим следствием такой цепочки зависимостей. В таких ситуациях говорят о ***транзитивной зависимости атрибута*** «Сл. тел.» от атрибута «Лич. № сотр.».

* В большинстве жизненных ситуаций в одной комнате для сотрудников установлен один общий телефон.

По определению таблица-отношение находится в ***третьей нормальной форме,*** если она находится во второй нормальной скорме и каждое ее не ключевое поле-атрибут *нетранзитивно* зависит от первичного ключа. Легко увидеть, что альтернативным определением третьей нормальной формы является *взаимная независимость неключевых атрибутов и их полная функциональная зависимость от первичного ключа.*

Для преобразования из второй в третью нормальную форму таблицу-отношение разделяют на две или более проекции так, чтобы конечные поля-атрибуты в цепочках транзитивной зависимости вынести в отдельные таблицы, связав разделившиеся части таблицы внешними ключами по полям-атрибутам, находящимся внутри цепочек транзитивной зависимости. На рис. 3.6 проиллюстрирован процесс приведения таблицы из второй в третью нормальную форму путем разделения цепочки транзитивной зависимости «Лич.№ сотр.» — «Кабинет» — «Сл. тел.». Внутреннее в этой цепочке поле-атрибут «Кабинет» стало соответственно внешним ключом в первой таблице и первичным ключом во второй таблице.

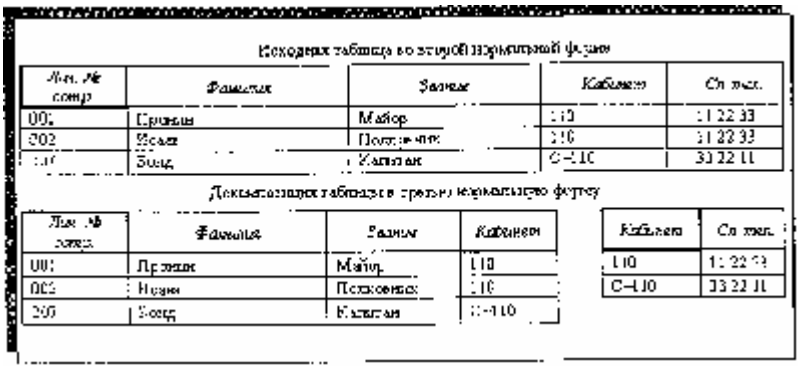


*Рис. 3.6. Пример приведения таблицы в третью нормальную форму*

На практике третья нормальная форма устраняет большинство аномалий схем таблиц-отношений, а также ситуации дублирования данных, и после декомпозиции исходных таблиц-отношений до третьей нормальной формы процесс нормализации заканчивается. Вместе с тем в некоторых случаях третью нормальную форму можно также «улучшить», в частности приведением таблицы-отношения в *нормальную форму Бойса-Кодда.*

Такие ситуации связаны с наличием так называемых ***детерминантов***—совокупности атрибутов (составных атрибутов), от которых функционально полно зависят другие атрибуты. В результате таблица может находиться в третьей нормальной форме, т.е. все его неключевые атрибуты взаимно функционально независимы, но имеется полная функциональная зависимость некоторых атрибутов от совокупности других атрибутов (детерминантов). Пример такой таблицы приведен на рис. 3.7.

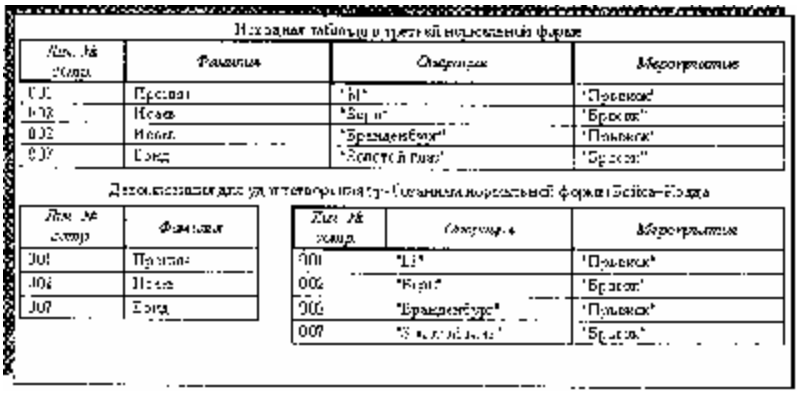


*Рис. 3.7. Пример приведения таблицы из третьей нормальной формы в форму Бойса-Кодда*

В данной таблице имеются два детерминанта — («Лич. № сотр.», «Операция») и («Фамилия», «Операция»), от каждого из которых функционально полно зависит поле-атрибут «Мероприятие».

Таблица-отношение находится в ***нормальной форме Бойса-Кодда*** тогда и только тогда, когда каждый его детерминант является возможным ключом. Очевидно, что если в таблице имеется всего один возможный ключ, то он одновременно является детерминантом, и нормальная форма Бойса-Кодда совпадает с третьей нормальной формой.*

* Поэтому иногда нормальную форму Бойса-Кодда считают частным случаем третьей нормальной формы.

Таблица, приведенная на рис. 3.7, не удовлетворяет требованию нормальной формы Бойса-Кодда, так как если установить ключом детерминант («Лич.№ сотр.», «Операция»), то поле-атрибут «Фамилия» будет функционально зависеть от части составного ключа (от поля «Лич.№ сотр.») и нарушатся требования второй нормальной формы. Следствием та кой ситуации является дублирование данных по полям-атрибутам «Лич. № сотр.» и «Фамилия».

Для приведения таблицы в нормальную форму Бойса-Кодда необходимо произвести декомпозицию таблицы так, чтобы исключить случаи пересечения по некоторым полям-атрибутам имеющихся детерминантов.

Встречаются также случаи, требующие «улучшения» и нормальной формы Бойса—Кодда. Такие ситуации связаны с ***многозначной зависимостью атрибутов.*** В таблице-отношении с полями-атрибутами X, Y, Z существует многозначная зависимость атрибута Y от атрибута X тогда и только тогда, когда любое значение из множества Y, соответствующее паре значений атрибутов X и Z, зависит только от значения Y. Для примера рассмотрим таблицу на рис. 3.8. При этом будем считать, что каждый сотрудник, привлеченный к какой-либо операции, в обязательном порядке участвует во

всех проводимых в рамках данной операции мероприятиях. В этом случае единственно возможным ключом является совокупность всех трех полей атрибутов (каждый сотрудник может участвовать в разных операциях, в одной операции может участвовать несколько сотрудников).

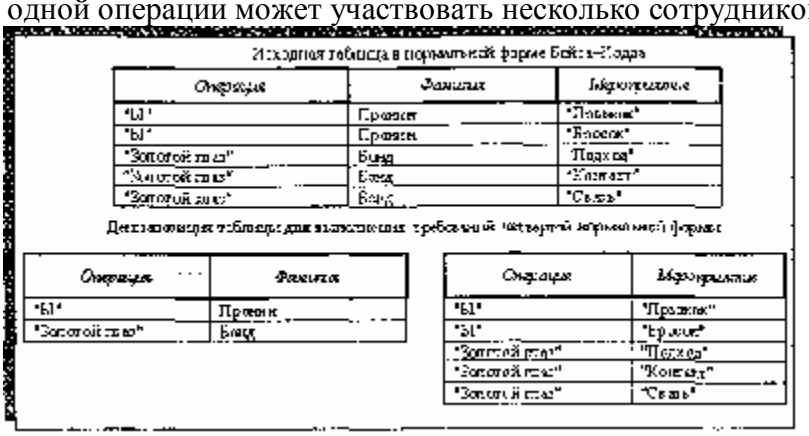



*Рис. 3.8. Пример декомпозиции таблицы из нормальной формы Бойса-Кодда в четвертую нормальную форму*

Так как имеется единственный возможный составной ключ, то данная таблица автоматически находится в нормальной форме Бойса—Кодда. При этом имеется многозначная зависимость поля-атрибута «Фамилия» от поля-атрибута «Операция» (для любой пары значений атрибутов «Операция»—«Мероприятие» значение атрибута «Фамилия» фактически определяется только значением атрибута «Операция» при сформулированном выше условии участия каждого сотрудника автоматически во всех мероприятиях данной операции). Аналогично имеется многозначная зависимость поля-атрибута «Мероприятия» от поля-атрибута «Операция». В такой ситуации для внесения информации о новом сотруднике, вовлекаемом в какую-либо операцию, придется добавить столько строк-кортежей, сколько мероприятий проводится в рамках данной операции.

Подобные аномалии устраняет *четвертая нормальная форма.* Таблица-отношение находится в ***четвертой нормальной форме*** тогда и только тогда, когда в случае существования многозначной зависимости атрибута Y от атрибута X все остальные атрибуты функционально зависят от атрибута X.

Приведение таблицы в четвертую нормальную форму основывается на *теореме Фейджина,* в которой доказывается возможность проецирования без потерь* таблицы с атрибутами X, Y, Z в две таблицы с атрибутами X, Y и X, Z, когда существует многозначная зависимость атрибута Y от атрибута X. Процесс декомпозиции проиллюстрирован на рис. 3.8.

* Проецирование без потерь предполагает полное и безызбыточное восстановление исходной таблицы путем операции соединения таблиц—результатов декомпозиции.

Наиболее сложной при нормализации является *пятая нормальная форма,* связанная с наличием в таблице-отношении зависимостей соединения. В таблице-отношении с полями-атрибутами X, Y, ..., Z имеется ***зависимость соединения*** тогда и только тогда, когда таблица может быть без потерь восстановлена на основе операций соединения своих проекций, например (X, Y), (X, Z), (Y, Z) и т. д. по полям-атрибутам X, Y,..., Z.

Таблица-отношение может находиться в четвертой нормальной форме, но когда в ней имеется зависимость соединения, могут возникать аномалии при операциях добавления/удаления строк-кортежей. Для примера рассмотрим таблицу, приведенную на рис. 3.9. Ключом таблицы является совокупность всех трех полей-атрибутов, так как сотрудник может входить в состав разных групп и участвовать в разных мероприятиях, каждое из которых может проводиться разными группами. В таблице нет детерминантов, отсутствуют функциональные и многозначные зависимости, т. е. таблица находится в четвертой нормальной форме. Тем не менее в данной таблице нельзя удалить информацию по участию Бонда в мероприятии «Контакт», не удалив при этом вообще информацию о мистере Бонде в таблице. Нельзя также добавить строку-запись о вхождении мистера Бонда еще и в группу «F», если при этом он не участвовал ни в одном мероприятии.

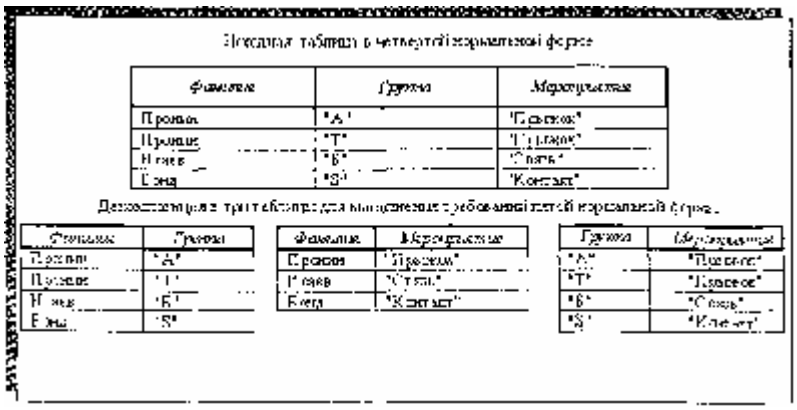


*Рис. 3.9 Пример декомпозиции таблицы из четвертой в пятую нормальную форму*

Подобные аномалии устраняются приведением таблицы в пятую нормальную форму. Таблица-отношение находится в ***пятой нормальной форме*** тогда и только тогда, когда любая зависимость соединения в ней следует из существования некоторого возможного ключа.

Приведение таблицы в пятую нормальную форму осуществляется путем ее декомпозиции сразу на несколько таблиц отношений. Если предположить, что в таблице, представленной на рис. 3.9. имеется зависимость соединения по составным атрибутам «Фамилия»-«Группа», «Фамилия»-«Мероприятие», «Группа»-«Мероприятие»,* то, разбив таблицу на три проекции по соответствующим полям-атрибутам, можно удовлетворить требованиям пятой нормальной формы и устранить отмеченные аномалии.

* Наличие зависимости соединения является нетривиальным предположением, основывающимся в большинстве случаев на эвристических соображениях, т.е. в данном случае на уверенности, что при соединении трех таблиц «Фамилия»-«Группа», «Фамилия»-«Мероприятие» и «Группа»-«Мероприятие» не произойдет каких-либо потерь пли появления лишних данных относительно исходной таблицы.

Из-за нетривиальности зависимости соединения пятая нормальная форма практически не используется.

Нормализация исходных таблиц при проектировании банка данных, как уже отмечалось, проводится для рационализации группировки полей-атрибутов в схемах таблиц с целью устранения аномалий и дублирования данных. Вместе с тем процесс нормализации, как правило, приводит к декомпозиции исходных таблиц на множество связанных и несвязанных более простых таблиц. В результате при обработке данных приходится осуществлять множество операций соединения таблиц, что накладывает высокие требования к вычислительным ресурсам при больших объемах банка данных. Поэтому, как уже отмечалось, на практике в большинстве случаев при проектировании схем реляционных баз данных ограничиваются третьей нормальной формой таблиц-отношений.

Результатом проектирования и нормализации таблиц является законченная схема (логическая структура) базы данных. Технологически описание схемы базы данных помещается в ***каталог базы данных***, который в реляционных СУБД, в свою очередь, представляет также ***таблицу***,* структура (поля) которой описывает объекты базы данных (таблицы), их названия, поля, параметры и т. д. Обычно каталог базы данных хранится в *файле БД вместе с данными.* В определенных случаях (системы «Клиент-сервер», распределенные системы, системы на основе репликации) может устанавливаться специальный режим размещения и доступа к каталогу базы данных.

* В некоторых СУБД, как уже указывалось, каталог базы данных именуют *системными таблицами.*

Для повышения эффективности схемно-структурного проектирования банков данных на рынке программных средств СУБД появился специальный класс программ, называемых ***CASE-системами***.* Наиболее известными из них являются Designer 2000 компании «Oracle», ErWin компании «LogicWorks», PowerBulder компании «PowerSoft». Такие системы предоставляют проектировщику банка данных средства концептуального проектирования баз данных на основе техники семантического моделирования. При этом широко используются средства визуализации определения и описания информационных объектов, связей и их атрибутов, что делает процесс проектирования максимально наглядным и позволяет проектировщику сосредоточиваться на смысловом аспекте структуры банка данных. Разработанная таким образом концептуальная схема банка данных *транслируется* CASE-системой в схему соответствующего реляционного банка, избавляя проектировщика от утомительных процедур «ручного» перевода концептуальной (семантической) схемы в реляционную.

* Computer Aided Software Engineering—системы автоматизированного проектирования.

Тенденция расширения средств описания схем реляционных банков данных в сторону оснащения их элементами семантического моделирования наблюдается и в самих СУБД. В некоторых СУБД имеются специальные инструменты визуального графического определения связей между таблицами и ограничений целостности по ним. Кроме того, появляются также специальные анализаторы структуры таблиц базы данных с точки зрения рациональности и дублирования данных, которые позволяют в некоторой степени автоматизировать рассмотренные выше процессы нормализации таблиц.

**Вопросы и упражнения**

1. Врачи поликлиники ведут прием и обследование пациентов. Выделите основные объекты-сущности предметной области и отношения между ними для концептуального проектирования банка данных АИС, автоматизирующей учет обследований пациентов. Изобразите средствами ER-модели концептуальную схему.
2. В учебном заведении преподаватели проводят занятия по учебным дисциплинам со студентами разных групп. Выделите основные объекты-сущности предметной области и отношения между ними для концептуального проектирования банка данных АИС, автоматизирующей учет расписаний. Изобразите средствами ER-модели концептуальную схему.
3. Учет материальных средств по подразделениям предусматривает их закрепление за определенными сотрудниками. Выделите основные объекты-сущности предметной области и отношения между ними для концептуального проектирования банка данных АИС, автоматизирующей учет матсредств и материально ответственных. Изобразите средствами ER-модели концептуальную схему.
4. При концептуальном проектировании банка данных АИС, автоматизирующей ведение Табеля рабочего времени сотрудников организации, выделены следующие объекты-сущности:


С учетом того что Табель является основой для начисления сотрудникам заработной платы, определите необходимые атрибуты по каждому объекту-сущности концептуальной схемы.

5. При проектировании таблицы «Преподаватели» выделены следующие атрибуты — ФИО, Кафедра (Истории, Математики, Информатики), Должность (Зав. кафедры, Профессор, Преподаватель, Ассистент), Ученая степень (Кандидат наук. Доктор наук). Ученое звание (Старший научный сотрудник, Доцент, Профессор, Академик), Пед. стаж. Определите и обоснуйте для каждого атрибута тип поля и другие параметры (обязательность заполнения, словарно-списочный характер и тип словаря, индексируемость и тип индекса, возможные ограничения целостности данных). Выберите из имеющихся атрибутов или предложите дополнительно ключ таблицы.
6. При проектировании таблицы «Автомобили» базы данных «Запасные части» выделены следующие атрибуты — Модель, Производитель (ВАЗ, АЗЛК, ГАЗ, ИЖМаш, УАЗ), Категория (Легковой, Грузовой, Специальный), Грузоподъемность, Год начала производства, Год прекращения производства, Фото. Определите и обоснуйте для каждого атрибута тип поля и другие параметры (обязательность заполнения, словарно-списочный характер и тип словаря, индексируемость и тип индекса, возможные ограничения целостности данных). Выберите из имеющихся атрибутов или предложите дополнительно ключ таблицы.
7. При концептуальном проектировании в базе данных по учету выдачи пропусков на вход в административное здание сотрудникам организации выделены следующие объекты-сущности, для каждого из которых создается таблица со следующими полями:

• «Сотрудник»— *Таб_№, ФИО, Должность, Подразделение;*
• «Подразделение»—*№№, Наименование, Руководитель;*
• «Пропуск»— *Таб_№_сотр., №_пропуска, Дни, Время, Кто подписал.*

С учетом того что у сотрудника может быть только один пропуск, определите и обоснуйте типы, способы и другие параметры создания связей между таблицами.

8. При концептуальном проектировании в базе данных «Технологические операции» выделены следующие объекты-сущности, для каждого из которых создается таблица со следующими полями:

• «Подразделения» — *№№, Наименование, Профиль* (Производственно-технологический, Сбытовой, Снабженческий, Организационно-управленческий);

• «Операции»—*Код, Наименование, Описание;*

• «Комплектующие» — *Код, Наименование, Тип* (Крепеж, Электрооборудование, Резинотехнические изделия), *Количество, Минимально необходимое количество на складе.*

С учетом того что одно подразделение может участвовать в выполнении нескольких технологических операций и, соответственно, одна операция может выполняться несколькими подразделениями, и, кроме того, для одной операции может требоваться определенное количество различных комплектующих, а каждое комплектующее может, в свою очередь, комплектовать несколько различных технологических операций, определите типы, способы и другие параметры создания связей между основными таблицами.

9. Приведите к первой нормальной форме следующую ненормализованную таблицу (в жирной рамке ключ таблицы):

| Таб. № | ФИО | Подразделение | Командировка | | | | |
|---|---|---|---|---|---|---|---|
| | | | №№ | Дата начала | Дата окончания | Организация | Город |
| 231 | Иванов И.И. | 3-й отдел | 7234 | 01.10.99 | 20.10.99 | ПО "Кристалл" | Черноморск |
| | | | 7245 | 15.11.99 | 21.11.99 | ПО "Заря" | Гряжск |
| 233 | Петров П.П. | 3-й отдел | 7347 | 15.12.99 | 15.12.99 | ЗАО "Сталь" | Урюпинск |

10. Приведите ко второй нормальной форме следующие та блицы, находящиеся в первой нормальной форме (в жирной рамке ключ таблицы):

| Таб. № | Спектакль | Роль | Фамилия актера | Звание | Амплуа |
|---|---|---|---|---|---|
| 12 | Горе от ума | Чацкий | Миронов | Заслуж. артист | Герой |
| 12 | Свадьба Фигаро | Фигаро | Миронов | Заслуж. артист | Герой |
| 9 | Горе от ума | Фамусов | Пальнов | Народ. артист | Разнохаракт. |

| №№ клиента | №№ кредита | Сумма | Дата выдачи | Погашен | ФИО клиента | Адрес | Телефон |
|---|---|---|---|---|---|---|---|
| 532 | 1254 | 100000р. | 10.01.96 | Да | Чистяков А.А. | Рахова-15 | 11 22 33 |
| 532 | 1347 | 1000000р. | 10.01.97 | Нет | Чистяков А.А. | Рахова 15 | 11 22 33 |
| 673 | 1368 | 2000000р | 11.03.97 | Нет | Яловой В.В. | Вощето-2 | 33 22 .1 |

11. Приведите к третьей нормальной форме следующие таблицы, находящиеся во второй нормальной форме (в жирной рамке ключ таблицы):

| Рейс | Маршрут | Тип самолета | Количество мест |
|---|---|---|---|
| 3437 | Москва — Нью-Васюки | Ил-62 | 180 |
| 23-пс | Москва — Черноморск | Як-42 | 120 |
| 777 | Москва — Гряжск | Як-42 | 120 |

| Завод. №- | Производитель | Марка | Сырье | В эксплуатации | Производительность |
|---|---|---|---|---|---|
| 34-071 | "Раздолье" | Ректиф-2 | Картофель | 01.03.97 | 200л/сутки |
| 5677-с | "Зеленогорье" | Ректиф-3 | Зерно | 03.02.98 | 500л/сутки |
| 15628 | "Мечта" | Ректиф-2 | Картофель | 01.03.97 | 200л/сутки |

## 4. ВВОД, ОБРАБОТКА И ВЫВОД ДАННЫХ В ФАКТОГРАФИЧЕСКИХ АИС

Как уже отмечалось, цикл функционирования автоматизированных информационных систем включает сбор, комплектование данных, поиск и выдачу сведений для удовлетворения информационных потребностей абонентов систем. Если исключить организационно-технологические аспекты сбора,

комплектования и выдачи информации, технология работы пользователей с базами данных АИС включает ввод (загрузку), обработку и вывод данных.

Предоставление пользователю средств реализации функций ввода, обработки и выдачи данных является одной из основных функций интерфейса автоматизированных информационных систем.

## 4.1. Языки баз данных

Как следует из рассмотрения внутренней схемы баз данных, одной из основных функций систем управления базами данных является создание и поддержание собственной системы размещения и обмена данными между внешней (дисковой) и оперативной памятью. От эффективности реализации в каждой конкретной СУБД данной функции (формат файлов данных, индексирование, хэширование и буферизация) во многом зависит и эффективность функционирования СУБД в целом. Поэтому основные усилия создателей первых СУБД в конце 60-х — начале 70-х годов были сосредоточены именно в этом направлении.

Однако такой подход приводил к «самостийности», уникальности каждой конкретной СУБД и созданной на ее основе автоматизированной информационной системы. В результате для реализации любой функции по вводу, обработке или выводу данных требовались квалифицированные программисты для написания специальных программ на алгоритмических языках высокого уровня (в 70-х годах ФОРТРАН, КОБОЛ и др.), «знающих» особенности структуры и способы размещения данных во внешней и оперативной памяти. В итоге работа с базами данных осуществлялась через посредника в виде квалифицированного программиста, «переводящего» информационные потребности пользователя в машинный код,* что схематично иллюстрируется на рис. 4.1.

* В этом плане примечателен афоризм Чарльза Бахмана, который метко подметил, что «программист— это штурман в море данных». Статью по поводу присуждения ему премии Тьюринга за пионерские работы в области технологий баз данных он так и назвал — «The Programmer As Navigator» (Программист как штурман). [**Вацкевич Д**. Стратегии клиент/сервер, – К.: Диалектика, 1996. С. 216.]

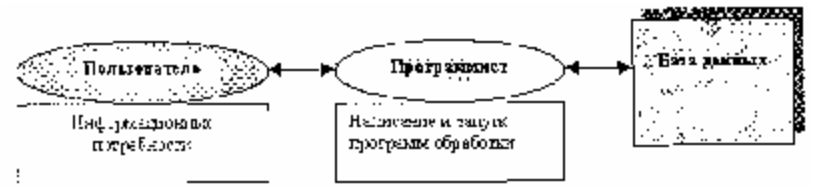



*Рис. 4.1. Схема взаимодействия пользователя с базой данных в ранних СУБД*

Такое положение дел приводило к большим накладным расходам при создании и эксплуатации автоматизированных информационных систем и в определенной степени сдерживало распространение вычислительной техники в процессах информационного обеспечения деятельности предприятий и организаций.

Основателем теории реляционных СУБД Е. Коддом было выдвинуто предложение о создании специального *языка для общения* (взаимодействия) *пользователя-непрограммиста с базами данных.* Идея такого языка сводилась к набору из нескольких фраз-примитивов английского языка («выбрать», «обновить», «вставить», «удалить»), через которые пользователь-непрограммист ставил бы «вопросы» к СУБД по своим информационным потребностям. В этом случае дополнительной функцией СУБД должна быть интерпретация этих «вопросов» на низкоуровневый язык машинных кодов для непосредственной обработки данных и предоставление результатов пользователю. Так родилась уже упоминавшаяся по структуре СУБД «машина данных». Иначе говоря, машина данных «понимает» язык базы данных и в результате разделяет собственно данные и задачи по их обработке. В таком подходе взаимодействие пользователя с базой данных можно проиллюстрировать схемой, приведенной на рис. 4.2.

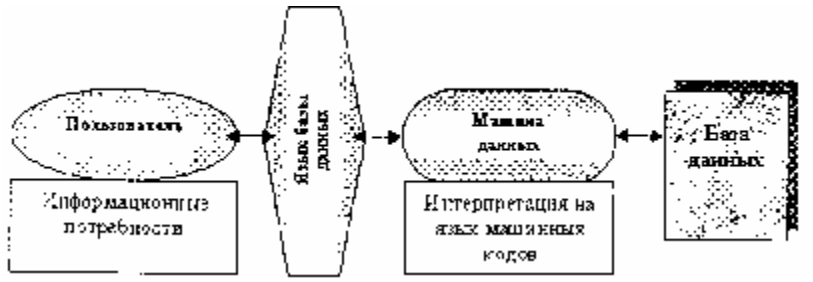


*Рис. 4.2. Схема взаимодействия пользователя с базой данных через язык баз данных*

В практику эти идеи впервые претворились в ходе реализации проекта *System R* (1975-1979 гг.) с участием еще одного известного специалиста по базам данных Криса Дейта. В ходе проекта System R был создан язык SEQUEL, трансформировавшийся впоследствии в ***язык структурированных запросов SQL*** (Structured Query Language).* При этом дополнительно к возможностям формирования «вопросов» к базе данных пользователю также решено было предоставить и возможность описания самой структуры данных, ввода данных и их изменения.

* Добавим также, что примерно в то же время в компании IBM был создан еще один реляционный язык—QBE (Query-By-Example), т. е. язык запросов по образцу, применявшийся впоследствии во многих коммерческих системах обработки табличных данных и послуживший идеологической основой для создания визуальных «конструкторов» запросов в современных СУБД.

Идеи языка SQL оказались настолько плодотворными, что он быстро завоевал популярность и стал широко внедряться в создаваемых в конце 70-х и в 80-х годах реляционных СУБД. Однако плодотворность идей языка SQL в отличие от первоначального замысла проявилась вовсе не в том, что на нем стали «разговаривать» с базами данных пользователи, не являющиеся профессиональными программистами. Язык SQL, в конечном счете, позволил, как уже отмечалось, *отделить низкоуровневые функции по организации структуры и обработке данных от высокоуровневых функций,* позволяя при создании и эксплуатации банков данных сосредоточиваться на смысловом, а не техническом аспекте работы с данными.

Быстрое и массовое распространение языка SQL в реляционных СУБД к середине 80-х годов привело фактически к принятию его в качестве ***стандарта*** по организации и обработке данных. В 1986 г. Американским национальным институтом стандартов (ANSI) и Международной организацией по стандартизации (ISO) язык был стандартизирован де-юре, т. е. признан стандартным языком описания и обработки данных в реляционных СУБД. В 1989 г. ANSI/ISO была принята усовершенствованная версия SQL — SQL2, а в 1992 г. третья версия — SQL3.

*Язык SQL* относится к так называемым ***декларативным*** (непроцедурным) языкам программирования. В отличие от процедурных языков (С, Паскаль, Фортран, Кобол, Бейсик) на нем формулируются предложения (инструкции) о том, «что сделать», но не «как сделать, как получить». *Машина данных* в СУБД исполняет роль *интерпретатора* и как раз строит машинный код, реализующий способ получения результата, задаваемого ***SQL-инструкциями.***

Язык SQL состоит из *двух* частей:

• ***языка описания (определения) данных — DDL*** (Data Definition Language);

• ***языка манипулирования данными — DML*** (Data Manipulation Language).

***Синтаксис SQL-инструкций*** включает:

• *название* инструкции (команду);

• *предложения,* определяющие источники, условия операции;

• *предикаты,* определяющие способы и режимы отбора записей, задаваемых предложениями;

• *выражения,* значения которых задают свойства и параметры выполнения инструкции и предложения.

Структуру SQL-инструкций можно разделить на две основные части, схематично представленные на рис. 4.3.*

* Квадратные скобки, как это общепринято, означают необязательность элемента

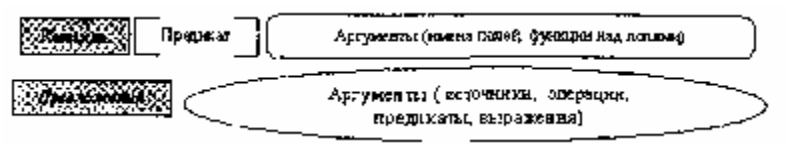



*Рис. 4.3. Структура SQL-инструкций.*

Первая часть включает название (команду) SQL-инструкции, предикат (необязательный элемент) и аргументы инструкции, которыми являются перечисляемые через запятую имена полей одной или нескольких таблиц.

Вторая часть состоит из одного или нескольких предложений, аргументы которых могут задавать источники данных (имена таблиц, операции над таблицами), способы, условия и режимы выполнения команды (предикаты сравнения, логические и математические выражения по значениям полей таблиц).

Перечень SQL-инструкций разделяется по частям языка SQL.

В состав языка DDL входят несколько базовых инструкций, обеспечивающих основной набор функций при *создании реляционных таблиц* и связей между ними.

*CREATETABLE...* — создать таблицу;

*CREATEINDEX...* — создать индекс;

*ALTERTABLE...* — изменить структуру ранее созданной таблицы;

*DROP...* — удалить существующую таблицу и базы данных.

В структуре инструкций *CREATETABLE* и *ALTERTABLE* важную роль играет предложение *CONSTRAINT* (создать ограничения на значения данных) со следующими установками — *NOT NULL* (не допускаются нулевые, точнее «пустые» значения по соответствующему полю, иначе говоря, определяется поле с обязательным заполнением), *AUTOINC* (поле с инкрементальным, т. е. последовательно возрастающим с каждой новой записью, характером значений) и *PRIMARY KEY* (определение для поля уникального, т. е. без повторов, индекса, что в результате задает режим заполнения данного поля с уникальными неповторяющимися по различным строкам значениями).

В состав языка DML также входят несколько базовых инструкций, охватывающих тем не менее основные операции *по вводу, обработке и выводу данных.*

*SELECT...* — выбрать данные из базы данных;

*INSERT...* — добавить данные в базу данных;

UPDATE... — обновить данные в базе данных;

*DELETE...* — удалить данные;

*GRANT...* — предоставить привилегии пользователю;

*REVOKE...* — отменить привилегии пользователю;

*COMMIT...* — зафиксировать текущую транзакцию;

*ROLLBACK...* — прервать текущую транзакцию.

Важное значение имеют разновидности инструкции *SELECT—SELECT... INTO ...* (выбрать из одной или нескольких таблиц набор записей, из которого создать новую таблицу) и *UNION SELECT,* которая в дополнении с исходной инструкцией *SELECT (SELECT... UNION SELECT...*) реализует операцию объединения таблиц.

Помимо предложения *CONSTRAINT* в SQL-инструкциях используются следующие *предложения:*

*FROM...* — указывает таблицы или запросы, которые содержат поля, перечисленные в инструкции *SELECT;*

*WHERE...* — определяет, какие записи из таблиц, перечисленных в предложении *FROM,* следует включить в результат выполнения инструкции *SELECT, UPDATE или DELETE;*

*GROUP BY...* — объединяет записи с одинаковыми значениями в указанном списке полей в одну запись;

*HAVING...* — определяет, какие сгруппированные записи отображаются при использовании инструкции *SELECT* с предложением *GROUP BY;*

*IN...* — определяет таблицы в любой внешней базе данных, с которой ядро СУБД может установить связь;

*ORDERBY...* — сортирует записи, полученные в результате запроса, в порядке возрастания или убывания на основе значений указанного поля или полей.

В качестве источника данных по предложению *FROM,* помимо таблиц и запросов, могут использоваться также результаты *операций соединения таблиц* в трех разновидностях—*INNER JOIN... ON..., LEFT JOIN. ..ON...* и *RIGHT JOIN...ON...* (внутреннее соединение, левое и правое внешнее соединение, соответственно*).

---

* Особенности разновидностей операций соединения рассматриваются в п. 4.3.2.1.2.

*Предикаты* используются для задания способов и режимов использования записей, отбираемых на основе условий в инструкции SQL. Такими предикатами являются:

*ALL...* — отбирает все записи, соответствующие условиям, заданным в инструкции SQL, используется по умолчанию;

*DISTINCT...* — исключает записи, которые содержат повторяющиеся значения в выбранных полях;

*DISTINCTROW...* — опускает данные, основанные на целиком повторяющихся записях, а не на отдельных повторяющихся полях;

*TOPn...* — возвращает *n* записей, находящихся в начале или в конце диапазона, описанного с помощью *предложения ORDER BY*;

***Выражениями*** в инструкциях SQL являются любые комбинации операторов, констант, значений текстовых констант, функций, имен полей, построенные по правилам математических выражений и результатом которых является конкретное, в том числе и логическое значение.

Язык SQL, конечно же, с точки зрения профессиональных программистов построен довольно просто, но, вместе с тем, как уже отмечалось, надежды на то, что на нем станут общаться с базами данных пользователи-непрограммисты, не оправдались. Причина этого, вероятно, заключается в том, что, несмотря на простоту, язык SQL все же является формализованным искусственным языком, осваивание и использование которого в большинстве случаев тяготит конечных пользователей. Исследования основ и способов интерфейса человека с компьютером, эргономических и психологических основ работы с компьютерной информацией, проведенные в конце 70-х и в 80-х годах, показали, что пользователи-специалисты в конкретных предметных областях (а не в области вычислительной техники и программирования) более склонны к диалогово-визуальным формам работы с вычислительными системами и компьютерной информацией.

Поэтому с конца 80-х годов в развитии СУБД наметились две тенденции:

• СУБД для конечных пользователей;

• СУБД для программистов (профессионалов).

В ***СУБД для конечных пользователей*** имеется развитый набор диалоговых и визуально-наглядных средств работы с базой данных в виде специальных диалоговых интерфейсов и пошаговых ***«мастеров»,*** которые «ведут» пользователя по пути выражения им своих потребностей в обработке данных. Например, при создании новой таблицы соответствующий «мастер» последовательно запрашивает у пользователя имя таблицы, имена, типы и другие параметры полей, индексов и т. д. При этом интерфейсная часть СУБД формирует для ядра СУБД (машины данных) соответствующую и порой весьма сложную инструкцию SQL.

В ***профессиональных СУБД*** язык базы данных (SQL) дополняется элементами, присущими процедурным языкам программирования — описателями и средствами работы с различного типа переменными, операторами, функциями, процедурами и т. д. В результате формируется специализированный на работу с данными декларативно-процедурный язык высокого уровня, который встроен в СУБД (точнее надстроен над ядром СУБД). Такие языки называют *«включающими»* (см. рис. 2.1). На основе включающего языка разрабатываются полностью автономные прикладные информационные системы, реализующие более простой и понятный для специалистов в определенной предметной области (скажем, в бухгалтерии) интерфейс работы с информацией.

С учетом этапов в развитии программных средств СУБД такие языки получили название языков ***четвертого поколения — 4GL*** (Forth Generation Language). Языки 4GL могут быть непосредственно встроены в сами СУБД, а могут существовать в виде *отдельных сред программирования.* В последнем случае в таких средах разрабатываются *прикладные части* информационных систем, реализующие только интерфейс и высокоуровневые функции по обработке данных. За низкоуровневым, как говорят, «сервисом» к данным такие прикладные системы обращаются к ***SQL-серверам,*** являющимися отдельными специализированными разновидностями СУБД. «Общение» между прикладными системами и SQL-серверами происходит соответственно на языке SQL.

Свои языки 4GL имеют практически все развитые профессиональные СУБД—Orac/e, SyBase, Informix, Ingres, DB2, отечественная СУБД ЛИНТЕР. Распространенными отдельными средами программирования для создания информационных систем в настоящее время являются системы Visual Basic фирмы Microsoft и Delphi фирмы Borland Intemational. Кроме того, уже упоминавшиеся CASE-средства автоматизированного проектирования — PowerBuilder фирмы PowerSoft, Oracle Designer фирмы Oracle, SQLWindows фирмы Gupta и др., также, как правило, имеют свои встроенные языки 4GL.

В заключение следует отметить, что в последнее время наметилась тенденция встраивания развитых языков уровня 4GL и в СУБД для конечных пользователей. В качестве примера можно привести СУБД Access фирмы Microsoft, имеющей один из наиболее развитых интерфейсов по созданию и работе с базами данных для конечных пользователей, и в то же время оснащенной встроенным языком уровня 4GL — VBA (Visual Basic for Application), являющегося диалектом языка Visual Basic.

## 4.2. Ввод, загрузка и редактирование данных

Базы данных создаются для удовлетворения информационных потребностей пользователей. Однако для выполнения этой главной задачи базу данных необходимо не только правильно организовать и построить, но и *наполнить* самими *данными.* Как правило, эта задача требует больших затрат особенно на начальном этапе ввода информационных систем в эксплуатацию. Поэтому способам, удобству ввода и редактирования данных в СУБД всегда придавалось и придается важное значение.

### 4.2.1. Ввод и редактирование данных в реляционных СУБД

В настоящее время можно выделить *четыре основных способа* ввода, загрузки и редактирования данных в современных реляционных СУБД:

- непосредственный ввод и редактирование данных в табличном режиме;
- ввод и редактирование данных через формы;*
- ввод, загрузка и редактирование данных через запросы на изменения;
- ввод данных через ***импорт*** из внешних источников.

* Если формы предназначены только для ввода, просмотра и изменения данных их еще называют входными (вводными) формами.

Ввод данных в ***табличном режиме*** и ***через формы*** является наиболее естественным с точки зрения табличного характера организации данных в реляционных СУБД. Как отмечалось при рассмотрении реляционной модели организации данных, единичным элементом информации, имеющим отдельное смысловое значение, является кортеж, т. е. табличная строка-запись, состоящая из дискретного набора значений по полям таблицы. Иначе говоря, данные в реляционные базы вводятся или удаляются кортежами-записями.

*Отображать кортежи-записи* можно *двумя способами,* располагая поля записи *вертикально* или *горизонтально* (см. рис. 4.4 и рис. 4.5).

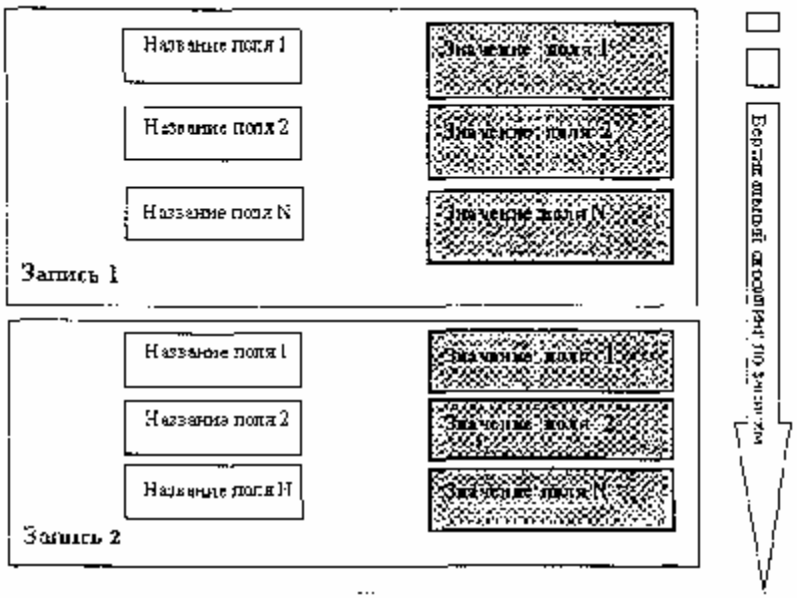



*Рис. 4.4. Вертикальный способ расположения полей записей*

В первом случае пользователь «видит» и имеет доступ, как правило, сразу ко всем полям одной записи и его внимание сосредоточивается на одной записи как отдельном объекте. Записи располагаются друг за другом вертикально, и на экране компьютера обеспечивается вертикальная прокрутка (скроллинг) записей.

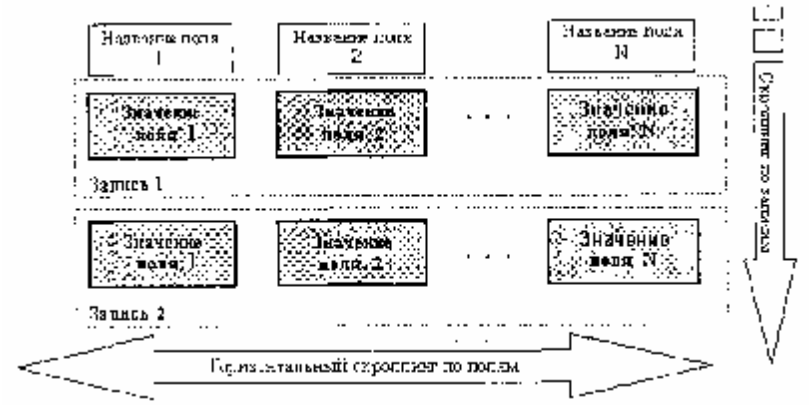



*Рис. 4.5. Горизонтальный способ расположения полей записей*

Во втором способе каждая запись отображается в виде табличной строки и на экране может отображаться не одна, а несколько строк, что дает возможность пользователю производить анализ и просмотр сразу группы записей. Вместе с тем при достаточно большом количестве полей (столбцов) таблицы все они могут не уместиться на экране по горизонтальным строкам. В этом случае пользователь видит сразу не всю запись (записи), а лишь некоторый вертикальный фрагмент, и восприятие записи как отдельного объекта несколько расплывается. Для просмотра всех полей организуется их горизонтальная прокрутка. С учетом того что, как правило, ключевые поля располагаются первыми в строках-записях, при использовании горизонтальной прокрутки происходит «отрыв» ключевой информации, идентифицирующей каждую конкретную строку, от информации по другим полям записи.

Таким образом, и тот и другой способ отображения записей имеет свои преимущества и недостатки. Практически все реляционные СУБД предоставляют возможность работы с данными и тем и другим способом.

Реализация непосредственного ввода данных в табличном режиме или через формы осуществляется через ***«открытие»*** соответствующей ***таблицы*** базы данных. При открытии таблицы страницы файла данных, содержащие просматриваемые записи таблицы, помещаются в буферы оперативной памяти и отображаются в том или ином режиме. Непосредственный ввод и корректировка данных при этом осуществляются через использование ***табличного курсора,*** позаимствованного из технологии работы в табличных редакторах.

В табличном режиме табличный курсор может свободно перемещаться по ячейкам таблицы, определяя в каждый момент так называемую *текущую строку* и *текущую ячейку.* Вводимые с клавиатуры данные автоматически помещаются в текущую ячейку, т. е. имитируется работа с таблицами в табличных редакторах. Вместе с тем по сравнению с табличными редакторами имеется все же одно принципиальное отличие. Единичным элементом ввода данных в СУБД, как уже отмечалось, является кортеж-запись, т. е. табличная строка целиком, а не отдельно взятая ячейка. Поэтому СУБД в режиме открытой таблицы явно (через специальные команды) или неявно (при перемещении табличного курсора на другую строку) осуществляет фиксацию изменений в существующей строке или фиксирует новую строку в файле базы данных, т. е. фиксирует соответствующую транзакцию. При этом проверяется соответствие введенных или откорректированных данных установленным типам полей, уникальность значений ключевых полей, выполнение других ограничений целостности данных. Если обнаруживается какое-либо несоответствие, то отвергается фиксация сразу всей строки, а не конкретной ячейки.

Ввод новой записи осуществляется через активизацию в конце таблицы специальной «пустой» строки открытой таблицы. В некоторых случаях таблицы могут открываться только для ввода новых данных—так называемый *режим открытия на добавление.* В этом случае в открытой таблице показывается только одна «пустая» строка для ввода новых данных.

*Вертикальный способ отображения полей записей* в современных СУБД вместе с идеями электронных бланков трансформировался в ***технику форм.*** Естественным и интуитивно-понятным способом работы со структурированной информацией для большинства «обычных»* людей являются всевозможные бланки, анкеты и т.п. «бумажные» формы. Формы в СУБД как раз и выполняют функции предоставления пользователям привычного интерфейса при вводе структурированных данных с имитацией «заполнения» бланков, анкет и т.п.

* То есть в данном контексте не являющихся профессиональными программистами.

Таким образом, *форма* в СУБД представляет собой *специальный экранный объект, включающий поля для ввода данных одной записи базовой таблицы и другую поясняющую информацию.* На рис. 4.6 приведен пример формы для ввода, просмотра и изменения данных в таблице «Сотрудники» базы данных известной организации.

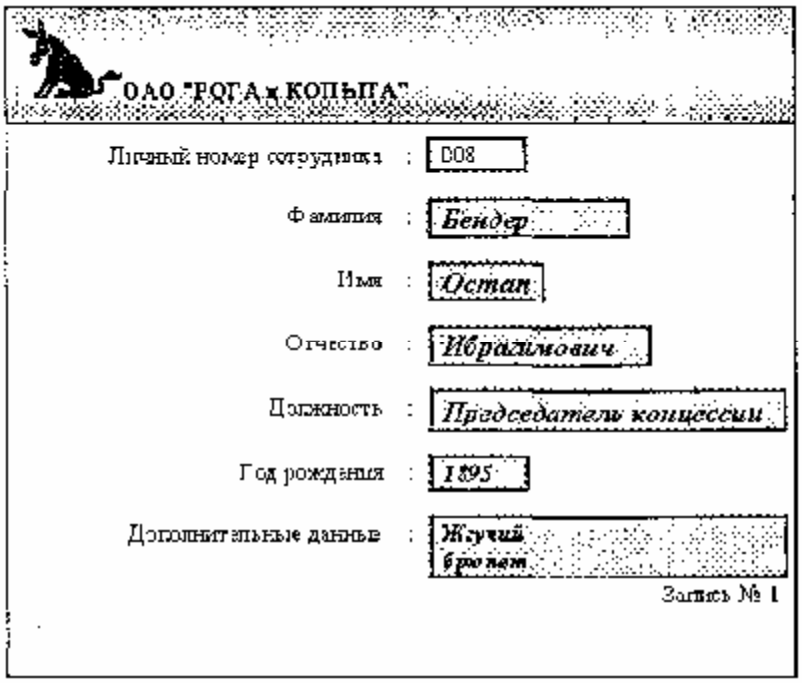



*Рис. 4.6. Пример формы для ввода/просмотра/изменения данных*

Записи соответствующей таблицы через форму *«прокручиваются»* по вертикали. Присутствует также аналог табличного курсора, определяющий текущее поле для ввода/изменения данных. Так же как и в табличном режиме, форма может открываться только на ввод новых данных, т.е. в *режиме добавления* без возможности просмотра ранее введенных в таблицу данных.

Форма может отображать записи или предоставлять возможность для ввода новых записей одной *(базовой)* таблицы. Однако идея экранных форм в реляционных СУБД оказалась более плодотворной, чем просто предоставление удобств для ввода/просмотра сразу всех полей одной записи. В определенных случаях «бумажная» информационная технология, которую автоматизирует банк данных АИС, предусматривает накопление и образование данных сразу в комплексе по ряду информационных задач. К примеру, при ведении учета командировок сотрудников в бухгалтерии используются специальные бланки, в которых отображается информация по сотруднику (ФИО, Подразделение, Должность, Сл. тел.) и данные собственно по командировке (Дата начала. Дата окончания, Полученный аванс, Фактические расходы, Пункты назначения, Служебные задания). При проектировании базы данных для автоматизации такого учета, исходя из требований нормализации таблиц, перечисленные выше в «бумажном» бланке данные распределяются сразу по нескольким связанным таблицам «Сотрудник», «Командировка», «Пункты», «Служебные задания».

Техника форм СУБД предоставляет возможность создавать *«комплексные»* электронные бланки для *ввода* информации сразу *в несколько связанных таблиц.* Такие формы называются *структурными* (сложными) и обеспечивают естественный в технологическом плане совместный ввод данных в связанные таблицы. Чаще всего такой способ позволяет вводить и просматривать записи, находящиеся в таблицах, связанных отношением «Один-ко-многим». При прокрутке записей в главной форме, отражающей записи базовой таблицы на стороне «один», в структурных элементах, называемых иногда «подчиненными» формами, прокручиваются связанные записи из таблиц на стороне «многие». На рис. 4.7 приведен пример такой формы.

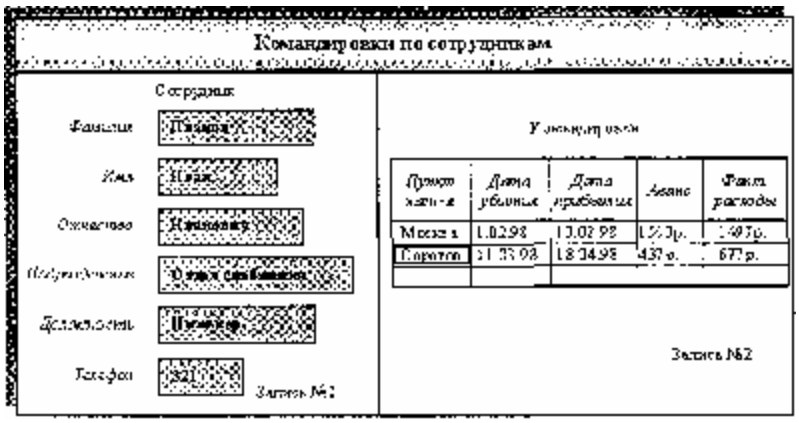



*Рис. 4.7. Пример формы для ввода данных в таблицы, связанные отношением «Один-ко-многм»*

Зачастую при создании базы данных новой АИС часть данных уже имеется в электронном виде в других, ранее созданных базах данных. Если это базы данных того же формата, т. е. созданные и функционирующие под управлением той же СУБД, или другой реляционной СУБД, поддерживающей основанный на языке SQL специальный протокол обмена данными между реляционными СУБД — ODBC* (Open Database Connectivity), то имеется возможность вводить, или, как в этом случае говорят, загружать данные из таблиц, находящихся в файлах других (внешних) баз данных. Такая загрузка реализуется на основе техники запросов на изменение данных, в качестве источников которых указываются таблицы в других БД.**

* Стандартный протокол доступа к данным на серверах баз данных SQL...

** Запросы на изменения данных рассматриваются в п. 4.3.2.2.

Ряд СУБД предоставляет возможность загружать табличные данные, созданные и находящиеся под управлением не СУБД, а приложений другого типа — табличных и текстовых редакторов. В этом случае говорят об ***импорте данных из внешних источников.*** Ввод данных при этом осуществляется на основе «знания» СУБД формата внешних табличных данных и соответствующей их трансформации в структуры реляционных таблиц. Некоторые СУБД предоставляют возможность ввода в реляционные таблицы текстовых данных, размеченных специальными разделителями на последовательно расположенные дискретные порции. Каждая такая порция помещается в соответствующее поле табличной строки по принципу последовательного заполнения строк таблицы *«Слева-направо, Сверху-вниз».* При этом СУБД проверяет соответствие вводимых значений установленным типам полей, а также другим параметрам полей и ограничениям целостности данных.

Таким образом, в современных реляционных СУБД имеется развитый арсенал возможностей по вводу и загрузке данных, который позволяет эффективно решать задачи по наполнению БД данными.

### 4.2.2. Особенности ввода и загрузки данных в СУБД с сетевой моделью организации данных

Характерной особенностью *логической* и *физической структуры* данных в сетевых СУБД, как уже отмечалось, является хранение информации по связям между информационными объектами как отдельных самостоятельных объектов — наборов экземпляров связей. В результате более сложная, чем в реляционных СУБД структура данных, определяет использование в сетевых СУБД преимущественно нелинейных структур физической организации данных, что обеспечивает более эффективный доступ к данным, но вместе с тем вызывает повышенные затраты на изменения (добавление, удаление, редактирование) данных. Любые процессы *модификации* данных в большинстве случаев приводят к необходимости «перетряски» всего банка данных, что чрезвычайно *замедляет* ввод и редактирование данных.

Поэтому в СУБД с сетевой организацией данных непосредственный ввод данных (ручной ввод с клавиатуры) чаще всего построен по принципу ***«стакана и емкости».*** Сначала данные вводятся в так называемое ***«входное сообщение».*** Специальный интерфейс в таких СУБД предоставляет пользователю возможности ввода данных во входное сообщение по аналогии с непосредственным вводом данных в реляционных СУБД — табличном режиме или в режиме форм. При этом данные физически размещаются во *временном файле* — стакане. Кроме того, в отличие от реляционных СУБД, пользователь может вводить данные сразу в различные (связанные) таблицы, непосредственно «переходя» по полям-отсылкам из одного объекта (таблицы) в другой.

После подготовки файла входного сообщения, который размещается, так же как и файл базы данных, на внешней дисковой памяти, осуществляется загрузка входного сообщения в базу данных, т.е. «стакан» выливается в «емкость».* СУБД при этом «перетасовывает» всю базу данных, вставляя новые, изменяя месторасположение «старых» записей, модифицируя физические адреса отсылок и т.д.

* Отсюда, собственно, исторически и возникли различия в понимании терминов «ввод» и «загрузка».

Дополнительной проблемой при этом является *отождествление* новых и старых записей. В реляционных СУБД эта проблема решается исключительно на основе уникальности значений ключевых полей. Новая запись с уже существующим значением ключевого поля в реляционных СУБД автоматически отвергается и данные по ней можно ввести только путем корректировки полей уже существующей записи с соответствующим значением ключевого поля.*

* В этом, кстати, проявляется один существенный недостаток реляционных СУБД—отсутствие возможностей темпорального отображения данных, т.е. отображения данных с предысторией их изменения.

В сетевых СУБД, ввиду поддержки полей с множественным характером значений, обеспечивается *«слияние»* данных. При слиянии записей из входного сообщения с уже существующими записями в базе данных поля с множественным типом значений объединяются, а по полям с единичным характером значений устанавливаются дифференцированный режим замены или отвержения нового значения.

## 4.3. Обработка данных

Обработка данных представляет собой емкое понятие, включающее широкий набор различных функций и операций по удовлетворению информационных потребностей пользователя. Тем не менее в технологическом плане этот широкий набор удобно разделить на *три* группы:

- поиск, фильтрация и сортировка данных;
- запросы к базе данных;
- механизм реализации событий, правил (триггеров) и процедур в базе данных.

На практике работа пользователя с базой данных может включать сразу весь комплекс операций, однако с методической точки зрения целесообразно рассмотреть их последовательно.

### 4.3.1. Поиск, фильтрация и сортировка данных

Операции по поиску, фильтрации и сортировке данных реализуют самые *простые информационно-справочные потребности* пользователей, но являются, вероятно, наиболее частыми при работе с базами данных.

Отличительная особенность операций по поиску, фильтрации и сортировке данных заключается в том, что они осуществляются *в режиме открытой таблицы* или *формы.* Забегая несколько вперед, следует отметить главную отличительную особенность этих операций по сравнению с запросами на выборку к базам данных — *результатом* операций по поиску или фильтрации данных является изменение *состояния просмотра* открытой таблицы (формы), но не самих данных, которые физически остаются в той же таблице и в том же порядке. Например, результатом поиска какой-либо конкретной записи в открытой таблице является установление табличного курсора на ключевое поле искомой записи-строки или «показ» (отображение) в открытой форме полей искомой записи.

Собственно ***поиск*** данных реализуется в виде:

- поиска записи по ее номеру;
- поиска записи (записей) по значению (значениям) какого-либо поля;
- поиска записей с помощью фильтров (фильтрация).

*Поиск записи по ее номеру* производится на основе *механизма распределения записей по страницам* файла данных. Результатом такого поиска является перевод табличного курсора в ключевое поле соответствующей записи или «показ» в форме полей искомой записи.

*Поиск записи по значению поля* осуществляется также на основе механизма распределения записей по страницам файла данных с использованием техники *вхождения образца* в значения просматриваемого поля. Результатом поиска является установка табличного курсора в соответствующее поле найденной записи. Если записей с искомым значением выделенного поля

несколько, тогда, как правило, реализуется последовательная «остановка» (последовательный просмотр) табличного курсора в соответствующих полях найденных записей.

***Фильтр*** представляет собой набор условий, применяемых для отбора подмножества записей. *Результатом* фильтрации является «показ» (отображение) в открытой таблице или форме только отфильтрованных записей с временным «скрытием» всех остальных записей. При этом остальные записи физически никуда не перемещаются, не удаляются и вновь отображаются в открытой таблице после «снятия» фильтра.

Набор условий, определяющих фильтр, формируется в различных СУБД по-разному, но общепринятым является использование ***выражений*** *в условиях отбора данных.* Под выражением в данном случае понимается структура, подобная обычному математическому выражению. Аргументами выражения могут быть числа, даты, текст, имена полей, которые соединяются знаками математических операций, неравенств (+, -, *, /, >, <, =) и логических операций (AND, OR, NOT). При этом текстовые значения и аргументы заключаются в кавычки («Иванов»), даты в символы # (#01.01.98#).

Как отмечалось при рассмотрении реляционной модели данных, строки в таблицах формируются и хранятся в неупорядоченном виде. Вместе с тем одной из простых, но частых информационных потребностей пользователя при работе с базой данных является как раз *упорядочение записей по возрастанию/убыванию* или *по алфавиту* по определенному полю (например, по полям дат, по полям с размерами должностных окладов, по полю с фамилией сотрудников и т. п.). Такие процедуры реализуются ***сортировкой*** данных, которая упорядочивает последовательность расположения строк открытой таблицы по значениям какого-либо поля. При этом в файле базы данных строки таблицы физически остаются не упорядоченными. Иначе говоря, сортировка строк открытой таблицы происходит только в буферах страниц в оперативной памяти. Новый порядок расположения строк таблицы (т. е. их размещение по страницам файла БД) может быть зафиксирован специальной командой при закрытии таблицы.

