МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ГРАЖДАНСКОЙ АВИАЦИИ

Н.Д. Пригонюк,
В.И. Петров

ОСНОВЫ ПОСТРОЕНИЯ ЗАЩИЩЕННЫХ БАЗ ДАННЫХ

Учебное пособие

Москва
2019

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВОЗДУШНОГО ТРАНСПОРТА

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ (МГТУ ГА)»

Кафедра основ радиотехники и защиты информации

Н.Д. Пригонюк, В.И. Петров

ОСНОВЫ ПОСТРОЕНИЯ ЗАЩИЩЕННЫХ БАЗ ДАННЫХ

Учебное пособие

Утверждено Редакционно-издательским советом МГТУ ГА в качестве учебного пособия

Москва
2019

УДК
ББК 6Ф7.3
П-75

Печатается по решению редакционно-издательского совета
Московского государственного технического университета ГА

Рецензенты:
Болелов Э.А. (МГТУ ГА) – канд. техн. наук, доцент;
Иванов О.В. (ООО «ЦИБИТ») – генеральный директор

Пригонюк Н.Д.
П-75 Основы построения защищенных баз данных: Учебное пособие. /Н.Д. Пригонюк, В.И. Петров. — Воронеж: ООО «МИР», 2019. — 76 с.

ISBN

Учебное пособие издается в соответствии с рабочей программой учебной дисциплины «Основы построения защищенных баз данных» по учебному плану для студентов V курса специальности 10.05.02 всех форм обучения.

В данном учебном пособии в систематизированном виде рассматриваются теоретические вопросы построения, организации, проектирования, защиты информации и применения баз данных. Основное внимание уделяется вопросам обеспечения процессов концептуального проектирования целостных систем баз данных. Изложение рассматриваемых вопросов выполняется путем описания содержательных и формальных постановок задач, методов и результатов их решения, сопровождается большим количеством примеров и иллюстраций.

По основным разделам представлены контрольные вопросы.

Рассмотрено и одобрено на заседании кафедры 06.06.2019 г. и методического совета 06.06.2019 г.

В авторской редакции.

ББК 6Ф7.3
Св. тем. план 2019 г.
поз. 29

ПРИГОНЮК Николай Дмитриевич, ПЕТРОВ Виктор Иванович

ОСНОВЫ ПОСТРОЕНИЯ ЗАЩИЩЕННЫХ БАЗ ДАННЫХ

Учебное пособие

Подписано в печать 08.07.2019 г.
Формат 60x80/16 Печ. л. 3 Усл. печ. л. 2,79
Заказ 509/ Тираж 35 экз.

Московский государственный технический университет ГА
125993 Москва, Кронштадтский бульвар, д.20
Отпечатано ООО «МИР»
394033, г. Воронеж, Ленинский пр-т 119А, лит. Я, оф. 215

ISBN

СОДЕРЖАНИЕ

Введение

Современный уровень развития вычислительной техники, телекоммуникаций позволяет обеспечить автоматизированной информационной поддержкой практически любой вид деятельности человека. Бурное совершенствование компьютерных технологий и их массовое внедрение в процессы функционирования организационных и организационно-технических систем придают особую значимость вопросам эффективного проектирования соответствующих автоматизированных систем различного назначения, основным функциональным ядром которых являются базы данных. Понятие «база данных» (БД), появившееся впервые в научно-технической литературе в конце 60-х годов прошлого века, сегодня является одним из центральных в области современных и новейших компьютерных технологий. С ним связано рождение, становление и развитие новых наукоемких направлений профессиональной деятельности: системный анализ и проектирование бизнес-процессов и баз данных; разработка и администрирование баз данных и систем управления базами данных; построение защищенных БД, системное и прикладное программирование в области создания общего и специального программного обеспечения, а также CASE (Computer-Aided System/Software Engineering — технология автоматизированного проектирования систем/программного обеспечения) и CALS (Continuous Acquisition and Life-cycle Support — непрерывная поддержка жизненного цикла изделия) технологий.

Материалы учебного пособия объединены в семь разделов, первый из которых представляет собой введение в проблематику баз данных и содержит определение их роли и места в архитектуре АС. Здесь же определяются понятия модели данных и ее многоуровневого представления в АС, а также раскрывается основное содержание этапов жизненного цикла баз данных и взаимосвязь основных критериев оценки их качества.

Со второго раздела по седьмой материалы полностью посвящены изложению вопросов безопасности данных. Схематично изложена проблематика защиты данных, включающая в себя рассмотрение терминологии, классификацию угроз безопасности БД, требования к системе безопасности БД, разграничение доступа к БД, аудит, организацию многоуровневой защиты БД. Также здесь поясняется, что понимается под защищенностью данных, и рассматривается многоуровневая модель безопасности, целостность данных с помощью управления транзакциями, языковым средствам поддержки декларативных ограничений целостности в языке SQL, различным средствам обеспечения конфиденциальности данных и управлению транзакциями с целью восстановления БД после программных или аппаратных сбоев.

1. Базы данных и их место в архитектуре автоматизированных систем

1.1 Актуальность вопросов создания систем баз данных

Разработка программного (ПО) и информационно-лингвистического обеспечения (ИЛО) для любой автоматизированной системы (АС) является наиболее значимой составной частью процесса ее создания. Понятие «программа» и «программное обеспечение» обозначают объекты, которые отличаются друг от друга прежде всего объемами команд (операторов, строк кода). Программа состоит из сотен команд, а ПО — из десятков и сотен программ, содержит тысячи команд и занимает многие мегабайты памяти вычислительных комплексов (ВК).

Система баз данных (СБД) является ядром любой АС, так как она обеспечивает существование в памяти ЭВМ или ВК динамической информационной модели (ДИМ) или семиотической (знаковой) модели соответствующего фрагмента реального мира. Здесь под моделью понимается совокупность символов (знаков), изменяющаяся во времени (отсюда и

свойство динамичности) и описывающая состояние наблюдаемой (управляемой, проектируемой, исследуемой) системы. Модель такого класса может отражаться некоторым табло, например табло в аэропорту о вылетах и прилетах самолетов. Отмеченная центральная роль СБД в АС или в любой автоматизированной информационной системе (АИС) может быть условно представлена схемой основных процессов: ведения (обновления) СБД и поиска информации (рис.1.1).

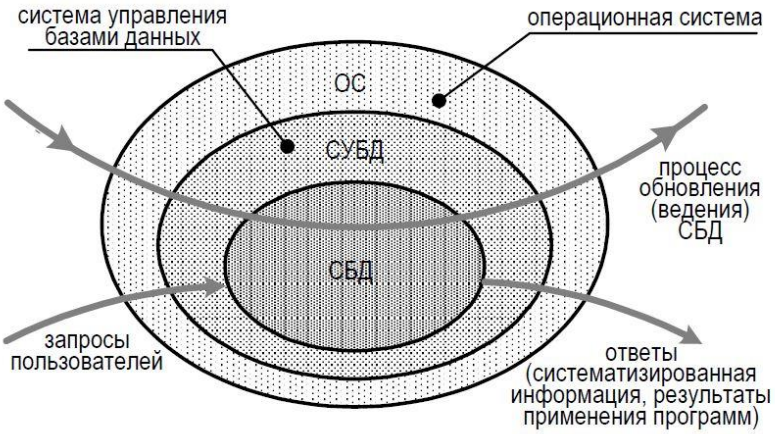

Рис.1.1. Схема основных процессов, обеспечиваемых СБД.

Все сказанное свидетельствует о том, что СБД является одной из самых важных составляющих любой АС, определяющей эффективность функционирования и требующей значительных трудозатрат для ее создания и эксплуатации.

Чтобы создавать, эксплуатировать и совершенствовать СБД, необходимо знать не только конкретные системы управления базами данных (SQL, Oracle, Paradox, Access и т.д.), но и специальные законы и правила их проектирования, информационной защиты, реализации и оценки качества.

На знаниях теоретических основ базируется новая область профессиональной деятельности человека: администрирование баз данных — деятельность по планированию, проектированию, информационной защите и эксплуатации информационных ресурсов АС.

1.2 Основные понятия общей теории систем

Рассмотрим основные понятия, связанные с развитием концепций систем баз данных.

Информационные технологии – процессы, методы поиска, сбора, хранения, обработки, предоставления, распространения информации и способы осуществления таких процессов и методов.

Информация – сведения (сообщения, данные) независимые от формы их предоставления.

Сообщение – форма предоставления информации в виде речи, текста, изображения, цифровых данных, графиков, таблиц и т.д.

Данные – информация предоставления в виде пригодном для обработки автоматизированными средствами при возможном участии человека.

Метод – (путь к чему либо) совокупность приемов или способов практического или теоретического освоения деятельности, определенным образом упорядоченная деятельность.

Принцип – (основа) первоначально исходное положение, руководящая идея в теории, науке и политике, основное правило деятельности, поведения и т.д.

Связь – это то, что связывает элементы между собой.

Информационная система – есть совокупность содержащейся в базе данных информации и обеспечивающих ее обработку информационных технологий и технических средств.

База данных (БД) – поименованная совокупность структурированных данных, относящихся к определенной предметной области.

БД используются для хранения информации об объектах какой-либо предметной области.

Предметная область – часть реального мира, подлежащая изучению с целью автоматизации. Предметную область можно представить как множество взаимосвязанных объектов.

Объект (сущность) - это выделенный элемент предметной области, подлежащий хранению в БД. Другими словами - это «нечто, о чем мы хотим хранить информацию в БД». Объект может быть реальным (человек, населенный пункт, какой-либо предмет) и абстрактным (событие, счет покупателя и т.д.).

Для каждого объекта выделяют набор признаков (характеристик, свойств или атрибутов) которые позволяют описать объект в рамках выбранной предметной области.

Если рассматривать человека как объект, о котором мы хотим хранить информацию в БД, то можно заметить что для предметных областей связанных с медициной наиболее значимыми наборами характеристик человека могут оказаться: рост, вес, пол и т.д. Для производства набор значимых характеристик человека иной: возраст, должность, рейтинг и т.д.

Далее будем широко использовать два основных понятия из общей теории систем: элемент и система.

Элемент — это минимальный, неделимый объект, рассматриваемый как единое целое при решении конкретной задачи.

Неделимость элемента представляет собой лишь удобное допущение, но не физическое свойство. Переход от одной задачи к другой может потребовать либо разложения элементов на составные части (элементы), либо их объединения в единое целое (элемент).

Например, для разработчика автоматизированной системы ЭВМ, рабочие места пользователей, сервер баз данных, сервер связи являются элементами, а для разработчика самой ЭВМ она представляется набором элементов (центральный процессор, память, канал и т.д.).

Система (греч. «целое, составленное из частей», «соединение») — множество взаимосвязанных друг с другом элементов, обладающее целостностью и структурой и связанное с некоторой «средой».

Это не просто набор элементов, а взаимосвязанная их совокупность, объединенная единством цели существования, правилами взаимоотношений и обладающая некоторым ценным качеством, не присущим отдельным элементам.

Системы, содержащие в своем составе хотя бы один решающий механизм, называются сложными. В противном случае — простыми.

Рассмотрим наиболее важные свойства систем.

Целостность — свойство элементов и их взаимосвязей соответствовать (удовлетворять) законам и правилам системы.

Структура — состав элементов и связей между ними. Описывает наиболее консервативную сторону системы, состоящую в относительно устойчивых пространственно-временных связях, определяющих функциональное назначение системы.

Формально любая система S имеет свои входы и выходы, которые обладают различной физической сутью (рис.1.2).

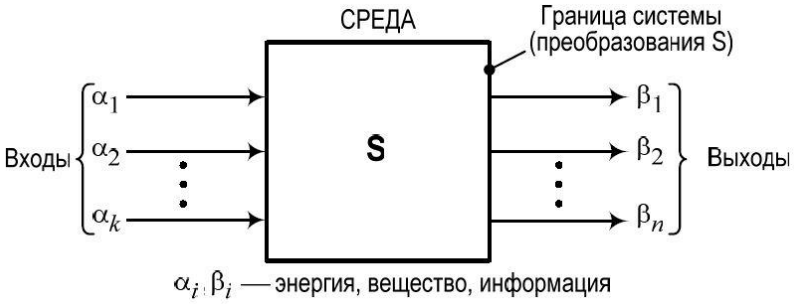

Рис.1.2. Входы и выходы системы.

Определение структуры системы и ее границ представляет собой довольно сложный процесс, который носит итерационный характер и связан с устранением двух типов ошибок: учет второстепенных и неучет существенных связей. При этом реальный мир делится на систему и среду, которая влияет на состояние и поведение системы. Совокупность взаимосвязей между элементами системы определяет ее организационную сложность.

Сложность системы определяется количеством ее элементов, связей, возможных реакций на воздействия среды (т.е. множеством возможных состояний и поведением).

Состояние системы определяется совокупностью значений характеристик, описывающих элементы и связи системы в конкретный i-й момент времени. Такая совокупность характеристик называется вектором состояния системы и обозначается:

$X_i=\langle x_1,x_2,...,x_n\rangle$, где x_j — наименование j-й компоненты вектора состояния.

Например, пусть задана система «сеть ЭВМ» (рис.1.3). Такая система может описываться, например, следующим вектором состояния: $X=\langle x_{11},x_{12},x_{21},x_{22},x_{31},x_{32},y_{11},y_{12},y_{21},y_{22},y_{31},y_{32}\rangle$, где x_{i1} — быстродействие i-й ЭВМ (целое, оп/с), x_{i2} — свободная оперативная память i-й ЭВМ (целое, кб), y_{i1} — быстродействие i-го канала (целое, бит/с), y_{i2} — состояние i-го канала (символьное, з — занят, с — свободен).

Поведение системы — упорядоченная во времени последовательность значений вектора состояния $X^1,X^2,\ldots,X^k$, где X^i — значение вектора состояния в i-й момент времени.

Решающие механизмы в сложных системах осуществляют акты принятия решений путем выбора альтернатив из множества возможных решений. Этот выбор может быть случайным или интеллектуальным.

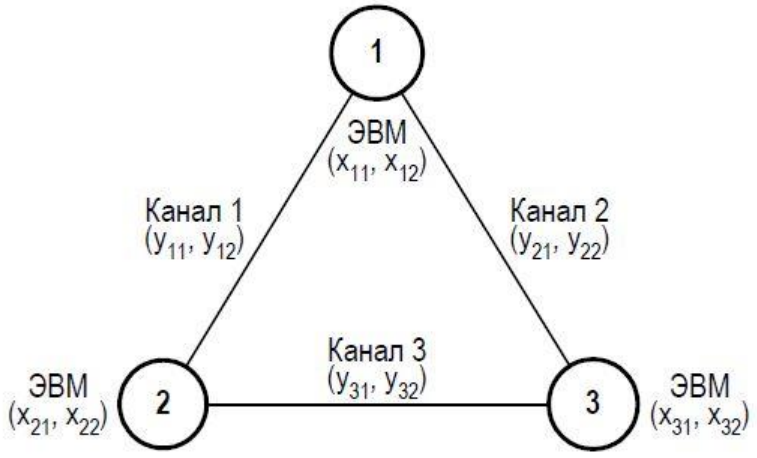

Рис.1.3. Система — сеть ЭВМ.

Сложные системы обладают свойством стремления к достижению своего предпочтительного состояния, называемого целью поведения, т.е. сложные системы обладают целенаправленным поведением за счет наличия решающих механизмов. Такое поведение обеспечивается выбором решений, опирающимся на опыт существования системы, зафиксированный в ее памяти (в биологических системах — генетическая память, в организационных — опыт и знания людей). Сложные системы дополнительно характеризуются следующими свойствами:

устойчивостью, т.е. способностью к существованию, которая может реализовываться даже за счет обмена со средой;

помехоустойчивостью, т.е. способностью системы противостоять воздействиям среды, мешающим ее целенаправленному поведению;

самоорганизацией, т.е. способностью системы к восстановлению своего целенаправленного поведения, разрушенного в результате воздействия среды.

Управляемость системы определяет ее способность к изменению своего состояния под воздействием решающего механизма.

Действия системы, связанные с достижением целей, называются функциями, а порядок выполнения действий — функционированием.

1.3 Архитектура автоматизированного банка данных

Автоматизированный банк данных – совокупность баз данных, программных, языковых, технических и др. средств, предназначенных для централизованного накопления и коллективного многоцелевого использования данных.

Место автоматизированного банка данных (АБнД) в АС, основные его компоненты и их взаимосвязь представлены на рис.1.4.

Как было отмечено выше, АБнД представляет собой организационно-техническую систему (ОТС), включающую информационные, математические, программные, языковые и технические средства, а также персонал, обеспечивающие накопление и коллективное многоаспектное использование данных для решения прикладных задач различных пользователей той или иной предметной области. Под предметной областью, в общем случае, понимается любая система операций большого масштаба в области экономики, социальных отношений или другого направления. АБнД обеспечивает создание и поддержку актуального состояния динамической информационной модели (ДИМ) предметной области (в которой он создан) и взаимодействие с нею всех категорий пользователей (лиц, принимающих решения; подсистем; задач и т.д.). Такое взаимодействие осуществляется с помощью языков описания и манипулирования данными (ЯОД и ЯМД), являющихся составной частью любой современной системы управления базами данных (СУБД).

В общем случае информационной моделью будем называть совокупность сведений о состоянии и развитии процессов функционирующей ОТС, структурированных с помощью формальных средств (называемых моделью данных) и запомненных в памяти ЭВМ. Эти сведения представляются значениями характеристик, включенных в вектор состояния системы (объекта управления) и называемых атрибутами (элементами данных). Значения атрибутов хранятся в памяти ЭВМ (скорость полета, высота полета, ФИО командира экипажа и т.п.) и могут представляться числами, символами, кодами, а их постоянное обновление обеспечивает свойство динамичности модели. Таким образом, ДИМ есть изменяемая во времени совокупность значений некоторого состава характеристик (атрибутов, показателей, реквизитов), определенного соответствующими задачами (приложениями) и относящегося к сущностям и их взаимосвязям, контролируемого (наблюдаемого, исследуемого) реального мира.

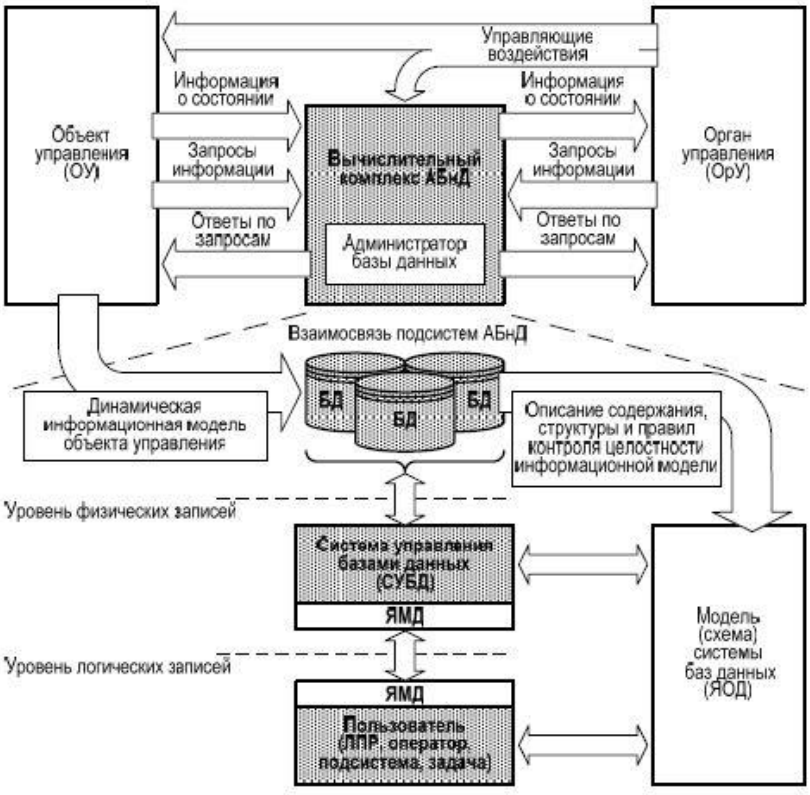

Рис.1.4. Место АБнД в АС и его архитектура.

Такая изменяющаяся во времени совокупность значений взаимосвязанных типов характеристик называется базой данных (БД). В этом понятии сосредоточена основная суть системного подхода к организации информационных процессов и решению прикладных функциональных задач.

Для создания БД необходимо выявить информационные потребности всех прикладных задач в виде необходимых им элементов данных и объединить эти элементы в систему для реализации процедур обновления и поиска их значений.

Сущностями реального мира, информация о которых хранится в базах данных, могут являться:

— объекты (пункты управления, сооружения, самолеты, материальные средства, документы и т.п.);

— явления природы (погода, наводнения, ураганы и т.д.);

— абстрактные понятия (экипаж, полет, воздушная обстановка, скорость и т.д.);

— конкретные люди (летчики, обслуживающий персонал, работники предприятия, операторы и т.д.).

Принцип системности в организации БД иллюстрируется схемами (рис.1.5), демонстрирующими суть объединения данных всех задач должностных лиц некоторой предметной области.

При позадачном подходе данные каждой задачи существуют независимо друг от друга, а их дублирование (места пересечений) часто приводит к противоречиям из-за асинхронности процедур обновления. Интеграция в единую БД позволяет устранить возникновение подобных противоречий за счет единства места накопления и централизации процедур обновления. Пользователям выдаются копии одних и тех же данных. Все функции по управлению данными осуществляет специальная группа лиц, называемая администратором

банка данных (АБнД) и включающая администраторов баз данных. Эти люди отвечают за эффективность функционирования АБнД.

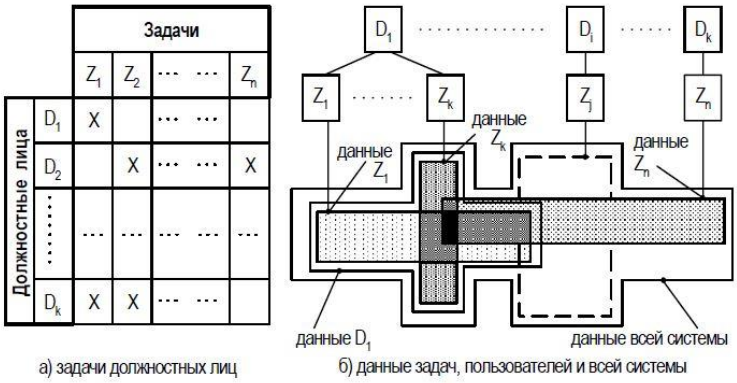

Рис.1.5. База данных для всех задач.

Разработка каждой СУБД непосредственно связана с той или иной моделью данных и соответствующим ЯОД, используя который АБнД создает схему базы данных. Схема БД сообщается пользователю и СУБД и является основой для функционирования АБнД.

1.4. Классификация моделей данных

Далее будем использовать следующее основное определение.

Определение 1.1. Совокупность соглашений о способе и средствах формализованного описания сущностей и их связей, имеющих отношение к автоматизируемым процессам предметной области, будем называть моделью данных.

Установим соответствия основных понятий реального мира и модели данных так, как это показано на рис.1.6.

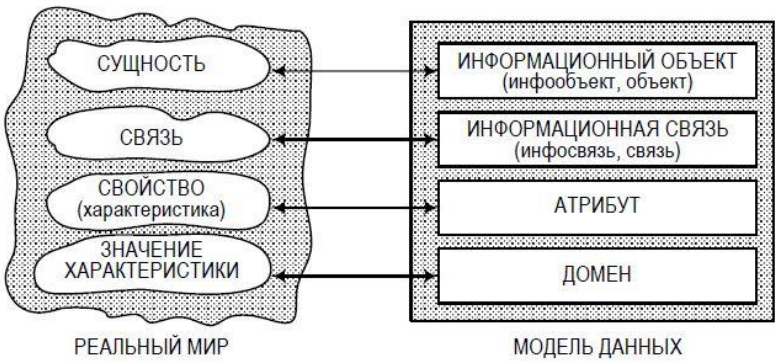

Рис.1.6. Взаимосвязь понятий реального мира и модели данных.

Модель данных будем представлять в виде тройки:

$$M=\langle S, P, Q\rangle,$$

где S — структура модели (типы элементов, связей и их значений); P — множество операций над экземплярами S (правила интерпретации и манипулирования); Q — множество правил (условий, ограничений) целостности модели, определяющих совокупность допустимых состояний (значений) экземпляров S.

Схема базы данных представляет собой модель данных M, записанную на ЯОД конкретной СУБД. Заметим, что в действительности каждая СУБД поддерживает некоторую

свою собственную (свойственную только ей) модель данных. Тем не менее, по типу структуры S все используемые модели данных могут быть разбиты на три класса: иерархические, сетевые и реляционные. Так же классифицируются и СУБД. Вообще говоря, классификация модели данных может рассматриваться с различных точек зрения, которые представляются таблицей:

Основание деления (точка зрения)	Наименование типа модели
По типу структуры	— иерархическая — сетевая — реляционная
По уровню использования при проектировании	— инфологическая — концептуальная — логическая — физическая
По уровню использования при эксплуатации	— внешняя — системно-логическая — внутренняя

В дальнейшем мы будем оперировать понятиями, связанными с первыми двумя точками зрения. Основное и главное отличие трех типов структур моделей данных (иерархической, сетевой и реляционной) состоит в способе представления и манипулирования инфосвязями.

