**Технологии разработки и защиты баз данных**

Преподаватель: Юрий Юрьевич Прошкин

Оглавление

[БД и информационные системы 4](#_Toc2598145)

[Основные понятия 4](#_Toc2598146)

[Архитектура информационных сетей 5](#_Toc2598147)

[Система управления базой данных 7](#_Toc2598148)

[Не-реляционные модели данных 11](#_Toc2598149)

[Иерархическая модель данных 11](#_Toc2598150)

[Сетевая модель 11](#_Toc2598151)

[Многомерная модель 12](#_Toc2598152)

[Объектно-ориентированная модель 13](#_Toc2598153)

[Реляционная модель данных 14](#_Toc2598154)

[Основные понятия 14](#_Toc2598155)

[Виды связей в реляционных СУБД 15](#_Toc2598156)

[Проектирование БД. Проблемы проектирования БД 18](#_Toc2598157)

[Нормализация отношений 20](#_Toc2598158)

[Проектирование РБД (реляционная БД) с использованием ER-технологии 22](#_Toc2598159)

[Характеристика связей 23](#_Toc2598160)

[Получение отношений из диаграмм ER-типа 25](#_Toc2598161)

[Предварительные отношения из ER-диаграмм 25](#_Toc2598162)

[Предварительные отношения для бинарных связей 1:1 25](#_Toc2598163)

[Предварительные отношения для бинарных связей 1:М 29](#_Toc2598164)

[Предварительные отношения для бинарных связей степени N:M 31](#_Toc2598165)

[Связи более высокого порядка 31](#_Toc2598166)

[Использование ролей 31](#_Toc2598167)

[Организация связей сущностей 34](#_Toc2598168)

[Обеспечение целостности 35](#_Toc2598169)

[Реляционная алгебра. Реляционные операции. 36](#_Toc2598170)

[Операции над отношениями 36](#_Toc2598171)

[Структурированный язык запросов SQL 40](#_Toc2598172)

[Общая характеристика языка. 40](#_Toc2598173)

[Основные операторы языка 41](#_Toc2598174)

БД и информационные системы

Основные понятия

В основе решения многих задач лежит обработка информации. Для облегчения обработки информации создаются информационные системы (ИС). Автоматизированными называют ИС, в которых применяют технические средства, в частности, ЭВМ. **ИС – это совокупность аппаратно-программных средств, задействованных для решения некоторой прикладной задачи.**

**Банк данных (БнД) является разновидностью ИС, в которой реализованы функции централизованного хранения и накопления обрабатываемой информации, организованной в одну или несколько БД.**

БнД в общем случае состоит из следующих компонентов: базы (нескольких баз) данных, системы управления БД, словаря данных, администратора, вычислительной системы и обслуживающего персонала.

БД представляет собой совокупность специальным образом организованных данных, хранимых в памяти вычислительной системы и отображающих состояние объекта и их взаимосвязей в рассматриваемой предметной области.

Логическую структуру, хранимой в БД, называют моделью представления данных. **К основным моделям представления данных (моделям данных) относятся следующее: иерархическая, сетевая, реляционная, постреляционная, многомерная и объектно-ориентированная.**

**СУБД – комплекс языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями.**

**Приложение представляет собой программу или комплекс программ, обеспечивающих автоматизацию обработки информации для прикладной задачи**. Приложения могут создаваться в среде или вне среды СУБД. С помощью системы программирования, использующей средства доступа к БД.

**Словарь данных (СД) представляет собой подсистему БнД, предназначенную для централизованного хранения информации о структурах данных, взаимосвязях файлов БД друг с другом, типах данных и форматах их представления, принадлежности данных пользователю, кодах защиты и разграничения доступа и т.п.**

**Администратор БД (АБД) есть лицо или группа лиц, отвечающих за выработку требований к БД, её проектирования, создания, эффективное использование и сопровождения.**

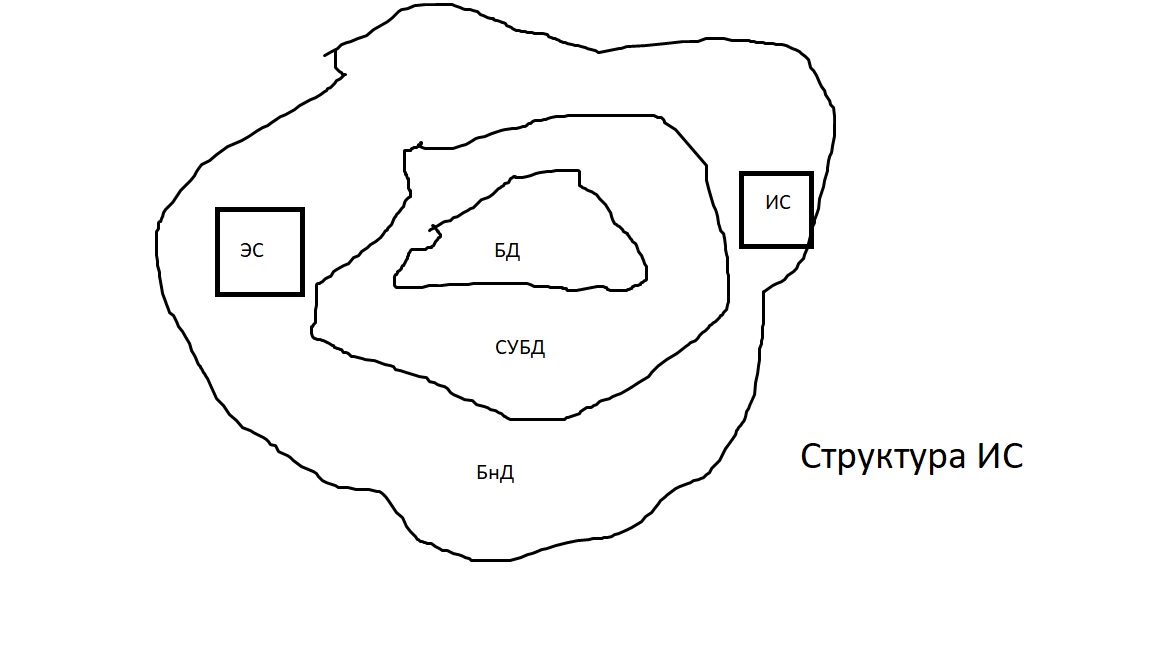
В процессе эксплуатации АБД обычно следит за функционированием ИС, обеспечивает защиту от несанкционированного доступа, контролирует избыточность, непротиворечивость, сохранность и достоверность хранимой БД информации.

Для пользовательских СУБД роль АБД возлагается на пользователя СУБД.

В вычислительной среде АБД, как правило, взаимодействует с администратором сети. В обязанности последнего входят контроль за функционированием аппаратных-программных средств сети, реконфигурации сети, восстановления ПО после сбоев и отказа оборудования, профилактические мероприятия и обеспечение разграничения доступа.

**Вычислительная система (ВС) представляет собой совокупность взаимосвязанных и согласовано действующих ЭВМ или процессоров и других устройств, обеспечивающих автоматизацию процессов приёма, обработки и выдачи информации потребителю.**

Обслуживающий персонал выполняет функции поддержания технических и программных средств в работоспособном состоянии. Он проводит профилактические, регламентные, восстановительные и другие работы по планам, а также по мере необходимости.



Архитектура информационных сетей

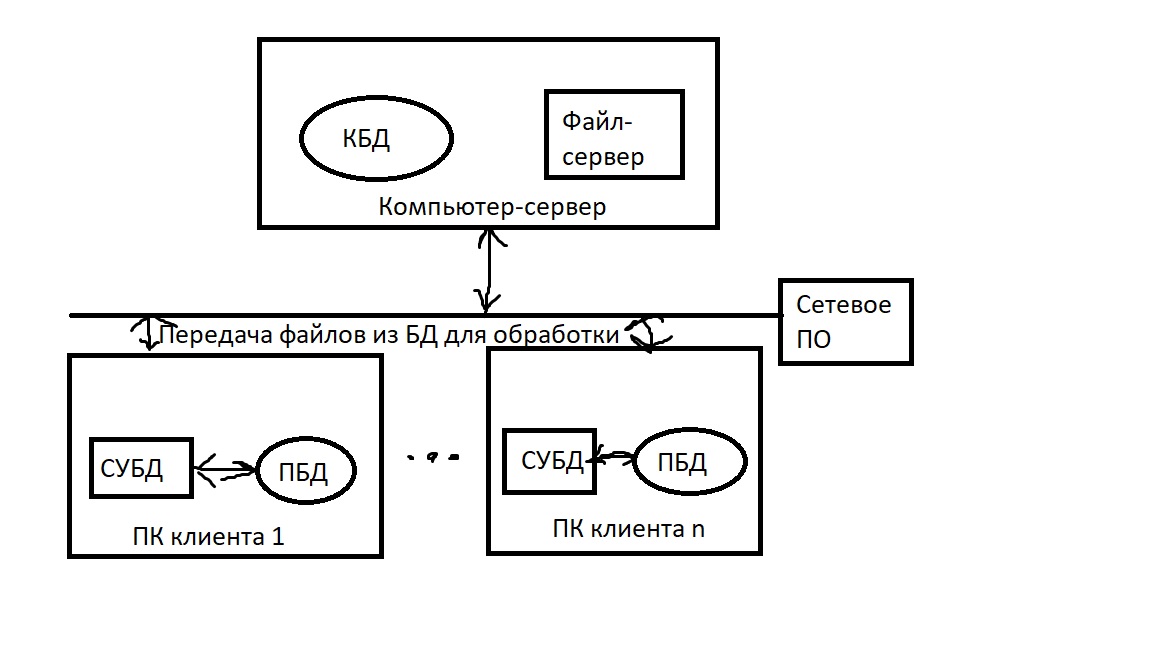
Эффективность функционирования ИС во многом зависит от её архитектуры. В настоящее время перспективной является архитектура «Клиент-сервер».

Сервер определённого ресурса в компьютерной сети называется компьютер (программа), управляющий этим ресурсом, клиентом – компьютер (программа), использующий этот ресурс. В качество ресурса компьютерной сети могут выступать, к примеру, БД, файловые системы, службы печати, почтовые службы. Тип сервера определяется видом ресурса, которым он управляет. Например, если управляемым ресурсом является БД, то соответствующий сервер называется сервером БД.

Достоинством организации по архитектуре «Клиент-сервер» является удачное сочетание централизованного хранения, обслуживания и коллективного доступа к общей корпоративной информации с индивидуальной работой пользователя над персональной информацией. Архитектура «Клиент-сервер» допускает различные варианты реализации.

Исторически первыми появились распределенные ИС с применением «Файл-сервера». В таких ИС по запросу пользователя файлы БД передаются на персональные компьютеры, где и производится их обработка.

Недостатком такого варианта архитектуры является высокая интенсивность передачи обрабатываемых данных. При чем зачастую передаются избыточные данные: вне зависимости от того сколько записей из БД требуется пользователю, файлы базы передаются целиком.

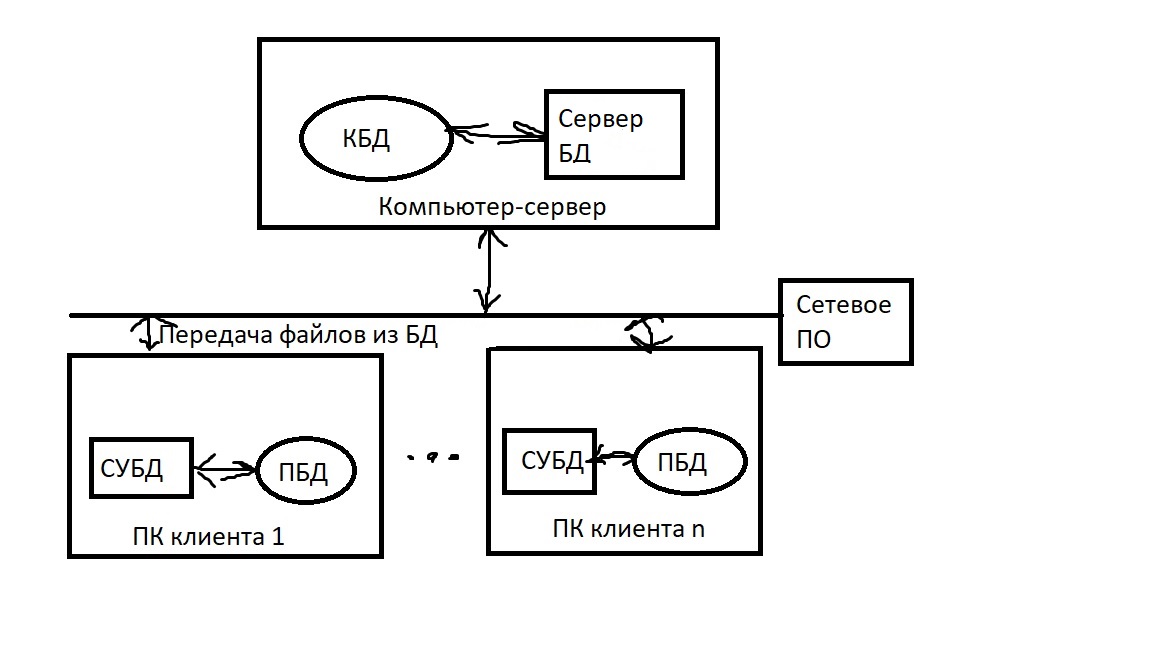


*Структура ИС с «Файл-сервер»(КБД также связан с файл-сервером)*

*КБД – корпоративная БД, ПБД – персональная БД.*

При архитектуре «Клиент-сервер» с использованием сервера БД, сервер БД обеспечивает выполнение основного объёма обработки данных.

Формируемые пользователем или приложением запросы поступают к серверу БД в виде инструкций языка SQL. Сервер БД выполняет поиск и извлечение нужных данных, которые затем передаются на компьютер пользователя. Достоинством такого подхода в сравнении с предыдущем является заметно меньший объем передаваемых данных.



*Структура ИС с сервером БД*

Важнейшим достоинством применением БД в ИС является обеспечение независимости данных от прикладных программ. Это дает возможность пользователям не заниматься проблемами представления данных на физическом уровне: размещение данных в памяти, методов доступа к ним и т.д.

Такая независимость достигается поддерживаемым СУБД многоуровневым представлением данных в БД на логическом (пользовательском) и физическом уровнях. Благодаря СУБД и наличию логического уровня представления данных обеспечивается отделение концептуальной (понятийной) модели БД от её физического представления в памяти ЭВМ.

Система управления базой данных

Классификация СУБД. В общем случае, под СУБД можно понимать любой программный продукт, поддерживающий процессы создания, ведения и использования БД. Рассмотрим, какие из имеющихся на рынке программ имеют отношение к БД и в какой мере они связаны с БД.

К СУБД относятся следующие основные виды программ:

1. полно-функциональные СУБД;
2. серверы БД;
3. клиенты БД;
4. средства разработки программ работы с БД.

Поло-функциональные СУБД (ПФ СУБД) представляют собой традиционные СУБД, которые сначала появились для больших машин, затем для мини-машин и для ПЭВМ. Из числа всех СУБД современной ПФ СУБД являются наиболее многочисленными и мощными по своей возможности.