При больших объемах таблиц (при большом количестве строк-записей) операции сортировки могут занимать продолжительное время, которое существенно сокращается, если сортировка осуществляется по *индексированному полю.* В этом плане опыт эксплуатации базы данных может привести к уточнению списка индексированных полей в таблицах.

### 4.3.2. Запросы в реляционных СУБД

Запросы являются наиболее распространенным видом обработки данных при решении пользователями АИС *тематических, логических, статистических и технологических информационных задач.* Иначе говоря, для удовлетворения сложных информационных потребностей пользователи «общаются» с базой данных через запросы.

***Запрос*** представляет собой *спецификацию (предписание) на специальном языке (языке базы данных) для обработки данных.* В реляционных СУБД запросы к базе данных выражаются, соответственно, на языке SQL.

*Формирование* запросов в СУБД может осуществляться в *специальном редакторе* (командный режим) или через *наглядно-диалоговые средства (конструкторы)* и *пошаговые мастера* формирования запросов. Сформированный запрос в виде SQL-инструкции сохраняется в файле базы данных и впоследствии специальной командой СУБД может запускаться (открываться) на выполнение.

Все многообразие запросов можно проклассифицировать схемой, приведенной на рис. 4.8. Сточки зрения решаемых информационных задач и формы результатов исполнения запросов их можно разделить на три группы:

- запросы на выборку данных;
- запросы на изменение данных;
- управляющие запросы.

#### 4.3.2.1. Запросы на выборку данных

Запросы на выборку применяются для решения *тематических, логических и статистических информационных задач* и относятся к одному из наиболее часто применяемых видов запросов. Данный вид запросов реализуется SQL-инструкцией *SELECT* с предложением *FROM.*

*Результатом* исполнения запроса на выборку является ***набор данных,*** который представляет *временную таблицу* данных со структурой (поля, их типы и параметры), определяемой параметрами запроса и параметрами полей таблиц, из которых выбираются данные. В отличие от режимов поиска

и фильтрации запросами на выборку данные выбираются из «не открытых» таблиц базы данных. Результаты запросов на выборку помещаются в специальную временную таблицу, размещаемую на период исполнения («открытия») запроса в оперативной памяти. В этом смысле с точки зрения дальнейшей обработки данных запрос (как результат) в реляционных СУБД *тождественен* просто таблице данных, «открытие» которой осуществляется в результате выполнения запроса. Из этого следует возможность исполнения запросов над запросами, точнее над результатами исполнения других запросов, что существенно облегчает построение сложных запросов при решении логических и статистических информационных задач.

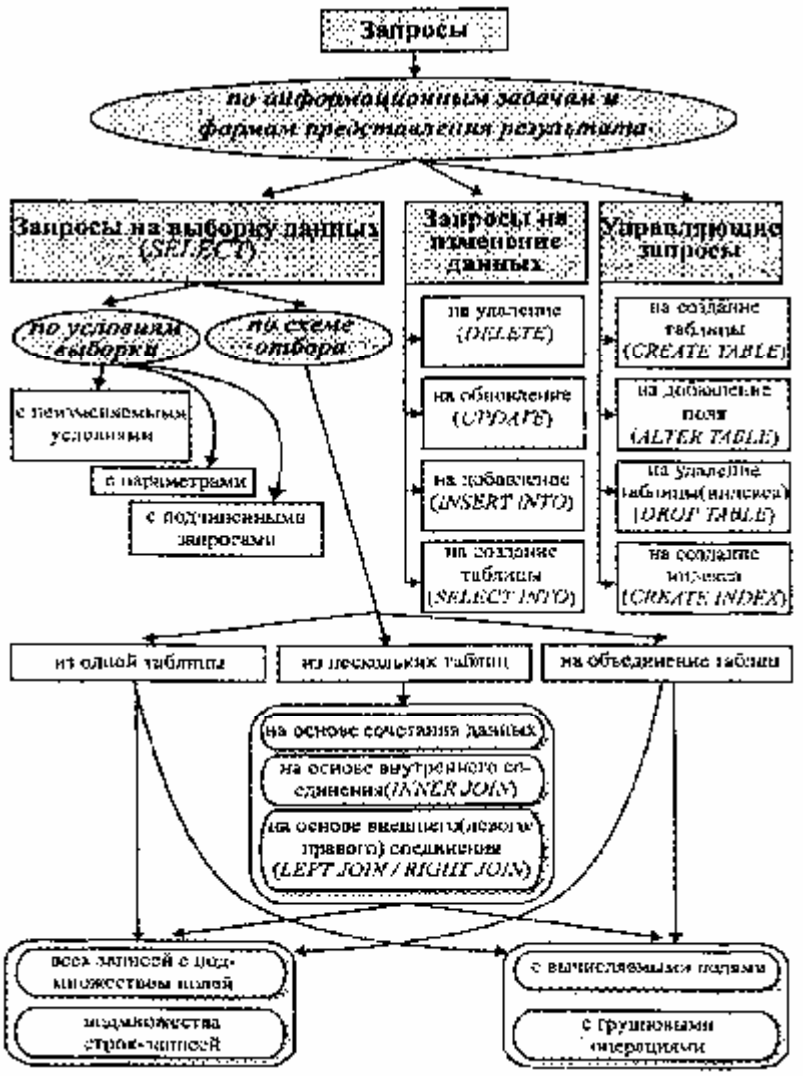



*Рис. 4.8. Классификация запросов в реляционных СУБД*

В большинстве СУБД наборы данных, формируемые запросами на выборку, являются ***динамическими.*** Динамичность означает, что с результатом исполнения запроса можно производить все те же операции, что и с данными в режиме открытой таблицы. Иначе говоря, *изменения* данных, осуществляемые в наборе данных, сформированных по запросу, *фиксируются* в исходных таблицах, из которых выбираются данные, и, наоборот, — изменения данных в исходных таблицах, если они производятся в открытых таблицах после исполнения запроса, отображаются в наборе данных по результатам «открытого», т. е. исполняемого в это же время, запроса.

Запросы *на выборку классифицируются* по *двум* критериям — по формированию условий выборки и по схеме отбора данных.

По *формированию условий выборки* запросы можно подразделить на *три* группы:

• запросы ***со статическими*** (неизменяемыми) условиями отбора;

• запросы с параметрами;

• запросы с подчиненными запросами.

В запросах первого вида *условия выборки* данных определяются при формировании самого запроса и являются *неизменными* при всех последующих выполнениях запроса. В ***запросы с параметрами*** вставляются специальные средства для *диалогового задания* пользователем конкретных *параметров в условиях отбора* в *момент исполнения* запроса. Таким образом, при запуске на исполнение запроса

с параметрами пользователь может варьировать и уточнять условия выборки данных. В запросах третьей группы условия отбора данных определяются по результатам исполнения вставленной в тело внешнего запроса внутренней инструкции *SELECT*.

По *схеме отбора* данных запросы на выборку подразделяются также на *три* группы:

- запросы на выборку данных из одной таблицы;
- запросы на выборку данных в один набор из нескольких таблиц;
- запросы на объединение данных.

### 4.3.2.1.1. Запросы на выборку данных из одной таблицы

Запросы на выборку данных из одной таблицы по смыслу и назначению *сходны с фильтрацией данных* в открытой таблице. Различие заключается лишь в форме представления результата (в частности, запросом на выборку можно отображать не просто подмножество записей исходной таблицы, но и *подмножество полей* исходной таблицы) и в технологии последующей работы с результатом (над набором данных, как уже отмечалось, можно исполнить другой запрос).

Различают запросы *на выборку всех записей с произвольным набором полей* и запросы *на выборку подмножества записей.*

На рис. 4.9 приведен пример запроса, формирующего полный список сотрудников организации из таблицы «Сотрудники», но с сокращенным набором полей («Таб. №», «Фамилия», «Имя», «Отчество»), а также представлен вариант SQL-инструкции, реализующий данный запрос.

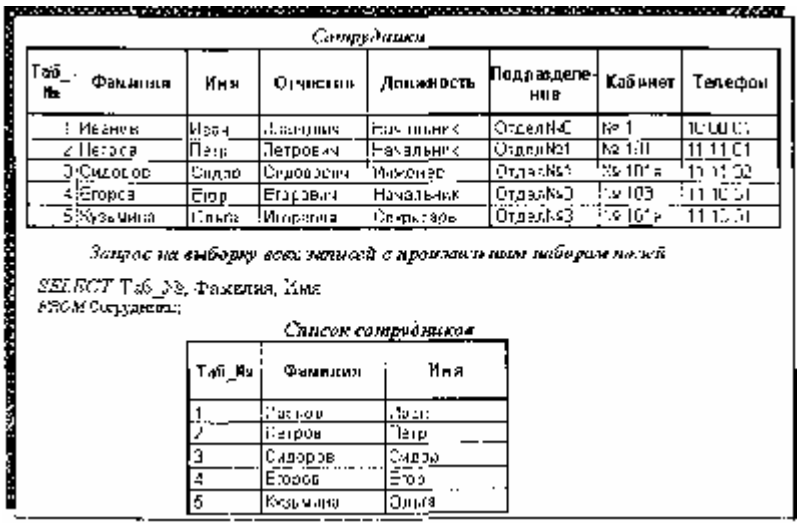



*Рис 4.9. Пример запроса на выборку всех записей по группе полей*

В запросах на *отбор подмножества записей* в SQL-инструкции *SELECT через* предложение *WHERE* помещается выражение, определяющее условие отбора данных. На рис. 4.10 приведен пример реализации запроса на отбор подмножества записей из таблицы «Сотрудники» для сформирования списка работников инженерно-технического и экономического профиля.

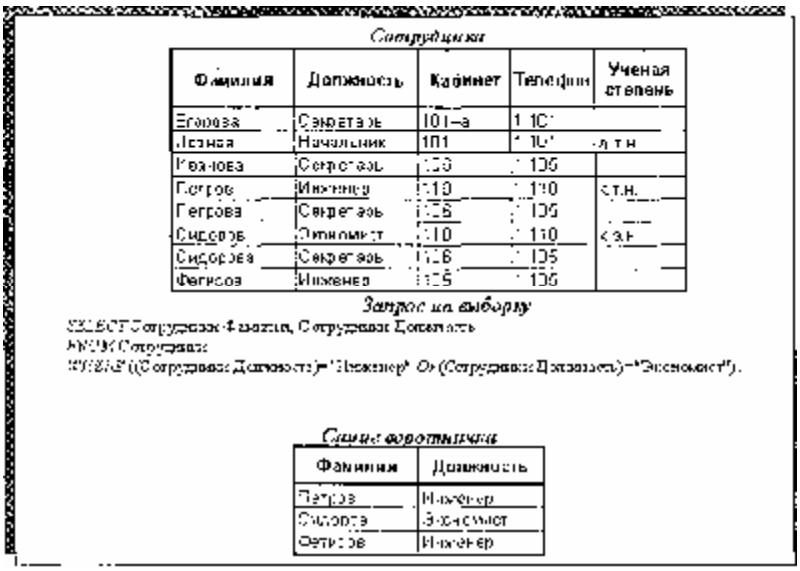


*Рис. 4.10. Пример запроса на выборку подмножества записей*

В запросах на выборку данных широко применяются *предикаты отбора ALL, DISTINCT, DISTINCTROW* и *TOPn.* Предикат *ALL* используется по умолчанию и устанавливает вывод в наборе данных всех записей, формируемых по условию отбора, задаваемого предложением *WHERE,* и в большинстве случаев в инструкции *SELECT* ***опускается***.

Предикат *DISTINCT* используется для исключения в наборе отбираемых данных тех записей, значения которых по определенному полю повторяются, т. е. уже раз вошли в набор.

На рис. 4.11 приведен пример запроса, отбирающего из таблицы «Сотрудники» данные по полю «Должность» без предиката отбора (т.е. с предикатом *ALL)* и с предикатом *DISTINCT.* В данном случае использование предиката *DISTINCT* позволяет сформировать простой список должностей без повторов.

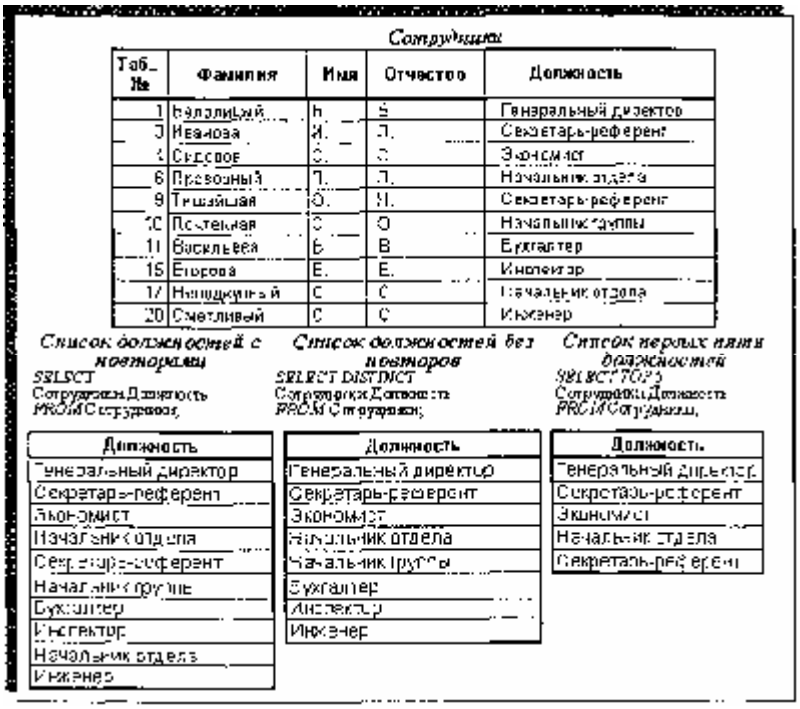



*Рис. 4.11. Пример запросов с предикатами ALL, DISTINCT* и *TOPn.*

Предикат *DISTINCTROW* имеет аналогичное предикату *DISTINCT* назначение для исключения из набора тех записей, значения которых повторяются по всем полям, включенным в набор данных.

Предикат *TOP n* обеспечивает включение в набор данных первых *n* записей, сформированных по условию отбора. Пример запроса с предикатом *TOPn* также приведен на рис. 4.11.

В запросах на выборку помимо предложений *FROM* и *WHERE* используются предложения *GROUP BY, HAVING* и *ORDER BY* для дополнительной обработки отбираемых записей.

Предложение *GROUP BY* объединяет (группирует) записи с одинаковыми значениями определенных полей в одну запись. Предложение *HAVING* выполняет функцию предложения *WHERE,* позволяя задавать дополнительные условия для отбора сгруппированных предложением *GROUP BY* записей. Предложение *ORDER BY* обеспечивает сортировку отобранных записей в зависимости от способа *ASC* (по возрастанию) или *DESC* (по убыванию). На рис. 4.12 приведен пример запроса, формирующего в порядке убывания список сгруппированных по полям «Категория» и «Профиль» записей из таблицы «Подразделения» при условии отбора подразделений с категорий выше третьей и отбора сгруппированных записей при условии основного профиля подразделений.

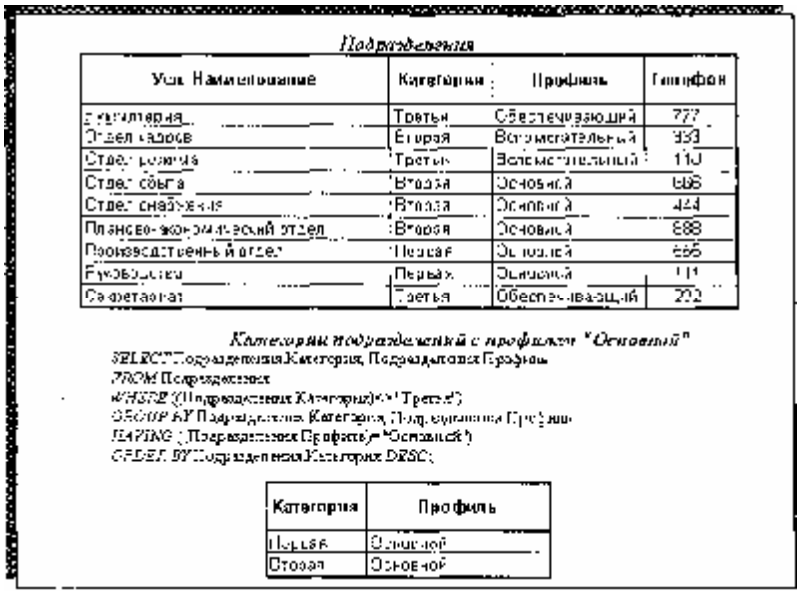



*Рис. 4.12. Пример запроса на выборку данных с предложениями GROUP BY, HAVING* и *ORDER BY*

Определенную специфику имеет отбор записей с ***«пустыми»*** значениями определенных полей. В трактовке реляционных СУБД и языка SQL «пустых», т.е. неопределенных, значений полей не бывает. Иначе говоря, значением числового поля может быть число, равное «0», а значением других типов полей (текстовые, дата) может быть нулевое значение— «Null».

Отбор записей с пустыми значениями может применяться для решения некоторых тематических и технологических задач, когда нужно отдельно сформировать и проанализировать набор данных с записями, содержащими нулевые для числовых полей, или не имеющие в силу каких-либо причин определенного значения, для других типов полей. На рис. 4.13 представлен пример запроса, отбирающего данные из таблицы «Сотрудники», приведенной на рис. 4.10, с «пустыми» значениями по полю «Ученая степень», иначе говоря, формирующий список сотрудников, не имеющих ученых степеней.

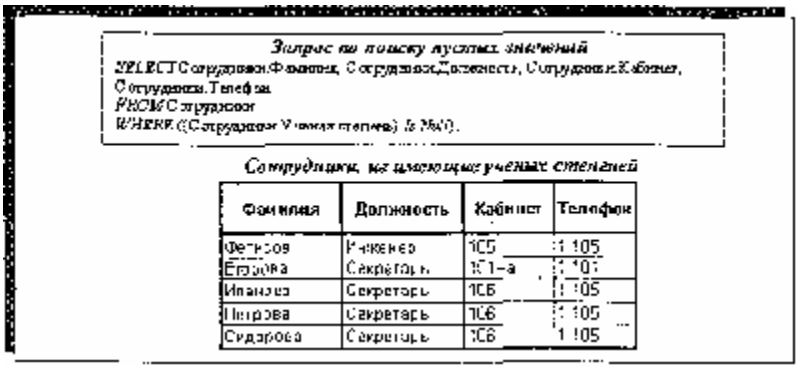



*Рис. 4.13. Пример запроса по поиску данных с пустыми значениями определенного поля*

### 4.3.2.1.2. Запросы на выборку данных из нескольких таблиц

Запросы на выборку данных из нескольких таблиц, как правило, предназначены для решения *логических* информационных задач и, в свою очередь, подразделяются на *три* группы:

- запросы на сочетание данных;
- запросы на соединение данных;
- запросы на объединение данных.

***Запросы на сочетание*** строятся на основе операции *скалярного произведения реляционных таблиц* и по смыслу направлены на *формирование полного набора сочетании строк-записей,* представленных в исходных таблицах. Запросы на сочетание строятся на основе SQL-инструкции *SELECT* и предложения *FROM* с простым перечислением отбираемых полей и их таблиц.

Для примера на рис. 4.14 приведен запрос на выборку сочетания данных из таблицы «Подразделения» и таблицы «Мероприятия». Формирование и исполнение такого запроса может быть обусловлено потребностями автоматического формирования новой таблицы для составления определенных

планов или графиков, где нужно предусмотреть в исходном виде полный набор сочетаний данных по подразделениям и по мероприятиям.

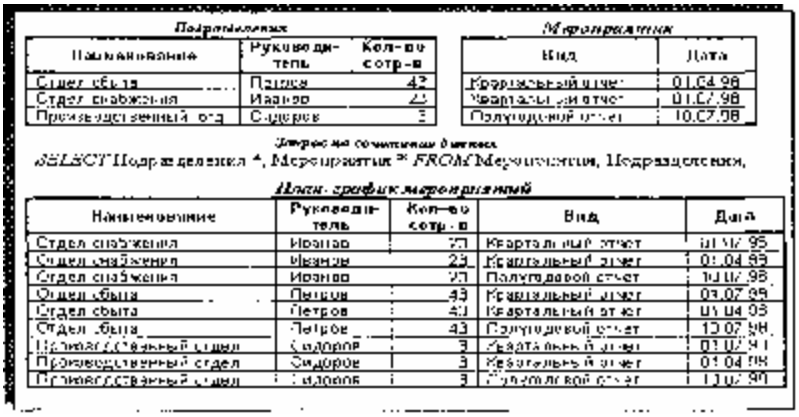



*Рис. 4.14. Пример реализации запроса на сочетание дачных из двух таблиц*

***Запросы на соединение,**** в свою очередь, подразделяются на запросы на основе внутреннего соединения ***(INNER JOIN)*** и запросы на основе правого или левого внешнего соединения ***(RIGHT JOIN*** и ***LEFT JOIN).***

* В некоторых источниках данный тип запросов называют запросами на объединение (*JOIN*). Англ. термин *JOIN* переводится в глагольном виде как «объединяться», «соединяться», что и обусловливает неодинаковое его использование в русском переводе в разных источниках. В данном контексте более правильным является его перевод как «соединение», так как в реляционной модели данных операции «объединения» и «соединения» различны.

Запросы на выборку, строящиеся на *основе внутреннего соединения,* реализуют рассматриваемую по реляционной модели данных *операцию соединения реляционных таблиц.* Данная операция является одной из наиболее характерных и частых при решении *логических информационных задач,* когда нужно получить и просмотреть данные из разных таблиц, *связанных определенной логикой или предварительно установленными в схеме базы данных связями.* Напомним, что при реализации операции соединения двух таблиц выделяется *поле соединения,* которое должно быть одинакового типа в соединяемых таблицах. Результатом соединения таблиц является новая таблица, содержащая все поля, или часть полей первой таблицы и все или часть полей второй таблицы. Строки итоговой таблицы при внутреннем соединении образуются из сцепления строк первой и второй таблиц, когда их значения по соединяемому полю совпадают.

Запросы на внешнее соединение строятся на основе модификации операции соединения. *При левом внешнем соединении (LEFT JOIN)* строки итоговой таблицы образуются из всех строк первой (левой) таблицы с «прицеплением» строк второй таблицы, если значения *поля соединения* совпадают. Если среди строк второй (правой) таблицы нет строк с соответствующим значением поля соединения, то в итоговой таблице присоединяемые поля заполняются пустыми значениями. При правом внешнем соединении *(RIGHT JOIN)* строки итоговой таблицы строятся по противоположному правилу.

В большинстве случаев запросы на основе внутреннего соединения, по сути, являются процессом *денормализации* связанных таблиц, на которые база данных разделяется при проектировании, исходя из требований рационализации размещения данных.

Запросы на соединение реализуются на основе включения в предложение *FROM* в качестве источника данных конструкции вида «имя_1-й_таблицы *INNER (LEFT/RIGHT) JOIN*имя_2-й_таблицы *ON*имя__поля_соединения_1-й_таблицы=имя_поля_соединения_2-й_таблицы». На рис. 4.15 приведен пример реализации операций внутреннего, а также левого и правого внешних соединений таблиц «Сотрудники» и «Исполнение» (документов) по полю «Фамилия».

На рис. 4.15 приведены также варианты построения SQL-инструкций, для реализации соответствующих запросов. Как видно из рисунка, выбор *типа соединения* определяется *целями* дальнейшего *использования результатов* запроса.

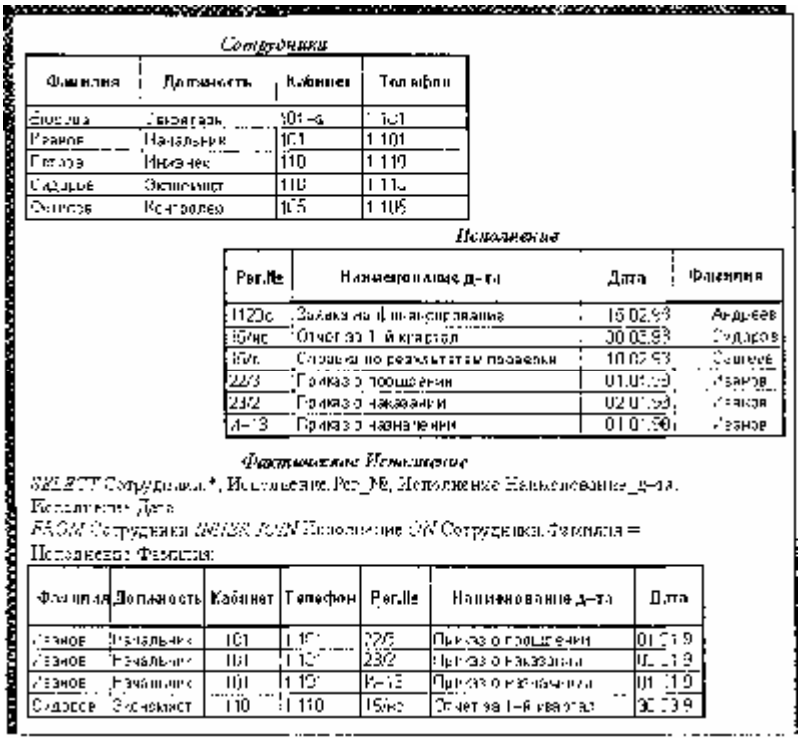


*Рис. 4.15, а. Левое внешнее соединение*

При внутреннем соединении целью является получение новой таблицы с итоговыми данными по уже состоявшимся связям.

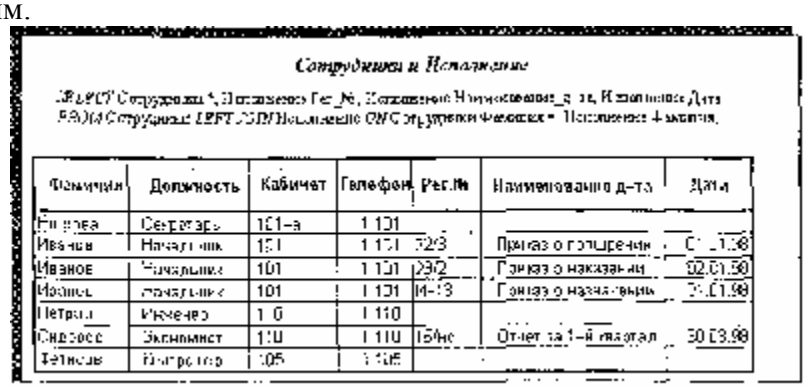


*Рис. 4.15, б. Левое внешнее соединение*

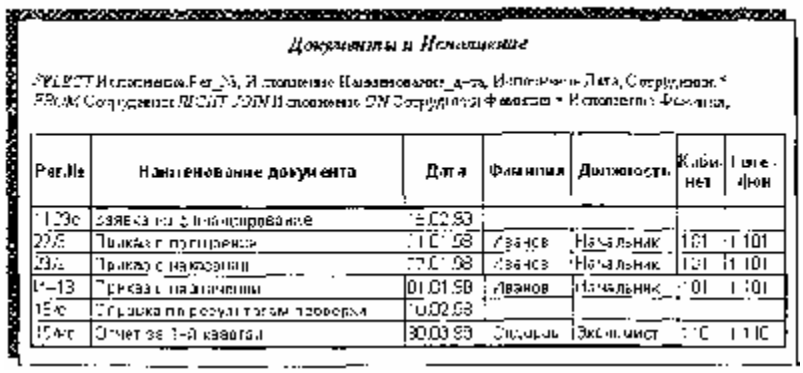


*Рис. 4.15, в. Правое внешнее соединение*

Внешнее соединение по смыслу направлено на создание итоговой таблицы для просмотра и анализа состоявшихся и еще несостоявшихся связей. При этом для левого внешнего объединения упор делается на анализ связей от первой таблицы (в нашем случае от сотрудников, чтобы просмотреть и проанализировать, кто и какие документы исполнил, а кто вообще не исполнил ни одного документа). Иначе говоря, *информация по связям* служит в качестве *дополнительного аспекта, дополнительной характеристики* для записей левой таблицы. Для правого внешнего объединения упор делается на анализ связей от второй таблицы (в нашем случае от «Исполнения», чтобы просмотреть и проанализировать, какие документы исполнены и какими сотрудниками, записи о которых находятся в таблице «Сотрудники»).

В некотором смысле антиподом запросов на соединение является специальный вид запросов на выборку, называемый ***запросом на поиск записей без подчиненных.***

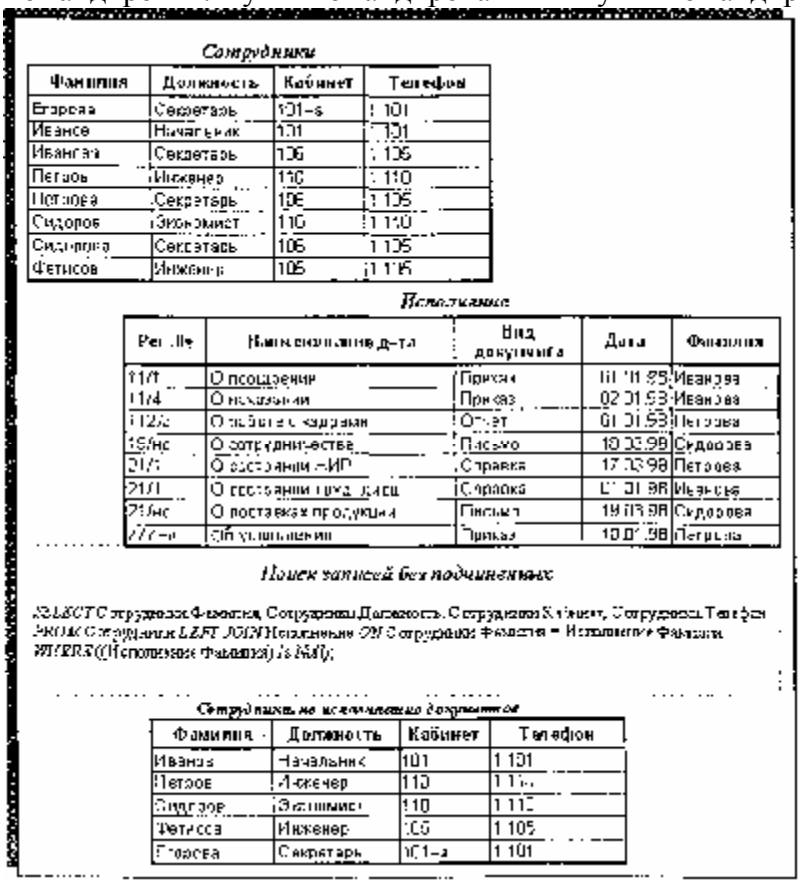
Поиск записей без подчиненных применяется для анализа данных в связанных таблицах, когда связи в силу каких-либо причин не состоялись. Реализуется данный вид запроса на основе запроса на левое (правое) внешнее соединение с дополнительным условием отбора записей с пустыми значениями по полю соединения в правой (левой) таблице. По сути, запрос на поиск записей без подчиненных противоположен запросу на внутреннее соединение. Примером запроса по поиску записей без подчиненных, представленным на рис. 4.16, является запрос, строящий набор записей по таблице «Сотрудники», которые не исполнили ни одного документа, т. е. не имеют подчиненных записей в таблице «Исполнение».

Запросы на соединение могут решать и более сложные логические информационные задачи по анализу связанных данных в цепочках из нескольких таблиц. В качестве примера такого рода запросов* можно привести следующий запрос по формированию набора записей сотрудников, командированных в январе 1998 г. в организации г. Саратова со служебным заданием «Сопровождение поставок», данные из которого выбираются из последовательно связанных отношением «Один-ко-многим» 4-х таблиц — «Сотрудники», «Командировки», «Пункт командирования», «Задания»:

* В данном случае далеко не самого сложного.

SELECTСотрудники*

*FROM* ((Сотрудники *INNER JOIN*Командировки *ON*Сотрудники.ФИО = Командировки.ФИО) *INNER JOIN*Задания *ON*Командировки.Служебное задание = Задания.Наименование) *INNER JOIN*Пункт командирования *ON*Командировки. Пункт командирования = Пункт командирования. Наименование



*Сотрудники*

| Фамилия | Должность | Кабинет | Телефон |
|---|---|---|---|
| Егорова | Секретарь | 101-а | 1 101 |
| Иванов | Начальник | 101 | 1 101 |
| Иванова | Секретарь | 106 | 1 105 |
| Петров | Инженер | 110 | 1 110 |
| Петрова | Секретарь | 106 | 1 105 |
| Сидоров | Экономист | 110 | 1 110 |
| Сидорова | Секретарь | 106 | 1 105 |
| Фетисов | Инженер | 106 | 1 105 |

*Исполнение*

| Рег. № | Наименование акта | Вид документа | Дата | Фамилия |
|---|---|---|---|---|
| 11/1 | О поощрении | Приказ | 01.01.98 | Иванова |
| 11/4 | О назначении | Приказ | 02.01.98 | Иванова |
| 112/2 | О работе с кадрами | Отчет | 01.01.98 | Петрова |
| 15/нс | О сотрудничестве | Письмо | 10.03.98 | Сидорова |
| 21/1 | О создании НИР | Справка | 17.03.98 | Петрова |
| 21/1 | О создании трудового | Справка | 01.01.98 | Иванова |
| 21/нс | О поставках продукции | Письмо | 19.03.98 | Сидорова |
| 777-п | Об увольнении | Приказ | 10.01.98 | Петрова |

*Поиск записей без подчиненных*

SELECT Сотрудники.Фамилия, Сотрудники.Должность, Сотрудники.Кабинет, Сотрудники.Телефон
FROM Сотрудники LEFT JOIN Исполнение ON Сотрудники.Фамилия = Исполнение.Фамилия
WHERE ((Исполнение.Фамилия) Is Null);

*Сотрудники, не исполнившие документов*

| Фамилия | Должность | Кабинет | Телефон |
|---|---|---|---|
| Иванов | Начальник | 101 | 1 101 |
| Петров | Инженер | 110 | 1 110 |
| Сидоров | Экономист | 110 | 1 110 |
| Фетисова | Инженер | 106 | 1 105 |
| Егорова | Секретарь | 101-а | 1 101 |

Рис. 4.16. Пример запроса по поиску записей без подчиненных

*WHERE* ((Пункт командирования.Город) = «Саратов») *AND* ((Задания.Наименование) = «Сопровождение поставок») *AND* ((Командировки. Дата убытия) *Between #1/1/98#And#1/31/98#);*

***Запросы на объединение данных*** реализуют *операцию объединения реляционных таблиц* и решают задачи создания наборов данных, объединяющих однотипные по смыслу записи (по группам однотипных полей) из нескольких таблиц. Строятся запросы на объединение через SQL-инструкцию *SELECT—UNION SELECT.* При этом запрос состоит из первой инструкции *SELECT,* в которой перечисляются отбираемые поля и условия отбора записей из первой таблицы, и последующих ин-

струкций *UNION SELECT,* в которых указываются отбираемые поля и условия отбора записей из других таблиц. Обязательным условием является одинаковое количество отбираемых полей в первой инструкции *SELECT* и последующих инструкциях *UNION SELECT*. При этом типы и длина полей в первой инструкции и последующих инструкциях могут не совпадать.

При необходимости в итоговом наборе данных наименования отбираемых полей можно изменить через ключевое слово *AS* после соответствующего поля в первой инструкции *SELECT*.

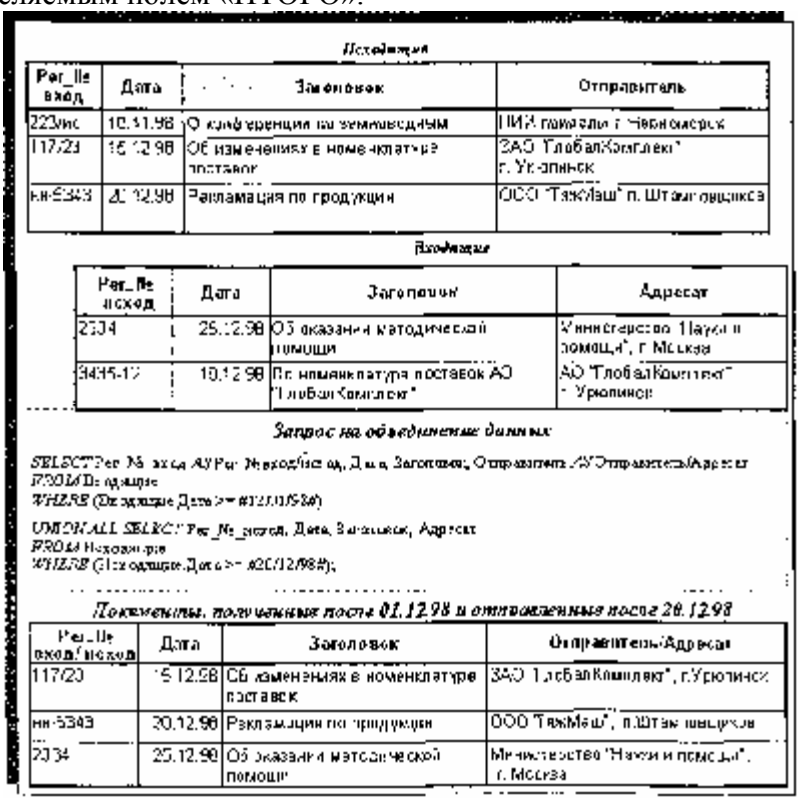
По умолчанию повторяющиеся записи не возвращаются, но через использование предиката *ALL* после ключевого слова *UNION* можно обеспечить режим отбора всех, в том числе и повторяющихся записей.

На рис. 4.17 приведен пример отбора и объединения данных из таблиц «Исходящие» и «Входящие» базы «Документооборот» с целью формирования общего списка документов, поступивших после 1 декабря 1998 г., и документов, отправленных после 20 декабря 1998 г. В запросе первые и последние поля переименованы, чтобы объединить смысл этих полей в исходных таблицах.

### 4.3.2.1.3. Вычисления и групповые операции в запросах

Во многих случаях при формировании набора данных по запросам на выборку требуется производить определенные вычисления или определенные операции по непосредственной обработке отбираемых данных. В реляционных СУБД такие возможности предоставляются через *вычисляемые поля* и *групповые операции* в запросах над отбираемыми данными.

***Вычисляемые поля.*** В инструкции *SELECT* в списке отбираемых полей добавляется выражение, по которому вычисляется новое поле, и посредством ключевого слова *AS* определяется его имя в формируемом наборе данных. На рис. 4.18 приведен запрос, формирующий ведомость начислений сотрудникам с вычисляемым полем «ИТОГО».



*Исходящие*

| Рег_№ вход | Дата | Заголовок | Отправитель |
|---|---|---|---|
| 223/исх | 10.11.98 | О конференции по земноводным | ПИЯ пожаловать г.Новосибирск |
| 117/23 | 15.12.98 | Об изменениях в номенклатуре поставок | ЗАО "ГлобалКомплект" г. Урюпинск |
| нн-5343 | 20.12.98 | Рекламация по продукции | ООО "ТяжМаш" п. Штамповщиков |

*Входящие*

| Рег_№ исход | Дата | Заголовок | Адресат |
|---|---|---|---|
| 2034 | 25.12.98 | Об оказании методической помощи | Министерство "Науки и помощи", г. Москва |
| 3445-12 | 10.12.98 | По номенклатуре поставок АО "ГлобалКомплект" | АО "ГлобалКомплект" г. Урюпинск |

*Запрос на объединение данных*

```
SELECT Рег_№_исхода AS Рег_№_входящего, Дата, Заголовок, Отправитель AS Отправитель/Адресат
FROM Исходящие
WHERE (Исходящие.Дата >= #12/01/98#)
UNION ALL SELECT Рег_№_исхода, Дата, Заголовок, Адресат
FROM Входящие
WHERE (Входящие.Дата >= #12/20/98#);
```

*Документы, полученные после 01.12.98 и отправленные после 20.12.98*

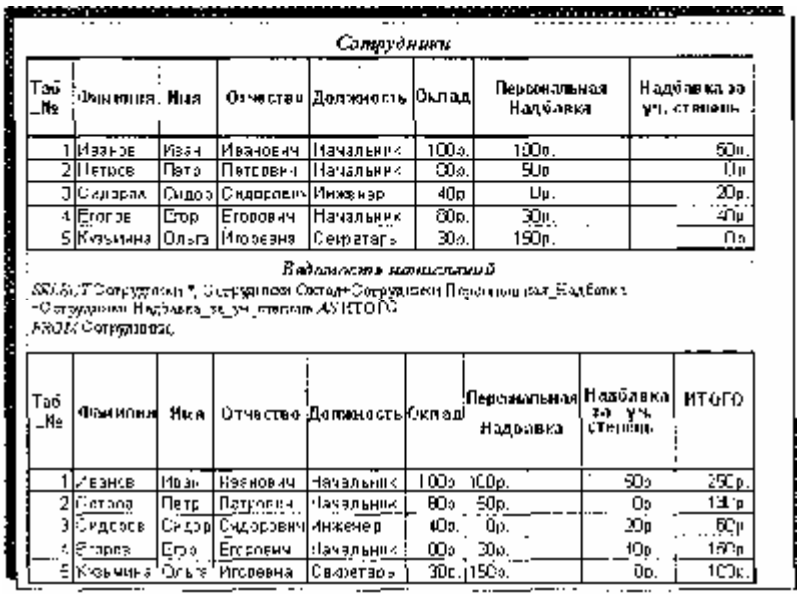
| Рег_№ вход/исход | Дата | Заголовок | Отправитель/Адресат |
|---|---|---|---|
| 117/23 | 15.12.98 | Об изменениях в номенклатуре поставок | ЗАО "ГлобалКомплект", г.Урюпинск |
| нн-5343 | 20.12.98 | Рекламация по продукции | ООО "ТяжМаш", п.Штамповщиков |
| 2034 | 25.12.98 | Об оказании методической помощи | Министерство "Науки и помощи", г. Москва |

*Рис. 4.17. Пример запроса на объединение*

***Групповые операции***. В процессе отбора и обработки данных важное значение имеют группирование данных по значениям какого-либо поля и осуществление тех или иных операций над сгруппированными записями. Групповые операции осуществляются на основе SQL-предложения *GROUP BY* в сочетании со ***статистическими функциями SQL***. В большинстве диалектов языка SQL в состав инструкции *SELECT* допускается включение статистических функций SQL, которые осуществляют те или иные ***групповые вычислительные операции*** над отбираемыми записями.

К числу статистических функций SQL относятся:

*SUM*(выражение) — вычисляет сумму набора значений;



*Сотрудники*

| Таб_№ | Фамилия | Имя | Отчество | Должность | Оклад | Персональная Надбавка | Надбавка за уч. степень |
|---|---|---|---|---|---|---|---|
| 1 | Иванов | Иван | Иванович | Начальник | 100р. | 100р. | 50р. |
| 2 | Петров | Петр | Петрович | Начальник | 80р. | 50р. | 0р. |
| 3 | Сидоров | Сидор | Сидорович | Инженер | 40р. | 0р. | 20р. |
| 4 | Егоров | Егор | Егорович | Начальник | 80р. | 30р. | 40р. |
| 5 | Кузьмина | Ольга | Игоревна | Секретарь | 30р. | 150р. | 0р. |

*Надбавка начальникам*

```
SELECT Сотрудники.*, Сотрудники.Оклад+Сотрудники.Персональная_Надбавка
+Сотрудники.Надбавка_за_уч_степень AS ИТОГО
FROM Сотрудники;
```

| Таб_№ | Фамилия | Имя | Отчество | Должность | Оклад | Персональная Надбавка | Надбавка за уч. степень | ИТОГО |
|---|---|---|---|---|---|---|---|---|
| 1 | Иванов | Иван | Иванович | Начальник | 100р. | 100р. | 50р. | 250р. |
| 2 | Петров | Петр | Петрович | Начальник | 80р. | 50р. | 0р. | 130р. |
| 3 | Сидоров | Сидор | Сидорович | Инженер | 40р. | 0р. | 20р. | 60р. |
| 4 | Егоров | Егор | Егорович | Начальник | 80р. | 30р. | 40р. | 150р. |
| 5 | Кузьмина | Ольга | Игоревна | Секретарь | 30р. | 150р. | 0р. | 180р. |

*Рис. 4.18. Пример запроса на выборку с вычисляемым полем*

*AVG*(выражение) — вычисляет среднее арифметическое набора чисел;

*Min*(выражение) — вычисляет минимальное значение из набора значений;

Max(выражение) — вычисляет максимальное значение из набора значений;

*StDev*(выражение) — вычисляет среднеквадратичное отклонение набора значений;

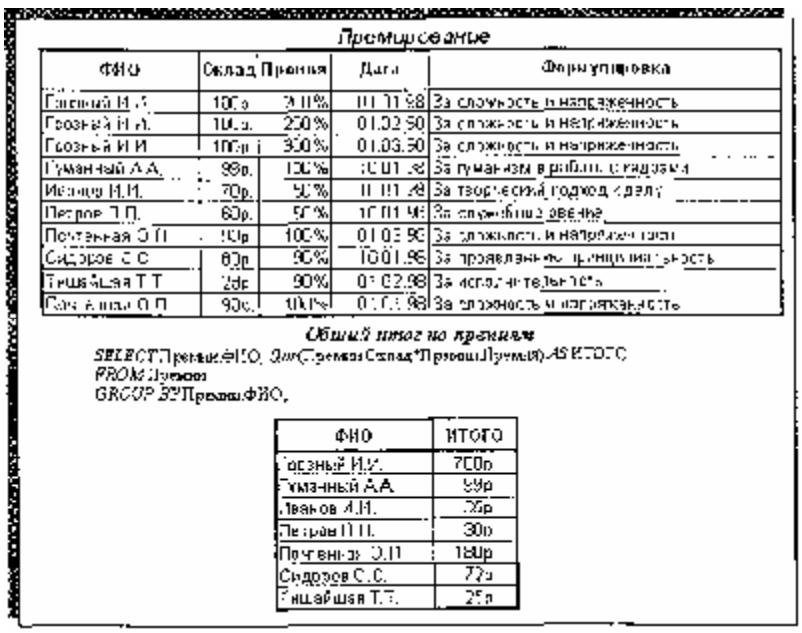
*Count*(выражение) — вычисляет количество записей, содержащихся в наборе;

*Var*(выражение) — вычисляет дисперсию по набору значений.

К числу функций, используемых в групповых операциях, относятся также функции First(выражение) и Last(выражение), вычисляющие (возвращающие), соответственно, первое и последнее значения поля в наборе данных. В выражениях в качестве аргумента допускается использование имен полей таблиц.

Собственно сами групповые вычисления задаются посредством включения в SQL-инструкцию *SELECT* вычисляемого поля на основе выражения со статистическими функциями, выполняемыми над наборами данных, формируемыми предложением *GROUP BY.*

Для примера на рис. 4.19 приведен запрос, формирующий итоговые данные по общей сумме премиальных каждого из премированных сотрудников. Группирование данных производится по полю «ФИО», т. е. все записи с одинаковыми значениями поля «ФИО», объединяются в одну и дополнительно формируется вычисляемое поле «ИТОГО», рассчитываемое как сумма сгруппированных в одну записей по произведению полей «Оклад» и «Премия».



*Премирование*

| ФИО | Оклад | Премия | Дата | Формулировка |
|---|---|---|---|---|
| Грозный И.В. | 100р. | 200% | 01.01.98 | За сложность и напряженность |
| Грозный И.В. | 100р. | 200% | 01.02.98 | За сложность и напряженность |
| Грозный И.В. | 100р. | 300% | 01.03.98 | За сложность и напряженность |
| Гуманный А.А. | 99р. | 100% | 10.01.98 | За гуманизм в работе с людьми |
| Иванов И.И. | 70р. | 90% | 11.01.98 | За творческий подход к делу |
| Петров П.П. | 60р. | 50% | 10.01.98 | За служебное рвение |
| Почтенная О.П. | 10р. | 100% | 01.03.98 | За сложность и напряженность |
| Сидоров С.С. | 80р. | 90% | 10.01.98 | За проявленную принципиальность |
| Тишайшая Т.Т. | 25р. | 90% | 01.02.98 | За исполнительность |
| Сочувствующая О.П. | 90р. | 100% | 01.01.98 | За сложность и напряженность |

*Общий итог по премиям*

```
SELECT Премии.ФИО, Sum(Премии.Оклад*Премии.Премия) AS ИТОГО
FROM Премии
GROUP BY Премии.ФИО;
```

| ФИО | ИТОГО |
|---|---|
| Грозный И.В. | 700р |
| Гуманный А.А. | 99р |
| Иванов И.И. | 63р |
| Петров П.П. | 30р |
| Почтенная О.П. | 180р |
| Сидоров С.С. | 72р |
| Тишайшая Т.Т. | 25р |

*Рис. 4.19. Пример запроса с групповой функцией Sum*

В некоторых СУБД в отдельный вид выделяются *запросы по поиску повторов,* а также вводится специальная разновидность запросов на выборку в виде так называемых *перекрестных запросов.*

***Запросы по поиску повторов*** применяются для анализа наличия повторяющихся групп значений по определенному полю и их количественных (статистических) данных. В качестве примера на рис. 4.20 приведен запрос по поиску повторов в таблице сотрудники по полю «Должность», формирующий в итоге штатную расстановку по заполненным должностям.

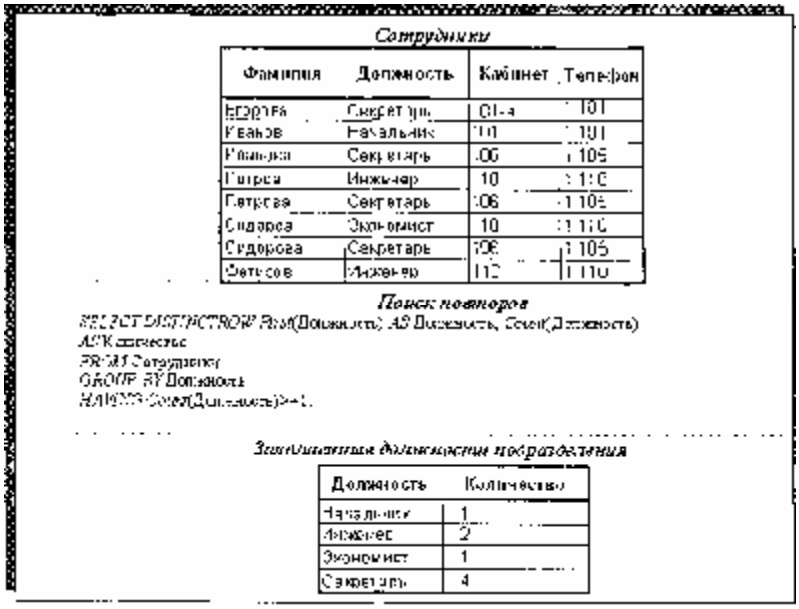



*Рис. 4.20. Пример запроса по поиску повторяющихся значений*

Более сложные статистические задачи решают ***перекрестные запросы.*** Название «перекрестный» отражает принцип формирования и представления результатов таких запросов.

На рис. 4.21 иллюстрируется принцип построения итоговой (сводной) таблицы перекрестного запроса. В исходной (базовой) таблице для перекрестного запроса выбираются два поля. По повторяющимся значениям одного поля формируются названия заголовков строк итоговой (сводной) таблицы — «боковик» сводной таблицы. По повторяющимся значениям другого поля образуются названия столбцов итоговой таблицы — «шапка» сводной таблицы. В ячейках сводной таблицы отражаются результаты статистических функций по группам данных в каких-либо полях исходной таблицы.

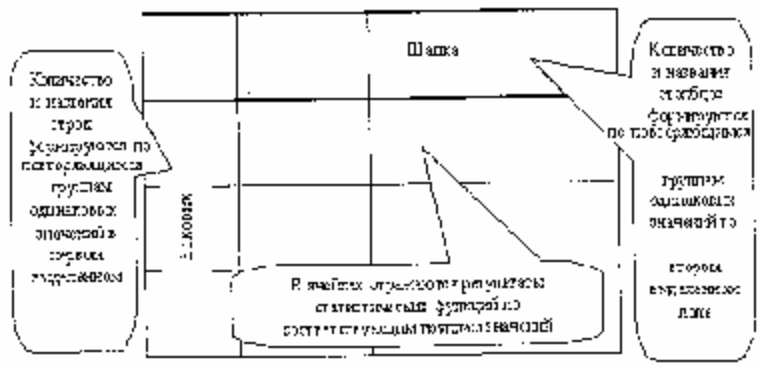
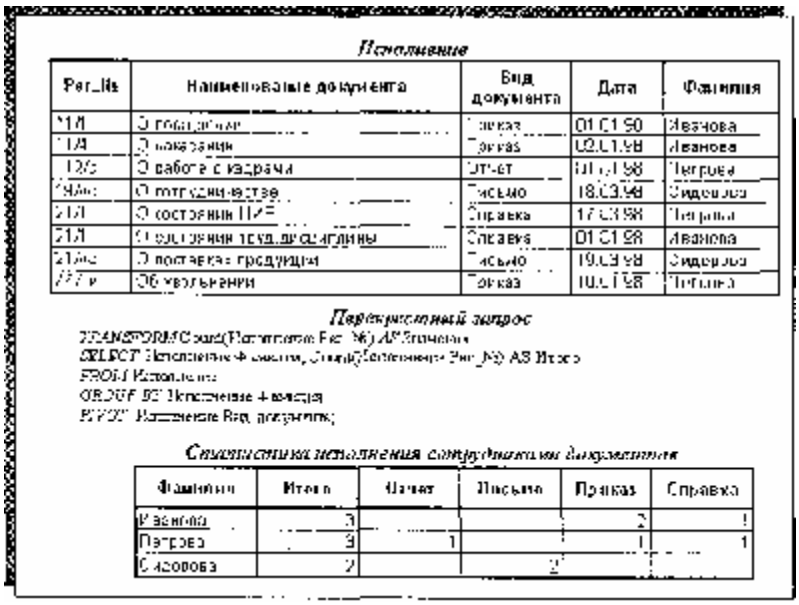



*Рис. 4.21. Принцип формирования результатов перекрестного запроса*

Для примера на рис. 4.22 представлен запрос по формированию статистических данных о количестве исполненных различными сотрудниками документов с разнесением по видам документов. Полем для формирования боковика определено поле «Фамилия», полем для формирования шапки определено поле «Вид документа». На рисунке представлен также вариант SQL-инструкции, реализующей данный запрос.*

* В диалекте SQL MS Access.



*Исполнение*

| Рег_№ | Наименование документа | Вид документа | Дата | Фамилия |
|---|---|---|---|---|
| [illegible] | [illegible] | [illegible] | 01.01.98 | Иванова |
| [illegible] | [illegible] | [illegible] | 02.01.98 | Иванова |
| [illegible] | [illegible] | [illegible] | 01.01.98 | Петрова |
| [illegible] | [illegible] | [illegible] | 18.03.98 | Сидорова |
| [illegible] | [illegible] | [illegible] | 17.03.98 | Петрова |
| [illegible] | [illegible] | [illegible] | 01.01.98 | Иванова |
| [illegible] | [illegible] | [illegible] | 19.03.98 | Сидорова |
| [illegible] | [illegible] | [illegible] | 10.01.98 | Петрова |

*Перекрестный запрос*

TRANSFORM Count(Исполнение.Рег_№) AS Значение
SELECT Исполнение.Фамилия, Count(Исполнение.Рег_№) AS Итого
FROM Исполнение
GROUP BY Исполнение.Фамилия
PIVOT Наименование Вид документ;

*Статистика исполнения сотрудниками документов*

| Фамилия | Итого | Звонок | Письмо | Приказ | Справка |
|---|---|---|---|---|---|
| Иванова | 3 | | | 2 | 1 |
| Петрова | 3 | 1 | | 1 | 1 |
| Сидорова | 2 | | 2 | | |

*Рис. 4.22. Пример перекрестного запроса*

### 4.3.2.2. Запросы на изменение данных

Важное значение для решения различных *технологических информационных задач* по ведению базы данных имеют запросы на изменение данных. В отличие от непосредственного ввода данных в режимах открытой таблицы или формы они *вносят изменения сразу в группу записей за одну операцию.* Таким образом, *результатом* запросов на изменение является не набор данных, как в запросах на выборку, а *изменение данных* в самих таблицах.

Запросы ***на изменение данных*** широко применяются для ввода данных при *импорте из внешних источников, перемещения записей* или их элементов *из одних таблиц в другие таблицы,* при *массовой однотипной коррекции* или *чистке данных,* а также для *архивации* и *экспорта* данных.

Существует *четыре* разновидности запросов на изменение:

• запросы на удаление;
• запросы на обновление;
• запросы на добавление;
• запросы на создание таблицы.

При исполнении запроса ***на удаление*** *за одну операцию* осуществляется *удаление группы записей из одной или нескольких таблиц.* Запросы на удаление реализуются SQL-инструкцией *DELETE.* К примеру, из таблицы «Клиенты» с помощью запроса на удаление можно за одну операцию удалить всех клиентов, проживающих в районе «Марьина Роща». SQL-инструкция такого запроса может выглядеть следующим образом:

*DELETE*Клиенты.*, Клиенты.Район
*FROM* Клиенты
*WHERE* ((Клиенты.Район) = «Марьина Роща»));

Удаление записей одним запросом из нескольких таблиц может осуществляться путем перечисления через запятую в соответствующей SQL-инструкции имен таблиц и имен полей, задающих условия удаления, или по связям между таблицами при установке ограничений целостности связей в режим «Каскадного удаления связанных записей».

Запрос на ***обновление*** *за одну операцию* вносит *общие изменения в группу записей одной или нескольких таблиц.* Реализуются SQL-инструкцией *UPDATE.* Запросы на обновления применяются тогда, когда необходимо осуществить *глобальные однотипные изменения* в каком-либо наборе данных. В качестве примера приведем ситуацию, когда в результате очередной деноминации (девальвации) всем сотрудникам необходимо в 10 раз уменьшить (увеличить) должностные оклады. Вариант SQL-инструкции, реализующей такой запрос, может выглядеть следующим образом:

*UPDATE* Сотрудники
*SET*Сотрудники.Оклад=Оклад/10;

В качестве другого примера приведем ситуацию, когда всех работников-совместителей учебного учреждения необходимо перевести в категорию почасовиков:

*UPDATE* Сотрудники

*SET*Сотрудники.Статус=«Почасовик»
*WHERE*((Сотрудники.Статус)=«Совместитель»);

Обновление записей сразу в нескольких таблицах, также как и удаление, может осуществляться путем перечисления через запятую в инструкции *UPDATE* имен таблиц, полей, их значений и соответствующих условий, а также по связям между таблицами с предварительной установкой ограничений целостности связей в режим «Каскадного обновления связанных записей».