1.4.1 Иерархическая модель

Иерархическая модель данных определяется схемой, в которой объекты предметной области упорядочены с помощью одной или нескольких иерархических структур.

Отношения в иерархической модели данных организованы в виде совокупностей деревьев (основа – граф типа «дерево»), где дерево - структура данных, в которой тип сегмента потомка связан только с одним типом сегмента предка. Графически: Предок - узел на конце стрелки, а Потомок - узел на острие стрелки (рис. 1.7). В базах данных определено, что узлы - это типы записей, а стрелки представляют отношения один - к - одному или один - ко - многим.

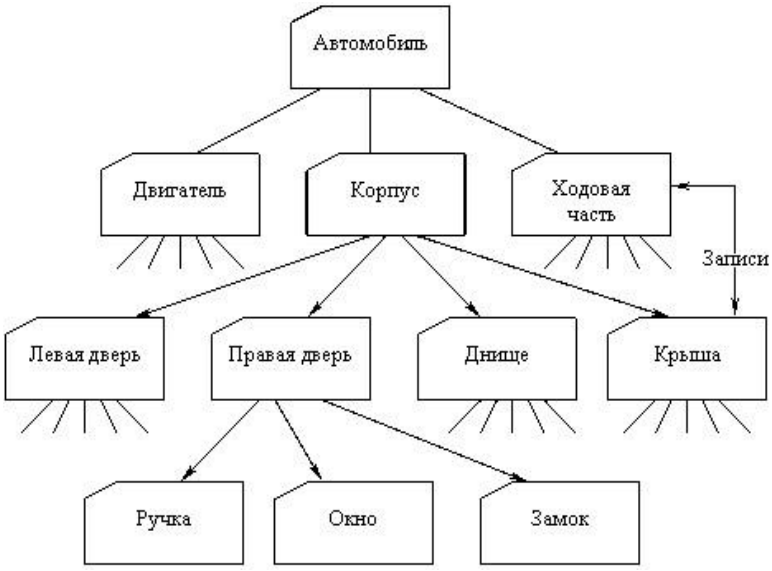

Рис. 1.7. Иерархическая база данных.

Основное преимущество данной модели состоит в естественности и простоте моделирования иерархических структур реального мира.

Недостатки модели состоят в следующем:

— невозможность существования подчиненных сегментов при отсутствии родительских (необходимо вводить фиктивные экземпляры);

— степень сложности поиска данных зависит от глубины их размещения в иерархии и определяется количеством операций сравнения, реализуемых соответствующими алгоритмами;

— изменение иерархии требует изменения обрабатывающих программ (слабая логическая независимость данных);

— значительные трудности при обновлении БД (при замене можно потерять уникальные данные, при добавлении иногда требуется вводить фиктивные сегменты, при исправлении требуется осуществлять просмотр всей базы данных);

— поиск данных требует указания всего пути по иерархии.

1.4.2 Сетевая модель

Сетевая модель данных определяется схемой, в которой объекты предметной области образуют узлы сети, а связи между ними описывают различные типы «M:N» отношений (рис. 1.8). Это делает сферу применения данной модели очень широкой и избавляет ее от ряда недостатков, присущих иерархической модели.

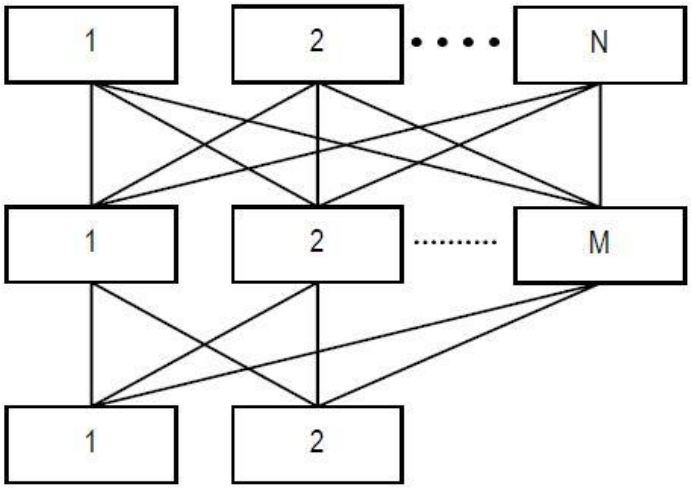

Рис.1.8. Схема сетевой БД.

Основные положительные качества этой модели по сравнению с иерархической состоят в следующем:

— сфера применения модели практически не ограничена;

— сложность поиска данных по симметричным запросам одинакова.

Недостатки сетевой модели состоят в следующем:

— сложность и чрезвычайная близость к способам физической организации данных (пользователь в деталях должен знать все цепочки и как они реализуются);

— при больших объемах для формирования запросов существуют значительные трудности для явного указания путей доступа к данным;

— высокая сложность выполнения операций обновления и контроля целостности базы данных.

1.4.3 Реляционная модель

Реляционная модель данных (или модель данных в виде отношений, relation — отношение) использует для представления сущностей реального мира и их связей только один тип элемента, называемый отношением (таблицей).

В том случае, когда реляционная модель включает несколько таблиц, ее целесообразно дополнять двумя схемами связей отношений, одна из которых определяет условия согласованного и целостного ведения БД (поддержания актуального состояния БД посредством операций обновления), а другая — допустимые операции соединения отношений.

Разработка реляционной модели данных была осуществлена Э.Ф. Коддом (E.F. Codd) на основе теоретического обобщения практического опыта применения иерархических и сетевых моделей, при этом, как выражался автор, предложенная им модель избавляет разработчика БД от «суеты представления». Это означает, что наиболее простым и естественным способом представления модели любой части реального мира является ее задание в виде конечного множества отношений (таблиц). По своей структуре все отношения представляют собой однотипные элементы, каждый из которых имеет вид таблицы с уникальным именем R_i и заголовком в виде наименований столбцов (имен атрибутов), а также строк (записей, кортежей), состоящих из значений, соответствующих наименованиям столбцов.

На отношениях как на элементах множества Э.Ф.Кодд определил операции (селекцию, проекцию, соединение и др.), которые позволяли получать новые отношения путем применения этих операций в определенной последовательности к известным отношениям. Получаемые таким способом новые отношения представляют собой ответы на некоторые запросы данных, а последовательность операций — соответствующие алгоритмы поиска данных (формирования ответов на запросы).

Множество отношений и операций над ними получили название реляционной алгебры, а правила построения формул над именами отношений и атрибутов стали называться языком манипулирования данными реляционной модели, основанным на алгебре. Таким образом, реляционная модель (в отличие от иерархической и сетевой) не ориентирована на конкретные типы запросов; она позволяет с одинаковой степенью сложности программировать с помощью алгебраических формул процедуры поиска данных на любые, заранее неизвестные, запросы. Языки запросов, основанные на алгебре, получили название процедурных, так как они обеспечивают точное описание требуемых пользователю последовательностей операций (процедур) над известными ему отношениями.

Реляционная модель данных открыла также второй путь для построения языков манипулирования данными: непроцедурный (декларативный) путь, основанный на функциональной (операторной) связи между формулами алгебры и исчисления отношений (предикатов). Такие языки не требуют от пользователя описывать процедуры поиска в виде последовательностей операций, а позволяют точно формулировать требуемые основные свойства искомых элементов (отношений). Эти языки послужили основой для разработки языков манипулирования данными, близких к естественным, проблематика разработки которых является одним из направлений теории систем искусственного интеллекта.

Положительные качества реляционной модели состоят в следующем:

— простота, ясность и однородность описаний объектов и связей;

— языки запросов обеспечивают работу не с отдельными значениями, а с множествами атрибутов и их значений;

— высокая степень независимости описаний схемы и запросов от физической организации данных;

— возможность непроцедурного (декларативного) описания запросов;

— возможность простой трансформации в иерархическую или сетевую модель.

Отрицательных качеств у реляционной модели практически нет.

Некоторые специалисты указывают на низкую скорость (реактивность) реляционных СУБД, но этот недостаток сегодня успешно преодолевается, так как он связан не со свойствами модели, а с уровнем развития элементной базы ЭВМ и их программного обеспечения (операционных систем, СУБД).

1.5 Многоуровневое представление данных в АС

Основой функционирования любой АС является многоуровневая организация представления данных, принятая в современных системах электронной обработки информации (рис.1.9).

Трехуровневая система представления данных (третий столбец таблицы) была впервые обоснована в отчете Исследовательской группы по СУБД Американского национального института стандартов (ANSI-SPARC Stady Group on DBMS) и впоследствии обсуждалась и развивалась в большом числе работ по базам данных.

Представление данных ANSI-SPARC было ориентировано в основном на процессы оперирования (манипулирования) данными с помощью использования одной из универсальных СУБД, обеспечивающих широкий спектр возможностей по ведению и поиску данных по запросам пользователей.

Развитие этой идеи состояло в разработке четырехуровневой схемы представления данных (четвертый столбец таблицы), обеспечивающей хорошую основу как для создания, так и для использования СБД. Описание модели данных с помощью языков описания данных (ЯОД) называют схемой данных. Схемы данных, как правило, содержат определение наименований атрибутов, правил их взаимосвязи, означивания, целостности и согласованности. В зависимости от уровня описания схема содержит в себе различный объем сведений, связанных с элементами операционных систем, СУБД, устройств памяти и ЭВМ. Схемы различных уровней и описание их взаимосвязей (отображение схемы одного уровня в схему другого уровня) разрабатываются в ходе проектирования информационного обеспечения АС и используются в процессе формирования процедур поиска и выдачи данных, требуемых в запросах пользователей.

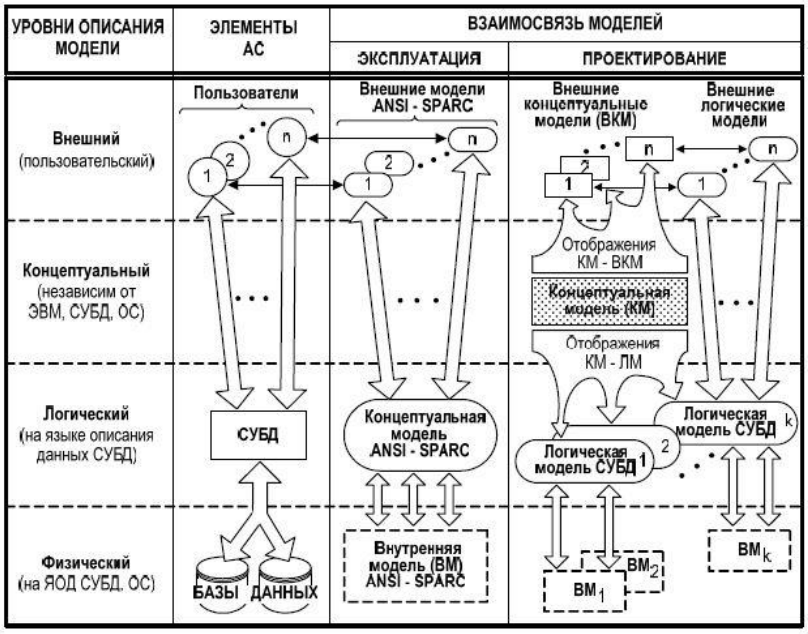

Рис.1.9. Многоуровневое представление данных в АС.

Внешний уровень представления наиболее близок конечным пользователям (пользователям-непрограммистам) и содержит определение данных в привычном для них виде: таблицы, графики, документы, карты и т.п. СУБД должна быть в состоянии, с одной стороны, обеспечить пользователей удобным для них представлением данных, а с другой — воспринимать запросы на информационный поиск в терминах такого представления. Схемы внешнего уровня часто называют подсхемами.

Концептуальный уровень связан с обобщенным проблемно-ориентированным и технико-независимым (т.е. независимым от конкретных типов СУБД, операционных систем и ЭВМ) представлением всех данных в АС. Это уровень деятельности администратора системы баз данных. Средства (язык), используемые для описания данных на этом уровне, должны быть просты для понимания конечными пользователями, а также точны и полны для формулирования логического представления.

Логический уровень будем соотносить с конкретным типом СУБД (в терминологии ANSI-SPARC этот уровень называется концептуальным). Логическим представлением будем называть запись концептуальной схемы с помощью ЯОД конкретной СУБД, ориентированной на определенную логическую структуру (дерево, сеть, отношение). Этот уровень соответствует области деятельности прикладных программистов и администратора БД.

Физический уровень описания (внутреннее представление) данных наиболее близок к физической памяти вычислительных систем; он содержит описание размещения и организации данных с учетом конкретных типов запоминающих устройств, операционных систем и методов доступа. Это уровень деятельности системных программистов.

Проводя аналогию с процессом разработки программ, можно сказать, что выполнение содержательной формулировки задачи соответствует определению внешнего представления данных, разработка метода и алгоритма решения задачи соответствует разработке концептуального представления. Запись алгоритма на конкретном языке программирования соответствует разработке логического представления, а коды машинных команд — внутреннему размещению данных (рис.1.10).

В большинстве существующих СУБД нет достаточно четкого разделения между средствами описания, предлагаемыми для внешнего, концептуального и логического уровней, а также для описания операций манипулирования данными. Как правило, эти потребности реализуются одним языком конкретной СУБД. Однако при проектировании системы, ее дальнейшей эксплуатации и развитии необходимы специальные средства для создания устойчивого технико-независимого описания данных как на внешнем, так и на концептуальном уровне.

Концептуальная модель (КМ) должна обладать полнотой, простотой, точностью и ясностью для конечных пользователей и для АБнД. На основе концептуальной модели АБнД разрабатывает внешние модели и способы их поддержки, а также систему отображений КМ на логический (Мл) и физический (Мф) уровни. Для систем распределенных БД также необходимо разрабатывать единую концепцию, которая должна играть роль фундамента для выбора наиболее обоснованного варианта распределения информационных ресурсов по узлам сети ЭВМ (определение локальных концептуальных моделей) и определения способов их взаимосвязи и согласования.

1.6 Основные элементы физической организации СБД

Вопросы физической организации данных в вычислительных системах имеют значительные объем и сложность.

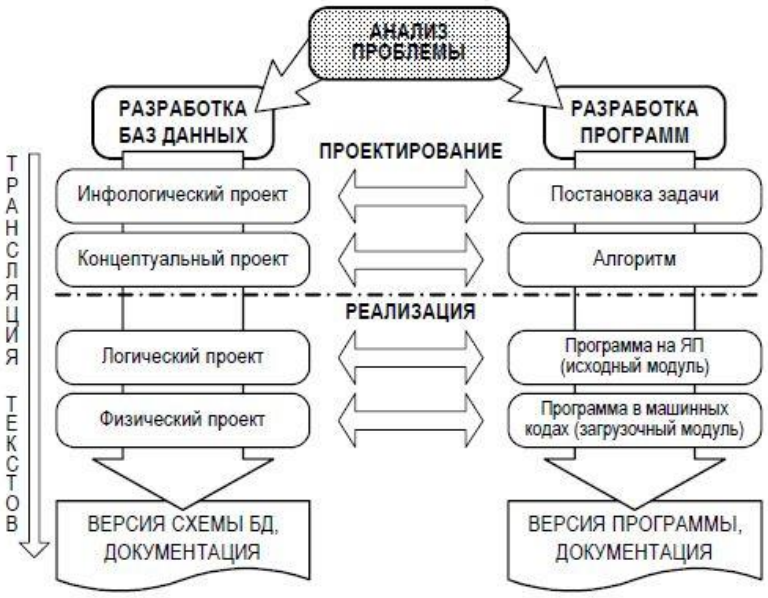

Рис.1.10. Обобщенная схема процесса проектирования программного комплекса.

Основными критериями для оценки качества физической организации данных являются:

— время реакции (ответа) системы на запрос;

— объемы используемой памяти.

Решение проблемы разработки физической организации данных, обеспечивающей одну из выбранных логических схем (иерархическую, сетевую или реляционную), связано с необходимостью решения ряда частных задач, среди которых основными можно считать следующие:

— как найти необходимую запись среди множества других (проблема адресации)?

— как организовать данные, чтобы время их поиска было минимальным (проблема поиска)?

— как организовывать древовидные и сетевые структуры в виде последовательностей записей (проблема представления)?

— как добавлять и исключать записи, не нарушая систем адресации и поиска (проблема ведения)?

— как обеспечить максимальную плотность записи (проблема сжатия данных)?

При разработке физической организации данных требуется разрешать множество противоречивых ситуаций:

— повышение плотности данных (экономия памяти) влечет за собой увеличение времени на адресацию и поиск данных;

— сокращение времени выбора данных требует применять более дорогие запоминающие устройства (ЗУ);

— обеспечение гибкости (разноаспектности) поиска данных, как правило, приводит к увеличению объема необходимой памяти и времени доступа и т.д.

В целом для достижения высокой эффективности (минимального времени реакции и оптимального использования памяти), как правило, необходимо применять сложные методы организации, которые иногда могут приводить к высокой степени сложности восстановления системы после сбоев и аварий. Все эти проблемы скрыты от пользователей СБД и являются сферой ответственности администрации СБД. Кратко рассмотрим существо некоторых из перечисленных задач и связанные с ними вопросы физической организации.

1.6.1 Основные понятия

Блокирование логических записей представляет собой объединение нескольких экземпляров логических записей в одну физическую запись, считываемую одной командой READ. После этого физическая запись деблокируется и прикладной программе передается требуемый ей экземпляр логической записи. Блокирование логических записей применяется для экономии памяти: так как физические записи разделяются промежутками, то таких промежутков было бы очень много в том случае, когда физическая запись совпадала бы с логической (рис.1.11). Различные экземпляры логических записей в рамках одной физической записи могут иметь различную длину.

Рис.1.11. Блокирование логических записей.

Последовательность физических записей может отличаться от логического представления их последовательности. Физическая последовательность выбирается из условий сокращения времени обработки (чтения) записей. На носителе с последовательным доступом (например, на магнитной ленте) записи размещаются в такой последовательности, чтобы переходов от одной части ленты к другой (без чтения) было бы минимальное количество (холостые перемотки). На носителях с прямым доступом (магнитные диски) записи могут читаться в любой последовательности (как на грампластинке). Но размещать записи следует таким образом, чтобы при их чтении механизм доступа (головка) не выполнял бы больших перемещений без считывания данных, а количество операций смены физических секций (томов) дисков или дискет было бы минимальным.

Параллельная секционная организация физических записей позволяет существенно сократить время чтения нескольких записей, когда имеется несколько механизмов доступа (головок чтения-записи) к секциям памяти, подключенных к одному каналу чтения (рис.1.12).

При параллельном чтении головки записи-чтения устанавливаются одновременно (параллельно), а записи читаются по готовности головки и канала. Иногда при размещении данных учитывается частота их использования: часто используемые данные размещаются в быстрой памяти (интегральные схемы, диски), редко используемые — в медленной (ленты). При размещении данных на дисках часто используемые данные запоминаются на внутренних цилиндрах дисков, редко используемые — на внешних, средне используемые — на средних.

1.6.2 Адресация логических записей

Экземпляры логических записей размещаются в физической памяти по байтам, имеющим свои машинные адреса. Эти адреса вычисляются с помощью специальных программ, использующих значения ключевых атрибутов записи (первичных ключей). При этом значение адреса (y) некоторой записи в физической памяти ЭВМ является функцией (f) от значения первичного ключа (x) логической записи, т.е. y=(x). Иногда требуется

использовать значения составного или сцепленного ключа, состоящего из двух y=f(x,z) и более значений различных атрибутов y=f(x,z,...).

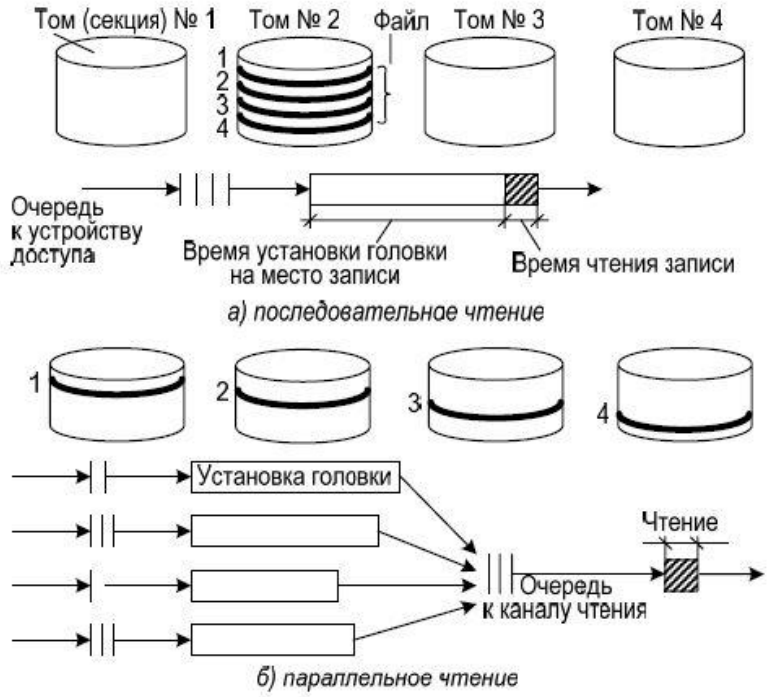

Рис.1.12. Последовательная и параллельная секционная организация физических записей.

На практике существует несколько способов реализации таких функций адресации записей, т.е. способов определения по значению первичного ключа местоположения (адреса) экземпляра записи с этим значением ключа. Подобные операции определения адреса выполняются при запоминании (добавлении) записи и при ее поиске (для чтения или удаления). Фактически это одна и та же операция, но способы ее выполнения отличаются именно процедурами добавления и исключения.

Последовательное размещение. Экземпляры записей размещаются один за другим независимо от значений ключа. При чтении считывается каждая запись и проверяется требуемое значение ключа. Такой последовательный просмотр называют сканированием памяти.

Часто применяются процедуры сортировки записей по убыванию или возрастанию значения ключа. Этот способ для большинства случаев требует много времени и может быть эффективен только при пакетной обработке.

Блочное размещение (с пропусками) записей и блочный поиск. Записи (рис.1.13), упорядоченные по значению ключа, сгруппированы в блоки фиксированной длины, например по 100 записей в блоке. При поиске нужной записи выполняется сравнение ключевых значений записей, стоящих в концах блоков: $K_i<K_n$, $K_i<K_{2n}$ и т.д., где K_i — искомое значение ключа, K_n — значение ключа последней записи в первом блоке. При выполнении условия поиска просматривается не более (n−1)-й записи из данного блока. Если запись с искомым значением ключа отсутствует, то по окончании поиска дается соответствующий ответ. При добавлении записи делаются вставка в соответствующий блок и его сортировка. Существует много способов для выполнения такой процедуры.

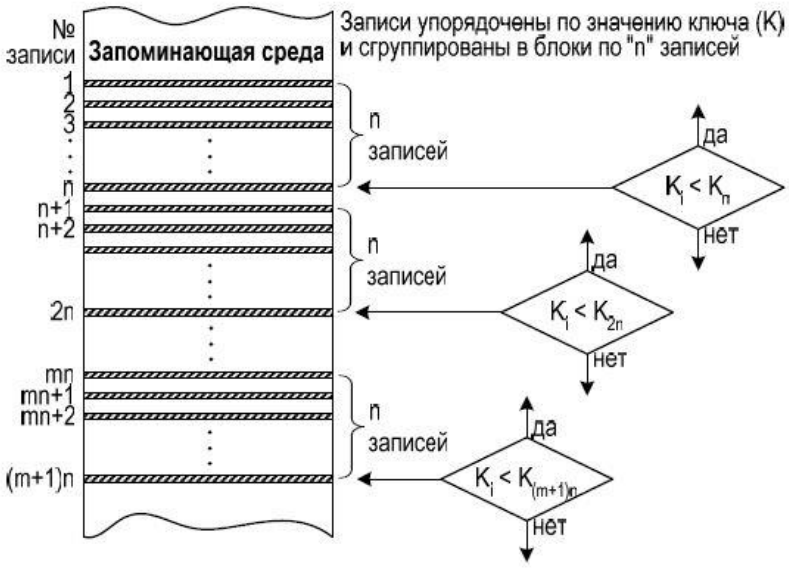

Рис.1.13. Блочное размещение записей.

Двоичный поиск (размещение) является разновидностью предыдущей процедуры (рис.1.14). При этом рассматривается значение ключа записи, находящейся в середине области поиска, и сравнивается с требуемым (искомым, записываемым). Затем следующая область делится пополам и т.д. Этот способ применяется для устройств памяти на интегральных схемах и не применяется в тех случаях, когда требуется перемещать механизм доступа.