Обычно ПФ СУБД имеют развитый интерфейс, позволяющий с помощью команд меню выполнять основные действия с БД: создавать и модифицировать структуры таблиц, вводить данные, формировать запросы, разрабатывать отчёты, выводить их на печать и т.п. Для создания запросов и отчётов необязательно программирование, а удобно пользоваться языком QBI (Query By Example – формулировки запросов по образцу).

Многие ПФ СУБД включают средства программирования для профессиональных разработчиков. Некоторые системы имеют в качестве вспомогательных и дополнительные средства проектирования схем БД или CASE-подсистемы. Для обеспечения доступа к другим БД или к данным SQL-серверов ПФ СУБД имеют факультативные модули.

Серверы БД предназначены для организации центра обработки данных в сетях ЭВМ. Эта группа БД в настоящее время менее многочисленна, но их количество постоянно растёт. Серверы БД реализуют функции управления БД, запрашиваемые другими (клиентскими) программами, обычно, с помощью операторов SQL.

В роли клиентских программ для серверов БД в общем случае могут использоваться различные программы: ПФ СУБД, электронные таблицы, текстовые процессоры, программы электронной почты и т.д.

**При этом элементы пары «Клиент-сервер» могут принадлежать одному или разным производителям ПО.**

Средства разработки программ работы с БД могут использоваться для создания разновидностей следующих программ:

1. клиентских программ;
2. серверов БД и их отдельных компонентов;
3. пользовательских приложений.

Программы первого и второго видов довольно малочисленны, т.к. предназначены, главным образом, для системных программистов. Пакетов третьего вида гораздо больше, но меньше, чем ПФ СУБД.

К средствам разработки пользовательских приложений относятся системы программирования, например, Clipper, разнообразные библиотеки программ для различных языков программирования, а также пакеты автоматизации разработок (в том числе систем типа «Клиент-сервер»).

**По характеру использования СУБД делят на персональные и многопользовательские.**

Персональные СУБД обычно обеспечивают возможность создания персональных БД и недорогих приложений, работающих с ними. Персональные СУБД или разработанные с их помощью приложения зачастую могут выступать в роли клиентской части многопользовательской СУБД.

Многопользовательские СУБД включают в себя сервер БД и клиентскую часть и, как правило, могут работать в неоднородной вычислительной среде (разными типами ЭВМ и ОС). С точки зрения пользователя, **СУБД реализуют функции хранения, изменения (пополнение, редактирование и удаление) и обработки информации, а также разработки и получения различных выходных документов.** Для работы с хранящейся в БД информацией СУБД предоставляет программам и пользователям следующие два типа языков:

1. язык описания данных (ЯОД) – высокоуровневый непроцедурный язык декларативного типа, предназначенный для описания логической структуры данных;
2. язык манипулирования данными (ЯМД) – совокупность конструкций, обеспечивающих выполнение основных операций по работе с данными: ввод, модификацию и выборку данных по запросу.

**Перечисленные выше функции СУБД, в свою очередь, используют следующие основные функции более низкого уровня, которые назовём низкоуровневыми:**

1. управление данными во внешней памяти;
2. управление буферами оперативной памяти;
3. управление транзакциями;
4. ведение журнала изменений в БД;
5. обеспечение целостности и безопасности БД.

Реализация функции управления данными во внешней памяти в разных системах может различаться и на уровне управления ресурсами (используя файловые системы ОС или непосредственное управление устройствами ПЭВМ), и по логике самих алгоритмов управления данными. В основном, методы и алгоритмы управления данными являются «внутренним делом» СУБД и прямого отношения к пользователю не имеют. Качества реализации этой функции наиболее сильно влияет на эффективность работы специфических ИС, например, с огромными БД, со сложными запросами, большим объемом обработки данных.

Необходимость буферизации данных и, как следствие, реализация функции управления буферами оперативной памяти обусловлена тем, что объем ОЗУ меньше объема внешней памяти.

Буферы представляют собой области оперативной памяти, предназначенной для ускорения обмена между внешней и оперативной памятью. В буферах временно хранятся фрагменты БД, данные из которых предполагается использовать при обращении к СУБД или планируется записать в базу после обработки.

Механизм транзакции используется в СУБД для поддержания целостности данных в базе. **Транзакцией называется некоторая неделимая последовательность операций над данными БД, которая отслеживается СУБД от начала и до завершения.** Если по каким-либо причинам (сбой и отказы оборудования, ошибки в ПО, включая приложения) транзакция остаётся незавершённой, то она отменяется. Говорят, что транзакциям присущи три основных свойства:

1. атомарность (выполняются все входящие в транзакцию операции или ни одна);
2. сериализуемость (отсутствует взаимное влияние выполняемых в одно и то же время транзакций);
3. долговечность (даже крах системы не приводит к утрате результатов зафиксированной транзакции).

Контроль транзакции важен в однопользовательских и многопользовательских СУБД, где транзакции могут быть запущены параллельно. В последнем случае говорят о сериализуемости транзакции. Под сериализации параллельно выполняемых транзакцией понимается составление такого плана их выполнения (сериального плана), при котором суммарный эффект реализации транзакций эквивалентен эффекту их последовательного выполнения.

При параллельном выполнении смеси транзакций возможно возникновение конфликтов (блокировок), разрешение которых является функцией СУБД. При обнаружении таких случаев обычно производится «откат» путём отмены изменений, произведенных одной или несколькими транзакциями. Ведение журнала изменения в БД (журнализация изменений) выполняется СУБД для обеспечения надёжности хранения данных в базе при наличии аппаратных сбоев и отказов, а также ошибок в ПО.

**Журнал СУБД – это особое БД или часть основной БД, непосредственно недоступная пользователю и используемая для записи информации обо всех изменениях БД.** В различных СУБД в журнал могут заноситься записи, соответствующие изменениям СУБД на разных уровнях: от минимальной внутренней операции модификации страницы внешней памяти до логической операции модификации БД (например, вставки, записи, удаление столбца, изменение значения в поле) и даже транзакции.

Для эффективной реализации функции ведения журнала изменений в базе данных необходимо обеспечить повышенную надёжность хранения и поддержания в рабочем состоянии самого журнала.

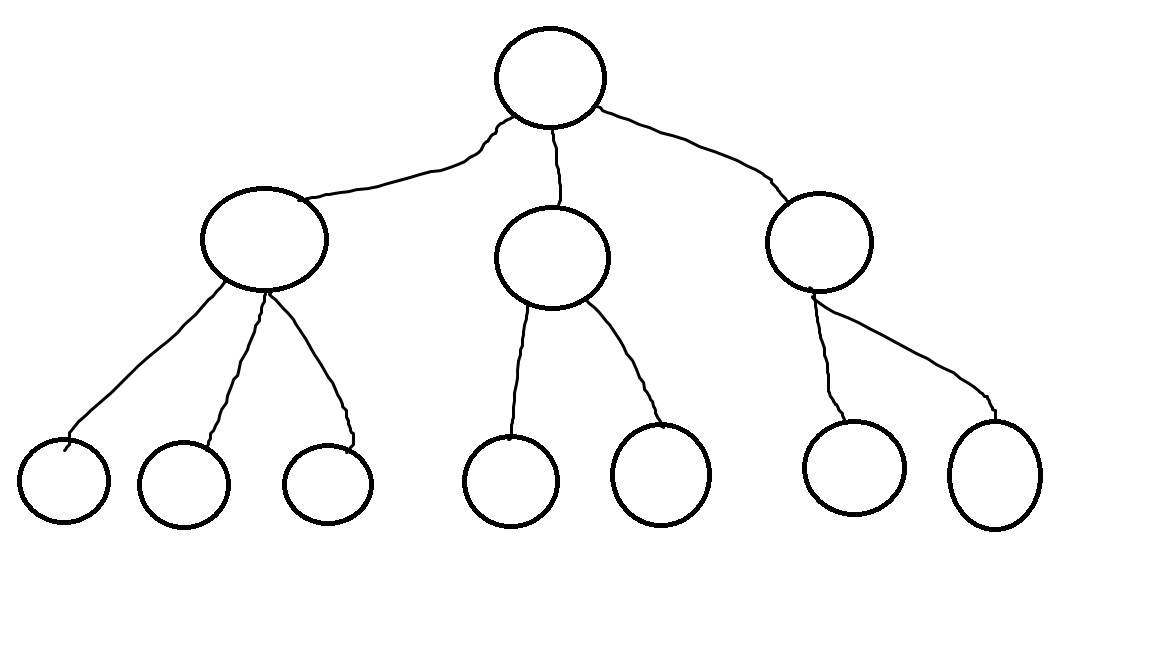
Обеспечение целостности БД составляет необходимое условие успешного функционирования БД, особенно для случая использования БД в сетях. **Целостность БД есть свойство БД, означающая что в ней содержится полное, непротиворечивая и адекватно отражающая предметную область информацию.** Поддержание целостности БД включает проверку целостности и её восстановление в случае обнаружений противоречий в БД. Целостное состояние БД описывается с помощью ограничений целостности в виде условий, которым должны удовлетворять хранимые в базе данные.

Обеспечение безопасности достигается в СУБД шифрованием прикладных программ, данных, защиты паролем, поддержкой уровней доступа к БД и к отдельным её элементам (таблицам, формам, отчётам и т.д.).

Не-реляционные модели данных

Иерархическая модель данных

В иерархической модели связи между данными можно описать с помощью упорядоченного графа (или дерева).



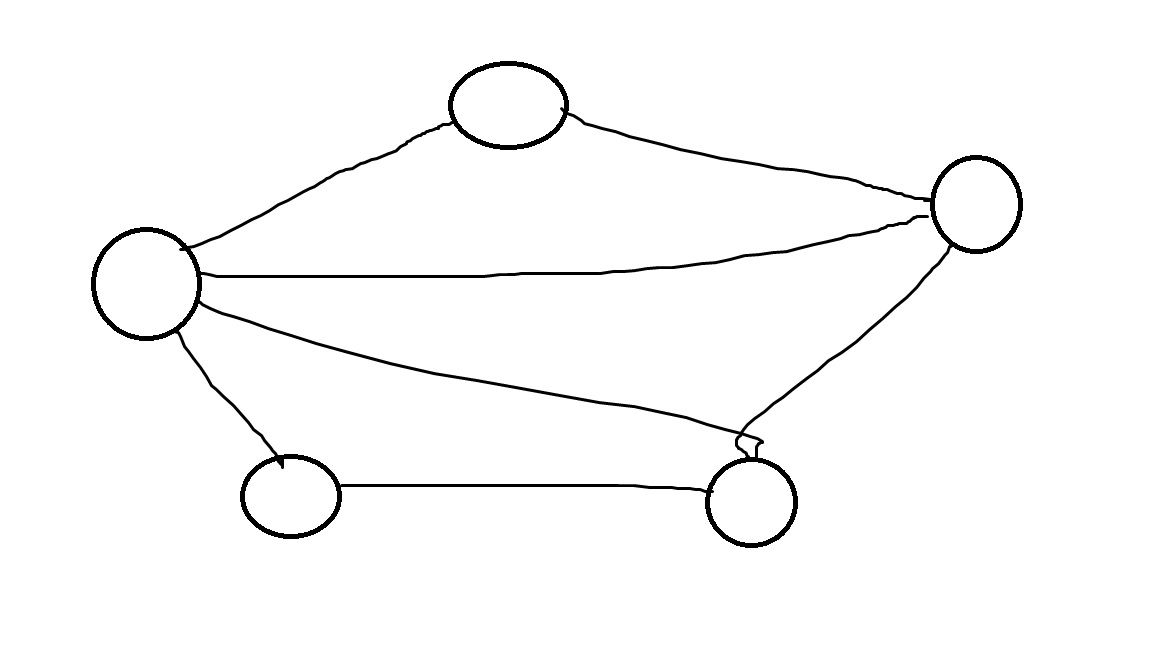
*Иерархическая модель представления данных*

К достоинствам иерархической модели данных относятся эффективное использование памяти ЭВМ и неплохие показатели времени выполнения основных операций над данными. Иерархическая модель удобна для работы с иерархически упорядоченной информацией.

Недостатком является её громоздкость для обработки информации с достаточно сложными логическими связями, а также сложность понимания для обычного пользователя.

Сетевая модель

Сетевая модель данных позволяет отображать разнообразные взаимосвязи элементов данных в виде произвольного графа, обобщая тем самым иерархическую модель данных.



*Представление связей данных в сетевой модели*

Достоинством сетевой модели данных является возможность эффективной реализации по показателям затрат памяти и оперативности. В сравнении с иерархической моделью сетевая модель предоставляет большие возможности в смысле допустимости образования произвольных связей. Недостатком сетевой модели данных является высокая сложность и жёсткость схемы БД, построенной на её основе, а также сложность для понимания и выполнения обработки информации в БД обычным пользователем. Кроме того, в сетевой модели данных ослаблен контроль целостности связей вследствие допустимости установления произвольных связей между записями.

Многомерная модель

Многомерный подход представления данных в базе появился практически одновременно с реляционным, но реально работающих многомерных СУБД (МСУБД) до настоящего времени было очень мало. В развитии концепции ИС модно выделить следующие 2 направления:

1. системы оперативной (транзакционной) обработки;
2. системы аналитической обработки (системы поддержки принятия решений).

Многомерные СУБД являются узкоспециализированными СУБД предназначенными для интерактивной аналитической обработки информации.

Основные понятия, используемые в многомерных СУБД: агрегируемость, историчность, прогнозируемость данных.

**Агрегируемость данных означает рассмотрение информации на различных уровнях её обобщения.** В ИС степень детальности представления информации для пользователя зависит от его уровня: аналитик, пользователь-оператор, управляющий, руководитель.

**Историчность данных предполагает обеспечение высокого уровня статичности (неизменности) собственно данных и их связей, а также обязательность привязки данных ко времени.** Статичность данных позволяет использовать при их обработке специализированные методы загрузки, хранения, индексации и выборки.

Временная привязка данных необходима для частого выполнения запросов, имеющих значения времени и даты в составе выборке. Необходимость упорядочивания данных по времени в процессе обработки и предоставление данных пользователю накладывает требования на механизмы хранения и доступа к информации. Так, для уменьшения времени обработки запросов, желательно, чтобы данные всегда были отсортированы в том порядке, в котором они наиболее часто запрашиваются.

**Прогнозируемость данных подразумевает задание функцией прогнозирования и применения их к различным временным интервалам.**

Основным достоинством многомерной модели данных является удобство и эффективность аналитической обработки больших объемов данных, связанных со временем.

Недостатком многомерной модели данных является её громоздкость для простейших задач достаточно обычной оперативной обработки информации.

Объектно-ориентированная модель

В объектно-ориентированной модели при представлении данных имеется возможность идентифицировать отдельные записи базы. Между записями БД и функциями их обработки устанавливаются взаимосвязи с помощью механизмов, подобных соответствующим средствам в объектно-ориентированных языках программирования.

Для выполнения действий над данными в рассматриваемой модели БД применяются логические операции, усиленные объектно-ориентированными механизмами инкапсуляции, наследования и полиморфизма.

**Инкапсуляция ограничивает область видимости имени свойства пределами того объекта, в котором оно определено.**

**Наследование, наоборот, распространяет область видимости свойства на всех потомках объекта.**

**Полиморфизм в объектно-ориентированных языках программирования означает способность одного и того же программного кода работать с разными разнотипными данными.** Иными словами, он означает допустимость в объектах разных типов иметь методы (процедуры или функции) с одинаковыми именами. Во время выполнения объектной программы одни и те же методы оперируют с разными объектами в зависимости от типа аргумента.