Запрос на ***добавление*** осуществляет добавление *группы записей* из *одной или нескольких таблиц в конец другой или группы других таблиц.* При этом *количество* и *типы полей** при вставке записей *должны совпадать.* Запросы на добавление могут вставлять записи из текущей (открытой) базы данных в другую (внешнюю) базу данных. В этом случае запросы на добавление реализуют функции *экспорта* данных, решая задачи по ***обмену, архивации*** или ***резервированию данных.*** Однако чаще данные запросы применяются для добавления записей из одной таблицы базы данных в другую таблицу.

* Не обязательно имена, но обязательно типы полей.

Запросы на добавление реализуются SQL-инструкцией *INSERT INTO.* Предположим, в базе данных имеются две таблицы «Студенты» и «Научные работники» с однотипным набором полей. Предположим также, что 100% студентов группы И-405 приняли участие в конкурсе научных студенческих работ и опубликовали свои труды в университетском сборнике. Тем самым, будучи еще студентами, они перешли в разряд научных работников. В этом случае запросом на добавление одной операцией в таблицу «Научные работники» можно добавить группу новых записей. Вариант SQL-инструкции, реализующей такой запрос, может иметь вид:

*INSERT INTO* НаучныеРаботники
*SELECT*Студенты.*
*FROM* Студенты
*WHERE* ((Студенты.Группа)=«И-405»);

Запросы на ***создание таблицы*** *за одну операцию создают новую таблицу с заполненными данными на основе всех или части данных из одной или нескольких таблиц.* Так же как и запросы на добавление, эти запросы чаще всего решают задачи по *реформированию* (реорганизации) базы данных, *архивированию или резервированию* данных, а также могут применяться для создания отчетов или *состояний* базы данных по определенным временным промежуткам. Реализуются SQL-инструкцией *SELECT...INTO.* Для примера приведем задачу создания специального набора (отчета) данных за месяц, скажем за январь, из таблицы «Заказы» в виде отдельной таблицы (для отдельного хранения или обработки). Вариант соответствующей SQL-инструкции может выглядеть следующим образом:

*SELECT*Заказы *
*INTO* Заказы Января
*FROM*Заказы
*WHERE*((Заказы.Дата)=*BETWEEN*#1/01/*98#AND*#1/02/98#;

### 4.3.2.3. Управляющие запросы

В большинстве современных СУБД проектирование и создание таблиц осуществляются через специальные диалогово-наглядные конструкторы или пошаговые мастера. Тем не менее, как уже отмечалось, в составе языка описания данных DDL имеются ряд SQL-инструкций, на основе которых строятся запросы по созданию/модификации реляционных таблиц или отдельных их элементов. Такие запросы называются ***управляющими***.

Имеется *четыре* вида управляющих запросов:

- запросы на создание таблицы;*
- запросы на добавление в существующую таблицу нового поля или индекса;
- запросы на удаление таблицы или индекса определенного поля таблицы;
- запросы на создание индекса для поля или группы полей таблицы.

* В отличие от одноименного запроса из группы запросов на изменение данный тип запроса не использует в качестве исходных данных другие уже существующие в базе данных таблицы, т. е. создает новую пустую таблицу.

***Запросы на создание таблицы*** реализуются SQL-инструкцией *CREATE TABLE* с ключевыми словами, определяющими типы полей *(CHARACTER, INTEGER, DATETIME* и т.д.), предложением *CONSTRAINT для* создания ограничений на значения полей или связей между таблицами, ключевым

словом *UNIQUE,* задающим свойство уникальности (требование на отсутствие совпадений) индекса таблицы, а также ключевого слова *PRIMARYKEY,* определяющего ключевое поле создаваемой таблицы.

В качестве примера приведем запрос на создание таблицы «Сотрудники» с полями «Фамилия», «Имя», «ДатаРождения», уникальным составным индексом «ИндексСотрудники» для полей «Фамилия», «Имя», «ДатаРождения», с тем же набором полей для составного ключа «КлючСотрудники».*

* В стандартах SQL и в большинстве диалектов SQL символы кириллицы в названиях полей не допускаются.

*CREATE TABLE*Сотрудники
(Фамилия *TEXT,* Имя *TEXT,* ДатаРождения *DA TETIME, CONSTRAINT*ИндексСотрудники *UNIQUE*(Имя, Фамилия, ДатаРождения)
КлючСотрудники *PRIMARY KEY*);

***Запросы на добавление полей или индексов*** реализуются SQL-инструкцией *ALTER TABLE* с использованием зарезервированных слов *ADD COLUMN* (добавить поле) и *ADD CONSTRAINT* (добавить индекс). Этим же запросом с помощью зарезервированного слова *DROP COLUMN* можно удалить поле из существующей таблицы. Как правило, запросы на добавление полей также используются для создания внешних ключей, задающих связи-отношения между таблицами. С этой целью используются зарезервированные слова *FOREIGN KEY и REFERENCES.*

Для примера приведем запросы по добавлению в таблицу «Сотрудники» нового поля «Оклад», добавлению нового индекса «ОкладСотрудники», удалению поля «Оклад», добавлению внешнего ключа «№_Отдела» и удалению внешнего ключа:

*ALTER TABLE* Сотрудники *ADD COLUMN* Оклад *CURRENCY*;
*ALTER TABLE* Сотрудники *ADD CONSTRAINT*ОкладСотрудники Оклад;
*ALTER TABLE* Сотрудники *DROPCOLUMN*Оклад;
*ALTER TABLE*Сотрудники*ADD CONSTRAINT*Работа*FOREIGN KEY*(№_Отдела)
*REFERENCES* Подразделения (№_Отдела);
*ALTER TABLE* Сотрудники *DROP CONSTRAINT*Работа;

***Запросы на удаление таблицы или индекса*** реализуются SQL-инструкцией *DROP TABLE* с указанием имени удаляемой таблицы или индекса. Следующий пример иллюстрирует удаление из базы данных таблицы «Сотрудники» и удаление индекса «ОкладСотрудники»:

*DROPTABLE* Сотрудники;
*DROPINDEX*ОкладСотрудники *ON* Сотрудники;

***Запросы на создание индекса*** реализуются SQL-инструкцией *CREATEINDEX с* использованием зарезервированного слова *UNIQUE для* запрета повтора значений в индексируемом поле и *необязательного предложения WITH с параметрами DISALLOW NULL* и *IGNORE NULL* для запрета/разрешения нулевых (пустых) значений в индексируемом поле. Зарезервированное слово *PRIMARY* позволяет определить создаваемый индекс ключом таблицы (при этом создаваемый индекс по умолчанию является уникальным, т.е. повторы значений не допускаются).

В следующем примере в таблице «Сотрудники» создастся уникальный индекс «ИндексСотрудника» по полю «Таб_№» с запретом пустых значений:

*CREATE UNIQUE INDEX* ИндексСотрудника
*ON* Сотрудники (Таб_№)
*WITH DISALLOW NULL;*

### 4.3.2.4. Подчиненные (сложные) запросы

Как уже отмечалось, *источником* данных для запросов могут быть результаты выполнения *других запросов.* Возможны *два* варианта построения таких запросов.

*Первый вариант* реализуется через указание в SQL-инструкциях в качестве имен таблиц и имен полей имен запросов и полей запросов. Синтаксис таких запросов ничем не отличается от обычных запросов, а его исполнение осуществляется в две фазы. По запуску основного запроса сначала неявно запускается запрос, формирующий источник данных, и по завершению его исполнения запускается основной (внешний) запрос.

*Второй вариант* реализуется через *включение* в тело внешней (главной) SQL-инструкции ***внутренней инструкции SELECT.*** При этом результат исполнения внутренней инструкции *SELECT*

используется для формирования условия отбора записей в главном (внешнем) запросе или в качестве выражения для нового вычисляемого поля. Такие запросы называются ***подчиненными.***

Использование внутренней инструкции *SELECT для формирования условий отбора записей во внешнем запросе* возможно одним из *трех* способов:

• через предикаты сравнения «для некоторых/для всех» — *ANY, SOME, ALL;*

• через предикат вхождения *IN;*

• через предикат существования *EXISTS.*

В *первом способе* конструкция запроса может выглядеть следующим образом:

*SELECT.. FROM...WHERE*Выражение◊[*ANY*|*SOME*|*ALL*] (SELECT...);

где ◊ — оператор сравнения.

Как правило, выражение включает поле из списка полей внешней SQL-инструкции или функцию от этих полей. Внутренняя инструкция *SELECT* должна возвращать набор данных по одному полю или по вычисляемому полю, при принципиальной сравнимости с выражением во внешней SQL-инструкции (главным образом по типу данных).

***Предикаты ANY*** и ***SOME*** *(«для некоторых»),* являющиеся синонимами, используются для отбора в главной SQL-инст-рукции тех записей, которые удовлетворяют сравнению *с какой-либо записью* (т. е. *хотя бы с одной),* из отобранных во внутренней инструкции *SELECT.*

Для примера на рис. 4.23, *а* приведен запрос по отбору из таблицы «Заявки» тех записей, которые могут быть удовлетворены в соответствии с данными по таблице «Квартиры», т. е. тех записей из таблицы «Заявки», которые удовлетворяют сравнению по полям «КолКомп», «Площадь» и «Этаж» хотя бы с некоторыми *(ANY)* записями из таблицы «Квартиры», выбираемыми при условии «Продано=Нет».

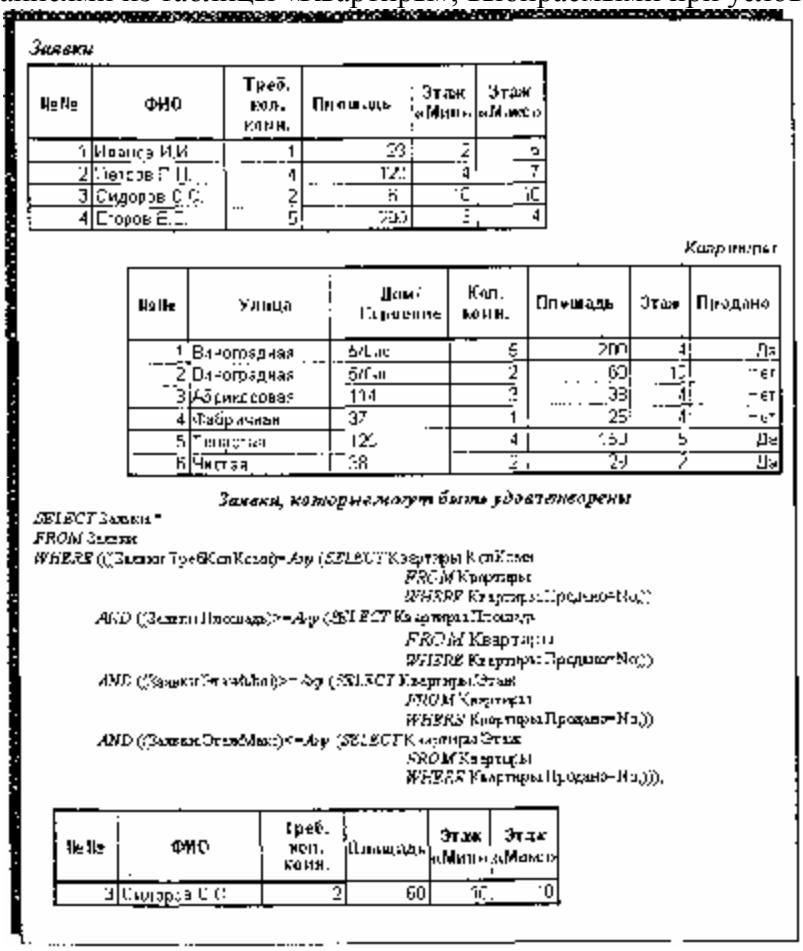
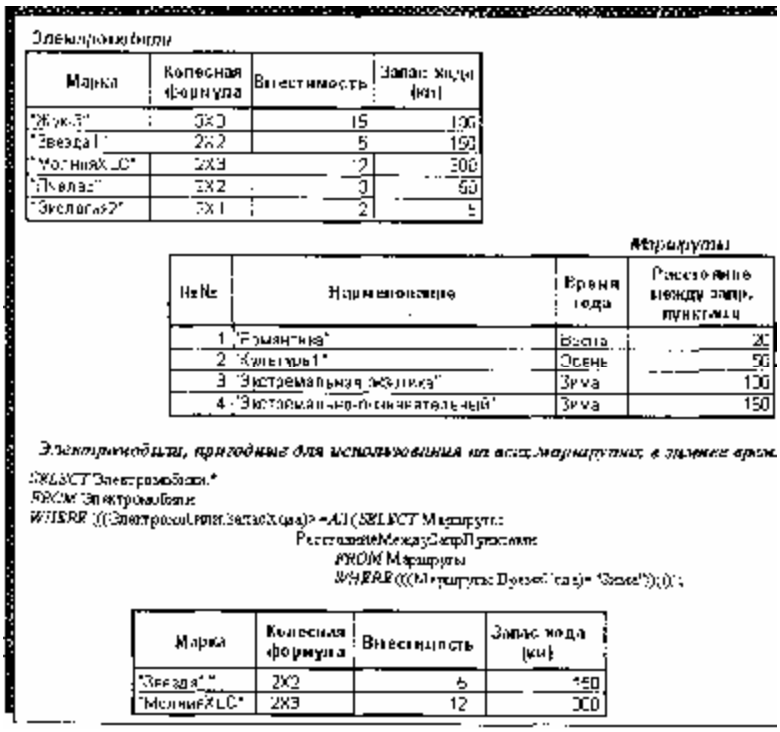



*Рис. 4.23, а. Пример подчиненного запроса на основе предиката ANY*

***Предикат ALL*** *(для всех)* используется для отбора в главном запросе только тех записей, которые удовлетворяют сравнению *одновременно со всеми записями,* отобранными в подчиненном запросе. Пример выполнения запроса с предикатом *ALL* приведен на рис. 4.23, *б.*



Электромобили

| Марка | Колесная формула | Вместимость | Запас хода (км) |
|---|---|---|---|
| "Жук-1" | 3X3 | 15 | 100 |
| "Звезда1" | 2X2 | 5 | 150 |
| "МолнияXLC" | 2X3 | 12 | 300 |
| "Пчела1" | 2X2 | 3 | 90 |
| "Электро62" | 3X1 | 2 | 5 |

Маршруты

| №№ | Наименование | Время года | Расстояние между экстр. пунктами |
|---|---|---|---|
| 1 | "Романтика" | Весна | 20 |
| 2 | "Кольцевой" | Осень | 50 |
| 3 | "Экстремальная езда" | Зима | 100 |
| 4 | "Экстремально-познавательный" | Зима | 150 |

*Электромобили, пригодные для использования на всех маршрутах в зимнее время*

```
SELECT Электромобили.*
FROM Электромобили
WHERE ((Электромобили.ЗапасХода)>=ALL (SELECT Маршруты.
                                       РасстояниеМеждуЭкстрПунктами
                                  FROM Маршруты
                                  WHERE ((Маршруты.Время года)="Зима"));();
```

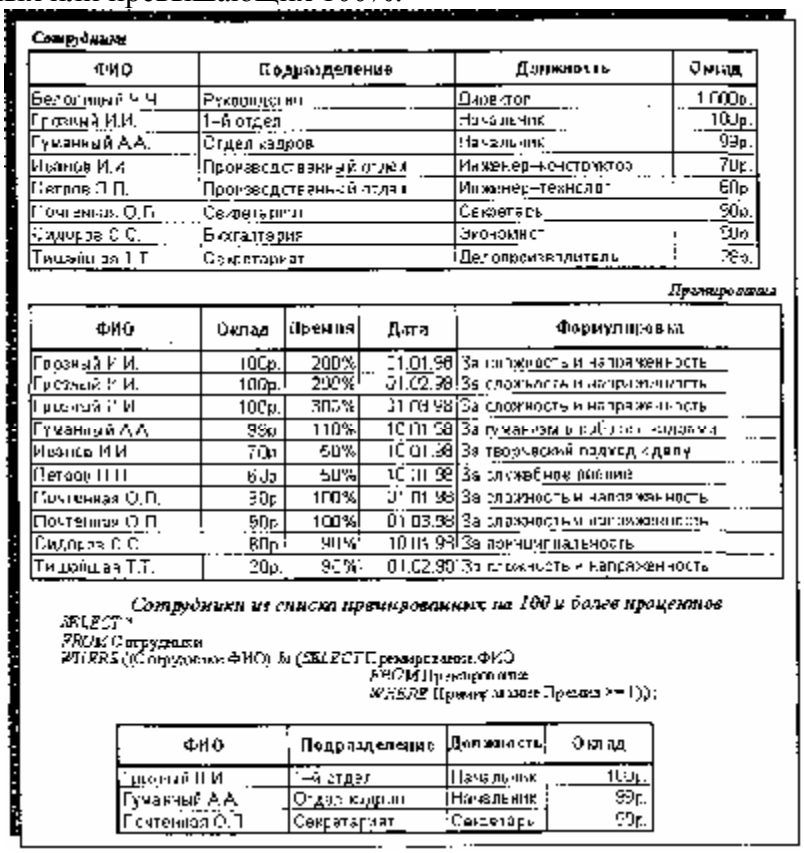
| Марка | Колесная формула | Вместимость | Запас хода (км) |
|---|---|---|---|
| "Звезда1" | 2X2 | 5 | 150 |
| "МолнияXLC" | 2X3 | 12 | 300 |

*Рис. 4.23, б. Пример подчиненного запроса па основе предиката ALL*

Принцип действия запроса по *второму способу* заключается в поиске среди результирующего набора записей внешней SQL-инструкции тех записей, для которых значение определенного выражения входит в список записей, отбираемых внутренней инструкцией *SELECT*. Конструкция запроса с предикатом *IN* выглядит следующим образом:

*SELECT ... FROM ... WHERE* Выражение [*NOT*]*IN (SELECT...);*

В качестве примера на рис. 4.24 приведен запрос по тем же исходным данным (таблицы «Сотрудники» и «Премирование») для отбора записей только тех сотрудников, записи которых по премиям были в списке премий, равных или превышающих 100%.



Сотрудники

| ФИО | Подразделение | Должность | Оклад |
|---|---|---|---|
| Белогорский Ч.Ч. | Руководство | Директор | 1 000р. |
| Грозный И.И. | 1-й отдел | Начальник | 100р. |
| Гуманный А.А. | Отдел кадров | Начальник | 99р. |
| Иванов И.И. | Производственный отдел | Инженер-конструктор | 70р. |
| Петров П.П. | Производственный отдел | Инженер-технолог | 60р. |
| Почтенная О.П. | Секретариат | Секретарь | 90р. |
| Сидоров С.С. | Бухгалтерия | Экономист | 90р. |
| Тихонова Т.Т. | Секретариат | Делопроизводитель | 80р. |

Премирование

| ФИО | Оклад | Премия | Дата | Формулировка |
|---|---|---|---|---|
| Грозный И.И. | 100р. | 200% | 21.01.98 | За сложность и напряженность |
| Грозный И.И. | 100р. | 200% | 01.02.98 | За сложность и напряженность |
| Грозный И.И. | 100р. | 305% | 31.08.98 | За сложность и напряженность |
| Гуманный А.А. | 99р. | 110% | 10.01.98 | За гуманизм в работе с людьми |
| Иванов И.И. | 70р. | 50% | 10.01.98 | За творческий подход к делу |
| Петров П.П. | 60р. | 50% | 10.01.98 | За служебное рвение |
| Почтенная О.П. | 90р. | 100% | 01.01.98 | За сложность и напряженность |
| Почтенная О.П. | 90р. | 100% | 01.03.98 | За сложность и напряженность |
| Сидорова С.С. | 80р. | 90% | 10.01.98 | За пунктуальность |
| Тихонова Т.Т. | 20р. | 90% | 01.02.98 | За сложность и напряженность |

*Сотрудники из списка премированных на 100 и более процентов*

```
SELECT *
FROM Сотрудники
WHERE ((Сотрудники.ФИО) In (SELECT Премирование.ФИО
                            FROM Премирование
                            WHERE (Премирование.Премия >= 1)));
```

| ФИО | Подразделение | Должность | Оклад |
|---|---|---|---|
| Грозный И.И. | 1-й отдел | Начальник | 100р. |
| Гуманный А.А. | Отдел кадров | Начальник | 99р. |
| Почтенная О.П. | Секретариат | Секретарь | 90р. |

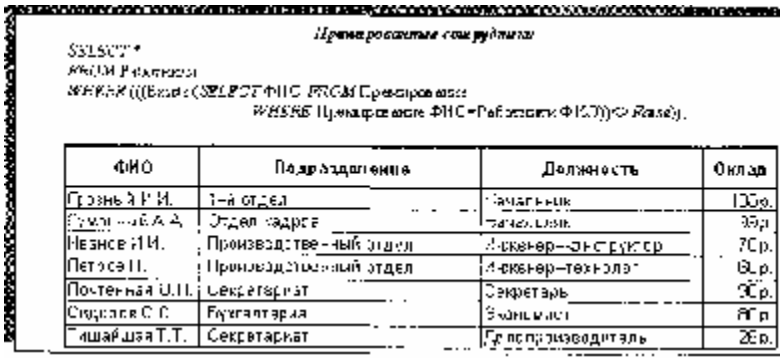
*Рис. 4.24. Пример с подчиненным запросом на основе предиката IN*

Следует добавить, что предикат *NOT IN* используется для отбора во внешней SQL-инструкции только тех записей, которые содержат значения, не совпадающие ни с одним из отобранных внутренней инструкцией *SELECT*.

В *третьем способе* построения подчиненного запроса ***предикат EXISTS*** (с необязательным зарезервированным словом *NOT)* используется в логическом выражении для определения того, должен ли подчиненный запрос *возвращать какие-либо записи.* Исходя из этого, каждая запись, отбираемая во внешней SQL-инструкции, идет в итоговый набор данных только тогда, когда при ее условиях отбирается (существует) хотя бы одна запись по внутренней инструкции *SELECT.* Конструкция запроса в этом случае может выглядеть следующим образом:

*SELECT...FROM..WHERE*([NOT]*Exists*(*SELECT...*));

В следующем примере на рис. 4.25 по таблицам, представленным на рис. 4.24, иллюстрируется запрос по отбору записей сотрудников, премированных хотя бы один раз, т.е. таких, у которых существует запись в списке премированных.



| ФИО | Подразделение | Должность | Оклад |
|---|---|---|---|
| Грозный Р.И. | 1-й отдел | Начальник | 130р. |
| [illegible] | Отдел кадров | [illegible] | 90р. |
| Незнов И.И. | Производственный отдел | Инженер-конструктор | 70р. |
| Петров П. | Производственный отдел | Инженер-технолог | 60р. |
| Постная О.П. | Секретариат | Секретарь | 30р. |
| Сидоров С.С. | Бухгалтерия | Бухгалтер | 80р. |
| Тишайшая Т.Т. | Секретариат | Делопроизводитель | 25р. |

*Рис. 4.25. Пример с подчиненным запросом на основе предиката EXISTS*

Следует заметить, что использование предиката *NOT EXISTS* сформирует список ни разу не премированных сотрудников. В последнем примере можно также увидеть, что альтернативным решением для реализации такого запроса является использование запроса на внутреннее соединение *(INNER JOIN).*

Внутренняя инструкция *SELECT* может также использоваться в качестве выражения для вычисляемого поля внешней SQL-инструкции. Конструкция запроса в этом случае может выглядеть следующим образом:

*SELECT...*(*SELECT...*ИмяВычПоля*FROM..WHERE...;*

Обязательным условием при этом является то, чтобы внутренняя инструкция *SELECT* для каждой отбираемой по внешней SQL-инструкции записи возвращала не более чем одну запись но не более чем одному полю.

Приведем пример подобного запроса, отбирающего все записи по полю «Марка» из таблицы «Товары», с формированием дополнительного поля «Категория», значения которого возвращаются внутренней инструкцией *SELECT* из таблицы «ТипыТоваров» при условии совпадения значения поля «КодТипа» из таблицы «Товары» для текущей записи внешней SQL-инструкции с аналогичным полем «КодТипа» в записях таблицы «ТипыТоваров»:

```
SELECT  Товары.Марка, (SELECTТипыТоваров.Категория FROMТипы
                    WHERE(Товары.КодТипа=ТипыТоваров.КодТипа);)
                    AS Категория
FROMТовары;
```

Нетрудно заметить, что целью использования в данном примере внутренней инструкции *SELECT* является формирование в наборе данных, отбираемых из таблицы на стороне «Многие», дополнительного поля по значению какого-либо поля из связанной таблицы на стороне «Один». Исходя из этого, альтернативным способом решения данной задачи может быть использование запроса на внутреннее соединение:

```
SELECTТовары.Марка,ТипыТоваров.Категория
FROMТовары INNER JOINTТипыТоваровONТовары.Код-Типа = ТипыТоваров.КодТипа;
```

В тех случаях, когда отбор данных внешней SQL-инструкцией осуществляется из таблицы на стороне «Один», внутренняя инструкция *SELECT* может использоваться для дополнительного поля, формируемого на основе групповой операции по группам соответствующих связанных записей в таблице на стороне «Многие». Приведем пример отбора записей по полю «Категория» из таблицы «ТипыТоваров» с дополнительным полем «СредняяЦена», формируемым на основе статистической функции *AVG* по полю «Цена» для групп связанных записей в таблице «Товары»:

*SELECT* ТипыТоваров.Категория, *(SELECT* Avg(Товары.Цена)
*AS* СредняяЦена
*FROM*Товары
*GROUPBY*Товары.КодТипа
*HAVING*(ТипыТоваров.КодТипа=Товары.КодТипа);)*AS*СредняяЦена
*FROM*ТипыТоваров;

Опять-таки отметим, что альтернативным решением по данному примеру может быть использование следующего запроса на внутреннее соединение:

*SELECT* ТипыТоваров.Категория, Avg(Товары.Цена) AS СредняяЦена
*FROM*ТипыТоваров *INNER JOINT*Товары *ON* ТипыТоваров.КодТипа=Товары.КодТипа
*GROUP BY* ТипыТоваров.Категория;

### 4.3.2.5. Оптимизация запросов

Как уже отмечалось, запросы, являющиеся предписаниями по обработке данных, *интерпретируются* (или компилируются, т. е. переводятся) *машиной данных* СУБД в машинные коды и выполняются. При этом, однако, *декларативный* характер языка SQL («что сделать») приводит к *неоднозначности* в определении *конкретной схемы* и *конкретного порядка обработки данных* (наличию нескольких вариантов «как сделать»). Под ***оптимизацией*** запросов понимается *такой способ обработки запросов, когда по начальному представлению запроса вырабатывается процедурный план его выполнения, наиболее оптимальный при существующих в базе данных управляющих структурах.** Оптимизация осуществляется в соответствии с *критериями,* заложенными в *оптимизатор процессора* запросов СУБД (см. рис. 2.1).

* Кузнецов С.Д. Введение в СУБД. // СУБД. — № 4. — 1995. — С. 98.

В общей ***схеме обработки запроса*** выделяют:

- лексический и синтаксический разбор запроса;
- логическую оптимизацию;
- семантическую оптимизацию;
- построение процедурных планов выполнения запросов и выбор оптимального;
- непосредственное выполнение запроса.

***Лексический и синтаксический разбор*** запроса формируют *внутреннее представление запроса,* содержащее вместо имен таблиц, полей и связей базы данных их истинные внутренние идентификаторы и указатели, находящиеся в системном каталоге базы данных.

***Логическая оптимизация запроса*** может включать различные *эквивалентные преобразования,* «улучшающие» представление запроса. Такие преобразования можно разбить на *три* группы:

- преобразования предикатов сравнения;
- преобразования порядка реляционных операций (соединения, объединения, выборки);
- приведение запросов с подчиненными запросами к запросам на соединение *(JOIN).*

*Преобразования предикатов сравнения,* улучшающие в целях оптимизации представление запроса, в свою очередь, разделяются на:

- приведение предикатов сравнения к каноническому виду;
- приведение логического условия сравнения к каноническому виду.

*Каноническим* называется такой вид предикатов сравнения, который содержит сравнение *простых выражений.* Можно выделить три типа таких сравнений:

- *Имя поля Операция сравнения Константное арифметическое выражение;*
- *Имя поля Операция сравнения Арифметическое выражение;*
- Арифметическое выражение Операция сравнения Константное арифметическое выражение.

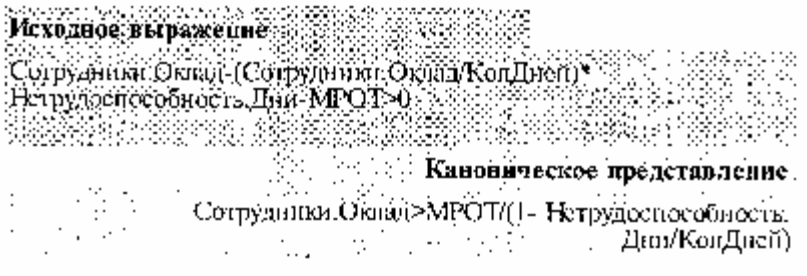
В первом типе под *«Константным арифметическим выражением»* понимается такое выражение, которое содержит *константы* и так называемые *объемлющие переменные,* в любой момент имеющие одинаковое значение в отношении всех

кортежей таблицы (базы данных). Примером приведения предиката сравнения к первому оптимизируемому каноническому виду является следующее выражение (сотрудники, у которых оклад в десять раз превышает величину минимального размера оплаты труда):


где МРОТ — объемлющая переменная, определяющая величину минимального размера оплаты труда.

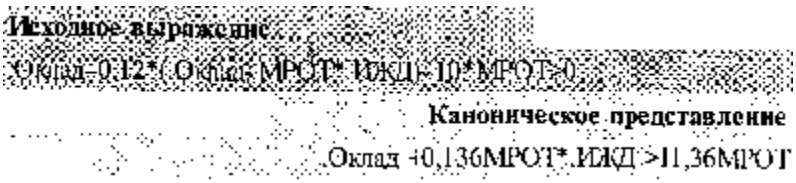
На этом примере легко понять суть последующей оптимизируемости канонических представлений. Правая часть такого сравнения одинакова для всех просматриваемых записей-кортежей и определяется (вычисляется) один раз для всех. В исходном выражении помимо собственно операции сравнения необходимо при выборке каждого кортежа производить арифметические вычисления, что существенно увеличивает количество операций при выполнении соответствующего запроса.

Во втором типе под *«Арифметическим выражением»* понимается такое выражение, в котором может присутствовать имя поля другой таблицы, полностью раскрыты скобки, произведено приведение и упорядочение членов. Примером приведения предиката сравнения ко второму оптимизируемому каноническому виду является следующее выражение (сотрудники, оклад которых с учетом вычетов по болезни больше величины минимальной оплаты труда):



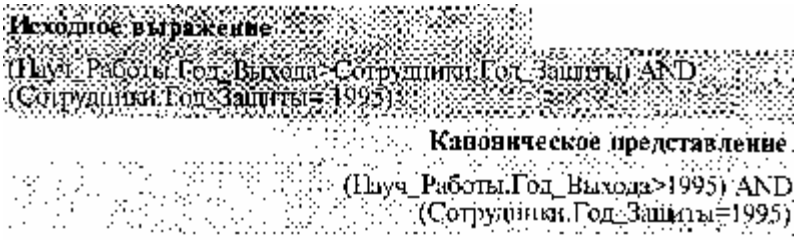

где КолДней — переменная, равная количеству рабочих дней в данном месяце.

Примером приведения предиката сравнения к третьему оптимизируемому каноническому виду является следующее выражение (сотрудники, чей оклад после вычета подоходного налога с учетом льгот на иждивенцев в десять раз превышает величину минимальной оплаты труда):

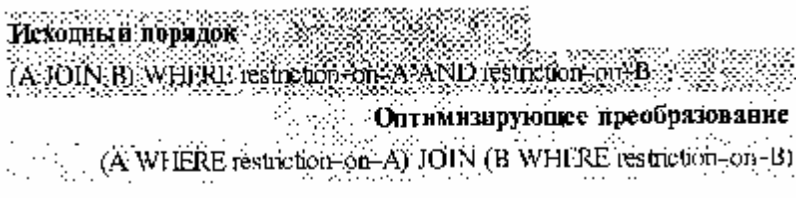



где ИЖД — имя поля той же таблицы «Сотрудники», с данными по количеству иждивенцев для конкретного сотрудника; названия таблицы «Сотрудники» для краткости опущены.

*Приведение логических условий сравнения к каноническому виду* преследует ту же цель снижения числа операций при выполнении запроса на основе поиска общих предикатов и различных упрощений логических выражений. Для примера можно привести следующее оптимизируемое логическое выражение (научные работы, которые вышли после защиты их авторами диссертаций, защищенными в 1995 г.):




*Преобразования порядка реляционных операций* также направлены на сокращение возможного количества операций при обработке запросов. Одними из наиболее частых реляционных операций в запросах являются операции *соединения (JOIN)* и операции *ограничения (WHERE* restriction). В этом отношении общим правилом оптимизирующего преобразования запросов будет замена последовательности операции соединения с последующими ограничениями на предварительные ограничения с последующим соединением:

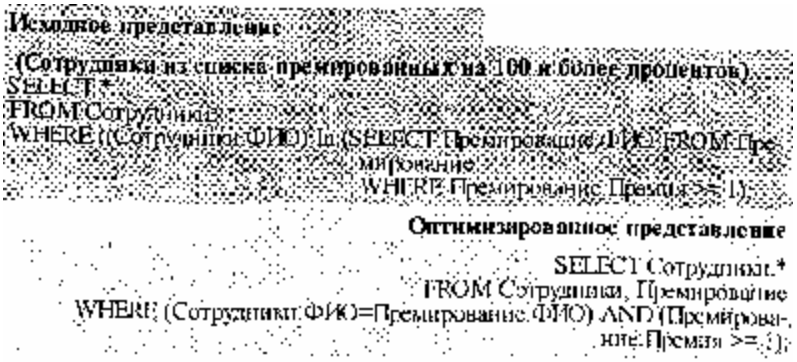



где A и B — имена таблиц.

Очевидно, что в большинстве случаев количество операций по реализации наиболее затратной операции соединения таблиц будет меньше после предварительно проведенных операций ограничения по отбору записей из исходных таблиц. Особенно данное правило актуально при наличии ограничений на отбор полей при соединении таблиц, связанных отношений «Один-ко-многим». В этом случае перенос ограничений в условиях отбора на таблицу, находящуюся на стороне «многие» до операции *JOIN* может существенно ускорить выполнение запроса.

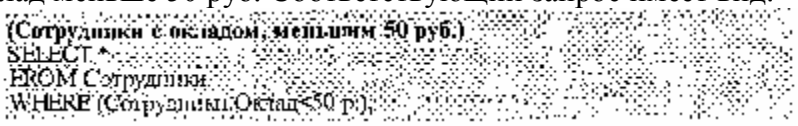
Одним из проявлений вариантности языка SQL является ***эквивалентность*** *выражения некоторых подчиненных и сложных запросов с соединениями.* При выполнении запросов с подчиненными запросами для каждого кортежа-записи в исходном наборе внешнего запроса выполняется подчиненный запрос. Иначе говоря, всякий раз при вычислении предиката внешнего запроса вычисляется подчиненный запрос. Поэтому резервом для оптимизации подобных запросов является поиск возможных путей сокращения количества операций за счет эквивалентных преобразований, приводящих к совмещению операций формирования набора кортежей-записей внешнего и внутреннего (подчиненного) запроса.

*Каноническим представлением запроса* по данным из n таблиц называется запрос, содержащий n–1 предикат соединения и не содержащий предикатов с подчиненными запросами. Если вернуться к примеру подчиненного запроса с предикатом In на рис. 4.24, то его эквивалентным оптимизируемым выражением будет следующее:




*Логическая* оптимизация запросов не учитывает *семантики* конкретной базы данных, проявляемой в ограничениях целостности на значения полей таблиц и связей между ними. В результате ядро СУБД всякий раз при выполнении логически оптимизированного запроса еще и проверяет ограничения целостности. Часть записей-кортежей, сформированных по результатам операций запроса, при этом может быть отвергнута именно по ограничениям целостности. ***Семантическая оптимизация*** запросов основывается на *слиянии внутреннего представления запроса и ограничений целостности* конкретной базы данных *до непосредственного выполнения запроса* и призвана за счет совместной проверки ограничений целостности и условий запроса сократить количество выполняемых операций.

Для примера предположим, что в таблице «Сотрудники» по полю «Оклад» наложено ограничение целостности, заключающееся в том, что оклад не может быть меньшим величины минимального размера оплаты труда МРОТ, равного 84 руб. Предположим также, что нужно сформировать список сотрудников, чей оклад меньше 50 руб. Соответствующий запрос имеет вид:



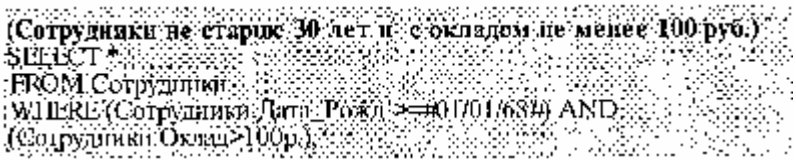

Без семантической оптимизации данный запрос будет выполняться следующим образом — будет последовательно извлекаться каждая запись в таблице «Сотрудники» и проверяться на выполнение условия отбора. Результатом выполнения запроса будет пустое множество записей. С учетом

внутренней семантической оптимизации в ответ на запрос без последовательного перебора всех записей сразу будет выдано пустое множество записей.

После логической и семантической оптимизации строится процедурный план выполнения запросов. ***Процедурным планом запроса*** называется *детализированный порядок выполнения операций доступа к базе данных физического уровня.* Уже упоминавшаяся многовариантность способов выполнения SQL-инструкций соответственно приводит к набору альтернативных процедурных планов выполнения запросов, среди которых оптимизатор запросов ядра СУБД должен выбрать оптимальный в соответствии с определенными *критериями.*

Общепринятым критерием оптимальности процедурных планов является *минимизация* ***стоимости выполнения запросов.*** При этом под стоимостью выполнения запроса понимаются вычислительные ресурсы (ресурсы процессора и ресурсы дисковой и оперативной памяти), необходимые для выполнения запросов.

Для иллюстрации вариантности процедурных планов рассмотрим запрос по выборке записей из таблицы «Сотрудники» по возрасту не старше 30 лет и с должностным окладом более 100руб.:




Если по полям «Дата_Рожд» и «Оклад» таблицы «Сотрудники» существуют индексы, то возможны *три варианта* плана выполнения запроса:

1) последовательно без учета индексации просматривать (сканировать) записи таблицы «Сотрудники» и отбирать записи при выполнении требуемых условий;
2) сканировать индекс поля «Дата_Рожд» с условием выборки «>=#01/01/68#», выбирать соответствующие записи из таблицы «Сотрудники» и среди них отбирать те, которые удовлетворяют условию по полю «Оклад»;
3) сканировать индекс поля «Оклад» с условием выборки «>100 руб.», выбирать соответствующие записи из таблицы «Сотрудники» и среди них отбирать те, которые удовлетворяют условию по полю «Дата_Рожд».

Стоимость каждого варианта в конечном счете определяется главным образом количеством пересылаемых страниц (блоков) из файла данных в буферы оперативной памяти ввиду того, что, как уже отмечалось, время операций доступа к конкретным записям в оперативной памяти на несколько порядков меньше времени процессов обмена между внешней и оперативной памятью, и тем самым основные затраты приходятся именно на эту операцию.

Если количество записей в таблице «Сотрудники» невелико и все они умещаются в одной странице (в одном блоке) файла базы данных, то наименее затратным будет первый вариант. Если записи таблицы «Сотрудники» распределены по множеству страниц, менее затратными являются 2-й и 3-й варианты. При этом различия между ними будут определяться так называемой селективностью значений по полям «Дата_Рожд» и «Оклад».

*Селективность* определяется главным образом характером статистического распределения значений по соответствующим полям. Исходя из мощности (количества записей), вида (равномерное, нормальное) и параметров распределения (среднее значение, максимальное и минимальное значение) можно получить *приблизительные оценки количества страниц* (блоков) файла базы данных, *пересылка которых потребуется в оперативную память* в ходе выполнения запроса. Так, если по приведенному примеру имеются некоторые априорные или апостериорные данные о том, что распределение значений по возрасту сотрудников является нормальным со средним значением 27 лет, а распределение величин должностных окладов является равномерным в интервале от 50 руб. до 500 руб., то, очевидно, наименее затратным будет 2-й вариант процедурного плана выполнения запроса, так как потребует меньшего количества пересылок страниц файла базы данных.

***Стратегии оптимизатора*** *по оценкам стоимости* выполнения запросов могут быть *упрощенными* (когда статистические распределения любых полей являются по умолчанию равномерными) либо более *сложными.* В сложных стратегиях при ведении базы данных (добавление, удаление, изменение записей) осуществляется мониторинг за параметрами апостериорных статистических распределений значений по полям базы данных (отслеживается минимальное, максимальное, среднее значение и другие параметры). В этом случае оценки стоимости процедурных планов выполнения запросов

являются более точными, и тем самым повышается эффективность оптимизации запросов и в целом эффективность обработки данных.

### 4.3.3. Процедуры, правила (триггеры) и события в базах данных

Рассмотренные выше способы обработки данных через *запросы, фильтрацию, поиск* и *сортировку* данных реализуют достаточно *простые информационные потребности* пользователей АИС либо являются лишь отдельными элементами в последовательности взаимоувязанных операций при решении сложных информационных задач в предметной области АИС. План последовательности таких операций, отражающий определенный алгоритм реализации информационных задач, может быть достаточно сложным, и каждый раз при возникновении соответствующей информационной потребности должен создаваться (воспроизводиться) и реализовываться пользователем заново.

К примеру, одной из задач АИС по делопроизводству могут быть функции организации, а также контроля за прохождением и отработкой входящих документов. Реализация этой функции может осуществляться по следующему алгоритму:

- при появлении новых записей в таблице «документы» с категорией «входящие» сформировать набор записей входящих документов за определенный период времени, скажем за рабочий день;
- известить пользователей АИС, имеющих полномочия на принятие резолюций по входящим документам (руководители организации или их секретариаты), и предоставить им доступ к сформированному набору данных;
- получить результаты резолюций на входящих документах, известить и предоставить соответствующие документы пользователям-исполнителям, включить контроль на исполнение документов;
- получить от пользователей-исполнителей данные по исполнению документа и снять соответствующие документы с контроля либо известить пользователей, наложивших соответствующие резолюции, о не исполнении к установленному сроку их резолюций.

Реализация таких *сложных алгоритмов обработки данных* в ранних СУБД осуществлялась через создание и постоянное выполнение в АИС прикладных программ на языках высокого уровня (Фортран, Кобол, С), которые *постоянно опрашивали базу данных на предмет обнаружения соответствующих ситуаций* и реализовывали сложные алгоритмы обработки. Как и в случае с простыми запросами до появления языков баз данных, такой подход требует квалифицированных посредников-программистов между пользователем и базой данных и, кроме того, обусловливает значительные вычислительные затраты на функционирование АИС.

Разработчиками известной СУБД SyBase был предложен другой оригинальный подход, который можно проиллюстрировать следующей схемой:

Функционирование базы данных согласно приведенной схеме осуществляется следующим образом:


*Рис. 4.26. Принцип механизма событий, правил и процедур в базах данных*

1. В базе данных определяются так называемые ***события*** (database events), ***связанные с изменениями данных —*** добавление новой записи (ей) в определенную таблицу, изменение записи(ей), удаление записи (ей).* Для реализации механизма событий в языке SQL введены специальные конструкции (Create DBEvent «Имя» — создать событие, Exec SQL Get DBEvent — получить событие и т.д.);
2. *Для каждого события в базе* данных определяются ***правила*** (triggers) ***по проверке определенных условий состояния данных.*** Соответственно в SQL введены конструкции для описания правил (Create Rule «Имя» — создать правило);
3. *В зависимости от результатов проверки правил* в базе данных запускаются на выполнение предварительно определенные ***процедуры.*** Процедуры представляют собой *последовательности команд по обработке данных, имеющие отдельное смысловое значение,* и могут реализовываться на упрощенном макроязыке (последовательность команд запуска запросов или выполнения других действий, например по открытию-закрытию форм, таблиц и т. п.) или на языке 4GL, встроенном в СУБД.

* События могут быть также и производные от перечисленных событии, например, событие, заключающееся в том, что какая-либо транзакция (процесс, пользователь) намеревается изменить запись (но еще не изменила)—так называемое

событие «До обновления». Аналогично, могут быть определены события «После обновления», «До удаления», «После удаления» и т. п.

В последнем случае процедуры представляют собой подпрограммы для осуществления сложных операций обработки и диалога с пользователем. В язык SQL также введены специальные конструкции описания процедур (Create Procedure «Имя» — создать процедуру).

Суть идеи механизма *событий, правил* и *процедур* заключается в том, что они после определения *хранятся(!!!)** вместе с данными. Соответствующие конструкции введены в стандарт SQL — SQL2. Ядро СУБД *при любом изменении состояния базы данных проверяет, не произошли* ли ранее «поставленные на учет» *события,* и, если они произошли, обеспечивает *проверку* соответствующих *правил* и *запуск* соответствующих *процедур* обработки.

* То есть зарегистрировать для автоматической обработки в БД.

В отличие от традиционного подхода, когда специальные прикладные программы постоянно «опрашивают» базу данных для обнаружения ситуаций, требующих обработки, «событийная» техника более экономична и естественна в технологическом плане. Кроме того, SQL-инструкции, реализующие технику «событий-правил-процедур» в некоторых СУБД с развитым интерфейсом, могут быть созданы, так же как и сложные запросы, через специальные конструкторы и мастера, что дает возможность их освоения и использования пользователями-непрограммистами.

В настоящее время механизм событий, правил и процедур широко распространен и в той или иной мере реализован практически в любой современной СУБД.

### 4.3.4. Особенности обработки данных в СУБД с сетевой моделью организации данных

Основные принципы и способы обработки данных, рассмотренные для реляционных СУБД, также характерны и для немногих сохранившихся, и продолжающих развиваться СУБД с сетевой моделью организации данных.

В сетевых СУБД подобным же образом, как и в реляционных СУБД, реализуются операции поиска, фильтрации и сортировки данных. Распространенность и популярность *языка SQL* реляционных СУБД привели к тому, что *подобные языки* для реализации запросов к базам данных были разработаны или просто *«внедрены» в сетевые СУБД.* При этом так же, как и реляционные СУБД, современные сетевые СУБД предоставляют пользователю и специальные диалогово-наглядные средства формирования запросов. Также в сетевые СУБД встраиваются специальные макроязыки для формирования сложных последовательностей взаимосвязанных запросов (аналог процедур), хранящихся вместе с базой данных.

Вместе с тем обработка данных в СУБД с сетевой моделью организации данных, как уже отмечалось, характеризуется уже упоминавшейся принципиальной особенностью, которой нет в реляционных СУБД. Это непосредственная ***«навигация»*** по связанным данным (по связанным записям) в разных информационных объектах (аналоги таблиц в реляционных СУБД). Как уже отмечалось, возможность непосредственной навигации обусловлена тем, что в сетевых СУБД ссылки-связи между записями различного типа (различных таблиц) задаются не через внешние ключи, а через специальные указатели на физические адреса расположения связанных записей.

Просматривая, к примеру, в сетевой СУБД записи объекта «Лицо» и выбрав запись «Иванов» (т. е. поместив табличный курсор на соответствующую запись), можно через активизацию поля «Работает» вызвать на экран поля связанной записи в объекте «Организация» и просмотреть соответствующие данные, а далее, при необходимости, через активизацию поля «Адрес» в записи по объекту «Организация» вызвать и просмотреть данные по дислокации места работы сотрудника «Иванов» и т. д. (см. рис. 4.27).

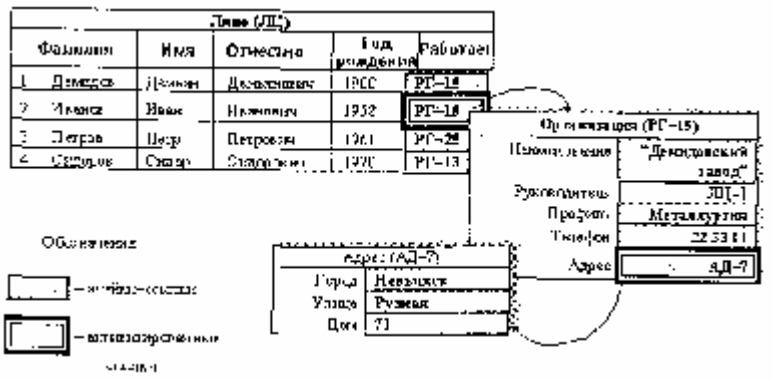


*Рис. 4.27. Навигация по связанным записям в сетевых СУБД*

В реляционных СУБД для реализации такого просмотра понадобилось бы создать и выполнить запрос на выборку данных из трех таблиц на основе внутреннего *(INNER JOIN)* соединения при условии отбора соответствующей фамилии сотрудника:

*SELECT* Лицо.*, Организация.*, Адрес.*

*FROM*(Лицо*INNERJOIN*(Адрес*INNERJOIN*Организация*ON*Адрес.№№=Организация.Адрес)

*ON*(((Лицо.Работает = Организация.Наименование)

*WHERE*(((Лицо.Фамилия)=»Иванов»));

При этом, если пользователю необходимо посмотреть те же данные, но для другого сотрудника, то необходимо изменить условия отбора по фамилии и заново выполнить запрос. В сетевых же СУБД для этого достаточно лишь «вернуться» в исходный объект «Лицо», переместить курсор на другую запись и повторить навигацию.

Данный пример показывает, что *навигационные возможности сетевых СУБД* позволяют пользователю реализовывать свои *информационные потребности* («беседовать» с базой данных) более *естественным интерактивным способом,* шаг за шагом уточняя свои потребности, и тем самым более глубоко и наглядно анализировать (изучать) данные.

Навигационный подход к анализу и просмотру данных, естественный уже для ранних сетевых СУБД, впоследствии (в конце 80-х годов) был реализован в технике гипертекста, и в созданной на его основе новой разновидности документальных информационных систем — гипертекстовых информационно-поисковых систем.

Вместе с тем навигация по связанным данным порождает и ряд своих *специфических проблем,* таких как «потеря ориентации» и трудности с ***визуализацией цепочек «пройденных» информационных объектов*** (записей). Схемы баз данных, отражающих сложные предметные области, могут насчитывать десятки различных информационных объектов и еще большее количество связей между ними. В результате такие базы данных представляют сложное многомерное информационное пространство из множества разнотипных наборов записей, пронизанных и опутанных порой несметным количеством связей. «Путешествуя» в таком клубке, легко «сбиться с пути», потерять общую картину состояния данных.* При этом следует иметь в виду, что особенности человеческого мышления таковы, что человек способен удержать в представлении с полным отслеживанием всех связей и нюансов не более 3-4 сложных объектов.**

---

* То есть оказаться в ситуации, которая образно выражается известной поговоркой «За деревьями леса не видно».

** Этим, кстати, еще из древности определяется троичная система организации структуры воинских подразделений для эффективного управления ситуацией на поле боя.

Иногда объектом анализа являются не конкретные реквизиты связанных записей, а сама *схема связанных записей,* т. е. визуализированная цепочка имен связанных от исходного информационного объекта записей. Использование *множественного типа значений в полях* информационных объектов сетевых СУБД позволяет реализовывать все типы связей, что приводит к «пучковости» исходящих или входящих связей типа «один-ко-многим», «многие-ко-многим». Визуализация таких цепочек на двумерном экране компьютера может представлять существенные графические сложности.

На рис. 4.28 для примера приведен вариант изображения цепочки связанных записей с корневой записью «Иванов» объекта «Лицо». Такая визуализация позволяет быстро составить общее представление о полученном образовании, трудовой деятельности и проживании данного лица. При этом длина цепочки ограничена тремя последовательно связанными записями, но, как видно из

рисунка, и в этом, в общем-то простом для многих жизненных ситуаций случае, достаточно сложно отобразить общую схему связей, не «запутывая» ее восприятие.

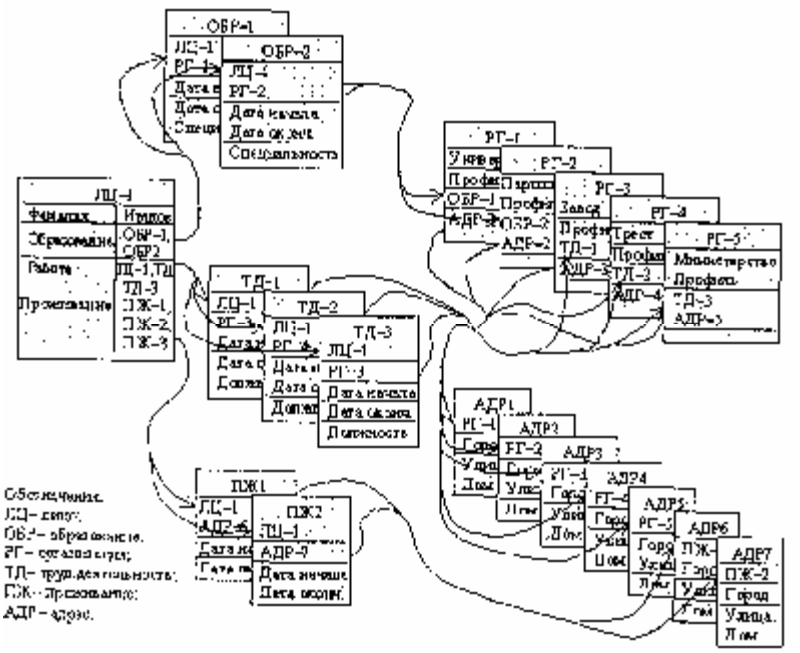


*Рис. 4.28. Пример визуализации цепочки связанных записей*

Навигация по связанным записям в реляционных СУБД открывает новые возможности анализа данных на основе иных, нежели реляционные, семантических принципах. В частности, становится возможным реализация процедур поиска и построения смысловых окрестностей какой-либо записи по ее связям в базе данных, применение различных процедур информационного анализа на основе алгоритмов поиска на графах и т. п.

Одним из направлений развития современной теории и техники СУБД является линия объектно-ориентированных СУБД, которые на витке наработанных в конце 70-х и в 80-х годах решений по реляционным СУБД, обеспечивают новые возможности по обработке данных на основе методов навигации и визуализации, впервые представленных в сетевых СУБД.

## 4.4. Вывод данных

Результаты обработки данных должны использоваться в том порядке и в тех формах, которые приняты в предметной области АИС. Решение этой задачи обеспечивается комплексом функций СУБД, определяемым термином «вывод данных».

В более узком плане под ***выводом данных*** понимается *комплекс функций СУБД по предоставлению пользователю результатов обработки, хранения и накопления данных в наиболее удобном для восприятия виде, документирования выводимой информации, а также по передаче данных в другие (внешние) системы и форматы.*

Вывод данных осуществляется:

- через выходные (выводные) формы;
- через «отчеты»;
- через экспорт данных.

***Выходные формы*** по смыслу аналогичны входным формам, т. е. формам для ввода, просмотра и редактирования данных. При этом, как правило, *базовым источником дачных* для форм являются не таблицы данных, а результаты выполнения *запросов.* Таким образом, главной функцией выводных форм является предоставление пользователю результатов выполнения запросов в наиболее удобном и привычном «бланковом» виде.

В отличие от входных, особенностью выводных форм как экранных объектов является то, что помимо надписей и полей с данными в них присутствуют так называемые ***элементы управления—****кнопки, переключатели, поля-списки,* которые используются для задания пользователем тех или иных параметров выполнения запросов. В развитых СУБД *запросы с параметрами* реализуются через технику форм, в которых пользователь через элементы управления определяет конкретные условия отбора. На рис. 4.29 приведен пример формы для отображения запроса, формирующего список

командировок сотрудников, в зависимости от выбранного пользователем через переключатели года и в соответствующем списке месяца.

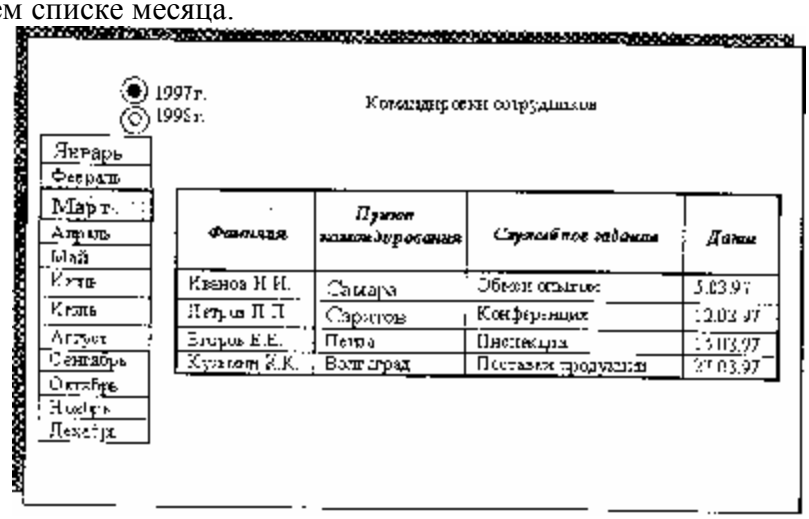



*Рис. 4.29. Пример формы для реализации и отображения результатов запроса с параметрами*

***Отчеты*** решают задачу ***документирования выводимых данных,*** т. е. *представления результатов обработки и накопления данных в форме текстового документа,* который можно распечатать или приобщить к другому текстовому документу. Отчеты во многом аналогичны выводным формам и, по сути, представляют *печатные формы* для результатов накопления и обработки данных. Отличительной особенностью отчетов является то, что они строятся по правилам текстовых документов, т. е. отображаемые данные разделяются на страницы и разделы с соответствующими элементами (поля, колонтитулы) и параметрами форматирования (шрифт, отступы, выравнивание).

Так же как и в формы, в отчеты могут помещаться *элементы управления,* среди которых особое значение имеют *вычисляемые поля,* т. е. поля, содержимое которых формируется на основе вычисления определенных статистических функций по помещаемым в отчет данным. Для примера на рис. 4.30 приведен отчет для вывода данных по командировкам сотрудников. Поля с надписями «ИТОГО» являются как раз вычисляемыми элементами через функцию «Сумма» поданным в поле «Аванс».

Помимо полей с данными и вычисляемых полей в отчеты могут внедряться различные графические объекты для формирования логотипов и других поясняющих рисунков и, кроме того, такие средства наглядного отображения табличных данных, как диаграммы. На рис. 4.30 во втором разделе отчета приведена диаграмма, отображающая распределение итогового аванса на командировочные расходы сотрудников известной конторы по месяцам первого квартала.