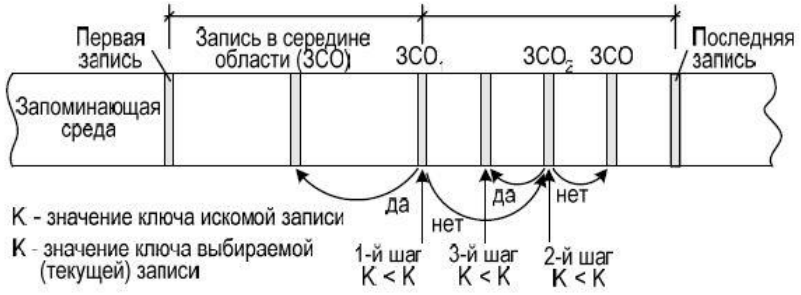

Рис.1.14. Двоичный поиск в массиве записей.

Индексно-последовательный поиск (организация, размещение). Обычно сканирование всего массива записей требует много времени, и такой последовательный просмотр применяют для определения необходимой записи в небольшой области, которая, в свою очередь, определяется с помощью индекса (одного или нескольких уровней).

Индекс — это специальная таблица, входом в которую является значение ключевого атрибута (ключа), а выходом — относительный или абсолютный адрес записи в памяти ЭВМ. Если записи упорядочены по значениям ключа, то индекс имеет более простую структуру и объем по сравнению с индексом для произвольного (неупорядоченного) размещения записей в основном массиве (рис.1.15). Индекс для упорядоченных массивов содержит указание адреса не каждой записи, а блока записей, которые необходимо сканировать. Но даже в этом случае индекс может быть очень большим для сканирования, тогда для него создается свой индекс, т.е. индекс индекса (индекс 2-го уровня). Поля записей, для которых строятся индексы, не должны иметь значений, повторяющихся в других записях, т.е. эти значения

должны быть уникальными в пределах рассматриваемой совокупности записей (одного массива, файла).

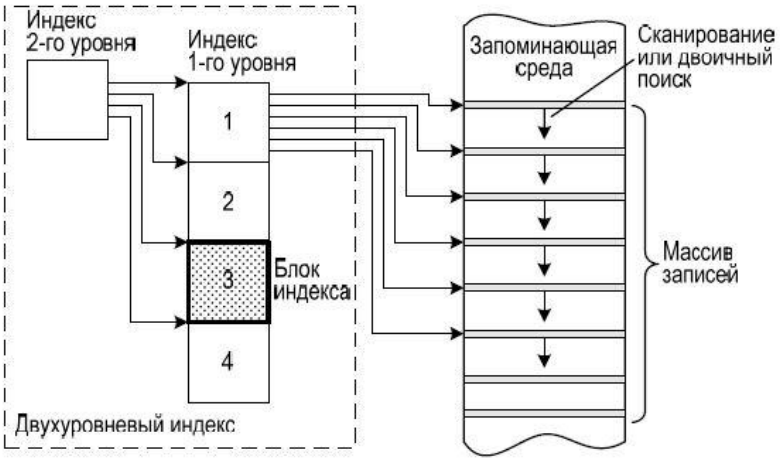

Рис.1.15. Индексно-последовательная организация.

Поиск данных с помощью индексов позволяет значительно сокращать время обработки. Если требуется осуществлять поиск по значениям неключевых полей, по которым индексы отсутствуют, то для ускорения такой обработки применяют инвертированные файлы. Такие файлы позволяют по значению некоторого неключевого атрибута (поля) получать значения ключевых полей всех записей, в которых существует это неключевое значение.

Преобразование ключа в адрес. Наиболее известными сегодня являются следующие три вида такой адресации, которая иногда называется прямой:

— значение ключа непосредственно используется как адрес записи (этот способ самый быстрый, но не всегда возможен);

— адрес является значением некоторой очень простой функции от значения ключа. Этот способ по быстроте почти такой же, как первый, и применяется иногда в два этапа: по значению ключа записи определяется ее уникальный порядковый номер, по которому определяется адрес в массиве (при этом в массиве записей могут быть свободные места);

— перемешивание (рандомизация, хеширование — hashing). В данном способе значение ключа (рис.1.16) преобразуется специальной программой в квазислучайное целое число, называемое номером участка, которое преобразуется в адрес физической записи.

Если в запоминающей среде есть место, то логическая запись записывается на это свободное место, если места нет, то — в зону переполнения, которая при необходимости сканируется. Этот метод сопоставим по эффективности с индексным. Плотность заполнения запоминающей среды до 80…90%. Большинство записей можно найти за одно обращение к программе. На практике существует большое разнообразие алгоритмов для рассмотренных процедур, а также комбинации рассмотренных способов адресации.

Рис.1.16. Перемешивание записей.

1.6.3 Поиск данных по нескольким ключам

До сих пор, применяя понятие ключ, мы подразумевали значение некоторого поля записи, которое однозначно идентифицирует эту запись и используется для определения ее местоположения в памяти ЭВМ. Значения ключей могут также использоваться для создания индексов. На практике часто возникает потребность поиска записей по различным полям и их комбинациям, значения которых не идентифицируют одну конкретную запись, а указывают на некоторое множество записей. Такие поля и их комбинации называются вторичными ключами и используются для организации вторичных индексов (или инвертированных файлов).

В качестве вторичных ключей применяются комбинации полей, значения которых наиболее часто используются для поиска информации. При этом ускорение поиска за счет использования вторичных ключей обеспечивается дополнительными расходами памяти ЭВМ для организации вторичных индексов.

Организация и ведение системы индексов связаны с разнообразными способами представления записей на физическом уровне.

Отметим, что понятие «ключ записи» имеет помимо поискового аспекта и другой аспект, связанный с контролем целостности БД.

1.6.4 Виртуальная память и иерархическая организация памяти

Технические и программные средства ЭВМ сегодня позволяют представлять и использовать основную (быструю) память ЭВМ не в том объеме, в котором она физически существует, а в объеме, сопоставимом с емкостью всей внешней памяти ЭВМ. В этом смысл виртуальности (кажущегося существования) бесконечных объемов основной оперативной памяти.

Такой эффект достигается за счет разбиения всего объема памяти на страницы, размещаемые на внешних запоминающих устройствах (ЗУ). При обращении программы к отсутствующей странице операционная система осуществляет прерывание ее выполнения и с большой скоростью перемещает требуемую страницу в основную память, после чего продолжается выполнение прерванной программы. Операционная система (ОС) осуществляет контроль и управление процессом распределения страниц памяти между основной памятью и внешними ЗУ. Такая идея непосредственно связана с принципом иерархической организации памяти, реализация которого на практике стала возможной при появлении устройств с прямым доступом.

Иерархическая система памяти строится следующим образом: на нижних уровнях системы размещаются бóльшие объемы, имеющие меньшую стоимость хранения одного байта данных и меньшее быстродействие, чем устройства верхних уровней (рис.1.17). Поиск запрашиваемых программой данных осуществляется по уровням иерархии сверху вниз. Обычно используют от двух до пяти уровней иерархии.

Уровень памяти	Емкость (байт)	Стоимость (у.е./байт)	Среднее время доступа, мкс
Регистры, быстродействующая буферная память (кэш), основная память на интегральных схемах	$n \times 10^6$	10	0,5
Память большой емкости на магнитных сердечниках	$n \times 10^7$	1	5
Барабаны, диски	$n \times 10^9$	0,01	10^5
Архивные или библиотечные ЗУ, подключенные к ЭВМ	$n \times 10^{12}$	0,0001	10^7

Рис.1.17. Характеристики иерархической системы памяти.

1.7 Структура процесса проектирования СБД и критерии качества

В широком смысле проектирование баз данных АС представляет собой процесс выработки и документирования решений:

1) по составу информационных элементов (имена атрибутов и соответствующие им множества допустимых значений);

2) по организации элементов в структуры (соответствующие принятым в системе уровням представления данных) и определению связей структур различных уровней (отображений их друг друга);

3) по определению ограничений целостности БД и соответствующих процедур их контроля;

4) по разграничению доступа к БД и описанию процессов первоначальной загрузки и ведения баз данных;

5) по разработке или выбору требуемого программного обеспечения, а также формированию организационно-методических и инструктивных материалов.

В узком смысле проектирование баз данных понимается как определение структуры БД и разработка ее схемы на ЯОД конкретной СУБД. Наши интересы будут несколько шире этого понимания, но при этом основное внимание мы уделим проблемам концептуального проектирования. Рассмотрим методы определения состава элементов и их организации в таблицы, методы определения и оценки сложности алгоритмов процедур контроля целостности и согласованности БД. Эти методы и приемы могут применяться как при ручном проектировании, так и для реализации их в системах автоматизированного проектирования (САПР) БД. Краткая содержательная формулировка проблемы проектирования системы баз данных (СБД) может быть выражена следующим образом: *необходимо за минимальное время создать хорошо продуманную СБД, обладающую свойствами расширяемости (учет новых требований), целостности и удовлетворительной реактивности (скорость ответа на запросы).* Решение этой проблемы удобно рассматривать в соответствии с фазами и этапами жизненного цикла СБД (рис.1.18).

ФАЗА АНАЛИЗА И ПРОЕКТИРОВАНИЯ		
1	Проектирование модели функционирования (МФ)	Системный анализ требований к структуре и содержанию процессов обработки данных
2	Информационно-логическое проектирование (ИЛП)	Анализ, систематизация и формулирование понятий, их смыслового содержания и закономерностей информационной модели
3	Концептуальное проектирование (КП)	Построение независимой от ТО и ОПО информационной модели и системы правил контроля ее целостности
4	Логическое проектирование (ЛП)	Разработка ориентированной на конкретный тип СУБД логической организации данных и процедур контроля целостности
5	Физическое проектирование (ФП)	Разработка организации записей на физических носителях, определение методов доступа
ФАЗА РЕАЛИЗАЦИИ И ФУНКЦИОНИРОВАНИЯ		
1	Реализация базы данных, создание, загрузка и испытание СБД	
2	Анализ функционирования и поддержка	
3	Модернизация и адаптация	

Рис.1.18. Фазы жизненного цикла системы баз данных.

В этой схеме опущены два важных этапа создания АС: выбор (или разработка) комплекса технических средств (КТС) и выбор (или разработка) системного программного обеспечения. Основой для решения этих вопросов являются первые три этапа (МФ, ИЛП и КП), которые позволяют обоснованно сформулировать систему требований, определяющих желаемый облик создаваемой системы, ее технического и программного обеспечения. Только после разработки основных проектных решений по составу и типам ЭВМ, ОС и СУБД

возможно приступать к определению логической и физической организации данных, используя при этом результаты концептуального проекта.

На рис.1.19 показаны содержание и взаимосвязь этапов фазы системного анализа и проектирования информационной базы.

Решение проблемы представляет собой два взаимосвязанных процесса: структурирование функций и структурирование их информационного содержания.

Функциональная модель (или модель функционирования — МФ) информационной системы представляет собой систему схем и подробных описаний технологической последовательности процедур и действий должностных лиц организационной системы по переработке информации, включая описание задач (информационных запросов) и документооборота по всем подсистемам (функциональным областям), процессам и функциям. При этом должны быть описаны состав, содержание, циркуляция и требования по обработке документов в каждом процессе. Основной задачей разработки МФ является сбор и систематизация всех требований, предъявляемых к содержанию процесса обработки данных всеми известными и потенциальными пользователями информационных ресурсов организационно-технической системы.

Анализ результатов разработки МФ и реализация сложных по смысловому содержанию действий по структурированию информации приводят к созданию информационно-логической (инфологической) модели данных ($М_{ил}$), которую мы будем формализовать (формально выражать) в виде, соответствующем по структуре определению модели данных:

$$М_{ил} = \langle S_{ил}, P_{ил}, Q_{ил} \rangle,$$

где $S_{ил}$ — инфологический граф (ИЛГ), включающий множество типов инфообъектов и инфосвязей, задаваемых именами своих типов и составом типов своих атрибутов, а также множествами их значений (доменов); $P_{ил}$ — правила интерпретации инфологического графа данных; $Q_{ил}$ — закономерности предметной области, существенные для контроля целостности и согласованности информационной модели.

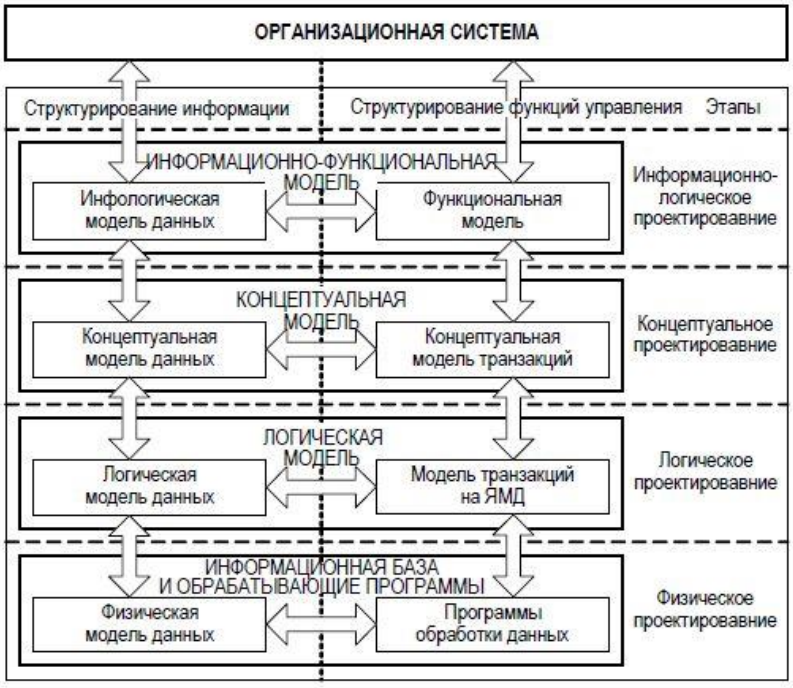

Рис.1.19. Взаимосвязь процессов и результатов разработки программных комплексов.

Инфологическая модель $М_{ил}$ обеспечивает первоначальную формализацию описания информационного содержания автоматизируемых процессов, согласовывая и объединяя в себе представления всех категорий пользователей. Основными критериями оценки качества $М_{ил}$ являются:

— полнота и простота ее понимания;

— детальность, ясность и согласованность (системность) описаний ее элементов.

Хорошие методы, средства и технологии, применяемые на этом этапе, должны позволять в максимальной мере обеспечить достижение указанных свойств модели.

Концептуальная модель (КМ) данных информационной системы представляет собой точное математическое описание элементов данных (атрибутов и их значений), их смысловых связей и организационной структуры с соответствующей системой алгоритмов контроля целостности и согласованности БД. Концептуальная модель должна быть ясной, простой, однозначно понимаемой и легко трансформируемой при необходимости выполнить ее изменения. Так как этим требованиям наилучшим образом удовлетворяют формализмы реляционной модели данных, то далее КМ будем представлять тройкой:

$$М_к = \langle S_к, Р_к, Q_к \rangle,$$

где $S_к$ — схема модели реляционного типа; $Р_к$ — совокупность операций реляционной алгебры; $Q_к$ — система алгоритмов, описывающих процедуры контроля целостности и согласованности модели. В целом результаты разработки концептуальной модели СБД содержат не только систематизированные требования к компонентам информационного обеспечения АС, но и все исходные данные для их создания (рис.1.20.).

Концептуальная модель разрабатывается с целью создания точного и устойчивого во времени представления данных в системе, независимого от систем общего программного обеспечения (ОПО) и технического обеспечения (ТО). Основными критериями оценки качества $М_к$ являются:

— ясность, простота и однозначность понимания;

— системность и выразительная сила;

— сложность алгоритмов контроля целостности и согласованности.

Концептуальная модель транзакций описывает с помощью операций реляционной алгебры процедуры обработки базы данных в процессе ее ведения или поиска информации. По своей сути эти процессы реализуются процедурами контроля целостности и обработки запросов. В общем случае транзакцией (обработкой запросов) называется последовательность логически связанных действий, переводящих информационную систему из одного состояния в другое. Каждая транзакция либо должна завершиться полностью, либо система должна быть возвращена в исходное состояние. В диалоговых системах транзакцией называется акт приема запроса или сообщения, обработка его и выдача ответа. В системах баз данных под транзакцией понимается любая операция обновления БД, при которой БД находится в неустойчивом состоянии.

Логическое проектирование состоит в разработке логической модели данных Мл на основе знания концептуальной модели. Логическая модель определяет правила построения и взаимосвязи логических записей БД, а также обрабатывающих программ обновления (ведения) данных в терминах ЯОД и ЯМД конкретных СУБД. Основными критериями оценки качества Мл являются:

— ясность и простота записи (понимания);

— полнота реализации требований концептуальной модели;

— уровень сложности формулировки запросов к БД.

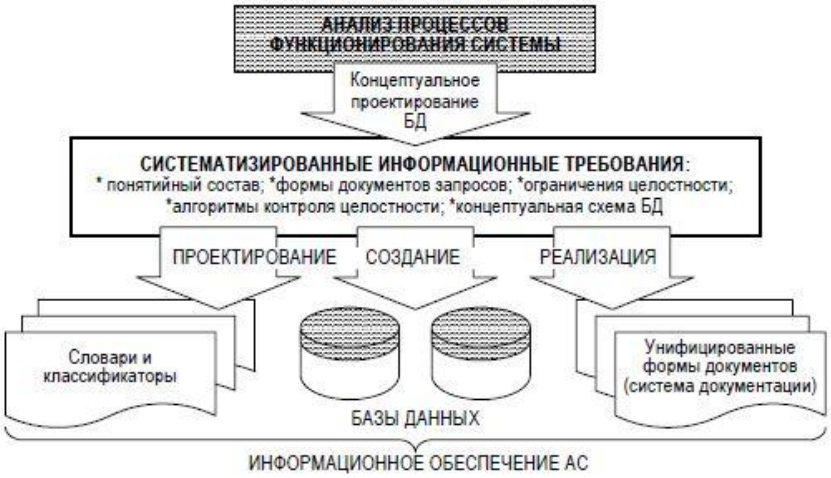

Рис.1.20. Взаимосвязь результатов проектирования БД
с разработкой компонентов информационного обеспечения АС.

Физическое проектирование состоит в определении правил организации записей и их размещения на физических носителях информации, а также в окончательной отладке программ обработки (обновления) БД, специфицированных на предыдущем этапе.

Результатом является полностью готовая к внедрению (опытной эксплуатации) система БД. Основными критериями оценки качества $М_ф$ являются:

— время реакции (отклика на запрос) системы;

— объемы различных видов памяти ЭВМ;

— уровень сложности процедур реструктуризации БД.

Фаза реализации, функционирования и модернизации требует осуществить либо загрузку СБД и проведение испытаний функционирования системы, либо имитационное моделирование процессов функционирования (рис.1.21).

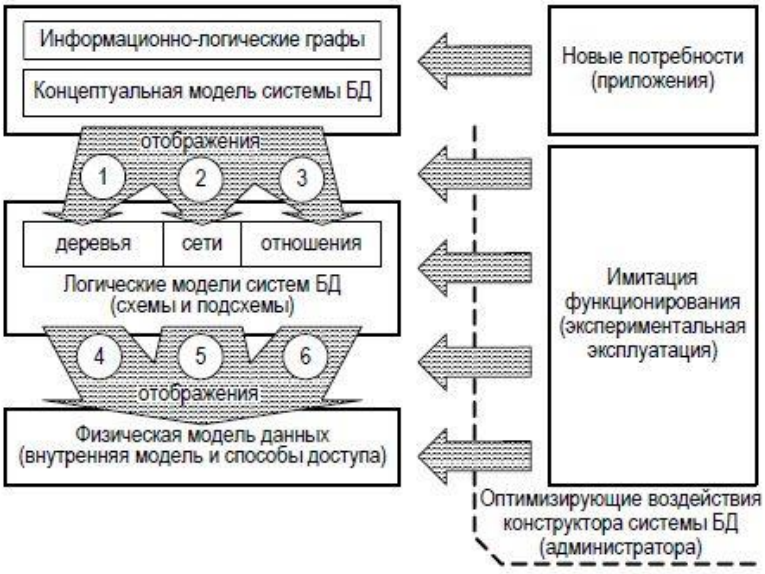

Рис.1.21. Взаимосвязь объектов и фаз проектирования СБД.

Только на этой фазе можно получить такие характеристики, как время реакции системы на различные категории запросов, время, необходимое на создание, обновление, реорганизацию и контроль целостности СБД. Этап анализа функционирования и поддержки СБД используется для оценки качества СБД и определения путей ее совершенствования. Этап

модернизации и адаптации обеспечивает внесение в систему изменений, возникающих вследствие появления новых требований (приложений, задач), а также при необходимости оптимизации системы путем реорганизации (реинжениринга) СБД и/или внесения изменений в ПО. Реорганизация СБД, как правило, связана с необходимостью перепроектирования системы моделей МФ, $М_{ил}$, $М_к$, $М_л$ и $М_ф$, начиная с одной из них. Таким образом, процесс разработки СБД будем определять как процесс последовательной трансформации (отображения) моделей:

$$МФ \rightarrow М_{ил} \rightarrow М_к \rightarrow М_л \rightarrow М_ф.$$

На каждом последующем шаге разработки увеличивается количество понятий, с помощью которых описываются результаты проектирования: к описанию модели предыдущего уровня как бы добавляется описание проектных решений последующего уровня.

Данный процесс целесообразно реализовывать в рамках единойСАПР СБД (рис.1.22), позволяющей проектировщику СБД накапливать в базах данных САПР информацию, необходимую для выполнения проектных расчетов, анализировать решения и запоминать их варианты, разрабатывать проектную документацию, возвращаться с любого последующего этапа на любой предыдущий, а также получать по запросам необходимую справочную информацию и т.п.

САПР СБД позволяют содержать в своих собственных базах данных (репозиториях) согласованную систему различных представлений о создаваемой СБД: от самых простых «взглядов» конечных пользователей до абсолютно точного, учитывающего все детали технической реализации, «взгляда» АБД. За рубежом такие инструментальные средства на ранних этапах назывались «Словарями данных» или «Словарями-справочниками данных» (Data Dictionary System — DDS), а впоследствии — CASE-системами. Базы данных таких систем содержат как бы метаданные (данные о данных, которые будут содержаться и обрабатываться в создаваемых базах данных) разрабатываемой АС и играют роль базы знаний, которая может передаваться, использоваться и пополняться на стадиях опытной и практической эксплуатации системы. Другими словами, созданная АС может и должна содержать в себе информацию о самой себе, которая может быть быстро доступна эксплуатационному персоналу для решения возникающих проблем применения АС (эксплуатация, реорганизация и т.д.).

Использование САПР СБД позволяет разработчикам достигать наилучших значений критериев качества проектных решений на различных этапах проектирования и реализации СБД. Состав критериев и их взаимосвязь отражены на рис.1.23.

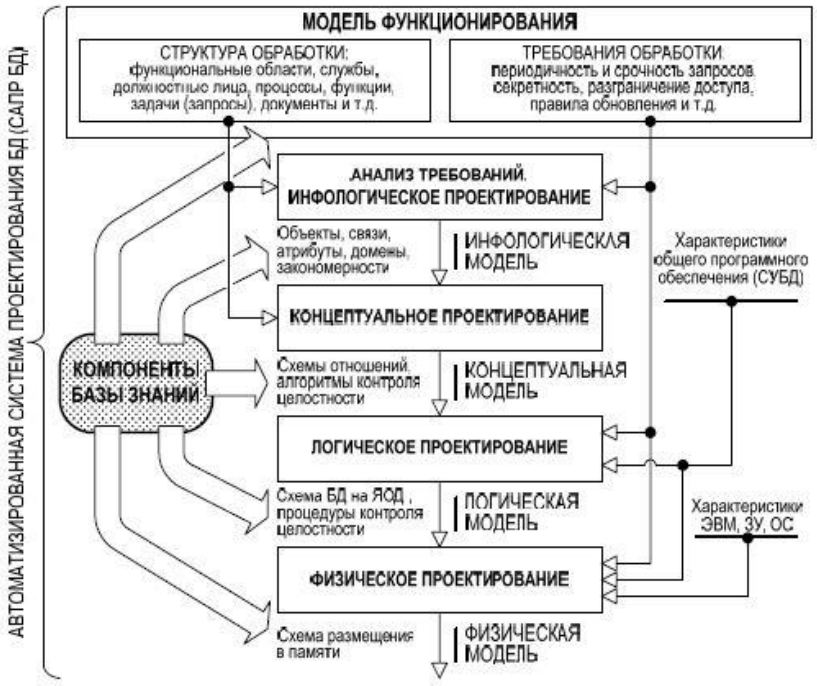

Рис.1.22. База знаний САПР СБД.