Основным достоинством объектно-ориентированной модели данных в сравнении с реляционной является возможность отображения информации о сложных взаимосвязях объектов. Объектно-ориентированная модель данных позволяет идентифицировать отдельную запись БД и определять функции их обработки.

Недостатком объектно-ориентированной модели являются высокая понятийная сложность, неудобство обработки данных и низкая скорость выполнения запросов.

Реляционная модель данных

Основные понятия

Реляционная модель данных предложена сотрудником фирмы IBM Эдгаром Коддом и основывается на понятии «отношения» (relation). Наглядной формой представления отношения является привычная для человеческого восприятия двумерная таблица. В реляционных СУБД отслеживается взаимное соответствие записей различных таблиц. Мощь реляционных СУБД заключается в их способности выбирать соответствующие данные из этих таблиц и создавать ответы на вопросы, которые нельзя получить только из одной такой таблицы.

Физическое размещение данных в реляционных базах на внешних носителях легко осуществляется с помощью обычных файлов.

Реляционная модель данных (РМД) некоторой предметной области представляет собой набор отношений, изменяющихся во времени.

**Элементы реляционной модели.**

|  |  |
| --- | --- |
| Элемент РМ | Форма представления |
| Отношение | Таблица |
| Схема отношения | Строка заголовков столбцов таблицы (заголовок таблицы) |
| Картеж | Строка таблицы |
| Сущность | Описание свойств объекта |
| Атрибут | Заголовок столбца таблица |
| Домен | Множество допустимых значений атрибута |
| Значение | Значение поля в записи |
| Первичный ключ | Один или несколько атрибутов |
| Тип данных | Тип значения элементов таблицы |

**Отношение является важнейшим понятием и представляет собой двумерную таблицу, содержащую некоторые данные.**

**Сущность есть объект любой природы, данные о котором хранятся в БД. Данные о сущности хранятся в отношениях. Атрибуты представляют собой свойства, характеризующие сущность.** В структуре таблицы каждый атрибут именуется, и ему соответствует заголовок некоторого столбца таблицы. **Схема отношения (заголовок отношения) представляет собой список имён атрибутов.** Каждому картежу соответствует строка таблицы. Множество собственно картежей отношения часто называют мощностью (содержимым), телом отношения.

**Домен представляет собой множество всех возможных значений определенного атрибута отношений.**

**Первичным ключом (ключом отношения, ключевым атрибутом) называется атрибут отношения, однозначно идентифицирующий каждого из его картежей.** Ключи, обычно, используют для достижения следующих целей:

1. исключение дублирования значений в ключевых атрибутах (остальные атрибуты в расчёт не принимаются);
2. упорядочивание картежей;
3. ускорение работы с картежами отношения;
4. организации связывания таблиц.

Пример отношения:

|  |  |  |  |
| --- | --- | --- | --- |
| Фамилия | Адрес | Ном. Телефона | Дата регистр. |
| Иванов | Орёл | 123-34234-123 | 21.01.1999 |
| Егоров | Мценск | 453463-234234 | 20.12.2001 |
| Александров | Ливны | 235235-234234 | 12.01.2009 |
| Петров | Минск | 5252523-23523 | 02.05.2017 |
| Хлебонасущенский | Расейняй | 123123-25253 | 03.06.2018 |

Виды связей в реляционных СУБД

При проектировании реальных БД информацию обычно размешают в нескольких таблицах. Таблицы при этом связаны семантикой информации. В реляционных СУБД для указания связей таблиц производят операции их связывания. Многие СУБД при связывании таблиц автоматически выполняют контроль целостности вводимых данных в соответствии с установленными связями. Между таблицами могут устанавливать бинарные (между двумя таблицами), тернарные (между тремя таблицами), и, в общем случае, n-арные связи. Наиболее часто встречаются бинарные.

**При связывании двух таблиц выделяют основную и дополнительную (подчиненную) таблицы. Логическое связывание таблиц происходит при помощи ключа связи.** Ключ связи, по аналогии с обычным ключом таблицы, состоит из 1 или нескольких полей, которые, в данном случае, называют полярной связью. Существуют следующие основные виды связи:

1. 1 к 1 (1:1);
2. 1 ко многим (1:M);
3. многие к 1 (M:1);
4. многие ко многим (M:M,M:N).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Характеристика полей | 1:1 | 1:M | М:1 | M:N |
| Поля связи основной таблицы | Являются ключом | Являются ключом | Не являются ключом | Не являются ключом |
| Поля связи дополнительной таблицы | Являются ключом | Не является ключом | Является ключом | Не является ключом |

Дадим характеристику названным видам связи между двумя таблицами и приведем примеры их использования

**1:1. Связь вида 1:1 образуется в случае, когда все поля связи основной и дополнительной таблиц являются ключевыми.** Поскольку значения в ключевых полях обеих таблиц не повторяются, обеспечивается взаимно-однозначная соответствие записей из этих таблиц. Сами таблицы, по сути, здесь становятся равноправными. Пример 1.

О1

|  |  |
| --- | --- |
| НЗК \*+ | ФИО |
| 1 | Иванов |
| 2 | Петров |
| 3 | Сидоров |

Д1

|  |  |
| --- | --- |
| НЗК \*+ | Город |
| 1 | Орел |
| 2 | Мценск |
| 3 | Ливны |

**1:M. Связь 1:М имеют место в случае, когда одной записи основной таблицы соответствует несколько связей вспомогательной таблицы. Пример 2.**

Пусть имеются две таблицы: О2 и Д2. В таблице О2 содержится информация о видах компьютерных устройств, а в таблице Д2 – сведения о фирмах производителях этих устройств и количестве данного товара на складе.

О2

|  |  |  |
| --- | --- | --- |
| Код\*+ | Название устройства | Сведения |
| 1 | Bloody RV8 | Компьютерная мышь |
| 2 | Logitech C7 | Клавиатура |

Д2

|  |  |  |  |
| --- | --- | --- | --- |
| Код+ | Производитель | Название устройств+ | Количество на складе |
| 1 | 4Tech | Bloody RV8 | 3 |
| 2 | Logitech | Logitech C7 | 6 |

**Связь вида М:1 имеет место в случае, когда одной или нескольким записям основной таблицы ставится в соответствие одна запись дополнительной таблицы.** Пример.

В основной таблице (О3) содержится информация о товарах частного магазина, их поставщиках и количестве единиц товара на складе. В дополнительной (Д3) содержится подробные сведения о поставщиках.

О3

|  |  |  |  |
| --- | --- | --- | --- |
| Код поставщика+ | Название товара | Информация о товаре | Количество |
| 1 | Помидор | Овощ | 2 |
| 2 | Куриное мясо | Мясо | 5 |
| 1 | Огурец | Овощ | 135 |
| 3 | Кефир | Напиток | 12 |
| 2 | Говядина | Мясо | 12 |

Д3

|  |  |  |
| --- | --- | --- |
| Код поставщика\*+ | Название поставщика | Информация |
| 1 | ООО Овощебаза | Поставляет овощи |
| 2 | ООО Мясобаза | Поставляет мясные продукты |
| 3 | ООО Молобаза | Поставляет молочные продукты |

**Связь вида М:N. Самый общий вид связи M:N возникает в случаях, когда нескольким записям основной таблицы соответствует несколько записей дополнительной таблицы**. Пример.

Пусть в основной таблице О4 содержится информация о том, на каких автобусах могут работать водители некоторого автобусного парка. Таблица Д4 содержит сведения о том, кто из штата механиков эти автобусы обслуживает.

О4

|  |  |
| --- | --- |
| Водитель | Автобус+ |
| Жуков | 053 |
| Шмелёв | 053 |
| Николаев | 054 |
| Жуков | 054 |
| Завацкий | 054 |
| Медведев | 055 |

Д4

|  |  |
| --- | --- |
| Автобус+ | Механик |
| 053 | Есепов |
| 053 | Зайцев |
| 054 | Гордеев |
| 054 | Есепов |
| 055 | Гордеев |
| 055 | Зайцев |

Исходя из определений полей связей этих таблиц, можно составить новую таблицу О4-Д4, записями которой будут псевдозаписи. Таблица О4-Д4 будет выглядеть следующим образом.

|  |  |  |
| --- | --- | --- |
| Водитель | Автобус | Механик |
| Жуков | 053 | Есепов |
| Жуков | 053 | Зайцев |
| Шмелёв | 053 | Есепов |
| Шмелёв | 053 | Зайцев |
| Николаев | 054 | Гордеев |
| Николаев | 054 | Есепов |
| Жуков | 054 | Гордеев |
| Жуков | 054 | Есепов |
| Завадский | 054 | Гордеев |
| Завадский | 054 | Есепов |
| Медведев | 055 | Гордеев |
| Медведев | 055 | Зайцев |

Аналогично связи 1:1, связь M:N не устанавливает подчиненности таблиц.

**Из перечисленных видов связи наиболее широко используется связь вида 1:М. Связь вида 1:1 можно считать частным случаем связи 1:М. Связь М:1, по сути, является «зеркальным отображением» связи 1:М. Оставшийся вид связи, М:N, характеризуется как слабый вид связи или, даже, как отсутствие связи.**

Для обеспечения достоверности хранимой информации необходим контроль целостности связи, заключающийся в анализе содержимого двух таблиц на соблюдение следующих **правил:**

1. каждой записи основной таблицы соответствует нуль, или поле записи дополнительной;
2. в дополнительной таблице нет записей, которые не имеют родительских записей в основной таблице;
3. каждая запись дополнительной таблицы имеет только одну родительскую запись основной таблицы.

Проектирование БД. Проблемы проектирования БД

**Проектирование ИС, включающих в себя БД, осуществляется на физическом, концептуальном и логическом уровнях.**

Решения проблем на физическом уровне во многом зависит от используемой СУБД, зачастую автоматизировано и скрыто от пользователя. В ряде случаев пользователю предоставляется возможность настройки отдельных параметров системы, которые не составляют большой проблемы.

**На концептуальном уровне определяется общая структура информационного массива – она и называется моделью данных**. В соответствии с выбранной моделью строится ИС, в которой будут храниться данные, а также программы, ведущие их обработку (так называемое «манипулирование данными»).

**Логическое проектирование заключается в определении структуры и числа таблиц, формировании запросов к БД**, определении типа отчётных документов, разработки алгоритмов обработки информации, создании форм для ввода и редактирования данных в базе и решения ряда других задач.

При проектировании структур данных для автоматизированных систем можно выделить три основных подхода:

1. сбор информации об объектах решаемой задачи в рамках одной таблицы и последующая декомпозиция её на несколько взаимосвязанных таблиц на основе процедуры нормализации отношений;
2. формирование знаний о системе (определение типов исходных и их взаимосвязей) и требование к обработке данных, получение с помощью case-системы **(системы автоматизации проектирования и разработки БД),** готовые схемы БД или даже готовой прикладной ИС;
3. структурирование информации для использования в ИС в процессе проведения системного анализа на основе совокупности правил и рекомендаций.

**Избыточное дублирование данных и аномалии. Следует различать простое (неизбыточное) и избыточное дублирование данных. Наличие первого из них допускается в БД, а избыточное дублирование данных может приводить к проблемам при их обработке.**

Пример неизбыточного дублирования (обычного).

|  |  |
| --- | --- |
| Группа | Студент |
| 207 | Горлов |
| 307 | Комов |
| 207 | Иванов |
| 307 | Тюрин |

Пример избыточного дублирования.

|  |  |  |
| --- | --- | --- |
| Группа | Студент | Куратор |
| 207 | Горлов | Савостикова |
| 307 | Комов | Прошкин |
| 207 | Иванов | Савостикова |
| 307 | Тюрин | Прошкин |

Пример декомпозиции.

|  |  |
| --- | --- |
| Группа+ | Студент |
| 207 | Горлов |
| 307 | Комов |
| 207 | Иванов |
| 307 | Тюрин |

|  |  |
| --- | --- |
| Куратор | Группа+\* |
| Савостикова | 207 |
| Прошкин | 307 |
|  | |

Процедура декомпозиции отношения ГСК (Группа-Студент-Куратор) на два отношения ГС и ГК является основной процедурой нормализации отношений.

Избыточное дублирование данных создает проблемы при обработке картежей отношения, названные Э. Коддом «аномалиями обновления отношения». **Аномалиями будет называть такую ситуацию в таблицах БД, которая приводит к противоречиям в БД, либо существенно осложняет обработку данных.**

Выделяют **три основных вида аномалий:**

1. аномалии модификаций (или редактирования);
2. аномалии удаления;
3. аномалии добавления.

**Аномалии модификаций проявляются в том, что изменения значения одного данного может повлечь за собой просмотр всей таблицы и соответствующее изменение некоторых других записей таблицы. Аномалии удаления состоят в том, что при удалении какого-либо данного из таблицы может пропасть и другая информация, которая не связана напрямую с удаляемым данным. Аномалии добавления возникают в случаях, когда информацию в таблицу нельзя поместить до тех пор, пока она неполная, либо вставка новой записи требует дополнительного просмотра таблицы.**

Средством исключения избыточности в отношении и, как следствие, аномалий является нормализация отношений.

Нормализация отношений

**Процесс нормализации – это разбитие таблицы на две или более с целью ликвидации дублирования данных и потенциальной их противоречивости.** Окончательная цель нормализации сводится к получению такого проекта БД, в котором каждый факт появляется в лишь одном месте.

Каждая таблица в реляционной модели удовлетворяет условию, в соответствии на пересечении любой строки и столбца таблицы всегда находится единственная атомарное значение, и никогда не может быть множество таких значений. **Говорят, что таблица, удовлетворяющая такому условию, находится в первой нормальной форме (1НФ).**

**Теперь в дополнении к 1НФ можно определить дальнейшие уровни нормализации. По существу таблица находится во 2НФ, если она находится в 1НФ и удовлетворяет кроме того некоторому дополнительному условию. Таблица находится в 3НФ, если она находится во 2НФ и, помимо этого, удовлетворяет ещё другому дополнительному условию.**

Теория нормализации основывается на наличии той или иной зависимости между столбцами таблицы. Рассмотрим два вида таких зависимости: функциональная и многозначная.

**Функциональная зависимость, по сути, является связью типа М:1 между множеством атрибутов, рассматриваемого отношения:** если в отношении R, содержащего атрибуты A & B, атрибут B функционально зависит от атрибута A, то каждое отдельное значение атрибута A связано только с одним значением атрибута B (при чем в качестве A & B могут выступать группы атрибутов). **Таким образом, при наличии функциональной зависимости A→B строки, имеющие одинаковое значение атрибута A совпадают и по значению атрибута B. Однако обратное неверно: одно и то же значение атрибута может соответствовать разным значениям атрибута A.**

**Многозначная зависимость.** Говорят, что один атрибут таблицы многозначно определяет другой атрибут той же таблицы, если для каждого значения первого атрибута существует хорошо определенное множество соответствующих значений второго атрибута.

**Пример**. Рассмотрим таблицу «Приём экзаменов (зачётов)» таблица отражает связь дисциплины и формы отчетности с фамилией преподавателя. В этой таблице существует многозначная зависимость «Дисциплина - Преподаватель»: дисциплину «матанализ» ведут несколько преподавателей (Раков, Рыбин, Карпов) и, соответственно, все они могут участвовать в приёме экзаменов (зачётов).