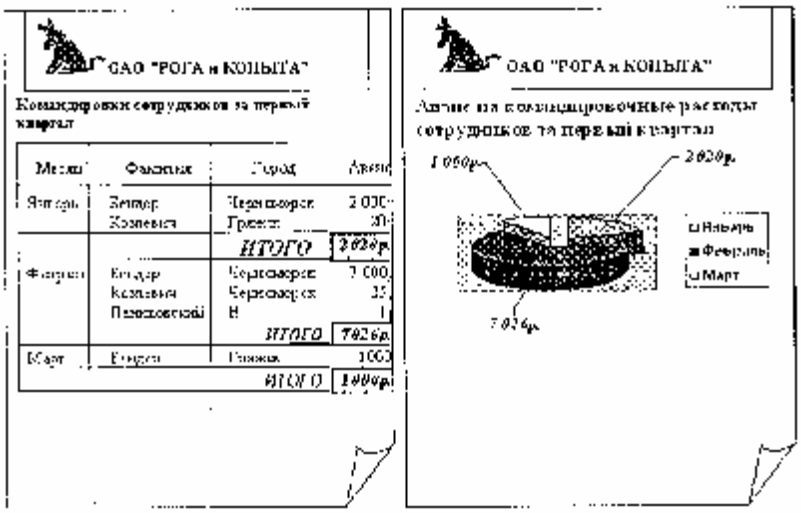



*Рис. 4.30. Пример отчета*

***Экспорт данных*** решает *технологические задачи резервирования, архивирования данных* или *передачи* накопленных в АИС *данных во внешние системы* и *форматы* и реализуется через уже *рассмотренные запросы на добавление* данных и *запросы на создание таблицы*. Таблицы-приемники в этом случае находятся во внешних файлах баз данных, созданных под управлением СУБД того же типа или СУБД, поддерживающей протокол ODBC.

Некоторые СУБД обеспечивают ***экспорт данных в текстовые файлы.*** При этом табличные данные в строках текстовых файлов размещаются последовательно по строкам и ячейкам экспортируемой таблицы, т. е. слева направо, сверху вниз, отделяясь друг от друга специальными разделителями, например символом « \ ».* Такой порядок размещения табличных данных в текстовых файлах получил название «***унифицированного формата обмена данными***» **(УФОД).** Соответственно, как уже отмечалось при рассмотрении вопроса по вводу данных, многие СУБД имеют специальные режимы не только экспорта, но и импорта данных их текстовых файлов, построенных на основе УФОД-формата.

* Слэш налево.

**Вопросы и упражнения**

1. Кем и в каких целях применяется язык SQL в реляционных СУБД?
2. Какова структура и каковы функции структурных элементов SQL-ииструкций?
3. Что включают и в каких целях используются «включающие» языки?
4. Поясните процесс «открытия» таблиц и форм. Что происходит при этом с данными?
5. В чем преимущества и недостатки представления и отображения данных в табличном виде и в виде экранных форм?
6. В текстовых и табличных редакторах изменения данных (корректировка, добавление, удаление) фиксируются во внешней памяти в момент закрытия файлов (если не было явной предварительной команды «Сохранить»). Каков порядок фиксации изменений данных в таблицах СУБД?
7. В чем сходства и различия фильтрации данных и запросов на выборку данных?
8. Постройте запрос по формированию списка студентов 1980 года рождения с реквизитами — ФИО, Уч. Группа, Дата рождения, из таблицы «Студенты» (№№, ФИО, Уч. Группа, Дата рождения, Год поступления). К какому типу относится данный запрос?
9. Постройте запрос по формированию списка сотрудников руководящего звена не старше 35 лет, с окладом свыше 1000 р. и с полным набором реквизитов из таблицы «Сотрудники» (Таб.№, ФИО, Должность — Начальник отдела, Зам. начальника отдела, Начальник сектора, Ведущий инженер. Старший инженер, Инженер, Техник, Оклад, Дата Рождения). К какому типу относится данный запрос?
10. Интерпретируйте на естественном языке следующую SQL-инструкцию:

```
SELECTСотрудники.Таб. —№,Сотрудиики.Фамилия,Сотрудники.Имя
FROM Сотрудники
WHERE ((Сотрудники.Должность=«Инженер» Or Сотрудники.—Должность=«Методист») AND
(Сотрудники.Оклад> 100р.));
```

11. Постройте запросы по формированию списка организационных форм, списка профилей деятельности и списка сочетаний организационной формы с профилем деятельности организаций из таблицы «Организации»—Код, Код ОКПО, Наименование, Условное наименование, Профиль деятельности (Производственный, Коммерческий, Посреднический, Научно-производственный), Организационная форма (ЗАО, ОАО, и т. д.).
12. Интерпретируйте на естественном языке следующие SQL-инструкции:

```
SELECT Квартиры.№, Здания.№_Дома, Здания. Улица
FROM Квартиры INNER JOIN Здания ON Квартиры.№№_Здания = Здания. №№
WHERE (((Квартиры.Кол Комнат=1) Or (Квартиры. КолКомнат=4)) AND ((Квартиры.Этаж >=4)
        AND (Квартиры-.Этаж<=6)));
SELECT Сотрудники.Таб_№. Сотрудники. ФИО, Подразделения. Наименование,
        Sum(Нетрудоспособность. ДатаОкончания Нетрудоспособность.Дата Начала) AS
        ОбщКолНетр
FROM (Сотрудники INNER JOIN Подразделения ON Сотрудники.№_Подразделения =
        Подразделения.№№) INNER JOIN Нетрудоспособность ON Нетрудоспособность.
        №_Сотрудника = Сотрудники. Та6_№)
WHERE (Нетрудоспособность.ДатаНачала>=#01.01.1999#)
        AND(Нетрудоспособность.ДатаНача.1а<=#31.12.1999#)
        AND способность.ДатаОкончания<=#31.12.1999#)
```

GROUP BY Нетрудоспособность. №_Сотрудника;

13. Постройте запрос по формированию списка категорий фильмов видеотеки с группировкой по кинокомпаниям, и вышедших в 90-х годах из таблицы «Фильмы»—№№, Название, Режиссер, Год выхода, Кинокомпания, Категория (Комедия, Психологическая драма. Боевик, Триллер, Детектив, Мистика), Инв.№№ видеокассеты.
14. В базе данных с таблицами «Подразделения»—№№, Наименование, Руководитель; «Сотрудники»—Таб№, ФИО, №№ подразделения, Должность, «Материальные средства» — Инв.№, Наименование, Тип, №№ Подразделения, Таб № мат. ответственного сотрудника. Начальная стоимость, % амортизации, Постройте запрос по формированию списка материально ответственных сотрудников со следующим набором реквизитов — Таб. №, ФИО, Наименование подразделения. Должность.
15. В базе данных с таблицами из предыдущего примера постройте запрос по формированию перечня всех подразделений с данными по их средствам вычислительной техники при следующем наборе реквизитов—№№ подразделения. Наименование, Руководитель, Инв.№ мат. средства. Наименование мат. средства, Тип мат. средства, ФИОмат. ответственного сотрудника.
16. В базе данных с таблицами «Лицо» —№№. ФИО, Дата рождения, Месторождения, Паспортные данные; «Владение» — Код владения, №№ Лица. №№ имущества. Вид (Единоличное, Совместное), Доля, Дата приобретения. Данные документа. Дата окончания владения; «Имущество» — №№ имущества, Категория (Недвижимость, Автотранспорт, Акции, Ювелирные изделия. Художественные произведения. Бытовая техника. Земельный надел), Описание. Стоимость, постройте запрос по формированию списка лиц (ФИО, Дата рождения. Месторождения, Паспортные данные), имеющих в единоличном владении недвижимость на сумму свыше 10 000 минимальных размеров оплаты труда.
17. В базе данных с таблицами из предыдущего примера постройте запросы по формированию списка лиц (№№, ФИО, Дата рождения, Месторождения, Паспортные данные), имеющих в совместном владении земельные наделы, и дополнительными реквизитами —Доля и Стоимость доли, а также запрос по формированию сведений о самой высокой стоимости имущества по всем возможным категориям.
18. Постройте запрос по формированию списка всех запасных частей, относящихся к ходовой части со всеми реквизитами из таблицы «Запчасти» — Код, Код автомобиля. Наименование, Тип (Двигатель, Кузов, Ходовая часть. Электрооборудование, Аксессуары), Марка, Количество на складе. Цена единицы, Поставки прекращены, с дополнительным реквизитом Общая стоимость.
19. Постройте запрос по формированию набора записей со всеми реквизитами из таблицы «Преподаватели»—№№, ФИО, Уч. степень, Уч. звание. Пед. стаж. Специализация, для которых имеются вакансии по прикладной математике в таблице «Вакансии» со следующим набором реквизитов—№№, Вуз, Должность, Треб. пед. стаж. Специализация. К какому типу относится данный запрос?
20. Постройте запрос для формирования набора записей со всеми реквизитами по оборудованию из таблицы «Оборудование»—Зав.№, Производитель, Марка, Сырье, Производительность, которое может применяться на всех предприятиях, использующих в качестве сырья очищенное зерно, данные по которым приведены в таблице «Предприятие» — Наименование. Треб. производительность и,. Используемое сырье. К какому типу относится данный запрос?
21. Постройте запрос по формированию списка сотрудников с полным набором реквизитов из таблицы «Сотрудники» — Таб_№, ФИО, Должность, Подразделение, Телефон, которые входят по таблице «Штатное расписание» — Наименование должности. Категория, Оклад, в пятерку наиболее оплачиваемых должностей. К какому типу относится данный запрос?
22. Интерпретируйте на естественном языке следующие SQL-инструкции:

```
SELECT Автомобили. *
FROM Автомобили
WHERE((Автомобили.Код)
        = Any (SELECTЗапчасти.Код_автомобиля
                FROM ((Запчасти INNER JOIN Поставки ONЗапчасти.
                Код = Поставки.Код_запчасти) INNER JOIN Поставщики ON
                Поставки. Код_поставщика = Поставщики.Код)
                WHERE(Поставщики.Город=«Саратов») AND (Зап-
                части.Тип=«Стеклооборудование»););
```

*SELECT*Клиенты. *
*FROM* Клиенты *INNER JOIN* Счета *ON* Клиенты.Код = Счета.КодКлиента
*WHERE*(Счета.Остаток>=*All*(*SELECT*Товары.Стоимость
*FROM*Товары
*WHERE*Товары.Категория= «Бытовая техника»)););

23. Постройте запрос по переименованию производителя автомобилей завода «ИжМаш» в «ИжVWМаш» в таблице «Автомобили»
(Код, Производитель, Модель, ГодНачалаПроизводства, ГодПрекр Производства, Фото.
24. Оптимизируйте следующие условия отбора записей по таблицам «Имущество» и «Сотрудники»: когда налог превышает тысячу единиц минимального размера оплаты труда (МРОТ) —
Имущество. Стоимость ***СтавкаНалога — 1000*МРОТ > 0** когда десятикратная стоимость с учетом амортизации больше оклада сотрудников —
10*(*Имущество. Стоимость — Имущество. Стоимость** *Имущество*.%Износа) — *Сотрудники. Оклад* > 0
25. Оптимизируйте следующее условие по отбору записей по таблицам «Сотрудники» и «Премирование»:
сотрудники, премированные на величину более должностного оклада, равного 1000 р. —
(*Премирование.Сумма* > *Сотрудники. Оклад*) AND (*Сотрудники. Оклад* = 1000р.)
26. Согласно одному из проектов закона о декларировании расходов все операции по оплате приобретении или услуг гражданами, стоимостью свыше 1000 МРОТ, должны осуществляться только безналичным расчетом через банки, а данные по таким операциям автоматически сообщаться в налоговые органы. В базе данных АИС финансово-кредитной организации имеется и ведется таблица «Проводки» (№№, Дата/Время, Сумма, №Счета, Приход/ расход). Предложите на основе технологии «События-Правила-Процедуры» вариант построения схемы обработки данных при принятии и вступлении в силу подобного закона.

## 5. РАСПРЕДЕЛЕННЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

В некомпьютерных информационных технологиях информационные ресурсы организаций и предприятий, с одной стороны, разделены и распределены логически (по различным подразделениям, службам) и физически (находятся в различных хранилищах, картотеках, помещениях). С другой стороны, информационные ресурсы создаются и используются своей определенной частью или в целом коллективно или индивидуально. Иначе говоря, с одними и теми же документами, картотеками и прочими информационными массивами могут в рамках общего проекта или в своей части одновременно работать несколько сотрудников и подразделений.

Первоначальные подходы к созданию баз данных АИС заключались в сосредоточении данных логически и физически в одном месте — на одной вычислительной установке. Однако такая организация информационных ресурсов чаще всего является не совсем естественной с точки зрения традиционных («бумажных») информационных технологий конкретного предприятия (организационной структуры) и при внедрении АИС происходит «ломка» привычных информационных потоков и структур.* Все информационные ресурсы предприятия, организации сосредотачиваются централизованно в одном месте, что требует определенных технологических, кадровых и материальных затрат и может порождать ряд новых проблем и задач. Следует отметить, что такому подходу также способствовала и господствующая на начальном этапе автоматизации предприятий и организаций в 70-х годах тогдашняя парадигма вычислительных систем — общая мощная вычислительная установка (main frame) и групповая работа пользователей с удаленных терминалов через системы разделения времени.

* В данном контексте более понятен сам термин — «внедрение», предполагающий сопротивление.

Опыт внедрения автоматизированных систем управления в различных организационных структурах в 70-е— 80-е гг. показал не всегда высокую эффективность подобной автоматизации, когда новые технологические информационно-управленческие подразделения (отдел автоматизации, отдел АСУ, информационная служба и т. п.) и новые электронные информационные потоки зачастую функционировали вместе с сохраняющимися традиционными организационными структурами, а также вместе с традиционными («бумажными», «телескопными») информационными потоками.

Осознание подобных проблем постепенно стало приводить к мысли о распределенных информационных системах.

## 5.1. Понятие распределенных информационных систем, принципы их создания и функционирования

Впервые задача об исследовании основ и принципов создания и функционирования распределенных информационных систем была поставлена известным специалистом в области баз данных К. Дейтом в рамках уже не раз упоминавшегося проекта System R, что в конце 70-х — начале 80-х годов вылилось в отдельный проект создания первой распределенной системы (проект System R*). Большую роль в исследовании принципов создания и функционирования распределенных баз данных внесли также и разработчики системы Ingres.

Собственно в ***основе распределенных АИС*** лежат две основные идеи:

- много организационно и физически распределенных пользователей, одновременно работающих с общими данными — общей базой данных (пользователи с разными именами, в том числе располагающимися на различных вычислительных установках, с различными полномочиями и задачами);
- логически и физически распределенные данные, составляющие и образующие тем не менее единое взаимосогласованное целое — общую базу данных (отдельные таблицы, записи и даже поля могут располагаться на различных вычислительных установках или входить в различные локальные базы данных).

Крис Дейт сформулировал также ***основные принципы*** создания и функционирования распределенных баз данных. К их числу относятся:

- ***прозрачность расположения данных для пользователя*** (иначе говоря, для пользователя распределенная база данных должна представляться и выглядеть точно так же, как и нераспределенная);
- ***изолированность пользователей друг от друга*** (пользователь должен «не чувствовать», «не видеть» работу других пользователей в тот момент, когда он изменяет, обновляет, удаляет данные);
- ***синхронизация и согласованность*** (непротиворечивость) ***состояния данных*** *в любой момент времени.*

Из основных вытекает ряд *дополнительных принципов:*

- *локальная автономия* (ни одна вычислительная установка для своего успешного функционирования не должна зависеть от любой другой установки);
- *отсутствие центральной установки* (следствие предыдущего пункта);
- *независимость от местоположения* (пользователю все равно где физически находятся данные, он работает так, как будто они находятся на его локальной установке);
- *непрерывность функционирования* (отсутствие плановых отключений системы в целом, например для подключения новой установки или обновления версии СУБД);
- *независимость от фрагментации данных* (как от горизонтальной фрагментации, когда различные группы записей одной таблицы размещены на различных установках или в различных локальных базах, так и от вертикальной фрагментации, когда различные поля-столбцы одной таблицы размещены на разных установках);
- *независимость от реплицирования* (дублирования) данных (когда какая-либо таблица базы данных, или ее часть физически может быть представлена несколькими копиями, расположенными на различных установках, причем «прозрачно» для пользователя);
- *распределенная обработка запросов* (оптимизация запросов должна носить распределенный характер — сначала глобальная оптимизация, а далее локальная оптимизация на каждой из задействованных установок);
- *распределенное управление транзакциями* (в распределенной системе отдельная транзакция может требовать выполнения действий на разных установках, транзакция считается завершенной, если она успешно завершена на всех вовлеченных установках);
- *независимость от аппаратуры* (желательно, чтобы система могла функционировать на установках, включающих компьютеры разных типов);
- *независимость от типа операционной системы* (система должна функционировать вне зависимости от возможного различия ОС на различных вычислительных установках);

- *независимость от коммуникационной сети* (возможность функционирования в разных коммуникационных средах);
- *независимость от СУБД** (на разных установках могут функционировать СУБД различного типа, на практике ограничиваемые кругом СУБД, поддерживающих SQL).

* Данное свойство характеризуют также термином *«интероперабельность».*

В обиходе СУБД, на основе которых создаются распределенные информационные системы, также характеризуют термином *«Распределенные СУБД»,* и, соответственно, используют термин *«Распределенные базы данных».*

Важнейшую роль в технологии создания и функционирования распределенных баз данных играет *техника* ***«представлений»*** (Views).

Представлением называется ***сохраняемый в базе данных авторизованный глобальный запрос на выборку данных.*** Авторизованность означает возможность запуска такого запроса только *конкретно поименованным в системе пользователем.* Глобальность заключается в том, что выборка данных может

осуществляться *со всей базы данных,* в том числе из данных, расположенных на других вычислительных установках. Напомним, что результатом запроса на выборку является набор данных, представляющий временную на сеанс открытого запроса таблицу, с которой (которыми) в дальнейшем можно работать, как с обычными реляционными таблицами данных. В результате таких глобальных авторизованных запросов для конкретного пользователя создается некая ***виртуальная база данных*** *со своим перечнем таблиц, связей,* т. е. *со «своей» схемой и со «своими» данными.* В принципе, с точки зрения информационных задач, в большинстве случаев пользователю безразлично, где и в каком виде находятся собственно сами данные. Данные должны быть такими и логически организованы таким образом, чтобы можно было решать требуемые информационные задачи и выполнять установленные функции.

Схематично идея техники представлений проиллюстрирована на рис. 5.1.

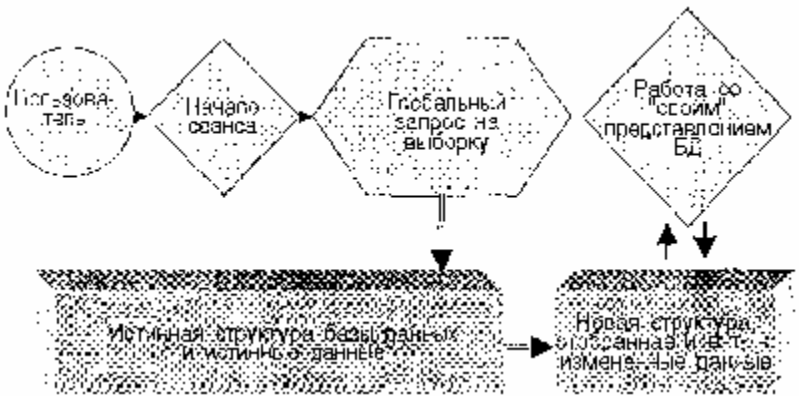



*Рис. 5.1. Основная идея техники представлений*

*При входе* пользователя в распределенную систему ядро СУБД, идентифицируя пользователя, *запускает* запросы его ранее определенного и хранимого в базе данных *представления* и формирует ему «свое» *видение базы данных,* воспринимаемое пользователем как обычная (локальная) база данных. Так как представление базы данных виртуально, то «настоящие» данные физически находятся там, где они находились до формирования представления. При осуществлении пользователем манипуляций с данными ядро распределенной СУБД по системному каталогу базы данных само определяет, где находятся данные, вырабатывает стратегию действий, т. е. определяет, где, на каких установках целесообразнее производить операции, куда для этого и какие данные необходимо переместить из других установок или локальных баз данных, проверяет выполнение ограничений целостности данных. При этом большая часть таких операций прозрачна (т. е. невидима) для пользователя, и он воспринимает работу в распределенной базе данных, как в обычной локальной базе.

Технологически в реляционных СУБД техника представлений реализуется через введение в язык SQL-конструкций, позволяющих аналогично технике «событий-правил-процедур» создавать именованные запросы-представления:

*CREATE VIEW*ИмяПредставленият *AS*
SELECT...

*FROM...*
...;

В данных конструкциях после имени представления и ключевого слова *AS* размещается запрос на выборку данных, собственно и формирующий соответствующее представление какого-либо объекта базы данных.

Авторизация представлений осуществляется применением команд (директив) *GRANT,* присутствующих в базовом перечне инструкций языка SQL (см. п. 4.1) и предоставляющих полномочия и привилегии пользователям:

*GRANT SELECTON* ИмяПредставления *TO* ИмяПользователя1, ИмяПользователя2,...;

Соответственно директива *REVOKE* отменяет уставленные ранее привилегии.

Несмотря на простоту и определенную изящность идеи «представлений», практическая реализация подобной технологии построения и функционирования распределенных систем встречает ряд серьезных *проблем.* Первая из них связана с размещением ***системного каталога*** базы данных, ибо при формировании для пользователя «представления» распределенной базы данных ядро СУБД в первую очередь должно «узнать», где и в каком виде в действительности находятся данные. Требование отсутствия центральной установки приводит к выводу о том, что системный каталог должен быть на любой локальной установке. Но тогда возникает *проблема обновлений.* Если какой-либо пользователь изменил данные или их структуру в системе, то эти изменения должны отразиться во всех копиях системного каталога. Однако размножение обновлений системного каталога может встретить трудности в виде недоступности (занятости) системных каталогов на других установках в момент распространения обновлений. В результате может быть не обеспечена непрерывность согласованного состояния данных, а также возникнуть ряд других проблем.

Решение подобных проблем и практическая реализация распределенных информационных систем осуществляется *через отступление* от некоторых рассмотренных выше *принципов создания и функционирования распределенных систем.* В зависимости оттого, какой принцип приносится в «жертву» (отсутствие центральной установки, непрерывность функционирования, согласованного состояния данных и др.) выделились *несколько* самостоятельных *направлений* в технологиях распределенных систем — *технологии «**Клиент-сервер**», технологии **реплицирования**, технологии **объектного связывания**.*

Реальные распределенные информационные системы, как правило, построены на основе сочетания всех трех технологий, но в методическом плане их целесообразно рассмотреть отдельно. Дополнительно следует также отметить, что техника представлений оказалась чрезвычайно плодотворной также и в другой сфере СУБД—защите данных. Авторизованный характер запросов, формирующих представления, позволяет предоставить конкретному пользователю те данные и в том виде, которые необходимы ему для его непосредственных задач, исключив возможность доступа, просмотра и изменения других данных.

## 5.2. Технологии и модели «Клиент-сервер»

Системы на основе технологий ***«Клиент-сервер»*** исторически выросли из *первых централизованных многопользовательских автоматизированных информационных систем,* интенсивно развивавшихся в 70-х годах (системы main frame), и получили, вероятно, наиболее широкое распространение в сфере информационного обеспечения крупных предприятий и корпораций.

В технологиях «Клиент-сервер» *отступают* от одного из *главных принципов* создания и функционирования распределенных систем — ***отсутствия центральной установки.*** Поэтому можно выделить *две* основные *идеи,* лежащие в основе клиент-серверных технологий:

• общие для всех пользователей данные на одном или нескольких серверах;

• много пользователей ***(клиентов)*** на различных вычислительных установках, совместно (параллельно и одновременно) обрабатывающих общие данные.

Иначе говоря, системы, основанные на технологиях *«Клиент-сервер», распределены только в отношении пользователей,* поэтому часто их не относят к «настоящим» распределенным системам, а считают отдельным, уже упоминавшимся *классом* ***многопользовательских систем.***

Важное значение в технологиях «Клиент-сервер» имеют понятия сервера и клиента.

Под ***сервером*** в широком смысле понимается любая *система, процесс, компьютер, владеющие* каким-либо *вычислительным ресурсом* (памятью, временем, производительностью процессора и т. д.).

***Клиентом*** называется также любая *система, процесс, компьютер, пользователь, запрашивающие у сервера* какой-либо *ресурс, пользующиеся* каким-либо *ресурсом или обслуживаемые сервером иным способом.*

В своем *развитии* системы «Клиент-сервер» прошли несколько *этапов,* в ходе которых сформировались различные ***модели*** систем «Клиент-сервер». Их реализация и, следовательно, правильное понимание основаны на *разделении структуры СУБД на три компонента:*

- *компонент представления,* реализующий функции ввода и отображения данных, называемый иногда еще просто как интерфейс пользователя (см. рис. 2.1);
- *прикладной компонент,* включающий набор запросов, событий, правил, процедур и других вычислительных функций, реализующий предназначение автоматизированной информационной системы в конкретной предметной области;
- *компонент доступа к данным,* реализующий функции хранения, извлечения, физического обновления и изменения данных (машина данных).

Исходя из особенностей реализации и распределения (расположения) в системе этих трех компонентов различают *четыре* ***модели технологий «Клиент-сервер»:****

- *модель* ***файлового сервера*** (File Server — FS);
- *модель* ***удаленного доступа к данным*** (Remote Data Access—RDA);
- *модель* ***сервера базы данных*** (DataBase Server **— DBS);**
- *модель* ***сервера приложении*** (Application Server — AS).

* ***Ладыженский Г.М.*** Системы управления базами данных — коротко о главном//СУБД.—№№1-4.—1995.

### 5.2.1. Модель файлового сервера

Модель файлового сервера является наиболее простой и характеризует собственно не столько способ образования фактографической информационной системы, сколько общий способ взаимодействия компьютеров в локальной сети. Один из компьютеров сети выделяется и определяется ***файловым сервером,*** т. е. *общим хранилищем любых данных.* Суть FS-модели иллюстрируется схемой, приведенной на рис. 5.2.

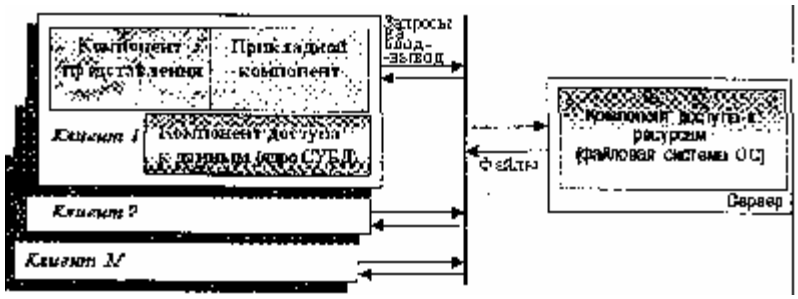



*Рис. 5.2. Модель файлового сервера*

В **FS-модели** все основные компоненты размещаются на клиентской установке. При обращении к данным ядро СУБД, в свою очередь, обращается с запросами на ввод-вывод данных за сервисом к файловой системе. С помощью функций операционной системы в оперативную память клиентской установки полностью или частично на время сеанса работы копируется файл базы данных. Таким образом, сервер в данном случае выполняет чисто пассивную функцию.

*Достоинством* данной модели являются ее простота, отсутствие высоких требований к производительности сервера (главное — требуемый объем дискового пространства). Следует также отметить, что программные компоненты СУБД в данном случае не распределены, т. е. никакая часть СУБД на сервере не инсталлируется и не размещается.

С другой стороны также очевидны и *недостатки* такой модели. Это, прежде всего, высокий сетевой трафик, достигающий пиковых значений особенно в момент массового вхождения в систему пользователей, например в начале рабочего дня. Однако более существенным с точки зрения работы с общей базой данных является отсутствие специальных механизмов безопасности файла (файлов) базы данных со стороны СУБД. Иначе говоря, разделение данных между пользователями (параллельная работа с одним файлом данных) осуществляется только средствами файловой системы ОС для одновременной работы нескольких прикладных программ с одним файлом.

Несмотря на очевидные недостатки, модель файлового сервера является естественным средством расширения возможностей персональных (настольных) СУБД в направлении поддержки многопользовательского режима и, очевидно, в этом плане еще будет сохранять свое значение.

### 5.2.2. Модель удаленного доступа к данным

Модель удаленного доступа к данным основана на учете специфики размещения и физического манипулирования данных во внешней памяти для реляционных СУБД. В ***RDA-модели*** компонент доступа к данным в СУБД полностью отделен от двух других компонентов (компонента представления и прикладного компонента) и размещается на сервере системы. Компонент доступа к данным реализуется в виде самостоятельной программной части СУБД, называемой ***SQL-сервером***, и инсталлируется на вычислительной установке сервера системы. Функции SQL-сервера ограничиваются низкоуровневыми операциями по организации, размещению, хранению и манипулированию данными в дисковой памяти сервера. Иначе говоря, SQL-сервер играет роль *машины данных.* Схема RDA-модели приведена на рис. 5.3.

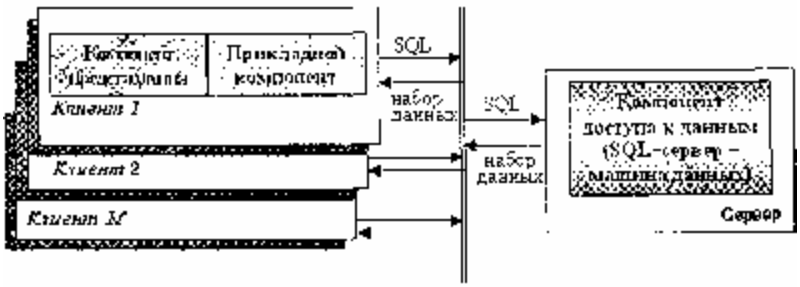



*Рис. 5.3. Модель удаленного доступа к данным (RDA -модель)*

В файле (файлах) базы данных, размещаемом на *сервере* системы, находится также и *системный каталог базы данных,* в который помещаются в том числе и сведения о зарегистрированных клиентах, их полномочиях и т. п.

На *клиентских* установках инсталлируются отделенные программные части СУБД, реализующие *интерфейсные* и *прикладные* функции. Пользователь, входя в клиентскую часть системы, регистрируется через нее на сервере системы и начинает обработку данных. Прикладной компонент системы (библиотеки запросов, процедуры обработки данных) полностью размещается и выполняется на клиентской установке. При реализации своих функций прикладной компонент формирует необходимые SQL-инструкции, направляемые SQL-серверу. *SQL-сервер,* представляющий специальный программный компонент, ориентированный на *интерпретацию SQL-инструкций* и *высокоскоростное выполнение низкоуровневых операций с данными,* принимает и *координирует* SQL-инструкции от различных клиентов, выполняет их, проверяет и обеспечивает выполнение ограничений целостности данных и направляет клиентам результаты обработки SQL-инструкции, представляющие как известно наборы (таблицы) данных.

Таким образом, общение клиента с сервером происходит через SQL-инструкции, а с сервера на клиентские установки передаются только результаты обработки, т. е. наборы данных, которые могут быть существенно меньше по объему всей базы данных. В результате резко *уменьшается загрузка сети,* а сервер приобретает активную центральную функцию. Кроме того, ядро СУБД в виде SQL-сервера обеспечивает также традиционные и важные функции по обеспечению ограничений целостности и безопасности данных при совместной работе нескольких пользователей.

Другим, может быть неявным, *достоинством* RDA-модели является *унификация интерфейса* взаимодействия прикладных компонентов информационных систем с общими данными. Такое взаимодействие стандартизовано в рамках языка SQL уже упоминавшимся специальным протоколом ODBC* (Open Database Connectivity), играющим важную роль в обеспечении *интероперабельности,* т. е. независимости от типа СУБД на клиентских установках в распределенных системах. Иначе говоря, специальный компонент ядра СУБД на сервере (так называемый ***драйвер ODBC)*** способен воспринимать, обрабатывать запросы и направлять результаты их обработки на клиентские установки, функционирующие под управлением реляционных СУБД других, не «родных» типов. Такая возможность существенно повышает гибкость в создании распределенных информационных систем на базе интеграции уже существующих в какой-либо организации локальных баз данных под управлением настольных или другого типа реляционных СУБД. Специальные драйверы ODBC могут

обеспечить возможность использования настольной СУБД в качестве клиента SQL-сервера «тяжелой» многопользовательской клиент-серверной СУБД.

* Стандартный протокол доступа к данным на серверах баз данных SQL.

К *недостаткам* RDA-модели можно отнести высокие требования к клиентским вычислительным установкам, так как прикладные программы обработки данных, определяемые спецификой предметной области АИС, выполняются на них. Другим недостатком является все же существенный трафик сети, обусловленный тем, что с сервера базы данных клиентам направляются наборы (таблицы) данных, которые в определенных случаях могут занимать достаточно существенный объем.

### 5.2.3. Модель сервера базы данных

Развитием RDA-модели стала модель сервера базы данных. Ее *сердцевиной* является рассмотренный ранее *механизм хранимых процедур.* В отличие от RDA-модели, определенные для конкретной предметной области АИС события, правила и процедуры, описанные средствами языка SQL, хранятся вместе с данными на сервере системы и на нем же выполняются. Иначе говоря, прикладной компонент полностью размещается и выполняется на сервере системы. Схематично DBS-модель приведена на рис. 5.4.

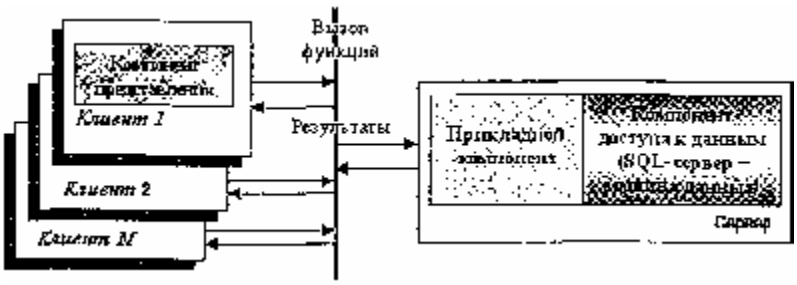



*Рис. 5.4. Модель сервера базы данных (DBS-модель)*

На *клиентских установках* в DBS-модели размещается только *интерфейсный компонент* (компонент представления) АИС, что существенно *снижает требования* к вычислительной установке *клиента.* Пользователь через интерфейс системы на клиентской установке направляет на сервер базы данных только лишь *вызовы необходимых процедур, запросов* и *других функций по обработке данных.* Все затратные операции по доступу и обработке данных выполняются на сервере и клиенту направляются лишь результаты обработки, а не наборы данных, как в RDA-модели. Этим обеспечивается существенное *снижение трафика* сети в DBS-модели по сравнению с RDA-моделью.

Следует, однако, заметить, что на сервере системы выполняются процедуры прикладных задач одновременно всех пользователей системы. В результате резко *возрастают требования к вычислительной установке сервера,* причем как к объему дискового пространства и оперативной памяти, так и к быстродействию. Это основной *недостаток* DBS-модели.

К *достоинствам* же ***DBS-модели,*** помимо разгрузки сети, относится и более активная роль сервера сети, размещение, хранение и выполнение на нем *механизма событий, правил и процедур,* возможность более адекватно и эффективно «настраивать» распределенную АИС на все нюансы предметной области системы. Также более надежно обеспечивается согласованность состояния и изменения данных, и, вследствие этого, повышается надежность хранения и обработки данных, эффективно координируется коллективная работа пользователей с общими данными.

### 5.2.4. Модель сервера приложений

Чтобы разнести требования к вычислительным ресурсам сервера в отношении быстродействия и памяти по разным вычислительным установкам, используется модель сервера приложений. Суть ***AS-модели*** заключается в переносе прикладного компонента АИС на специализированный в отношении повышенных ресурсов по быстродействию дополнительный сервер системы. Схема AS-модели приведена на рис. 5.5.

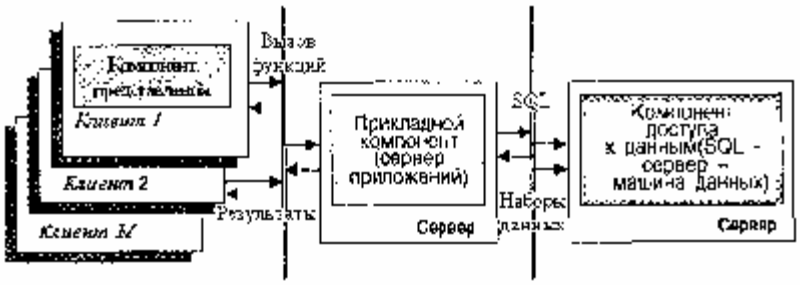



*Рис. 5.5. Модель сервера приложений (AS-модель)*

Как и в DBS-модели, на *клиентских установках* располагается только *интерфейсная* часть системы, т.е. компонент представления. Однако вызовы функций обработки данных направляются на *сервер приложений,* где эти функции совместно выполняются для всех пользователей системы. За выполнением *низкоуровневых операций* по доступу и изменению данных сервер приложений, как в RDA-модели, обращается к *SQL-серверу,* направляя ему вызовы SQL-процедур, и получая, соответственно, от него наборы данных. Как известно, последовательная совокупность операций над данными (SQL-инструкций), имеющая отдельное смысловое значение, называется *транзакцией.* В этом отношении сервер приложений от клиентов системы управляет формированием транзакций, которые выполняет SQL-сервер. Поэтому программный компонент СУБД, инсталлируемый на сервере приложений, еще называют также ***монитором обработки транзакций*** (Transaction Processing Monitors — TRM), или просто монитором транзакций.

AS-модель, сохраняя сильные стороны DBS-модели, позволяет более *оптимально построить вычислительную схему информационной системы,* однако, как и в случае RDA-модели, *повышает трафик сети.*

В еще не устоявшейся до конца терминологии по моделям и технологиям «Клиент-сервер» *RDA-модель* характеризуют еще как модель с так называемыми ***«толстыми»,*** а *DBS-модель* и *AS-модель* как модели, соответственно, с ***«тонкими» клиентами.*** По *критерию звеньев* системы *RDA-модель* и *DBS-модель* называют ***двухзвенными (двухуровневыми) системами,*** а *AS-модель* ***трехзвенной (трехуровневой) системой.***

В практических случаях используются смешанные модели, когда простейшие прикладные функции и обеспечение ограничений целостности данных поддерживаются хранимыми на сервере процедурами (DBS-модель), а более сложные функции предметной области (так называемые «правила бизнеса») реализуются прикладными программами на клиентских установках (RDA-модель) или на сервере приложений (AS-модель).

### 5.2.5. Мониторы транзакций

Сердцевиной и основой *эффективности функционирования многопользовательских систем «Клиент-сервер»* является ***эффективное управление транзакциями.*** Собственно само понятие транзакции возникло именно в процессе исследования принципов построения и функционирования централизованных многопользовательских реляционных СУБД. Транзакции играют важную роль в *механизме обеспечения* СУБД *ограничений целостности* базы данных. Ограничения целостности непосредственно проверяются по завершению очередной транзакции. Если условия ограничений целостности данных не выполняются, то происходит «откат» транзакции (выполняется SQL-инструкция *ROLLBACK),* в противном случае транзакция фиксируется (выполняется SQL-инструкция *COMMIT).*

Помимо обеспечения целостности данных механизм транзакций оказался чрезвычайно полезным для практической реализации одного из основополагающих принципов распределенных многопользовательских систем — ***изолированности пользователей***. Как уже отмечалось, единичные действия пользователей с базой данных ассоциированы с транзакциями. В том случае, когда *от разных пользователей поступают транзакции,* время выполнения которых перекрывается, монитор транзакций обеспечивает специальную *технологию их взаимного выполнения и изоляции* с тем, чтобы избежать нарушений согласованного состояния данных и других *издержек совместной обработки.* К числу подобных издержек относятся:

- потерянные изменения;
- «грязные» данные;
- неповторяющиеся чтения.

***Потерянные изменения*** возникают тогда, когда две транзакции одновременно *изменяют один и тот же объект базы данных.* В том случае, если в силу каких-либо причин, например, из-за нарушений целостности данных, происходит откат, скажем, второй транзакции, то вместе с этим отменяются и все изменения, внесенные в соответствующий период времени первой транзакцией. В результате первая еще не завершившаяся транзакция при повторном чтении объекта не «видит» своих ранее сделанных изменений данных. Очевидным *способом преодоления* подобных ситуаций является запрет изменения данных любой другой транзакцией до момента завершения первой транзакции—так называемая ***блокировка*** объекта.

***«Грязные» данные*** возникают тогда, когда *одна транзакция изменяет* какой-либо объект данных, а *другая* транзакция в этот момент *читает данные из того же объекта.* Так как первая транзакция еще не завершена, и, следовательно, не проверена согласованность данных после проведенных, или вовсе еще только частично проведенных изменений, то вторая транзакция может «видеть» соответственно несогласованные, т. е. «грязные» данные. Опять-таки *способом недопущения* таких ситуаций может быть *запрет чтения объекта* любой другой транзакцией, пока не завершена первая транзакция, его изменяющая.

***Неповторяющиеся чтения*** возникают тогда, когда *одна* транзакция *читает* какой-либо объект базы данных, а *другая* до завершения первой его *изменяет* и успешно *фиксируется.* Если при этом первой, еще не завершенной, транзакции требуется повторно прочитать данный объект, то она «видит» его в другом состоянии, т. е. чтение не повторяется. Способом недопущения подобных ситуаций в противоположность предыдущему случаю является *запрет изменения* объекта любой другой транзакцией, когда первая транзакция на чтение еще не завершена.

Механизм изоляции транзакций и преодоления ситуаций несогласованной обработки данных в общем виде основывается на технике ***сериализации транзакций.*** *План (способ) выполнения совокупности транзакций называется сериальным, если результат совместного выполнения транзакций эквивалентен результату некоторого последовательного их выполнения.* Сериализацией транзакций в этом смысле является организация их выполнения по некоторому сериальному плану. Существуют *два* различных *подхода* сериализации транзакций:

• синхронизационные захваты (блокировки) объектов базы данных;

• временные метки объектов базы данных.

В первом подходе определяется два основных *режима захватов* — ***совместный режим*** (Shared) и ***монопольный режим*** (exclusive). При совместном режиме осуществляется разделяемый захват, требующий только операций чтения. Поэтому такой захват еще называют *захватом по чтению.* При монопольном режиме осуществляется неразделяемый захват, требующий операций обновления данных. Такой захват, соответственно, еще называют *захватом по записи.*

Наиболее распространенным вариантом первого подхода является реализация ***двухфазного протокола синхронизационных захватов (блокировок) объектов*** базы данных — 2PL (Two-Phase Locks). В соответствии с данным протоколом выполнение транзакции происходит *в два этапа.* На *первом* этапе *(первая фаза)* перед выполнением любой операции транзакция запрашивает и накапливает захваты необходимых объектов в соответствующем режиме. После получения и накопления необходимых захватов (блокировок) осуществляется *вторая фаза* — фиксация изменений (или откат по соображениям целостности данных) и освобождение захватов.

При построении сериальных планов *допускается совмещение только захватов по чтению.* В остальных случаях транзакции должны «ждать», когда необходимые объекты разблокируются (освободятся). Более изощрённые стратегии сериализации транзакций основываются на ***«гранулировании»*** объектов захвата (файл базы данных, отдельная таблица, страница файла данных, отдельная запись-кортеж). Соответственно при этом *расширяется номенклатура синхронизационных режимов захватов,* например в совместном режиме (по чтению) может быть захвачен и в целом файл и отдельные его страницы, или в другом случае обеспечивается совместный режим захвата файла с возможностью монопольного захвата отдельных его объектов — таблиц, страниц или записей-кортежей). Грануляция синхронизационных блокировок позволяет строить более эффективные планы сериализации транзакций.

Существенным *недостатком* синхронизационных захватов является возможность возникновения ***тупиковых ситуаций*** (Deadlock). Предположим, одна транзакция на первой фазе установила монопольный захват одного объекта, а другая транзакция монопольный захват второго объекта. Для осуществления полного набора своих операций первой транзакции еще требуется совместный захват

второго объекта, а второй транзакции, соответственно, совместный захват первого объекта. Ни одна из транзакций не может закончить первую фазу, т. е. полностью накопить все необходимые захваты, так как требуемые объекты уже монопольно захвачены, хотя были свободны к моменту начала осуществления транзакций.

Непростой проблемой является *автоматическое обнаружение* (распознавание) таких *тупиковых ситуации* и их разрешение *(разрушение)*. Распознавание тупиков основывается на *построении и анализе графа ожидания транзакций,* состоящего из вершин-транзакций и вершин-объектов захвата. Исходящие из вершин-транзакций дуги к вершинам-объектам соответствуют требуемым захватам, а дуги, исходящие из вершин-объектов к вершинам-транзакциям соответствуют полученным захватам. При тупиковых ситуациях в графе ожидания транзакций наблюдаются петли (циклы), обнаружение которых обеспечивается специальным алгоритмизируемым механизмом редукции графа, подробное изложение которого можно найти в специальной литературе по теории графов. Отметим только, что применение подобного механизма распознавания тупиков требует построения и постоянного поддержания (обновления) графа ожидания транзакций, что *увеличивает накладные расходы* при выполнении транзакции и снижает производительность обработки данных.

Еще одной проблемой является технология, или, лучше сказать, *алгоритм разрушения тупиков.* В общем плане такие алгоритмы основываются на выборе *транзакции-жертвы,* которая временно откатывается для предоставления возможности завершения выполнения операций другим транзакциям, что, конечно же, в определенной степени нарушает принцип изолированности пользователей. В качестве «жертвы» выбирается или самая *«дешевая»* транзакция в смысле затрат на выполнение, или транзакция с наименьшим *приоритетом.*

Более простой *альтернативой* технике синхронизационных захватов является ***техника временных меток***. Суть этого метода заключается в том, что каждой транзакции приписывается *временная метка, соответствующая моменту начала выполнения транзакции.* При выполнении *операции над объектом транзакция «помечает» его своей меткой и типом операции (чтение или изменение)*. Если при этом другой транзакции требуется операция над уже «помеченным» объектом, то выполняются действия по следующему алгоритму:

- проверяется, не закончилась ли транзакция, первой «пометившая» объект;
- если первая транзакция закончилась, то вторая транзакция помечает его своей меткой и выполняет необходимые операции;
- если первая транзакция не закончилась, то проверяется конфликтность операций (напомним, что конфликтно любое сочетание, кроме «чтение-чтение»);
- если операции неконфликтны, то они выполняются для обеих транзакций, а объект до завершения операций помечается меткой более поздней, т. е. более молодой транзакции;
- если операции конфликтны, то далее происходит откат более поздней транзакции и выполняется операция более ранней (старшей) транзакции, а после ее завершения объект помечается меткой более молодой транзакции и цикл действий повторяется.

В результате того что при таком алгоритме конфликтность транзакций определяется более грубо, чем при синхронизационных блокировках, реализация *метода временных меток* вызывает *более частые откаты транзакций.* Несомненным же *достоинством* метода временных меток является ***отсутствие тупиков,*** и, следовательно, отсутствие накладных расходов на их распознавание и разрушение.

СУБД идеологии «Клиент-сервер», называемые иногда в противовес однопользовательским («настольным») «тяжелыми» системами (Oracle, SyBase, Informix, Ingres и др.), составляют одно из наиболее интенсивно развивающихся направлений централизованных многопользовательских систем, охватывая своим управлением сотни тысяч гигабайтов фактографических данных, и в этом отношении еще долгое время будут играть роль фактического стандарта корпоративных информационных систем.

## 5.3. Технологии объектного связывания данных

*Унификация взаимодействия прикладных компонентов с ядром информационных систем в виде SQL-серверов, наработанная для клиент-серверных систем, позволила выработать аналогичные решения и для интеграции разрозненных локальных баз данных под управлением настольных СУБД в сложные децентрализованные гетерогенные распределенные системы. Такой подход получил название* ***объектного связывания данных.***

С узкой точки зрения, технология объектного связывания данных решает *задачу обеспечения доступа из одной локальной базы*, открытой одним пользователем, *к данным в другой локальной базе** (в другом файле), возможно находящейся на другой вычислительной установке, открытой и эксплуатируемой другим пользователем.

* На практике другой локальной базой может быть и база данных сервера централизованной многопользовательской корпоративной системы.

Решение этой задачи основывается на поддержке современными «настольными» СУБД (MS Access, MS FoxPro, dBase и др.) технологии «***объектов доступа к данным***» — DAO. При этом следует отметить, что под объектом понимается интеграция данных и методов их обработки в одно целое (объект), на чем, как известно, основываются объектно-ориентированное программирование и современные объектно-ориентированные операционные среды. Другими словами, СУБД, поддерживающие DAO, получают возможность внедрять и оперировать в локальных базах объектами доступа к данным, физически находящимся в других файлах, возможно на других вычислительных установках и под управлением других СУБД.

Технически технология DAO основана на уже упоминавшемся ***протоколе ODBC,*** который принят за *стандарт доступа* не только *к данным* на SQL-серверах клиент-серверных систем, но и в качестве стандарта доступа к любым данным под управлением реляционных СУБД. Непосредственно для доступа к данным на основе протокола ODBC используются инициализируемые на тех установках, где находятся данные, специальные программные компоненты, называемые ***драйверами ODBC,*** или инициализируемые ядра тех СУБД, под управлением которых были созданы и эксплуатируются внешние базы данных. Схематично принцип и особенности доступа к внешним базам данных на основе объектного связывания иллюстрируются на рис. 5.6.

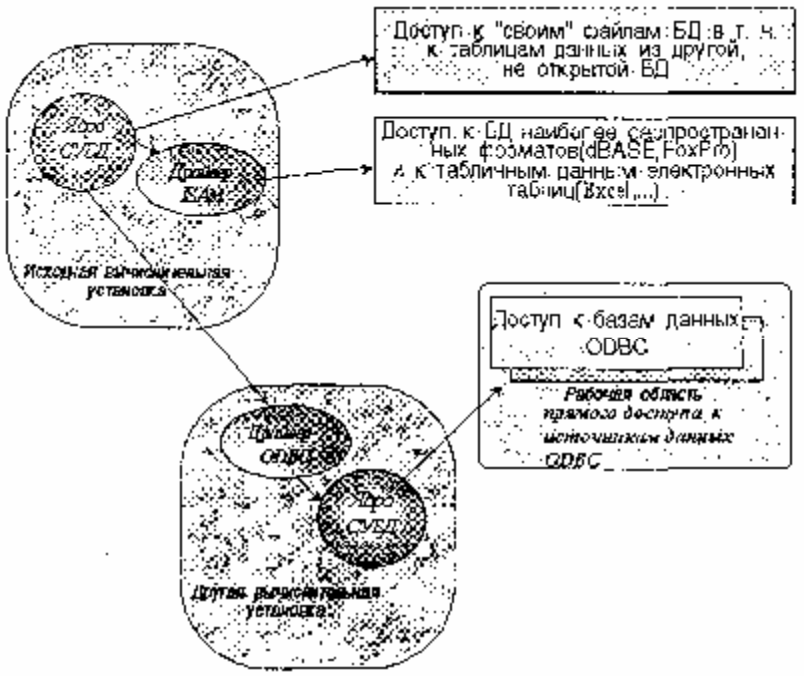



*Рис. 5.6. Принцип доступа к внешним данным на основе протокола ODBC*

Прежде всего, современные настольные СУБД обеспечивают возможность ***прямого доступа к объектам*** (таблицам, запросам, формам) ***внешних баз данных «своих» форматов.*** Иначе говоря, в открытую в текущем сеансе работы базу данных пользователь имеет возможность вставить специальные ссылки-объекты и оперировать с данными из другой (внешней, т. е. не открываемой специально в данном сеансе) базы данных. *Объекты из внешней базы данных, вставленные в текущую базу данных,* называются ***связанными,*** и, как правило, имеют специальные обозначения для отличия от внутренних объектов. При этом следует подчеркнуть, что *сами данные физически* в файл (файлы) текущей базы данных не помещаются, а *остаются в файлах «своих» баз дачных.* В системный каталог текущей базы данных помещаются все необходимые для доступа сведения о связанных объектах — внутреннее имя и внешнее, т. е. истинное имя объекта во внешней базе данных, полный путь к файлу внешней базы и т. п.

*Связанные объекты* для пользователя ничем не отличаются от *внутренних объектов.* Пользователь может также открывать связанные во внешних базах таблицы данных, осуществлять поиск, изменение, удаление и добавление данных, строить запросы по таким таблицам и т. д. Связанные объекты можно интегрировать в схему внутренней базы данных, т. е. устанавливать *связи между внутренними и связанными таблицами.*

Технически оперирование связанными объектами из внешних баз данных «своего» формата мало отличается от оперирования сданными из текущей базы данных. *Ядро СУБД* при обращении к данным связанного объекта по системному каталогу текущей базы данных находит сведения о месте нахождения и других параметрах соответствующего файла (файлов) внешней базы данных и прозрачно, т. е. невидимо для *пользователя открывает* этот *файл* (файлы), а далее обычным порядком организует в оперативной памяти *буферизацию страниц внешнего файла данных для непосредственно доступа и манипулирования данными.* Следует также заметить, что на основе возможностей многопользовательского режима работы с файлами данных современных операционных систем, с файлом внешней базы данных, если он находится на другой вычислительной установке, может *в тот же момент времени работать и другой пользователь,* что и обеспечивает коллективную обработку общих распределенных данных.

Для иллюстрации на рис. 5.7 приведен пример схемы БД, организованной на основе техники объектного связывания данных распределенной системы коллективной обработки данных трех подразделений некоторой организации — службы ДОУ, отдела кадров и бухгалтерии.* На вычислительных установках перечисленных подразделений, объединенных в локальную сеть, создаются и эксплуатируются локальные базы данных, структурные схемы которых отражают задачи и особенности предметных областей сведений, необходимых для информационного обеспечения деятельности соответствующих подразделений. При этом часть данных, распределенных по таким таблицам, как «Сотрудники», «Подразделения», «Штатные категории» и др., являются общепотребными для всех или для группы подразделений. Логично исключить дублирование ввода, редактирования, корректировки и хранения таких общих данных, возложив эти функции на пользователей тех локальных баз данных, где это наиболее естественно и обоснованно сточки зрения функциональных особенностей соответствующих подразделений и сложившихся информационных потоков, а пользователям локальных баз данных других подразделений предоставить доступ к ним. Так, к примеру, таблицы данных по сотрудникам и подразделениям наиболее естественно заполнять данными и хранить в локальной базе кадрового подразделения, обеспечивая доступ к ним пользователей локальных баз службы ДОУ и бухгалтерии. Структурную схему локальных баз этих подразделений в этом случае целесообразно построить на основе объектного связывания данных. На рис. 5.7 связанные, т. е. физически находящиеся в других базах данных, таблицы выделены специальной раскраской и обрамлением. Стрелками на рисунке показаны связи типа «Один-ко-многим» (острие стрелки соответствует стороне «многие»).

---

* Данный пример является чисто иллюстративным и никаким образом не может претендовать на отражение каких-либо реальных баз данных.

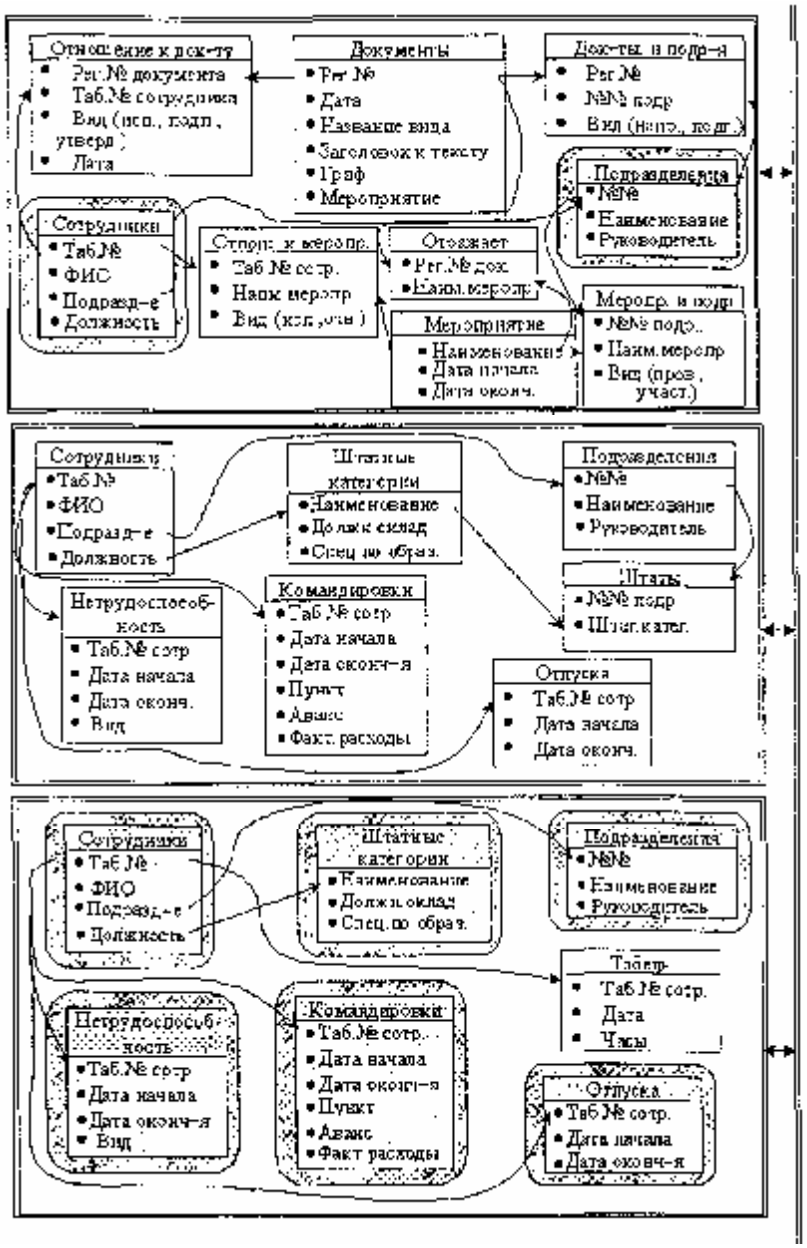



*Рис. 5.7. Пример схем локальных баз данных со связанными объектами*

На рис. 5.8 приведен еще один пример схемы локальных баз данных, использующих совместные данные по линии информационного обеспечения производства и сбыта продукции. Предметные области сведений по этим трем локальным базам данных очень близки и переплетены, однако, как и в предыдущем примере, с учетом специфики подразделений, ведение и размещение таких общепотребных таблиц как «Продукция», «Типы, номенклатура» целесообразно осуществлять в локальной базе *Планово-производственного отдела,* таблиц «Комплектующие», «Сырье», «Поставщики» в локальной базе *Отдела снабжения,* а таблиц «Заказы» и «Клиенты» в локальной базе *Отдела сбыта.* Опять-таки данные по таблице «Сотрудники» могут быть получены на основе объектного связывания из локальной базы *Отдела кадров.*

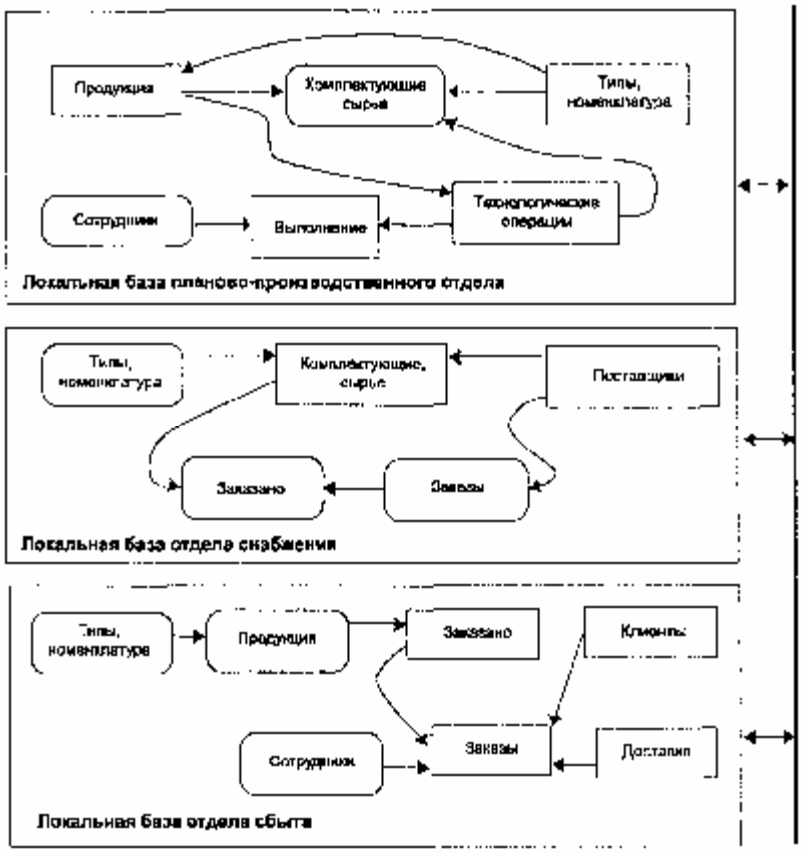



*Рис. 5.8. Пример схем локальных баз данных со связанными объектами*

Нетрудно заключить, что подобный принцип построения распределенных систем при больших объемах данных в связанных таблицах приведет к ***существенному увеличению трафика сети,*** так как по сети постоянно передаются, даже не наборы данных, а страницы файлов баз данных, что может приводить к пиковым перегрузкам сети. Поэтому представленные схемы локальных баз данных со взаимными связанными объектами нуждаются в дальнейшей тщательной проработке с точки зрения интенсивности, направленности потоков данных в сети между локальными базами исходя из информационных технологий, обусловленных производственно-технологическими и организационными процессами.