ПРОЕКТЫ СБД		
ИНФОЛОГИЧЕСКИЙ	**КОНЦЕПТУАЛЬНЫЙ**	**РЕАЛИЗАЦИЯ СИТЕМЫ**
Полнота, ясность, детальность, согласованность описания типов данных	*Точность, однозначность понимания, выразительная сила, системность* концептуальной схемы базы данных	*Полнота* информационной базы
	Простота языка описания концептуальной схемы	*Простота* взгляда пользователя
	Постоянство внешних концептуальных схем	*Постоянство* взгляда пользователя
	Простота изменения концептуальной схемы	*Простота* изменения структуры
Полнота, точность определения доменов и структуры ФЗ	*Точность* описания алгоритмов контроля целостности и согласованности	*Целостность и согласованность*
Полнота, точность описания информационных потребностей пользователей	*Точность* внешних концептуальных схем и простота связи с глобальной концептуальной схемой	*Защита* пользователей от взаимного влияния и *разграничение* доступа
Согласованность словарей и классификаторов данных	*Согласованность* языков описания концептуальных схем	*Простота* информационной совместимости с другими системами

Рис.1.23. Состав и взаимосвязь критериев проектных решений.

Контрольные вопросы

1. Дайте определения понятиям: «система», «структура», «целостность», «состояние системы», «поведение системы».

2. Дайте определения понятиям: «автоматизированный банк данных», «динамическая информационная модель», «предметная область».

3. Поясните роль, предназначение и взаимосвязь основных элементов АБнД.

4. Дайте определение понятию «модель данных».

5. Назовите основные понятия, которые используются при формулировке модели данных, и укажите их связь с понятиями, применяемыми для описания реального мира.

6. Дайте определение схемы базы данных и назовите основные типы моделей данных.

7. Определите суть структуры и элементы иерархической модели данных.

8. Определите суть структуры и элементы сетевой модели данных.

9. Определите суть структуры и элементы реляционной модели данных.

10. Сопоставьте основные характеристики иерархической, сетевой и реляционной моделей данных.

11. Перечислите основные уровни представления данных в АС, определите их содержание и предназначение.

12. Перечислите основные способы адресации логических записей и их особенности.

13. Определите основное содержание процессов проектирования баз данных в широком и узком смысле.

14. Определите содержание фаз и этапов жизненного цикла систем баз данных.

15. Дайте определение понятию «функциональная модель» информационной системы.

16. Дайте определение понятию «инфологическая модель данных».

17. Дайте определение понятию «концептуальная модель информационной системы».

18. Опишите процесс разработки СБД, основные результаты этапов и критерии их оценки.

2. Защита информации в СУБД

2.1 Концепция безопасности баз данных

Информация, хранящаяся в БД, является наиболее ценным ресурсом информационной системы и, естественно, должна быть надлежащим образом защищена.

Защита информации в автоматизированных информационных системах, одним из основных элементов которых является БД, представляет собой совокупность методов, средств и мероприятий, предназначенных для предупреждения искажений, уничтожения и несанкционированного использования защищаемой информации.

В настоящее время в информационных системах имеет место ситуация, для которой характерны следующие особенности:

- резкое увеличение объемов накапливаемой и хранимой на машинных носителях информации;
- концентрация в БД информации различного назначения и принадлежности;
- резкое расширение круга лиц, имеющих непосредственный доступ к ресурсам информационных систем;
- функционирование систем с БД в локальных и корпоративных сетях.

Указанные особенности привели к существенному повышению уязвимости информации по отношению к внешним неконтролируемым воздействиям. Результатом таких воздействий может быть несанкционированная модификация (изменение) или несанкционированное использование информации. Значительно увеличилось число попыток несанкционированных действий как со стороны пользователей систем с БД, так и лиц, не являющихся пользователями.

Перечисленными особенностями функционирования информационных систем объясняется возникновение целого ряда возможных способов несанкционированного доступа к данным, основными из которых являются:

- хищение или копирование магнитных и немагнитных носителей информации;
- программный доступ к «чужим» данным, находящимся в БД или хранящимся в оперативной памяти;
- сбор информации, остающейся в оперативной памяти после завершения решения задачи;
- несанкционированное получение данных под видом законного пользователя и др.

Современные средства общесистемного программного обеспечения, особенно операционные системы и СУБД, реализуя широкие возможности организации вычислительного процесса по обработке данных, имеют целый ряд потенциальных каналов несанкционированного доступа. Кром того, в создаваемых системах с БД, содержащих обширную информацию по различным объектам, существуют возможности получить защищаемые данные косвенным путем – посредством постановки серии вполне лояльных запросов.

2.2 Основные понятия в теории безопасности баз данных

В теории защиты информации, информационной безопасности вообще и безопасности БД в частности используется множество специфических понятий, в которые часто вкладывается интуитивный смысл. Отчасти это обусловлено становлением теории защиты информации и информационной безопасности как новой, интенсивно развивающейся отрасли знаний, которая существует в таком качестве сравнительно недолго. Для обеспечения однозначной трактовки понятий в излагаемом ниже материале рассмотрим основную

терминологию в области защиты информации и, в частности, безопасности и защиты данных, хранящихся в БД.

Защита данных – процесс предупреждения несанкционированного разрушения, получения или модификации данных в процессе их хранения и обработки.

Механизм защиты данных – это организационно-техническая совокупность методов и средств, создаваемая с целью защиты данных.

Средства защиты – ресурсы информационной системы, выделяемые для организации и обеспечения защиты ее данных.

Технические средства защиты – физические и аппаратные устройства, способные выполнять функции защиты данных.

Под физическими средствами защиты понимаются механические, электронно-механические или электромеханические устройства, специально предназначенные для создания физических препятствий на пути к защищаемой информации.

Аппаратными средствами защиты являются различные электронные устройства, которые включаются в состав аппаратных средств информационной системы и выполняют (самостоятельно или в комплексе с другими средствами) некоторые функции защиты.

Программные средства защиты – специальные программы, включаемые в состав программного обеспечения системы для осуществления функций защиты данных.

Программные средства ввиду их универсальности, простоты реализации и гибкости являются важнейшей и непременной частью механизма защиты современных информационных систем. В соответствии с выполняемыми функциями различают программы:

- идентификации, т.е. опознавания (терминала, пользователя);
- разграничения доступа (к задачам, элементам БД);
- криптографического закрытия информации;
- защиты программ (операционных систем, СУБД, программ пользователей);
- уничтожения остаточной информации, формирования грифа секретности выдаваемых документов, ведения регистрационных журналов, тестового контроля системы защиты (вспомогательные) и др.

Криптографические средства защиты – программные средства защиты, основанные на преобразовании защищаемых данных, в результате чего осуществляется скрытие смысла хранимой в системе или передаваемой по каналам связи информации.

Организационные средства защиты – специальные организационно- технические и организационно-правовые мероприятия, акты и правила, осуществляемые в процессе создания, внедрения и применения автоматизированных систем по защите информации.

Конфиденциальная информация (sensitive information) – информация, которая требует защиты.

Доступ к информации (access to information) – ознакомление с информацией, ее обработка (в частности, копирование), модификация, уничтожение.

Субъект доступа (access subject) – лицо или процесс, действия которого регламентируются правилами разграничения доступа.

Объект доступа (access object) – единица информации автоматизированной системы, доступ к которой регламентируется правилами разграничения доступа. Объектами доступа (контроля) в СУБД является практически все, что содержит конечную информацию: таблицы (базовые или виртуальные), представления, а также более мелкие элементы данных: столбцы и строки таблиц и даже поля строк (значения).

Таблицы БД и представления имеют владельца или создателя. Их объединяет еще и то, что все они для конечного пользователя представляются как таблицы, т.е. как нечто именованное, содержащее информацию в виде множества строк (записей) одинаковой структуры. Строки таблиц разбиты на поля именованными столбцами. Наличие указанных

факторов позволяет реализовать систему разграничения доступа к любому из указанных объектов.

Правила разграничения доступа (security policy) – совокупность правил, регламентирующих права субъектов доступа по отношению к объектам доступа.

Санкционированный доступ (authorized access to information) – доступ к информации, который не нарушает правил разграничения доступа.

Несанкционированный доступ (unauthorized access to information) – доступ к информации, который нарушает правила разграничения доступа с использованием штатных средств, предоставляемых средствами вычислительной техники или автоматизированными системами.

Идентификатор доступа (access identifier) – уникальный признак объекта или субъекта доступа, позволяющий однозначно выделить его среди прочих.

Идентификация (identification) – присвоение объектам и субъектам доступа идентификатора и (или) сравнение предъявляемого идентификатора с перечнем присвоенных идентификаторов.

Пароль (password) – идентификатор субъекта доступа. Пароль должен храниться в секрете и использоваться для идентификации субъекта путем сравнения вводимого пароля с образцом, хранящимся в системе защиты.

Аутентификация (установление подлинности, опознавание) (authentification) – проверка принадлежности субъекту доступа предъявленного им идентификатора.

В СУБД на этапе подключения к БД производится идентификация и проверка подлинности пользователей. В дальнейшем пользователь или процесс получает доступ к данным согласно его набору полномочий. В случае разрыва соединения пользователя с БД текущая транзакция откатывается, а при восстановлении соединения требуются повторная идентификация пользователя и проверка его полномочий.

Уровень полномочий субъекта доступа (subject privilege) – совокупность прав доступа субъекта доступа (далее используется термин «привилегия»).

Нарушитель правил разграничения доступа (security policy violator) – субъект доступа, который осуществляет несанкционированный доступ к информации.

Модель нарушителя правил разграничения доступа (security policy violator model) – абстрактное (формализованное или неформализованное) описание нарушителя правил разграничения доступа.

Целостность информации (information integrity) – способность информационной системы обеспечить неизменность информации в условиях случайного и (или) преднамеренного искажения (разрушения).

Метка конфиденциальности (sensitivity label) – элемент информации, характеризующий конфиденциальность объекта.

Многоуровневая защита (multilevel secure) – защита, обеспечивающая разграничение доступа субъектов с различными правами доступа к объектам различных уровней конфиденциальности.

Полномочия – права пользователя, терминала, задачи (программы) или другого объекта осуществлять чтение, запись или модификацию защищаемых данных.

Матрица полномочий – прямоугольная матрица ||aij||, элементы которой aij содержат информацию о правах (полномочиях) i-го субъекта (объекта) относительно j-го элемента защищаемых данных.

Профиль полномочий – строка матрицы полномочий с полным перечнем полномочий одного объекта (субъекта) относительно всех элементов защищаемых данных.

Проверка полномочий – установление соответствия объектов и действий над ними, содержащихся в запросе, объектам и действиям, разрешенным запрашивающему субъекту (объекту) в соответствии с его профилем полномочий.

2.3 Угрозы безопасности баз данных

БД представляет собой важнейший корпоративный ресурс, который должен быть надлежащим образом защищен с помощью соответствующих средств системы.

Потенциальными опасностями (угрозами безопасности) для БД являются:

- похищение и фальсификация данных;
- утрата конфиденциальности (нарушение тайны);
- нарушение неприкосновенности личных данных;
- утрата целостности;
- потеря доступности.

Похищение и фальсификация данных всегда совершаются людьми, поэтому основное внимание должно быть сосредоточено на сокращении количества удобных ситуаций для выполнения подобных действий. Например, следует строго ограничить доступ к данным о выплачиваемой зарплате, точно зафиксировать количество отпечатанных платежных поручений, а также организовать четкий учет и автоматическое уничтожение всех файлов, которые могут быть использованы для несанкционированных попыток печати платежных документов. Следует заметить, что похищение и фальсификация вовсе необязательно связаны с изменением каких-либо данных.

Понятие конфиденциальности означает необходимость сохранения данных в тайне. Как правило, конфиденциальными считаются те данные, которые являются критичными для всей организации, тогда как понятие неприкосновенности личных данных касается требования защиты информации об отдельных сотрудниках. Следствием нарушения в системе защиты, вызвавшего потерю конфиденциальности данных, может быть утрата позиций в конкурентной борьбе, финансовые потери, снижение обороноспособности государства и т.д.

Утрата целостности данных приводит к искажению или разрушению данных, что может иметь серьезные последствия для дальнейшей работы организации.

Недоступное состояние БД означает невозможность извлечения из нее каких-либо данных. В некоторых случаях те события, которые послужили причиной перехода системы в недоступное состояние, могут одновременно вызвать и разрушение данных в базе.

Угроза безопасности может быть вызвана ситуацией или событием, способным нанести вред организации. Причиной угрозы может служить человек, происшествие или стечение обстоятельств.

Любая опасность должна рассматриваться как потенциальная возможность нарушения функций системы защиты, которая в случае своей реализации может оказать то или иное негативное влияние. Например, следствием просмотра и раскрытия засекреченных данных могут стать похищение и фальсификация, утрата конфиденциальности и нарушение неприкосновенности личных данных сотрудников организации.

К основным типам опасностей, которые могут быть как намеренными, так и непреднамеренными, могут быть отнесены:

- использование прав доступа другого человека;
- несанкционированное изменение или копирование данных;
- изменение программ;
- применение непродуманных методик, допускающих смешивание конфиденциальных и обычных данных в одном документе;
- подключение к кабельным сетям;
- ввод некорректных данных хакерами;
- шантаж эксплуатационного и руководящего персонала систем;
- создание «лазеек» в систему;
- похищение данных, программ и оборудования;
- отказ систем защиты, вызвавший превышение допустимого уровня доступа;

- нехватка персонала, необоснованные сокращения специалистов, забастовки;
- недостаточная обученность персонала;
- просмотр и раскрытие засекреченных данных;
- электронные наводки и радиация;
- разрушение данных в результате отключения или перенапряжения в сети электропитания;
- пожары, наводнения, диверсии;
- физическое повреждение оборудования;
- обрыв или отсоединение кабелей;
- внедрение компьютерных вирусов.

Технические, программные, криптографические и организационные средства защиты в пределе должны парировать любую из перечисленных угроз безопасности БД.

2.4 Требования к системе обеспечения безопасности баз данных

Очевидно, что абсолютная ценность информации существенно различается в зависимости от уровня организаций, эксплуатирующих системы с БД. Особую ценность она представляет для правительств государств, оборонных и силовых структур, крупных банков и фирм. Потеря или искажение информации для таких организаций могут привести к катастрофическим последствиям для страны и вызвать негативные последствия в мировой экономике.

Безопасность БД в течение долгого времени находилась как бы на втором плане в сравнении с такими областями, как безопасность сетей и коммуникаций, но некоторые нашумевшие происшествия, например появление вируса «Микеланджело» и сетевого червя «Интернет», привлекли внимание к проблеме защиты БД и информационных массивов.

Безопасность БД в значительной степени обеспечивается при выполнении следующих основных требований:

- установки в системе резервного оборудования с развернутым программным обеспечением;
- выполнения регулярного резервного копирования данных на внешние носители;
- авторизации пользователей;
- шифрования данных с помощью криптографических методов;
- наличия административных мер по сохранности данных.

К современным криптографическим системам защиты информации предъявляются следующие общепринятые требования:

- зашифрованное сообщение должно поддаваться чтению только при наличии ключа;
- число операций, необходимых для определения использованного ключа шифрования по фрагменту шифрованного сообщения и соответствующего ему открытого текста, должно быть не меньше общего числа возможных ключей;
- число операций, необходимых для расшифровывания информации путем перебора всевозможных ключей, должно иметь строгую нижнюю оценку и выходить за пределы возможностей современных компьютеров (с учетом возможности использования сетевых вычислений);
- знание алгоритма шифрования не должно влиять на надежность защиты;
- незначительное изменение ключа должно приводить к существенному изменению вида зашифрованного сообщения;
- структурные элементы алгоритма шифрования должны быть неизменными;
- дополнительные биты, вводимые в сообщение в процессе шифрования, должны быть полностью и надежно скрыты в шифрованном тексте;

- длина шифрованного текста должна быть равной длине исходного текста;
- не должно быть простых и легко устанавливаемых зависимостей между ключами, используемыми в процессе шифрования;
- любой ключ из множества возможных ключей должен обеспечивать надежную защиту информации;
- алгоритм должен допускать как программную, так и аппаратную реализацию, при этом изменение длины ключа не должно вести к качественному ухудшению алгоритма шифрования.

3. Реализация механизмов безопасности баз данных

3.1 Защита от несанкционированного доступа

Для выполнения основных требований к обеспечению безопасности БД необходимо реализовать в системе две фундаментальные функции:

- проверку полномочий;
- проверку подлинности (аутентификацию).

Проверка полномочий основана на том, что каждому пользователю или процессу информационной системы соответствует набор действий, которые он может выполнять по отношению к определенным объектам.

Проверка подлинности означает достоверное подтверждение того, что пользователь или процесс, пытающийся выполнить санкционированное действие, действительно тот, за кого он себя выдает. Рассмотрим эти понятия подробнее.

Проверка полномочий. Базовая модель проверки полномочий представлена на рис. 3.1. Теоретически можно поддерживать матрицу безопасности, устанавливающую отношения между всеми пользователями и процессами системы, с одной стороны, и всеми объектами – с другой. В каждом элементе матрицы хранится список, содержащий некоторое подмножество операций, которые данный пользователь или процесс может выполнять по отношению к данному объекту.

	File1	File2	File3	File4	File5	. . .
Мусанова	Ч, М	Ч, М, С,		Ч, М, С,	Ч, М, С,	
Съянов	Ч, С,		Ч, М, С,		Ч, С,	
Федорова	Ч, М, С, У		Ч, С,			
Штырлова	Ч, М, С, У	Ч, М, С,			Ч, С,	
. . .						

где Ч - Чтение; С - Создание; М - Модификация; У – Удаление.

Рис. 3.1. Базовая модель проверки полномочий.

На первый взгляд, система управления безопасностью, поддерживающая такую матрицу, могла бы гарантировать высокий уровень защиты информации. В реальности все несколько сложнее.

На рис. 3.2 показано, почему такая базовая модель недостаточна для обеспечения надлежащего уровня безопасности.

Если Процесс 2 не имеет соответствующих полномочий, но ему удается выдать себя за Процесс 1, то он сможет выполнять действия, разрешенные Процессу 1, но не разрешенные Процессу 2. Поэтому очевидно, что необходимы дополнительные меры защиты.

Проверка подлинности. В силу описанных выше причин в безопасной среде должна присутствовать модель проверки подлинности, которая обеспечивает подтверждение заявленных пользователями или процессами идентификаторов (имен). Проверка полномочий приобрела еще большее значение в условиях массового распространения распределенных

вычислений. При существующем высоком уровне связности вычислительных систем необходимо контролировать все обращения к системе.

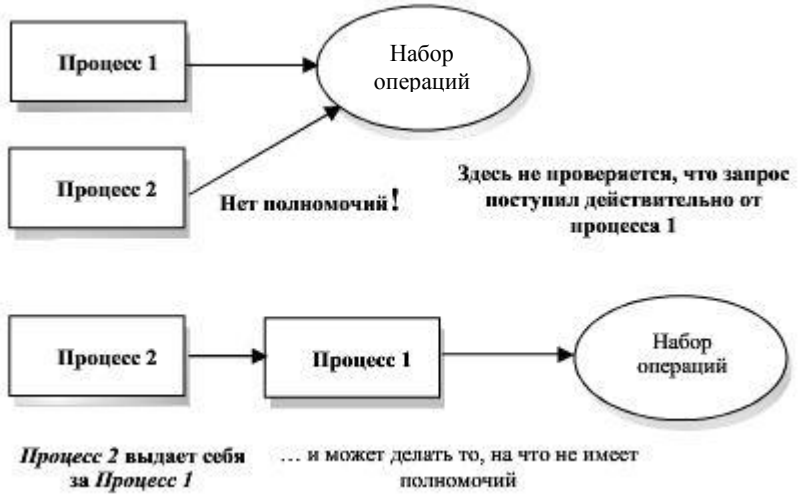

Рис. 3.2. Иллюстрация необходимости проверки подлинности субъекта в дополнение к проверке полномочий.

Проверка подлинности или, иначе, опознавание как функция защиты представляет собой установление подлинности субъекта, обращающегося к БД, или объекта, привлекаемого к работе в процессе обработки защищаемой информации.

В самом общем случае опознаванию подлежат:

- *субъекты:*
  - пользователи информационной системы (при обращении к системе);
  - администратор БД при выполнении своих функций над защищаемыми данными;
  - прикладные и системные программисты, если они участвуют в отладке или эксплуатации программ, в процессе автоматизированной обработки защищаемой информации;
  - инженерно-технический персонал, участвующий в техническом обслуживании аппаратно-программных средств, при обработке защищаемой информации.
- *объекты:*
  - удаленные терминалы, с которых осуществляется доступ пользователей к системе;
  - программы, используемые для обработки защищаемой информации;
  - элементы БД, содержащие защищаемую информацию и др.

Применение того или иного способа опознавания, а также сложность этой процедуры определяются особенностями организационно-технического построения и функционирования информационной системы; степенью секретности защищаемой информации, ее объемом и др.

Наиболее распространен подход к опознаванию, основанный на использовании паролей. В зависимости от сложности этой процедуры различают три основные группы опознавания: простое, усложненное и особое опознавание.

Простое опознавание, как правило, сводится к сравнению кода (пароля), предъявляемого терминалом или пользователем, с эталонным кодом (паролем), хранящимся в памяти ЭВМ. При усложненном опознавании обычно используется дополнительная информация: система разовых паролей, персональная информация пользователя и т.д. Усложненное опознавание осуществляется обычно в режиме диалога: система формирует вопросы, на которые опознаваемый должен дать ответы. По содержанию ответов система принимает решение об опознавании. При этом могут быть использованы такие характеристики объектов опознавания, как личная подпись, характеристики голоса, отпечатки

пальцев и др. В последнее время разрабатываются методы опознавания на основе криптографического преобразования персональной информации.

При использовании простого метода опознавания эта процедура может быть представлена в виде следующей последовательности действий.

При необходимости получения данных, хранящихся в БД информационной системы, пользователь на своем терминале формирует по установленному формату запрос-сообщение, в котором указывает свой пароль. В памяти ЭВМ набор всех паролей хранится в специальном массиве. Правомочность запроса пользователя устанавливается программой опознавания, которая сравнивает пароли, хранящиеся в памяти ЭВМ, с паролями, вводимыми пользователем.

Поскольку пользователь может допустить ошибку при вводе пароля, то обычно предусматриваются повторный запрос и сравнение паролей.

Работа пользователя при этом может блокироваться, если и при повторных вводах пароль не совпадает с эталоном, хранящимся в памяти ЭВМ.

Важнейшим параметром пароля является длина. Если длина пароля мала, можно, зная структуру запроса, осуществить перебор всех возможных значений и получить несанкционированный доступ к системе. Требуемую длину пароля можно рассчитать по эмпирической формуле:

$$A^S \geq 4.42 \cdot 10^4 \frac{R \cdot M}{E \cdot P},$$

где A – число символов алфавита, из которого составляется пароль; R – скорость передачи данных (символов запроса) по каналам связи, (симв./мин); M – время, в течение которого могут предприниматься попытки подбора пароля (в месяцах при работе 24 ч/сут);
E – объем запроса-сообщения, символы; P – допустимая вероятность раскрытия пароля.

Например, если P ≥0,001, M = 3 месяца, E = 20 символов, R =600 симв./мин, то при A = 26 должно выполняться условие S ≥ 7, т.е. длина пароля должна быть не менее 7 символов. Предполагается, что символы паролей выбираются случайно и должны быть приняты эффективные меры по их защите в памяти ЭВМ. При работе с паролями должна соблюдаться и такая мера предосторожности, как предупреждение их распечатки или вывода на экраны дисплеев.

3.2 Способы разграничения доступа

Основной способ разграничения доступа к данным основан на использовании замков управления доступом, что позволяет объявить любой элемент БД закрытым и присвоить ему персональный замок. После этого доступ к данному элементу будет разрешен только в том случае, если в запросе будет предъявлен ключ именно к этому замку. Язык описания данных, используемый для описания логической структуры БД, позволяет закрыть замком любую логическую структуру на всех иерархических уровнях.

Для создания более гибкой системы разграничения доступа предусматривается возможность выделения одному и тому же элементу БД нескольких замков, каждый из которых предназначается для защиты элементов данных относительно определенного режима обработки (чтение, запись, добавление, изменения и др.). Сам замок может быть задан в виде постоянного кода, значения переменной или результата некоторой процедуры. Если замок задан константой или переменной, то для доступа к данным необходимо простое совпадение замка и предъявленного ключа. Если же замок задан процедурой, то доступ к данным будет разрешен только в случае получения вполне определенного результата процедуры.

Процедуры могут быть различными по содержанию, например: «произвести над ключом некоторые преобразования»; «ответить на вопрос, заданный системой» и т.д.

Наибольшее распространение в системах обработки данных получили способы разграничения доступа по спискам, матричный, по уровням (кольцам) секретности.