Другая многозначная зависимость «Дисциплина – Форма отчётности»: по одной и той же дисциплине могут проводится и экзамен, и зачёт. При этом «Форма отчётности» и «Преподаватель» не связаны функциональной зависимостью, что приводит к избыточности (чтобы добавить фамилию ещё одного преподавателя, придётся внести в таблицу две новые строки).

|  |  |  |
| --- | --- | --- |
| Дисциплина | Преподаватель | Форма отчётности |
| Матанализ | Раков | Экзамен |
| Матанализ | Рыбин | Экзамен |
| Матанализ | Карпов | Экзамен |
| Матанализ | Раков | Зачёт |
| Матанализ | Рыбин | Зачёт |
| Матанализ | Карпов | Зачёт |

**Нормальные формы. Таблица находится в 1НФ тогда и только тогда, когда она не содержит одинаковых строк и в любом допустимом значении этой таблицы каждая её строка содержит только одно значение для каждого атрибута.**

**Таблица находится во 2НФ, если она удовлетворяет определению 1НФ и все её атрибуты, не входящие в первичный ключ, связаны полной функциональной зависимостью с первичным ключом.**

**Таблица находится в 3НФ, если она удовлетворяет определению 2НФ и ни один из её неключевых атрибутов не связан функциональной зависимостью с любым другим неключевым атрибутом.**

**Таблица находится в нормальной форме Бойса-Кодда (НФБК) тогда и только тогда, когда любая функциональная зависимость между её неключевыми атрибутами сводится к полной функциональной зависимости от возможного первичного ключа.**

В следующих нормальных формах (4НФ и 5НФ) учитываются не только функциональные, но и многозначные зависимости между атрибутами. Для того, чтобы привести определение этих нормальных форм, введем понятие полной декомпозиции таблицы.

**Полной декомпозицией таблицы называют такую совокупность произвольного числа её проекций, соединение которых полностью совпадает с содержимым таблицы.**

Далее дадим определение высших нормальных форм.

**Таблица находится в 5НФ тогда и только тогда, когда в каждой её полной декомпозиции все проекции содержат возможный ключ.**

**4НФ является частным случаем 5НФ, когда полная декомпозиция должна быть соединением ровно двух проекций.**

Проектирование РБД (реляционная БД) с использованием ER-технологии

**Постановка задачи проектирования РБД.**

Целью разработки БД является определение её логической структуры. В результате проектирования должен быть определён состав реляционных таблиц, для каждой таблицы – состав её атрибутов и логические связи между таблицами, первичный ключ, потенциальные ключи и внешние ключи. Для каждого атрибута должны быть заданы тип данного, его размер и ограничение целостности. При этом получаемая логическая модель оценивается по достижению следующих целей проектирования:

1. возможность хранения всех необходимых данных в БД;
2. исключение избыточных данных;
3. свeдEние числа хранимых отношений в БД к минимуму;
4. нормализация отношений для упрощения решения проблем, связанных с обновлением, добавлением и удалением данных.

Один из подходов проектирования БД: метод «сущность-связь». Разработку логической модели можно осуществлять различными методами. Наиболее формализованным и простым для понимания является метод «сущность-связь», или ER-метод (ER – Entity Relation).

**Суть метода состоит в построении ER-диаграмм, отображающих в графической форме основные объекты предметной области и связи между ними, и в определении характеристик этих связей. Затем по чётким правилам делается переход от ER-диаграмм к таблицам БД. Осуществляется наполнение таблиц атрибутами и проверка их на выполнение условий нормализаций (НФ БК). Определяются ключевые атрибуты таблиц и связи между таблицами. Результатом проектирования является схема данных БД.**

Сущности и связи

**Сущность** – это объект, информация о котором должна быть представлена в БД.

**Экземпляр сущности** – это информация о конкретном представителе объектов.

**Связь** – соединение между двумя или более сущностями.

**Экземпляр** связи – это конкретная связь между представителями объектов.

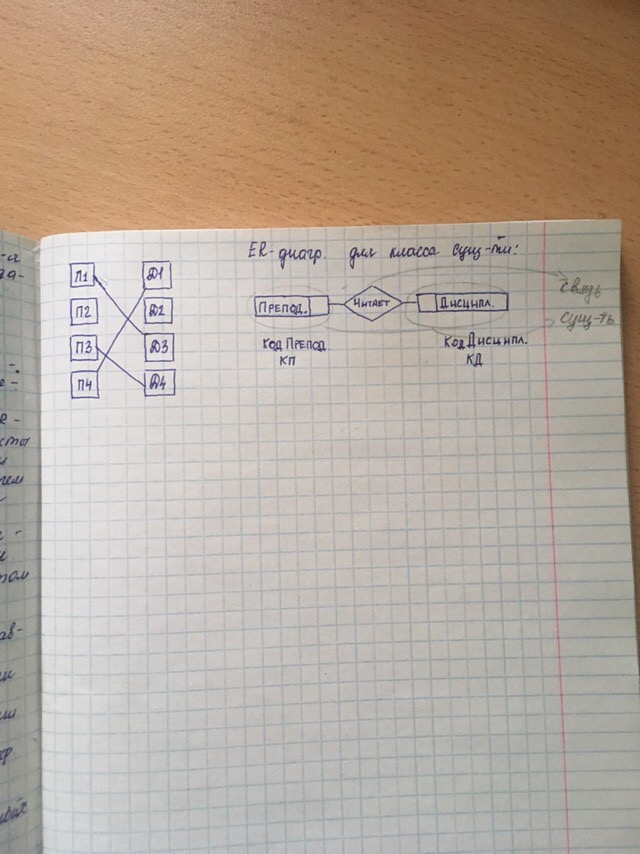
**Атрибут** – свойство сущности или связи.

**Атрибут** или набор атрибутов, использующийся для однозначной идентификации экземпляра сущности, называется ключом сущности.

Построение ER-диаграмм. Различают ER-диаграммы для экземпляра сущности и ER-диаграммы для классов сущности. Ниже приведены ER-диаграммы обоих типов для БД.

«Преподаватель читает дисциплину»

ER-диаграмма для экземпляра сущности:



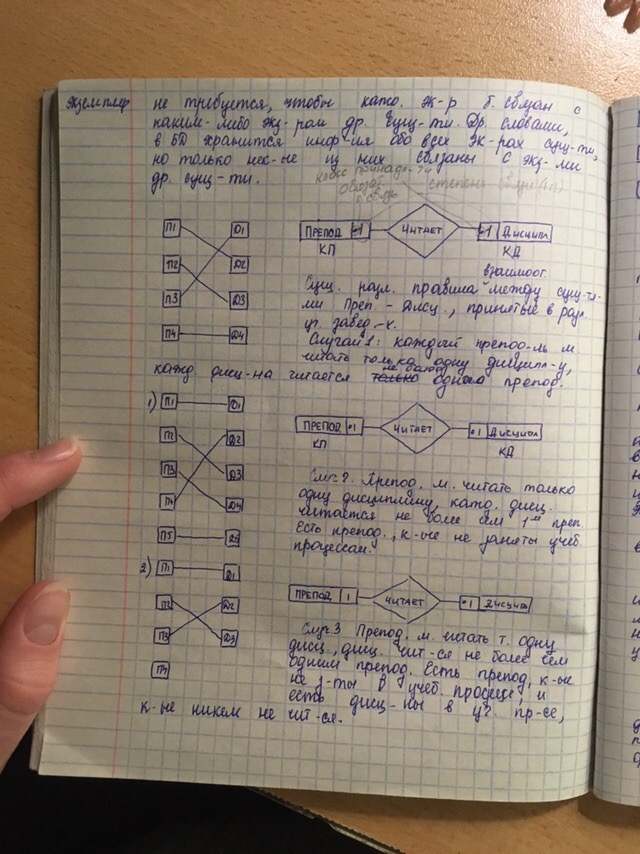
Характеристика связей

Связь между сущностями имеет две характеристики: степень связи и класс принадлежности сущности к связи. Значения этих характеристик могут быть определены из анализа связей между экземплярами сущностей. Степень связи показывает, сколько экземпляров одной сущности могут быть связаны с каждым экземпляром другой сущности. Степень связи оценивается с каждой из сторон. Степень связи может иметь одно **из трёх значений:**

1. один к одному;
2. один ко многим (либо 1:М, либо М:1);
3. многие ко многим (M:N).

Связь один к одному означает, что каждый экземпляр первой сущности может быть связан только с одним экземпляром второй сущности и наоборот. Связь один ко многим означает, что каждый экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, а каждый экземпляр второй сущности может быть связан только с одним экземпляром первой сущности. Связь многие ко многим означает, что каждый экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, и наоборот.

Класс принадлежности сущности может быть обязательным и необязательном. При обязательном классе принадлежности каждый экземпляр сущности обязательно должен быть связан с другой сущностью, при необязательном классе принадлежности не требуется, чтобы экземпляр был связан с каким-либо экземпляром другой сущности. Другими словами, в БД хранится информация обо всех экземплярах сущности, но только несколько из них связаны с экземплярами другой сущности.



Получение отношений из диаграмм ER-типа

Общий подход к БД с использованием ER-методы состоит в выполнении **следующих шагов:**

1. построение диаграмм ER-типа, включающие в себя все сущности и связи, важнейшие с точки зрения интересов организации;
2. анализ связи и определения их характеристик: степень связи и класс принадлежности;
3. построение набора предварительного отношения с указанием предполагаемого первичного ключа для каждого отношения;
4. подготовка списка всех представляющих интерес атрибутов (тех из них, которое не были перечислены в диаграмме ER-типа в качестве ключей сущности) и назначение каждого из этих атрибутов одному из предварительных отношений с тем условием, чтобы эти отношения находились в НФБК;
5. проверки, все ли полученные отношения находятся НФБК;
6. построение схемы данных;
7. если полученное в итоге отношение не находится в НФБК или если некоторым атрибутам не находится логических обоснований в предварительных отношениях, то в этих случаях необходимо пересмотреть ER-диаграммы на предмет устранения возможных затруднений.

Предварительные отношения из ER-диаграмм

Предварительные отношения для бинарных связей 1:1

Перечень обязательных правил генераций отношений из диаграмм ER-типа.

**Правило** **1**. Если степень бинарной связи 1:1 и класс принадлежности обеих сущностей является обязательным, то требуется только одно отношение. Первичным ключом этого отношения может быть любой из 2-х сущностей.

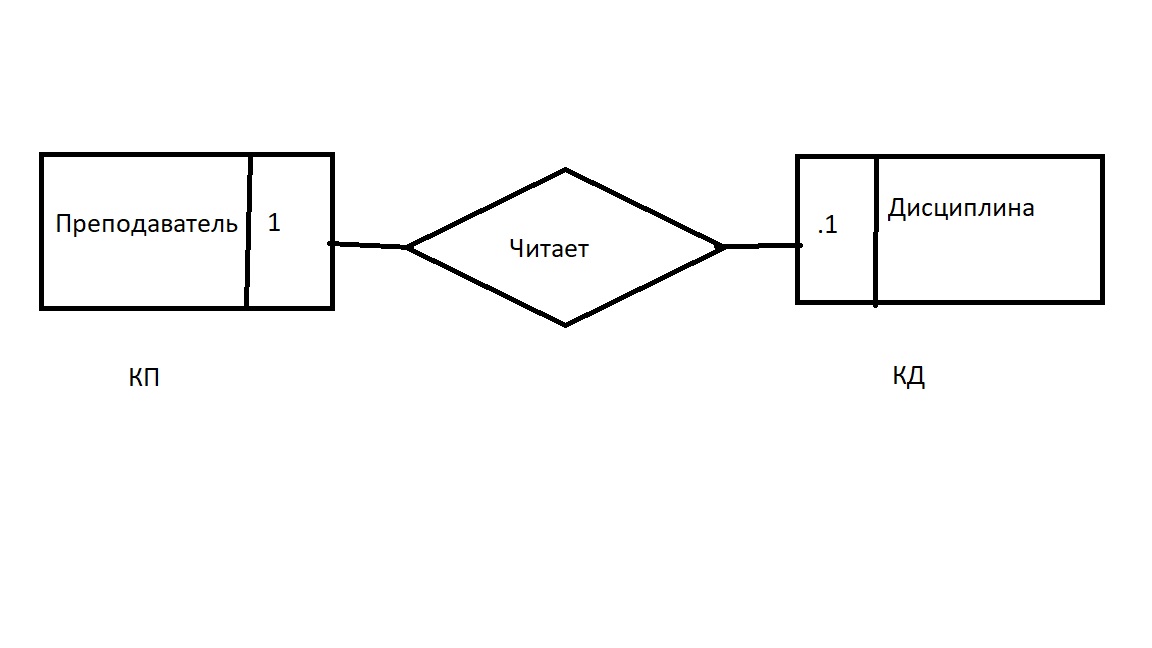
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| КП | Фамилия | Телефон | КД | Дисциплина | Часы |
| П1 | Иванов | 111 | Д1 | Физика | 100 |
| П2 | Петров | 222 | Д2 | Химия | 110 |
| П3 | Сидоров | 333 | Д3 | БД | 70 |
| П4 | Репин | 444 | Д4 | Математика | 85 |

Гарантируется однократное появление каждого значения КП и КД. Отношение никогда не будет содержать ни пустых данных, ни повторяющихся групп избыточных данных.