Не менее существенной проблемой является ***отсутствие надежных механизмов безопасности данных*** и обеспечения ***ограничений целостности.*** Так же как и в модели файлового сервера, совместная работа нескольких пользователей с одними и теми же данными обеспечивается только функциями операционной системы по одновременному доступу к файлу нескольких приложений.

Аналогичным образом обеспечивается доступ к данным, находящимся в *базах дачных наиболее распространенных форматов* других СУБД, таких, например, как базы данных СУБД FoxPro, dBASE, а так же к табличным данным электрон

При этом доступ может обеспечиваться как *непосредственно ядром СУБД,* так и специальными дополнительными драйверами ISAM (Indexed Sequential Access Method), входящими, как правило, в состав комплекта СУБД. Такой подход реализует ***интероперабельность*** построенных подобным образом распределенных гетерогенных систем, т.е. «разномастность» типов СУБД, поддерживающих локальные базы данных. При этом, однако, объектное связывание ограничивается только непосредственно таблицами данных, исключая другие объекты базы данных (запросы, формы, отчеты), реализация и поддержка которых зависят от специфики конкретной СУБД.

Доступ к базам данных других СУБД (см. рис. 5.6) реализуется через технику *драйверов ODBC,* которые *инсталлируются и выполняются на тех вычислительных установках, где находятся удаленные данные.* «Идеология» в данном случае такова. В составе настольной СУБД, поддерживающей локальную базу данных, можно инсталлировать дополнительный программный компонент, называемый драйвером ODBC.* Инсталлируемый драйвер **ODBC** «регистрируется» в специальном подкаталоге системного каталога операционной системы.** Так образуется ***рабочая область прямого доступа к источникам данных ODBC.***

* В операционной системе Windows драйверы ODBC реализуются посредством файлов DDL (библиотеки динамической компоновки).

** В операционной системе Windows данный подкаталог так и называется — ODBC.

Для непосредственного доступа к источникам данных ODBC ядро СУБД по системному каталогу внутренней локальной базы данных определяет *местонахождение источника,* по *протоколу взаимодействия приложений* (API) осуществляет *вызов (запуск)* на вычислительной установке удаленных данных драйвера ODBC и направляет ему по протоколу ODBC *SQL-ичструкщт* на доступ и обработку данных. При этом *режим* такого доступа регулируется рядом *параметров* (интервал вызова процедур, максимальное время обработки запроса, количество однократно пересылаемых по сети записей из набора данных, формируемых по запросам, время блокировок записей и т. д.). Данные параметры записываются в специальный реестр операционной системы при инсталляции и регистрации соответствующего драйвера ODBC.

При таком подходе каждая локальная СУБД на своей вычислительной установке выполняет роль SQL-сервера, т. е. машины данных, в случае обращения на доступ извне (из других вычислительных установок) к данным из «ее» файлов данных. Так как непосредственную обработку данных в данном случае выполняет «родная» СУБД, знающая все особенности логической и физической структуры «своих» файлов данных, то обеспечивается, как правило, более эффективная обработка, а самое главное, проверяются и выполняются ограничения целостности данных по логике предметной области источников данных.

Определенной проблемой технологий объектного связывания является появление «брешей» в системах защиты данных и разграничения доступа. Вызовы драйверов ODBC для осуществления процедур доступа к данным помимо пути, имени файлов и требуемых объектов (таблиц), если соответствующие базы защищены, содержат в открытом виде пароли доступа, в результате чего может быть проанализирована и раскрыта система разграничения доступа и защиты данных.

## 5.4. Технологии реплицирования данных

Во многих случаях *узким местом распределенных систем,* построенных на основе технологий «Клиент-сервер» или объектного связывания данных, является *недостаточно высокая производительность* из-за необходимости передачи по сети большого количества данных. Определенную альтернативу построения быстродействующих распределенных систем предоставляют технологии реплицирования данных.

***Репликой*** называют *особую копию базы данных для размещения на другом компьютере сети с целью автономной работы пользователей с одинаковыми (согласованными) данными общего пользования.*

Основная идея реплицирования заключается в том, что пользователи работают автономно с одинаковыми (общими) данными, растиражированными по локальным базам данных, обеспечивая с учетом отсутствия необходимости передачи и обмена данными по сети ***максимальную для своих вычислительных установок производительность.*** Программное обеспечение СУБД для реализации такого подхода соответственно дополняется *функциями тиражирования **(реплицирования) баз данных,*** включая тиражирование как самих данных и их структуры, так и системного каталога с информацией о размещении реплик, иначе говоря, с информацией о конфигурировании построенной таким образом распределенной системы.

При этом, однако, возникают *две проблемы* обеспечения одного из основополагающих принципов построения и функционирования распределенных систем, а именно —*непрерывности согласованного состояния данных*:

• обеспечение согласованного состояния во всех репликах количества и значений общих данных;

• обеспечение согласованного состояния во всех репликах структуры данных.

***Обеспечение согласованного состояния общих данных,*** в свою очередь, основывается на реализации одного из *двух принципов*:

• *принципа непрерывного размножения обновлений* (любое обновление данных в любой реплике должно быть немедленно размножено);

• *принципа отложенных обновлений* (обновления реплик могут быть отложены до специальной команды или ситуации).

***Принцип непрерывного размножения обновлений*** является основополагающим при построении так называемых «*систем реального времени»,* таких, например, как системы управления воздушным

движением, системы бронирования билетов пассажирского транспорта и т. п., где требуется непрерывное и точное соответствие реплик или других растиражированных данных во всех узлах и компонентах подобных распределенных систем.

Реализация принципа *непрерывного размножения обновлений* заключается в том, что *любая транзакция считается успешно завершенной, если она успешно завершена на всех репликах системы.* На практике реализация этого принципа встречает существенные затруднения, связанные с ***тупиками,*** как и в технологиях синхронизационных захватов систем «Клиент-сервер». Предположим, что на одной вычислительной установке пользователь обновляет данные в своей реплике. На время осуществления транзакции (транзакций) соответствующие записи в базе данных этой реплики ядром локальной СУБД заблокированы от изменения другими пользователями. Вместе с тем транзакция может быть зафиксирована и, следовательно, разблокированы соответствующие данные только тогда, когда данная транзакция послана и также завершена на других репликах системы. Предположим также, что в другой реплике системы, находящейся на другом компьютере сети, в это же время другой пользователь проводит свои обновления (транзакции) с теми же записями, которые, естественно, в этот момент также заблокированы от изменений для других пользователей. Так образуется тупик. Одна транзакция не может быть зафиксирована в своей реплике, потому что заблокированы соответствующие записи в другой реплике. А разблокировка этих записей в другой реплике также невозможна до тех пор, пока не разблокируются соответствующие записи в первой реплике, т. е. когда завершится транзакция в первой реплике. Создается тупиковая ситуация.

Для *обнаружения* (распознавания) *тупиков* в реплицированных системах применяются *такие же алгоритмы,* которые были разработаны в *мониторах транзакций* централизованных систем «Клиент-сервер», — строится и поддерживается аналогичный граф ожидания транзакций и применяются аналогичные алгоритмы распознавания и разрушения тупиков, основанные в основном на технике приоритетов.

В целом ряде предметных областей распределенных информационных систем режим реального времени с точки зрения непрерывности согласования данных не требуется. Такие системы автоматизируют те организационно-технологические структуры, в которых информационные процессы не столь динамичны. Если взять, к примеру, автоматизированную информационную систему документооборота, то традиционная «скорость» перемещения и движения служебных документов соответствует рабочему дню или в лучшем случае рабочим часам. В этом случае обновление реплик распределенной информационной системы, если она будет построена на технологии реплицирования, требуется, скажем, только лишь один раз за каждый рабочий час, или за каждый рабочий день.

Такого рода информационные системы можно строить на основе ***принципа отложенных обновлений.*** Накопленные в какой-либо реплике изменения данных *специальной командой* пользователя направляются для обновления всех остальных реплик систем. Такая *операция* называется ***синхронизацией реплик.*** Возможность *конфликтов и тупиков* в этом случае при синхронизации реплик *существенно снижается,* а немногочисленные подобные конфликтные ситуации легко разрешить *организационными мерами.*

Решение *второй проблемы согласованности данных,* а именно **— *согласованности структуры данных,*** осуществляется через *частичное отступление,* как и в системах «Клиент-сервер», *от принципа отсутствия центральной установки* и основывается на ***технике «главной» реплики.***

Суть этой техники заключается в том, что *одна из реплик* базы данных системы объявляется *главной.* При этом *изменять структуру базы данных можно только в главной реплике.* Эти изменения структуры данных *тиражируются на основе принципа отложенных обновлений,* т. е. через специальную *синхронизацию реплик.* Частичность отступления от принципа отсутствия центральной установки заключается в том, что в отличие от чисто централизованных систем, *выход из строя главной реплики не влечет сразу гибель всей распределенной системы,* так как остальные реплики продолжают функционировать автономно. Более того, на практике СУБД, поддерживающие технологию реплицирования, позволяют пользователю с определенными полномочиями (администратору системы) преобразовать любую реплику в главную и тем самым полностью восстановить работоспособность всей системы.

Процесс синхронизации реплик в современных СУБД включает обмен только теми данными, которые были изменены или добавлены в разных репликах. С этой целью в системном каталоге базы данных создаются специальные таблицы текущих изменений и организуется система *глобальной иденти-*

*фикации* (именования) *всех объектов распределенной системы,* включая раздельное поименование одинаковых объектов (вплоть до записей таблиц) в разных репликах.* Такой подход несколько увеличивает объем базы данных, но позволяет существенно ограничить *транспортные расходы* на синхронизацию реплик.

* Так называемая техника глобальных уникальных идентификаторов (GUID).

Важным, с точки зрения гибкости и эффективности функционирования распределенных информационных систем, построенных на технологиях реплицирования, является возможность создания так называемых ***частичных реплик*** и включения в реплики как ***реплицнруемых,*** так и ***нереплицируемых объектов.*** Частичной репликой называется *база данных, содержащая ограниченное подмножество записей полной реплики.* Распространенным способом создания частичных реплик является использование фильтров, устанавливаемых для конкретных таблиц полной (главной) реплики. Частичные реплики позволяют решить некоторые проблемы, связанные с разграничением доступа к данным и повышают производительность обработки данных. Так, к примеру, в реплику базы данных для определенного подразделения целесообразно реплицировать только те записи таблицы «Сотрудники», которые относятся к данному подразделению, исключив тем самым доступ к другим записям. Техника частичных реплик также снижает затраты на синхронизацию реплик, так как ограничивает количество передаваемых по сети изменений данных.

Возможность включения в реплики объектов базы данных, которые не подлежат репликации, позволяет *более гибко и адекватно настроить схему* и прочие *объекты* БД (запросы, формы и отчеты) *на специфику предметной области, особенности ввода данных и решаемые информационные задачи по конкретному элементу распределенной системы.*

На рис. 5.9 иллюстрируется подход к организации общей схемы распределенной информационной системы по делопроизводству некоторой организационной структуры на основе технологий репликации данных.

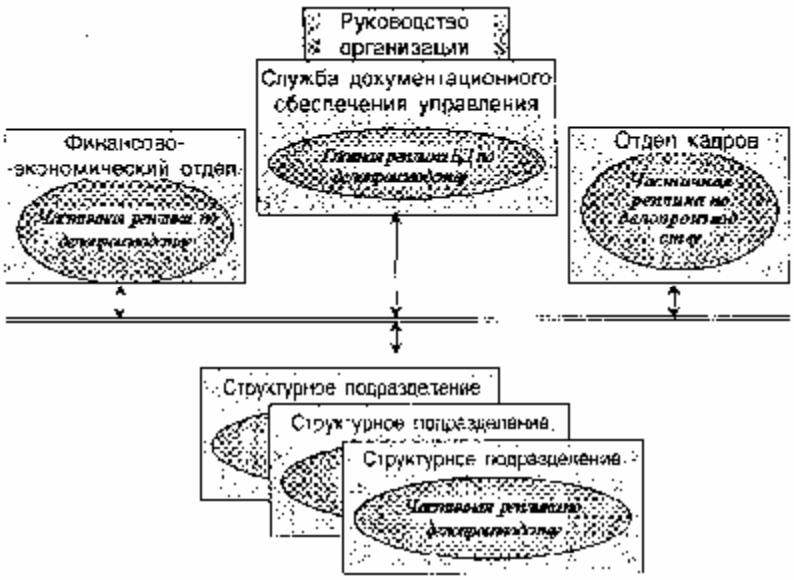



*Рис. 5.9. Пример подхода к организации схемы распределенной информационной системы по делопроизводству на основе техники реплицирования*

*Технологии репликации данных* в тех случаях, когда не требуется обеспечивать большие потоки и интенсивность обновляемых в информационной сети данных, являются *экономичным решением проблемы создания распределенных информационных систем* с элементами централизации по сравнению с использованием дорогостоящих* «тяжелых» клиент-серверных систем.

* Хотя, конечно же, более надежных и функциональных.

На практике для совместной коллективной обработки данных применяются *смешанные технологии, включающие элементы объектного связывания данных, репликации и клиент-серверных решений.* При этом дополнительно к проблеме логического проектирования, т. е. проектирования логической схемы организации данных (таблицы, поля, ключи, связи, ограничения целостности), добавляется не менее сложная проблема ***транспортно-технологического проектирования информационных потоков, разграничения доступа*** и т.д. К сожалению, пока не проработаны теоретико-

методологические и инструментальные подходы для автоматизации проектирования распределенных информационных систем с учетом факторов как логики, так и информационно-технологической инфраструктуры предметной области.

Тем не менее развитие и все более широкое распространение распределенных информационных систем, определяемое самой распределенной природой информационных потоков и технологий, является основной перспективой развития автоматизированных информационных систем.

### Вопросы и упражнения

1. Что «распределено» в распределенных информационных системах и каковы основные принципы создания и функционирования распределенных информационных систем?
2. Поясните суть техники «представлений» в базах данных и задачи, которые решаются на основе техники «представлений».
3. Какой из основных принципов распределенных информационных систем принесен в «жертву» в системах «Клиент-сервер»? Поясните преимущества и недостатки такого подхода.
4. На какие компоненты подразделяется программное обеспечение систем «Клиент-сервер»? Какие функции выполняются каждым компонентом?
5. Поясните принципы и схемы RDA, DBS и AS-моделей систем «Клиент-сервер» и дайте их сравнительную характеристику. Какие системы называются системами с «тонкими» («толстыми») клиентами, с 2- или 3-уровневой (2- или 3-звенной) архитектурой?
6. Охарактеризуйте роль и место монитора транзакций в СУБД систем «Клиент-сервер».
7. Какие издержки совместной обработки общих данных предотвращает монитор транзакций в системах «Клиент-сервер»?
8. Дайте определение сериальному плану выполнения транзакций и охарактеризуйте основные подходы к механизмам построения таких планов.
9. Поясните суть двухфазного протокола синхронизационных захватов объектов и механизм возможного образования тупиковых ситуаций.
10. Как распознаются и разрушаются тупиковые ситуации в технике двухфазного протокола синхронизационных захватов?
11. В чем заключается гранулирование объектов захватов и почему такой подход обеспечивает более эффективные стратегии построения сериальных планов выполнения транзакций?
12. Каковы достоинства и недостатки техники временных меток при синхронизационных захватах объектов?
13. В чем заключается суть и механизм объектного связывания данных при построении распределенных информационных систем из разрозненных локальных баз данных?
14. Существует ли системный каталог распределенной базы данных, построенной на основе технологии объектного связывания, и каким образом выражается логическая схема такой базы данных? В чем достоинства и недостатки такого подхода?
15. Какие (функции, кроме традиционных, выполняют локальные («настольные») СУБД при использовании технологий объектного связывания?
16. Охарактеризуйте главную идею и основные подходы к построению распределенных информационных систем в технологиях реплицирования данных.
17. Поясните принцип отложенных обновлений и процессов синхронизации реплик.
18. Чем главная реплика отличается от остальных? Как наличие главной реплики соотносится с принципами распределенных систем?
19. Что обеспечивается возможностью создания частичных реплик?

## 6. ДОКУМЕНТАЛЬНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

В развитии программного обеспечения СУБД в 70-е—80-е годы превалировало направление, связанное с фактографическими информационными системами, т. е. с системами, ориентированными на работу со структурированными данными. Были разработаны основы и модели организации фактографических данных, отработаны программно-технические решения по накоплению и физическому хранению таких данных, реализованы специальные языки запросов к базам данных и решен целый ряд других задач по эффективному управлению большими объемами структурированной информации. В результате основу информационного обеспечения деятельности

предприятий и организаций к началу 90-х годов составили фактографические информационные системы, вобравшие в себя в совокупности колоссальный объем структурированных данных.*

* В этом смысле очень характерным является рекламный девиз корпорации Oracle: «*Мы храним триллионы бийт».*

Вместе с тем создание и эксплуатация фактографических информационных систем требует либо изначально структурированных данных, таких, например, как отчеты датчиков в АСУ ТП, финансовые массивы бухгалтерских АИС и т. д., либо предварительной структуризации данных, как, например, в информационной системе кадрового подразделения, где все данные по сотрудникам структуризируются по ряду формализованных позиций. При этом зачастую структуризация данных требует больших накладных, в том числе и организационных расходов, что, в конечном счете, приводит к материальным издержкам информатизации.

Кроме того, входные информационные потоки в целом ряде организационно-технологических и управленческих сфер представлены неструктурированными данными в виде служебных документов и иных текстовых источников. Извлечение из текстов данных по формализованным позициям для ввода в фактографические системы может приводить к ошибкам и потере части информации, которая в исходных источниках имеется, но в силу отсутствия в схеме базы данных адекватных элементов не может быть отражена в банке данных фактографических АИС.

В результате, несмотря на интенсивное развитие и распространение фактографических информационных систем, огромная часть неструктурированных данных, необходимых для информационного обеспечения деятельности различных предприятий и организаций, остается в неавтоматизированном или слабо автоматизированном* виде. К таким данным относятся огромные массивы различной периодики, нормативно-правовая база, массивы служебных документов делопроизводства и документооборота.

* Представлена в электронном виде в текстовых файлах, но без средств систематизации, обработки, анализа и эффективного поиска 6.1. Общая характеристика и виды документальных информационных систем.

Потребности в системах, ориентированных на накопление и эффективную обработку неструктурированной или слабоструктурированной информации привели к возникновению еще в 70-х годах отдельной ветви программного обеспечения систем управления базами данных, на основе которых создаются документальные информационные системы.

Однако теоретические исследования вопросов автоматизированного информационного поиска документов, начавшись еще в 50-х—60-х годах, к сожалению, не получили такой строгой, полной и в то же время технически реализуемой модели представления и обработки данных, как реляционная модель в фактографических системах. Не получили также стандартизации (как язык SQL) и многочисленные попытки создания универсальных так называемых информационно-поисковых языков, предназначенных для формализованного описания смыслового содержания документов и запросов по ним. В итоге, несмотря на то, что первые системы автоматизированного информационного поиска документов появились еще в 60-х годах, развитые коммерческие информационно-поисковые системы, ориентированные на накопление и обработку текстовых документов, получили распространение лишь в конце 80-х — начале 90-х годов.

## 6.1. Общая характеристика и виды документальных информационных систем

Напомним, что в фактографических информационных системах единичным элементом данных, имеющим отдельное смысловое значение, является запись, образуемая конечной совокупностью полей-атрибутов. Иначе говоря, информация о предметной области представлена набором одного или нескольких типов структурированных на отдельные поля записей.

В отличие от фактографических информационных систем, ***единичным элементом данных*** *в документальных информационных системах* является *неструктурированный на более мелкие элементы* ***документ.*** **В** качестве неструктурированных документов в подавляющем большинстве случаев выступают, прежде всего, ***текстовые документы,*** представленные в виде текстовых файлов, хотя к классу неструктурированных документированных данных могут также относиться звуковые и графические файлы.

Основной *задачей* документальных информационных систем является *накопление и предоставление пользователю документов, содержание, тематика, реквизиты и т. п. которых адекватны его информационным потребностям.* Поэтому можно дать следующее определение *документальной*

*информационной системы* — ***единое хранилище документов с инструментарием поиска и отбора необходимых документов.*** *Поисковый характер* документальных информационных систем исторически определил еще одно их название — *информационно-поисковые системы* (ИПС), хотя этот термин не совсем полно отражает специфику документальных ИС.* *Соответствие найденных документов информационным потребностям пользователя* называется ***пертинентностью.*** В силу теоретических и практических сложностей с формализацией смыслового содержания документов пертинентность относится скорее к качественным понятиям, хотя, как будет рассмотрено ниже, может выражаться определенными количественными показателями.

* Поиск информации (данных) осуществляется и в фактографических ИС. Таким образом термин ИПС определяет функциональное назначение ИС, но не отражает специфики представления и обработки данных. Специфика документальных ИПС заключается в том, что они удовлетворяют информационные потребности пользователя, предоставляя ему документы, в которых содержится интересующая пользователя информация.

В зависимости от особенностей реализации хранилища документов и механизмов поиска документальные ИПС можно разделить на *две группы:*

- системы на основе индексирования;
- семантически-навигационные системы.

В ***семантически-навигационных*** системах документы, помещаемые в хранилище (в базу) документов, оснащаются специальными *навигационными конструкциями,* соответствующими *смысловым связям* (отсылкам) между различными документами или отдельными фрагментами одного документа. Такие конструкции реализуют некоторую *семантическую** (смысловую) *сеть* в базе документов. *Способ и механизм выражения информационных потребностей* в подобных системах заключаются в *явной навигации пользователя по смысловым отсылкам между документами.* В настоящее время такой подход реализуется в ***гипертекстовых ИПС.***

* Семантика (от греч. «semantikos»—обозначающий)—смысловая сторона языка, отдельных слов и частей слова, а также—раздел языкознания, изучающий значения слов.

В системах ***на основе индексирования*** исходные документы помещаются в базу без какого-либо дополнительного преобразования,* но при этом смысловое содержание каждого документа отображается в некоторое ***поисковое пространство.*** Процесс отображения документа в поисковое пространство называется ***индексированием*** и заключается в присвоении каждому документу некоторого индекса-координаты в поисковом пространстве. Формализованное представление (описание) индекса документа называется ***поисковым образом документа*** (ПОД). Пользователь выражает свои информационные потребности средствами и *языком поискового пространства,* формируя ***поисковый образ запроса*** **(ПОЗ)** к базе документов. Система на основе определенных критериев и способов ищет документы, поисковые образы которых соответствуют или близки поисковым образам запроса пользователя, и выдает соответствующие документы. Соответствие найденных документов запросу пользователя *называется релевантностью.*** Схематично общий принцип устройства и функционирования документальных ИПС на основе индексирования иллюстрируется на рис. 6.1.

* За исключением возможного сжатия (архивирования).

** На практике термин *релевантность* часто отождествляют с термином *пертинентность,* хотя в строгом отношении они различны.

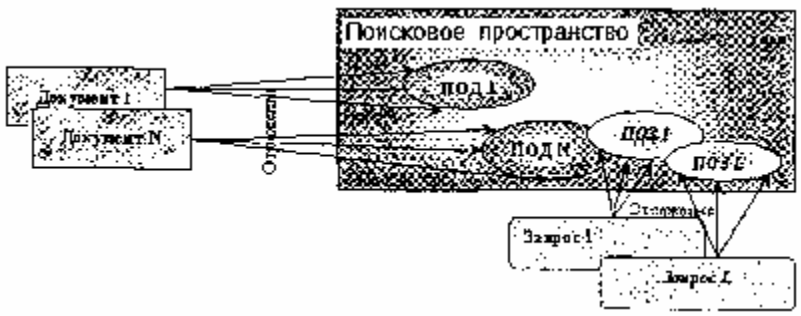



*Рис. 6.1. Общий принцип устройства и функционирования документальных ИПС на основе индексирования*

Особенностью документальных ИПС является также то, что в их функции, как правило, включаются и ***задачи информационного оповещения*** пользователей по всем новым поступающим в систему

документам, соответствующим заранее определенным информационным потребностям пользователя.* *Принцип решения* задач информационного оповещения в документальных ИПС на основе индексирования аналогичен принципу решения задач поиска документов по запросам и основан на *отображении в поисковое пространство информационных потребностей пользователя в виде* так называемых ***поисковых профилей пользователей*** (ППП). Информационно-поисковая система по мере поступления и индексирования новых документов сравнивает их образы с поисковыми профилями пользователей и принимает решение о соответствующем оповещении. Принцип решения задач информационного оповещения схематично иллюстрируется на рис. 6.2.

* Задачи информационного оповещения основаны на идеологии т.н. **избирательного распространения информации *(ИРИ),*** наработанной в библиотечном деле.

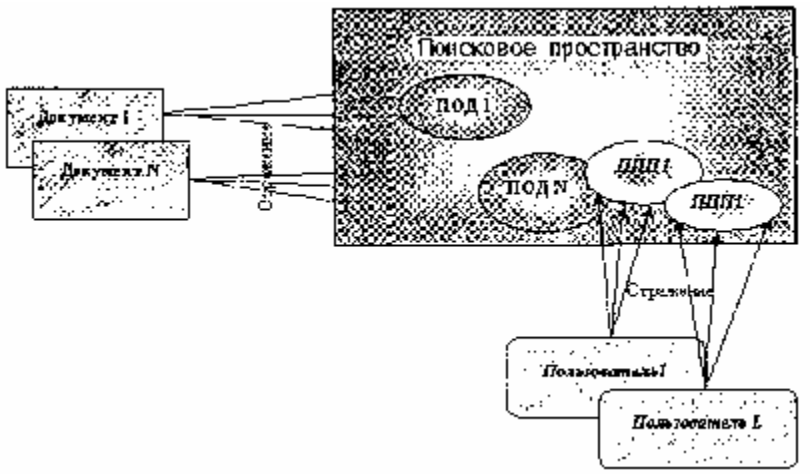



*Рис. 6.2. Принцип решения задач информационного оповещения в документальных ИПС на основе индексирования*

*Поисковое пространство,* отображающее поисковые образы документов и реализующее механизмы информационного поиска документов так же, как и в СУБД фактографических систем, строится на основе *языков документальных баз данных,* называемых информационно-поисковыми языками (ИПЯ). ***Информационно-поисковый язык*** представляет собой некоторую *формализованную семантическую систему, предназначенную для выражения содержания документа и запросов по поиску необходимых документов.* По аналогии с языками баз данных фактографических систем ИПЯ можно разделить на структурную и манипуляционную составляющие.

*Структурная составляющая* ИПЯ (поискового пространства) документальных ИПС на основе индексирования реализуется ***индексными указателями*** в форме *информационно-поисковых каталогов, тезаурусов* и *генеральных указателей.*

***Информационно-поисковые каталоги*** являются традиционными технологиями организации информационного поиска в документальных фондах библиотек, архивов и представляют собой *классификационную систему знаний по определенной предметной области. Смысловое содержание документа* в информационно-поисковых каталогах *отображается* тем или иным *классом каталога,* а *индексирование* документов заключается в *присвоении* каждому документу специального *кода (индекса)* соответствующего по содержанию класса (классов) каталога и создания на этой основе специального индексного указателя.

***Тезаурус*** представляет собой специальным образом *организованную совокупность основных лексических единиц (понятий) предметной области (словарь терминов) и описание парадигматических отношений между ними.* Парадигматические отношения выражаются *семантическими отношениями между элементами словаря, не зависящими от любого контекста.* Независимость от контекста означает обобщенность (абстрагированность) смысловых отношений, например отношения «род-вид», «предмет-целое», «субъект-объект-средство-место-время действия». Так же, как и в информационно-поисковых каталогах, в системах на основе тезаурусов в информационно-поисковое пространство отображается не весь текст документа, а только лишь выраженное средствами тезауруса смысловое содержание документа.

***Генеральный указатель**** (глобальный словарь-индекс) в общем виде представляет собой *перечисление всех слов (словоформ), имеющихся в документах хранилища, с указанием (отсылками) координатного местонахождения каждого слова* (№ документа — № абзаца — № предложения — № слова). *Индексирование* нового документа в таких системах производится через дополнение

*координатных отсылок тех словоформ генерального указателя, которые присутствуют в новом документе.* Так как поисковое пространство в таких системах *отражает полностью весь текст документа* (все слова документа), а не только его смысловое содержание, то такие системы получили название ***полнотекстовых ИПС.*****

* Исторически в специальной литературе употреблялся также термин «конкорданс».

** В специальной литературе такие системы иногда называют системами без лексического контроля, т. е. без учета возможной синонимичности отдельных групп словоформ, объединения отдельных групп словоформ в общие смысловые группы, семантических отношении между словоформами.

Структурная составляющая ИПЯ *семантически-навигационных систем* реализуется в виде техники смысловых отсылок в текстах документов и специальном навигационном интерфейсе по ним и в настоящее время представлена ***гипертекстовыми технологиями.***

*Поисковая (манипуляционная) составляющая* ИПЯ реализуется дескрипторными и семантическими языками запросов.

В ***дескрипторных языках*** документы и запросы представляются *наборами некоторых лексических единиц* (слов, словосочетаний, терминов) — *дескрипторов, не имеющих между собой связей,* или, как еще говорят, *не имеющих грамматики.* Таким образом, каждый *документ* или запрос ассоциируется или, лучше сказать, *представлен* некоторым *набором дескрипторов. Поиск* осуществляется через поиск документов с *подходящим набором дескрипторов.* В качестве элементов-дескрипторов выступают либо элементы *словаря ключевых терминов,* либо элементы *генерального указателя* (глобального словаря всех словоформ). В силу отсутствия связей между дескрипторами, набор которых для конкретного документа и конкретного запроса выражает, соответственно, поисковый образ документа — ПОД или поисковый образ запроса ПОЗ, такие языки применяются, прежде всего, в полнотекстовых системах.

***Семантические языки*** содержат грамматические и семантические конструкции для выражения (описания) смыслового содержания документов и запросов. Все многообразие семантических языков подразделяется на две большие группы:

• предикатные языки;

• реляционные языки.

В ***предикатных языках*** в качестве элементарной осмысленной конструкции высказывания выступает *предикат,* который представляет собой *многоместное отношение некоторой*

*совокупности грамматических элементов.* Многоместность отношения означает, что каждый элемент предиката играет определенную роль для группы лексических элементов в целом, но не имеет конкретных отношений с каждым элементом этой группы в отдельности. Аналогом предикатного высказывания в естественном языке выступает *предложение,* констатирующее определенный факт или описывающее определенное событие.

В ***реляционных языках*** лексические единицы высказываний могут вступать только в *бинарные* (друг с другом), но не в совместные, т. е. не многоместные отношения.

В качестве *лексических единиц* семантических языков выступают *функциональные классы естественного языка,* важнейшими из которых являются:

• *понятия-классы* (общее определение совокупности однородных элементов реального мира, обладающих некоторым характерным набором свойств, позволяющих одни понятия-классы отделять от других);

• *понятия-действия* (лексический элемент, выражающий динамику реального мира, содержит универсальный набор признаков, включающий субъект действия, объект действия, время действия, место действия, инструмент действия, цель и т. д.);

• *понятия-состояния* (лексические элементы, фиксирующие состояния объектов);

• *имена* (лексические элементы, идентифицирующие понятия-классы);

• *отношения* (лексические элементы, служащие для установления связей на множестве понятий и имен);

• *квантификаторы* (всеобщности, существования и т. д.).

Семантические языки составляют языково-манипуляционную основу информационно-поисковых каталогов, тезаурусов и семантически-навигационных (гипертекстовых) ИПС, описывая своими средствами собственно сами каталоги, тезаурусы, семантические сети и выражая смысловое содержание документов и запросов.

В заключение общей характеристики документальных ИПС приведем основные ***показатели эффективности*** их функционирования. Такими показателями являются полнота и точность информационного поиска.

***Полнота информационного поиска*** *R* определяется отношением числа найденных пертинентных документов *A* к общему числу пертинентных документов *C*, имеющихся в системе или в исследуемой совокупности документов:

$$R=A/C.$$

***Точность информационного поиска*** *P* определяется отношением числа найденных пертинентных документов *A* к общему числу документов *L,* выданных на запрос пользователя:

$$P=A/L$$

Наличие среди отобранных на запрос пользователя нерелевантных документов называется ***информационным шумом*** системы. Коэффициент информационного шума *к*, соответственно, определяется отношением числа нерелевантных документов *(L–A),* выданных в ответе пользователю к общему числу документов *L,* выданных на запрос пользователя:

$$\kappa = \frac{L - A}{L}$$

В идеале полнота информационного поиска и точность информационного поиска должны приближаться к единице, хотя на практике их значения колеблются в пределах от 60 до 90%.

## 6.2. Информационно-поисковые каталоги и тезаурусы

Как уже отмечалось, информационно-поисковые каталоги основаны на классификации сведений по определенной предметной области и исторически были первыми системами информационного поиска документов в библиотечном и архивном деле, возникнув еще в средние века по сложившейся тогда схеме разделения наук и искусств.

Современные библиотечные классификации основываются на системах десятичной классификации Дьюи (1876 г.) и правил построения алфавитно-предметных рубрик Ч. А. Каттера (1876 г.). Впоследствии на развитие информационно-поисковых каталогов огромное влияние оказали работы С. Р. Ранганатана (система аналитико-синтетической классификации двоеточием — Colon Classification, 30-е гг.), У. Е. Баттена (карты Баттена на основе оптического совпадения, 30-е—40-е годы), К. Муерса (дескрипторная система «Зато-кодирования», 1947-1948 гг.) и М. Тауба (система унитермов Тауба, 1951 г.). В России первые отечественные системы библиотечно-библиографической классификации были разработаны в XIX веке ученым-натуралистом П. К. Демидовым и академиком К. Э. Бэром.

### 6.2.1. Классификационные системы поиска документов

Основные направления развития систем классификационного индексирования документов можно проиллюстрировать схемой, приведенной на рис. 6.3.

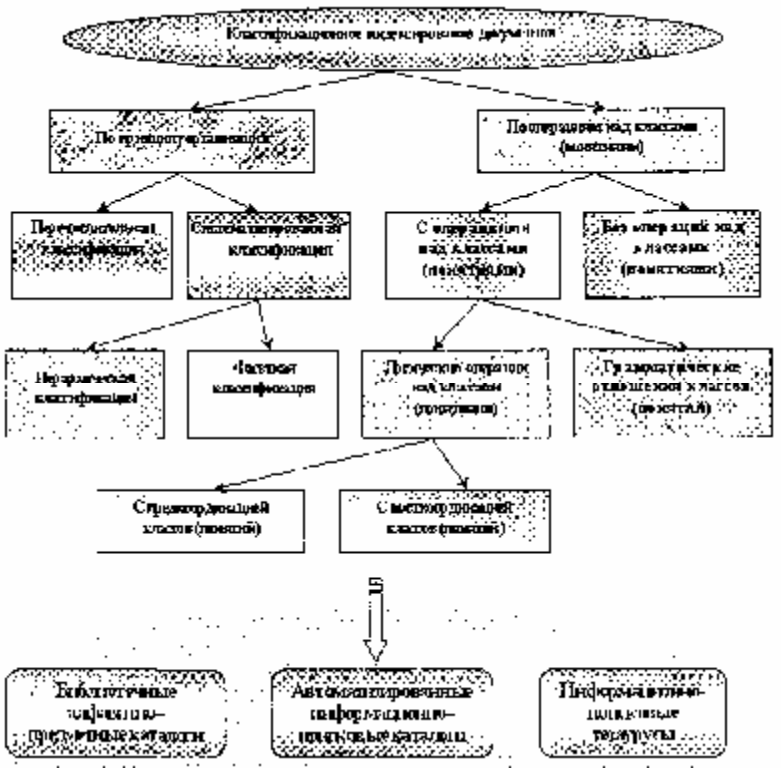


*Рис. 6.3. Системы классификационного индексирования документов*

Первоначальные подходы к классификации тематики (предмета) документов основывались на формировании *списка предметных заголовков,* располагаемых в *алфавитном порядке.* Каждая предметная рубрика получала определенный *цифровой* или *буквенно-цифровой код.* Содержание (предметы) документа индексировалось перечислением кодов тех рубрик, которые отражали предметы документа. Такие подходы получили название ***перечислительной классификации.***

Особенностью систем перечислительной классификации является возможность индексирования документов любым количеством предметов (рубрик), отражающих содержание документа. Для осуществления поиска необходимых документов по классификатору (каталогу) определяются коды интересующих абонента предметов (рубрик) и далее отбираются из хранилища те документы, которые проиндексированы соответствующими кодами. Для удобства поиска и отбора по каждому документу формируется специальная карточка, на которую наносится информация о кодах предметных рубрик документа, а также, как правило, об авторе, названии и др. библиографических данных документа, его физическом местонахождении, и реферат, который уже на естественном языке в сжатом виде отражает содержание документа. Поиск и отбор документов непосредственно осуществляется по отбору карточек с необходимыми индексными кодами для последующего извлечения из хранилища собственно самих документов.

Перечислительная классификация иллюстрируется на рис. 6.4.

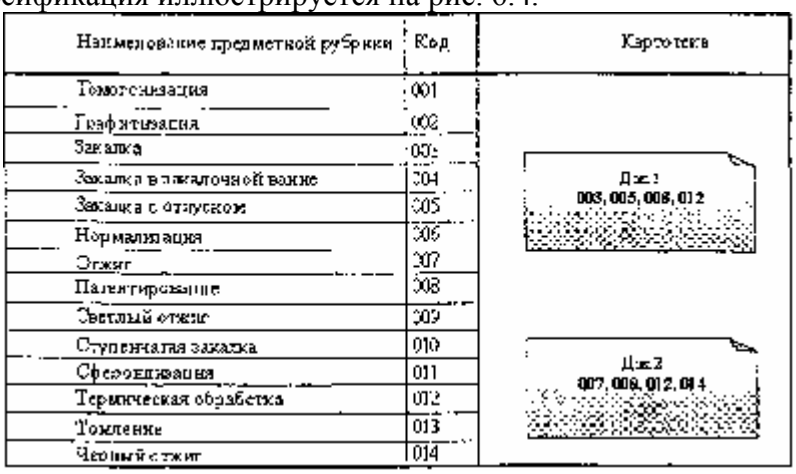



*Рис. 6.4. Индексирование документов на основе перечислительной классификации*

В приведенном на рис. 6.4 примере документ № 1, в котором речь идет об описании патента по технологии закалки с отпуском, проиндексирован кодами 003 (Закалка), 005 (Закалка с отпуском), 008 (Патентирование) и 012 (Термическая обработка). Документ № 2 с описанием патента по

черному отжигу проиндексирован кодами 007 (Отжиг), 008 (Патентирование), 012 (Термическая обработка) и 014 (Черный отжиг).

*Отсутствие систематизированных связей и отношений между предметными рубриками* является основным *недостатком перечислительной классификации.* Так, в приведенном примере рубрика «Закалка отпуском» является под рубрикой рубрики «Закалки» и интуитивно ясно, что если документ получил код «Закалки отпуском», то тем самым он автоматически относится и к более широкой рубрике «Закалка».

Приемом, способствующим в определенной степени преодолению данного недостатка, является использование в списке рубрик специальных *перекрестных ссылок* через конструкцию «см. также». В этом случае в классификаторе вместе с рубрикой «Закалка» помещается следующая конструкция:

*«см. также* Закалка в закалочной ванне

Закалка с отпуском»

Перекрестные ссылки ориентируют пользователя на смысловую связь некоторых рубрик, позволяя более адекватно строить выражение своих информационных потребностей.

При ***систематизированной классификации*** список предметных рубрик строится, как иерархическая структура, в виде перевернутого дерева. Вся *предметная область* ИПС разбивается на ряд *взаимоисключающих (непересекающихся) рубрик.* Каждая рубрика, в свою очередь, может включать несколько *подрубрик* по принципу «Род-Вид». Таким образом, при систематизированной классификации используются уже некоторые *семантические основы предметной области,* выражаемые в *родо-видовых отношениях* основных категорий, понятий и классов. Представление иерархической классификации производится либо в виде древовидного графа рис. 6.5 *а),* либо в табличном виде рис. 6.5 *б)*.

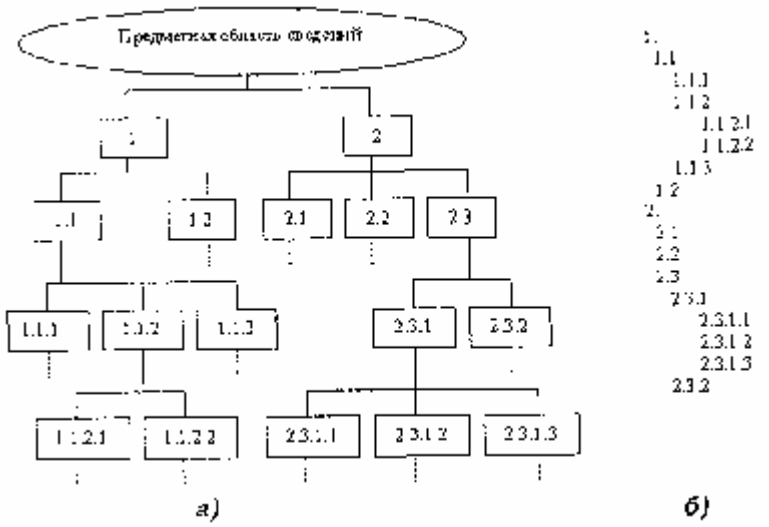



*Рис. 6.5. Древовидная а) и табличная формы б) представления иерархической классификации*

Так же, как и при перечислительной классификации, содержание документа индексируется кодами соответствующих рубрик, однако при этом *отпадает необходимость в явном указании более общих рубрик,* к которым относятся отмеченные подрубрики. В результате индексирование и поиск документов на основе иерархической классификации позволяют более адекватно отражать содержание документов и обеспечивают большую точность поиска. Так, документ из предыдущего примера с описанием патента по технологии закалки с отпуском на основе иерархической классификации может быть проиндексирован только рубрикой «Закалка с отпуском», обозначение которой включает указание на автоматическое отнесение содержания документа и к более широкой рубрике «Закалка» и к еще более широкой рубрике «Термическая обработка».

Перечислительный и иерархический подходы к классификации впоследствии воплотились в широко используемых в *библиотечной практике* ***алфавитно-предметных каталогах,*** наиболее распространенным из которых в настоящее время является *универсальная десятичная классификация (УДК).* В основе УДК лежит классификационная схема Дьюи, дополненная правилами образования сложных рубрик, а также специальными определителями, служащими для более детального описания документов (определители формы и характера документа, определители времени и т. д.). При этом систематизированная классификация позволяет строить сам каталог (картотеку документов) в структурно-иерархическом виде,* что существенно упрощает выражение

пользователем своих информационных потребностей, и, тем самым, ускоряет и повышает точность поиска.

* Каждый класс каталога—ящик с набором карточек документов с соответствующим содержанием.

*Недостатком* как перечислительной, так и иерархической классификации является принципиальная невозможность *заранее перечислить все темы,* по которым существуют или могут существовать документы. Выход из таких ситуаций путем добавления к классификатору новых рубрик (классов, предметов) не может эффективно решить проблему, так как требует в таких случаях переиндексирования всего ранее накопленного документального фонда, что чаще всего нереально по техническим и технологическим аспектам.

Принцип организации классификационного индексирования документов, преодолевающего в определенной степени такие ограничения перечислительной и иерархической классификации, был предложен в 30-х годах выдающимся индийским библиотековедом и математиком Ш.Р. Ранганатаном, развит впоследствии в работах английской группы по исследованию классификаций (Classification Research Group) и получил название ***аналитико-синтетической*** или иначе ***фасетной классификации.*** Идея фасетной классификации состоит в том, что вся *предметная область сведений разбивается на ряд исходных групп рубрик* (фасет) ***по организационно-технологическому*** *или* ***семантическому принципу,*** *отражающему специфику предметной области.*

Фасеты выступают в роли «кирпичиков», из которых можно сложить (сконструировать) любую, даже самую сложную и узкую предметную рубрику. Внутри фасет предметные рубрики строятся и упорядочиваются по алфавитно-иерархическому принципу. Так, к примеру, предметная область документов по производству технологического оборудования разбивается на четыре фасета* — «Конфигурации», «Материалы», «Типы разрушений» и «Напряжения и нагрузки». На основе соединения подрубрик фасет «конструируются» любая конкретная и узкая тематика — см. рис. 6.6.

* Примеры на рис. 6.4 и 6.6 заимствованы из источника — **Ланкастер Ф.У.** Информационно-поисковые системы.—М.: Мир, 1972.

В фасетной классификации, фрагмент которой приведен на рис. 6.6, документ, где речь идет о нагрузках на сжатие трубчатых конструкций из никелевых сплавов, получит индекс Ac Bgt Lg, который будет отражать достаточно узкую тематику, исключая, как и в иерархической классификации, отбор документов с более широкими рубриками.

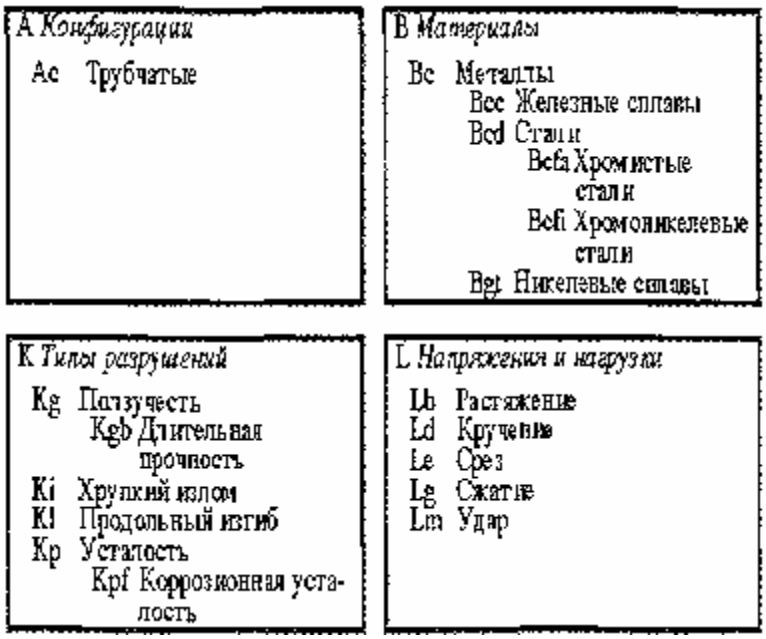



*Рис. 6.6. Пример фрагмента фасетиой классификации*

Основное *достоинство* фасетной классификации заключается в возможности ограниченным небольшим перечнем фасетных рубрик отразить (сконструировать) огромное количество узких специализированных рубрик и, тем самым, наиболее точно и полно проиндексировать содержание документов.

Специфической *проблемой* фасетной классификации является влияние на эффективность поиска документов *порядка следования обозначений рубрик фасет.* Психологические особенности поиска

таковы, что пользователь в первую очередь сосредоточивает внимание на обозначениях тех подрубрик, которые стоят первыми в цепном списке сконструированной формулы, и если интересующие его в первую очередь сведения отражаются рубрикой, стоящей не на первом месте, то он может «с ходу» отвергнуть всю формулу. Для преодоления этого недостатка используется так называемая *пермутация,** при которой для документа приводится список всех возможных вариантов написания сконструированной фасетной формулы на основе циклической перестановки, например:

Ac Bgt Lg
LgAcBgt
Bgt Lg Ac и т. д.

* Пермутация —от слова «перестановка».

Однако такой подход не всегда полностью решает проблему, так как комбинаций по перестановкам может быть очень много, что, в свою очередь, *утяжеляет и усложняет поиск.* Другим подходом является, напротив, *жесткая регламентация порядка изложения фасет,* что в определенной степени ориентирует первоначальное внимание пользователя на тех фасетах, информация по которым интересует его в большей степени.

*Сильной* стороной фасетной классификации является более глубокое, чем при иерархической классификации, использование *семантики.* Фасеты, как уже отмечалось, отражают определенные семантические основы предметной области ИПС, содержащие помимо родо-видовых и некоторые прочие семантические, в частности ролевые, отношения.* Рядом исследователей предлагались универсальные или специализированные фасетные классификации («Индивидуальность», «Материя», «Энергия», «Пространство» и «Время» — Ранганатан; «Предмет в целом», «Вид», «Часть», «Материал», «Свойство», «Процессы», «Операции», «Факторы» — Миллз). Поэтому, в отличие от перечислительной и иерархической классификации, для разработки фасетной классификации предметной области сведений конкретной ИПС используются те же методологические подходы, что и при разработке информационно-логических схем предметных областей фактографических систем (выделение основных фрагментов-сущностей, анализ отношений между ними и т.д.).

* Специальные указатели типа «Объект (субъект) действия», «Инструмент (средство) действия».

### 6.2.2. Координация понятий в классификационных системах

Еще одним аспектом развития систем классификации и поиска документов является ***координация понятий*** (классов, рубрик), выражающаяся в использовании *различных операций над совокупностью понятий при индексировании документов или при поиске документов* (см. рис. 6.3). При этом выделяют *два* направления — *использование только логических операций** (объединение, пересечение, дополнение, включение) и *использование определенной грамматики понятий,*** классов, рубрик в рамках определенного семантического языка.

* Точнее операций из теории множеств.
** Так называемые синтагматические отношения.

Рассмотрим содержание простейших логических операций в отношении классификационных понятий. Под *классом* (понятием) будем понимать совокупность (множество) документов, проиндексированных кодом соответствующего класса. *Объединением* классов *X* и *Y* называется множество документов *X И Y,* которые проиндексированы кодом класса *X* или кодом класса *Y* или одновременно кодами обоих классов. *Пересечением классов X* и *Y* называется множество документов $X \cap Y$, одновременно проиндексированных классом *X* и классом *Y*. *Дополнением* класса *X* классом называется множество документов X' = ¬ X, не проиндексированных кодом класса X.* В формальной логике операция объединения может выражаться терминами «логическая сумма», дизъюнкция или «операция ИЛИ», операция пересечения терминами «логическое произведение», конъюнкция, или «операция И», операция дополнения терминами «логическое отрицание» или операция «НЕ».

* В свою очередь обратно X является дополнением X',

Еще одной важной операцией является операция включения. Класс *X* является *включением* класса *Y→X,* когда любой документ, проиндексированный классом *X,* является одновременно документом, проиндексированным классом *Y.*

Рассмотренные операции в терминах теории множеств иллюстрируются на рис. 6.7.

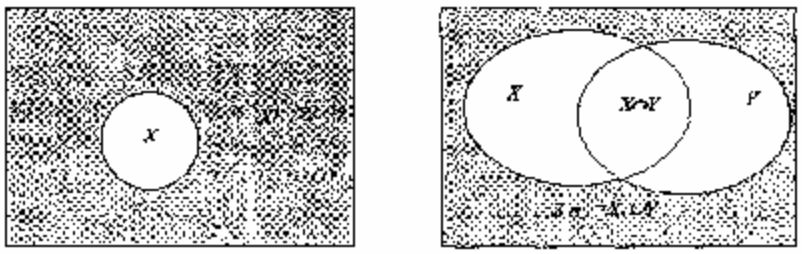


*Рис. 6.7. Иллюстрация операции над классами*

*Логические операции* над понятиями предоставляют возможности отображения при индексировании документов и формировании запросов *более сложных и многоаспектных понятий.* Так, к примеру, рубрика Ac Bgt Lg из примера на рис. 6.6 по фасетной классификации в терминах логических операций представляет собой пересечение трех классов — Ac ∩ Bgt ∩ Lg, соответственно. При этом использование дополнительных операций объединения (ИЛИ) и отрицания (НЕ) обеспечивает построение более сложных, чем при чисто фасетной классификации, комбинаций рубрик, классов и понятий. Кроме того, операция включения дает возможность так называемого цепного представления и описания иерархических структур каталогов, формализуя систематизированный аспект классификаторов.

Идеи координации понятий, т. е. использования операций над классами, активно развивались в 40-с—50-с гг. в первых механизированных системах организации поиска документов (уже упоминавшиеся карты У. Баттена на основе оптического совпадения, система «Зато-кодирования» К. Муэрса и система унитермов М. Тауба). При этом определилось два направления координации понятий — *предкоординация* и *посткоординация* (см. рис. 6.3).

***Предкоординация*** понятий предусматривает использование операций над классами *при индексировании* документов. Иначе говоря, индекс документа представляет собой конструкцию из исходных понятии (классов) классификатора, построенную на основе логических операций. В системах на основе ***посткоординации*** понятий логические операции над классами осуществляются *при поиске* документов, т. е. в процессе формирования поискового образа запроса. Технология и механизм поиска при этом включают предварительный отбор всех документов с индексами классов (рубрик), входящих в логическую конструкцию запроса, с последующим осуществлением собственно логических операций над отобранными совокупностями (множествами) документов.

### 6.2.3. Информационно-поисковые тезаурусы

Особую роль в развитии информационно-поисковых систем сыграли работы Мортимера Тауба, разработавшего в 1951 году *систему унитермов.* В системе Тауба содержание документа индексируется совокупностью *терминов* в виде *однословных обозначений* — ***унитермов.*** Например, документ по теории информационного поиска может быть проиндексирован двумя унитермами — «Информационный», «Поиск». В качестве унитермов чаще всего выступают элементы *словаря ключевых терминов* по определенной предметной области.

В системе Тауба первоначально не предполагалось какой-либо связи или отношений между унитермами и, следовательно, ее можно отнести к чисто дескрипторным системам. Вместе с тем сразу же проявились и такие специфические проблемы дескрипторных систем, как *ложная координация понятий.* Явление ложной координации заключается в такой координации понятий (классов, терминов), которые хотя по отдельности и присутствуют в содержании документа, но комбинируются по смыслу с другими понятиями (терминами, классами). Так, например, в содержании документа, в котором речь идет об информационном обеспечении поисковых бригад при ликвидации чрезвычайных происшествий и последствий стихийных бедствий, также присутствуют в числе прочих унитермы — «Информационный» и «Поиск», и, следовательно, он совершенно неправильно может быть выдан на запрос по теории информационного поиска.

Другой *проблемой* в системах на основе унитермов являются *синонимичность и омонимичность** некоторых *терминов,* что приводит к неоднозначности индексирования документов. Для

преодоления ложной координации и других проблем стали вводить *составные термины, указатели связи* и *ролей терминов* («род— вид», «средство действия» и т. п.), заново открывая в некотором смысле предметную иерархическую рубрикацию со связями, и внося тем самым в чисто дескрипторную систему элементы семантики. Так появилось отдельное направление информационно-поисковых систем, получившее название тезаурусов.

* Синонимы — одинаковые или близкие по смыслу слова, омонимы — слова, одинаковые в написании и звучании, но имеющие разный смысл — ключ (в замке), ключ (источник воды).

*Тезаурус (с греч.* «хранилище», «запас», «сокровищница») в узком смысле представляет собой специальный словарь-справочник, в котором перечислены ключевые слова-дескрипторы определенной предметной области, указаны синонимичные им ключевые слова, установлены способы устранения синонимии, омонимии, полисемии, определены родо-видовые и ассоциативные связи дескрипторов.*

* Строгое определение информационно-поискового тезауруса *(нормативный словарь дескрипторного ИПЯ с зафиксированными в нем парадигматическими отношениями лексических единиц)* приведено в ГОСТ 7.74-96 СИБИД. Информационно-поисковые языки.—М.: Изд-во стандартов, 1997.

В более общем плане в тезаурусе выделяют ***классификационную схему* и *алфавитный перечень дескрипторов-ключевых слов.*** Классификационная схема определяет систематизацию дескрипторов по уровням иерархии исходя из «родо-видовых» или ролевых отношений. Алфавитный перечень содержит словарный фонд дескрипторов для индексирования документов.

Внешним *отличием* информационно-поисковых тезаурусов от информационно-поисковых каталогов на основе предметной иерархической рубрикации со связями и ролевыми отношениями является то, что в тезаурусах помимо классификационной схемы присутствуют сами ключевые слова и дескрипторы, объединяемые под названием классов, рубрик и т. д. В каталогах же присутствуют только лишь обозначения (названия) классов, понятий и т. д., но не определены и нет самих ключевых терминов, им соответствующих.

Главная *идея* информационно-поисковых тезаурусов заключается в *повышении эффективности и автоматизации индексирования документов в рамках дескрипторного подхода.* Иначе говоря, в системах на основе информационно-поисковых тезаурусов ПОД представлен набором дескрипторов (ключевых терминов). Однако в процессе индексирования документов учитываются семантические (родо-видовые, ролевые, синонимичные, омонимичные, полисемичные и ассоциативные) отношения между дескрипторами, что, в конечном счете, обеспечивает более адекватный содержанию ПОД и повышает эффективность поиска документов (по точности, полноте и шуму).