Разграничение доступа по спискам осуществляется в том случае, если профили прав пользователей на доступ заданы в виде списков. При этом задается либо список пользователей для каждого элемента БД, имеющих право доступа к нему, либо перечень тех элементов базы для каждого пользователя, к которым ему разрешен доступ. В любом случае процедура разграничения осуществляется в следующей последовательности:

- по данным, содержащимся в запросе, выбирается соответствующая строка списка: перечень пользователей, допущенных к запрашиваемому элементу, или перечень элементов БД, к которым допущен обратившийся с запросом пользователь;
- в выбранной строке проверяется наличие имени пользователя, обратившегося с запросом, или имени элемента БД, к которому обращается пользователь;
- по данным проверки принимается решение о допуске. При положительном решении разрешается доступ к запрашиваемым данным, при отрицательном – запрос на доступ блокируется.

По результатам проверки предусматриваются санкции за попытку несанкционированного доступа: предупреждение пользователя о том, что им допущены несанкционированные действия, отключение пользователя от системы, формирование сигнала (сообщения) о попытке несанкционированных действий.

Матричное разграничение доступа (рис. 3.3) является более гибким по сравнению с предыдущим. Оно позволяет регулировать не только доступ к данным, но и режимы обработки данных (чтение, запись, модификация и др.). Обеспечивается это тем, что профили прав пользователей задаются в виде матрицы, в строках которой представлен список пользователей, а в столбцах – перечень имен элементов БД. Элементами матрицы являются коды, каждый из которых содержит информацию о полномочиях соответствующих пользователей относительно соответствующих элементов БД. Множество возможных прав определяется разрядностью кода. Если код прав пользователей будет двухразрядным, то его различные значения могут иметь, например, такое содержание: 00 – доступ запрещен; 01 – разрешается только чтение; 10 – разрешается только запись; 11– разрешается чтение и запись (рис. 3.3).

Недостатки данного способа:

- в системах с большим объемом защищаемых данных матрицы полномочий являются чрезвычайно громоздкими;
- динамическое ведение матриц в процессе функционирования системы является достаточно сложной процедурой.

Разграничение доступа по уровням (кольцам) секретности заключается в том, что базы (массивы) защищаемых данных делятся на части в соответствии с уровнями их секретности: «общего доступа», «секретно», «совершенно секретно» и т. д. Полномочия каждого пользователя задаются максимальным уровнем секретности данных, доступ к которым ему разрешен. В соответствии с этим пользователю разрешается доступ ко всем данным, уровень секретности которых не выше уровня его полномочий. Данный способ разграничения доступа является наименее гибким из всех рассмотренных. Однако аппаратные средства современных систем обеспечивают возможность изоляции различных колец, что существенно упрощает реализацию данного способа.

В целом проблема защиты данных от несанкционированного доступа является комплексной и требует к себе такого подхода, при котором взаимосвязано решаются вопросы разработки технических средств, программного обеспечения и архитектуры вычислительных комплексов, предназначенных для использования в информационных системах.

3.3 Защита от вывода

Как правило, системы с БД функционируют в локальных сетях. Локальная сеть по своей сути является многопользовательской средой, поэтому опасность случайного или умышленного просмотра, изменения или удаления информации БД в этом случае особенно велика. Системы защиты данных предназначены именно для многопользовательской среды.

Рис. 3.3. Схема матричного разграничения доступа.

В такой среде очевидной угрозой для БД является возможность ее копирования на внешние запоминающие устройства того или иного типа (дискеты, компакт-диски и т.п.) кем-либо из пользователей сети с целью переноса ее на другой компьютер для попыток взлома в спокойной обстановке. Для исключения вывода БД на внешние устройства посторонними пользователями должны быть использованы средства сетевой операционной системы.

Процедура защиты данных от вывода может быть представлена в виде набора следующих мероприятий:

- каждый сетевой пользователь перед предоставлением ему доступа к сети должен быть идентифицирован с помощью уникального имени и пароля, при этом пользователи должны сохранять свои пароли в тайне и менять их каждые 2-3 месяца;
- каждый идентифицированный пользователь должен быть авторизован. Только после этого ему может быть предоставлен доступ к определенным сетевым компонентам, например папкам на сервере, принтерам и другим общим ресурсам. Естественно, для эпизодических пользователей должны быть закрыты возможности самостоятельного вывода

информации на устройства большой емкости. С целью обеспечения процедуры авторизации для каждого пользователя необходимо создать сетевую пользовательскую учетную запись, в которую должна входить идентификационная информация и сведения о его правах доступа к ресурсам сети. Содержащий эти данные сетевой файл должен быть закодирован и доступен только сетевому администратору;

- работу сетевых пользователей следует контролировать, чтобы избежать несанкционированного доступа к сетевым ресурсам. При обнаружении нескольких попыток нарушения защиты сети к пользователю должны быть применены административные санкции, при этом в качеств е очевидной санкции должен быть запрет на дальнейшую работу в сети;
- сама сеть должна быть защищена от несанкционированного доступа и изменения информации с помощью специальных систем безопасности. Эти системы должны обеспечивать защиту от взлома хакерами и проверку на наличие в сети вирусов;
- хранимые на сетевых серверах данные должны быть защищены от сбоев аппаратных средств и их повреждений с помощью периодического резервирования.

При использовании сети с выделенными серверами за выполнением перечисленных мероприятий должны следить сетевые администраторы. В одноранговых сетях защита сети должна поддерживаться всеми пользователями, которые работают с ее общими ресурсами.

3.4 Целостность баз данных

Под **целостностью** («правильностью») БД понимается такое ее состояние, которое удовлетворяет заданным в формализованной форме и известным СУБД условиям и ограничениям, наложенным на БД.

Условия целостности могут быть заданы в программных, табличных, предикатных и других формах.

Если рассматривать понятие целостности БД с математической точки зрения, то оно означает существование некоторого закона (правила, критерия) L_1 принадлежности кортежей $t_1, t_2, \ldots, t_n$ к отношению r.

Этот закон в применении к реляционным таблицам конкретизируется в спецификации ограничений целостности. Эти ограничения накладываются на таблицу БД в одной из указанных выше форм. Суть ограничений заключается в задании базового или определенного программистом типа данных, соответствующего домену $dom(A_i)$ атрибута A_i реляционной таблицы.

Пример 3.1: Пусть определены реляционная таблица r с атрибутами X, Y и ограничение целостности Y=sin(X). При использовании библиотечной функции sin(X) для атрибута Y должен быть выбран тип данных, соответствующий типу результата, возвращаемого функцией sin(X), например DOUBLE PRECISION (двойная точность). Значения атрибутов X, Y должны соответствовать ограничению целостности. При невыполнении ограничения могут быть проведены соответствующие корректировки для приведения таблицы в состояние целостности.

Под *ссылочной целостностью* понимается свойство связности множества реляционных таблиц, составляющих БД, обеспечиваемое корректным объявлением внешних ключей. Корректность объявления внешних ключей означает, что кортеж с некоторым значением данного набора атрибутов может быть включен в отношение тогда и только тогда, когда это отношение является актуальным значением первичного ключа указанного другого отношения, т.е. ссылки должны выполняться на реально существующие кортежи отношения.

Средством наложения ограничений целостности является управляющий оператор CREATE TABLE языка SQL.

Этот оператор можно применять не только для создания таблиц с заданными наборами полей и определения типов данных для этих полей, но и для определения или до определения доменов в БД. Формальное описание полного синтаксиса оператора CREATE TABLE можно представить в следующем виде:

<оператор CREATE TABLE>::= CREATE TABLE <имя таблицы>
(<имя столбца> <тип данных> [<размер>]
[<ограничения на столбец>…]
[<значение по умолчанию>] [,…]
<ограничения на таблицу> [,…])
<ограничения на столбец>::= NOT NULL | UNIQUE |
CHECK <предикат> | PRIMARY KEY |
REFERENCES <имя таблицы> [(<имя столбца>)]
<ограничения на таблицу>::= UNIQUE (<список столбцов>) |
CHECK (<предикат>) | PRIMARY KEY (<список столбцов>) |
FOREIN KEY (<список столбцов>) |
REFERENCES <имя таблицы> (<список столбцов>)
<значение по умолчанию>::= DEFAULT VALUE=
< выражение>

Пример 3.2:
CREATE TABLE Tab_1(Field_1 integer NOT NULL unique,
Field_2 char(10) NOT NULL unique, CITY char(10)
CHECK (CITY in (‘Москва’, ‘Лондон’, ‘Берлин’)),
SNN decimalCHECK (SNN <1));

Здесь задано использование только трех городов и наложено ограничение на значения поля SNN, которые могут быть только меньше единицы.

3.5 Аудит

Процедура аудита предназначается для проверки внешней аудиторской фирмой задействованных средств управления и соответствия уровня защищенности данных установленным требованиям.

В ходе выполнения проверки аудиторы знакомятся с используемыми ручными процедурами, проверяют все компьютерные системы и состояние всей документации на систему. Обычно аудиторская проверка предусматривает контроль следующих процедур и механизмов управления:

- обеспечения точности вводимых данных;
- поддержания точности процедур обработки данных;
- предотвращения появления и своевременного обнаружения ошибок в процессе выполнения программ;
- корректного тестирования, документирования и сопровождения разработанных программных средств;
- предупреждения несанкционированного изменения программ;
- предоставления прав доступа и контроля за их использованием;
- поддержания документации в актуальном состоянии.

Все процедуры и средства контроля должны быть достаточно эффективными, в противном случае они должны быть подвергнуты пересмотру.

Для определения активности использования БД и анализа нештатных ситуаций в системе используются ее файлы журнала.

Проведение аудиторских проверок и регулярный контроль содержимого файлов журнала с целью выявления ненормальной активности в системе часто позволяет обнаружить и пресечь любые попытки нарушения защиты.

3.6 Задачи и средства администратора безопасности баз данных

Наиболее важными задачами администратора безопасности БД (АББД) являются:

- создание и поддержка пользовательских учетных записей;
- периодическое резервирование системных файлов и файлов БД;
- сжатие файлов и их восстановление в случае сбоев;
- шифрование и дешифрование файлов БД;
- анализ системного журнала с целью выявления попыток получения несанкционированного доступа;
- проведение полных проверок компьютеров сети на наличие вирусов и др.

Естественно, что круг этих задач может быть расширен в зависимости от используемой системы защиты в конкретной информационной системе и принятой политики безопасности.

Как правило, в любой СУБД имеются многочисленные сервисные функции, позволяющие АББД решать перечисленные задачи. Далее рассмотрим конкретные средства, используемые АББД.

3.7 Многоуровневая защита. Контроль вывода

Контроль вывода данных из БД предполагает включение в подсистему защиты СУБД специальных средств, идентифицирующих и документирующих все операции вывода и учетную информацию пользователей БД, выполняющих эти операции. Одной из функций подсистемы контроля вывода может быть защита данных от копирования.

Под системой защиты от копирования понимается система, которая обеспечивает выполнение программой своих функций только при опознании некоторого уникального некопируемого элемента. Таким элементом (называемым ключевым) может быть дискета, определенная часть ПЭВМ или специальное устройство, подключаемое к ней.

В системе защиты от копирования обычно выполняются следующие процедуры:

- идентификация средств, из которых будет запускаться программа;
- аутентификация среды, из которой запущена программа;
- реакция на запуск из несанкционированной среды;
- регистрация санкционированного копирования;
- противодействие изучению работы алгоритмов функционирования системы.

Под средой, из которой будет запускаться программа, подразумевается дискета или ПЭВМ (при установке программы на жестком диске).

Идентификация заключается в закреплении за средой некоторых специально созданных или измеренных редко повторяющихся и трудно подделываемых характеристик – идентификаторов.

Идентификация дискеты может быть проведена двумя методами:

- нанесением повреждений на некоторую часть поверхности дискеты и контроль за ними в дальнейшем («лазерная дыра»);

- нестандартное форматирование дискеты и контроль его в дальнейшем.

Реакция на запуск обычно сводится к выдаче соответствующего сообщения.

Регистрация санкционированного копирования заключается в изменении счетчика установок.

При реализации перечисленных процедур выполняются следующие операции:

- создание дискет, защищенных от копирования;

- проверка ключевой дискеты;
- установка программ на ПЭВМ;
- защита программ от отладчика и модификаций.

Функция установки программы на жесткий диск состоит в «привязке» программы к конкретной ПЭВМ. Существует четыре способа реализации данной функции.

- работа защищенной программы только при наличии ключевой дискеты;
- закрепление за программой конкретного местоположения на диске;
- сохранение сигнатуры (специфичных только для данной ПЭВМ характеристик) при установке программы на диск;
- привязка к аппаратному ключу при установке программы на диск.

Простейший аппаратный ключ – это пассивное устройство, в состав которого входит постоянное запоминающее устройство, подключаемое к интерфейсу ввода-вывода.

Существует несколько базовых подходов к нестандартному форматированию дискет.

Нарушение последовательности секторов. Внесенные изменения проверяются при выполнении команды «Чтение идентификатора».

Изменение межсекторной дистанции. Относительные промежутки между секторами вычисляются на основе измерения временных интервалов между последовательно выполняемыми командами «Чтение идентификатора».

Форматирование с кодом длины 0 или 1. При форматировании дорожки код длины сектора выбирается равным 0 или 1, однако коды длины, записываемые в параметры CYL, HEAD, SEC и NO секторов, могут быть другими. Этот подход позволяет создавать резервные сектора для записи информации о ключевой дискете.

Контроль длины дорожек. При форматировании дорожки последний сектор записывается с кодом длины большим, чем код длины форматирования.

Наиболее распространенными методами вскрытия систем защиты от копирования являются:

- создание копий ключевой дискеты;
- модификация кода программы с целью обхода проверки;
- моделирование обращений к ключевой дискете;
- использование аппарата установки/снятия;
- снятие программы из памяти.

4. Теоретические основы безопасности в СУБД

4.1 Критерии защищенности баз данных

База данных может считаться защищенной, если для нее обеспечиваются конфиденциальность, целостность и доступность. Перечисленные аспекты информационной безопасности обеспечиваются средствами СУБД и могут рассматриваться в качестве критериев защищенности БД.

Политика безопасности определяется администратором данных, при этом функции защиты данных не должны быть ограничены только рамками СУБД и создаваемого с помощью ее приложения. Абсолютная защита данных практически нереализуема, поэтому обычно ограничиваются относительной защитой информации – гарантированно защищают ее на тот период времени, пока несанкционированный доступ к ней влечет какие-либо последствия. В ряде случаев для реализации функций защиты дополнительная информация может быть запрошена из операционных систем, в окружении которых работают сервер баз данных и клиент, обращающийся к этому серверу.

При рассмотрении с более конкретных позиций защищенность БД может считаться обеспеченной, если выполняются следующие условия:

- содержимое таблиц БД закодировано, что обеспечит невозможность просмотра файлов с помощью утилит чтения файлов или других аналогичных средств;
- пользователи после входа в сеть должны еще раз идентифицироваться перед открытием файла БД. Необходимо ввести секретный пароль, отличающийся от пароля для доступа к сети. Файл БД с информацией о пользователях и их паролях должен быть закодирован, при этом должен быть применен способ кодирования, обладающий надлежащей крипто-устойчивостью. Доступ к этому файлу может иметь только сетевой администратор. Использование отдельного пароля для файла БД обычно практикуется в качестве обязательной меры для общедоступных БД в одноранговых сетях, которые имеют простые механизмы защиты. В архитектуре «клиент-сервер» использование интегрированной защиты Windows для доступа к БД с проверкой учетной записи доменом операционной системы может считаться достаточным средством защиты;
- пользователи должны иметь специальное разрешение на доступ к БД и содержащимся в ней таблицам. Если отдельным пользователям запрещен просмотр определенных столбцов таблицы, то для их работы должны быть предусмотрены запросы, включающие только разрешенные для просмотра столбцы таблицы. В случае необходимости полномочия на доступ к тем или иным объектам могут быть сняты с помощью средств СУБД;
- для табличных данных должна быть предусмотрена возможность их учета и контроля. В противном случае это может явиться стимулом к «компьютерным кражам». Обновления, проводимые пользователями в таблицах, содержащими финансовые данные, должны быть отмечены в файле регистрации, который желательно хранить в другой БД. В нем должна сохраняться информация об идентификации пользователя, вносившего изменения, а также дата и время внесения этих изменений. Файлы регистрации полезны также при восстановлении записей БД, сделанных между последним резервным копированием БД и моментом восстановления БД из резервной копии;
- операции, обновляющие данные в нескольких таблицах, должны выполняться с помощью транзакций, которые можно отменить (выполнить откат транзакции), если обновления всех участвующих в транзакции таблиц не могут быть выполнены сразу.

Чаще всего причиной отказа системы защиты БД является неполное или халатное выполнение своих функций администратором БД, пренебрегающим реализацией перечисленных мероприятий.

4.2 Многоуровневая модель безопасности баз данных

Сочетание средств проверки полномочий и проверки подлинности субъектов доступа является мощным средством обеспечения безопасности информационных систем и БД. Если все пользователи, работающие в интерактивном режиме или запускающие пакетные приложения, достаточно надежны и имеют доступ к максимально закрытой информации, хранимой в системе, то указанный подход к обеспечению безопасности может быть вполне достаточным. Например, если в системе хранится информация, классифицированная по уровням от полностью открытой до совершенно секретной, но все пользователи системы имеют доступ к самым секретным данным, то в такой системе достаточно иметь надежные механизмы проверки полномочий и проверки подлинности. Такая модель называется «работой на высшем уровне (секретности)» (running at system high).

Однако она оказывается неудовлетворительной, если в организации необходимо создать действительно многоуровневую среду защиты информации. Многоуровневая защита создается, если в вычислительной системе хранится информация, относящаяся к разным классам секретности, но часть пользователей не имеет доступа к максимально секретному классу информации.

Классический пример подобной среды – вычислительная система закрытого учреждения, в БД которой (централизованной или распределенной) может содержаться информация от полностью открытой до совершенно секретной. При этом права пользователей могут быть разными – от допуска только к несекретной информации до допуска к совершенно секретным данным. Таким образом, пользователь, имеющий низший статус благонадежности, может выполнять свою работу в системе, содержащей сверхсекретную информацию, но ни при каких обстоятельствах не должен быть допущен к ней.

Многоуровневая защита БД строится обычно на основе модели Белл-ЛаПадула (Bell-LaPadula). Эта модель предназначена для управления субъектами (активными процессами, запрашивающими доступ к информации) и объектами (файлами, представлениями, записями, полями или другими элементами данной информационной модели).

В рассматриваемой модели объекты доступа подвергаются классификации, при этом сформированные классы называются классами доступа.

Субъекты доступа причисляются к одному из уровней благонадежности (clearance), которые называются просто уровнями.

Класс доступа характеризуется двумя компонентами. Первый компонент определяет иерархическое положение класса. Второй компонент представляет собой множество элементов из неиерархического набора категорий, которые могут относиться к любому уровню иерархии. Например, в частной компании может применяться следующая иерархия классов (сверху вниз):

- секретно;
- для ограниченного распространения;
- конфиденциально;
- для служебного пользования;
- несекретно.

Второй компонент в такой компании мог бы принадлежать к набору категорий:

- недоступно для части сотрудников;
- финансовая информация компании;
- информация об окладах.

Очевидно, что можно определить матрицу соотношений между иерархическими и неиерархическими компонентами. Однако это не решает проблему многоуровневой защиты. Например, если некоторый объект классифицирован как совершенно секретный, но ему не приписана ни одна из категорий неиерархического набора, то он может предоставляться иностранным правительствам, в то время как менее секретный объект может иметь категорию «недоступно для иностранных правительств» и, следовательно, не должен им предоставляться. Для исключения приведенных в примере аномалий в модели Белл-ЛаПадула (рис. 4.1) создается «решетка», где неиерархические компоненты каждого уровня иерархии автоматически приписываются и всем более высоким уровням (так называемое «обратное наследование»).

Из вышеизложенного следует, что субъект класса CL1 имеет доступ к объекта класса CL2, если CL2 не выше CL1. Например, пользователь класса «совершенно секретно» имеет доступ к информации классов «совершенно секретно», «секретно», «конфиденциально», «несекретно». В модели Белл-ЛаПадула этот принцип известен под названием простого свойства секретности (simple security property).

Менее очевидно другое свойство, обозначаемое символом λ, которое позволяет субъекту иметь право на запись в объект только в том случае, если класс субъекта такой же или ниже, чем класс объекта для записи данных. Это означает, что информация, принадлежащая какому-либо уровню секретности, никогда не может быть записана в какой-либо объект, имеющий более низкий уровень секретности, поскольку это могло бы привести по неосторожности к деградации защиты классифицированной информации.

Рис.4.1. Решетка классов доступа в модели Белл-ЛаПадула.

С учетом двух принятых в модели Белл-ЛаПадула ограничений (простое свойство секретности и λ - свойство) можно представить себе БД с многоуровневой защитой.

Многозначность. Решение проблемы поэлементной классификации, указанной выше, основано на идее многозначности, когда в рамках одного отношения может существовать множество кортежей с одним и тем же значением первичного ключа.

При отсутствии многозначности отношение с многоуровневой защитой, следовало бы обрабатывать следующим образом.

В БД можно маскировать значения, недоступные пользователю или процессу по соображениям безопасности, и такое маскирование обычно осуществляется при помощи пустых значений (null values). В этом случае процесс вместо секретных элементов будет видеть пустые значения.

Каждый кортеж существует в единственном экземпляре, поэтому непривилегированный процесс может попытаться записать неклассифицированные данные в столбцы кортежа, содержащие как бы пустые значения. В связи с этим в СУБД с многоуровневой защитой возникает проблема обработки таких операций, поскольку на самом деле значение столбца непустое, т.е. замещать его нельзя.

Естественно, что подобные попытки должны отвергаться средствами системы безопасности.

Однако неудача при выполнении операции модификации открывает определенные возможности, получившие название косвенных каналов (для получения закрытой информации), т.е. пути для снижения классификации данных (преднамеренного или случайного, что будет обсуждаться далее).

Отказ в выполнении вполне вроде бы законной транзакции на модификацию пустых значений в БД с многоуровневой защитой выдает информацию о том, что эти элементы на самом деле не пусты, а содержат классифицированные данные. Уведомляя не наделенного соответствующими полномочиями пользователя или процесс о наличии в том или ином кортеже классифицированных данных, мы открываем тем самым косвенный канал.

Допустим, что СУБД с многоуровневой защитой симулирует выполнение подобных модификаций, чтобы не раскрывать скрытый канал. Но что будет, если пользователь, только что изменивший некоторые данные, попытается выполнить запрос или приложение, которое обращается к этим данным, а ему ответят, что они отсутствуют? Значит, опять создается косвенный канал. Можно попытаться поддерживать в СУБД с многоуровневой защитой нечто вроде журнала таких модификаций и использовать его для последующих выборок.

Важно отметить, что многозначность – это существенное свойство данных с многоуровневой защитой, а как она будет реализована – это вопрос второстепенный, не имеющий отношения к самой идее по-разному представлять информацию для пользователей разных уровней благонадежности.

Итак, многозначность стала общепринятой моделью реализации БД с многоуровневой защитой, в особенности реляционных. Направления будущего развития средств безопасности СУБД тесно связаны с моделями многоуровневой защиты, в которых, в силу проблемы косвенных каналов, важную роль будет играть реализация многозначности.

Косвенные каналы. В определениях высших уровней безопасности компьютерных систем и БД по классификации «Оранжевой книги» проблеме косвенных каналов уделяется большое значение. **Косвенный канал** – это механизм, посредством которого субъект, обладающий высоким уровнем благонадежности, может предоставлять информацию менее благонадежным субъектам. Согласно λ-свойству модели Белл-ЛаПадула, непосредственная запись информации в объекты более низкого класса секретности запрещена и косвенные каналы обычно реализуются при помощи непрямых способов передачи информации.

«Оранжевая книга» идентифицирует два типа косвенных каналов:

- косвенные каналы памяти;
- косвенные каналы времени.

Косвенный канал памяти открывается, когда некоторая область памяти, например, содержащая объект БД, прямо или косвенно используется как средство раскрытия классифицированных материалов. Выше уже рассматривались пути возникновения косвенных каналов памяти. Без соблюдения λ-свойства Белл-ЛаПадула классифицированные данные легко могли бы быть скомпрометированы путем явной записи, например, посредством Троянского коня, сверхсекретных данных в область, относящуюся к низшему классу секретности, где ее потом может прочитать менее благонадежный пользователь или процесс.