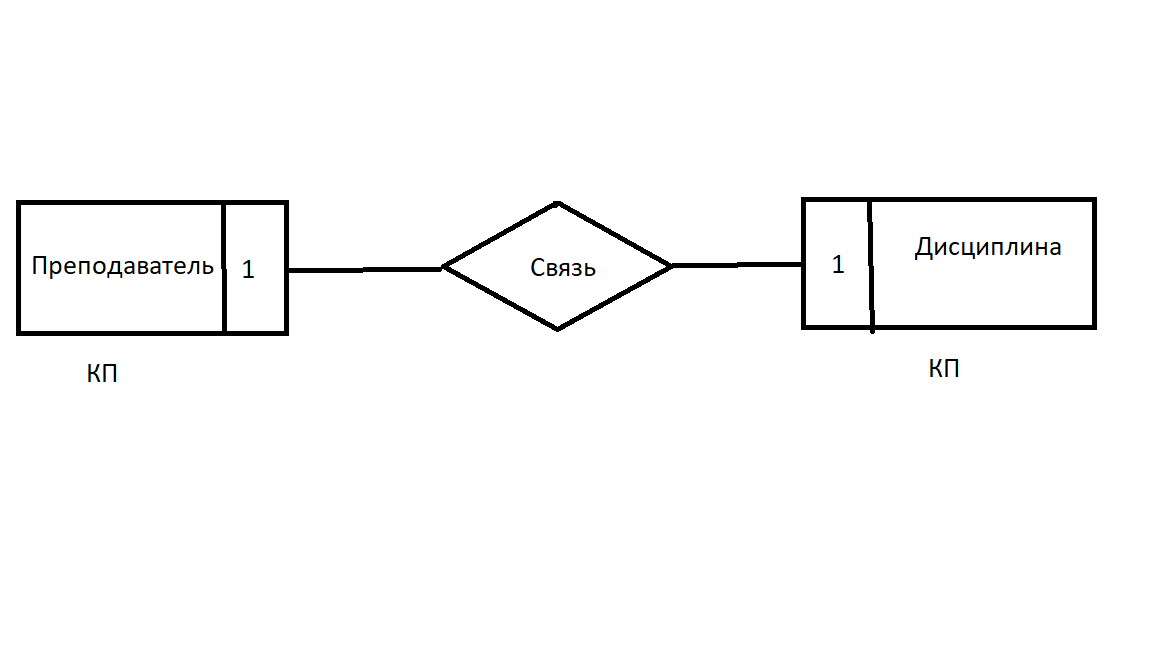
**Правило 2**. Если степень бинарной связи 1:1 и класс принадлежности одной сущности является обязательным, а другой – необязательным, то необходимо построение двух отношений. Под каждую сущность выделяется одно отношение, при этом ключ сущности должен служить первичным ключом для соответствующего отношения. Кроме того, ключ сущности, для которого класс принадлежности является необязательным, добавляется в качестве атрибута в отношение, выделенное для сущности с обязательным классом принадлежности. Пример. Преподаватель читает дисциплину. Построить самостоятельно ER-диаграмму и отношение.

|  |  |  |
| --- | --- | --- |
| КП | Фамилия | Телефон |
| П1 | Иванов | 111 |
| П2 | Петров | 222 |
| П3 | Сидоров | 333 |
| П4 | Репин | 444 |
| П5 | Хвостиков | 555 |

|  |  |  |  |
| --- | --- | --- | --- |
| КД | Дисциплина | Часы | КП |
| Д1 | Физика | 100 | П1 |
| Д2 | Химия | 110 | П2 |
| Д3 | БД | 70 | П3 |
| Д4 | Математика | 85 | П4 |



**Правило 3**. Если степень бинарной связи равна 1:1 и класс принадлежности ни одной из сущностей не является обязательной, то необходимо использовать 3 отношения: по одному для каждой сущности и одно отношение для связи. При чем ключ каждой сущности используется в качестве первичного ключа отношения. Отношение связи должно иметь в числе своих атрибутов ключи каждой сущности.



|  |  |  |
| --- | --- | --- |
| КП | Фамилия | Телефон |
| П1 | Иванов | 111 |
| П2 | Петров | 222 |
| П3 | Сидоров | 333 |
| П4 | Репин | 444 |
| П5 | Хвостиков | 555 |

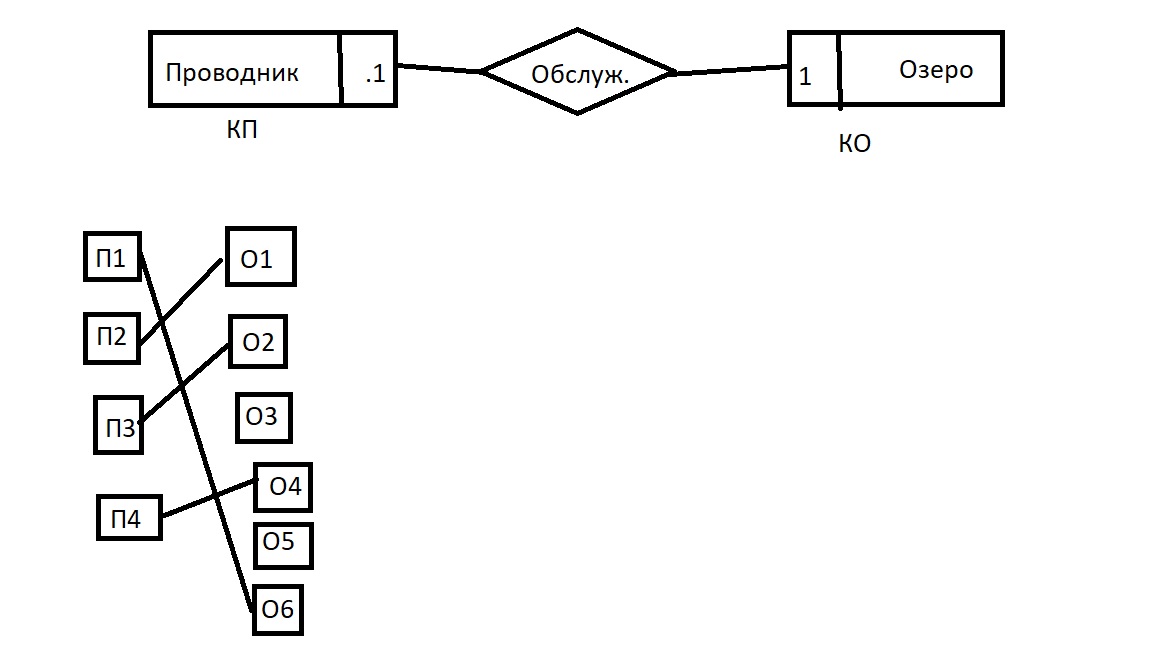
|  |  |  |
| --- | --- | --- |
| КД | Дисциплина | Часы |
| Д1 | Физика | 100 |
| Д2 | Химия | 110 |
| Д3 | БД | 70 |
| Д4 | Математика | 85 |
| Д5 | Философия | 90 |

|  |  |
| --- | --- |
| КП | КД |
| П1 | Д1 |
| П2 | Д2 |
| П3 | Д3 |
| П4 | Д4 |

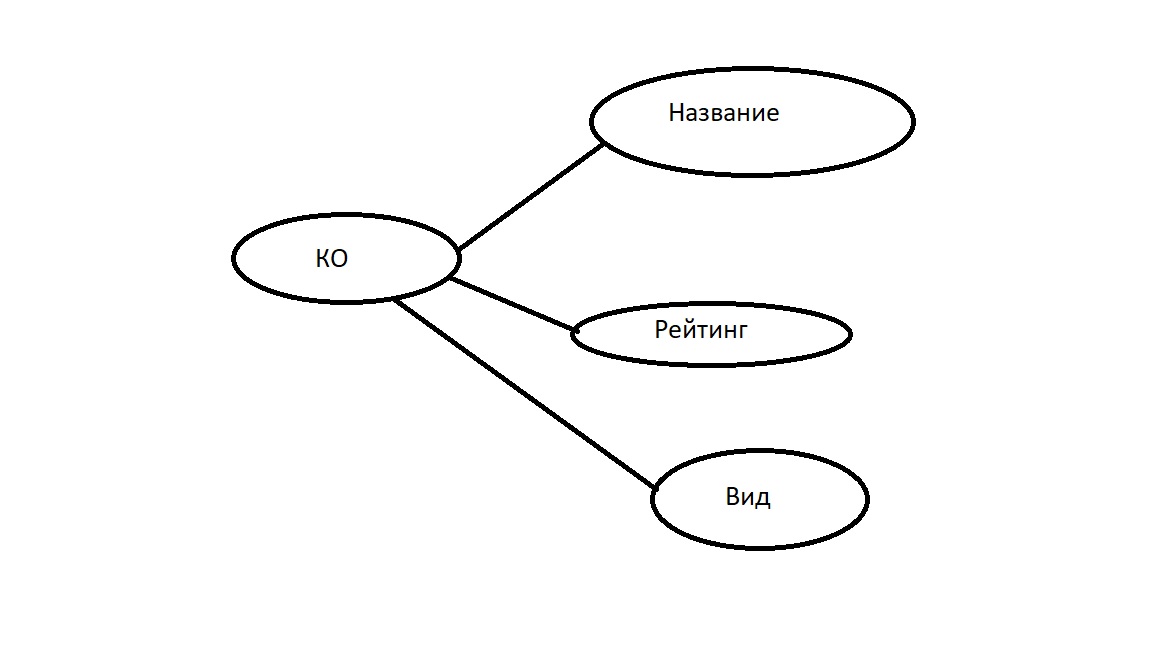
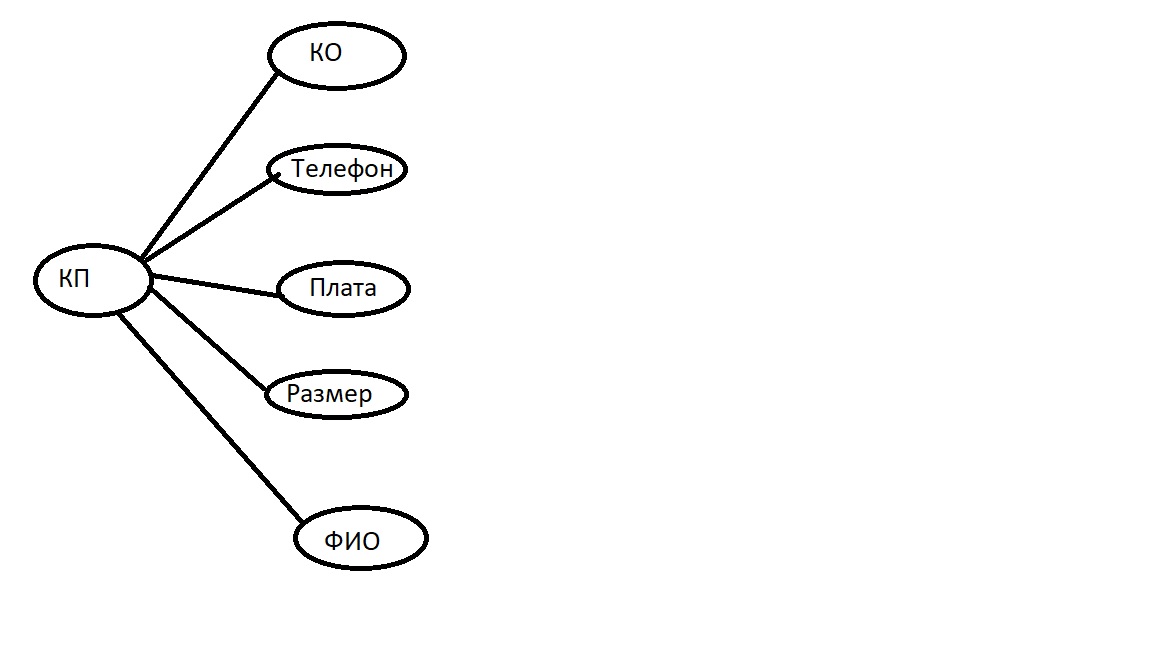
Пример. Постановка задачи. Предметная область: профессиональные рыболовные проводники мещерских озёр и озёра, которые они обслуживают. Разрешается закрепление не более одного проводника за одним озером, и, по соглашению между проводниками, каждый из них обслуживает только одно озеро. Таким образом, степень связи 1:1.

Атрибуты: имя проводника, код проводника, номер телефона, ежедневная плата, максимально допустимое число людей в группе рыбаков, название озера, код озера, рыболовный рейтинг, и основной вид вылавливаемой в озере рыбы.

Решение задачи. Предположение при создании ER-диаграммы: все проводники имеют работу, некоторые озёра проводниками не обслуживаются. Следовательно, класс принадлежности сущности «проводник» обязательный, а сущности «озеро» необязательный.



Проводник (код проводника, имя проводника, номер телефона, код озера, ежедневная плата, максимальное количество рыбаков в группе); озеро (код озера, название озера, рыболовный рейтинг, основной вид вылавливаемой рыбы).



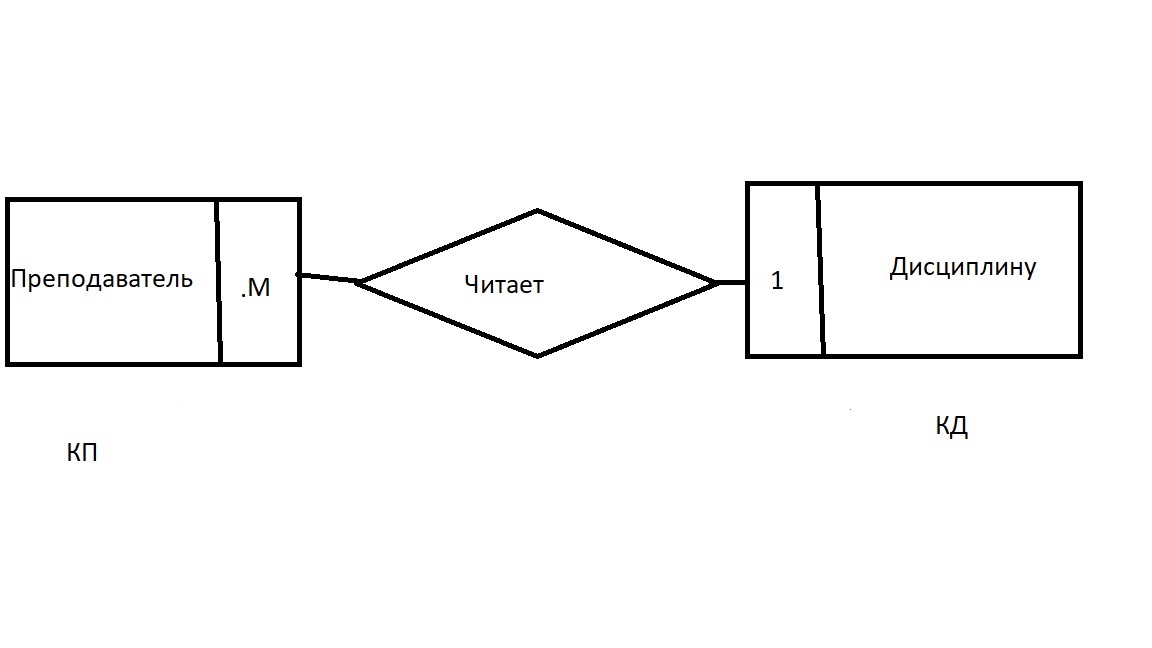
Предварительные отношения для бинарных связей 1:М

В такой ситуации используются два правила. Каждое из них определяется классом принадлежности N-связной сущности, класс принадлежности 1-связной на результат не влияет.