Разработка тезаурусов и их внедрение в информационно-поисковые системы интенсивно осуществлялись в 60-е и 70-е годы. При этом в соответствии с тематическим профилем выделились *многоотраслевые, отраслевые* и *узкотематические тезаурусы.* Первым многоотраслевым тезаурусом за рубежом явился «Тезаурус технических и научных терминов», вышедший в декабре 1967 г. в США. В 1972 г. под редакцией Ю. И. Шемакина был разработан первый отечественный многоотраслевой «Тезаурус научно-технических терминов». В семидесятые годы тезаурусы были разработаны практически для всех отраслей деятельности, а также создано большое количество узкотематических специализированных тезаурусов.

На основе практики разработки и использования информационно-поисковых тезаурусов были также разработаны специальные представления тезаурусов, закрепленные в нашей стране в соответствующих ГОСТах.* Согласно ГОСТ 18383-73 форма представления тезауруса включает алфавитное перечисление *статей по каждому дескриптору* (термину) в следующем виде:**

...

РЕФЕРАТ

| | |
|---|---|
| с | резюме |
| в | СВЕРТЫВАНИЕИНФОРМАЦИИ |
| н | РЕФЕРАТАВТОРСКИЙ |
| | РЕФЕРАТГРАФИЧЕСКИЙ |
| | РЕФЕРАТИНФОРМАТИВНЬШ |
| | РЕФЕРАТ«ТЕЛЕГРАФНОГОСТИЛЯ» |
| | РЕФЕРАТУКАЗАТЕЛЬНЫЙ |
| | РЕФЕРИРОВАНИЕ |

а АННОТАЦИЯ

...

где в качестве буквенных обозначений выступают следующие:

с — термины-синонимы;

в — термины, подчиняющие заглавный термин, т.е. выше по иерархии;

н — термины, подчиненные заглавному, т. е. ниже по иерархии;

а — термины, ассоциированные с заглавным термином.

---

* ГОСТ 18383-73. Тезаурус информационно-поисковый. Общие положения. Форма представления.

** Пример позаимствован из работы: **Соколов А.В.** Информационно-поисковые системы: Учеб. пособие для вузов/Под ред. А. Б. Рябова.—М.: Радио и связь, 1981.

Еще одной *особенностью тезаурусов* является применяемая на практике *возможность расширения словарной базы новыми ключевыми терминами,* появляющимися при накоплении документов в ходе эксплуатации системы. В этом плане различают ***базовые*** и ***рабочие тезаурусы***. Базовые тезаурусы выступают в качестве нормативных пособий по лексике в той или иной отрасли знаний или предметной области. Рабочие тезаурусы в стартовом виде строятся на основе базовых тезаурусов и дополняются в процессе индексирования и анализа появления в документах новых или специфичных терминов (так называемые профессионализмы, иногда жаргонные термины и т. д.). В результате возникает еще один специфический компонент эксплуатации соответствующих ИПС, называемый *ведением тезауруса.*

### 6.2.4. Автоматизация индексирования документов

Важным в практическом плане аспектом информационно-поисковых систем являются технологии, принципы и механизмы индексирования документов применительно к той или иной классификационной схеме.

Развитие теории информационного поиска документов, создание первых механизированных информационно-поисковых систем поначалу не предполагали какой-либо автоматизации (механизации) индексирования документов. ***Индексирование*** *осуществлялось специально подготовленными специалистами-экспертами в предметной области ИПС,* которые могли осуществлять многоаспектный и глубокий анализ смыслового содержания документа и относить его (индексировать) к тем или иным классам, рубрикам, ключевым терминам. Такой подход обусловливал высокие накладные расходы на создание и ведение документальных информационно-поисковых систем, так как требовал наличия в организационном штате высококвалифицированных специалистов-индексаторов. Кроме того, в процесс индексирования при этом вносился *человеческий фактор* (субъективность поисковых образов одного документа, проиндексированного разными специалистами и т. п.).

Поэтому в теории информационного поиска в 50-х-60-х годах выделилось отдельное направление исследований, связанное с вопросами автоматизации индексирования документов. Идеи и начало этих исследований были инициированы появлением уже упоминавшейся системы унитермов Тауба. Индексирование документов набором однословных дескрипторов-терминов (унитермов), имеющихся в тексте документа, позволило *снизить профессиональные требования к индексаторам* и, фигурально выражаясь, *механистицировать* процесс индексирования.*

---

* Иначе в некотором смысле приблизить к чисто механической работе по выявлению в тексте унитермов.

С применением и все более широким использованием вычислительной техники в информационно-поисковых документальных системах эти подходы трансформировались в задачи и технологии ***автоматического,*** т.е. ***без участия специалистов, индексирования документов.***

Огромную роль в исследовании и последующем развитии теории информационного поиска документов сыграли результаты *Кренфилдского* (I и II) *проекта,* проводившегося в конце 50-х — начале 60-х годов Английской ассоциацией специальных библиотек и информационных бюро. В ходе экспериментальных исследований эффективности нескольких различных по типу информационно-поисковых систем (система на основе УДК, фасетная система, система унитермов и некоторые их разновидности), проведенных в ходе реализации Кренфилдского проекта, выявились факторы противоречивого влияния некоторых семантических показателей классификационных ИПС (глубина уровней классов при индексировании, объем словарной базы и др.) на полноту и точность

информационного поиска. Выявилась общая принципиальная закономерность—при повышении полноты поиска на основе использования тех или иных семантических методов при индексировании происходит снижение точности поиска и наоборот. Еще одним «неожиданным» результатом явилось небольшое отличие в показателях эффективности поиска документов в системах с развитой семантикой индексирования и в системах на основе неконтролируемой лексики.

Последний результат активизировал в дальнейшем внимание к более простым и менее дорогим дескрипторным системам с неконтролируемой или слабоконтролируемой лексикой (унитермы, полнотекстовые системы), в которых на основе посткоординации при обработке запросов удается достичь вполне приемлемых показателей полноты и точности поиска. Этими же обстоятельствами был обусловлен импульс исследованиям технологий автоматического индексирования и уже на новом уровне возродилась идея полной механизации (точнее, уже автоматизации) индексирования документов.

Сформировалось два, хотя и близких, но различных по содержанию подхода автоматическому индексированию. Первый подход основан на ***использовании словаря ключевых слов*** (терминов) и применяется в системах на основе *информационно-поисковых тезаурусов.* Индексирование в таких системах осуществляется путем последовательного автоматического поиска в тексте документа каждого ключевого термина. На этой основе строится и поддерживается ***индекс системы,*** собственно и реализующий поисковое пространство документов.

Применяется два типа образования индекса — прямой и инвертированный (см. рис. 6.8).

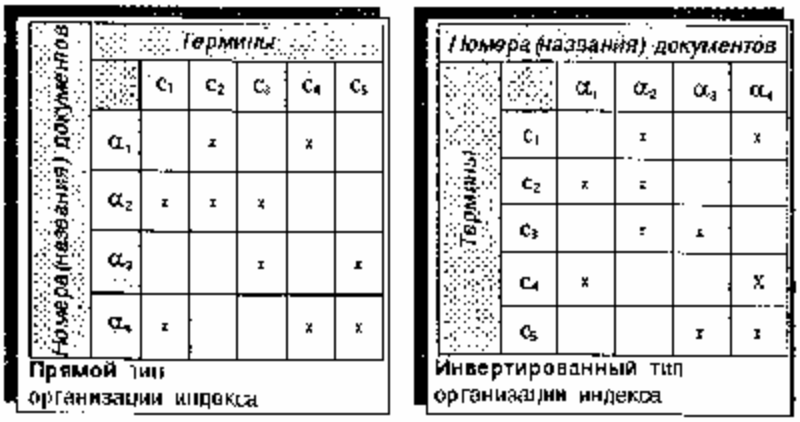



*Рис. 6.8. Прямой и инвертированный типы организации индекса*

***Прямой тип индекса*** строится по схеме «Документ-термины». Поисковое пространство в этом случае представлено в виде матрицы размерностью NxM (N — количество документов, М —количество ключевых терминов). Строки этой матрицы представляют поисковые образы документов.

***Инвертированный тип индекса*** строится по обратной схеме— «Термин —документы». Поисковое пространство соответственно представлено аналогичной матрицей только в транспонированной форме. Поисковыми образами документов в этом случае являются столбцы матрицы.

На основе автоматического индексирования документов по ключевым терминам могут решаться также и задачи автоматической классификации документов, т. е. автоматического отнесения документов к тем или иным классификационным рубрикам. Такие задачи особенно актуализировались в связи с интенсивным развитием в 90-х годах глобальных информационных сетей, появлением «электронной» периодики, книг и огромных массивов прочей неструктурированной текстовой информации в компьютерной форме. Автоматическое распознавание в больших объемах текстовой информации документов по определенной тематике позволяет существенно снизить затраты на предварительный отбор информации из внешних источников для пополнения базы документов ИПС по соответствующей предметной области. Принцип решения таких задач аналогичен решению задач информационного оповещения (см. рис. 6.2). Для конкретного класса документов (рубрики) строится поисковый образ, который в системах на основе индексирования по ключевым терминам может быть представлен набором определенных терминов или их сочетаний. Поисковые образы документов из внешних источников сравниваются по определенному критерию с поисковым образом рубрики, и на этой основе принимается решение о внесении документов в базу, т. е. об отнесении содержания документа к предметной области ИПС.

Второй подход к автоматическому индексированию применяется в ***полнотекстовых*** системах. В процессе индексирования «на учет», т. е. в индекс заносится информация обо всех словах текста

документа (отсюда, как уже отмечалось, и название «полнотекстовые»). Более подробно особенности полнотекстового индексирования рассматриваются в следующем параграфе.

## 6.3. Полнотекстовые информационно-поисковые системы

Процессы массовой компьютеризации и информатизации деятельности предприятий, организаций в конце 80-х и в 90-х годах привели к накоплению огромных массивов неструктурированной текстовой компьютерной информации, с одной стороны, и доступности (всеобщей распространенности и персо-нальности) вычислительной техники, с другой стороны. Возникла потребность в программном инструментарии, который бы обеспечивал эффективный поиск нужных текстовых данных.

Семантические подходы к автоматизации такого рода задач (информационно-поисковые каталоги, фасетные и тезаурусные системы) не могли быть в полной мере использованы в массовой персональной автоматизации, т. е. на рабочем месте отдельного пользователя или для небольшой рабочей группы, так как требовали серьезной предварительной проработки соответствующей предметной области.* Потребовались средства, которые бы в максимальной степени освобождали пользователя от необходимости сложной предварительной структуризации предметной области и затратных процедур индексирования при накоплении, получении и агрегировании текстовых данных, но в то же время создавали бы эффективный и интуитивно понятный поисковый инструментарий необходимых документов.

---

* В этом отношении примечательным является следующее замечание — файловые системы ОС ПК предусматривают создание произвольной схемы логических дисков, каталогов, подкаталогов и т.п., которые по логике должны отображать структуру предметной области сведений пользователя ПК и, тем самым, в упрощенном утрированном виде решать задачи систематизации размещения документов-файлов для быстрого и эффективного их нахождения. Однако в большинстве случаев пользователями такая адекватная их потребностям система каталогов не создается из-за недостаточной их квалификации или нетривиальности самой структуры предметной области и данные зачастую размещаются довольно хаотично.

В результате на рынке программных продуктов в конце 80-х годов появились полнотекстовые ИПС и программные средства их создания, называемые иногда полнотекстовыми СУБД.

### 6.3.1. Информационно-технологическая структура полнотекстовых ИПС

Полнотекстовые ИПС строятся на основе *информационно-поисковых языков дескрипторного типа.* Их информационно-технологическая структура представлена на рис. 6.9 и включает следующие элементы:

- хранилище (базу) документов;
- глобальный словарь системы;
- индекс документов инвертированного типа;
- интерфейс ввода (постановки на учет) документов в систему;
- механизм (машину) индексирования;
- интерфейс запросов пользователя;
- механизм поиска документов (поисковую машину);
- механизм извлечения (доставки) найденных документов.

***Хранилище документов*** может быть организовано как единая *локально сосредоточенная информационная структура* в виде специального файла (файлов) с текстами документов. Организация такого файла предусматривает указательную конструкцию на основе массива адресов размещения документов. Для компактного хранения документов они могут быть сжаты архиваторами.

Другой вариант не предусматривает создания локально сосредоточенного хранилища документов, а ограничивается лишь массивом адресов расположения документов в соответствующей компьютерной информационной инфраструктуре (структура дисков и каталогов отдельного компьютера или локальной информационной сети, информационная инфраструктура глобальной информационной сети). Файлы текстовых документов *распределены* и размещаются в тех узлах и элементах информационной инфраструктуры, которые соответствуют технологии создания и обработки документов (документообороту). Вместе с тем все они *учтены* в полнотекстовой ИПС (т.е. проиндексированы по содержанию и зафиксированы по месторасположению) для эффективного поиска и доступа к ним. Такой подход более логичен с точки зрения технологий документооборота

или распределенного характера систем (например, система WWW сети Интернет), но недостатком имеет необходимость постоянного отслеживания и учета возможных перемещений документов.

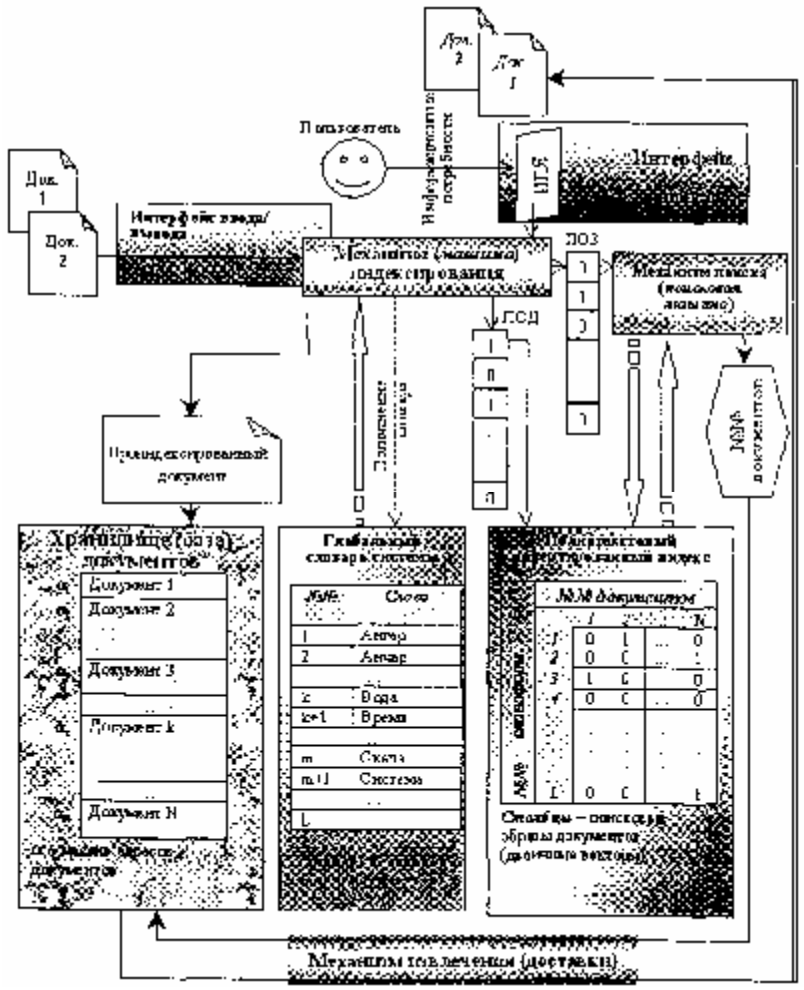



*Рис. 6.9. Информационно-технологическая структура полнотекстовых ИПС*

Одним из наиболее характерных элементов полнотекстовых ИПС является ***глобальный словарь системы.*** Глобальные словари могут быть статическими и динамическими.

*Статические словари* не зависят от содержания документов, вошедших в хранилище, а определены изначально в системе. В качестве таких статических словарей в том или ином виде, как правило, выступают словари основных словоформ соответствующего языка (русского, английского, немецкого и т. д.).

*Динамические словари* определяются набором словоформ, имеющихся в накапливаемых в хранилище документах. Изначально такой словарь пуст, но с каждым новым документом в него помещаются новые словоформы, которых еще не было в ранее накопленных документах. Такой подход более экономичен и обеспечивает некоторую настройку словарной базы на предметную область документов.

Элементы глобального словаря выступают в качестве *дескрипторов* ИПЯ системы. Поступающие через ***интерфейс ввода/вывода*** документы подвергаются операции *индексирования* по глобальному словарю. ***Механизм индексирования*** в полнотекстовых МПС полностью автоматизируется и заключается в создании специального двоичного вектора, компоненты которого показывают наличие или отсутствие в данном документе слова с соответствующим номером (позицией) из глобального словаря. В результате на «учет» в системе ставятся все слова текста документа, откуда, повторимся, происходит и название — «полнотекстовые ИПС».

Важной особенностью, оказывающей существенное влияние на *эффективность* ***полнотекстовых ИПС,*** является наличие либо отсутствие морфологического разбора при индексировании документов и запросов. *Морфологический разбор* позволяет распознавать как одну общую словоформу все однокоренные слова (вода, водный, водяной), а также лексемы, т. е. одни и те же слова, отличающиеся в тексте различными окончаниями, приставками и суффиксами (водный, водного, водному, воду, воде и т. п.). Такой процесс основывается на *нормализации* глобального словаря

системы, объединяющей в одну словоформу (в одну позицию) все однокоренные слова и лексемы. Кроме того, при морфологическом разборе отбрасываются так называемые неинформативные слова *(стоп-слова)* — предлоги, союзы, восклицания, междометия и некоторые другие грамматические категории. В большинстве случаев морфологический разбор осуществляется в системах со статическим глобальным словарем. Для русского языка в качестве такого нормализованного глобального словаря используется составленный в 1968 году академиком И. К. Зализняком морфологический словарь русского языка. Он позволяет распознать и соответственно нормализовать более 3 млн. словоформ.

В результате индексирования ПОД каждого нового документа представляется набором словоформ из глобального словаря, присутствующих в тексте документа, и поступает в виде соответствующего двоичного вектора для дополнения ***индекса системы.*** Индекс строится по инвертированной схеме и в двоичном виде отражает весь (полный) текст учтенных или накопленных документов.

При удалении документа из системы соответственно удаляется и поисковый образ документа, т. е. соответствующий столбец индекса.

Пользователь *языком запросов ИПЯ* полнотекстовой ИПС через соответствующий ***интерфейс запросов*** выражает свои информационные потребности по поиску документов, которые в общем плане, так же как и документы, индексируются и в виде двоичных векторов поисковых образов запросов поступают на ***поисковую машину. Механизм поиска*** основывается на тех или иных алгоритмах и критериях сравнения поискового образа запроса с поисковыми образами документов, образующими индекс системы. Результатом поиска является определение номеров документов, поисковые образы которых соответствуют или близки поисковому образу запроса. Далее специальная подсистема на основе установленных в хранилище документов указательных конструкций ***извлекает*** и ***доставляет*** соответствующие документы пользователю.

Таким образом, программное обеспечение полнотекстовых ИПС обеспечивает полный технологический цикл ввода, обработки, поиска и получения документов. В практическом плане ИПС могут поставляться как готовый информационный продукт, т. е. с уже сформированной базой документов и интерфейсом поиска и доступа к ним.* В других случаях поставляется программная среда, позволяющая такую базу создать и сформировать тем самым документальную информационно-поисковую систему. Такие программные средства иногда называют полнотекстовыми СУБД.

* Такими информационными продуктами, основанными в том числе и на полнотекстовых технологиях, являются многочисленные юридические информационно-справочные системы — «Кодекс», «Гарант», «Консультант плюс» и др.

### 6.3.2. Механизмы поиска документов в полнотекстовых ИПС

В полнотекстовых ИПС поиск документов осуществляется по индексу системы через дескрипторный язык запросов с логическими операциями над словоформами, а также через другие механизмы использования поисковых образов документов и запросов.

Принцип и механизм ***поиска документов по индексу*** системы очевидны. Пользователь должен указать путем перечисления и ввода в систему тех словоформ, набор которых выражает его информационные потребности. К примеру, если пользователю необходимо найти документы, содержание которых касается экспорта редкоземельных элементов, то *запрос к системе может выглядеть следующим образом «экспорт редкоземельные элементы».* В ответ система по индексу определит номера (группу) документов, где присутствует слово «экспорт», группу документов, где присутствует слово «редкоземельные», и группу документов, где присутствует слово «элементы». Ясно, что полнота и точность такого поиска будут оставлять желать много лучшего, так как в первой группе документов могут присутствовать в том числе и документы, в которых речь идет об экспорте чего-то другого, например леса, или об экспорте вообще. Во второй группе документов могут присутствовать документы, в которых речь идет, в том числе, о добыче или производстве редкоземельных элементов, но не об их экспорте. В третьей группе документов могут присутствовать и документы, в которых речь идет, скажем, о преступных элементах, что, конечно же, совершенно может не соответствовать благим информационным потребностям пользователя.

Слабая эффективность подобного *способа выражения информационных потребностей* преодолевается некоторыми *реляционными дополнениями* такого чисто дескрипторного языка запросов на основе ***посткоординации,*** только не понятий, а *словоформ.* В язык запросов вводятся логические операции

отношений дескрипторов запроса — операция логического «И», операция логического «ИЛИ», операция логического отрицания «НЕ».

Если словоформы запроса из приведенного выше примера объединить операцией логического «И», то система отберет только те документы, в которых одновременно присутствуют словоформы «Экспорт», «Редкоземельные», «Элементы». Несмотря на возможность ложной координации словоформ, такое усовершенствование чисто дескрипторного характера языка запросов приводит к существенному повышению эффективности поиска и предоставляет пользователю более развитые возможности по выражению своих информационных потребностей.

Следует также добавить, что подобные принципы построения языка запросов *повышают требования к квалификации пользователя,* в частности по пониманию и оперированию логическими операциями. Вместе с тем, как показывает практика, большинство так называемых «неподготовленных» пользователей способно самостоятельно осваивать и применять подобные, в общем-то, интуитивно понятные языковые конструкции.

На практике язык запросов полнотекстовой ИПС дополняется также операциями работы с датами и в ряде систем возможностями ***координатного анализа*** текста документов. Ранее неявно предполагалось, что единичным объектом поиска словоформ и соответственно областью действия логических операторов является документ, а не более мелкие его составляющие — абзацы, предложения. В системах с координатным анализом область действия логических операторов можно сужать вплоть до предложения. Примером таких возможностей является запрос на отыскание таких документов, где словоформы «экспорт», «редкоземельные», «элементы» присутствуют одновременно (операция «И») внутри одного предложения. Координатный анализ позволяет еще более повысить эффективность поиска релевантных документов, но требует более детального индексирования. Для словоформ словаря системы в индексе должны при осуществлении координатного анализа фиксироваться не только номера документов, но номера абзацев, номера предложений и номера соответствующих словоформ в порядке следования слов в соответствующих предложениях.

Отличительной особенностью поиска документов по индексу является *практическая независимость времени* (скорости) *поиски от объема базы документов,* особенно если используется статический словарь. Для любого запроса, независимо от текущего объема базы документов, выполняется приблизительно одинаковое количество операций, связанных с просмотром строк индексного массива и определением совокупности номеров релевантных документов. Следующей стадией выполнения запроса является собственно извлечение из базы (файла документов) самих документов. Для этого обычно в полнотекстовой ИПС создается специальный массив (см. рис. 6.9) адресов начала расположения документов.

В системах с динамически поддерживаемыми словарями время поиска при увеличении объема базы документов сначала также увеличивается (т. к. пропорционально увеличивается объем словаря и, соответственно, объем индекса), а затем так же, как в системах со статическими словарями, перестает зависеть от объема базы документов. Это объясняется тем, что с некоторой границы объема базы документов словарь системы уже набирает практически полный набор словоформ, присущих конкретной предметной области, и вероятность появления в новом документе слова, которого еще не было в словаре системы, резко падает.

Как уже отмечалось, повышению эффективности поиска способствует морфологический разбор документов и запросов. Помимо существенного уменьшения объема словаря и, соответственно, индекса системы, морфологический разбор повышает и эффективность поиска, так как не реагирует на несущественные с точки зрения смыслового содержания грамматические различия искомого текста документов и запросов. Если вернуться опять-таки к примеру с запросом «экспорт редкоземельные элементы», то система с морфологическим разбором отберет не только те документы, в которых встречается буквальное сочетание словоформ «экспорт», «редкоземельные», «элементы», но и такие фразы, как «К вопросу об экспорте редкоземельных элементов», «Проблемы экспорта редкоземельные элементов» и т.п.

Морфологический разбор в принципе дает возможность пользователю формировать запросы на *естественном языке.* Система при обработке запроса удаляет из него все «стоп-слова», остальные словоформы нормализует и, оставляя пользователя в полной иллюзии о том, что она действительно его «понимает», выполняет таким образом выхолощенный запрос. Некоторое время тому назад наблюдалось сильное увлечение таким подходом, от которого, к счастью, вскоре разработчики

полнотекстовых ИПС отошли. Использование якобы естественного языка запросов на самом деле не позволяет применять логические операторы и другие развитые возможности, связанные с координатным анализом местонахождения и контекстного окружения искомых слов, терминов, сочетаний и т. д.

Еще одной важной характеристикой поиска документов по индексу, в том числе с учетом логических операций посткоординации и морфологического разбора, является то, что такой поиск основывается на упрощенном детерминированном подходе. Иначе говоря, критерием поиска является вхождение или невхождение того или иного дескриптора-словоформы запроса в поисковый образ документа без учета общей «похожести» ПОД и ПОЗ. Масса остальных дескрипторов поискового образа документа не рассматривается. Поэтому в развитых полнотекстовых ИПС реализуются более тонкие и сложные алгоритмы поиска, основанные на сравнении ПОД и ПОЗ в целом по тем или иным критериям похожести, близости.

Такой подход позволяет предоставлять пользователям более эффективные возможности выражения своих информационных потребностей без их явной формализации и структуризации по словоформам. В частности, пользователь может поставить ИПС задачу поиска документов, *«похожих» по содержанию* на какой-либо другой (известный ему релевантный, точнее пертинентный) документ или фрагмент документа. В этом случае не только ПОД, но и ПОЗ представляют собой полномасштабные двоичные векторы, часть дескрипторов которых будет совпадать, а часть не совпадать, и возникнет необходимость в использовании более тонких критериев определения близости документов и запроса. Кроме того, становится возможным определение количественных мер (показателей) близости, т.е. релевантности документов и запросов.

### 6.3.3. Методы количественной оценки релевантности документов

Количественные показатели релевантности — процент соответствия содержимого документа запросу, ранжирование (самый релевантный документ, менее релевантный, еще менее релевантный) и т. п., позволяют существенно увеличить конечную эффективность использования документальной системы, предоставляя пользователю возможность после отбора документов сразу сосредоточиваться на наиболее важных из них.

Определение количественных показателей релевантности документов в полнотекстовых ИПС основывается на тех или иных подходах по вычислению мер близости двоичных векторов документов и запросов.

Документ $D_k$ представляется в системе двоичным вектором:

$$\vec{D}_k(d_{k,1}, d_{k,2}, ..., d_{k,L})$$

где $d_{k,i}$ =1, если словоформа под номером $i$ присутствует в $k$-м документе, и 0, если отсутствует.

Аналогичным образом представляются поисковые образы запроса Z пользователя:

$$\vec{Z}(z_1, z_2, ..., z_L)$$

где $z_k$ = 1, если словоформа под номером $k$ присутствует в запросе, и $z_k$ = 0, если отсутствует.

***Критерии релевантности*** подразделяются по *моделям представления* и *сопоставления документов* и *запросов, к которым* относятся:

- булева модель;
- модель нечетких множеств;
- пространственно-векторная модель;
- вероятностно-статистическая модель.

В качестве показателя (меры) релевантности документов используется так называемое *значение статуса выборки* (retrieval status value — *RSV).* В ***булевой модели*** критерием релевантности является *полное совпадение векторов ПОД и ПОЗ.* Соответственно *RSV* в булевой модели определяется как логическая сумма операций попарного логического произведения соответствующих элементов векторов ПОД и ПОЗ:

$$RSV_k = \sum_{i=1}^{L} d_{k,i} \,\&\, z_i ,$$

где $k$ = 1,..., $N$, $N$ — количество документов в базе, $L$ — количество словоформ в словаре, &—логическая операция «И».

Значением *RSV* в булевой модели может быть единица (релевантный документ) или ноль (нерелевантный документ). По сути, булева модель не дает количественной меры релевантности и ничем не отличается от простого поиска по индексу системы с логической операцией «И» словоформ-дескрипторов.

В системах на основе ***модели нечетких множеств*** значения компонент векторов ПОД и ПОЗ могут принимать не только два альтернативных значения —1 и 0 (термин принадлежит документу или не принадлежит), но и такое значение, как «неполная, частичная принадлежность». Соответственно в модели нечетких множеств переопределены и логические операции, чтобы учитывать возможность неполной принадлежности подобных логических элементов анализируемым множествам (поисковым образам запросов). Вычисление значений статуса выборки *RSV* производится аналогичным булевой модели образом с учетом переопределения операции & («И»).

Несмотря на некоторое расширение выразительных возможностей представления и сопоставления документов и запросов, модель нечетких множеств, как и булева модель, не дает по-настоящему количественной меры релевантности, хотя достоинством обеих моделей является их простота и невысокие вычислительные затраты на реализацию.

В системах на основе ***пространственно-векторных моделей*** поисковое пространство представлено многомерным пространством, каждое измерение которого соответствует словоформе (термину) из словаря системы. Например, если в словаре всего три словоформы, то поисковое пространство является трехмерным, и т. д. В исходном варианте пространство имеет евклидову метрику, т. е. представляется ортогональным базисом нормированных векторов, отражающих соответствующие словоформы словаря системы. Поисковый образ документа и запроса в поисковом пространстве представляется многомерным вектором единичной длины, координаты которого отражают наличие или отсутствие в документе соответствующих словоформ. В случае трехмерной размерности пространственно-векторная модель иллюстрируется на рис. 6.10.

*Показатель релевантности* (по аналогии с булевой моделью будем обозначать его *RSV)* для пространственно-векторной модели в простейшем случае определяется ***скалярным произведением векторов ПОД и ПОЗ:***

$$RSV_k = \sum_{i=1}^{t} d_{k,i} \cdot z_i .$$

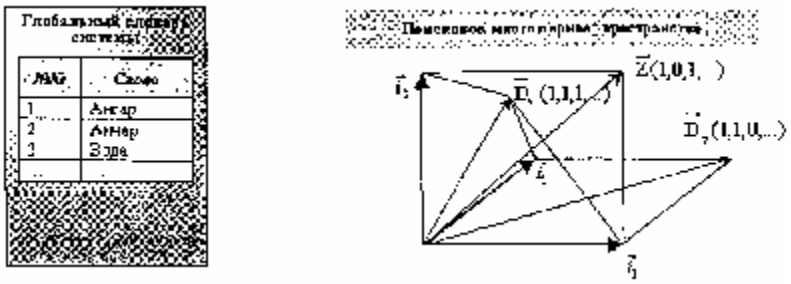



*Рис. 6.10. Иллюстрация пространственно-векторной модели представления и сопоставления документов и запросов*

Таким образом, определяемый показатель релевантности *RSV* может изменяться в диапазоне от 0 до *N(N* — число словоформ или терминов в словаре системы) и действительно количественно отражает степень релевантности документов. Так, в приведенном на рис. 6.10 примере значение $RSV_1 = 2$, а значение $RSV_2 = 1$. Для выдачи пользователю конкретного набора релевантных документов информационно-поисковые системы ограничиваются выдачей документов, показатель релевантности которых запросу *RSV* превышает некоторый заранее установленный порог.

Следует также заметить, что при таком подходе абсолютные значения показателя релевантности зависят не только собственно от самой степени релевантности, но и от количества *N* словоформ в словаре системы. Поэтому на практике применяют *нормализованный вариант RSV,* определяя его с учетом ортогональности и ортонормированности поискового пространства как косинус угла между вектором ПОД и вектором ПОЗ:

$$RSV_k = \frac{\sum_{i=1}^{t} d_{k,i} \cdot z_i}{N} .$$

В этом случае *RSV* принимает значения от 0 до 1 и не зависит от объема словаря системы.

Определенным недостатком такого подхода к расчету количественной меры релевантности является *нечувствительность к степени* ***соответствия отсутствующих словоформ*** *(терминов) в ПОД и ПОЗ.* Интуитивно понятно, что чем ближе содержание документа и запроса, тем меньше в документе должно быть словоформ (терминов), которых нет в запросе. Если, к примеру, в словаре системы всего 6 элементов и имеется два документа $\vec{Д}_1(1,1,0,1,0,0)$ и $\vec{Д}_2(1,1,1,1,1,1)$, то для запроса $\vec{З}(1,1,0,0,0,0)$ значение *RSV* для обоих документов будет равно 2 (33%), хотя интуитивно понятно, что более близким по содержанию является первый документ, а второй документ, скорее всего, затрагивает более широкую тематику, не обязательно интересующую пользователя.

Такой *чувствительностью* обладает показатель релевантности, определяемый следующим образом:

$$RSV_k = \sum_{i=1}^{L} d_{k,i} \cdot z_i + \sum_{i=1}^{L} d_{k,i} \cdot \bar{z}_i ,$$

где $\bar{d}_{k,i}$ и $\bar{z}_i$ — дополнение к элементам $d_{k,i}$ и $z_i$, т. е. $\bar{d}_{k,i} = 1$, если $d_{k,i} = 0$ и наоборот.

Если вернуться к предыдущему примеру с документами $\vec{Д}_1(1,1,0,1,0,0)$, $\vec{Д}_2(1,1,1,1,1,1)$ и запросом $\vec{З}(1,1,0,0,0,0)$, то *RSV* для первого документа будет равным 5 (83%), а для второго документа 2 (33%), что выглядит, конечно же, «справедливее».

Более развитым, но и более сложным подходом к определению мер близости ПОД и ПОЗ ***является учет разной значимости словоформ*** (терминов) ***и их зависимости друг от друга***. В пространственно-векторной модели это означает *отход от ортогональности и ортонормированности базисных векторов поискового пространства.* В этом случае скалярное произведение векторов ПОД и ПОЗ более гибко и осмысленно отражает близость соответствующих векторов и, тем самым, смысловое содержание документов и запросов.

В простейшем варианте подобного расширения пространственно-векторной модели различные словоформы в глобальном словаре системы дополняются ***специальными весовыми коэффициентами,*** отражающими *важность соответствующей словоформы (термина) для конкретной предметной области.* Соответственно поисковые векторы документов и запросов в этом случае превращаются из двоичных векторов в обычные, т. е. с любыми значениями (а не только 0 или 1) своих компонент. Иногда такой подход называют «окрашиванием»* глобального словаря системы. Следует также заметить, что в случае перехода от глобального словаря (отражающего все слова и словоформы) к словарю терминов происходит вырождение полнотекстового характера ИПС и она переходит в категорию систем на основе тезаурусов.

---

* В смысле окрашивания по определенной предметной области.

На практике применяются также и другие подходы, расширяющие возможности двоичной (ортогональной и ортонормированной) пространственно-векторной модели. Такие подходы базируются на ***вероятностно-статистической модели.*** При этом можно выделить две разновидности вероятностно-статистического подхода:

- придание весовых коэффициентов словоформам (терминам) глобального словаря вне контекста конкретного документа;
- придание весовых коэффициентов компонентам векторов ПОД по итогам индексирования конкретного документа (с учетом контекста конкретного документа).

*Первый подход* основан *на анализе итогов индексирования совокупности документов, уже вошедших в базу* (хранилище) ИПС. Совокупность словоформ (терминов), обязательно присутствующих в любом документе базы, считается наиболее адекватно отражающей тематику предметной области ИПС, и соответствующие словоформы (термины предметной области) получают наибольший вес, наибольшую значимость в словаре системы, по которому производится индексирование документов. В качестве *числовых характеристик весов значимости терминов* используются те или иные ***статистические параметры,*** такие, например, как относительная или абсолютная частота вхождения термина в документы базы системы. Разновидностью такого подхода является *учет количества вхождений в совокупность документов* базы тех или иных *словоформ или терминов.*

Более сложные варианты развития *первого подхода* основываются на технологиях *«обучения»* и *настраивания* ИПС на конкретные предметные области. Традиционный способ обучения основывается на использовании ***обучающей выборки документов.*** Такая выборка формируется либо на основе отбора текстов экспертами в конкретной предметной области, либо путем использования

документов по соответствующим рубрикам каталогов библиотек и т. п. Далее осуществляется исследование обучающей выборки на предмет статистических показателей вхождений в документы выборки тех или иных словоформ или терминов. Результатом обучения является *«окрашенность»* (различные весовые коэффициенты словоформ) словаря системы.

*Другой подход* основывается на *апостериорном выделении* в поисковом пространстве ***«сгущений»*** векторов ПОД и последующем анализе совокупности и количественных данных вхождения в такие группы документов тех или иных словоформ (терминов). Предполагается, что такие группы соответствуют особенностям тематики конкретной предметной области, и словоформы, в них входящие, получают наибольшие весовые коэффициенты на основе тех или иных статистических параметров. Еще одним вариантом является *учет* ***дискриминируемости (различимости) термина***. Если при внесении в текст одного из двух близких по векторам ПОД документов какого-либо термина происходит резкое *«расщепление»* этих векторов, то такой термин считается *более информативным и значимым,* и его коэффициент важности, соответственно, должен быть выше.

При втором подходе к реализации вероятностно-статистической модели *различия в весах значимости словоформ или терминов* проявляются ***по результатам индексирования конкретного документа.*** В простейшем варианте анализируется, *сколько раз тот или иной термин входит в данный документ.* Словоформам или терминам, имеющим наибольшее количество вхождений, присваиваются более высокие веса в векторе ПОД. В векторах запросов (ПОЗ) все словоформы или термины считаются равнозначными, но их различные веса в векторах ПОД обеспечивают большую релевантность тех документов, где соответствующие словоформы или термины встречаются наиболее часто.

Отдельной ветвью развития второго подхода является ***использование обратной, интерактивной связи с пользователем.*** В этом случае информационно-поисковая система стремится настроиться не столько на определенную предметную область, сколько на специфические особенности тематики информационных потребностей конкретного пользователя. В общем виде *для каждого пользователя ИПС* создает *свое поисковое пространство* с индивидуальным окрашиванием компонентов векторов ПОД. Такое *индивидуальное окрашивание* производится путем *запрашивания системой у пользователя его оценки релевантности выданных на каждый текущий запрос документов.* Уточнив у пользователя, какие на его взгляд документы наиболее релевантны, система анализирует особенности и статистические параметры вхождения тех или иных словоформ (терминов) в эти наиболее релевантные документы, переопределяет и уточняет их весовые коэффициенты. Тем самым в последующих запросах более адекватно и глубже учитываются информационные потребности конкретного пользователя.

Существуют и другие разновидности вероятностно-статистических подходов к расширению пространственно-векторной модели поиска документов, но, к сожалению, из-за отсутствия в документации на коммерческие ИПС соответствующей информации по деталям механизмов поиска и релевантности документов оценить и проанализировать их эффективность довольно затруднительно.

В целом же информационно-поисковые полнотекстовые системы являются одним из наиболее интенсивно развивающихся направлений документальных информационных систем, существенно продвигая теорию и практику информационного поиска документов и развивая методы анализа и автоматизированной обработки текстовой неструктурированной информации.

## 6.4. Гипертекстовые информационно-поисковые системы

Анализ организации работы различных аналитических служб и отдельно взятого аналитика показывает, что основой их информационного обеспечения в традиционных «бумажных» технологиях являются различные *тематические подборки,* папки с текстовыми документами (служебные документы, копии статей из специальной периодики, выписки из книг, газетные вырезки и т. п.), систематизированные по расположению на основе какого-либо критерия (в алфавитном порядке по названиям, хронологически по дате документов, ранжированием по важности или по иным критериям). Причем документы в таких папках-подборках, как правило, снабжаются еще специальными *пометками и взаимными отсылками* по каким-либо смысловым ассоциациям. Отталкиваясь от какого-либо одного, релевантного документа, аналитик по отсылкам отбирает из подборки и все, ассоциированные по данному смысловому содержанию, документы. Процесс отбора документов по

ссылкам в определенной степени напоминает *навигацию* по географическим картам, чем и определяется название соответствующего подхода к организации документального поиска.

В отличие от информационно-поисковых систем на основе индексирования документов, семантически-навигационные системы изначально возникли и развивались как чисто компьютерные системы и прошли пока еще короткий, но уже достаточно богатый период развития.

Считается, что первым идеи ***ассоциативно-навигационного подхода к анализу текстовой информации*** выдвинул в 1945 году советник президента Рузвельта по науке Ванневар Буш. В своей статье «Как мы могли бы мыслить», где он излагал проект создания технической (точнее, (фотомеханической) системы, обеспечивающей «ассоциативное» связывание текстов, В. Буш писал: «Работа человеческой мысли построена на принципе ассоциаций. Анализируя какое-либо понятие или элемент, она непременно стремится поставить ему в соответствие какой-нибудь другой знакомый образ, подсказываемый ассоциацией мыслей, и это соответствие устанавливается благодаря трудноуловимой паутине связей, формируемых клетками человеческого мозга».* Идеи В. Буша, как это иногда бывает, намного опередили свое время, и потребовался более чем 20-летний период накопления опыта работы с компьютерной информацией, пока в 70-х годах не были предприняты первые попытки практической реализации систем с ассоциативным связыванием текстов, выразившиеся в технике так *называемого* ***гипертекста.***

* Цитируется по работе: **Елисеев В., Ладыженский Г.** Введение в Интранет / СУБД. — 1996. — № 5-6. — С. 24.

### 6.4.1. Гипертекст

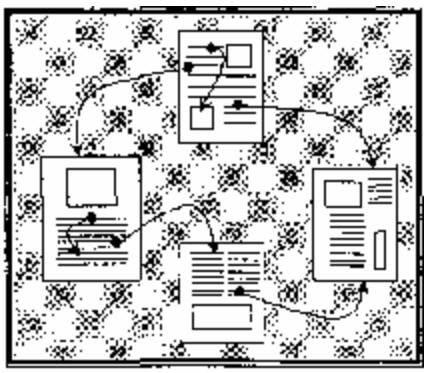


*Рис. 6.11. Принцип гипертекста*

**Гипертекст** в узком смысле представляет собой обычный текст, содержащий *ссылки на другие связанные по смыслу фрагменты того же текста (документа) или на другие тексты (на внешние документы)*. При этом ссылки для пользователя-читателя в тексте имеют вид выделенных слов или словосочетаний, обладающих какой-либо смысловой связью с текстом того фрагмента или другого текста, куда «направляет» ссылка (так называемая ***гиперссылка***).

Программное средство, отображающее гипертекст, например текстовый редактор или броузер сети Интернет, обеспечивает отображение гипертекста и навигацию пользователя-читателя по гиперссылкам. «Щелкнув» мышью по выделенному слову (т. е. по гиперссылке), пользователь-читатель открывает связанный по ссылке текст (другой фрагмент этого же текста или другой текст).

Привычным «бумажным» аналогом гипертекста являются оглавления и предметные указатели книг, содержащие ссылки на главы, разделы или фрагменты книги с соответствующей информацией. При этом ссылка выглядит как номер страницы, с которой начинается соответствующая глава или раздел, где находится соответствующий фрагмент текста. Отобрав в оглавлении или предметном указателе нужное название или термин и считав номер соответствующей страницы, читатель открывает книгу в искомом месте, т. е. переходит, или, выражаясь по-другому, осуществляет «навигацию» в нужное место книги.

В 70-е и 80-е годы, в особенности в период «персонализации» вычислительной техники, были предприняты многочисленные попытки создания специальных гипертекстовых оболочек, на основе которых либо совершенствовался примитивный текстово-командный интерфейс ранних операционных систем (знаменитая оболочка «Norton Commander» для ОС MS DOS), либо для прикладных программных средств создавались гипертекстовые справочные (help-овые) системы и руководства.

В конце 80-х — начале 90-х годов были предприняты первые попытки стандартизации гипертекста. Таким стандартом являлся стандарт American Cybernetics Hypertext System (ACI Hypertext), реализованный в среде встроенной системы макрокоманд широко известного в «узких» программистских кругах текстового редактора MultiEdit.

Впоследствии гипертекст стал широко использоваться в справочных системах программ-приложений операционной системы Windows и фирмой MicroSoft был разработан специальный пакет WinHelp для создания гипертекстовых справочных «систем помощи». В настоящее время техника гипертекста является фактическим стандартом создания разнообразных компьютерных справочных и учебных систем, руководств пользователя и энциклопедий.

Период взрывной интенсификации применения технологий гипертекста связан с бурным развитием и распространением в конце 80-х — начале 90-х годов глобальных информационных систем, и, в частности, сети Интернет. Идеи гипертекста как принципа ассоциативного связывания в распределенную информационную среду документов на территориально удаленных компьютерах были использованы группой специалистов под руководством Теодора Нельсона, который в 1988 г. представил проект гипертекстовой системы ***Xanadu*,** финансировавшийся впоследствии основателем известной компании Autodesk Джоном Уокером, который в то время пророчески предвещал все-объемлющее развитие и распространение гипертекстовых технологий. В 1989 г. в Лаборатории физики элементарных частиц европейского центра ядерных исследований (ЦЕРН) под руководством Тима Бернерса-Ли стартовал проект создания гипертекстовой системы обмена научными данными в сети Интернет, получивший впоследствии название «Всемирной паутины» — World-Wide Web (WWW). В 90-х годах паутина WWW стала одним из наиболее бурно развивающихся сегментов сети Интернет, создав немыслимую ранее глобальную гипертекстовую информационную инфраструктуру.

### 6.4.2. Структура, принципы построения и использования гипертекстовых ИПС

В *структуре* гипертекстовой ИПС можно выделить несколько функциональных подсистем (см. рис. 6.12). Основными из них являются:

- подсистема отображения документов и гиперссылок;
- подсистема навигации по связям (гиперссылкам);
- подсистема формирования связей (гиперссылок);
- и собственно сама гипертекстовая база (хранилище) документов.

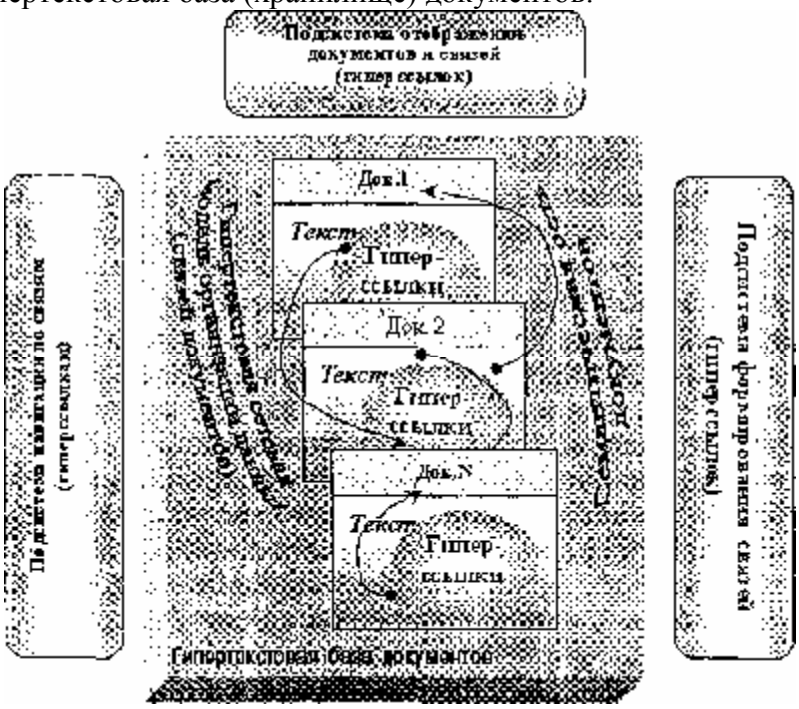



*Рис. 6.12. Структура гипертекстовой ИПС*

***Подсистема отображения документов и гиперссылок*** (гипертекста) базируется на принципах отображения документов в текстовых редакторах (страницы, поля, абзацы, шрифт, скроллинг и т. д.) с дополнительными приемами внешнего отображения в тексте гиперссылок. Как уже отмечалось, стандартным способом отображения *гиперссылок* является выделение в тексте специальным фоном,

цветом или шрифтом ключевых слов, имеющих определенную *смысловую* связь с тем фрагментом или документом, на который указывает ссылка. В развитых гипертекстовых системах, как, например, в системе WWW, в гипертексте могут отображаться также графика (рисунки, диаграммы), звуковые и даже видеоанимационные элементы, что в совокупности создает ***мультимедииную технологию*** работы с информацией. В этом случае в качестве гиперссылок могут также выступать и специальные изображения, значки, иконки, что дает возможность использования для отображения связей различных графических ассоциаций. В остальном подсистема отображения гипертекста напоминает обычный текстовый редактор, допуская стандартные операции просмотра (скроллинг, масштаб) и обработки текста (копирование, контекстный поиск и т. д.).

***Подсистема навигации по связям*** реализует специальный интерфейс перехода по гиперссылкам. Если гиперссылка указывает на другой фрагмент того же документа, то подсистема навигации обеспечивает скроллинг (прокрутку) отображения текста к соответствующему фрагменту. Если гиперссылка указывает на внешний документ, то стандартным приемом для систем, реализованных в оконно-графических операционных средах (MS Windows), является открытие в новом окне соответствующего документа. Приемом инициализации перехода по гиперссылке обычно является «щелчок мышью» по ключевому слову или графическому значку, обозначающему соответствующую гиперссылку, либо перевод текстового курсора на соответствующую гиперссылку и нажатие клавиши «Enter».

Для осуществления навигации в гипертекстовом документе *для каждой гиперссылки* хранится *адрес* расположения соответствующего документа или фрагмента. В современных гипертекстовых средах для удобства ориентирования пользователя применяется специальный прием «подсказки» адреса гиперссылки при осуществлении подготовительных операций перед ее активизацией (т. е. при переводе курсора мыши или текстового курсора на гиперссылку непосредственно перед щелчком или нажатием клавиши «Enter»).

***Навигация по гиперссылкам*** формирует для пользователя определенный ***сюжетно-тематический поток по цепочке ассоциаций.*** Нетривиальной проблемой, как и при навигации в банках фактографических систем с сетевой моделью организации данных, является способ отображения и ***визуализации*** *цепочек «пройденных» документов.* Так как такие цепочки документов могут быть неопределенно длинными, то открытие и отображение каждого следующего по проходу документа в дополнительном окне приводят к быстрому заполнению, а потом и наслоению окон с документами на экране компьютера. При этом документ, на который указывает гиперссылка из другого документа, может помимо непосредственной ассоциации включать и совершенно иной *содержательный контекст,* что быстро «уводит» пользователя от основной темы и дезориентирует его. Поэтому в большинстве систем используется только одно окно для отображения документов, а при переходе по гиперссылке к связанному документу происходит «выталкивание» предыдущего документа в специальный неотображаемый стек для пройденных документов. Дополнительно обеспечивается свободная навигация по сформированной таким образом цепочке документов (по пройденному пути) ***по принципу «Вперед-Назад»,*** что позволяет пользователю путем возвращений назад или перемещений вперед лучше анализировать сюжетно-тематическии поток ассоциаций.

*Способ формирования и отображения цепочки* пройденных документов по линейному принципу «Вперед-Назад» не всегда адекватно позволяет представить схему сюжетно-тематического потока документов из-за наличия возможных ветвлений в таких цепочках. Если из какого-либо документа (узла цепочки) имеется несколько гиперссылок на различные документы, то сценарием «разговора» пользователя с гипертекстовой базой может быть «спуск» от такого документа по имеющимся ветвям на определенную глубину, с последующим возвратом (подъемом) и спуском по другим ветвям. Линейно-списочный способ отображения цепочек пройденных документов в этом случае из-за многочисленных возвратов не дает общего представления и взгляда на ассоциативную окрестность связанных документов (см. рис. 6.13).

При наличии только *иерархических связей* между пройденными документами отработанным приемом отображения ***структуры ассоциативной цепочки пройденных документов*** может быть способ отображения файловой структуры информационных ресурсов компьютера, используемый в программах типа «Проводник» операционной системы MS Windows 95.

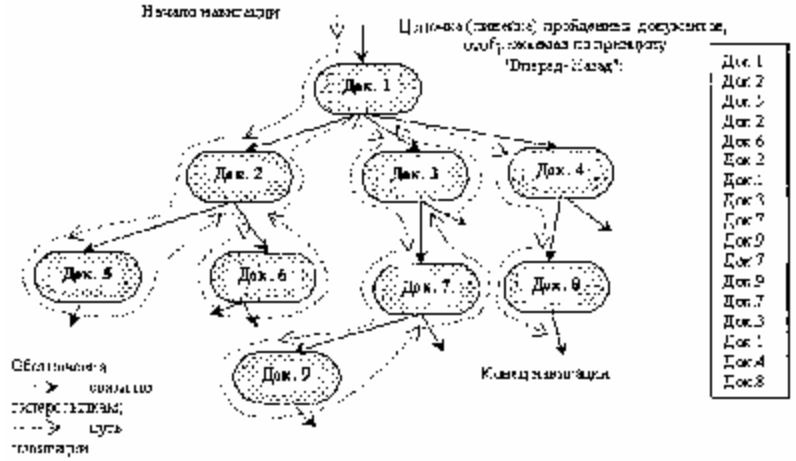



*Рис. 6.13. Навигация по гипертекстовой базе документов и отображение цепочек пройденных документов*

Однако гипертекстовые сети документов, как будет рассмотрено ниже, являются не иерархическими, а ***гетерогенными.*** В гетерогенных сетях могут существовать как одноуровневые и межуровневые связи, так и обратные связи (отсылки), что вырождает само понятие иерархии в таких сетях. Наглядно такие структуры можно представить в виде неограниченной совокупности объемно переплетенной паутины узлов, хотя в отдельных сегментах таких структур могут в определенной степени сохраняться иерархические отношения. Отсюда, видимо, и родилось соответствующее название для распределенной гипертекстовой среды сети Интернет. «Блуждание» по подобным «лабиринтам» может образовывать столь запутанные «следы», что их визуально-наглядное отображение весьма затруднительно.

Вместе с тем ***визуализация информационного поиска*** документов является чрезвычайно актуальной задачей, так как может предоставлять пользователям дополнительные аспекты анализа информации при аналитических исследованиях. Определенные методологические подходы к решению таких задач могут быть найдены на основе анализа семантической природы гетерогенных сетей гипертекстовых документов.

### 6.4.3. Модель организации данных в гипертекстовых ИПС

К сожалению, несмотря на интенсивное развитие и всеобщее распространение в последнее десятилетие гипертекстовых технологий, к настоящему времени еще не проработана полностью формализованная модель организации гипертекстовых данных, которая бы обеспечивала формализованные процедуры синтеза (разработки, проектирования) и анализа (использования) гипертекстовых ИПС. Причина этого заключается, как и в целом для всех типов документальных систем, в пока непреодолимых сложностях в формализованном описании смысла текстов на естественном языке.

Тем не менее в научной литературе имеется ряд работ, посвященных формальным моделям гипертекстовых структур.* Среди них можно выделить *теорию паттернов,* разработанную американским математиком У. Гренандером и развитую впоследствии для гипертекста Л. В. Шуткиным, *тензорную модель* А.В. Нестерова и подход *логико-смыслового моделирования,* представленный в работах М. М. Субботина, а также ряд других подходов.

* См., например, **Купер И. Р.** Обзор отечественных гипертекстовых технологии // Теория и практика общественно-научной информации. — Вып. 13. — 1997.

Первые два подхода основываются на формализации отдельных текстов специальными математическими конструкциями. В теории паттернов текст рассматривается как сложноорганизованная совокупность отдельных тем, каждая из которых может выражаться фрагментом текста с минимальным размером в виде одной строки. Для описания гипертекста в ***теории паттернов*** вводятся также специальные объекты — кнопки (аналог гиперссылки) и связи с идентификаторами и дополнительными параметрами (тип, направленность и т. д.). В результате размеченный гипертекст можно описывать теми или иными паттерновыми конфигурациями. Вместе с тем теория паттернов не содержит средств синтеза обычного текста в гипертекст.

***Тензорный подход*** основывается на идеологии ранее рассматривающейся фасетной классификации, которая позволяет формализовано описать смысловую структуру текста в виде тензора,* а гипертекстовую структуру в виде ансамбля тензоров. Таким образом, сильной стороной тензорного

подхода является возможность создания формализованных процедур анализа исходных текстов для создания гипертекстовых структур.

* В упрощенном виде тензор можно трактовать как математический объект, завершающий иерархически усложняющуюся цепочку—«скаляр—вектор—тензор», т. е. как многокомпонентный объект в многомерном пространстве с заданным линейным преобразованием его компонент при переходе от одной системы координат к другой.

Наиболее развитым в практическом плане является подход, основанный на ***логико-смысловом моделировании*** человеческого мышления, позволяющий на основе семантической близости текстовых фрагментов связывать их в цельный осмысленный текст — семантическую сеть. Математическим аппаратом для описания структуры гипертекста выступает теория графов. Критерием для связывания текстов или их фрагментов в семантическую сеть является возможность установления между ними логических связок типа «есть», «является условием», «является причиной» и т. д. Построение на основе анализа текста таких связываний образует формализованные «высказывания», комбинируя которые можно получать определенные выводы или, как говорят, новые знания, или подтверждать истинность (доказывать) составных высказываний. В наиболее развитом виде такой подход реализуется в так называемых базах знаний, составляющих основу особой ветви информационных систем, называемых экспертными системами.

Таким образом, при логико-смысловом моделировании структура гипертекста представляет (точнее, должна представлять) систему семантических связей между когнитивными элементами (понятиями, высказываниями) определенной предметной области. В результате сильной стороной такого подхода является возможность автоматизации создания (разметки) гипертекстовых структур на основе распознавания и соотнесения документов или их фрагментов к тем или иным узлам семантической сети.

Если вернуться к структуре гипертекстовой ИПС (рис. 6.12), то ее центральным элементом является ***гипертекстовая база документов***. По принципу формирования и управления гипертекстовыми базами их можно разделить на *открытые* (физически распределенные, или децентрализованные) и *замкнутые* (локально сосредоточенные).

В ***замкнутых базах*** гипертекстовые документы находятся в *едином локально-сосредоточенном и централизованно управляемом хранилище* (файле или группе файлов со специальным (форматом). Такое хранилище образует ***замкнутую семантическую сеть документов,*** гипертекстовые связи которых *не выходят за пределы хранилища.* Соответственно внесение в базу новых документов или удаление документов производится непосредственно в месте расположения такой локальной базы.