Но и при соблюдении λ - свойства Белл-ЛаПадула информация может быть скомпрометирована косвенными способами, опять же при посредстве Троянского коня. Когда в условиях отсутствия многозначности для скрытия секретной ин-формации от недостаточно благонадежных процессов используются пустые значения, то отказ, полученный низкоуровневым процессом при попытке модифицировать эти как бы пустые значения, открывает косвенный канал. Тем самым предоставляется информация о наличии секретных данных, маскируемых пустыми значениями (хотя содержание информации при этом не раскрывается). Однако при той же модели, путем кооперирования встроенного Троянского коня с низкоуровневым процессом, можно организовать передачу реальной классифицированной информации при помощи некоторого сигнального механизма. Троянский конь может открыть транзакцию, создать фиктивную таблицу известной структуры, заполнив ее фиктивными классифицированными данными и установив маски пустых значений. Низкоуровневый процесс затем мог бы в определенное время предпринять серию модификаций по отношению к этой таблице, интерпретируя коды результатов успешного или неуспешного завершения по отношению к выбранным строкам в соответствии с установленным кодом. Код может быть просто последовательностью бит ASCII-кодировки интересующих секретных данных. Например, для каждой восьмерки строк в таблице коды результата модификации будут интерпретироваться как 8-битный код одного символа. Могут применяться и более сложные коды. Закончив передачу данных по такому косвенному каналу, высокоприоритетный процесс может затем уничтожить (DROP) таблицу.

Описанный выше канал вполне может быть обнаружен. Это возможно с помощью журналов БД, в которых будут зафиксированы факты создания и уничтожения таблиц. Однако существуют и другие аналогичные способы, использующие уже имеющиеся таблицы. Поэтому политика безопасности организации должна учитывать подобные факторы.

Косвенные каналы времени до некоторой степени также имеют отношение к БД. При помощи неклассифицированного или низкоуровневого процесса сетевого анализа можно наблюдать закономерности трафика и времена отклика для всех системных операций. Отметив, сколько времени занимает передача несекретных данных известного объема (например, списка личного состава), и сравнивая эти данные со временем передачи классифицированной информации, можно оценить объем передаваемых секретных данных. Но даже и не имея сведений о передаче не-секретных данных, которые можно использовать для сравнения, и о структуре передаваемых секретных данных, но зная пропускную способность сети, протоколы (для определения накладных расходов) и другие характеристики сети, можно анализировать секретные передачи для оценки объемов данных. Сравнивая наблюдения за некоторый период, можно обнаружить те или иные тенденции.

Косвенные каналы времени обычно исключаются за счет соответствующих решений в сфере коммуникаций, например за счет заполняющего трафика (traffic padding), позволяющего замаскировать истинные объемы передаваемой информации. Таким образом, косвенные каналы времени – это, прежде всего, вопрос коммуникаций. Но по мере того, как растет степень распределенности БД, и возрастает значение мер безопасности, в надежной среде БД эти вопросы должны рассматриваться наряду с проблемами косвенных каналов памяти.

Приведем пример косвенного канала. Поскольку высокоуровневым процессам разрешено читать данные из объектов низкой классификации, может возникнуть следующая ситуация. Предположим, что высокоуровневый процесс посылает следующий SQL-запрос удаленной низкоуровневой СУБД:

```
SELECT name, rank, date_of_birth
FROM roster
WHERE duty_building=445
```

Возможно, это всего-навсего безобидный запрос на сведения обо всех сотрудниках, работающих в здании номер 445, но это может быть и запрос, сгенерированный Троянским конем, в котором в явном виде содержится секретная информация. Возможно, это сообщение о том, что 445 танков или вооруженных лиц пересылаются в определенный район для выполнения секретной миссии или число 445 имеет еще какой-то скрытый смысл (возможно, интерпретируемый при посредстве механизмов дешифрования, с помощью битовой маски и т. п.). Эта проблема исключительно важна, и она подводит к еще одной теме – безопасность среды распределенной БД. Все вопросы и направления исследований, которые рассматривались до сих пор, относились к БД вообще. Свойство распределенности БД создает дополнительные проблемы и угрозы безопасности. Состояние дел в области безопасности распределенных БД значительно отстает по сравнению с положением для централизованных СУБД. Тем не менее, на решение этих вопросов в настоящее время направлены значительные исследовательские усилия.

5. Механизмы обеспечения целостности в СУБД

5.1 Понятие целостности базы данных

Под **целостностью БД** понимается ее состояние, в котором данные являются логически согласованными. Только в этом случае данные представляют какую-либо ценность для пользователей.

На практике содержимое БД изменяется во времени в результате выполнения различных операций над данными. При осуществлении таких операций оказывается принципиально невозможным избежать состояния БД, когда содержащиеся в ней данные находятся в логически несогласованном или, иначе, не целостном состоянии.

Пример 5.1

Пусть в БД имеются отношение СТУДЕНТ с двумя атрибутами ИМЯ_СТУДЕНТА и СРЕДНИЙ_БАЛЛ и отношение УСПЕВАЕМОСТЬ с атрибутами ИМЯ_СТУДЕНТА, ДИСЦИПЛИНА, ОЦЕНКА. Значение атрибута СРЕДНИЙ_БАЛЛ отношения СТУДЕНТ представляет собой среднее значение оценок конкретного студента, полученных им по всем сданным дисциплинам. Если в отношение УСПЕВАЕМОСТЬ производится ввод данных об оценке, полученной студентом по сданной им новой дисциплине, то помимо ввода соответствующего нового кортежа в отношение УСПЕВАЕМОСТЬ в отношении СТУДЕНТ должно быть скорректировано и значение атрибута СРЕДНИЙ_БАЛЛ для этого студента. Очевидно, что из-за последовательного выполнения этих двух действий имеется момент времени, когда БД находится в рассогласованном, не целостном состоянии, т.е. когда находящееся в БД значение среднего балла студента не соответствует его новому фактическому значению после ввода новой оценки.

Как следует из примера 5.1, в процессе выполнения операций над хранимыми данными БД переходит из одного согласованного состояния в другое согласованное состояние. Между двумя этими состояниями данные в БД могут быть рассогласованными. Потенциальная опасность такой ситуации состоит в том, что возможны случаи, когда вследствие внешних причин (например, аварийного сбоя системы) операция по модификации данных не выполняется до конца и БД остается в рассогласованном состоянии. Сложности возникают также при одновременном выполнении операций по модификации данных несколькими пользователями одной БД.

5.2 Основные причины возникновения угроз целостности

Существует множество различных типов отказов, способных повлиять на функционирование БД. Каждый тип отказа требует особых способов обработки. Одни отказы влияют только на содержимое оперативной памяти, другие могут воздействовать и на внешнюю память. Наиболее частыми причинами отказов, приводящих к нарушению целостности БД, являются:

- аварийное прекращение работы системы, вызванное ошибкой оборудования или программного обеспечения, приведшей к разрушению содержимого оперативной памяти;
- отказ носителей информации, например, разрушение магнитной головки или появление неустранимого сбоя чтения, вследствие чего оказывается потерянной часть данных на внешнем запоминающем устройстве;
- ошибки прикладных программ, например, логические ошибки в программах, получающих доступ к БД, послужившие причиной сбоев при выполнении операций;
- отказы в сети электропитания, диверсии или стихийные бедствия (пожары, наводнения, землетрясения и т.п.);
- небрежное или легкомысленное обращение со стороны операторов или пользователей системы, послужившее причиной непреднамеренного разрушения данных или программ;
- преднамеренное разрушение или уничтожение данных, оборудования или программного обеспечения.

Какой бы ни была причина отказа системы, результатом может быть утрата содержимого оперативной памяти, включая буфер данных, и утрата копии БД на дисках.

5.3 Ограничения целостности в реляционной модели

В реляционную модель входят три составные части: описание структуры данных, описание операций, которые могут выполняться над данными, и ограничения целостности

данных. В данной лекции рассмотрим третью составляющую модели – ограничения целостности.

Очевидно, что БД представляет для пользователя какую-либо ценность до тех пор, пока хранимые в ней данные будут правильными, т.е. будут корректно отражать информацию о соответствующей предметной области. В связи с этим в СУБД желательно иметь средства, позволяющие контролировать корректность хранимых данных. Для осуществления такого контроля в системе необходимо наличие дополнительной информации, указывающей, какие данные считать правильными, а какие – неправильными. В реляционной модели такого рода информация указывается в ограничениях целостности данных, под которыми понимают набор условий и логических ограничений, которым должны удовлетворять данные, хранимые в БД.

Любое ограничение целостности является семантическим понятием. Оно отражает определенные свойства объектов-сущностей предметной области, информация о которых не может быть получена из анализа хранимых в базе значений данных. Например, из анализа фактически присутствующих в отношении значений атрибута нельзя сделать вывод о том, какие значения должны считаться правильными. То, что значение атрибута ВОЗРАСТ не может быть отрицательным числом, является внешним ограничением, накладываемым на значения атрибута исходя из семантики (смысла) предметной области.

БД находится в согласованном состоянии, если все находящиеся в ней данные удовлетворяют заданным для этой БД ограничениям целостности данных.

СУБД, имеющая средства поддержки ограничений целостности, реализует проверку и обеспечение согласованности и целостности хранимых данных лишь в той мере, в которой эти данные удовлетворяют заданным в системе ограничениям целостности. Поэтому более точно следует говорить, что СУБД обеспечивает не целостность данных вообще, а выполнение заданных в БД ограничений целостности данных.

Обеспечение целостности БД усложняется тем, что данные в ней могут изменяться во времени. Вследствие этого с ограничениями целостности должен быть связан набор правил и определенных действий, которые должны выполняться при модификации данных для обеспечения их корректности. В связи с этим возникает понятие реакции системы на попытку нарушения целостности данных. При этом возможны два варианта реакции системы: можно отвергнуть попытку нарушения целостности, а можно выполнить какие-либо действия, компенсирующие нарушение целостности.

Действия системы по проверке ограничений целостности можно представить в виде, изображенном на рис. 5.1.

Очевидно, что реакция системы в случае осуществления действий по модификации данных в большой степени зависит от вида затрагиваемых этими действиями ограничений целостности.

Ограничения целостности разделяют на четыре группы:

- уровня атрибута;
- уровня кортежа;
- уровня отношения;
- уровня базы данных.

Рассмотрим эти виды ограничений целостности более подробно.

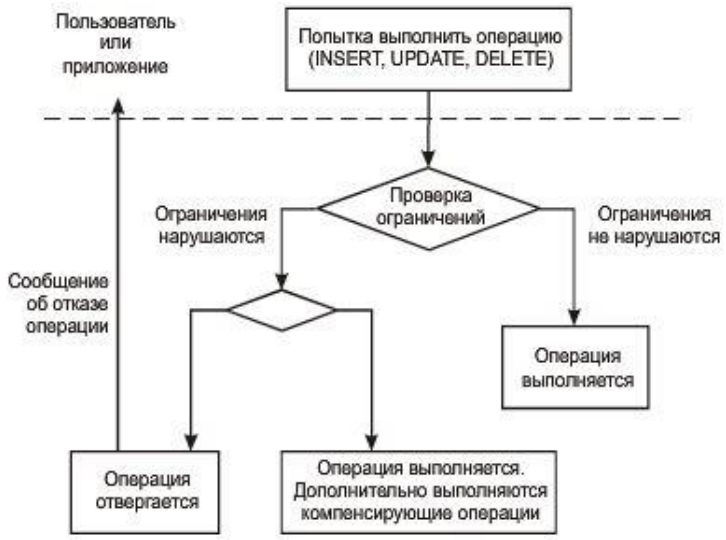

Рис. 5.1. Работа системы по проверке ограничений целостности.

5.3.1 Ограничения целостности уровня атрибута

Ограничения целостности уровня атрибута являются ограничениями, накладываемыми на допустимые значения атрибута. Эти ограничения исходят из того, что каждый атрибут определен на каком-либо конкретном домене.

Описание для атрибута его домена позволяет осуществлять контроль правильности вводимых в базу значений атрибутов.

Ограничение целостности атрибутов выглядит следующим образом: значения каждого атрибута берутся из соответствующего домена.

Домены ограничивают также операции сравнения значений атрибутов, поскольку сравнивать между собой можно только значения атрибутов, определенных на общих доменах.

В реальной жизни возможны ситуации, когда конкретные значения атрибута могут отсутствовать. Для обозначения отсутствующей информации о значении атрибута используют специальный маркер – NULL. Он представляет собой обозначение факта отсутствия действительного значения этого атрибута, которое на самом деле существует, но является в данный момент неизвестным или неопределенным.

При выполнении различных скалярных операций над значениями атрибутов, которые могут быть неизвестны или не определены, возникают проблемы, усложняющие СУБД.

5.3.2 Ограничения целостности уровня кортежа

Ограничения целостности уровня кортежа представляют собой ограничения, накладываемые на допустимые значения отдельного кортежа отношения и не являющиеся ограничением целостности атрибута. То, что ограничение наложено на кортеж, означает, что для его проверки не требуется никакой информации о других кортежах.

Примером такого рода ограничений в отношении АУДИТОРНЫЙ_ФОНД, имеющем следующие атрибуты НОМЕР_КОМНАТЫ, ДЛИНА, ШИРИНА, ВЫСОТА, ПЛОЩАДЬ, ОБЪЕМ, являются следующие ограничения, наложенные на значения атрибутов для каждого кортежа:

ПЛОЩАДЬ = ДЛИНА * ШИРИНА

ОБЪЕМ = ДЛИНА * ШИРИНА * ВЫСОТА

Эти ограничения связывает между собой значения атрибутов в каждом кортеже. Они не могут быть выражены с помощью задания доменов отношений и при этом для своей

проверки для конкретного кортежа не требует привлечения информации о других кортежах отношения.

5.3.3 Ограничения целостности уровня отношения

Ограничения целостности уровня отношения представляют собой ограничения, накладываемые только на допустимые значения отдельного отношения и не являющиеся ограничением целостности кортежа или атрибута. Это означает, что такого рода ограничение не может быть сведено к ограничению, накладываемому на кортеж или атрибут, и, с другой стороны, для его проверки не требуется информация о других отношениях БД.

К важным видам такого рода ограничений относятся ограничения потенциальных ключей, называемые еще ограничением целостности сущностей, а также ограничения, задаваемые функциональными зависимостями между атрибутами, многозначными зависимостями, зависимостями проекции-соединения.

К ограничениям отношения относится, например, требование, накладываемое на отношение СТУДЕНТЫ, имеющее атрибут НОМЕР_ГРУППЫ, состоящее в том, чтобы в каждой учебной группе было не меньше 25 студентов. Очевидно, что для обеспечения выполнения такого ограничения для конкретного кортежа необходимо использование информации о значениях данного атрибута в других кортежах.

В приложении к реляционным отношениям различают несколько видов ключей, в частности потенциальные (возможные), первичные и альтернативные ключи.

Пусть r – некоторая переменная отношения. Тогда потенциальный ключ (candidate key) K для отношения r – это подмножество множества атрибутов отношения r, всегда обладающее следующими свойствами:

- уникальности. Это свойство означает, что нет двух кортежей в текущем значении переменной отношения r с одинаковым значением K;
- неизбыточности. Никакое из подмножеств K не обладает свойством уникальности.

Иными словами, потенциальным ключом отношения называется его атрибут или набор (подмножество) атрибутов, значения которого (которых) позволяют однозначно идентифицировать кортежи отношения вследствие своей уникальности.

Из приведенного определения следует, что понятие потенциального ключа относится не к конкретному содержимому отношения в некотором текущем состоянии, а ко всем возможным значениям, которые может принимать переменная отношения.

В отношении может быть несколько потенциальных ключей. Например, в отношении СТУДЕНТ имеется два потенциальных ключа, а именно

КОД_СТУД и ПАСПОРТ.

Значения каждого из этих атрибутов могут служить идентификаторами кортежей, так как каждый из них однозначно и, очевидно, неизбыточно определяет запись о конкретном студенте. Действительно, не может быть двух студентов с одинаковыми кодами, так же как и с одинаковыми номерами паспортов.

В общем случае ключ может включать в себя не один, а несколько атрибутов. Например, в отношении УСПЕВАЕМОСТЬ потенциальный ключ включает в себя два атрибута КОД_СТУД,ДИСЦИПЛИНА, причем, в отличие от предыдущего примера, каждый из этих атрибутов в отдельности не является ключом.

Свойством уникальности обладает именно пара значений этих атрибутов (конкретный студент по конкретной дисциплине на экзамене получает единственный билет и только одну оценку).

Потенциальный ключ, имеющий в своем составе больше, чем один атрибут, называется составным. Потенциальный ключ, состоящий из единственного атрибута, называется простым.

Обратим внимание также на требование неизбыточности потенциального ключа. Это требование говорит о том, что не всякое множество атрибутов, совокупность значений которых обладает уникальностью и позволяет идентифицировать кортежи, является потенциальным ключом.

Действительно, в приведенном выше отношении УСПЕВАЕМОСТЬ сочетания трех атрибутов КОД_СТУД, ДИСЦИПЛИНА, ОЦЕНКА или КОД_СТУД, ДИСЦИПЛИНА, №БИЛЕТА, а также всех четырех атрибутов КОД_СТУД, ДИСЦИПЛИНА, №БИЛЕТА, ОЦЕНКА тоже позволяют однозначно идентифицировать кортежи отношения (первое свойство потенциального ключа). Однако эти наборы атрибутов не являются потенциальными ключами из-за своей избыточности для целей идентификации кортежей.

Важность потенциальных ключей в реляционной модели состоит в том, что они обеспечивают механизм адресации на уровне кортежей.

Единственный гарантированный способ точно указать на какой-нибудь кортеж – это указать значение некоторого потенциального ключа.

В реляционной модели по традиции один из потенциальных ключей выбирают в качестве первичного ключа (primary key), в таком случае остальные потенциальные ключи (при их наличии) будут называться альтернативными ключами (alternate key).

Выбор атрибута или набора атрибутов в качестве потенциального ключа отношения, означающий наложение на этот набор требования уникальности комбинации его значений, является одним из важных ограничений целостности атрибутов. Очевидно также, что это ограничение является ограничением уровня отношения, так как для его поддержки необходимо знание значений атрибутов для всех кортежей отношения. Ограничение потенциального ключа еще называют ограничением целостности сущности, т.е. каждая представленная в БД сущность должна быть идентифицируема.

Указание такого ограничения подразумевает, что, сообщая системе о том, что данный набор атрибутов является первичным или потенциальным ключом, мы вправе ожидать от нее действий, контролирующих уникальность значений этого набора атрибутов.

Ни один из атрибутов, составляющих первичный ключ, не может принимать NULL-значения. Для атрибутов альтернативных ключей NULL- значения могут допускаться, при этом, однако, с помощью значений таких ключей можно будет идентифицировать уже не все кортежи, а только кортежи с реально присутствующими значениями атрибутов альтернативного ключа.

Ограничения целостности уровня БД рассмотрены ниже при описании ссылочной целостности.

5.4 Декларативная и процедурная поддержка ограничений целостности в СУБД

Различают два способа реализации механизма обеспечения целостности данных в СУБД:

- декларативную поддержку ограничений целостности;
- процедурную поддержку ограничений целостности.

Декларативная поддержка ограничений целостности заключается в описании ограничений целостности средствами языка определения данных. Обычно средства декларативной поддержки определяют ограничения на значения доменов и атрибутов, целостность сущностей (потенциальные ключи отношений) и ссылочную целостность (целостность внешних ключей). Стандартом языка SQL предусмотрен широкий набор средств декларативной поддержки ограничений целостности.

Процедурная поддержка ограничений целостности заключается в использовании программного кода, реализуемого в СУБД в виде так называемых хранимых процедур и триггеров.

Под хранимыми процедурами понимают процедуры и функции (программный код), хранящиеся непосредственно в БД в откомпилированном виде и которые могут запускаться пользователями или приложениями, работающими с БД.

5.4.1 Применение триггеров

Триггеры представляют собой хранимые процедуры, особенность которых состоит в том, что их запуск связан с некоторыми событиями, происходящими во время работы БД. Триггер всегда реализует действие, связанное с событием, вызвавшим изменение в содержимом таблицы. В качестве таких событий выступают операции вставки, обновления и удаления строк таблиц.

С точки зрения реализации триггер представляет собой команду на языке SQL, которая автоматически выполняется СУБД при внесении некоторого изменения в указанную таблицу. В отличие от обычной SQL- подпрограммы триггер не имеет аргументов и выполняется неявно в каждом случае возникновения соответствующего события.

Если в БД определен некоторый триггер, то он запускается автоматически всегда при возникновении события, с которым он связан. Очень важным является то, что триггер не может быть обойден пользователем.

Он выполняется всегда и независимо от того, кто из пользователей и каким способом инициировал событие, вызвавшее его запуск.

Триггеры могут использоваться для достижения следующих целей:

- проверки корректности введенных данных и проверки выполнения сложных ограничений целостности данных, которые трудно или вообще невозможно поддерживать с помощью ограничений целостности, установленных для таблицы;
- выдачи предупреждений (например, с помощью электронной почты), которые напоминают о необходимости выполнить некоторые действия при обновлении таблицы, выполненном определенным образом;
- накопления аудиторской информации посредством фиксации сведений о внесенных изменениях и тех лицах, которые их выполнили;
- поддержки репликации.

Основное назначение триггеров – автоматическое выполнение действий по поддержанию целостности БД.

Триггеры могут быть как достаточно простыми, например, поддерживающими ссылочную целостность, так и довольно сложными, реализующими какие-либо сложные ограничения предметной области или сложные действия, которые должны произойти при наступлении некоторых событий.

Обычно хранимые процедуры и триггеры пишутся либо на специальном процедурном расширении языка SQL (например, PL/SQL для СУБД Oracle или Transact-SQL для MS SQL Server), или на некотором универсальном языке программирования, например C++, с включением в код операторов SQL в соответствии со специальными правилами такого включения.

Использование для обеспечения целостности триггеров и хранимых процедур более сложно, чем использование декларативных средств языка SQL, так как предполагает более низкоуровневые средства формирования программного кода и более высокие требования к квалификации разработчика БД. Однако триггеры позволяют реализовать в СУБД обработку гораздо более изощренных ограничений целостности данных, нереализуемых с использованием стандартных встроенных средств декларативной поддержки целостности.

Вообще говоря, наличие ограничения целостности (как декларативного, так и процедурного характера) всегда приводит к созданию или использованию некоторого

программного кода, реализующего это ограничение. Разница заключается в том, где такой код хранится и как он создается.

И в этом плане наличие, развитость и уровень реализации в конкретной СУБД средств описания и обеспечения целостности данных являются одним из важных показателей ее уровня с точки зрения возможностей и эффективности реализации на ее основе информационных систем с БД.

При реализации в СУБД ограничений целостности в виде триггеров соответствующий программный код является просто телом триггера. Если используется декларативное задание ограничений целостности, например средствами языка SQL, то возможны два подхода:

1. Использование для реализации ограничений функций, встроенных в ядро СУБД, с сохранением задаваемого при декларировании исходного текста ограничения в виде некоторого объекта СУБД. В этом случае проверка ограничения выполняется функциями ядра СУБД (например, ядра Oracle) со ссылкой на этот объект. Ограничение целостности при этом нельзя модифицировать иначе, как путем использования декларативных операторов создания и модификации ограничений.

2. Автоматическая генерация программного кода триггеров, выполняющих необходимые действия по проверке ограничений, при декларировании ограничения СУБД. В некоторых СУБД, например Visual FoxPro, допускается последующее «ручное» редактирование автоматически сгенерированного кода триггера.

Если система не поддерживает ни декларативную поддержку ссылочной целостности, ни триггеры (как, например FoxPro 2.5, Clipper и др.), то программный код, следящий за корректностью БД, приходится размещать в пользовательском приложении. Это существенно затрудняет разработку программ и, главное, не защищает от возможности внесения пользователем некорректных данных напрямую в БД. Ситуация особенно усложняется в случае, когда с БД работает не одно, а множество различных приложений. Каждое из таких приложений должно содержать один и тот же код, отвечающий за поддержание целостности БД, который при необходимости изменения характера ограничения целостности необходимо модифицировать синхронно во всех приложениях.

5.4.2 Способы задания триггеров

Задание триггеров осуществляется с помощью команды CREATE TRIGGER, имеющей следующий формат:

CREATE TRIGGER имя_триггера
BEFORE|AFTER <триггерное событие> ON <имя таблицы>
[REFERENCING <список_старых_или_новых_псевдонимов>}
[FOREACH {ROW| STATEMANT}]
[WHEN (условие_триггера)]
<тело_триггера>

Триггерные события включают вставку, удаление и обновление строк в таблице. В последнем случае для триггерного события можно указать конкретные имена столбцов таблицы.

Время запуска триггера определяется с помощью ключевых слов BEFORE и AFTER. При указании ключевого слова BEFORE триггер запускается до выполнения связанных с ним событий, а при указании ключевого слова AFTER – после их выполнения.

Выполняемые триггером действия задаются SQL-командой, которая может быть выполнена одним из следующих способов:

- для каждой строки (FOR EACH ROW), охваченной данным событием (триггер на уровне строки);

- только один раз для каждого события (FOR EACH STATEMENT) (триггер на уровне команды). Этот способ выполнения команды используется по умолчанию.