**Правило 4.** Если степень бинарной связи равна 1:N(M) и класс принадлежности N(M)-связной сущности является обязательным, то достаточным является использование двух отношений, по одному на каждую сущность, при условии, что ключ каждой сущности служит в качестве первичного ключа для соответствующего отношения. Дополнительно ключ 1-связанной сущности должен быть добавлен как атрибут в отношения, отводимое N(M)-связанной сущности. Пример: преподаватель читает дисциплину.

|  |  |  |  |
| --- | --- | --- | --- |
| КП | Имя | Телефон | КД |
| 1П | Жарких | 111 | Д1 |
| 2П | Петров | 222 | Д1 |
| 3П | Алексеев | 333 | Д2 |

|  |  |  |
| --- | --- | --- |
| КД | Название дисциплины | Часы |
| Д1 | История | 100 |
| Д2 | Философия | 200 |

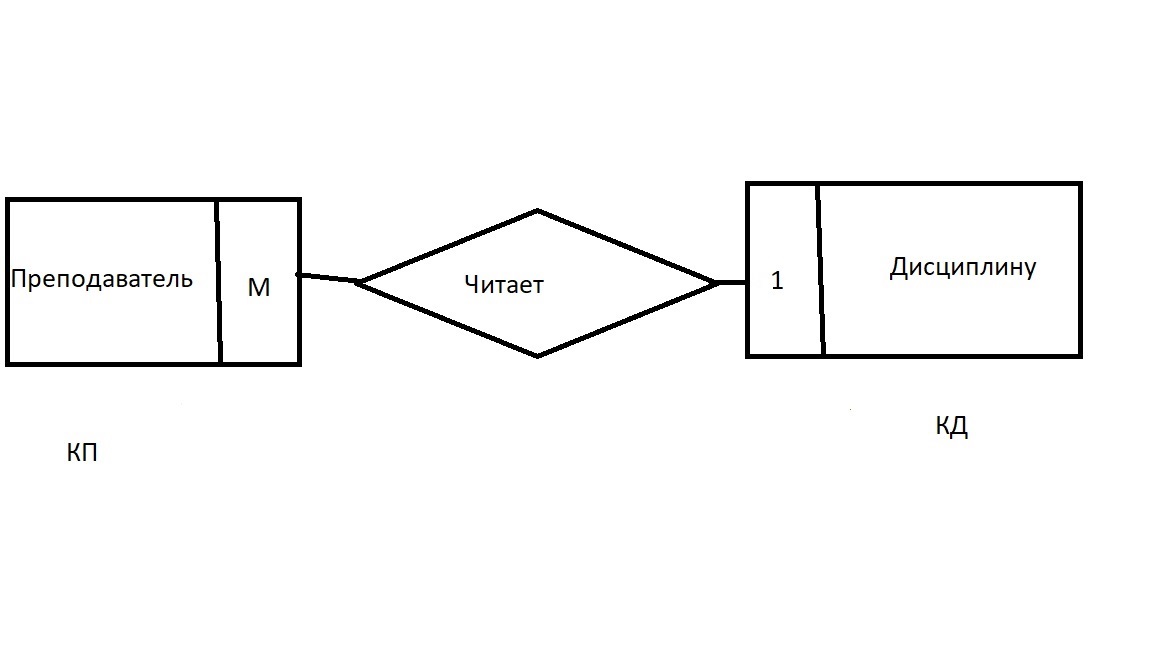


**Правило 5**. Если степень бинарной связи равна 1:М(N) и класс принадлежности N(M)-связной сущности является необязательным, то необходимо формирование трёх отношений: по одному для каждой сущности и одно отношение для связи. При чем ключ каждой сущности используется в качестве первичного ключа соотв. отношения. Отношение связи должно иметь в числе своих атрибутов ключи каждой сущности.

|  |  |  |
| --- | --- | --- |
| КП | Имя | Телефон |
| 1П | Жарких | 111 |
| 2П | Петров | 222 |
| 3П | Алексеев | 333 |

|  |  |  |
| --- | --- | --- |
| КД | Название дисциплины | Часы |
| Д1 | История | 100 |
| Д2 | Философия | 200 |

|  |  |
| --- | --- |
| КД | КД |
| 1П | Д1 |
| 2П | Д1 |
| 3П | Д2 |



Предварительные отношения для бинарных связей степени N:M

**Правило 6.** Если степень бинарной связи N:M, то для хранения данных необходимо три отношения: по одному для каждой сущности и одно – для связи. При чем ключ каждой сущности используется в качестве первичного ключа соотв. отношения. Отношение связи должно иметь в числе атрибутов ключи каждой сущности.

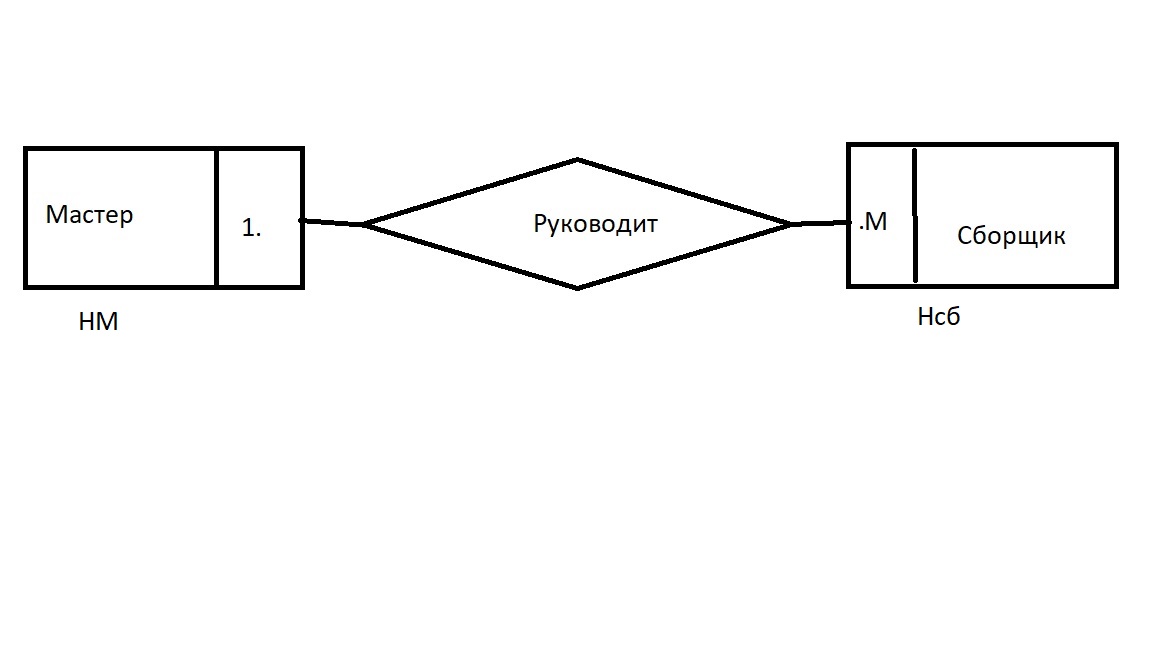
Связи более высокого порядка

**Правило 7.** В случае трёхсторонней связи необходимо использовать четыре (n+1) предварительных отношения. N-количество сущностей. По одному для каждой сущности, и одно – для связи. При чем ключ каждой сущности должен служить в качестве первичного ключа для соотв. отношения. Отношение, порождаемое связью, будет иметь среди своих атрибутов ключи от каждой сущности. Аналогично, когда связь N(M)-сторонняя, требуется n+1 предварительных отношений.

Использование ролей

В некоторых ситуациях одних сущностей и связей может оказаться недостаточно для полноценного моделирования организации. Одна из таких ситуаций возникает тогда, когда экземпляры некоторой сущности должны играть разные роли в деятельности предприятия.

Рассмотрим задачу. В ней идет речь о небольшом предприятии - поставщики автомобилей. На предприятии две категории служащих: мастера и сборщики.



В отношение необходимо добавить следующие атрибуты: СлФам (фамилия служащего), РТел (рабочий телефон мастера), ДТел (домашний телефон служащего), СлАдрес (адрес служащего), Тставка (почасовая тарифная ставка сборщика), оклад (месячный оклад мастера), СбКод (рабочий код сборщика), СфКомп (сфера компетенции мастера).

Мастер (НМ, СлФам, РТел, ДТел, СлАдрес, Оклад, СфКомп).

Сборщик (НСб, СлФам, ДТел, СлАдрес, Тставка, СбКод, НМ). - **неправильное.**

В результате предварительные отношения будут преобразованы к виду:

Мастер (НМ, Оклад, Сфкомп);

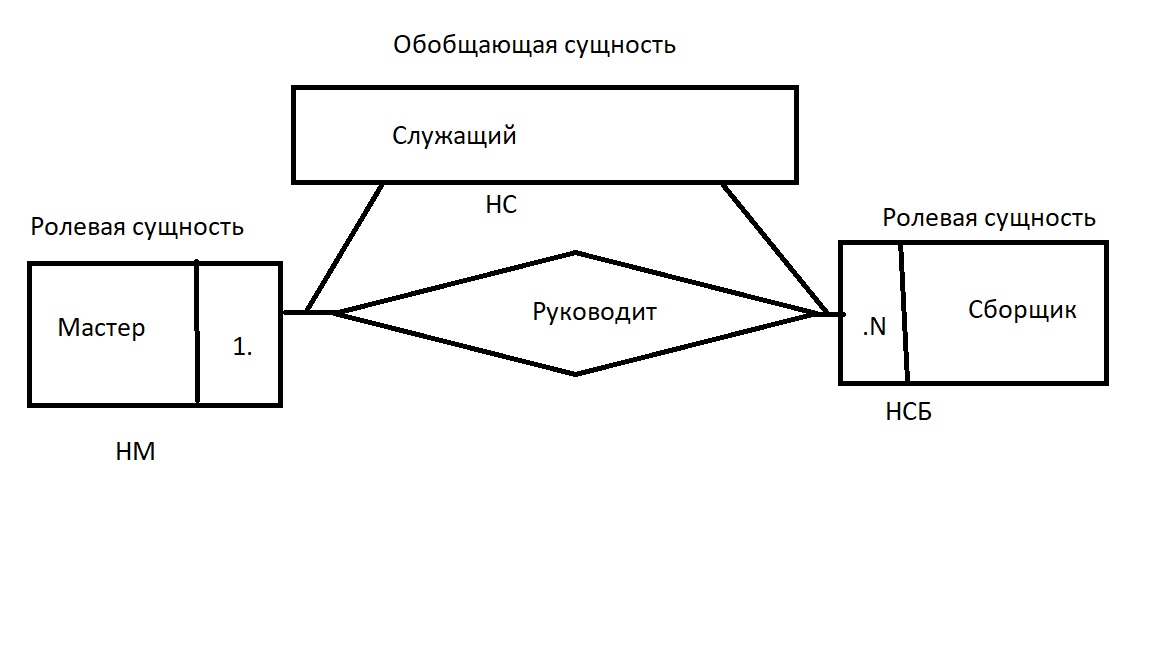
Сборщик (Нсб, Тставка, Сбкод, НМ);

Остальные атрибуты: Дтел, СлФам, СлАдрес – можно продублировать в обоих отношениях под разными именами. Однако этот способ имеет ряд недостатков: во-первых, присутствует дублирование информации; во-вторых, усложняется поиск, например, при поиске домашнего телефона служащего Петрова необходимо просмотреть обе таблицы.

Можно предложить другое решение, ввести дополнительное отношение «Служащий», обобщающее сущности «Мастер» и «Сборщик», и все оставшиеся атрибуты приписать этому отношению. Прежде чем сформулировать правило, рассмотрим некоторые понятия.

**Роль, или ролевая сущность, описывает функции, выполняемые сущностью в организации. Между ролевыми сущностями могут быть связи, зависящие от деятельности организации и выполняемых ими функциями.**

**Обобщающая сущность описывает общие свойства всех ролевых сущностей. Между обобщающей сущностью и ролевыми сущностями устанавливается связь типа «дерево», иерархическая связь.**



**Правило 8.** Обобщающей сущности соответствует одно отношение, при чем ключ сущности служит в качестве ключа отношения, общие для всех ролевых сущностей атрибуты приписываются этому отношению. Ролевые элементы и связи, их соединяющие, порождают такое число отношений, которое определяется ранее описанными правилами, при чем каждая роль трактуется как обычная сущность. Связываются отношения с помощью ключевого атрибута. Каждому значению ключевого атрибута ролевой сущности соответствует одна запись обобщающего отношения с таким же значением ключа.

**Связь между таблицами**. Из рассмотренных примеров следует, что результатом проектирования по методу «сущность-связь» является совокупность связанных таблиц. Связь между таблицами осуществляется через одинаковые по смыслу, типу, значению атрибутов. Назовем их атрибутами связи. В одних таблицах эти атрибуты играют роль ключевых, в других - вводятся дополнительно по правилам 1-8 для отображения связи.

Рекомендации по разработки структур. В качестве обобщения материала предыдущего подраздела приведем наиболее важные рекомендации. Неучёт которых может привести к аномалиям при обработке данных в БД. Остановимся на двух вопросах: исходя из каких соображений нужно создавать отношения и каким образом следует их связывать. Какими должны быть таблицы сущностей.

**Основное правило при создании таблиц сущностей – это «каждой сущностей – отдельную таблицу».** Поля таблиц сущности могут быть двух видов: ключевые и не ключевые. Ведение ключей в таблице практически во всех реляционных СУБД позволяет обеспечить уникальность значений в записях таблице по ключу, ускорить обработку записей таблицы. А также выполнить автоматическую сортировку записей по значениям в ключевых полях.

Обычно достаточно определения простого ключа. Реже – вводят составной ключ. В некоторых СУБД пользователям предлагается определить создаваемое ключевое поле нумерации (в Access – это поле типа «счётчик»), которое упрощает решение проблемы уникальности записей таблицы.

Иногда в таблицах сущностей имеются поля описания свойств или характеристик объектов. Если в таблице есть значительное число повторений по этим полям и эта информация имеет существенный объем, то лучше их выделить в отдельную таблицу (придерживаясь правила: «каждой сущности – отдельную таблицу»). Тем более, следует образовать дополнительную таблицу, если свойства – взаимосвязанные. В более общем виде последние рекомендации можно сформулировать так: информацию о сущностях следует представить таким образом, чтобы неключевые поля в таблицах были взаимно независимыми и полностью зависели от ключа (см. 3НФ). В процессе обработки таблиц сущностей надо иметь ввиду следующее: новую сущность легко добавить и изменить, но при удалении следует уничтожить все ссылки на неё из таблиц связей, иначе таблицы связей будут некорректными. Многие современные СУБД блокируют некорректные действия в подобных операциях.

Организация связей сущностей

Записи таблицы связей предназначены для отображения связей между сущностями, информация о которых находится в соответствующих таблицах сущностей. Обычно одна таблица связи описывает взаимосвязь двух сущностей. Поскольку таблицы сущностей в простейшем случае имеют по одному ключевому полю, то таблица связей двух сущностей должна иметь два атрибута.

**Более сложные связи (не-бинарные) следует сводить к бинарным.** Для описания взаимосвязей N(M)-объектов требуется N-1 таблицей связи. Транзитивных связей не должно быть. Избыток связей приводит к противоречиям.

**Не следует включать в таблицы связей характеристики сущностей, иначе также неизбежны аномалии. Их лучше хранить в отдельных таблицах сущностей.**

**С помощью таблиц связей можно описывать несколько специфичный вид связей – линейную связь, или слабую связь**. Примером линейной связи можно считать отношение принадлежности сущностей некоторой другой сущности более высокого порядка (системы, состоящие из узлов; лекарство, состоящее их компонентов; сплавы металлов и т.д.). В этом случае для описания связей достаточно одной таблицы связи.

При работе с таблицами связей следует иметь ввиду, что любая запись из таблицы связей легко может быть удалена, поскольку сущности некоторое время могут обойтись и без связей. При добавлении или изменении содержимого записей таблицы надо контролировать правильность ссылок на существующие объекты, т.к. связь без объектов существовать не может. Большинство современных СУБД контролируют правильность ссылок на объекты.

Обеспечение целостности

**Под целостностью понимают свойство БД, означающее, что она содержит полную, не противоречивую и адекватно отражающую предметную область информацию. Различают физическую и логическую целостность. Физическая целостность означает наличие физического доступа к данным и то, что данные не утрачены. Логическая целостность означает отсутствие логических ошибок в БД, к которым относятся нарушения структуры БД, или её объектов, удаление или изменение установленных связей между объектами и т.д.**

Поддержание целостности БД включает проверку (контроль) целостности и её восстановление в случае обнаружения противоречия в базе. Целостное состояние БД задаётся с помощью ограничения целостности в виде условий, которым должны удовлетворять хранимые в БД данные.

Среди ограничений целостности можно выделить два основных типа ограничений: ограничение значений атрибутов отношений и структурные ограничения на кортеже отношений.

Примером ограничений значений атрибутов отношений является требование недопустимости пустых или повторяющихся значений в атрибутах, а также контроль принадлежности значений атрибутов заданному диапазону. Наиболее гибким средством реализации контроля значений атрибутов являются хранимые процедуры и триггеры, имеющиеся в некоторых СУБД.