В ***открытых базах*** гипертекстовые документы не образуют единое локально размещенное хранилище, а располагаются автономно в любых элементах (узлах) информационной среды. При этом информационная среда может ограничиваться файловой структурой одного компьютера (диски, каталоги, подкаталоги), локальной или глобальной информационной сетью. В открытых базах семантическая гипертекстовая сеть документов не управляется из одного центра (узла), а совместно строится и поддерживается всеми пользователями, работающими в узлах информационной среды (сети). Несмотря на *полную децентрализацию* создания и функционирования, при определенных соглашениях (протоколах) об установлении и поддержании связей-гиперссылок, такие открытые семантические структуры тем не менее представляют единый развивающийся по определенным закономерностям организм.

В настоящее время техника гиперссылок, применяемая в гипертекстовых системах, предполагает лишь ***однонаправленные связи,*** позволяющие осуществлять *навигацию только в прямом направлении.* «Вернуться» обратно в исходный документ можно только по запомненной цепочке пройденных документов, т. е. по схеме «Вперед-Назад». При этом прямой переход по гиперссылке осуществляется из определенного места, точнее контекста исходного документа, а возврат осуществляется обратно в документ в целом, т. е. фактически в его начало, что может разрывать контекст (сюжетно-тематический поток) анализа информации. В ранних гипертекстовых системах (проект *Xanadu)* предполагался двунаправленный характер гиперссылок, но практическая реализация такого подхода существенно усложняет протоколы навигации, так как требует более детального координатного адресования объектов и субъектов гиперссылок, идентифицирования пользователей и поддержания устойчивости документов (в смысле координатной структуры).

В результате ***модель организации дачных в гипертекстовых базах*** описывается *ориентированными невзвешенными графами с петлями и циклами.* По определению граф ***G*** представляет структуру,

состоящую из множества вершин $x_1, x_2, ..., x_n$ и множества ребер $a_1, a_2, ..., a_n$, их соединяющих. По ребрам осуществляется движение, переход от одной вершины к другой. Ориентированные ребра, по которым переход возможен только в одном направлении, называются дугами. Применительно к структуре гипертекстовой базы вершины графа соответствуют документам, а дуги гиперссылкам. Невзвешенность означает равнозначность любых дуг по переходу, или, иначе говоря, одинаковую «стоимость» перехода по любой гиперссылке. Петлей называется дуга, начальная и конечная вершины которой совпадают, т. е. применительно к гипертексту внутренняя гипер-ссылка на другой фрагмент того же документа. Путем (или ориентированным маршрутом) называется последовательность дуг, в которой конечная вершина любой дуги, кроме последней, является начальной вершиной следующей дуги. В невзвешенном графе, когда стоимость (вес) всех дуг одинакова, длиной пути является число дуг, входящих в путь. Путь $a_1, a_2, ..., a_q$ называется замкнутым, если в нем начальная вершина первой дуги $a_1$ совпадает с конечной вершиной последней дуги $a_q$. Если в замкнутом пути любая вершина графа используется не более одного раза (за исключением начальной и конечной, которые совпадают), то такой замкнутый путь называется циклом. Пример графа приведен на рис. 6.14.

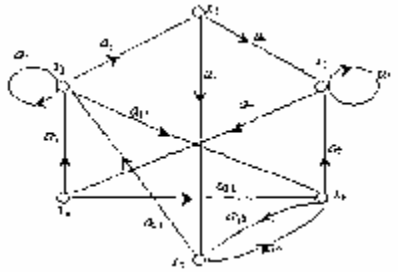


*Рис. 6.14. Пример невзвешенного графа с петлями и циклами*

Для *алгебраического задания графов,* позволяющего эффективно алгоритмизировать машинное представление и оперирование графами, используются *матрицы смежности* и *инциденций.* Элементы $\varphi_{ij}$ матрицы смежности $\Psi$ графа ***G*** определяются следующим образом:

$\varphi_{ij} = 1$, если в ***G*** существует дуга $a_{ij}$;

$\varphi_{ij} = 0$, если в ***G*** не существует дуга $a_{ij}$.

Матрица смежности полностью определяет структуру графа. В частности, для графа, приведенного на рис. 6.14, матрица смежности выглядит следующим образом:

$$\Psi = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Матрица инциденций $\Omega$ графа ***G*** с n вершинами и m дугами представляет собой матрицу размерности nxm и ее элементы $\sigma_{ij}$ определяются следующим образом:

$\sigma_{ij} = 1$, если вершина $x_i$ является начальной вершиной дуги $a_j$;

$\sigma_{ij} = -1$, если вершина $x_i$ является конечной вершиной дуги $a_j$;

$\sigma_{ij} = 0$, если вершина $x_i$ не является концевой вершиной дуги $a_j$ или если дуга $a_j$ является петлей.

Графовая модель организации гипертекстовых данных является мощным инструментом, так как предоставляет ряд отработанных в теории графов алгоритмов для решения задач анализа и синтеза структур гипертекстовых баз данных, навигации и документального поиска в такого рода структурах.

Вместе с тем, как показала практика развития гипертекстовых структур, модель ориентированных невзвешенных графов с петлями и циклами является лишь приближенным средством отражения

реального процесса восприятия и анализа человеком документальной текстовой информации, не учитывая ряда гносеологических и семантических аспектов.

Анализ работы человека с документальными источниками информации показывает, что *ассоциативный ряд восприятия фрагментов и документов не однороден.* ***Ассоциативные отношения*** выражаются в нескольких *формах,* в качестве основных из которых можно отметить:

(a) ***сноски*** (переходы к ним используются с целью пояснения какого-либо термина, факта и т. д. с обязательным и скорым возвратом, т. е. без прерывания контекста восприятия основного повествования, мысли, идеи);

(b) ***примеры*** (переходы по ним используются для иллюстрации частных проявлений объектов, процессов, явлений, и также с обязательным и скорым возвратом без прерывания основного контекста);

(c) ***отступления, параллельные темы*** (переходы к ним используются для обогащения основной темы с необязательным или нескорым возвратом, что может приводить к прерыванию контекста изложения основной темы);

(d) ***подобие по форме и содержанию*** (переходы используются для более глубокого уяснения основной темы через анализ других подобных по форме, содержанию, структуре или другим критериям тем, фрагментов, объектов, в том числе для рассмотрения других точек зрения и подходов, с необязательным возвратом, что приводит к длительному прерыванию исходного контекста с возможным формированием нового контекста);

(e) ***особенности*** (переходы используются для рассмотрения отличий конкретной темы или объекта изложения от подобных по форме или содержанию объектов с обязательным возвратом без прерывания основного контекста);

(f) ***подобие по сущности*** (переходы используются для построения ассоциативного ряда подобных или однородных объектов, являющихся частными проявлениями одного общего явления процесса, объекта, возврат не обязателен, что приводит к прерыванию исходного контекста, в том числе и для формирования более общего или более широкого контекста).

Перечисленные формы ассоциативных отношений определяют необходимость *дифференциации* ***типов*** *связей-гиперссылок* в гипертекстовых базах документов. По признаку прерывания контекста материала можно выделить *два типа* гиперссылок:

• ***с прерыванием контекста,*** назовем их ***навигационными*** гиперссылками;

• ***без прерывания контекста,*** т. е. с обязательным возвратом, назовем их ***листовыми*** гиперссылками.

Навигационные гиперссылки формируют ассоциативные связи-отношения (c), (d) и (f) типа. Переходы по *навигационным связям* не имеют каких-либо пространственных и иных ограничений и призваны ***формировать многоплановый сюжетно-тематический поток.***

*Листовые* гиперссылки формируют *ассоциативные связи-отношения* (a), (b) и (e) типа. Переходы по листовым гиперссылкам ограничиваются единичной длиной к вершинам (узлам), из которых нет другого выхода. Направленность дуг-связей по листовым гиперссылкам является обратной по отношению к навигационным гиперссылкам. Это означает, что прямой переход по ним осуществляется не в конкретное место отсылаемого документа, а в целом на документ (в начало) листовой вершины, и наоборот, возврат в документ исходной вершины происходит адресно, т. е. в место расположения листовой гиперссылки.

Кроме ассоциативных отношений при восприятии документальных источников важную роль имеют и ***классификационные отношения фрагментов и документов*** в следующих основных формах:

i) ***«родо-видовая»*** иерархия (переходы используются для углубления, детализации рассмотрения или выбора темы, (фрагмента, сюжета);

ii) ***иерархически-логические*** соотношения в форме «вводный материал — основной материал — заключительный материал» (переходы используются для построения или изменения логико-тематического повествования);

iii) ***ролевые отношения,*** например такие, как «Объект-субъект-средство-место-время-участники действия» и др. (переходы используются для формирования или расчленения целостного представления сложных разноплановых явлений, процессов, событий).

Реализация дифференцированного подхода к образованию и использованию гиперссылок в открытых децентрализованно развивающихся системах является непростой проблемой, так как требует переработки и усложнения протоколов передачи и использования гипертекста, т. е. массового

принятия в сети новых и более сложных правил всеми пользователями и разработчиками информационных узлов распределенной гипертекстовой информационной инфраструктуры.

Поэтому подходы, связанные с *дифференциацией характера гиперссылок,* нашли свое воплощение в первую очередь в *закрытых* (локальных) *гипертекстовых ИПС.* В качестве примера развитых в этом смысле гипертекстовых систем можно привести информационно-справочные системы помощи в среде ОС MS Windows.

***Модель организации данных в гипертекстовых справочных системах*** Microsoft Windows основана на *сочетании дифференциации ассоциативных гиперссылок и иерархического принципа организации фрагментов и документов.* Схематично модель организации данных можно отобразить схемой, представленной на рис. 6.15.

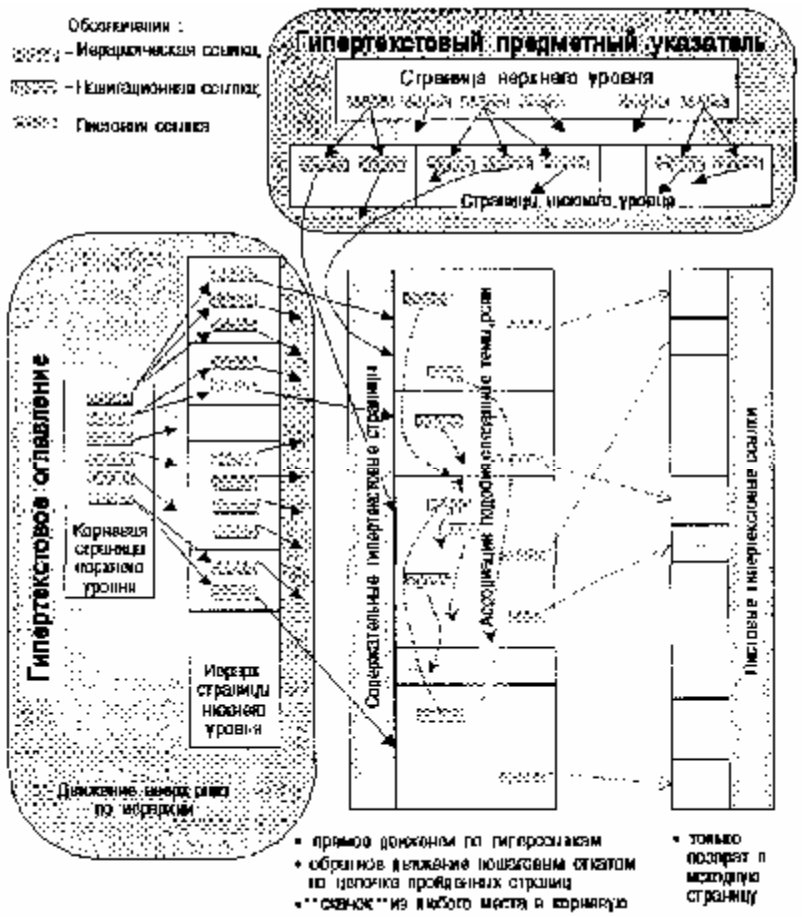



*Рис. 6.15. Модель организации данных в гипертекстовых справочных системах Microsoft Windows*

Как видно из представленной схемы, данная модель сочетает апробированные и интуитивно понятные большинству пользователей по аналогии работы с книгой иерархическую навигационную структуру (гипертекстовые оглавление и предметный указатель) с дифференцированными ассоциативными гиперссылками, выражающими рассмотренные выше различные типы ассоциаций при изучении и восприятии текстовой информации.

Вместе с тем использование справочных гипертекстовых систем все же не может полноценно заменить традиционные книги и учебники, так как большинство таких систем не обеспечивает привычный пользователю по обычным книгам последовательный повествовательный поток, разрывая его по пространственной или предметной иерархии, и, кроме того, требуют от пользователя новых навыков работы с текстовой информацией и более точного осознания в любой момент своих информационных потребностей.

Как и в моделях организации фактографических данных, в модели организации гипертекстовых данных важное значение имеет ***целостная составляющая.*** Применительно к гипертекстовым данным целостность и согласованность данных означает, прежде всего, ***целостность ссылок*** и выражается следующим принципом — ***«для каждой гиперссылки должен существовать адресат».*** Иначе говоря, целостность гипертекстовых данных выражается в отсутствии оборванных, ведущих в «никуда» связей.

*Контроль целостности ссылок* возможен на основе создания и ведения единого централизованного *реестра гиперссылок,* как это и осуществляется в *замкнутых гипертекстовых базах.* Специальный

компонент программного обеспечения гипертекстовой СУБД при удалениях документов (страниц) по реестру гиперссылок находит имеющиеся в других документах ссылки на удаляемый документ и аннулирует их.

В *открытых распределенных гипертекстовых системах* реализация принципа целостности ссылок встречает существенные трудности, так как децентрализованный принцип функционирования таких систем затрудняет создание и ведение единого реестра гиперссылок. В случае распределенной гипертекстовой среды за информацию на любом узле отвечает отдельный независимый пользователь, вольный по своему усмотрению добавлять или удалять гипертекстовые страницы (документы). Ввиду отсутствия централизованного реестра и однонаправленного* характера гиперссылок, при удалении какой-либо гипертекстовой страницы пользователь не может знать, имеются ли в других документах гиперссылки на удаляемую страницу. В таких ситуациях ***гиперссылки*** из других страниц, отсылающие на удаляемые страницы, оказываются ***оборванными.***

* То есть гиперссылка находится в источнике отсылки, а на отсылаемом адресате никакой информации по гиперссылке нет.

Еще более сложной проблемой является обеспечение ***согласованности данных.*** Применительно к гипертекстовым системам согласованность данных заключается ***в поддержании адекватности семантики гиперссылок.*** Говоря иначе, должна обеспечиваться *устойчивость смысловых ассоциаций по гиперссылкам.* Однако если изменить содержание того документа, на который отсылает гиперссылка из другого документа, то смысловая ассоциация, закладываемая в гиперссылку, может нарушиться, и в отсылаемом документе речь может пойти на совершенно другую тему.

Тривиальное решение проблемы согласованности гипертекстовых данных заключается в *запрете изменения содержания документов,* после внесения их в гипертекстовую базу. Такой подход применяется в некоторых системах на основе *замкнутых* гипертекстовых баз документов.

В открытых системах с децентрализованным характером функционирования такой подход неприемлем. Вместе с тем одним из возможных направлений решения этой проблемы является практикуемая в среде WWW ***идеология «публикаций».*** Среда WWW в этом смысле трактуется как гигантское электронное апериодическое издание, на страницах которого каждый желающий может «опубликовать» свои документы. Проблема согласованности данных по гиперссылкам может решаться в такой идеологии через введение в гиперссылки темпоральных параметров существования и соответствующих временных ограничений на содержательную изменчивость гипертекстовых публикаций. Иначе говоря, могут *быть определены «времена жизни» гиперссылок, в течение которых гипертекстовые публикации не могут быть изменены.* Однако, как и в случае введения двунаправленного характера гиперссылок, такой подход потребует перестройки протоколов и других соглашений в гигантской распределенной информационной инфраструктуре.

### 6.4.4. Формирование связей документов в гипертекстовых ИПС

Еще одним важным элементом в структуре гипертекстовых ИПС является подсистема формирования связей документов (см. рис. 6.12). Как и в случае систем на основе индексирования документов, существует *два подхода* к формированию связей документов в гипертекстовых ИПС —***ручной*** **и** ***автоматизированный.***

В *первом подходе* смысловые связи содержания документа с другими документами системы определяются самим *пользователем* (автором документа, администратором и т. п.). Такой подход имеет свои преимущества, так как пользователь устанавливает смысловые ассоциации нового документа с другими документами базы на *основе многоаспектного многокритериального анализа содержания документа,* что не может быть в полной мере воспроизведено никакими автоматизированными формальными или эвристическими алгоритмами.

Вместе с тем *у ручного подхода* имеется и ряд *существенных недостатков.* Человеческие возможности по скорости и объему смыслового анализа текстовых документов ограничены и не могут во многих случаях обеспечить приемлемые временные или организационные расходы на обработку и установление связей при больших потоках поступления документов в систему. В качестве примера можно привести гипертекстовую систему, агрегирующую в реальном масштабе времени поток новостных сообщений информационных агентств и другие тому подобные ситуации.

Однако даже если временных или иных ограничений на ввод документов в гипертекстовую ИПС нет, то другой проблемой является ограниченность человеческой памяти пользователя (администратора) по

содержанию введенных ранее в систему документов. Иначе говоря, пользователь, устанавливая гипертекстовые ассоциации нового документа, помимо смыслового содержания вводимого документа, одновременно должен представлять и помнить смысловое содержание всех других ранее введенных в систему документов, что, конечно же, без дополнительных классификационных или иных приемов в большинстве случаев нереально.

Кроме того, ручной подход, как и в случае индексирования документов, требует определенной квалификации пользователя-анализатора в соответствующей предметной области ИПС, что приводит к дополнительным проблемам.

Тем не менее в некоторых областях ручной способ установления гиперссылок сохраняет свое значение или является единственно возможным. Это, прежде всего, касается среды WWW в сети Интернет. Гипертекстовые ссылки публикуемых на Web-узлах документов на другие документы Сети пользователи определяют сами,* исходя из собственных представлений об ассоциации своей страницы с другими публикациями и узлами WWW. Вместе с тем такой подход не может по-настоящему полно и адекватно ассоциировать содержание публикуемой страницы с ресурсами Сети, так как ни один пользователь или Web-мастер, конечно же, не может знать и представлять всех ресурсов Сети. Отчасти эта проблема решается через так называемые поисковые машины, размещающиеся на известных всем пользователям узлах WWW и представляющие собой, как правило, сочетание информационно-поисковых классификационных каталогов и полнотекстовых ИПС, индексирующих все публикации в WWW. В этом случае гипертекстовые ассоциативные цепочки образуются через отсылку на узел поисковой машины, а от него к релевантным документам, располагающимся на других узлах сети.

---

* Совместно с так называемыми Web-мастерами и Web-дизайнерами.

***Автоматизированный подход к формированию и установлению гипертекстовых связей*** применяется в развитых *замкнутых* гипертекстовых ИПС. В основе автоматизации формирования гиперссылок лежит использование принципов ***поиска релевантных по смыслу документов,*** применяемых в системах на основе индексирования.

На практике применяются *две* основные *технологии* автоматизированного установления ассоциативных гипертекстовых связей:

• технология поисковых образов документов на основе техники ключевых слов (терминов);

• технология полнотекстового индексирования и поиска.

Использование ***технологии ключевых слов*** имеет несколько разновидностей. Один из вариантов предусматривает предварительное создание для предметной области гипертекстовой ИПС *взвешенного словаря ключевых терминов.* При вводе нового документа в системе производится его индексирование по словарю ключевых терминов и формируется ПОД. В простейшем случае в качестве ПОД используется суммарный вес терминов, присутствующих в тексте документа. Далее поисковый образ нового документа сравнивается с поисковыми образами ранее введенных документов и при превышении определенного порога «сходства» устанавливаются гипертекстовые связи с соответствующими документами.

В другом варианте используется предварительно созданная *классификационная рубрикация предметной области.* С каждой рубрикой связывается опять-таки предварительно созданный набор ключевых терминов или их сочетаний. На основе входного индексирования *производится соотнесение вводимого документа с той или иной рубрикой и на этой основе устанавливаются гипертекстовые связи* с соответствующей группой документов.

***Полнотекстовые технологии*** по сути аналогичны технике ключевых слов с учетом только более широкого текстового базиса индексирования и использования тех или иных критериев установления близости поисковых образов документов. В некоторых системах практикуются полуавтоматизированные технологии на основе полнотекстового поиска. В таких системах пользователь-анализатор выделяет из текста документа наиболее характерные по его содержанию фрагменты, которые используются в качестве запроса-образца для сформирования ПОЗ и полнотекстового поиска релевантных документов, с которыми и устанавливаются гипертекстовые связи.

Иногда применяются и более тонкие полуавтоматизированные подходы. Пользователь, анализируя содержание вводимого документа, может через технику ключевых терминов, или через классификационную рубрикацию, или через возможности полнотекстового поиска выбрать группу

предварительно сходных (ассоциированных) по смыслу документов. Далее просматривая документы этой группы, он отмечает действительно релевантные из них, определяя и устанавливая тем самым соответствующие связи (гиперссылки) вводимого документа.

Таким образом, в ***технологиях автоматизированного формирования гипертекстовых связей*** документов сливаются *все подходы, наработанные в сфере документальных информационных систем для формализации смыслового содержания текстовых документов.*

**Вопросы и упражнения**

1. Приведите основные отличия фактографических и документальных информационных систем по форме представления данных и способам удовлетворения информационных потребностей пользователей.
2. В чем отличие понятий пертинентность и релевантность?
3. Поясните, что отражают поисковые образы документов и запросов.
4. Что является результатом индексирования документов в документальных ИПС?
5. Объясните основные различия дескрипторных и семантических ИПЯ.
6. Дайте определения показателей эффективности документальных ИПС и охарактеризуйте соотношение между полнотой н точностью документального информационного поиска.
7. Объясните суть и дайте сравнительную характеристику перечислительной и систематизированной классификации документов.
8. По фрагменту фасетной классификации, приведенной на рис. 6.6, постройте индекс документа, в котором идет речь о коррозионной усталости трубчатых растяжных конструкций из хромоникелевых сплавов.
9. По фрагменту фасетпой классификации, приведенной на рис. 6.6, поясните, о чем может идти речь в документах, индекс которых выражается следующими фасетными формулами:

   Ac Bc Kg Lg
   Ac Bcd Kgb Lm
   Ac Bgt Ki Lg

10. Для документальной базы текстовых сообщений о планируемых или состоявшихся научных мероприятиях (конференции, семинары, симпозиумы) по тематикам, связанным с радиолокацией (радионавигация, загоризонтная локация, синтез и обработка радиолокационных сигналов, антенно-фидерная техника, устройства отображения радиолокационной иифорлации и т. д.), организуемых академическими, отраслевыми НИИ или учебными заведениями, составьте примерную схему фасетиой классификации и на ее основе проиндексируйте следующее сообщение:

21.08.2000г. Институтом математики и кибернетики Северосибирского отделения РАН в г. Северск проводится научно-практическая конференция «Теория и практика (фазокодоманипулированных (ФКМ) сигналов радиолокационных устройств».

11. Для чего применяется координация понятий (терминов) в документальных ИПС на основе индексирования? В чем суть и отличия понятии пред- и посткоординации?
12. В рамках фасетной классификации, приведенной на рис. 6.6, поясните следующие фасетные формулы с использованием координации понятий, отражающие содержание документов или запросов пользователя:

$$(\neg AC) \cap BGT \cap ((KI \cup KL) \cap KP) \cap LG$$
$$AC \cap BC \cap (KP \cap (\neg KPF)) \cap (LB \cup LD \cup LG)$$

13. В рамках фасетной классификации, приведенной на рис. 6.6, составьте фасетную (формулу с использованием предкоординации понятий для документа, в котором идет речь о разрушениях иетрубчатых конструкций из хромистых или хромоникелевых сталей от коррозионной усталости при сжатиях или ударах.
14. Какие отличия имеет тезаурус от словаря ключевых слов определенной предметной области?
15. В чем заключается ведение тезауруса?
16. Почему ИПС па основе тезаурусов обладают более высокой эффективностью документального информационного поиска по сравнению с системами, построенными па основе перечислительной классификации содержания документов по словарю ключевых слов?
17. Приведите сравнительную характеристику преимуществ и недостатков ручного и автоматизированного индексирования документов.
18. Что представляет собой индекс в документальных ИПС, какова его организация и смысл элементов?

19. В каком смысле индекс полнотекстовых систем отражает полный текст документов, агрегированных в ИПС?
20. По информационно-технологической структуре полнотекстовых ИПС, приведенной на рис. 6.9, выделите элементы, которые в совокупности составляют программные средства, характеризуемые термином «полнотекстовые СУБД».
21. По информационно-технологической структуре полнотекстовых ИПС, приведенной на рис. 6.9, выделите элементы, которые в совокупности можно назвать полнотекстовой базой документов.
22. Поясните, является ли обязательным наличие отдельно организованного и локально сосредоточенного хранилища документов в полнотекстовых ИПС.
23. Что происходит с индексом и, в частности, с поисковыми образами ранее накопленных (проиндексированных) документов в полнотекстовых ИПС с динамическим глобальным словарем при наличии во вновь поступающем на учет в систему документе новых словоформ?
24. В чем существо нормализации глобального словаря в полнотекстовых ИПС и что дает нормализация?
25. Используется ли координация понятий в полнотекстовых ИПС? Если да, то в какой форме?
26. С использованием логических операторов «И»(З), «ИЛИ»(Н), «НЕ»(П) постройте запросы на дескрипторном ИПЯ, в котором дескрипторами могут выступать любые слова (словоформы), по поиску документов, в которых идет речь о:
    - методике информационного поиска в фактографических или документальных ИПС, исключая семантически-навигационные системы;
    - моделях разграничения доступа в компьютерных системах, построенных в идеологии «Клиент-сервер», исключая системы с серверами приложений;
    - оптимизаторах запросов в реляционных СУБД, которые не используют апостериорный анализ статистики данных.

27. В чем преимущества и, может быть, недостатки полнотекстовых ИПС с координатным анализом текста документа при индексировании?
28. Объясните феномен практической независимости скорости поиска документов в полнотекстовых ИПС от объема хранилища документов.
29. Что и как в действительности «понимает» полнотекстовая ИПС с морфологическим разбором и нормализованным глобальным словарем в запросах пользователя, формулируемых на естественном языке?
30. Как определяется количественная мера документов при пространственно-векторной модели представления ПОД и ПОЗ?
31. Какие вероятностно-статистические расширения пространственно-векторной модели ПОД и ПОЗ позволяют повысить релевантность поиска документов в полнотекстовых ИПС? Что означает термин «окрашивание» глобального словаря?
32. Чем гипертекст отличается от просто текста?
33. В чем сходство и отличие чтения и восприятия традиционных текстовых (бумажных) документов и работы пользователя в гипертекстовой ИПС?
34. Для чего нужна визуализация цепочек «пройденных» документов в гипертекстовых ИПС, каким способом обычно она решается и какие проблемы при этом возникают?
35. Какие математические объекты описывают модель организации данных в гипертекстовых ИПС?
36. Какие проблемы восприятия гипертекста может решить дифференциация типов гиперссылок? Ответ поясните на примере модели организации данных в гипертекстовых справочных системах Microsoft Windows.
37. В чем заключаются ограничения целостности в гипертекстовой модели данных и какие проблемы в их контроле и поддержании могут возникать в распределенных открытых гипертекстовых ИПС (на примере WWW)?

# 7. АДМИНИСТРИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ И ЗАЩИТА ДАННЫХ

Использование автоматизированных информационных систем порождает общие информационные ресурсы в виде базы или совокупности баз данных, состояние и функционирование которых может критически влиять на жизнедеятельность предприятия, организации. В результате, как и для любого критического ресурса, требуются отдельное специфическое управление и контроль состояния, получившие в процессе внедрения АИС в практику информационного обеспечения деятельности предприятий и организаций специальный термин — «администрирование и защита данных (баз данных)».

В номенклатуре специалистов, обеспечивающих проектирование, создание, эксплуатацию и использование АИС, соответственно, выделилась отдельная категория, называемая администраторами систем (баз данных), играющих ключевую роль в процессах информационного обеспечения деятельности предприятий и организаций.

Взгляды на функции и содержание задач, решаемых в процессе администрирования баз данных, формировались вместе со становлением индустрии АИС, менялись вместе с изменениями и усложнениями программно-технических аспектов реализации АИС, но со временем постепенно сформировался некоторый их базовый перечень, и администрирование баз данных вошло неотъемлемым ключевым компонентом в теорию и практику автоматизированных информационных систем.

## 7.1. Администрирование информационных систем

В общем плане термин «администрирование» определяет комплекс процессов при создании, эксплуатации и использовании АИС, связанных с обеспечением надежности и эффективности функционирования АИС, безопасности данных и организацией коллективной работы пользователей различных категорий.

Этот комплекс процессов можно разделить по решаемым задачам на следующие группы:

- обеспечение и поддержание настройки структурного, интерфейсного и технологического компонентов АИС на структуру и процессы предметной области системы;
- обеспечение надежности и сохранности данных;
- организация и обеспечение коллективной работы пользователей с общими данными.

Первое направление обусловливает участие администратора системы в этапах проектирования и ввода АИС в эксплуатацию. Администратор при этом выступает экспертом в команде разработчиков по выбору СУБД и ее особенностям в плане реализации тех или иных компонент концептуальной схемы создаваемого банка данных, участвует в процессах создания типовых запросов, экранных форм для ввода и вывода данных, шаблонов отчетов. На этапе проектирования и, в особенности, в процессе дальнейшей эксплуатации при наполнении системы данными администратор системы производит анализ адекватности и эффективности спроектированной внутренней схемы базы данных и при необходимости может осуществлять ее корректировку через использование, например, техники нормализации таблиц в реляционных СУБД. Для обеспечения таких функций некоторые СУБД содержат специальный программный инструментарий в виде анализаторов быстродействия и оптимальности баз данных.

В эту же группу функций входит создание и поддержание словарно-классификационной базы (словари, справочники, ключевые слова, тезаурусы), которая должна адекватно отражать особенности предметной области информационной системы.

Еще одной важной функцией, особенно на этапе ввода информационной системы в эксплуатацию, является первоначальное наполнение системы данными. Во многих случаях для этого могут быть использованы данные из других информационных систем, находящихся, возможно, в других физических форматах и с другой логической организацией. Импорт данных из внешних источников, как правило, требует высокой квалификации по возможностям используемой в АИС СУБД, детального представления логической и физической организации данных в АИС и выполняется вследствие этого администратором системы. В фактографических АИС на основе реляционных СУБД подобные задачи решаются на основе запросов на создание таблиц, источник данных для которых находится во внешних базах или в файлах других форматов (электронные таблицы, текстовые файлы в формате УФОД). Соответственно, особенности функционирования АИС могут требовать решения и обратных задач, т. е. задач экспорта данных во внешние системы и другие форматы. Такие задачи в большинстве случаев также решаются администраторами АИС.

Обеспечение надежности и сохранности данных является одной из главных обязанностей администратора АИС и включает, в свою очередь, решение ряда следующих технологических и профилактических задач:

- планирование, конфигурирование и поддержание системы использования устройств внешней памяти, на которых размещаются файлы данных;
- архивирование и резервирование данных;
- восстановление данных после сбоев и повреждений;
- проверка и поддержание целостности данных.

Большой объем файлов баз данных, как уже отмечалось, обусловливает их размещение на устройствах дисковой (внешней) памяти. Поэтому отдельной задачей при проектировании АИС является *определение схемы размещения файлов базы данных системы на устройствах внешней памяти, контроль за ее состоянием в процессе эксплуатации АИС.*

В ряде приложений сохранность и работоспособность базы данных является чрезвычайно критическим аспектом либо в силу технологических особенностей (системы реального времени), либо в силу содержательного характера данных. В этих случаях применяются подходы так называемого *горячего резервирования* данных, когда база данных постоянно находится в виде двух идентичных (зеркальных) и параллельно функционирующих копий, размещаемых на двух раздельных системах дисковой памяти.

В других ситуациях для обеспечения сохранности данных используются операции *архивирования и резервирования данных.* В большинстве случаев архивирование производится обычными средствами архивации файлов для их компактного долговременного хранения, как правило, на внешних съемных носителях. Функции архивирования данных иногда могут входить и в перечень внутренних функций самих СУБД.

Резервирование данных, как правило, не предусматривает специального сжатия данных, а производится через создание специальных копий файлов данных в технологических или иных целях.

И архивирование, и резервирование, помимо технологических целей, преследуют также профилактические цели по еще одной чрезвычайно важной операции, выполняемой администратором АИС — ***восстановлению*** *данных после сбоев и повреждений.* Наличие резервной или архивной копии базы данных позволяет восстановить работоспособность системы при выходе из строя основного файла (файлов) данных. При этом, однако, часть данных, или их изменений, произведенные за время, прошедшее с момента последнего архивирования или резервирования, могут быть потеряны. Такие ситуации особенно критичны при коллективной обработке общих данных, реализуемых клиент-серверными системами. Поэтому в промышленных СУБД, реализующих технологии «Клиент-сервер», в большинстве случаев предусматривается ведение специального журнала текущих изменений базы данных, размещаемого отдельно от основных данных и, как правило, на отдельном носителе. Как уже отмечалось, такой подход называется журнализацией. В журнале изменений осуществляются непрерывная фиксация и протоколирование всех манипуляций пользователей с базой данных. В результате при любом сбое с помощью архивной копии и журнала изменений администратор системы может полностью восстановить данные до момента сбоя.

В перечень функций администратора по обеспечению восстановления данных входит также *профилактика дисковых носителей внешней памяти,* обеспечиваемая специальными программными инструментальными утилитами операционной системы — проверка состояния дисков, дефрагментация и т. д. Данные обязанности накладывают дополнительные требования к профессиональной подготовке администраторов как особого направления подготовки системных программистов.

***Проверка и поддержание целостности данных*** является также неотъемлемой функцией администраторов и заключается в обеспечении настройки и функционирования защитных механизмов СУБД, поддерживающих ограничения целостности данных и связей в конкретной базе данных. Как правило, в большинстве СУБД имеются встроенные механизмы автоматического поддержания и контроля целостности данных. Вместе с тем, в ряде случаев, логика предметной области не даст возможности устанавливать такие режимы обеспечения целостности связей, как, например, каскадное обновление и удаление связанных записей. При этом в процессе работы пользователей с базой данных может образовываться множество коллизий в виде «оборванных» связей, что снижает эффективность функционирования базы данных. В таких ситуациях одной из функций администратора базы данных являются периодический контроль целостности связей и устранение подобных коллизий. В некоторых СУБД для решения этих задач предусмотрены специальные режимы и механизмы «ревизии» и восстановления целостности базы данных. В других СУБД данные задачи решаются через технику запросов или через разработку, в том числе и самими администраторами АИС специальных программных утилит.

Сходные задачи ревизии данных решаются администратором также в тех случаях, когда устанавливаются временные регламенты хранения данных и «устаревшие» данные должны своевременно обновляться или удаляться из системы.

Большой комплекс функций администратора АИС связан с *организацией и обеспечением коллективной работы пользователей с общими данными.* Еще на этапе проектирования АИС с непосредственным участием будущего администратора системы разрабатывается организационная схема функционирования и использования АИС. Исходя из особенностей технологических процессов в предметной области и круга решаемых задач, определяются функциональные группы работников, отвечающих за ввод, обработку и использование общих данных системы. На этой основе строится перечень и схема пользователей системы, определяются их конкретные функции, полномочия, разрабатываются необходимые технологические и интерфейсные элементы (входные и выходные экранные формы, запросы, шаблоны отчетов и т. д.), прорабатываются и устанавливаются внутренние параметры и характеристики коллективной обработки данных (размещение данных, параметры блокировок, обмена и т. д.). Администратор АИС, по сути, является как раз организатором и руководителем этих технологических процессов организации работы эксплуатационного персонала и абонентов-пользователей системы.

Отдельным, но тесно переплетенным с остальными функциями, направлением этого комплекса обязанностей администратора системы является *создание и поддержание системы разграничения доступа к данным и защиты данных от несанкционированного доступа.* На основе системы и схемы функций, задач и полномочий пользователей и обслуживающего персонала администратором

строится и поддерживается схема категорирования объектов базы данных по критерию доступа различных пользователей и внешних процессов, осуществляется текущее управление этой схемой и аудит процессов обработки данных с точки зрения безопасности и разграничения доступа к данным.

Развитые СУБД в составе своих функций и возможностей, как правило, имеют специальный *инструментарий, обеспечивающий основной набор функций и задач администратора* — уже упоминавшийся анализатор быстродействия и оптимальности, утилиты архивирования, резервирования и ревизии базы данных, подсистему разграничения доступа и защиты данных.

В настольных однопользовательских системах, строящихся на основе СУБД с развитым интерфейсным набором инструментов по созданию и управлению базами данных, все или большая часть функций администратора может выполняться самими пользователями, которые в необходимых случаях могут лишь изредка прибегать к помощи и консультациям соответствующих специалистов. Вместе с тем рассмотренный перечень функций и решаемых задач, так или иначе, реализуется и в однопользовательских системах.

Следует также отметить, что организационно администраторы АИС являются отдельными штатными категориями работников информационных служб, подчиняясь непосредственно руководителям таких служб, или во многих случаях, собственно, и выполняя функции руководителя службы информационного обеспечения предприятия, организации, отдельного подразделения.

## 7.2. Разграничение доступа и защита данных

Одной из оборотных сторон компьютерных информационных технологий является обострение ***проблемы защиты информации.*** Данные в компьютерной форме сосредоточивают в физически локальном и небольшом объеме огромные массивы информации, несанкционированный доступ к которой или ее разрушение могут приводить порой к катастрофическим последствиям и ущербу. Возможность быстрого, во многих случаях практически мгновенного, и без следов копирования огромных массивов данных, находящихся в компьютерной форме, в том числе и удаленно расположенных, дополнительно ***провоцирует злоумышленников*** на несанкционированный доступ к информации, ее несанкционированную модификацию или разрушение.

Вместе с тем теоретическая проработка вопросов обеспечения безопасности информации и их практическая реализация долгое время отставали от уровня развития программной индустрии СУБД, и в коммерческих продуктах средства обеспечения безопасности данных стали появляться лишь в 90-х годах.

Импульсы развития и первые исследования теории и практики обеспечения безопасности данных в компьютерных системах были обусловлены, прежде всего, потребностями военной сферы, где проблема безопасности в целом, и компьютерной безопасности в частности стоят особенно остро. Начало этим процессам было положено исследованиями вопросов защиты компьютерной информации, проведенными в конце 70-х — начале 80-х годов национальным центром компьютерной безопасности (NCSC — National Computer Security Center) Министерства обороны США. Результатом этих исследований явилось издание Министерством обороны США в 1983 г. документа под названием ***«Критерии оценки надежных компьютерных систем»,*** впоследствии по цвету обложки получившего название ***«Оранжевой книги».*** Данный документ стал фактически первым стандартом в области создания защищенных компьютерных систем и впоследствии основой организации системы сертификации компьютерных систем по критериям защиты информации.

Подходы к построению и анализу защищенных систем, представленные в «Оранжевой книге», послужили методологической и методической базой для дальнейших исследований в этой сфере. В 1991 г. NCSC был издан новый документ — *Интерпретация «Критериев оценки надежных компьютерных систем» в применении к понятию надежной системы управления базой данных,* известный под сокращенным названием TDI или «Розовой книги», конкретизирующий и развивающий основные положения «Оранжевой книги» по вопросам создания и оценки защищенных СУБД.

В конце 80-х — начале 90-х годов аналогичные исследования по проблемам компьютерной безопасности были проведены во многих странах и созданы соответствующие национальные стандарты в этой сфере. В нашей стране *Государственной технической комиссией при Президенте РФ* были разработаны и в 1992 г. опубликованы ***«Руководящие документы по защите от несанкционированного доступа к информации»,*** определяющие требования, методику и стандарты построения защищенных средств вычислительной техники и автоматизированных систем.

### 7.2.1. Понятие и модели безопасности данных

Исследования по проблемам защиты компьютерной информации, проведенные в конце 70-х—начале 80-х годов, развитые впоследствии в различных приложениях и закрепленные в соответствующих стандартах, определяют в качестве составных элементов понятия *безопасности информации* три компонента:

- ***конфиденциальность*** (защита от несанкционированного доступа);
- ***целостность*** (защита от несанкционированного изменения информации);
- ***доступность*** (защита от несанкционированного удержания информации и ресурсов, защита от разрушения, защита работоспособности).

Составляющим безопасности информации противостоят соответствующие *угрозы.* Под ***угрозой безопасности информации*** понимается *осуществляемое или потенциально осуществимое воздействие на компьютерную систему, которое прямо или косвенно может нанести ущерб безопасности информации.* Угрозы реализуют или пытаются реализовать ***нарушители*** информационной безопасности.

Формализованное описание или представление *комплекса возможностей нарушителя по реализации тех или иных угроз безопасности информации* называют ***моделью нарушителя (злоумышленника).***

Качественное описание комплекса организационно-технологических и программно-технических мер по обеспечению защищенности информации в компьютерной системе (АИС) называют ***политикой безопасности.*** Формальное (математическое, алгоритмическое, схемотехническое) выражение и формулирование политики безопасности называют ***моделью безопасности.***

Модель безопасности включает:

- модель компьютерной (информационной) системы;
- критерии, принципы, ограничения и целевые функции защищенности информации от угроз;
- формализованные правила, ограничения, алгоритмы, схемы и механизмы безопасного функционирования системы.

В основе большинства моделей безопасности лежит ***субъектно-объектная модель*** *компьютерных систем,* в том числе и баз данных как ядра автоматизированных информационных систем. База данных АИС разделяется на ***субъекты базы данных*** (активные сущности), ***объекты базы данных*** (пассивные сущности) и порождаемые действиями субъектов ***процессы над объектами*** (см. рис. 7.1).

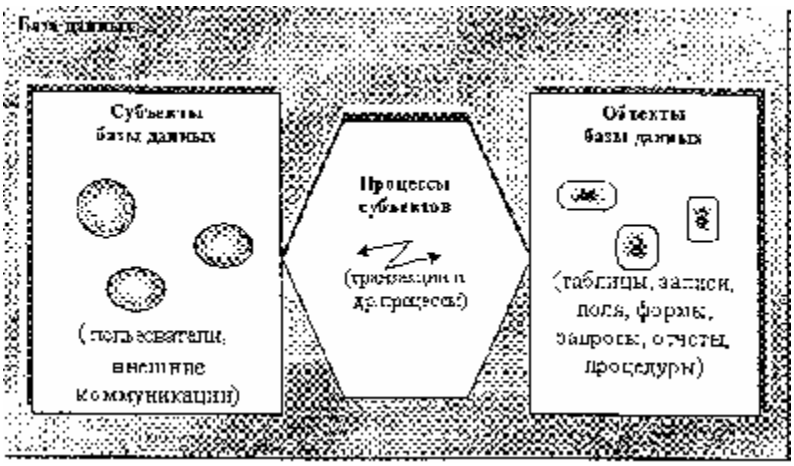



*Рис. 7.1. База данных АИС в моделях безопасности данных*

Определяются два основополагающих принципа безопасности функционирования информационных систем:

- ***персонализация*** *(идентификация) и* ***аутентификация*** (подтверждение подлинности) всех субъектов и их процессов по отношению к объектам;
- ***разграничение полномочий субъектов*** по отношению к объектам и обязательная проверка полномочий любых процессов над данными.

Соответственно в структуре ядра СУБД выделяется дополнительный компонент, называемый ***монитором*** (сервером, менеджером, ядром) ***безопасности*** (Trusted Computing Base— TCB), который реализует определенную ***политику безопасности*** во всех процессах обработки данных. Если в схемотехническом аспекте компьютерную систему представить как совокупность ядра, включающего компоненты представления данных и доступа (манипулирования) к данным, а также

надстройки, которая реализует интерфейсные и прикладные функции, то роль и место монитора безопасности можно проиллюстрировать схемой, приведенной на рис. 7.2.

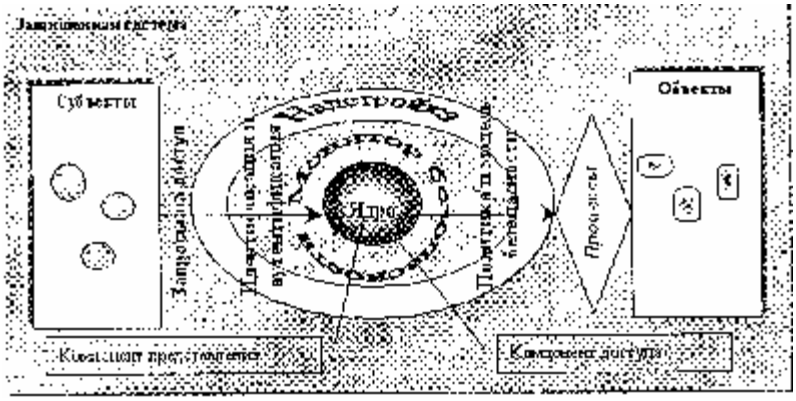



*Рис. 7.2. Схематический аспект защиты информации в компьютерных системах*

В узком смысле политика безопасности, реализуемая монитором безопасности компьютерной системы, собственно и определяет модель безопасности (вторая и третья компоненты).

Простейшая ***(одноуровневая)*** *модель безопасности* данных строится на основе ***дискреционного*** *(избирательного)* ***принципа разграничения доступа,*** при котором доступ к объектам осуществляется на основе ***множества разрешенных отношений доступа*** в виде троек — «субъект доступа – тип доступа – объект доступа». Наглядным и распространенным способом формализованного представления дискреционного доступа *является матрица доступа,* устанавливающая перечень пользователей (субъектов) и перечень разрешенных операций (процессов) по отношению к каждому объекту базы данных (таблицы, запросы, формы, отчеты). На рис. 7.3 приведен пример, иллюстрирующий матрицу доступа.

| | | Таблицы | | | | |
|---|---|---|---|---|---|---|
| | | Сотрудники: Установочные_данные | Сотрудники: Конфиденциальные_данные | Операции | Командировки | Задания |
| Пользователи | Иванов | Ч,М | | | | |
| | Петров | Ч | Ч | Ч,С,М | Ч,С,М | Ч,С,М |
| | Сидоров | Ч,М,С,У | Ч,М,С,У | | | |
| | Бессонцев | Ч,М,С,У | Ч,М,С,У | Ч,М,С,У | Ч,М,С,У | Ч,М,С,У |

Обозначения:
Ч – чтение;
М – модификация;
С – создание;
У – удаление (записей)

*Рис. 7.3. Модель безопасности на основе матрицы доступа (дискреционный принцип разграничения доступа)*

Важным аспектом моделей безопасности является *управление доступом.* Существует два подхода:

• добровольное управление доступом;

• принудительное управление доступом.

При ***добровольном управлении доступом*** вводится так называемое ***владение объектами.*** Как правило, *владельцами* объектов являются те субъекты базы данных, процессы которых *создали* соответствующие объекты. Добровольное управление доступом заключается в том, что *права на доступ к объектам определяют их владельцы*. Иначе говоря, соответствующие ячейки матрицы доступа заполняются теми субъектами (пользователями), которым принадлежат права владения над соответствующими объектами базы данных. В большинстве систем права владения объектами могут передаваться. В результате при добровольном управлении доступом реализуется полностью *децентрализованный принцип* организации и управления процессом разграничения доступа.

Такой подход обеспечивает *гибкость* настраивания системы разграничения доступа в базе данных на конкретную совокупность пользователей и ресурсов, но затрудняет общий контроль и аудит состояния безопасности данных в системе.

***Принудительный подход к управлению доступом*** предусматривает введение единого централизованного ***администрирования доступом.*** В базе данных выделяется специальный доверенный субъект (администратор), который (и только он), собственно, и определяет разрешения на доступ всех остальных субъектов к объектам базы данных. Иначе говоря, заполнять и изменять ячейки матрицы доступа может только администратор системы.

Принудительный способ обеспечивает *более жесткое* централизованное управление доступом. Вместе с тем он является менее гибким и менее точным в плане настройки системы разграничения доступа на потребности и полномочия пользователей, так как наиболее полное представление о содержимом и конфиденциальности объектов (ресурсов) имеют, соответственно, их владельцы.

На практике может применяться *комбинированный способ* управления доступом, когда определенная часть полномочий на доступ к объектам устанавливается администратором, а другая часть владельцами объектов.

Исследования различных подходов к обеспечению информационной безопасности в традиционных (некомпьютерных) сферах и технологиях показали, что одноуровневой модели безопасности данных недостаточно для адекватного отражения реальных производственных и организационных схем. В частности традиционные подходы используют ***категорирование информационных ресурсов по уровню конфиденциальности*** (совершенно секретно — СС, секретно — С, конфиденциально — К, и т. п.). Соответственно субъекты доступа к ним (сотрудники) также категорируются по соответствующим уровням доверия, получая так называемого ***допуска*** (допуск степени 1, допуск степени 2 и т. д.). Понятие допуска определяет ***мандатный (полномочный) принцип разграничения доступа к информации.*** В соответствии с мандатным принципом работник, обладающий допуском степени «1», имеет право работать *с любой* информацией уровня «СС», «С» и «К». Работник с допуском «2» соответственно имеет право работы *с любой* информацией уровня «С» и «К». Работник с допуском «3» имеет право работать *с любой* информацией только уровня «К».

Мандатный принцип построения системы разграничения доступа в СУБД реализует ***многоуровневую модель безопасности данных,*** называемую еще моделью Белл — ЛаПадула (по имени ее авторов — американских специалистов Д. Белла и Л. ЛаПадула), которая иллюстрируется схемой, приведенной на рис. 7.4.

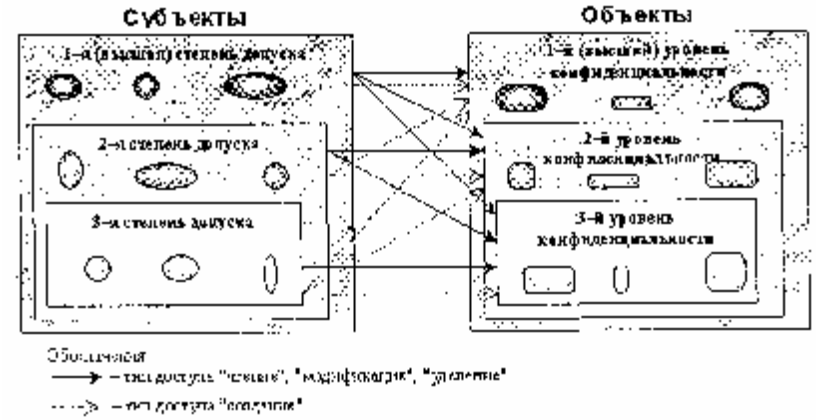



*Рис. 7.4. Модель безопасности данных Белл — ЛаПадула (мандатный принцип разграничения доступа)*

В модели Белл — ЛаПадула объекты и субъекты категорируются по *иерархическому мандатному принципу доступы.* Субъект, имеющий допуск 1-й (высшей) степени, получает доступ к объектам 1-го (высшего) уровня конфиденциальности и автоматически ко всем объектам более низких уровней конфиденциальности (т. е. к объектам 2-го и 3-го уровней). Соответственно, субъект со 2-й степенью допуска имеет доступ ко всем объектам 2-го и 3-го уровней конфиденциальности, и т. д.

В модели Белл —ЛаПадула устанавливаются и поддерживаются два основных *ограничения* политики безопасности:

• запрет чтения вверх (no read up — NRU);
• запрет записи вниз (no write down — NWD).

Ограничение NRU является логическим следствием мандатного принципа разграничения доступа, запрещая субъектам читать данные из объектов более высокой степени конфиденциальности, чем позволяет их допуск.

Ограничение NWD предотвращает перенос (утечку) конфиденциальной информации путем ее копирования из объектов с высоким уровнем конфиденциальности в неконфиденциальные объекты или в объекты с меньшим уровнем конфиденциальности.

Ограничения NRU и NWD приводят к тому, что по разным типам доступа («чтение», «создание», удаление», «запись») в модели Белл—ЛаПадула устанавливается разный порядок доступа конкретного субъекта к объектам. В частности, в схеме, приведенной на рис. 7.4, может показаться странным, что по типу доступа «создание» субъект (процесс) с допуском степени 3 (низшей) имеет возможность создавать объекты (записи) в объектах более высокого уровня конфиденциальности.* Такой подход, тем не менее, отражает реальные жизненные ситуации, когда работники, к примеру, кадрового подразделения могут заполнять формализованные карточки на новых сотрудников, направляя их в специальную картотеку личных данных сотрудников организации и порождая первые документы личных дел новых сотрудников, но не имеют при этом собственно самого доступа к этой картотеке по другим типам операций (чтение, удаление, изменение).

* В литературе подобные возможности доступа в многоуровневых моделях обозначаются *-свойством

Мандатный принцип разграничения доступа, опять-таки исходя из дополнительных способов разграничения доступа к конфиденциальной информации, наработанных в «бумажных» технологиях, в частности в военной сфере, может дополняться элементами ***функционально-зонального принципа разграничения доступа.*** В соответствии с функционально-зональным принципом, защищаемые сведения, помимо категории конфиденциальности, получают *признак функциональной тематики* (зоны), например сведения по артиллерии, сведения по авиации, и т. д. Соответственно, каждый работник, имеющий определенный допуск к конфиденциальным сведениям, еще по своим функциональным обязанностям имеет *определенный профиль деятельности,* который предоставляет допуск или *уточняет сферу допуска к категорированным сведениям только соответствующей тематики.* Такой подход предоставляет более гибкие и точные возможности организации работы с конфиденциальными сведениями и в СУБД реализуется через технику *профилей и ролей* пользователей.

На практике в реальных политиках мониторов безопасности баз данных чаще всего применяется дискреционный принцип с принудительным управлением доступом, «усиливаемый» элементами мандатного принципа в *сочетании* с добровольным управлением доступом (допуска субъектов устанавливает и изменяет только администратор, уровень конфиденциальности объектов устанавливают и изменяют только владельцы). В распределенных СУБД могут также применяться элементы функционально-зонального разграничения доступа в виде жесткой привязки объектов и субъектов к определенным устройствам, а также выделении специальных зон, областей со «своей» политикой безопасности.

При реализации политик и моделей безопасности данных в фактографических АИС на основе реляционных СУБД возникают специфические *проблемы разграничения доступа на уровне отдельных полей таблиц.* Эти проблемы связаны с отсутствием в реляционной модели типов полей с множественным (многозначным) характером данных.*

* Требование первой нормальной формы

Поясним суть подобных проблем следующим образом. Предположим, в базе данных имеется таблица с записями по сотрудникам с полями «Лич_№», «Фамилия», «Подразделение», «Должность». Предположим также, что в структуре организации имеется засекреченное подразделение «Отдел_0», известное для всех непосвященных под названием «Группа консультантов». Соответственно поля «Подразделение», «Должность» для записей сотрудников этого подразделения будут иметь двойные значения — одно истинное (секретное), другое для прикрытия (легенда). Записи таблицы «Сотрудники» в этом случае могут иметь вид:

*Таблица 7.1*

| *Лич_№* | *Фамилия* | *Подразделение* | *Должность* |
|---|---|---|---|
| 001 | Иванов | Руководство | Начальник |
| 002 | Петров | Отдел снабжения | Снабженец |
| 003 | Исаев | Отдел_0 [С] | Разведчик [С] |
| | | Группа консультантов [НС] | Консультант [НС] |
| 004 | Сидоров | Бухгалтерия | Бухгалтер |
| 007 | Бонд | Отдел_0 [С] | Агент [С] |
| | | Группа консультантов [НС] | Советник [НС] |

В таблице знаком [С] обозначены секретные значения, знаком [НС] несекретные значения соответствующих полей. Как видно из приведенной таблицы, требования первой нормальной формы в таких случаях нарушаются.

Более того, даже если взять более простой случай, когда конфиденциальные данные по некоторым полям пользователям, не имеющим соответствующей степени допуска, просто не показываются, т.е. такой пользователь «видит» в секретных полях некоторых записей просто пустые значения, то тем не менее возникают ***косвенные каналы нарушения конфиденциальности.*** Так как на самом деле в соответствующем поле данные имеются, то при попытке записи в эти поля неуполномоченный на это пользователь получит отказ, и, тем самым, «почувствует» что-то неладное, подозрительное, скажем, в отношении записи сотрудника с личным номером 007. Это и есть косвенный канал.

В более общем виде под *косвенным каналом нарушения конфиденциальности* подразумевается *механизм, посредством которого субъект, имеющий высокий уровень благонадежности, может предоставить определенные аспекты конфиденциальной информации субъектам, степень допуска которых ниже уровня конфиденциальности этой информации.*

В определенной степени проблему многозначности и косвенных каналов можно решать через нормализацию соответствующих таблиц, разбивая их на связанные таблицы, уровень конфиденциальности которых, или конфиденциальности части записей которых, будет различным.

Другие подходы основываются на расширении реляционной модели в сторону многозначности и допущения существования в таблицах кортежей с одинаковыми значениями ключевых полей.

### 7.2.2. Технологические аспекты защиты информации

Практическая реализация политик и моделей безопасности, а также аксиоматических принципов построения и функционирования защищенных информационных систем обусловливает необходимость решения ряда программно-технологических задач, которые можно сгруппировать по следующим направлениям:

- технологии идентификации и аутентификации;
- языки безопасности баз данных;
- технологии обеспечения безопасности повторного использования объектов;
- технологии надежного проектирования и администрирования.

#### 7.2.2.1. Идентификация и аутентификация

Технологии идентификации и аутентификации являются обязательным элементом защищенных систем, так как обеспечивают аксиоматический принцип персонализации субъектов и, тем самым, реализуют *первый (исходный) программно-технический рубеж защиты информации* в компьютерных системах.

Под ***идентификацией*** понимается различение субъектов, объектов, процессов по их образам, выражаемым именами.

Под *аутентификацией* понимается проверка и подтверждение подлинности образа идентифицированного субъекта, объекта, процесса.

Как вытекает из самой сути данных понятий, в основе технологий идентификации и аутентификации лежит идеология статистического распознавания образов, обусловливая, соответственно, принципиальное наличие *ошибок первого (неправильное распознавание)* и *второго (неправильное нераспознавание)* рода. Методы, алгоритмы и технологии распознавания направлены на обеспечение задаваемых вероятностей этих ошибок при определенных стоимостных или иных затратах.

В системотехническом плане структуру систем идентификации/аутентификации можно проиллюстрировать схемой, приведенной на рис. 7.5.

При регистрации объекта идентификации/аутентификации в системе монитором безопасности формируется его образ, информация по которому подвергается необратимому без знания алгоритма и шифра-ключа, т. е. криптографическому, преобразованию и сохраняется в виде ресурса, доступного в системе исключительно монитору безопасности. Таким образом формируется информационный массив внутренних образов объектов идентификации/аутентификации.