Обозначение <список_старых_или_новых_псевдонимов> может относиться к строке или к таблице:

- старой или новой строке (OLD/NEW или OLD ROW/NEW ROW), если используется триггер на уровне строки;
- старой или новой таблице (OLD TABLE/NEW TABLE), если используется триггер AFTER.

Очевидно, что старые значения неприменимы для событий вставки, а новые – для событий удаления.

Тело триггера не может содержать SQL-команды:

- для обработки транзакций, например COMMIT и ROLLBACK;
- для организации подключения, например CONNECT и DISCONNECT;
- для определения схемы и управления ею, например команды создания или удаления таблиц, пользовательских типов или других триггеров;
- для организации сеанса, например SET SESSION CHARACTERISTICS, SET ROLE, SET TIME ZONE.

Поскольку для таблицы может быть определено несколько триггеров, то большое значение имеет порядок их запуска. Запуск триггеров происходит по мере возникновения триггерных событий (INSERT, UPDATE, DELETE) в следующем порядке:

- исполнение табличного триггера BEFORE на уровне команды;
- для каждой строки, охваченной данной командой:

1) исполнение любого триггера BEFORE на уровне строки;
2) исполнение данной команды;
3) исполнение любого триггера AFTER на уровне строки;

- исполнение табличного триггера на уровне команды.

5.5 Управление транзакциями для обеспечения целостности баз данных

5.5.1 Свойства транзакций

Под **транзакцией** понимается логическая единица работы СУБД, представляющая собой последовательность операторов манипулирования данными, выполняющаяся как единое целое и переводящая БД из одного согласованного состояния в другое.

В СУБД для обеспечения при модификации данных корректного перевода БД из одного согласованного состояния в другое применяется механизм управления транзакциями. Поддержание механизма транзакций является одним из важных показателей уровня развитости СУБД, обеспечения в БД целостности данных и их устойчивости к возможным сбоям аппаратных и программных средств и конфликтам при совместной работе с данными нескольких пользователей.

Транзакция характеризуется четырьмя важными свойствами:

- атомарности;
- согласованности;
- изолированности;
- долговечности.

Атомарность. Транзакция выполняется как неделимая операция – либо выполняется вся операция целиком, либо она целиком не выполняется («все или ничего»).

Согласованность. Транзакция переводит БД из одного согласованного (целостного) состояния в другое согласованное (целостное) состояние без обязательной поддержки согласованности данных во все промежуточные моменты времени.

Изолированность. Транзакции определены изолированно одна от другой. Это означает, что транзакции, инициированные разными пользователями, не должны влиять друг на друга.

Долговечность. Если транзакция выполнена, то результаты ее работы должны сохраниться в БД, даже если в следующий момент произойдет сбой системы.

При работе с транзакциями применяются команды:

- COMMIT – зафиксировать транзакцию;
- ROLLBACK – откатить транзакцию.

Обычно транзакция начинается автоматически с момента присоединения пользователя (запроса) к БД и продолжается до тех пор, пока не произойдет одно из следующих событий:

- подана команда COMMIT (зафиксировать транзакцию). В режиме AVTOCOMMIT это происходит автоматически при успешном выполнении запроса. Команда COMMIT завершает текущую транзакцию. При этом гарантируется, что результаты работы завершенной транзакции фиксируются, т.е. сохраняются в БД;
- подана команда ROLLBACK (откатить транзакцию). По этой команде результат работы неудачной транзакции, т.е. транзакции, в процессе выполнения которой возникла какая-либо ошибка, полностью аннулируются. Тем самым БД приводится к состоянию, в котором она находилась перед началом неудачной транзакции;
- произошел сбой системы. Система из-за сбоя прекратила работу до фиксации результатов транзакции по команде COMMIT или до ее отказа по команде ROLLBACK, т.е. БД оказалась в логически несогласованном состоянии. В этом случае СУБД должна привести БД в согласованное состояние при перезагрузке системы.

Наличие в СУБД механизма поддержки транзакций подразумевает, что система гарантирует нахождение БД в логически согласованном (целостном) состоянии до начала и после окончания выполнения транзакции или автоматическое приведение данных в целостное согласование при восстановлении работоспособности системы. Такая гарантия подразумевает обеспечение обратимости всех операций, осуществляемых в процессе транзакции.

Следует заметить, что не все СУБД и не всегда обеспечивают поддержку свойств транзакций в полном объеме. Далее проблемы реализации механизма управления транзакциями рассматриваются более подробно.

Эти проблемы можно разделить на две группы:

- обеспечение согласованности данных при сбоях системы (обеспечение восстановления данных);
- обеспечение согласованности данных при совместной работе с БД нескольких пользователей.

5.5.2 Откат транзакций и восстановление данных после сбоев. Журнализация изменений базы данных

Механизм управления транзакциями предполагает возможность восстановления согласованного состояния данных после любого нарушения возможности корректного завершения транзакции из-за возникновения ошибок в ходе ее выполнения или аппаратных или программных сбоев системы. Очевидно, что восстановление исходного согласованного состояния БД возможно только при наличии определенной дополнительной информации о процессе модификации данных в ходе выполнения транзакции. В современных реляционных СУБД такая дополнительная информация формируется в виде специального журнала изменений БД или журнала транзакций.

Общими принципами восстановления согласованного состояния данных являются:

- результаты зафиксированных (во внешней памяти) транзакций должны быть сохранены в восстановленном состоянии БД;

- результаты незафиксированных (во внешней памяти) транзакций должны отсутствовать в восстановленном состоянии БД.

Из сказанного следует, что восстанавливается последнее согласованное состояние БД.

Возможны следующие ситуации, при которых должно производиться восстановление прежнего состояния БД.

Индивидуальный откат транзакции. Откат транзакции может быть инициирован путем подачи команды ROLLBACK во время выполнения транзакции. Откат транзакции может быть инициирован самой СУБД в случае возникновения какой-либо ошибки (например, деления на ноль) или при выборе транзакции в качестве «жертвы» при обнаружении ситуации синхронизационного тупика (см. ниже).

При индивидуальном откате транзакции для восстановления согласованного состояния в БД должны быть устранены последствия действия операторов модификации БД, которые выполнялись в этой транзакции.

Восстановление после внезапной потери содержимого оперативной памяти (мягкий сбой системы). Такая ситуация может возникнуть при аварийном выключении электрического питания, при возникновении неустранимого сбоя процессора (например, срабатывании контроля оперативной памяти) и т.д. Данные, хранящиеся на диске, остаются неповрежденными, утрачивается содержимое буферов БД в оперативной памяти.

Восстановление после отказа дисков (жесткий сбой системы). Эта ситуация может возникнуть из-за разрушения структуры записанных на дисках данных, повреждения поверхности дисков, головок записи-чтения и т.д.

Во всех трех случаях основой восстановления данных является избыточное хранение данных. Такая избыточная, необходимая для восстановления данных информация записывается в специальном журнале, и представляет собой последовательность записей о произведенных в ходе транзакции изменениях БД.

Журнализация изменений БД тесно связана не только с механизмом управления транзакциями, но и с буферизацией страниц БД в оперативной памяти. Проблема восстановления данных решилась бы гораздо проще, если бы все изменения данных сразу же фиксировались во внешней памяти системы. Такое решение, однако, при объективно существующей существенной разнице в скорости доступа к данным в оперативной памяти и внешней памяти привело бы к резкому падению эффективности системы.

Буферизация страниц БД в оперативной памяти является единственным реальным способом достижения удовлетворительной производительности СУБД.

Аналогичная проблема имеет место и при записи в журнал транзакций информации о том, какие изменения данных были осуществлены в ходе транзакции. Немедленная фиксация этих записей во внешней памяти также привела бы к существенному замедлению системы. Поэтому записи в журнал также буферизуются в оперативной памяти: при нормальной работе системы очередная страница буфера записей журнала выталкивается во внешнюю память журнала только при ее полном заполнении записями.

Таким образом, система поддерживает два типа буферов в оперативной памяти – буферы страниц БД и буферы журнала транзакций.

Страницы БД, содержимое которых в буфере (в оперативной памяти) отличается от содержимого страниц во внешней памяти, называют «грязными» (dirty) страницами. Система должна поддерживать список «грязных» страниц и обеспечивать время от времени их выталкивание во внешнюю память, осуществляя тем самым фиксацию изменений БД.

Очевидно, что описанная ситуация не порождает проблем при решении вопроса индивидуального отката транзакций, поскольку в этом случае для отката транзакции могут быть использованы журналы транзакций, расположенные как на диске, так и в буферах оперативной памяти.

Другое дело – мягкий сбой системы, когда содержимое буферов оперативной памяти утрачивается. В этом случае для обеспечения возможности восстановления данных необходимо выполнение определенных правил выталкивания буферов БД и буферов журнала транзакций.

Первое правило согласованной политики выталкивания буфера журнала и буферов страниц БД заключается в следующем: запись об изменении объекта БД должна попадать в журнал раньше, чем произведены сами изменения объекта. Соответствующий протокол журнализации (и управления буферизацией) называется Write Ahead Log (WAL) – «пиши сначала в журнал». Суть этого протокола состоит в том, что если требуется произвести изменения объекта БД, то перед этим нужно гарантировать запись об этих изменениях в журнал транзакций. Другими словами, если в памяти содержится объект, к которому применена некоторая команда модификации, то в памяти журнала транзакций содержится запись об этой операции. Однако при наличии в журнале записи о некотором изменении объекта в памяти БД может и не быть самого измененного объекта.

Второе правило требует, чтобы для каждой успешно завершившейся транзакции во внешней памяти должны быть реально зафиксированы записи журнала для этой транзакции. Какой бы сбой ни произошел, система должна обеспечивать возможность восстановления состояния БД, содержащего результаты всех зафиксированных к моменту сбоя транзакций.

Для этого при фиксации транзакции во внешнюю память журнала должны быть вытолкнуты все записи буфера журнала, относящиеся к изменениям БД, совершенным этой транзакцией.

Третье правило выталкивания буферов является следствием ограниченности объемов буферов БД и журнала транзакций. Периодически или при наступлении определенного события (например, количество страниц в dirty-списке превысило определенный порог или количество свободных страниц в буфере уменьшилось и достигло критического значения) система принимает так называемую контрольную точку. Принятие контрольной точки означает выталкивание во внешнюю память содержимого буферов БД вместе со специальной записью контрольной точки, которая представляет собой список всех осуществляемых в данный момент транзакций.

Таким образом, оказывается, что требованием, гарантирующим возможность восстановления последнего согласованного состояния БД, является выталкивание при фиксации транзакций во внешнюю память журнала всех записей об изменении БД этой транзакцией. При этом последней записью в журнал, производимой от имени данной транзакции, является специальная запись о конце этой транзакции.

5.5.3 Параллельное выполнение транзакций

Механизм управления транзакциями может использоваться не только для восстановления согласованности данных в случае возникновения сбоев системы, но и для повышения эффективности работы системы при совместном доступе к данным большого числа пользователей.

Как уже говорилось, одним из обязательных свойств транзакций является их изолированность – транзакции не должны оказывать никакого влияния друг на друга. Это свойство обеспечивается, если транзакции выполняются последовательно, т.е. следующая транзакция начинается только после окончания предыдущей. Простейшим механизмом, реализующим такой подход, является блокировка БД на время выполнения очередной транзакции. Однако при таком подходе эффективность доступа к данным резко падает, если в системе одновременно работает большое число пользователей.

Вообще говоря, одновременно выполняемые операции по доступу к БД не обязательно являются несовместимыми во времени, даже если они связаны не с чтением данных, а с их

модификацией. Поэтому естественным является желание использовать механизм, позволяющий распараллелить работу транзакций при условии обеспечения требования их изолированности, т.е. исключения конфликтных ситуаций при их выполнении.

Эта задача известна как задача обеспечения параллелизма (concurrency) или сериализации транзакций (от латинского serialize – располагать в последовательном порядке). Ее решение заключается в построении механизма одновременного, параллельного выполнения транзакций, который эквивалентен их последовательному выполнению.

Набор из нескольких транзакций, элементарные операции которых чередуются друг с другом, называется смесью транзакций, а последовательность, в которой выполняются элементарные операции заданного набора транзакций, называется графиком запуска набора транзакций.

Задача состоит в построении такого графика запуска набора транзакций, который был бы эквивалентен последовательному их выполнению.

Такой график называется правильным или сериальным.

Для построения правильного графика запуска набора транзакций необходимо рассмотреть проблемы нарушения целостности данных, которые могут возникать при одновременном выполнении двух транзакций.

Имеются три основные проблемы параллелизма:

- проблема потери результатов обновления;
- проблема незафиксированной зависимости (чтение «грязных данных», неаккуратное считывание);
- проблема несовместимого анализа.

5.5.4 Методы сериализации транзакций

Существует два способа организации сериализации транзакций:

- подход, основанный на синхронизационных захватах или блокировках объектов БД;
- подход, основанный на использовании временных меток.

Суть обоих подходов состоит в обнаружении конфликтов транзакций и их устранении. Рассмотрим эти подходы более подробно.

Заметим, что для каждого из этих подходов имеется две разновидности методов – пессимистическая и оптимистическая.

Пессимистические методы ориентированы на ситуацию, когда конфликты возникают часто. В этом случае конфликты распознаются и разрешаются немедленно при их возникновении.

Оптимистические методы, напротив, используются в случаях, когда конфликты транзакций достаточно редкие. Эти методы основываются на том, что результаты всех операций модификации БД сохраняются в рабочей памяти транзакций. Реальная модификация БД производится только на стадии фиксации транзакций. И только в этот момент проверяется, не возникают ли конфликты с другими транзакциями.

В качестве примера рассмотрим более распространенную пессимистическую разновидность методов сериализации транзакций. Пессимистические методы сравнительно просто трансформируются в соответствующие оптимистические варианты.

6. Механизмы обеспечения конфиденциальности в СУБД

6.1 Классификация угроз конфиденциальности в СУБД

Понятие **конфиденциальности** означает необходимость сохранения в тайне данных, которые являются критичными для организации в целом.

Потеря таких данных может, например, ослабить позиции организации среди конкурентов.

Угрозы конфиденциальности данных возможны со стороны:

- оборудования;
- сетевых коммуникаций;
- СУБД и прикладного программного обеспечения;
- базы данных;
- пользователей;
- программистов и операторов;
- администраторов баз данных.

К основным методам нарушения конфиденциальности относят:

- использование прав доступа другого пользователя;
- подключение к кабельным сетям;
- ввод хакерами некорректных данных;
- шантаж людей, имеющих доступ к данным;
- создание «лазеек» в систему;
- похищение данных, программ и оборудования;
- отказ систем защиты, вызвавший превышение допустимого уровня доступа;
- недостаточную обученность персонала;
- просмотр и раскрытие засекреченных данных.

Противодействие нарушению конфиденциальности осуществляется с помощью:

- шифрования данных;
- управления доступом к данным различных пользователей.

Основным методом противодействия утере конфиденциальности данных является их шифрование, под которым понимается кодирование данных с использованием специального алгоритма, в результате чего данные становятся недоступными для чтения любой программой, не имеющей ключа дешифрования.

Некоторые СУБД включают в себя средства шифрования данных.

Доступ к зашифрованным данным осуществляется только со стороны санкционированных пользователей, при этом данные предварительно декодируются.

Для организации защищенной передачи данных по незащищенным сетям применяются специальные системы шифрования, включающие следующие компоненты:

- ключ шифрования, предназначенный для шифрования исходных данных (обычного текста);

- алгоритм шифрования, задающий процедуру преобразования обычного текста в шифротекст с помощью ключа шифрования;

- ключ дешифрования, предназначенный для дешифрования шифро-текста;

- алгоритм дешифрования, задающий процедуру преобразования шифротекста в исходный текст с помощью ключа дешифрования.

Другим широко распространенным методом противодействия утере конфиденциальности является управление доступом к данным со стороны различных пользователей. Этот метод реализуется подсистемой управления доступом к данным, являющейся составной частью СУБД.

6.1.1 Особенности применения криптографических методов

Системы шифрования разделяют на симметричные и несимметричные.

В симметричных системах один и тот же ключ применяют как для шифрования, так и для дешифрования данных, при этом предполагается наличие защищенных линий связи, по которым осуществляется обмен ключами.

Большинство пользователей не имеют доступа к защищенным линиям связи, поэтому считается, что для получения надежной защиты ключ должен иметь длину не меньше длины самого сообщения. Однако большинство существующих систем шифрования построено на использовании ключей, которые короче самих сообщений. Обычно применяют ключи длиной 56-128 бит.

В настоящее время ключи длиной до 64 бит раскрываются с достаточной степенью вероятности правительственными службами развитых стран, для чего используется специальное и дорогое оборудование. Однако прогнозируется, что ключи длиной 128 бит в обозримом будущем можно использовать в качестве достаточно надежного средства шифрования.

Несимметричные системы шифрования предусматривают использование для шифрования и дешифрования различных ключей.

Как правило, симметричные системы являются более быстродействующими по сравнению с несимметричными. Однако на практике обе схемы применяются совместно, когда несимметричная система используется для шифрования ключа, сгенерированного случайным образом, а уже этот случайный ключ используется в симметричной системе для шифрования самого сообщения.

6.2 Средства идентификации и аутентификации

Механизм паролей является самым распространенным методом подтверждения личности (идентификации) пользователей, при этом с точки зрения обеспечения необходимого уровня защиты все используемые пароли должны держаться пользователями в секрете и обновляться через некоторые промежутки времени. Как правило, каждая организация предъявляет собственные требования к использованию паролей, однако наиболее общими являются следующие:

- пароль не должен отображаться на экране в ходе регистрации пользователя в системе;
- списки идентификаторов пользователей и их паролей должны храниться в системе в зашифрованном виде;
- пароли должны быть не менее установленной длины, обязательно содержать цифры и служебные символы;
- пароли должны подлежать замене через установленный интервал времени;
- в качестве паролей не следует применять реальные имена или адреса пользователей.

Для выявления слабых и устаревших паролей в системе следует использовать специальное программное обеспечение.

Процесс авторизации пользователей включает аутентификацию субъектов, требующих получения доступа к объектам.

Аутентификация представляет собой механизм определения факта, является ли пользователь тем, за кого он себя выдает.

6.3 Организация взаимодействия СУБД и базовой операционной системы

Предоставление пользователям доступа к компьютерной системе обычно осуществляет системный администратор, который создает учетные записи пользователей. Каждому пользователю присваивается уникальный идентификатор, который используется операционной системой при попытке входа в систему. С каждым идентификатором связывается определенный пароль, выбираемый пользователем и известный операционной системе.

При регистрации пользователь должен предоставлять системе свой пароль для выполнения проверки (аутентификации) того, является ли он тем, за кого себя выдает.

Указанная процедура позволяет организовать контролируемый доступ к компьютерной системе, но не обязательно предоставляет право доступа к СУБД или прикладной программе.

Для получения пользователем права доступа к СУБД может использоваться аналогичная процедура, выполняемая администратором БД, который создает индивидуальные идентификаторы пользователей в среде СУБД. Идентификатор пользователя СУБД также связывается с паролем, который должен быть известен только данному пользователю. Этот пароль используется СУБД для идентификации данного пользователя.

6.4 Средства управления доступом

6.4.1 Основные понятия в управлении доступом к данным

Подсистема управления доступом к данным, хранящимся в БД, является составной частью СУБД и базируется на ряде специфических понятий.

Представление (вид) является некоторым виртуальным отношением, которого реально в БД не существует, но которое динамически воспроизводится на основании одного или нескольких базовых отношений, реально хранящихся в БД.

С точки зрения пользователя представление может применяться как и базовое отношение, однако оно не хранится в БД, а его содержимое определяется как результат запроса к одному или нескольким базовым отношениям.

Любые операции над представлением автоматически преобразуются в операции над базовыми отношениями. Изменения в базовых отношениях сразу же отражаются на содержимом представления, и, наоборот, допустимые изменения в представлении заносятся в базовые отношения.

Представления позволяют организовать доступ пользователей к данным с помощью простых запросов, а главное – обеспечивают мощный и гибкий механизм защиты данных за счет скрытия некоторых частей БД от определенных пользователей.

Как правило, представления создаются с помощью операторов языка SQL.

Другой способ организации защиты данных состоит в использовании средств ограничения доступа, предоставляемых языком SQL, или, иначе, использовании средств дискреционной защиты.

Дискреционное управление доступом (discretionary access control) – разграничение доступа между поименованными субъектами и поименованными объектами. Субъект с определенным правом доступа может передать это право любому другому субъекту.

Дискреционная защита является многоуровневой логической защитой.

Логическая защита в СУБД представляет собой набор привилегий или ролей по отношению к защищаемому объекту. К логической защите можно отнести и владение таблицей (представлением). Владелец таблицы может изменять (расширять, отнимать, ограничивать доступ) набор привилегий (логическую защиту). Данные о логической защите

находятся в системных таблицах БД и отделены от защищаемых объектов (от таблиц или представлений).

Для управления правами доступа к данным каждому пользователю администратор БД присваивает идентификатор пользователя, защищаемый личным паролем. Любой SQL-оператор выполняется в среде СУБД от имени определенного пользователя.

Идентификатор пользователя применяется для определения набора объектов БД, на которые этот пользователь может ссылаться, и определения набора операций, которые пользователь может выполнять над указанными объектами.

Любой объект, создаваемый средствами языка SQL, имеет своего владельца, указываемого идентификатором пользователя. Только владелец объекта знает о существовании данного объекта и имеет право выполнять над ним любые операции.

Привилегиями или правами доступа называются действия, которые пользователю разрешено выполнять для конкретной таблицы или представления. При создании таблицы с помощью оператора CREATE TABLE пользователь автоматически назначается владельцем (субъектом права) созданного объекта (объекта права) и получает полный набор прав доступа к нему. Остальные пользователи не имеют к созданному объекту никаких прав доступа. Для предоставления другим пользователям возможности доступа к новой таблице ее владелец должен явно предоставить им необходимые привилегии с помощью оператора GRANT языка SQL.

6.4.2 Использование представлений для обеспечения конфиденциальности информации в СУБД

Создание специализированных представлений предназначено для ограничения прямого доступа некоторых пользователей к таблицам БД. Использование представлений обеспечивает ряд преимуществ:

- независимость от данных;
- актуальность, т.е. немедленное отображение в представлении изменений в таблицах;
- повышение защищенности данных;
- снижение сложности запросов;
- обеспечение возможности работы только с теми данными, которые действительно необходимы конечному пользователю;
- возможность создания образов БД, с которыми будут работать разные пользователи;
- обеспечение целостности данных.

Основным недостатком применения представлений является снижение производительности системы.

С точки зрения пользователя БД представление выглядит как реальная таблица данных, содержащая набор поименованных столбцов и строк данных. В действительности доступные в представлении строки и столбцы данных являются результатом выполнения запроса, заданного при определении представления.

6.4.3 Мандатное управление доступом

Средства мандатной защиты предоставляются специальными (trusted) версиями СУБД.

Мандатное управление доступом (mandatory access control) – это разграничение доступа субъектов к объектам данных, основанное на характеризуемой меткой конфиденциальности информации, которая содержится в объектах, и на официальном

разрешении (допуске) обращаться субъектам к информации такого уровня конфиденциальности.

Для чего же нужна мандатная защита? Средства произвольного управления доступом характерны для уровня безопасности С (секретно).

Как правило, их, в принципе, вполне достаточно для подавляющего большинства коммерческих приложений. Тем не менее, они не решают одной весьма важной задачи – задачи слежения за передачей информации. Средства произвольного управления доступом не могут помешать авторизованному пользователю законным образом получить секретную информацию и затем сделать ее доступной для других, неавторизованных пользователей.

Нетрудно понять, почему это так. При произвольном управлении доступом привилегии существуют отдельно от данных (в случае реляционных СУБД – отдельно от строк реляционных таблиц), в результате чего данные оказываются «обезличенными» и ничто не мешает передать их кому угодно даже средствами самой СУБД; для этого нужно лишь получить доступ к таблице или представлению.

Физическая защита СУБД главным образом характеризует данные (их принадлежность, важность, представительность и пр.). Это в основном метки безопасности, описывающие группу принадлежности и уровни конфиденциальности и ценности данных объекта (таблицы, столбца, строки или поля). Метки безопасности (физическая защита) неизменны на всем протяжении существования объекта защиты (они уничтожаются только вместе с ним) и территориально (на диске) располагаются вместе с защищаемыми данными, а не в системном каталоге, как это происходит при логической защите.

СУБД не дает проигнорировать метки конфиденциальности при получении доступа к информации. Такие реализации СУБД, как правило, представляют собой комплекс средств как на машине-сервере, так и на машине-клиенте, при этом возможно использование специальной защищенной версии операционной системы. Кроме разграничения доступа к информации посредством меток конфиденциальности, защищенные СУБД предоставляют средства слежения за доступом субъектов к объектам защиты (аудит).