**Структурные ограничения определяют требование целостности сущностей и целостности ссылок. Требование целостности сущностей состоит в том, что любой кортеж отношения должен быть отличим от любого другого кортежа этого отношения, т.е, иными словами, любое отношение должно обладать первичным ключом.**

Формулировка требования целостности ссылок тесно связано с понятием внешнего ключа.

**Напомним, что внешние ключи служат для связи отношений между собой.** При этом атрибут одного отношения (родительского) называется внешним ключом данного отношения, если он является первичным ключом другого отношения (дочернего). Говорят, что отношения, в котором определен внешний ключ, ссылается на отношения, в котором этот же атрибут является первичным ключом.

Требование целостности ссылок состоит в том, что для каждого значения внешнего ключа родительской таблицы должна найтись строка в дочерней таблице с таким же значением первичного ключа. Например, если в отношении R1 содержатся сведения о сотрудниках кафедры, а атрибут этого отношения «Должн» является первичным ключом отношения R2, то в этом отношении для каждой должности из R1 должна быть строка со соответствующим ей окладом.

Во многих современных СУБД имеются средства обеспечения контроля целостности БД.

R1 (родительская таблица)

|  |  |  |  |
| --- | --- | --- | --- |
| ФИО | Должн | Каф | Стаж |
| Иванов | Преподаватель | 25 | 5 |
| Петров | ст. преподаватель | 25 | 7 |
| Сидоров | Преподаватель | 25 | 10 |
| Егоров | Преподаватель | 24 | 5 |

Внешний ключ

R2 (дочерняя таблица)

Первичный ключ

|  |  |
| --- | --- |
| Должн | Оклад |
| Преподаватель | 500 |
| Ст. преподаватель | 800 |

Реляционная алгебра. Реляционные операции.

Операции над отношениями

Основными операциями над отношениями РМД являются восемь операций, входящих в реляционную алгебру Кодда**. Реляционная алгебра Кодда включает традиционные операции над множествами: объединение, пересечение, разность (вычитание), декартово произведение – и специальные операции: выбор, проекция, соединение, деление**. Совокупность этих операций образует замкнутую алгебру отношений. Замкнутость определяется тем, что аргументами операций реляционной алгебры являются отношения и результатом обработки всегда является новое отношение, которое также может быть аргументом в другой операции (по аналогии с обычной алгеброй чисел). Рассмотри основные операции реляционной алгебры.

**Объединение – операция выполняется над двумя совместными отношениями R1, R2 (с идентичной структурой d1, d2, d3, … dn) в результате операции объединения строится новое отношение R=R1UR2. Отношение R имеет тот же состав атрибутов и совокупность кортежей исходный отношений.** Причем в эту совокупность не включаются дубликаты по определению отношения.

Объединением двух совместимых отношений R1 и R2 одинаковой размерности (R1 UNION R2) является отношение R, содержащая все элементы исходных отношений (с исключением повторений). Пример.

R1 «Клиенты банка А»

|  |  |
| --- | --- |
| Город | Фамилия |
| Москва | Петров |
| Орёл | Смирнов |
| Ливны | Соколов |

R2 «Клиенты банка Б»

|  |  |
| --- | --- |
| Город | Фамилия |
| Мценск | Петров |
| Москва | Петров |
| Болхов | Семёнов |

R

|  |  |
| --- | --- |
| Город | Фамилия |
| Москва | Петров |
| Орёл | Смирнов |
| Ливны | Соколов |
| Мценск | Петров |
| Болхов | Семёнов |

**Пересечение – операция выполняется над двумя совместными отношениями R1, R2. Результирующее отношение R, которое является R=R1nR2 (пересечение) содержит кортежи, которые есть в каждом из двух исходных отношений.** Результат пересечения имеет тот же состав атрибутов, что и в исходных отношениях.

Пересечение двух совместимых отношений R1 и R2 одинаковой размерности (R1 INTERSECT R2) порождает отношение R с телом, включающим в себя кортежи, одновременно принадлежащие обоим исходным отношениям.

R

|  |  |
| --- | --- |
| Город | Фамилия |
| Москва | Петров |

**Вычитание – операция выполняется над двумя совместными отношениями R1, R2. В результате операции вычитания строится новое отношение R=R1-R2 с идентичным набором атрибутов, содержащее только те кортежи первого отношения R1, которые не входят в другое отношение R2.**

Вычитание совместимых отношении одинаковой размерности (R1 MINUS R2) есть отношение тело которого состоит из множества кортежей принадлежащих R1, но не принадлежащих отношению R2.

R (R1-R2)

|  |  |
| --- | --- |
| Город | Фамилия |
| Орёл | Смирнов |
| Ливны | Соколов |

R (R2-R1)

|  |  |
| --- | --- |
| Город | Фамилия |
| Мценск | Петров |
| Болхов | Семёнов |

**Декартово произведение выполняется над двумя отношениями R1, R2, имеющими в общем случае разный состав атрибутов: {d1,d2,d3,…dn}{p1,p2,p3,…pm}. В результате операции образуется новое отношение R, являющееся произведением R=R1xR2, которое включает все атрибуты исходных отношений {d1,d2,d3,…dn,p1,p2,p3,…pn}. Результирующее отношение состоит из всевозможных сочетаний кортежей исходных отношений R1 и R2. Мощность декартового отношения равна произведению мощностей исходных отношений.**

Произведение отношения R1 степени k1 и отношения R2 степени k2 (R1 TIMES R2), которые не имеют одинаковых имён атрибутов, есть такое отношение R степени (k1+k2), заголовок которого представляет сцепление заголовков отношений R1 и R2, а тело имеет картежи такие, что первые k1 элементов картежей принадлежат множеству R1, а последние k2 элементов – множеству R2. (R1 2-3, R2 3-5)

|  |  |
| --- | --- |
| **T** | **Y** |
| I | IV |
| II | V |
| III | VI |

|  |  |  |
| --- | --- | --- |
| **O** | **L** | **N** |
| X | XI | XII |
| XIII | XIV | XV |
| XVI | XVII | XVIII |
| XIX | XX | XXI |
| XXII | XXIII | XXIV |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **T** | **Y** | **O** | **L** | **N** |
| I | IV | X | XI | XII |
| II | V | XIII | XIV | XV |
| III | VI | XVI | XVII | XVIII |
| I | IV | XIX | XX | XXI |
| II | V | XXII | XXIII | XXIV |
| III | VI | X | XI | XII |
| I | IV | XIII | XIV | XV |
| II | V | XVI | XVII | XVIII |
| III | VI | XIX | XX | XXI |
| I | IV | XXII | XXIII | XIV |
| II | V | X | XI | XII |
| III | VI | XIII | XIV | XV |
| I | IV | XVI | XVII | XVIII |
| II | V | XIX | XX | XXI |
| III | VI | XXII | XXIII | XXIV |

Для выполнения выбора по заданному условию (предикату) осуществляется выборка подмножества кортежей. Результирующее отношение имеет ту же структуру, что и исходное отношение, но число его кортежей будет меньше (или равно) числа кортежей исходного отношения.

**Выборка (R WHERE f). Отношение R по формуле f представляет собой новое отношение с таким же заголовком и телом, состоящим из таких кортежей отношения R, которые удовлетворяют истинности логического выражения, заданного формулой f.**

**Проекция. Операция выполняется над одним отношением R. Операция проекция формирует новое отношения с заданным подмножеством атрибутов исходного отношения R.** Оно может содержать меньше кортежей, так как после отбрасывания в исходном отношении R части атрибутов (возможного исключения первичного ключа) могут образоваться кортежи-дубли, которые из результирующего отношения исключаются по определению.

Проекция отношения A на атрибуты X, Y, etc Z (A [X, Y, … Z]), где множество {x,y,…z} является подмножеством полного списка атрибутов заголовка отношения A, представляет собой отношения с заголовком X, Y, etc Z и телом, содержащим кортежи отношения A, за исключением повторяющихся кортежей. Повторение одинаковых атрибутов в списке X, Y, etc Z запрещается.

Пример. Ниже приведен пример исходного отношения R «Служащий» и результат проекции RP этого отношения на два его атрибута – «Должность» и «Номер отдела».

R «Служащий»

|  |  |  |
| --- | --- | --- |
| Служащий | Номер отдела | Должность |
| Иванов | 01 | Инженер |
| Петров | 02 | Инженер |
| Нестеров | 01 | Инженер |
| Никитич | 02 | Лаборант |

RP

|  |  |
| --- | --- |
| Номер отдела | Должность |
| 02 | Инженер |
| 01 | Инженер |
| 02 | Лаборант |

**Соединение – выполняется для заданного условия соединения над двумя логически связанными отношениями. Исходное отношение R1 и R2 имеют разные структуры, в которых есть одинаковые атрибуты – ключи связи.** Операция соединения формирует новое отношение, структура которого является совокупностью всех атрибутов исходных отношений. Результирующие кортежи формируются соединением каждого кортежа из R1 с теми кортежами R2, для которых выполняется условие соединения.

Соединение (Cs(R1,R2)) отношений R1 и R2 по условию заданному формулой f представляет собой отношение R, которое можно получить путем декартового произведения отношений R1 и R2 с последующим применением к результату операции выборки по формуле f.

**Правила записи формулы f такие же, как и для операции селекции.**

**В зависимости от условия выборки соединение называется: естественным – равенство значений общих атрибутов R1 и R2, эквисоединением - равенство значений атрибутов, входящих в условие соединения; тета-соединением – другой знак сравнения.**

**Деление – операция выполняется над двумя отношениями R1 и R2, имеющими в общем случае разные структуры и часть одинаковых атрибутов.** В результате операции образуется новое отношение, структура которого получается исключением из множества атрибутов отношения R1 множества атрибутов отношения R2. Результирующие строки образуются из тех строк отношения R1, значения несовпадающих атрибутов которых неодинаковы, а значения общих атрибутов образуют отношения, совпадающие с отношением R2.

Результатом деления отношения R1 с атрибутами A и B на отношение R2 с атрибутом B (R1 DIVIDEBY R2), где A и B – простые или составные атрибуты, при чем атрибут B – общий атрибут, определенный на одном и том же домене является отношение R с заголовком A и телом, состоящим из кортежей r таких, что в отношении R1 имеются кортежи (r, s), при чём множество значений s включает множество значений атрибута B отношения R2.

Дополнительные операции реляционной алгебры, предложенные Дейтом, включают следующие: **переименование – операция позволяет изменить имя атрибута отношения.**

**Расширение – операция порождает новое отношение, похожее на исходное, но отличающееся наличием добавленного атрибута, значения которого получаются путем некоторых скалярных вычислений.**

**Подведение итогов – операция выполняет «вертикальные» или групповые вычисления.**

Рассмотренные выше операции в той или иной мере реализуются в языке манипулирования данными (DML), обеспечивающим обработку реляционных таблиц. Развитие реляционного подхода привело к созданию реляционных языков.

К таким языкам относятся, например, язык SQL (Structured Query Language), язык QBE (Query By Example) и другие языки запроса.

Структурированный язык запросов SQL

Общая характеристика языка.

**Язык SQL предназначен для выполнения операций над таблицами (создание, удаление, изменение структуры) и над данными таблиц (выборка, изменение, добавление и удаление), а также некоторых сопутствующих операций. SQL является непроцедурным языком и не содержит операторов управления, организации подпрограмм, ввода-вывода и т.п. В связи с этим SQL автономно не используется, обычно он погружен в среду встроенного языка программирования СУБД. Язык SQL не обладает функциями полноценного языка разработки. А ориентирован на доступ к данным. Поэтому его включают в состав средств разработки программ. В этом случае его называют встроенным SQL.**

Различают два основных метода использования встроенного SQL: статический и динамический.

**При статическом использовании языка (статический SQL) в тексте программы имеются вызовы функций языка SQL, которые жёстко включаются в выполняемый модуль после компиляции.** Изменения в вызываемых функциях могут быть на уровне отдельных параметров вызовов с помощью переменных языка программирования.

**При динамическом использовании языка предполагается (динамический SQL) предполагается динамическое построение вызовов SQL – функций и интерпретация этих вызовов, например, обращение к данным удаленной базы в ходе выполнения программы, динамический метод, обычно, применяется в случаях, когда в приложении заранее неизвестен вид SQL-вызова и он строится в диалоге с пользователем.**

**Основным назначением языка SQL (как и других языков для работы с БД) является подготовка и выполнение запросов**. В результате выборки данных из одной или нескольких таблиц может быть получено множество записей, называемое представлением.

**Представление по существу является таблицей, формируемой в результате выполнения запроса.** Можно сказать, что оно является разновидностью хранимого запроса. По одним и тем же таблицам можно построить несколько представлений. Само представление описывается путём указания идентификатора представления и запроса, который должен быть выполнен для его получения.

Для удобства работы с представлениями в язык SQL введено понятие «курсора».

Курсор представляет собой своеобразный указатель, используемый для перемещения по наборам записей при их обработке.

Основные операторы языка

Операторы языка SQL можно условно разделить на два подъязыка: **язык определения данных (Data Definition Language - DDL) и язык манипулирования данными (Data Manipulation Language- DML).**

Основные операторы языка SQL приведены в таблице.

|  |  |  |
| --- | --- | --- |
| Вид | Название | Назначение |
| DLL | CREATE TABLE | Создание таблицы |
| DROP TABLE | Удаление таблицы |
| ALTER TABLE | Изменение структуры т-цы |
|  | CREATE VIEW | Создание представлений |
| DROP VIEW | Удаление представлений |
| DML | SELECT | Выборка записей |
| UPDATE | Изменение записей |
| DELETE | Удаление записей |
| INSERT | Вставка новых записей |

Рассмотрим формат и основные возможности важнейших операторов. Несущественные операнды и элементы синтаксиса (например, принятая во многих системах программирования правило ставить “;” в конце оператора) будем опускать.

**Оператор создания таблица имеет формат вида:** CREATE TABLE <ИМЯ ТАБЛИЦЫ> (<ИМЯ СТОЛБЦА><ТИП ДАННЫХ>[NOT NULL] [,(<ИМЯ СТОЛБЦА><ТИП ДАННЫХ>[NOT NULL]]…)

Обязательными операндами оператора являются «имя создаваемой таблицы» и «имя хотя бы одного столбца» (поля) с указанием типов данных, хранимых в этом столбце.

При создании таблицы для отдельных полей могут указываться некоторые дополнительные правила контроля вводимых в них значений. Конструкция «NOT NULL» («не пустое») служит именно таким целям и для столбца таблицы означает, что в этом столбце должно быть определено значение.

**Пример 1.** Создание таблицы. Пусть требуется создать таблицы «Goods» описания товаров, имеющую поля: «type» – вид товара, «comp\_id» - идентификатор компании-производителя. «Name» – название товара, «price» – цена товара. **Оператор определения таблицы может иметь следующий вид:**

CREATE TABLE Goods (type SQL\_CHAR(8) NOT NULL, comp\_id SQL\_CHAR(10) NOT NULL, Name SQL\_VARCHAR(20), price SQL\_DECIMAL(8,2))

**Оператор изменения структуры таблицы имеет формат вида:** ALTER TABLE <ИМЯ ТАБЛИЦЫ> ({ADD, MODIFY, DROP}<ИМЯ СТОЛБЦА> [<ТИП ДАННЫХ>] [NOT NULL] [,{ADD, MODIFY, DROP}<ИМЯ СТОЛБЦА> [<ТИП ДАННЫХ>] [NOT NULL]]…)

**Изменение структуры таблицы может состоять в добавлении (ADD), изменении (MODIFY) или удалении (DROP) одного или нескольких столбцов таблицы. Правило записи оператора ALTER TABLE такие же, как и оператора CREATE TABLE.** При удалении столбца указывать <тип данных> не нужно.