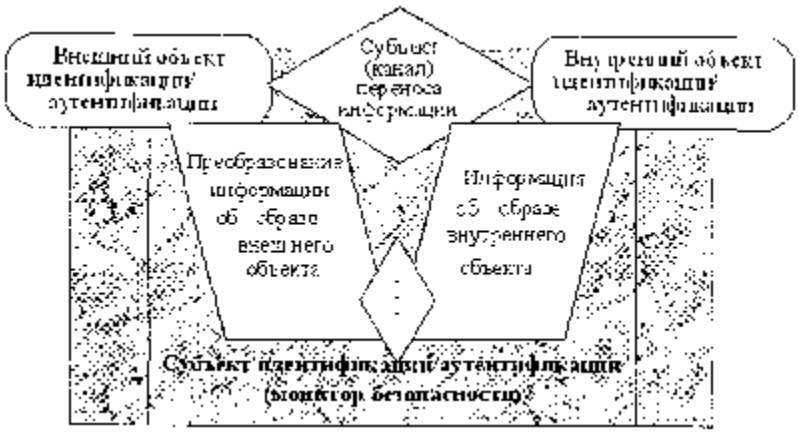



*Рис. 7.5. Системотехнический аспект идентификации/аутентификации*

Впоследствии при идентификации/аутентификации (очередной вход в систему пользователя, запрос процесса на доступ к объекту, проверка подлинности объекта системы при выполнении над ним действий и т. д.) объект через канал переноса информации передает монитору безопасности информацию о своем образе, которая подвергается соответствующему преобразованию. Результат этого преобразования сравнивается с соответствующим зарегистрированным внутренним образом, и при их совпадении принимается решение о распознавании (идентификации) и подлинности (аутентификации) объекта.

*Информационный массив внутренних образов* объектов идентификации/аутентификации является критическим ресурсом системы, несанкционированный доступ к которому дискредитирует всю систему безопасности. Поэтому помимо всевозможных мер по исключению угроз несанкционированного доступа к нему сама информация о внутренних образах объектов идентификации/аутентификации находится в зашифрованном виде.

Формирование образов осуществляется на разной методологической и физической основе в зависимости от объекта идентификации/аутентификации (пользователь-субъект; процесс; объект-ресурс в виде таблицы, формы, запроса, файла, устройства, каталога и т. д.). В общем плане для идентификации/аутентификации пользователей-субъектов в компьютерных системах могут использоваться их биометрические параметры (отпечатки пальцев, рисунок радужной оболочки глаз, голос, почерк и т. д.), либо специальные замково-ключевые устройства (смарт-карты, магнитные карты и т. п.). Однако при доступе непосредственно в АИС (в базы данных), чаще всего используются *парольные системы* идентификации/аутентифшкации.

Парольные системы основаны на предъявлении пользователем в момент аутентификации *специального секретного* (известного только подлинному пользователю) *слова* или набора символов — *пароля.* Пароль вводится пользователем с клавиатуры, подвергается криптопреобразованию и сравнивается со своей зашифрованной соответствующим образом учетной копией в системе. При совпадении внешнего и внутреннего *парольного аутентификатора* осуществляется распознавание и подтверждение подлинности соответствующего субъекта.

Парольные системы являются простыми, но при условии правильной организации подбора и использования паролей, в частности, безусловного сохранения пользователями своих паролей втайне, достаточно надежным средством аутентификации, и, в силу данного обстоятельства, широко распространены.

Основной недостаток систем парольной аутентификации заключается в принципиальной оторванности, отделимости аутентификатора от субъекта-носителя. В результате пароль может быть получен тем или иным способом от законного пользователя или просто подобран, подсмотрен по набору на клавиатуре, перехвачен тем или иным способом в канале ввода в систему и предъявлен системе злоумышленником.

Поэтому в некоторых случаях парольные аутентифи
ры могут усиливаться диалогово-вопросными сис

системами «коллективного вхождения». В диалогово
ных системах для каждого зарегистрирован
создается некоторая база вопросов я ответов
купности и в деталях могут быть известны только подлинному пользователю (например, сведения чисто личного характера). В результате внутренний образ субъекта существенно расширяется и появляется возможность варьирования аутентификатора при каждом следующем входе пользователя в систему. При входе пользователя в систему монитор безопасности (субъект аутентификации) формирует случайную выборку вопросов и, чаще всего, по статистическому критерию «n из m» принимает решение об аутентификации.

В системах коллективного вхождения парольную аутентификацию должны одновременно пройти сразу все зарегистрированные для работы в системе пользователи. Иначе говоря, поодиночке пользователи работать в системе не могут. Вероятность подбора, перехвата и т. д. злоумышленником (злоумышленниками) сразу всех паролей, как правило, существенно меньше, и, тем самым, надежность подобных систем аутентификации выше.

Аутентификации *в распределенных информационных системах* в принципе должны подвергаться и объекты (ресурсы, устройства), а также процессы (запросы, пакеты и т. д.). Аутентифицированный (подлинный) пользователь, обращаясь к объектам системы и порождая соответствующие процессы, должен, в свою очередь, убедиться в их подлинности,* например, отправляя распечатать сформированный в базе данных конфиденциальный отчет на сетевой принтер, специально предназначенный для распечатки соответствующих конфиденциальных документов.

---

* Так называемый принцип *надежного пути* и *гарантированности архитектуры*.

Как правило, для аутентификации объектов применяются технологии асимметричных криптосистем, называемых иначе системами с открытым ключом, рассмотрение которых выходит за допустимый объем данного пособия.

Для аутентификации процессов широкое распространение нашли технологии меток (дескрипторов) доступа.

***Технология меток*** или ***дескрипторов доступа*** отражает сочетание одноуровневой и многоуровневой моделей безопасности данных и основывается на присвоении администратором системы всем объектам и субъектам базы данных специальных дескрипторов доступа, содержащих *набор параметров уровня конфиденциальности, допустимых операциях, допустимых имен объектов или субъектов доступа и других особых условий доступа.* Субъект доступа, *инициируя* в соответствии со своим дескриптором (меткой) разрешенный *процесс,* передает ему свою *метку доступа* (помечает своей меткой). Ядро безопасности СУБД (ТСВ) проверяет ***подлинность метки процесса,*** сравнивая ее с меткой доступа пользователя-субъекта, от имени которого выступает процесс. При положительном результате метка доступа процесса сравнивается с меткой доступа объекта, операцию с которым намеревается осуществлять процесс. Если дескрипторы доступа процесса и объекта совпадают (или удовлетворяют правилам и ограничениям политики безопасности системы), монитор безопасности разрешает соответствующий доступ, т. е. разрешает осуществление процесса (операции).

Проверка подлинности метки процесса предотвращает возможные угрозы нарушения безопасности данных путем формирования субъектом для инициируемого им процесса такой метки, которая не соответствует его полномочиям.

Для проверки подлинности меток в системе формируется специальный ***файл (массив) учетных записей.*** При регистрации нового пользователя в системе для него создается учетная запись, содержащая его идентификационный номер (идентификатор), парольный аутентификатор и набор дескрипторов доступа к объектам базы данных (метка доступа). При инициировании пользователем (субъектом) какого-либо процесса в базе данных и передаче ему своей метки доступа ядро безопасности СУБД подвергает метку процесса криптопреобразованию, сравнивает ее с зашифрованной меткой соответствующего субъекта (пользователя) в массиве учетных записей и выносит решение о подлинности метки.

*Массив учетных записей,* в свою очередь, является *объектом высшей степени конфиденциальности в системе,* и доступен только администратору. Ввиду исключительной важности массива учетных записей для безопасности всей системы помимо шифрования его содержимого принимается ряд

*дополнительных мер к его защите,* в том числе специальный режим его размещения, проверка его целостности, документирование всех процессов над ним.

Таким образом, на сегодняшний день наработан и используется развитый набор технологий идентификации/аутентификации в защищенных компьютерных системах. Вместе с тем основные бреши безопасности чаще всего находятся злоумышленниками именно на этом пути.

### 7.2.2.2. Языки безопасности баз данных

Для определения конкретных назначений или установления правил и ограничений доступа при проектировании банков данных АИС, а также в целях управления системой разграничения доступа и в более широком смысле системой коллективной обработки данных администратору системы необходим специальный инструментарий. Такой инструментарий должен основываться на определенном *языке, позволяющем описывать и устанавливать те или иные назначения доступа и другие необходимые установки политики безопасности в конкретной АИС.*

В реляционных СУБД такой язык должен являться соответственно *составной частью языка SQL.* Исторически впервые подобные вопросы были поставлены и реализованы в упоминавшихся языках SEQUEL (созданного в рамках проекта System R фирмы IBM) и языка QUEL (созданного в рамках проекта INGRES) и послуживших в дальнейшем основой для языка SQL.

Как уже отмечалось в п. 4.1, в перечне базовых инструкций языка SQL представлены инструкции *GRANT* и *REVOKE,* предоставляющие или отменяющие *привилегии пользователям.* Структура инструкции *GRANT* выглядит следующим образом:

*GRANT*список_привилегий_через_запятую *ON* ИмяОбъекта
*TO*ИменаПользователей_через_запятую
[*WITHGRANTOPTION*];

где:

— список привилегий составляют разрешенные инструкции (операции) над объектом (таблицей) — *SELECT, INSERT, UPDATE, DELETE;*

— список пользователей представляется их именами-идентификаторами или может быть заменен ключевым словом *PUBLIC,* которое идентифицирует всех пользователей, зарегистрированных в системе;

— директива *WITH GRANT OPTION* наделяет перечисленных пользователей дополнительными особыми полномочиями по предоставлению (перепредоставлению) указанных в списке привилегий-полномочий другим пользователям.

В большинстве случаев право подачи команд *GRAND* и *REVOKE* по конкретному объекту автоматически имеют пользователи, создавшие данный объект, т. е. их *владельцы.* В других подходах этим правом наделяются доверенные субъекты, т. е. администраторы.

Хотя в явном виде такой подход не предусматривает создание матрицы доступа,* тем не менее, реализуется классический принцип *дискреционного* ***разграничения доступа*** с сочетанием как *добровольного,* так и *принудительного управления доступом.*

* На самом деле в большинстве СУБД привилегии и установки доступа, как и структура базы данных, «прописываются» в системных таблицах БД, т.е. в системном каталоге БД. который можно рассматривать, в том числе и в качестве матрицы доступа.

Как уже отмечалось, дискреционный принцип обладает большой гибкостью по настройке системы разграничения доступа на особенности предметной области базы данных и потребности пользователей, но не обеспечивает эффективной управляемости и затрудняет проведение какой-либо целенаправленной политики безопасности в системе. Преодоление этого недостатка достигается двумя путями — использованием техники «представлений» и специальными расширениями языка SQL.

Использование *техники представлений* является распространенным технологическим приемом формирования для конкретного пользователя своего «видения» базы данных (виртуальной БД) и осуществления на этой основе разграничения доступа к данным в реляционных СУБД. Напомним, что «представлением» называется глобальный авторизованный запрос на выборку данных, формирующий для пользователя «свое» представление определенного объекта (объектов), совокупность которых формирует некую виртуальную базу данных, со своей схемой (объектами) и данными (отобранными или специально преобразованными). При входе пользователя в систему в

процессе его идентификации и аутентификации ядро безопасности отыскивает для пользователя соответствующие представления-запросы и передает запрос основному ядру СУБД для выполнения. В результате выполнения запроса пользователь «видит» и имеет доступ только к тем объектам, которые соответствуют его полномочиям и функциям.

В целом создание системы разграничения доступа через технику представлений является более простым способом, чем непосредственное использование инструкций *GRANT,* и осуществляется в два этапа:

1. Для всех зарегистрированных пользователей в системе с помощью конструкций *CREATE VIEW* создаются свои представления объектов базы данных.
2. С помощью инструкций *«GRANT SELECT ON*ИмяПредставления *TO* ИмяПользователя» созданные представления авторизуются со своими пользователями.

Вместе с тем такой подход является более грубым по сравнению с применением инструкции *GRANT* непосредственно к объектам базы данных, т. к. не обеспечивает расщепления установок доступа к объектам на уровне отдельных операций *(SELECT, INSERT, UPDATE, DELETE).*

Поэтому другим подходом являются специальные расширения языка SQL, основанные на событийно-процедурной идеологии с введением специальных правил (*RULE)* безопасности:

*CREATESECURITYRULE*ИмяПравила
*GRANT*список_привилегий_через_запятую *ON* ИмяОбъекта
*WHERE* условия
*TO*ИменаПользователей_через_запятую
Реакция_на_нарушение_правила;

Введение правил безопасности обеспечивает более широкие и гибкие возможности реализации различных политик безопасности с большей степенью контроля и управляемости, но, как, впрочем, и техника представлений и непосредственное использование инструкций *GRANT,* не позволяет строить системы с *мандатным принципом разграничения доступа,* который считается более жестким и надежным.

Для решения этой задачи могут предлагаться более кардинальные расширения языка SQL с введением возможностей создания *объектов базы данных с метками конфиденциальности.* Следует, однако, заметить, что подобные примеры в коммерческих и сертифицированных по требованиям безопасности СУБД чрезвычайно редки.

В заключение по языкам безопасности баз отметим, что в современных СУБД для реализации установок, правил и ограничений доступа разрабатывается и используется специальный диалогово-наглядный интерфейс, автоматически формирующий соответствующие конструкции языка SQL и позволяющий в большинстве случаев обходиться без непосредственного программирования.

### 7.2.2.3. Безопасность повторного использования объектов

Компьютерная система в целом, устройства оперативной и внешней (дисковой) памяти в частности, являются классическим примером среды многократного повторного информационного использования. *Технологии обеспечения безопасности повторного использования объектов* направлены на предотвращение *угроз безопасности* от случайного или преднамеренного *извлечения интересующей злоумышленника информации по следам предшествующей деятельности или из технологического «мусора».*

Часть этих технологий реализуются на уровне операционных систем, а часть являются специфическими функциями, осуществляемыми в автоматизированных информационных системах СУБД. Данные технологии условно можно разделить на три группы:

- изоляция процессов;
- очистка памяти после завершения процессов;
- перекрытие косвенных каналов утечки информации.

***Изоляция процессов*** является стандартным принципом и приемом обеспечения надежности многопользовательских (многопроцессных) систем и предусматривает выделение каждому процессу своих непересекающихся с другими вычислительных ресурсов, прежде всего областей оперативной памяти. В СУБД данные задачи решаются мониторами транзакций, рассмотренными в п. 5.2.5.

***Очистка памяти после завершения процессов*** направлена непосредственно на предотвращение несанкционированного доступа к конфиденциальной информации после завершения работы процессов с конфиденциальными данными уполномоченными пользователями. Так же как и

изоляция процессов, чаще всего данная функция выполняется операционными системами. Кроме того, следует отметить, что очистке подлежат не только собственно участки оперативной памяти, где во время выполнения процессов размещались конфиденциальные данные, но и участки дисковой памяти, используемые в системах виртуальной памяти, в которых или операционная система, или сама СУБД АИС временно размещает данные во время их обработки. Ввиду этого функции очистки дисковой памяти после завершения выполнения отдельных транзакций, запросов, процедур могут входить в перечень обязательных функций самой СУБД в части ее надстройки над возможностями операционной системы по организации размещения данных и доступу к данным на внешней и в оперативной памяти (см. п. 2.1).

Как уже отмечалось, при реализации систем разграничения доступа возможны ***косвенные*** *каналы утечки информации.* Помимо проблем с разграничением доступа на уровне полей, источниками косвенных каналов утечки информации являются еще и ряд технологических аспектов, связанных с характеристиками процессов обработки данных. В этом плане различают косвенные каналы *«временные»** и *«по памяти».*

* Ударение на последнем слоге.

В первом случае некоторые элементы конфиденциальной информации или, во всяком случае, подозрение на наличие такой информации, неуполномоченный пользователь получает на основе анализа времени выполнения отдельных процессов уполномоченными пользователями (скажем, по неправдоподобно большому времени для очевидно простой или какой-либо типовой операции).

Во втором случае соответственно «подозрение» вызывает занимаемый объем некоторых объектов (файлов, таблиц и т. п.), объем содержимого которых, с точки зрения того, что «видит» пользователь, явно не соответствует их фактическому объему.

Специфический косвенный канал утечки информации, относящийся именно к СУБД, в том числе и при использовании техники «представлений», связан с агрегатными (статистическими) функциями обработки данных, который можно пояснить на следующем примере. Предположим, пользователь в своем «представлении» базы данных видит не всю таблицу «Сотрудники», а только записи, непосредственно относящиеся к его подразделению. Тем не менее, формируя и выполняя запрос на выборку данных с вычисляемым полем «МахОклад» при использовании соответствующей агрегатной статистической *функции* «Max()>>, пользователь может получить хоть и в обезличенном виде, но не предназначенные для него данные, в том случае, если процессор и оптимизатор запросов не «сливает» сам запрос с соответствующим «представлением» базы данных.

Иначе говоря, все операции обработки данных должны выполняться строго над той выборкой данных, которая соответствует «представлению» пользователя.

Приведенный пример иллюстрирует также то, что монитор безопасности СУБД должен реализовываться на нулевом уровне, т. е. на уровне ядра системы, являясь органичной составной частью компонент представления данных и доступа к данным (см. рис. 7.2).

В наиболее критичных сточки зрения безопасности информации случаях применяют еще один, наиболее жесткий вариант — ***технологию разрешенных процедур.*** В этом случае пользователям системы разрешается работать с базой данных ***исключительно*** *через запуск разрешенных процедур.*

Данный подход основывается на также уже рассмотренной ***технике хранимых (stored) процедур.*** Администратором системы для каждого пользователя формируется набор процедур обработки данных в соответствии с полномочиями и функциями пользователя. Для безопасности хранимые процедуры в файле данных *шифруются.*

Ядро СУБД после идентификации и аутентификации пользователя при его входе в систему предъявляет ему разрешенный набор процедур. Непосредственного доступа к самим данным пользователь не имеет и работает только с результатами их обработки по соответствующим процедурам.

Данные АИС размещаются в *файлах данных,* структура которых может быть известна нарушителю. Поэтому еще одной *угрозой безопасности* данных АИС является возможность *несанкционированного доступа к файлам данных вне программного обеспечения АИС (СУБД) средствами операционной системы или дисковых редакторов.* Для *нейтрализации* этой угрозы применяются ***методы и средства криптозащиты.*** В большинстве случаев криптографические средства защиты встраиваются непосредственно в программное обеспечение СУБД. Файл (файлы) базы данных* при размещении на устройствах дисковой памяти шифруется. Соответственно, криптографическая подсистема при открытии файла данных его дешифрует. Специфической особенностью СУБД

является особый порядок работы с файлами базы данных через организацию специальной буферизации страниц в оперативной памяти. Поэтому криптографическая подсистема встраивается в ядро СУБД, перехватывая все операции считывания страниц в оперативную память и, соответственно их дешифрируя, и все операции обратного «выталкивания» страниц из оперативной памяти на диски и, соответственно, шифруя данные.

---

* В том числе с авторизованными хранимыми процедурами и запросами представлений.

В развитых СУБД имеется возможность выборочного шифрования объектов базы данных исходя из уровня их конфиденциальности, вплоть до шифрования отдельных полей записей.

### 7.2.2.4. Надежное проектирование и администрирование

Часть *угроз безопасности* информации возникает из-за непреднамеренных (или преднамеренных) *ошибок на этапах жизненного цикла АИС* — при разработке программного обеспечения СУБД; при проектировании и создании на базе СУБД конкретной АИС, и, в том числе, при проектировании системы разграничения доступа; при администрировании и сопровождении системы и, в том числе, при реагировании и действиях пользователей во внештатных ситуациях; при технологических операциях по резервированию, архивированию и восстановлению информации после сбоев; при выводе АИС из эксплуатации. С целью нейтрализации или снижения вероятности данных угроз применяются ряд организационно-технологических и технических средств, решений, объединяемых в общую группу *технологий надежного проектирования и администрирования.* Их также условно можно разделить на следующие подгруппы:

- технологии надежной разработки программного обеспечения;
- технологии надежного проектирования и создания АИС;
- технические средства и специальный инструментарий администрирования АИС;
- протоколирование и аудит процессов безопасности.

*Технологии надежной разработки программного обеспечения* включают общие подходы к снижению ошибок при разработке программного кода и ряд более специфических аспектов, основанных на *изначальном учете в концепции и структуре ядра системы* (подсистема представления данных и подсистема доступа к данным) той или иной *модели и технологий безопасности данных.* Как показывает анализ выявленных уязвимостей в системах безопасности компьютерных систем, вероятность наличия и нахождения злоумышленниками брешей существенно выше в тех случаях, когда системы защиты реализуются в виде надстройки или внешней оболочки над ядром и интерфейсом исходных незащищенных систем.

*Технологии надежного проектирования* и создания на базе программного обеспечения СУБД конкретных АИС направлены на предотвращение логических ошибок в информационной инфраструктуре систем и в подсистемах разграничения доступа, строящихся на основе поддерживаемой СУБД модели и технологий безопасности данных. В этом отношении основным и широко распространенным является *структурно-функциональный подход.*

При наличии большого количества пользователей (субъектов) и объектов информационных систем (баз данных) схема разграничения доступа может быть очень сложной и запутанной, что создает трудности для администрирования и порождает предпосылки для логических ошибок. Для преодоления этой угрозы в рамках структурно-функционального подхода применяют *технику рабочих групп.*

***Рабочая группа*** объединяет *пользователей, имеющих какое-либо* ***общее*** *технологическое отношение к базе данных* (выполняющих похожие операции) и ***близкие*** *параметры конфиденциальности по отношению к общим данным.*

Администратор системы может создавать рабочие группы, рассматривая их как ***коллективных пользователей,*** с определенной идентификацией и набором полномочий, т.е. с созданием специальных учетных записей для рабочих групп. Каждый пользователь обязательно должен являться членом какой-либо рабочей группы. Полномочия, определенные для рабочей группы, автоматически распространяются на всех пользователей — членов группы, что является отражением некоторых элементов зонально-функционального принципа разграничения доступа. Дополнительно для каждого пользователя в его личной учетной записи могут быть уточнены и конкретизированы его полномочия.

Такой подход позволяет в большинстве случаев *существенно уменьшить количество субъектов доступа в системе, сделать схему разграничения доступа более простой, «прозрачной» ч управляемой,* и тем самым *снизить вероятность* таких *логических ошибок* как неправильное предоставление доступа конкретного пользователя к конкретному объекту, превышение полномочий конкретного пользователя по доступу к ряду объектов, предоставление избыточных прав доступа и т.п.

Процессы в базе данных в технологиях рабочих групп помечаются как меткой пользователя, так и меткой рабочей группы, и, соответственно ядро безопасности СУБД проверяет подлинность обеих меток.

*Проектирование* системы доступа на основе технологии рабочих групп может проводиться «сверху» *(дедуктивно)* и «снизу» *(индуктивно)*.

В первом способе сначала на основе анализа функциональной структуры и организационной иерархии пользователей (субъектов) формируются рабочие группы и осуществляются групповые назначения доступа. Далее каждый пользователь при его регистрации в системе включается в состав одной или нескольких групп, отвечающих его функциям. И, наконец, в заключение для каждого пользователя анализируются особенности его функциональных потребностей и доверительных характеристик и при необходимости осуществляются индивидуальные дополнительные назначения доступа. Формирование групп, групповые и индивидуальные установки доступа при этом *осуществляются администратором системы,* что соответствует принудительному способу управления доступом.

Такой подход позволяет снизить вероятность ошибочных назначений доступа и обеспечивает жесткую централизованную управляемость системой доступа, но может порождать, в свою очередь, дублирование групповых и индивидуальных полномочий доступа субъектов к объектам (проблема дублирования), а также избыточность доступа субъекта к одним и тем же объектам через участие в разных группах (проблема пересечения групп или в более широком смысле проблема оптимизации групп).

При втором (индуктивном) способе проектирования рабочих групп первоначально осуществляются индивидуальные назначения доступа субъектов (пользователей) к объектам. Назначения производятся на основе опроса и анализа функциональных потребностей и доверительных характеристик пользователей, и могут осуществляться администратором системы (принудительный способ управления доступом) или через индивидуальные запрашивания субъектами доступа владельцев объектов (принцип добровольного управления доступом). Далее, уже администратором системы, производится анализ общих или схожих установок доступа у различных субъектов, на основе которого они объединяются в рабочие группы. Выделенные общие установки доступа используются в качестве групповых назначений доступа. При этом анализ схожести доступа при большом количестве субъектов и объектов представляет непростую задачу и решается администратором системы в значительной степени эвристически.

Дополнительным организационным способом повышения надежности и безопасности в процессе администрирования и сопровождения системы ***является разделение общего администрирования и администрирования безопасности***. Общий администратор строит, поддерживает и управляет информационной инфраструктурой системы — информационно-логическая схема, категорирование конфиденциальности объектов (ресурсов и устройств), интерфейсные и диалоговые элементы, формы, библиотеки запросов, словарно-классификационная база, резервирование и архивирование данных. Администратор безопасности организует и управляет системой разграничения доступа — доверительные характеристики (допуска) пользователей, конкретные назначения доступа, регистрация и формирование меток доступа пользователей.

Доступ к массиву учетных записей пользователей имеет только администратор безопасности. Совмещение функций общего администрирования и администрирования безопасности одновременно одним пользователем* не допускается, что объективно повышает надежность системы.

---

* То есть администратор в течение одного сеанса работы может выполнять только одну функцию (роль) — или общего администратора или администратора безопасности.

*Технические средства и специальный инструментарий администрирования* АИС применяются для создания условий и возможностей восстановления данных после всевозможных сбоев, обеспечения более эффективной работы АИС, и уже рассматривались в п. 7.1.

***Протоколирование и аудит событии безопасности*** являются важным средством обеспечения управляемости состоянием и процессами безопасности, создают условия для расследования фактов нарушения информационной безопасности, анализа и исключения их причин, снижения отрицательных последствий и ущерба от них.

*Документированию* подлежат все события, критичные с точки зрения безопасности в системе:

• вход/выход пользователей;
• регистрация новых пользователей, смена привилегий и назначений доступа (все обращения к массивам учетных записей);
• все операции с файлами (создание, удаление, переименование, копирование, открытие, закрытие);
• обращения к/из удаленной системе(ы).

При этом по каждому такому событию устанавливается минимально необходимый перечень регистрируемых параметров, среди которых:

• дата и время события;
• идентификатор пользователя-инициатора;
• тип события;
• источник запроса (для распределенных систем — сетевое имя терминала, рабочей станции и т. п.);
• имена затронутых объектов;
• изменения, внесенные в учеты в системы, в том числе в массивы учетных записей;
• метки доступа субъектов и объектов.

В СУБД такой подход хорошо вписывается в *событийно-процедурную технологию* с использованием техники *журнализации.* При этом доступ к журналу событий имеет только администратор безопасности, который при обнаружении фактов или признаков нарушений безопасности имеет возможность восстановления хода событий, анализа и устранения источников и причин нарушения безопасности системы.

В этом отношении журнал событий безопасности является необходимым средством аудита безопасности. *Аудит безопасности* заключается в контроле и отслеживании событий в системе с целью выявления, своевременного обнаружения проблем или нарушений безопасности и сигнализации об этом администратору безопасности. Ввиду того что процессы доступа, различных процедур, операций, информационных потоков в компьютерных системах являются многоаспектными, не строго детерминированными, т. е. частично или полностью стохастическими, разработка аналитических, алгоритмических или иным образом аналитических автоматизированных процедур обнаружения фактов и признаков нарушений информационной безопасности является чрезвычайно сложной и неопределенной задачей. Поэтому в настоящее время разрабатывается ряд эвристических и нейросетевых технологий, которые в некоторых случаях с успехом воплощаются в специальном *программном инструментарии администратора безопасности,* обеспечивая автоматизированный аудит безопасности системы.*

---

* В простейших и наиболее распространенных случаях такой инструментарии основывается на базе ранее выявленных для данной компьютерной системы (ОС, СУБД) брешей безопасности при различных вариантах информационно-логической и информационно-транспортной инфраструктуры и работает по принципу антивирусных сканеров.

Таким образом, набор технологических задач, подходов, средств и инструментов, обеспечивающих практическую реализацию аксиоматических принципов, моделей и политик безопасности чрезвычайно широк, что объективно требует некоторой единой шкалы требований, методик разработки и оценки (сертификации) защищенных систем по вопросам безопасности.

Данные задачи обеспечивают стандарты защищенных систем, представленные в нашей стране «Руководящими документами Государственной технической комиссией при Президенте РФ по защите информации от несанкционированного доступа».

### 7.2.3. Требования и классы защищенности автоматизированных (информационных) систем в «Руководящих документах...» Государственной технической комиссии при Президенте РФ

Руководящие документы Гостехкомиссии при Президенте РФ, изданные в 1992 г., являются отечественными стандартами в сфере построения защищенных информационных систем. Методологической основой руководящих документов, определяющей систему взглядов на проблему информационной безопасности и основные принципы защиты компьютерных систем, является «Концепция защиты СВТ и АС от несанкционированного доступа (НСД) к информации».

«Концепция...» разделяет требования безопасности к средствам вычислительной техники как совокупности программных и технических элементов систем обработки данных, способных функционировать самостоятельно или в составе других систем, и требования безопасности к автоматизированным системам.*

* Система, состоящая из персонала и комплекса средств автоматизации его деятельности, реализующая информационную технологию выполнения установленных функций (по ГОСТ 34.003-90).

Особенностью руководящих документов является акцентирование требований в основном на первой составляющей безопасности информации — конфиденциальности (защите от несанкционированного доступа) и частично второй составляющей — целостности (защите от несанкционированного изменения информации).

«Концепция...» классифицирует модели нарушителя по четырем иерархическим уровням возможностей, содержание которых приведено на рис. 7.6.

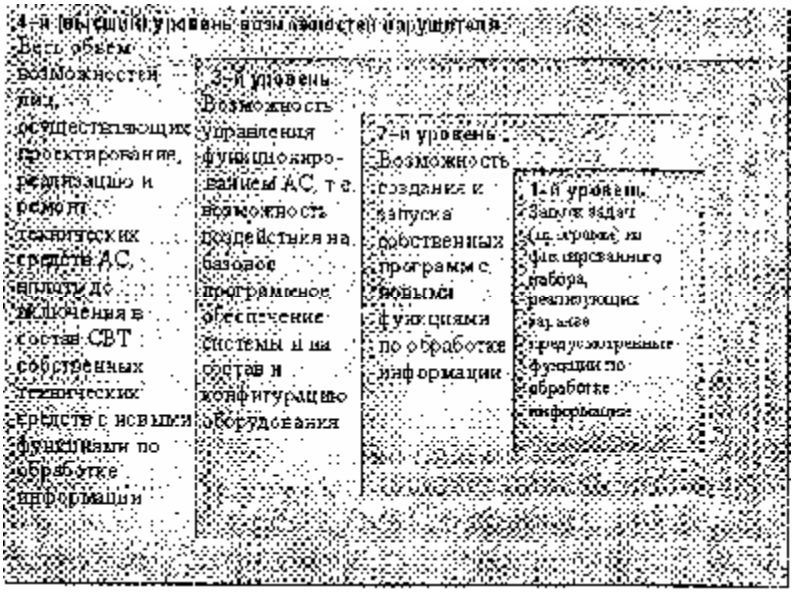



*Рис. 7.6. Уровни возможностей нарушителя, установленные в руководящих документах Гостехкомиссии*

В своем уровне нарушитель является специалистом высшей квалификации, знает все об АС и, в частности, о системе и средствах защиты.

Требования ***по классам*** защищенности АС разделяются на основе трех признаков:

- наличие в АС информации различного уровня конфиденциальности;
- наличие или отсутствие различий в уровне полномочий пользователей;
- наличие коллективного или индивидуального режима обработки данных.

Требования к средствам защиты информации (СЗИ) в руководящих документах определяются совокупностью параметров по ряду подсистем защиты, схема и структура которых приведены на рис. 7.7.

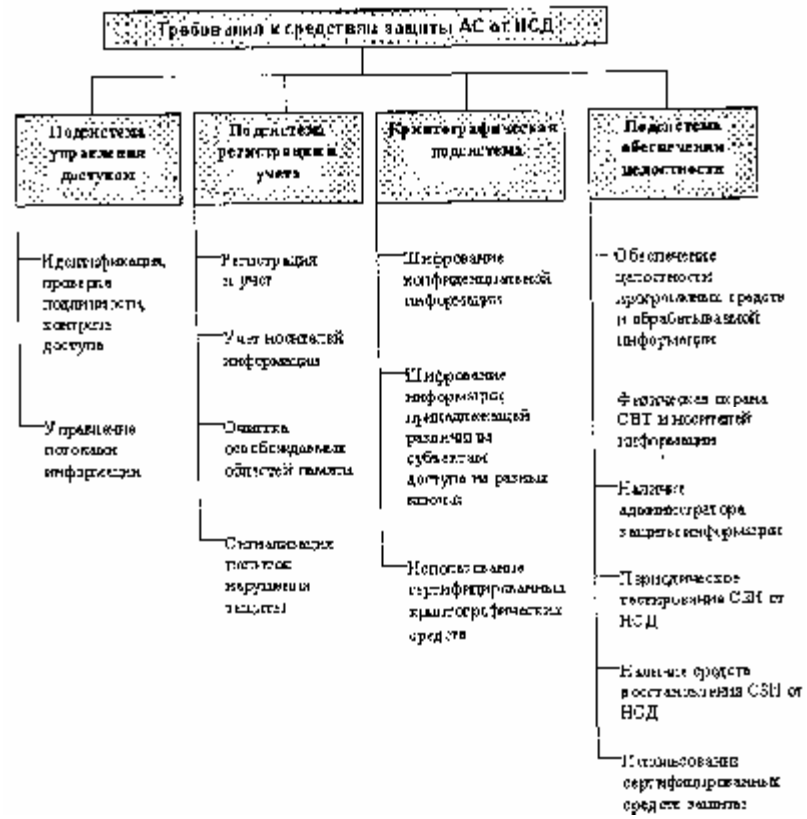



*Рис. 7.7. Составляющие и структура требований к СЗИ в АС в руководящих документах Гостехкомиссии*

В соответствии с этими признаками руководящие документы выделяют три группы классов защищенности, приведенные на рис. 7.8.

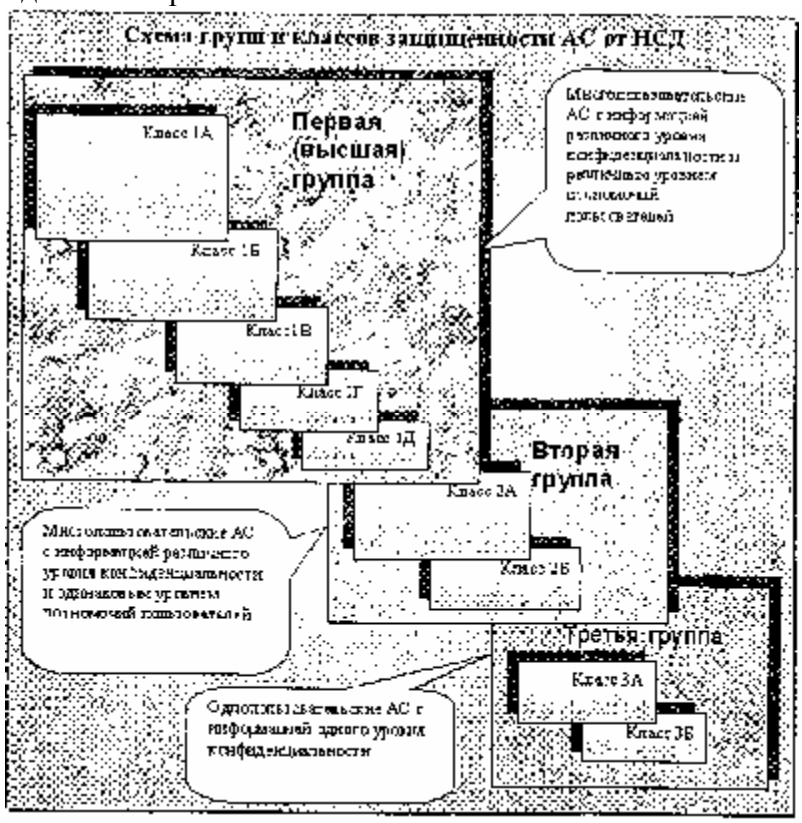



*Рис. 7.8 Группы и классы защищенности АС от НСД в руководящих документах Гостехкомиссии*

Каждая группа, в свою очередь, подразделяется на ряд подгрупп-классов, что в совокупности определяет 9 классов защищенности АС, иерархия которых характеризуется уже упоминавшимися особенностями конфиденциальности информации, полномочиями пользователей, режимом обработки и уровнем предъявляемых требований к средствам защиты — 1А, 1Б,1В,1Г,1Д,2А,2Б,3А,3Б.*

* Классы перечислены по направлению снижения требований к уровню защищенности.

Перечень требований по конкретным классам защищенности АС от НСД представлен в таблице 7.2.

*Таблица 7.2*

Требования к классам защищенности АС

| Подсистемы и требования | Классы | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 3Б | 3А | 2Б | 2А | 1Д | 1Г | 1В | 1Б | 1А |
| 1. Подсистема управления доступом | | | | | | | | | |
| *1.1. Идентификация, проверка подлинности и контроль доступа субъектов в систему:* | | | | | | | | | |
| к терминалам, ЭВМ, узлам сети ЭВМ, каналам связи, внешним устройствам ЭВМ, | - | - | - | - | - | - | + | + | + |
| к программам, | - | - | - | - | - | - | + | + | + |
| к томам, каталогам, файлам, записям, полям записей | - | - | - | - | - | - | - | + | + |
| *1.2. Управление потоками информации* | - | - | - | + | - | - | - | + | + |
| 2. Подсистема регистрации и учета | | | | | | | | | |
| *2.1. Регистрация и учет:* входа/выхода субъектов доступа в систему (узел сеть) | + | + | + | + | + | + | + | + | + |
| выдачи печатных (графических) выходных документов, | - | + | - | + | - | - | - | - | + |
| запуска/завершения программ и процессов (задач, заданий), | - | + | + | + | - | + | + | + | + |
| доступа программ субъектов к защищаемым файлам, включая их создание и уничтожение, изменение прав доступа, | | | - | + | - | + | + | - | - |
| доступа программ субъектов доступа к терминалам, ЭВМ, узлам сети ЭВМ, каналам связи, внешним устройствам ЭВМ, программам, каталогам, файлам, записям, полям записей, | - | + | - | - | - | + | - | + | + |
| изменения полномочий субъектов доступа, | - | + | - | - | - | - | - | + | + |
| создаваемых защищаемых объектов доступа. | - | + | + | + | - | | - | + | + |
| *2.2. Учет носителей информации* | + | + | + | + | + | + | - | + | + |
| *2.3. Очистка (обнуление, обеспечение перезаписи) освобождаемых областей оперативной памяти ЭВМ и внешних носителей* | | + | - | + | - | + | - | + | + |
| *2.4. Сигнализация попыток нарушения* | - | - | - | - | - | - | + | + | - |
| 3. Криптографическая подсистема | | | | | | | | | |
| *3.1. Шифрование конфиденциальной информации* | - | - | - | + | - | - | - | + | + |
| *3.2. Шифрование информации, принадлежащей различным субъектам доступа (группам субъектов) на различных ключах* | - | - | - | - | - | - | - | + | + |
| *3.3. Использование аттестованных (сертифицированных) криптографических средств* | - | - | - | + | - | - | - | + | + |
| 4. Подсистема обеспечения целостности | | | | | | | | | |
| *4.1. Обеспечение целостности программных средств и обрабатываемой информации* | + | + | + | + | + | + | + | + | + |
| *4.2. Физическая охрана средств вычислительной техники и носителей информации* | + | + | + | + | + | + | + | + | + |
| *4.3. Наличие администратора (службы) защиты информации в АС* | - | - | - | - | - | - | + | + | + |
| *4.4. Периодическое тестирование СЗИ НСД* | + | + | + | + | + | + | + | + | + |
| *4.5. Наличие средств восстановления СЗИ НСД* | + | + | + | + | + | + | + | + | + |
| *4.6. Использование сертифицированных средств защиты* | - | + | - | + | - | - | + | + | - |

В краткой характеристике можно отметить следующее.

***Группа 3*** классифицирует АС, в которых работает один пользователь, допущенный ко всей информации АС, размещенной на носителях одного уровня конфиденциальности. В группе устанавливается два класса — 3Б и 3А. *В классе 3Б,* соответствующем низшему уровню защиты, регламентируются требования обязательной парольной идентификации и аутентификации пользователя при входе в систему, регистрации входа/выхода пользователей, учета используемых внешних носителей, обеспечения целостности средств защиты информации (СЗИ), обрабатываемой информации и программной среды, а также наличие средств восстановления СЗИ.

*В классе 3А* выполняются все требования класса 3Б и дополнительно устанавливаются требования по регистрации распечатки документов, физической очистке освобождаемых областей оперативной памяти и внешних носителей, усиливаются требования по обеспечению целостности СЗИ и программной среды через проверку целостности при каждой загрузке системы, периодическое тестирование функций СЗИ при изменении программной среды и персонала АС.

***Группа 2*** классифицирует АС, в которых пользователи имеют одинаковые права доступа (полномочия) ко всей информации АС, обрабатываемой и (или) хранимой на носителях различного уровня конфиденциальности. В группе также устанавливается два класса — 2Б и 2А. Регламентируемые требования *класса 2Б* в основном совпадают с требованиями класса 3Б с некоторым усилением требований по подсистеме обеспечения целостности (при загрузке системы).

*В классе 2А* выполняются все требования класса 2Б с усилением требований по подсистеме управления доступом (идентификация терминалов, ЭВМ, узлов сети ЭВМ, каналов связи, внешних устройств, а также программ, томов, каталогов, файлов, записей, полей записей, что обеспечивает избирательное управление доступом) и усилением требований по подсистеме регистрации и учета (регистрация не только входа/выхода субъектов, но загрузки и инициализации операционной системы, программных остановов, регистрация выдачи документов, запуска программ, обращений к файлам и другим защищаемым объектам, автоматический учет создаваемых файлов, что обеспечивает регистрацию всех потенциально опасных событий). Дополнительно регламентируется управление потоками информации с помощью меток конфиденциальности (элементы полномочного управления доступом), очистка освобождаемых участков оперативной и внешней памяти, а также шифрование всей конфиденциальной информации, записываемой на совместно используемые различными субъектами доступа носители данных.

***Группа 1*** классифицирует многопользовательские АС, в которых одновременно обрабатывается и (или) хранится информация различных уровней конфиденциальности и не все пользователи имеют право доступа ко всей информации АС. Группа содержит пять классов—1Д, 1Г, 1В, 1Б и 1А.

*Класс 1Д* включает требования, содержательно и идеологически совпадающие с требованиями классов 3Б и 2Б.

*Класс* 1Г помимо выполнения всех требований класса 1Д включает требования, содержательно и идеологически сходные с требованиями класса 2А (за исключением требований по шифрованию информации) с учетом различий в полномочиях пользователей — избирательное управление доступом в соответствии с матрицей доступа, регистрация потенциально опасных событий, очистка освобождаемых участков оперативной и внешней памяти.

*Класс 1В* включает выполнение всех требований класса 1Г. Дополнительно регламентируется полномочное управление доступом (метки конфиденциальности объектов и полномочия субъектов доступа), усиливаются требования к подсистеме регистрации опасных событий, вводится требование наличия администратора защиты и его интерактивного оповещения о попытках несанкционированного доступа.

*Класс 1Б* включает все требования класса 1В и дополнительно требования по шифрованию информации (аналогично классу 2А).

*Класс 1А* (высший уровень защиты) включает все требования класса 1Б с дополнительным требованием использования разных ключей шифрования различными субъектами доступа.

Руководящие документы Гостехкомиссии, изучение опыта создания и эксплуатации защищенных АС позволяют выделить некоторые методические и организационные аспекты разработки защищенных АС**,** заключающиеся в следующей последовательности действий:

- анализ информационных ресурсов разрабатываемой АС и категорирование их конфиденциальности (определение объектов защиты);
- анализ основных угроз объектам защиты и разработка модели (уровня возможностей) нарушителя;
- определение комплекса организационных мер, программно-технических решений и средств нейтрализации угроз защищаемым объектам (формирование политики и модели безопасности данных) и определение необходимого класса (уровня) защиты;
- формирование основных технических решений по разработке СЗИ создаваемой АС с требуемым уровнем (классом) безопасности;
- разработка, сертификационные испытания или приобретение сертифицированных СЗИ (защищенных систем), приемка в эксплуатацию.

Основополагающим принципом создания защищенных АС является изначальное включение основных технических решений по подсистемам СЗИ в эскизный и рабочий проекты создания АС, т. к. процессы обеспечения безопасности данных, как уже отмечалось, должны реализовываться на уровне ядра системы (ядра ОС**,** СУБД).

В более детальном виде вопросы создания защищенных систем регламентируются в специальном документе Гостехкомиссии — «Временном положении по организации, изготовлению и эксплуатации программных и технических средств защиты информации от НСД в автоматизированных системах и средствах вычислительной техники» (1992 г.)

Для сертификации разрабатываемых защищенных систем и средств защиты в Российской Федерации созданы соответствующие системы сертификации, находящиеся в компетенции ФАПСИ при Президенте РФ (система РОСС RU.000103001 — криптографические средства защиты) и

Гостехкомиссии при Президенте РФ (система РОСС RU.000101.БИОО — СВТ, ОС, СУБД, АС, общесистемные и специализированные программные и технические средства).

По данным на июнь 2000 года среди общедоступных профессиональных коммерческих СУБД на российском рынке сертификат Гостехкомиссии класса 1Б имеют отечественная промышленная реляционная СУБД «ЛИНТЕР» (версия 5.х), относящаяся к классу систем «Клиент-сервер», разработанная и поддерживаемая Воронежским НПП «РЕЛЭКС», и специально подготовленный единичный экземпляр СУБД Oracle, версии 7.х.

**Вопросы и упражнения**

1. Какие (функции администратора связаны с проектированием и вводом АИС в эксплуатацию?
2. Поясните цели и различия операций резервирования и архивирования данных.
3. Какие операции обеспечивают надежность и работоспособность данных АИС?
4. Объясните цели, задачи и суть процессов журнализации в базах данных.
5. Какие задачи, и какими способами решаются в процессе ревизии баз данных?
6. Поясните необходимость и возможное содержание администрирования базы данных в однопользовательских АИС.
7. Дайте определение угрозам безопасности информации и поясните, ущерб каким свойствам (составляющим, характеристикам) информации они наносят в результате своего осуществления?
8. В чем заключается содержание, соотношение и различия понятии политики и моделей безопасности?
9. Дайте определения назначению, функциям и месту монитора безопасности в структуре и функционировании защищенных информационных систем.
10. Приведите определение и поясните суть принципа дискреционного разграничения доступа, способов его организации и управления им.
11. Объясните принцип мандатного разграничения доступа и основные ограничения политики безопасности в модели Белл—ЛаПадула.
12. Почему идентификация и аутентификация являются обязательными технологическими процессами в защищенных компьютерньк системах?
13. Приведите схему аутентификации объектов и поясните основные ее элементы. Где и в каком виде хранятся аутентификаторы?
14. Что является внутренним образом объекта аутентификации в парольных системах? Каковы достоинства и недостатки парольных систем?
15. Какие задачи решаются на основе технологии меток (дескрипторов) доступа?
16. Какие функции обеспечивают языки безопасности баз данных?
17. Какую модель безопасности, принципы организации и управления доступом обеспечивают SQL-инструкции *GRAND* и *REVOKE?*
18. Дайте определение «представления» базы данных. В чем преимущества и недостатки техники представлений в процессах разграничения доступа к данным?
19. Перечислите и поясните основные направления в расширениях языка SQL в реляционных СУБД для реализации более совершенных и надежных систем безопасности данных.
20. Структура базы данных «Табель рабочего времени» содержит следующие таблицы:

•«Сотрудники» — *Таб.№, ФИО, Должность;*

• «Подразделение» — *№№, Наименование, Руководитель;*

• «Табель» — *Таб.№ сотрудника. Дата, Кол-во отработанных часов;*

• «Нетрудоспособность» — *Тиб.№ сотрудника, Причина/Диагноз, Дата начала, Дата окончания, Наличие справки;*

• «Отпуска» — *Таб.№ сотрудника, Вид, Дата начала, Дата окончания.*

Текущие данные вносятся уполномоченными сотрудниками (табельщиками) по своим подразделениям. Для работы и доступа только к «своим» данным составьте представления объектов (таблиц) базы данных табельщику Иванову из отдела снабжения и предоставьте ему соответствующий доступ.

21. Структура базы данных «Учредительство» содержит следующие таблицы:

• «Лицо» — *Код, ФИО, Дата рождения. Месторождения, Паспортные данные;*

• «Организация» — *Код ОКПО, Наименование, Условное наименование, Профиль деятельности* (Производственный, Коммерческий, Посреднический, научно-производственный), *Организационная форма* (ЗАО, ОАО, и т. д.). *Код Руководителя, Код глав. бухгалтера, Телефон;*

- «Учредительство»—*Код, Код учредителя-лица, Код учредителя-организации, Код учрежденной организации, Дата учреждения, Данные документа учредительства, Доля капитала, Форма капитала;*
- «Адрес» — *Код, Код Лица, Код Организации, Город, Район, Улица, № дома, № квартиры, Дата начала, Дата окончания.*

Вводом и корректировкой данных занимаются несколько сотрудников, за каждым из которых закрепляется регистрация и ведение БД по различным организационным формам, профилю деятельности и району размещения регистрируемых организации. Для работы и доступа только к «своим» данным составьте представления объектов (таблиц) базы данных сотруднику Петрову, отвечающему за регистрацию посреднических акционерных обществ, размещающихся в Железнодорожном районе, и предоставьте ему соответствующий доступ.

22. Какие проблемы безопасности призваны решать технологии надежного проектирования и администрирования?
23. Что такое рабочая группа пользователей? Почему техника рабочих групп повышает безопасность в информационных системах?
24. В базе данных «Штаты подразделений» содержатся следующие таблицы:

- «Сотрудники»— *Таб.№, ФИО, Подразделение, Должность (шт. категория);*
- «Штатные категории» — *Код категории, Наименование* (Начальник отдела. Зам. начальника отдела. Начальник сектора, Ведущий инженер. Старший инженер. Инженер, Техник), *должностной оклад;*
- «Подразделения»— *№№, Наименование, Руководитель;*
- «Штаты» —*№№ подразделения, Код штатной категории, Количество должностей.*

База данных необходима для обеспечения работы руководителей структурных подразделений (справочные функции), сотрудников отдела кадров (ведение базы данных) и бухгалтерии (использование в начислении заработной платы). Составьте и обоснуйте целесообразную систему рабочих групп пользователей и таблицу доступа к объектам базы данных.

25. В базе данных «Преподаватели и занятия» содержатся следующие таблицы:

- «Преподаватели» — *Таб.№, ФИО, Кафедра* (Истории, Архивоведения, Документоведения), *Должность* (Зав.каф, Профессор, Преподаватель, Ассистент), *Ученая степень* (Кандидат наук, Доктор наук), *Ученое звание* (Старший научный сотрудник, Доцент, Профессор, Академик);
- «Контракты/Труд.Соглашения» — *№№. Код Преподавателя, Дата заключения, Срок действия, Ставка, Особые условия;*
- «Дисциплины»—*Код дисциплины, Наименование, Количество часов в учебном плане, Форма отчетности* (Экзамен, Дифференцированный зачет. Зачет);
- «Занятия» — *Дата, Время* (1-я пара, 2-я пара, 3-я пара, 4-я пара). *Аудитория, Вид* (Лекция, Практическое занятие, Семинар, Лабораторная работа. Экзамен, Зачет), *Преподаватель, Дисциплина, Код уч. группы;*
- «Итоги сессии» — *Код, Семестр, Код дисциплины, Код Студента, Отметка, Код Преподавателя*
- «Учебные группы» — *Код группы* (И101, И102, И103, И104, И105 и т.д.), *Специализация* (История, Архивоведение, Документоведение), *Таб.№_старосты, Таб.№№_куратора;*
- «Студенты» — *Таб.№№, ФИО, Год рождения, Уч. группа, Отметка о переводе на след.курс.*

База данных необходима для методистов учебного отдела (составление и ведение расписания занятий, учетная работа по распределению студентов по группам, итогам сессий), профессорско-преподавательскому составу (справки по расписанию занятий), работникам отдела кадров (ведение установочных данных по преподавателям и студентам), студентам (справки по расписанию занятий). Составьте и обоснуйте целесообразную систему рабочих групп пользователей и таблицу доступа к объектам базы данных.

26. Какие функции выполняют стандарты защищенности информационных (компьютерных) систем? Какие из них (зарубежные и отечественные) регламентируют АИС (СУБД)?
27. Поясните соотношение уровней возможностей нарушителя по Руководящим документам Гостехкомиссии с категориями специалистов—разработчик ПО АИС, системный программист, администратор АИС, прикладной программист, пользователь.
28. Приведите схему классов защищенности по Руководящим документам Гостехкомиссии и укажите основные особенности трех групп АС.
29. С какого класса, и в какой форме по Руководящим документам Гостехкомиссии регламентируются требования мандатного принципа разграничения доступа к данным?

30. Поясните примерную последовательность создания защищенной информационной системы.

## ЛИТЕРАТУРА


**1.** *Ансофф И.* Стратегическое управление. — М.: Экономика, 1989.

**2.** *Бойко В.В., Савинков В.М.* Проектирование баз данных информационных систем. — М.: Финансы н статистика, 1989.

**3.** *Саймон А.Р.* Стратегические технологии баз данных. Пер. с англ. — М.: Финансы и статистика, 1998.

**4.** *Васкевич Д.* Стратегии Клиент/Сервер. Руководство специалистам по выживанию для специалистов по реорганизации бизнеса. Пер. с англ. — К.: Диалектика, 1996.

**5.** *Месарович М., Такахара Я.* Общая теория систем: математические основы.—М.: Мир, 1978.

**6.** *Цаленко М. Ш.* Моделирование семантики в базах данных. — М.: Наука,1989.

**7.** *Шрейдер Ю.А.* Типология как основа интеграции знаний // НТИ, Сер. 2.—1981.— №11.—С. 1-5.

**8.** *Язык описания данных Кодасил* / Пер. с англ. Под ред. М.Р. Когаловского и Г.К. Столярова.—М.: Статистика, 1981.

**9.** *Codd E.F.* Relational Model of Data for Large Shared Data Banks // Comm/of ACM.—1970.—V.13,№6.

**10.** *Овчаров Л.А., Селетков С.Н.* Автоматизированные банки данных. — М.: Финансы и статистика, 1982.

**11.** *Ладыженский Г.М.* Системы управления базами данных—коротко о главном / СУБД. — 1995. — №№ 1—4.

**12.** *Гилула М.М.* Множественная модель данных в информационных системах.—М.: Наука, 1992.

**13.** *Кузнецов С.Д.* Введение в СУБД / СУБД. — 1995. — №№ 1-4, 1996.—№№ 1-4, 1997. —№№ 1,2.

**14.** *Трофимова И.П.* Системы обработки и хранения информации: Учеб. пособ. Для вузов по спец. «Автоматизированные системы обработки информ. и управл.».—М.: Высшая школа, 1989.

**15.** *Кнут Д.* Искусство программирования для ЭВМ. Т. 1. Основные алгоритмы.—М.: Мир, 1976.

**16.** *Спенс Р.* Clipper. Руководство по программированию. Версия 5.01 /Пер. с англ.—Минск: Тивали.—1994.

**17.** *Кнут Д.* Искусство программирования для ЭВМ. Т.3. Сортировка н поиск.—М.: Мир, 1978.

**18.** *Вирт Н.* Алгоритмы и структуры данных / Пер.с англ.—М.: Мир, 1989.

**19.** *Чен П.* Модель «Сущность-связь» — шаг к единому представлению данных / Пер. с англ.// СУБД. — 1995. — № 3. — С. 137-157.

**20.** *Дейт К.Д.* Введение базы данных.// Пер. с англ, 2000.

**21.** *Уэлдон Дж.-Л.* Администрирование баз данных.—М.: Финансы и статистика,1984.

**22.** *Зиндер Е.З.* Администратор базы данных—кто он?//СУ БД. — 1995.—№2.

23. *Зегжда Д.П.. Ивашко А.М.* Как построить защищенную информационную систему/Под науч. редакц. ЗегждыД.П. н Платонова В.В.— СПб: Мир и семья-95.— 1997.

24. *Галатенко В.* Информационная безопасность //СУБД.— 1995. —№№4-6.

25. *Мельников В.В.* Защита информации в компьютерных системах. — М.: Финансы и статистика; Электроинформ, 1997.

**26.** *Ланкастер Ф.У.* Информационно-поисковые системы.—М.: Мир, 1972.

**27.** *Белоногов Г.Г., Богатырев В.И.* Автоматизированные информационно-поисковые системы.—М., 1973.

**28.** *Черный А.И.* Введение в теорию информационного поиска. — М.: Наука,1975.

**29.** *Солтон Дж.* Динамические библиотечно-информационные системы. — М.: Мир, 1979.

**30.** *Соколов А.В.* Информационно-поисковые системы. — М.: Радио и связь,1981.

**31.** *Галанский Б.Л., Поляков В. И.* Информационные системы.— Томск: Изд-во Том. ун-та, 1989.

**32.** *Venkat N. Gudivada, Vijay V. Raghavan, William I. Grosky, Rajessh Kacanagottu.* Поиск информации в World Wide Web/ COMPUTER WEEK-MOSCOW.—№ 35. — 1997.

**33.** *Мазур Л.Н.* Информационные системы: Теоретические проблемы: Учебное пособие.—Екатеринбург: Изд-воУрГУ, 1997.

**34.** *Гольдштейн С.Л., Ткаченко Т.Я.* Введение в системологию и системотехнику//ИРРО.—Екатеринбург.—1994.

# АЛФАВИТНО-ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

## СОДЕРЖАНИЕ

Учебное издание
**ГАЙДАМАКИН НИКОЛАЙ АЛЕКСАНДРОВИЧ**
**АВТОМАТИЗИРОВАННЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ, БАЗЫ И БАНКИ ДАННЫХ**
*Вводный курс*
Заведующий редакцией *Т. А. Денисова*
Корректор *Е. Н. Клитина*
Верстка С. *В. Шамринов*
Лицензия серия ЛР № 066255 от 29.12.98. Подписано к печати 17.12.01 г.
Формат 84x108/32. Гарнитура «Times New Roman Cyr». Объем 11,5 п. л.
Печать офсетная. Бумага офсетная № 1. Тираж 3 000 экз. Заказ № 2394.
Издательство «Гелиос АРВ».
107014, г. Москва, Верхняя Красносельская ул., 16. Тел./fax: (095) 264-44-39.
e-mail: info@gelios-arv.ru www. gelios-arv.ru
Отпечатано с готовых пленок в РГУП «Чебоксарская типография № 1».

428019, г. Чебоксары, пр. И. Яковлева, 15.