Использование СУБД с возможностями мандатной защиты позволяет разграничить доступ собственно к данным, хранящимся в информационной системе, и доступ к именованным объектам данных. Единицей защиты в этом случае будет являться, например, запись о договоре N, а не таблица или представление, содержащее информацию об этом договоре. Пользователь, который будет пытаться получить доступ к договору, уже никак не сможет обойти метку конфиденциальности. Существуют реализации, позволяющие разграничивать доступ вплоть до конкретного значения некоторого атрибута в данной строке конкретной таблицы. Дело не ограничивается одним значением метки конфиденциальности – обычно сама метка представляет собой набор значений, отражающих, например, уровень защищенности устройства, на котором хранится таблица, уровень защищенности самой таблицы, уровень защищенности атрибута и уровень защищенности конкретного кортежа.

За исключением атрибута собственности (логическая защита), разбивающего данные (таблицы) на собственные (принадлежащие данному субъекту) и чужие, физическая защита разбивает данные более тонко. Но можно ли обойтись без физической защиты или, по крайней мере, попытаться это сделать, реализовав, например, сложный набор хранимых процедур. В общем-то, некоторое подобие такой защиты реализуемо в случае, когда метки добавляются в таблицу в качестве дополнительного атрибута, доступ к таблицам запрещается вообще и ни одно приложение не может выполнить интерактивный SQL-запрос, а только хранимую процедуру и т.п. Ряд реализаций подобного уровня защиты использует вызов набора хранимых процедур с весьма абстрактными (что очень желательно) именами. Система реализации защиты информации в данном случае достаточно сложна и предполагает определенный уровень доверия к администратору безопасности, так как он имеет право изменять структуру БД, а значит, хранимые процедуры и представления. Физически же

администратор безопасности в данном случае не изолирован от управления секретными данными.

Кроме того, защищенные СУБД позволяют разграничить доступ к информационной системе с тех или иных рабочих станций для тех или иных зарегистрированных пользователей, определить режимы работы, наложить ограничения по времени работы тех или иных пользователей с тех или иных рабочих станций. В случае реализации данных опций на прикладном уровне задача, как правило, сводится к созданию сервера приложений, который занимается отслеживанием, «кто и откуда пришел».

Отдельный комплекс серверных приложений (обычно – хранимых процедур, если в СУБД отсутствует мандатная защита) обеспечивает аудит.

6.5 Аудит и подотчетность

Процедура аудита предназначается для проверки внешней аудиторской фирмой задействованных средств управления и соответствия уровня защищенности данных установленным требованиям.

В ходе выполнения проверки аудиторы знакомятся с используемыми ручными процедурами, проверяют все компьютерные системы и состояние всей документации на систему. Обычно аудиторская проверка предусматривает контроль следующих процедур и механизмов управления:

- обеспечение точности вводимых данных;
- поддержание точности процедур обработки данных;
- предотвращение появления и своевременное обнаружение ошибок в процессе выполнения программ;
- корректное тестирование, документирование и сопровождение разработанных программных средств;
- предупреждение несанкционированного изменения программ;
- предоставление прав доступа и контроль за их использованием;
- поддержание документации в актуальном состоянии.

Все процедуры и средства контроля должны быть достаточно эффективными, в противном случае они должны быть подвергнуты пересмотру.

Для определения активности использования БД и анализа нештатных ситуаций в системе используются ее файлы журнала.

Проведение аудиторских проверок и регулярный контроль содержимого файлов журнала с целью выявления ненормальной активности в системе часто позволяет обнаружить и пресечь любые попытки нарушения защиты.

7. Механизмы, поддерживающие высокую готовность

7.1 Средства, поддерживающие высокую готовность

Средствами поддержки высокой готовности БД являются три функции любой современной СУБД: поддержка транзакций, управление параллельностью выполнения транзакций и восстановление БД. Эти функции позволяют сохранить достоверность и согласованность БД при наличии отказов оборудования или программных компонентов, а также при эксплуатации БД в многопользовательской среде.

Все перечисленные функции находятся во взаимной зависимости. И механизм управления параллельностью доступа, и средства восстановления предназначены для защиты БД от потери данных или их перехода в несогласованное состояние.

Рассмотрим только механизмы восстановления БД.

7.2 Средства восстановления баз данных

Средствами восстановления БД являются резервные копии БД и системного журнала, при этом основной единицей восстановления в системах с БД является транзакция.

Менеджер восстановления СУБД обеспечивает поддержку двух из четырех основных свойств транзакций – атомарности и продолжительности – даже при наличии сбоев в системе. Он гарантирует, что при восстановлении после сбоя для каждой отдельной транзакции в БД будут зафиксированы либо все внесенные ею изменения, либо не будет зафиксировано ни одного.

Запись в БД не выполняется за один шаг, поэтому существует вероятность, что после завершения транзакции внесенные ею изменения не будут реально отражены в БД, потому что еще не достигли файлов БД.

При выполнении операции чтения СУБД выполняет следующие действия:

- определяет дисковый адрес блока данных;
- считывает блок данных с диска и помещает его в буфер СУБД в оперативной памяти;
- копирует необходимые сведения из буфера СУБД в переменные.

При выполнении операции записи СУБД выполняет следующие действия:

- определяет дисковый адрес блока данных;
- считывает блок данных с диска и помещает его в буфер СУБД в оперативной памяти;
- копирует необходимые сведения из переменных в буфер СУБД;
- выводит блок данных из буфера СУБД в оперативной памяти на диск.

Буфера СУБД занимают определенную часть оперативной памяти и используются для обмена данными с дисковой памятью системы. Только после выгрузки соответствующего буфера на диск можно считать, что выполненные операции обновления приобрели постоянный характер. Выгрузка буферов в БД может инициироваться по специальной команде (например, по команде завершения транзакции) или же автоматически, как только буфер будет заполнен.

Если отказ системы произойдет между записью данных в буфер и выгрузкой буфера во вторичную память, менеджер восстановления должен уточнить состояние транзакции, выполнявшей запись в момент аварии.

Если транзакция уже выдала команду завершения, то для обеспечения продолжительности ее результатов менеджер восстановления должен повторно ее выполнить (прогнать), чтобы восстановить все изменения, внесенные ею.

С другой стороны, если на момент отказа системы транзакция еще не была завершена, менеджер восстановления должен отменить любые ее результаты (откатить транзакцию), что будет гарантировать соблюдение ее атомарности.

На рис. 7.1 приведен пример нескольких параллельно выполняющихся транзакций $T_1 \ldots T_6$.

Работа начинается в момент времени t0, а в момент времени tf происходит отказ системы. К моменту времени tc результаты транзакций T2 и T3 уже были выгружены на диск.

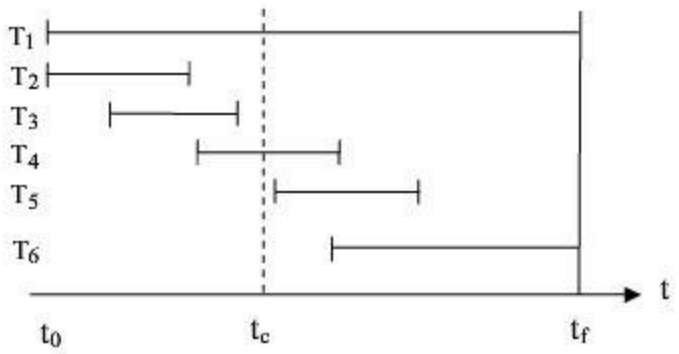

Рис.7.1.Иллюстрация операций отката и прогона.

Транзакции T_1 и T_6 еще не были завершены на момент отказа системы. Следовательно, при перезапуске СУБД менеджер восстановления должен выполнить их откат. Однако неясно, какие изменения, выполненные остальными (завершившимися) транзакциями, были перенесены в БД на диск.

Причина этой неопределенности состоит в том, что одна часть временных буферов СУБД могла быть уже выведена на диск, а другая – нет.

При отсутствии какой-либо дополнительной информации менеджеру восстановления потребуется принудительно повторно прогнать транзакции T_2, T_3, T_4 и T_6.

Современная СУБД должна предоставлять следующие средства восстановления:

- механизм резервного копирования, предназначенный для периодического создания копий БД;
- средства ведения журнала, в котором фиксируются текущее состояние транзакций и вносимые в БД изменения;
- механизм создания контрольных точек, обеспечивающий перенос на диск выполняемых в БД изменений с целью сделать их постоянными;
- менеджер восстановления, обеспечивающий восстановление согласованного состояния БД, нарушенного в результате отказа.

Механизм резервного копирования должен позволять создавать резервные копии БД и файла журнала через установленные интервалы без необходимости останова системы. Резервная копия БД используется в случае повреждения или разрушения файлов БД на диске. Резервное копирование может выполняться для БД в целом или в инкрементном режиме. В последнем случае в копию помещаются сведения только об изменениях, накопившихся с момента создания предыдущей полной или инкрементной копии системы. Как правило, резервные копии создаются на автономных носителях, например, на магнитных лентах.

7.2.1 Файл журнала

Для фиксации хода выполнения транзакций в БД СУБД использует специальный файл, называемый журналом. Он содержит сведения обо всех обновлениях, выполненных в БД.В файл журнала обычно помещается следующая информация о транзакциях:

- идентификатор транзакции, например T_1, T_2, T_3;
- время записи в журнал;
- тип записи журнала (начало транзакции START, операции вставки INSERT, обновления UPDATE или удаления DELETE, отмена или фиксация транзакции);
- идентификатор элемента данных, для которого реализуется операция обработки (операции вставки, удаления и обновления);
- копия элемента данных до операции, т.е. его значение до изменения (только операции обновления и удаления);

- копия элемента данных после операции, т.е. его значение после изменения (только для операций обновления и вставки);
- служебная информация файла журнала, включающая указатели на предыдущую и следующую записи журнала для этой транзакции (любые операции).

Кроме того, в файл журнала заносятся записи контрольных точек.

Очень часто файл журнала используется и для других целей, отличных от задач восстановления (например, для сбора сведений о текущей производительности, накопления контрольной информации и т.д.). В этом случае в файл журнала может помещаться множество дополнительной информации (например, сведения об операциях чтения, о регистрации пользователей, о завершении сеансов пользователей и т.д.).

Следует заметить, что файл журнала потенциально является узким местом с точки зрения производительности любых систем, поэтому скорость записи информации в файл журнала может оказаться одним из важнейших факторов, определяющих общую производительность системы с БД.

7.2.2 Создание контрольных точек

Помещаемая в файл журнала информация предназначена для использования в процессе восстановления системы после отказа. Одно из основных затруднений в этой схеме состоит в том, что, когда происходит отказ, может отсутствовать какая-либо информация о том, насколько далеко назад следует «откатиться» в файле журнала, чтобы начать повторный прогон уже завершенных транзакций. В результате может оказаться, что повторный прогон будет выполнен и для тех транзакций, которые уже были окончательно зафиксированы в БД. Для ограничения объема поиска и последовательной обработки информации в файле журнала используется технология создания контрольных точек, представляющих собой моменты синхронизации между БД и журналом регистрации транзакций.

Контрольные точки формируются через установленный интервал времени и инициируют:

- запись всех имеющихся в оперативной памяти записей журнала на диск;
- запись всех модифицированных блоков в буферах БД на диск;
- помещение в файл журнала записи контрольной точки, содержащей идентификаторы всех транзакций, которые были активны в момент создания контрольной точки.

Если транзакции выполняются последовательно, то после возникновения отказа файл журнала просматривается с целью обнаружения последней из транзакций, начавших свою работу до момента создания контрольной точки. Любая более ранняя транзакция будет зафиксирована в БД, т.е. ее изменения были перенесены на диск в момент создания последней контрольной точки. Следовательно, прогону подлежит только транзакция, которая была активна в момент создания контрольной точки, а также все транзакции, которые начали свою работу позже и для которых в журнале присутствуют записи как начала, так и завершения. Та транзакция, которая была активна в момент отказа, должна быть отменена. В случае, если транзакции выполняются в системе параллельно, потребуются повторный прогон всех транзакций, которые завершили свою работу с момента создания контрольной точки, и выполнение отката всех транзакций, которые были активны в момент отказа.

Если предположить, что на рис. 7.1 в момент времени t_c была создана контрольная точка, то это позволит установить, что результаты выполнения транзакций T_2 и T_3 уже внесены на диск. Поэтому менеджеру восстановления не потребуется выполнять повторный прогон этих транзакций.

Однако прогон потребуется выполнить для транзакций T_4 и T_5, которые завершили свою работу уже после создания контрольной точки. Кроме того, потребуется выполнить откат транзакций T_1 и T_6, которые все еще были активны в момент возникновения отказа.

7.2.3 Методы восстановления

Рассмотрим два метода восстановления, которые применимы в случае, когда БД не была полностью разрушена, а лишь утратила согласованное состояние.

Метод восстановления с использованием отложенного обновления.

В этом методе обновления не заносятся в БД до тех пор, пока транзакция не выдаст команду на фиксацию выполненных изменений. Если выполнение транзакции будет прекращено до достижения этой точки, никаких изменений в БД выполнено не будет, поэтому не потребуется и их отмена.

Однако в данном случае может потребоваться повторный прогон уже завершившихся транзакций, поскольку их результаты могли еще не достичь диска. При применении данного метода файл журнала используется для восстановления следующим образом:

- при запуске транзакции в журнал помещается запись Начало транзакции;
- при выполнении любой операции записи помещаемая в файл журнала строка содержит все необходимые данные, за исключением значения элементов данных до обновления. Запись изменений в буфера СУБД или саму БД не производится;
- при достижении транзакцией своей конечной точки в журнал помещается запись Транзакция завершена. Все записи журнала по данной транзакции выводятся на диск, после чего выполняется фиксация внесенных транзакцией изменений. Для внесения действительных изменений в БД используется информация, помещенная в файл журнала;
- в случае отмены выполнения транзакции записи журнала по данной транзакции аннулируются и на диск не выводятся.

Метод восстановления с использованием немедленного обновления.

При использовании этого метода все изменения вносятся в БД сразу же после их выполнения в транзакции, не дожидаясь ее завершения. Кроме необходимости повторного прогона изменений, выполненных транзакциями, закончившимися до появления сбоя, в данном случае может потребоваться выполнить откат изменений, внесенных транзакциями, которые не были завершены к этому моменту. При применении данного метода файл журнала используется для восстановления следующим образом:

- при запуске транзакции в журнал помещается запись Начало транзакции;
- при выполнении любой операции записи помещаемая в файл журнала строка содержит все необходимые данные;
- как только запись о начале транзакции будет помещена в файл журнала, все выполненные обновления вносятся в буфера БД;
- в собственно файлы БД изменения будут внесены при очередной разгрузке буферов БД на диск;
- когда транзакция завершает свое выполнение, в файл журнала заносится запись Транзакция завершена.

Очень важно, чтобы в файл журнала все записи или хотя бы определенная часть записей помещались до внесения изменений в БД. Такой вариант работы называется протоколом предварительной записи журнала.

Если изменения сначала будут внесены в БД и сбой в системе возникнет до помещения информации об этом в файл журнала, то менеджер восстановления не будет иметь возможности отменить или повторить данную операцию. При использовании протокола предварительной записи журнала менеджер восстановления всегда сможет безопасно предположить, что если для определенной транзакции в файле журнала отсутствует запись

Транзакция завершена, значит эта транзакция была активна в момент возникновения отказа и, следовательно, должна быть отменена.

Если выполнение транзакции было прекращено, то для отмены выполненных ею изменений может быть использован файл журнала, так как в нем сохранены сведения об исходных значениях всех измененных элементов данных. Поскольку транзакция может выполнить несколько изменений одного и того же элемента, отмена обновлений выполняется в обратном порядке. Независимо от того, были ли результаты выполнения транзакции внесены в саму БД, наличие в записях журнала исходных значений полей гарантирует, что БД будет приведена в состояние, отвечающее началу отмененной транзакции.

7.3 Оперативное администрирование

Администрирование БД может выполняться как на этапе ее разработки, так и во время эксплуатации. Оперативное администрирование выполняется при эксплуатации системы с целью обеспечения необходимых показателей качества и поддержания системы в состоянии постоянной готовности и включает в себя следующие мероприятия:

- определение требований защиты и поддержки целостности данных;
- обучение пользователей работе с системой;
- контроль текущей производительности системы и соответствующая настройка БД;
- регулярное резервное копирование;
- разработка требуемых механизмов и процедур восстановления;
- обеспечение полноты используемой документации, включая материалы, разработанные внутри организации;
- поддержка актуальности используемого программного и аппаратного обеспечения, включая заказ и установку пакетов обновлений в случае необходимости.

Представленный перечень задач, решаемых при оперативном администрировании БД, позволяет сделать вывод об их управленческом характере. Творческой технической задачей является администрирование данных, которое в основном выполняется на этапе разработки БД.

7.4 Тиражирование данных

Под тиражированием данных понимается резервное копирование БД и ее файла журнала на носитель, сохраняемый отдельно от системы.

Любая современная СУБД должна предоставлять средства резервного копирования, позволяющие восстанавливать БД в случае ее разрушения. Кроме того, рекомендуется создавать резервные копии БД и файла журнала с некоторой установленной периодичностью, а также организовывать хранение созданных копий в местах, обеспеченных необходимой защитой.

В случае отказа, в результате которого БД становится непригодной для дальнейшей эксплуатации, резервная копия и зафиксированная в файле журнала оперативная информация используются для восстановления БД до последнего согласованного состояния.

СУБД должна предоставлять средства ведения системного журнала, в котором будут фиксироваться сведения обо всех изменениях состояния БД и ходе выполнения текущих транзакций, что необходимо для эффективного восстановления БД в случае отказа. Преимущества использования системного журнала заключаются в том, что в случае нарушения работы или отказа СУБД базу данных можно будет восстановить до последнего известного согласованного состояния, воспользовавшись последней созданной резервной копией БД и оперативной информацией, содержащейся в файле журнала. Если в отказавшей системе функция ведения системного журнала не использовалась, БД можно будет

восстановить только до того состояния, которое было зафиксировано в последней созданной резервной копии. Все изменения, которые были внесены в БД после создания последней резервной копии, окажутся потерянными.

Современные СУБД, как правило, предоставляют средства создания контрольных точек, позволяющие зафиксировать в БД серию последних выполненных изменений.

Под контрольной точкой понимается момент синхронизации между состоянием БД и состоянием журнала выполнения транзакций. В этот момент все буфера принудительно выгружаются на диск.

Механизм создания контрольных точек может использоваться совместно с ведением системного журнала, что позволит повысить эффективность процесса восстановления. В момент создания контрольной точки СУБД выполняет действия, обеспечивающие запись на диск всех данных, хранившихся в оперативной памяти компьютера, а также помещение в файл журнала специальной записи контрольной точки.

Контрольные вопросы

1. Назовите особенности концепции безопасности баз данных.
2. Дайте определения понятиям: «защита данных», «механизм защиты данных», «средства защиты», «технические средства защиты», «аппаратные средства защиты», «программные средства защиты».
3. Дайте определения понятиям: «криптографические средства защиты», «организационные средства защиты», «конфиденциальная информация», «доступ к информации», «субъект доступа», «объект доступа».
4. Назовите основные угрозы безопасности баз данных.
5. Перечислите основные типы опасностей, намеренные и непреднамеренные, нарушения функций системы защиты баз данных.
6. Перечислите основные требования к системе обеспечения безопасности баз данных.
7. Назовите и охарактеризуйте фундаментальные функции защита от несанкционированного доступа.
8. Опишите способы разграничения доступа.
9. Дайте определение понятию «целостность баз данных» и поясните его роль в защите баз данных.
10. Поясните понятие «процедура аудита» и назовите контролируемые механизмы управления.
11. Назовите задачи и средства администратора безопасности баз данных.
12. Охарактеризуйте понятие «многоуровневая защита».
13. Назовите критерии защищенности баз данных.
14. Поясните роль, предназначение многоуровневой модели безопасности баз данных.
15. Перечислите механизмы обеспечения целостности в СУБД.
16. Назовите основные причины возникновения угроз целостности баз данных.
17. Перечислите ограничения целостности в реляционной модели.
18. Назовите и охарактеризуйте группы ограничения целостности.
19. Назовите и поясните способы реализации механизмов обеспечения целостности данных в СУБД.
20. Поясните понятие «триггеров» и назовите, для достижения каких целей они применяются.
21. Перечислите способы задания триггеров.
22. Дайте определения понятию «транзакция».
23. Перечислите и охарактеризуйте свойства транзакций.
24. Поясните процедуру откат транзакций и восстановления данных после сбоев.
25. Поясните роль, предназначение журнализации изменений базы данных.
26. Поясните процедуру параллельного выполнения транзакций.
27. Назовите методы сериализации транзакций.
28. Перечислите механизмы обеспечения конфиденциальности в СУБД.
29. Дайте определение понятию «конфиденциальности».
30. Поясните суть классификации угроз конфиденциальности в СУБД.
31. Поясните особенности применения криптографических методов обеспечения конфиденциальности в СУБД.
32. Назовите средства идентификации и аутентификации.
33. Перечислите средства управления доступом.
34. Поясните основные понятия в управлении доступом к данным.
35. Поясните использование представлений для обеспечения конфиденциальности информации в СУБД.

36. Охарактеризуйте мандатное управление доступом.
37. Назовите механизмы, поддерживающие высокую готовность.
38. Перечислите средства, поддерживающие высокую готовность.
39. Назовите средства восстановления баз данных.
40. Дайте определение понятию «файл журнала».
41. Поясните процедуру создания контрольных точек.
42. Назовите методы восстановления данных в базе данных.
43. Дайте определение понятию «оперативное администрирование».
44. Поясните процедуру тиражирования данных.

Заключение

Из выше изложенного материала можно сделать вывод, что защита баз данных состоит в обеспечении целостности, конфиденциальности и высокой доступности информации в базах данных. При этом необходимо учесть, что организация защиты носит индивидуальный характер для конкретной базы данных. Перечислим основные механизмы защиты баз данных.

В обеспечении целостности баз данных эффективно используется принцип минимальных привилегий, для его реализации необходимо:

- сформировать минимальную конфигурацию ПО и запущенных сервисов СУБД;
- предоставить доступ к ОС и базе данных только уполномоченным пользователям;
- ограничить доступ к административным аккаунтам ОС (root и пр.);
- ограничить доступ к административным аккаунтам СУБД (таким как SYSDBA, SYSOPER, SA);
- ограничить доступ пользователей только к тем объектам БД, которые им нужны для работы.

При этом механизмы обеспечения целостности будут следующие:

1. Физический уровень:
- физическая защита;
- дублирование и/или защита носителей (например, ASM для Oracle);
- резервное копирование данных (средства ОС и СУБД).
2. Средства СУБД:
- декларативные средства контроля целостности;
- транзакции и блокирование;
- триггеры (процедурные средства контроля целостности);
- средства имитозащиты (при передаче по сети).

Механизмы обеспечения конфиденциальности баз данных:

1. Разграничение доступа:
- идентификация/аутентификация пользователей;
- базовые средства SQL разграничения доступа;
- тщательный контроль доступа (RLS – row-level security).
2. Криптографические методы защиты.
3. Аудит (в особенности, суперпользователей).

Механизмы обеспечения высокой доступности:

- физическая защита, дублирование носителей;
- настройка производительности и оптимизация;
- управление транзакциями и блокировками;
- архивирование данных (управление задачами резервного копирования и восстановления).

Литература

1. Тарасов В.Г. Основы теории автоматизированных систем управления. — М.: ВВИА, 1988.

2. Дейт К.Д. Введение в системы баз данных, 7-е издание. — М.:Изд. Дом «Вильямс», 2001.

3. Марка Д.А., МакГоуэн К.Л. Методология структурного анализа и проектирования. — М.: МетаТехнология, 1993.

4. CASE Аналитик для IBM PC. Руководство аналитика. — М.:Эйтекс, 1993.

5. Маклаков С.В. BP-win, ER-win. CASE-средства разработки информационных систем. — М.: Диалог-МИФИ, 1999.

6. Григорьев Ю.А., Ревунков Г.И. Банки данных. — М.: Изд–во МГТУ им. Н.Э. Баумана, 2002.

7. Ветошкин В.М., Гузенко В.Г., Лялюк И.Н. Разработка дидактических сценариев автоматизированных учебных курсов.— М.: ВВИА им. Н.Е. Жуковского, 2004.

8. Ветошкин В.М., Гузенко В.Г., Чубаров Н.М., Швед А.Г. Технология концептуальной оптимизации систем баз данных и проект интеллектуальной САПР «СИНТЕЗ». В сб. материалов научно-технической конференции ВВС «Авиационные системы искусственного интеллекта и компьютерные технологии разработки информационно-программных систем». Часть 2. — М.: ВВИА им. Н.Е. Жуковского, 1995.

9. Кузнецов С.Д. Объектно-ориентированные базы данных — основные концепции, организация и управление: краткий обзор.
http://home.od.ua/~relayer/algo/dbms/oodb.htm.