**Пример 2.** Добавление поля таблицы.

Пусть в созданной ранее таблице «Goods» необходимо добавить поле «Number», отводимое для хранения величины запаса товаров**. Для этого следует записать оператор вида:**

ALTER TABLE Goods (ADD number SQL\_INTEGER)

Оператор удаления таблицы имеет **формат вида:**

DROP TABLE <ИМЯ ТАБЛИЦЫ>

Оператор позволяет удалить имеющуюся таблицу. Например, для удаления таблицы Goods достаточно записать **оператор вида:**

DROP TABLE Goods

Оператор выборки записей имеет формат вида:

SELECT [ALL | DISTINCT] <СПИСОК ДАННЫХ>

FROM <СПИСОК ТАБЛИЦ>

[WHERE <УСЛОВИЕ ВЫБОРК>]

[GROUP BY <ИМЯ СТОЛБЦА>[, <ИМЯ СТОЛБЦА>]…]

[HAVING <УСЛОВИЕ ПОИСКА>]

[ORDER BY <СПЕЦИФИКАЦИЯ>[, <СПЕЦИФИКАЦИЯ>]…]

Это наиболее важный оператор из всех операторов SQL. Функциональные возможности его огромны. Рассмотрим основные из них.

Оператор SELECT позволяет производить выборку и вычисления над данными из одной или нескольких таблиц. Результатом выполнения оператора является ответная таблица, которая может иметь (ALL), или не иметь (DISTINCT) повторяющиеся строки. По умолчанию в ответную таблицу включаются все строки, в том числе, и повторяющиеся. В отборе данных участвуют записи одной или нескольких таблиц, перечисленных в списке операнда FROM.

Список данных может содержать имена столбцов, участвующих в запросе, а также выражения над столбцами. В простейшем случае в выражениях можно записывать имена столбцов, знаки арифметических операций, константы и круглые скобки. Если в списке данных записано выражение, то наряду с выборкой данных выполняются вычисления, результаты которого попадают в новый (создаваемый) столбец ответной таблицы.

При использовании в списках данных имен столбцов нескольких таблиц для указания принадлежности столбца некоторой таблицы применяют **конструкцию вида:**

<ИМЯ ТАБЛИЦА>.<ИМЯ СТОЛБЦА>

Операнд WHERE задает условия, которым должны удовлетворять записи в результирующей таблице. Выражение <УСЛОВИЕ ВЫБОРКИ> является логическим. Его элементами могут быть имена столбцов, операции сравнения, арифметические операции, логические связки (AND, OR, NOT), скобки, специальные функции (LIKE, NULL, IN etc).

Операнд GROUP BY позволяет выделять в результирующем множестве записей группы. Группой является записи с совпадающими значениями в столбцах, перечисленных за ключевыми словами GROUP BY.

Выделение групп требуется для использования в логических выражениях операнда WHERE и HAVING, а также для выполнения операций (вычислений) над группами. В логических и арифметических выражениях можно использовать следующие групповые операции (функции): AVG (среднее значение в группе), MAX (максимальное значение в группе), MIN (минимальное значение в группе), SUM (сумма значений в группе), COUNT (число значений в группе).

Операнд HAVING действует совместно с GROUP BY и используется для дополнительной селекции записей во время определения групп. Правило записей, <УСЛОВИЕ ПОИСКА> аналогично правилам формирования <УСЛОВИЕ ВЫБОРКИ> операнда WHERE.

Операнд ORDER BY задаёт порядок сортировки результирующего множества. Обычно, каждая <СПЕЦИФИКАЦИЯ> представляет собой пару вида: <ИМЯ СТОЛБЦА> [ASC | DESC].

Замечание. Оператор SELECT может иметь и другие более сложные синтаксические конструкции.

**Поясним их смысл.**

Одной из таких конструкций, например, являются так называемые «подзапросы». Они позволяют формулировать вложенные запросы, когда результаты одного оператора SELECT используются в логическом выражении условия выборки операнда WHERE другого оператора SELECT. Вторым примером более сложной формы оператора SELECT является оператор, в котором отобранные записи предполагается в дальнейшем модифицировать (конструкция FOR UPDATE OF).

**СУБД после выполнения такого оператора блокирует (защищает) отобранные записи от модификации их другими пользователями.**

Ещё один случай специфического использования оператора SELECT — выполнение объединений результирующих таблиц при выполнении нескольких оператор SELECT (операнд UNION).

**Пример 4.1.** Выбор записей. Для таблицы EMP, имеющей поля: NAME, SAL, MGR и DEPT, требуется вывести имена сотрудников и размер их зарплаты, увеличенный на 100 ед. Оператор выбора можно записать **следующим образом:**

SELECT NAME, SAL+100

FROM EMP

**Пример 4.2.** Выбор с условием. Вывести названия таких отделов таблицы EMP, которых в данных момент отсутствуют руководители. Оператор SELECT для этого запроса **можно записать так**:

SELECT DEPT

FROM EMP

WHERE MGR IS NULL

**Пример 4.3.** Выбор с группированием. Пусть требуется найти MAX и MIN зарплаты для каждого из отделов (по таблице EMP**). Оператор SELECT для этого запроса.**

SELECT DEPT, MAX (SAL), MIN (SAL)

FROM EMP

GROUP BY DEPT

Оператор изменения записей имеет формат вида:

UPDATE <ИМЯ ТАБЛИЦЫ>

SET <ИМЯ СТОЛБЦА>={<ВЫРАЖЕНИЕ>, NULL}

[, SET <ИМЯ СТОЛБЦА>={<ВЫРАЖЕНИЕ>, NULL}…]

[WHERE <УСЛОВИЕ>]

Выполнение оператора UPDATE состоит в изменении значений в определенных операндом SET столбцах таблицы для тех записей, которые удовлетворяют условию, заданному операнду WHERE.

Новые значения полей записей могут быть пустыми (NULL), либо вычисляться в соответствии с арифметическим выражением. Правило записи арифметический и логических выражений аналогично соответствующим правилам оператора SELECT.

**Пример 5**. Изменение записей. Пусть необходимо увеличить на 500 ед. зарплату тем служащим, которые получают не более 6000 (по таблице EMP). Запрос, сформулированный с помощью оператора UPDATE, **может выглядеть так:**

UPDATE EMP

SET sal=sal+500

WHERE sal<=6000

Оператор вставки новых записей имеет форматы двух видов:

INSERT INTO <ИМЯ ТАБЛИЦЫ>

[(<СПИСОК СТОЛБЦОВ>)]

VALUES(<СПИСОК ЗНАЧЕНИЙ>)

INSERT INTO <ИМЯ ТАБЛИЦЫ>

[(<СПИСОК СТОЛБЦОВ>)]

<ПРЕДЛОЖЕНИЕ SELECT>

В первом формате оператор INSERT предназначен для ввода новых записей с заданными значениями в столбцах. Порядок перечисления имён столбцов должен соответствовать порядку значений, перечисленных в списке операнда VALUES.

Если <СПИСОК СТОЛБЦОВ> опущен, то в <СПИСОК ЗНАЧЕНИЙ> должны быть перечислены все значения в порядке столбцов структуры таблицы.

Во втором формате оператор INSERT предназначен для ввода в заданную таблицу новых строк, отобранных из другой таблицы с помощью предложения SELECT.

**Пример 6**. Ввод записей. Ввести в таблицу EMP запись о новом сотруднике. Для этого можно **записать такой оператор вида:**

INSERT INTO EMP

(name, sal, mgr, dept, date)

VALUES («Иванов», 15000, «Петров», «МВД», #04/03/2019#)

Оператор удаления записей имеет формат вида:

DELETE FROM <ИМЯ ТАБЛИЦЫ>

[WHERE <УСЛОВИЕ>]

Результатом выполнения является удаление из указанной таблицы строк, которые удовлетворяют условию, определенном операнду WHERE.

Если необязательный операнд WHERE опущен, то есть, условие отбора удаляемых записей отсутствует, удалению подлежат все записи таблицы.

**Пример 7.** Удаление записей. В связи с ликвидацией отдела игрушек (toy) требуется удалить из таблицы EMP всех сотрудников этого отдела. Оператор DELETE для этой задачи **будет выглядеть так:**

DELETE FROM EMP

WHERE Dept=”toy”

Защита БД

Введение

Проблему защиты информации БД целесообразно рассматривать совместно с проблемой защиты вычислительной системы в целом.

Знание принципов построения систем защиты и возможностей, предоставляемых различными компонентами вычислительной системы (ОС, программами обслуживания, СУБД, специализированными пакетами защиты и отдельными устройствами) позволяет оценить уязвимость ИС и грамотно организовать в ней защиту персональной информации.

Основные определения

**Под информацией (информационном обеспечением) в ВС понимается любой вид накапливаемых, хранимых и обрабатываемых данных**. Частным случаем информации является совокупность программ (системных и прикладных), обеспечивающих различную обработку данных.

Движение информации в ЭВМ неразрывно связано с функционированием программ её обработки и обслуживанием, поэтому при рассмотрении информации в ВС говорят об информационно-программном обеспечении (ИПО).

**Целью защиты ИПО является обеспечение безопасности хранимой и обрабатываемой информации, а так же используемых программных средств.**

Для обеспечения качественной защиты в ВС недостаточно организовать охрану помещений и установить программные средства защиты. Здесь требуется комплексный (системный) подход, предполагающий организацию защиты ВС в соответствии со структурой ВС. Задачами, решаемыми в ней, а также целями и возможностями защиты.

**Системы защиты нужно закладывать с начала разработки ИС и её ИПО и выполнять на всех этапах жизненного цикла.**

На практике, в силу ряда причин, создание системы защиты часто начинается, когда ВС и её ИПО уже разработаны. Создаваемая система защиты должна быть многоуровневой, адаптируемой к новым условиям функционирования, включать в себя рационально организованную совокупность имеющихся средств, методов и мероприятий. Защита должна быть от злоумышленников и от некомпетентных действий пользователей и обслуживающего персонала.

**Для эффективного построения системы защиты необходимо:**

1. Выделить уязвимые элементы ВС;
2. Выявить угрозы для выделенных элементов;
3. Сформировать требования к системе защиты;
4. Выбрать методы и средства удовлетворения предъявленным требованиям.

Безопасность ВС нарушается в следствие реализации одной или нескольких потенциальных угроз. Под угрозой ИПО понимается возможность преднамеренного или случайного действия, которое может привести к нарушению безопасности хранимой и обрабатываемой в ВС информации, в том числе и программ. **Основными видами угроз в ВС являются следующие:**

1. **Несанкционированного использования ресурсов ВС:**
2. использование данных (копирование, модификация, удаление, печать и т.д.);
3. копирование и модификация программ;
4. исследование программ для последующего вторжения в систему.
5. **Некорректного использования ресурсов ВС:**
6. случайный доступ прикладных программ к чужим разделам основной памяти;
7. случайный доступ к системным областям дисковой памяти;
8. некорректное изменение БД (ввод неверных данных, нарушение ссылочной целостности данных);
9. ошибочные действия пользователей и персонала.
10. **Проявления ошибок в программных и аппаратных средствах;**
11. **Перехвата данных в линиях связей и системах передачи;**
12. **Несанкционированной регистрации электро-магнитных излучений;**
13. **Хищение устройств ВС, носителей информации и документов;**
14. **Несанкционированного изменения компонентов ВС, средств передачи информации или их вывода из строя и т.д.**

Возможными последствиями нарушения защиты **являются следующие:**

1. **Получение секретных сведений;**
2. **Снижение производительности или остановка системы;**
3. **Невозможность загрузки ОС с жёсткого диска;**
4. **Материальный ущерб;**
5. **Катастрофические последствия.**

Целью защиты является обеспечение безопасности информации в ВС, которое может быть нарушена (случайно или преднамеренно), поэтому сущность защиты сводится к предотвращению угроз нарушения безопасности.

**Исходя из возможных угроз безопасности можно выделить три основные задачи защиты:**

1. **Защита информации от хищения;**
2. **Защита информации от потери:**
3. **Защита ВС от сбоев и отказов.**

**Защита информации от хищении** подразумевает предотвращения физического хищения устройств и носителей хранения информации, несанкционированного получения информации (копирование, подсмотра, перехвата и т.д.) и несанкционированного распространения программ.

**Защита информации от потери** подразумевает поддержание целостности и корректности информации, что означает обеспечение физической, логической и семантической целостности информации.

Информация в системе может быть потеряна как из-за несанкционированного доступа в систему пользователей, программ (в том числе, и компьютерных вирусов), некорректных действий пользователей и их программ, обслуживающего персонала, так и в случаях в сбоях и отказах ВС.

**Защита от сбоев и отказов аппаратно-программного обеспечения ВС** – является одним из необходимых условий нормального функционирования системы. Если ВС является ненадежной, информация в ней часто искажается и иногда утрачивается. Основная нагрузка на обеспечение хорошей защиты от сбоев и отказов в системе ложится на системные аппаратно-программные компоненты: процессор, основную память, внешние запоминающие устройства, устройства ввода-вывода и другие устройства, а также программы ОС.

При недостаточно надежных системных средствах защиту от сбоев следуют предусматривать в прикладных программах.

Под надежностью ПО понимается способность точно и своевременно выполнять возложенные на него функции. Степень надежности ПО определяется качеством и уровнем автоматизации процесса разработки, а также организацией его сопровождения.

Так как достичь 100% надежности программ на практике почти не удается, необходимо предусматривать средства быстрого восстановления работоспособности программ и данных после восстановления аппаратуры и ПО от сбоев и отказов.

**Для организации комплексной защиты информации в ВС в общем случае может быть предусмотрено четыре защитных уровня:**

1. Внешний уровень, охватывающий всю территорию расположения ВС;
2. Уровень отдельных сооружений или помещений, расположение устройств ВС и линий связей с ними;
3. Уровень компонентов ВС и внешних носителей информации;
4. Уровень технологических процессов хранения, обработки и передачи информации.

Первые три уровня обеспечивают, в основном, физического препятствие доступу путём ограждения, систем сигнализацией, организации пропускного режима, экранирование проводов и т.д.

Последний уровень предусматривает логическую защиту информации в том случае, когда физический доступ к ней имеется.

Методы и средства защиты

**Существующие методы защиты можно разделить на четыре основных класса:**

1. Физический;
2. Аппаратный;
3. Программный;
4. Организационный.

**Физическая защита используется, в основном, на верхних уровнях защиты и состоит в физическом преграждении доступа посторонних лиц в помещение ВС на пути к данным и процессу их обработки. Для физической защиты применяются следующие средства:**

1. Сверхвысокочастотные, ультразвуковые и инфракрасные системы обнаружения движущихся объектов, определение их размеров, скорости и направления их перемещения;
2. Лазерные и оптические системы, реагирующие пересечения нарушителями световых лучей;
3. Телевизионные системы наблюдения за охраняемыми объектами;
4. Кабельные системы, в которых небольшие объекты окружают кабелем, чувствительным к приближению нарушителя;
5. Системы защиты окон и дверей от несанкционированного проникновения, а также наблюдения и подслушивания;