

# Билеты

---

## 1. Назначение, компоненты базы, банка данных, СУБД. Роль и место баз данных в информационных системах

**Назначение банков данных:** банк данных предназначен для хранения больших массивов информации, быстрого поиска нужных сведений и документов. Ядром банка являются базы данных и базы знаний.

### Компоненты

- **База данных** является ядром банка данных. База данных содержит структурированные данные, организованные в таблицы, схемы и связи между ними.
- **Технические средства БнД:** универсальные ЭВМ и периферийные средства для ввода информации в базу данных и вывода информации. Если банк данных реализуется в сети, то необходимы технические средства для обеспечения ее работы.
- **Программные средства:** комплекс программ, обеспечивающих взаимодействие и функционирование всех частей информационной системы. Основу программных средств БнД составляют СУБД. Важной компонентой СУБД являются трансляторы или компиляторы для используемых ею языковых средств.
- **Языковые средства:** языки программирования и запросов, которые позволяют пользователям взаимодействовать с базой данных. Это могут быть структурированные языки запросов, такие как SQL (Structured Query Language), а также языки программирования для разработки приложений, связанных с базой данных.
- **Организационно-методические средства:** инструкции, методические и регламентирующие материалы, предназначенные для пользователей разных категорий, взаимодействующих с банком данных

### Назначение банка данных

- обеспечивать получение общих и/или детализированных отчетов по итогам работы;
- позволять легко определять тенденции изменения важнейших показателей;
- обеспечивать получение информации, критической по времени, без существенных задержек
- выполнять точный и полный анализ данных.

### Компоненты баз данных

- **Поле** — минимальный элемент базы данных, содержащий одну неделимую единицу информации. Каждое поле имеет имя и тип хранящихся в нем данных.
- **Запись** — совокупность разнородных полей, описывающая некоторую сущность предметной области.
- **Таблица базы данных** — набор однородных записей, который позволяет читать, изменять, добавлять и удалять записи, а также сортировать их по определенному условию и осуществлять поиск по заданным значениям.
- **Триггеры** – хранимые процедуры, которые автоматически срабатывают при модификации данных в таблице базы данных, с которой он связан: добавлении INSERT, удалении DELETE строки

в заданной таблице, или изменении UPDATE данных в определенном столбце заданной таблицы РБД. Они нужны для обеспечения целостности данных и реализации сложной бизнес-логики.

- **Индексы:** объект базы данных, создаваемый для повышения производительности поиска данных. Индекс формируется из значений одного или нескольких столбцов таблицы и указателей на соответствующие строки таблицы и позволяет искать строки, удовлетворяющие критерию поиска. Ускорение работы достигается в первую очередь за счёт того, что индекс имеет структуру, оптимизированную под поиск — например, сбалансированного дерева.
- **Хранимая процедура:** набор SQL- инструкций, который компилируется один раз и хранится на сервере. Хранимые процедуры очень похожи на обычные процедуры языков высокого уровня, у них могут быть входные и выходные параметры и локальные переменные, в них могут производиться числовые вычисления и операции над символьными данными, результаты которых могут присваиваться переменным и параметрам. В хранимых процедурах могут выполняться стандартные операции с базами данных (как DDL, так и DML). Кроме того, в хранимых процедурах возможны циклы и ветвления, то есть в них могут использоваться инструкции управления процессом исполнения.
- **Генератор:** генераторы предназначены для получения последовательностей уникальных чисел. Такие числа обычно используются как идентификаторы записи в таблицах, имеющих суррогатный первичный ключ.

## Компоненты СУБД

**Назначение СУБД:** СУБД должна предоставить пользователю базы данных возможность работать с ней, не вникая в детали на уровне аппаратного обеспечения. Иными словами, СУБД позволяет конечному пользователю рассматривать базу данных как объект более высокого уровня по сравнению с аппаратным обеспечением, а также предоставляет в его распоряжение набор операций, выражаемых в терминах языка высокого уровня (например, SQL)

- **Процессор запросов:** Это основной компонент СУБД, который преобразует запросы в последовательность низкоуровневых инструкций для контроллера базы данных;
- **Контроллер базы данных:** взаимодействует с пользователями прикладных программ и запросов. Контроллер базы данных принимает запросы и проверяет внешние и концептуальные схемы для определения тех концептуальных записей, которые необходимы для удовлетворения требований запросов. Затем контроллер базы данных вызывает контроллер файлов для выполнения поступившего запроса.
- **Контроллер файлов:** управляет файлами, предназначенными для хранения данных, и распределяет дисковое пространство. Он создает и поддерживает список структур и индексов, определенных во внутренней схеме. Если используются хешированные файлы, то в его обязанности входит и вызов функций хеширования для генерации адресов записей. Однако контроллер файлов не управляет физическим вводом и выводом данных непосредственно, а лишь передает запросы соответствующим методам доступа, которые считывают данные в системные буферы или записывают их оттуда на диск;
- **Препроцессор языка DML:** преобразует операторы языка DML (язык для манипулирования данными) в вызовы стандартных процедур базового языка. Для генерации соответствующего кода препроцессор языка DML должен взаимодействовать с процессором запросов;
- **Компилятор языка DDL:** преобразует DDL-команды в набор таблиц, содержащих метаданные. Затем эти таблицы сохраняются в системном каталоге, а управляющая информация – в заголовках файлов с данными;

- **Контроллер словаря:** управляет доступом к системному каталогу и обеспечивает работу с ним. Системный каталог доступен большинству компонентов СУБД.

## Роль БД в информационных системах

1. **Хранение данных:** БД предоставляют структурированное и надежное хранилище для хранения данных, которые могут быть доступны множеству пользователей и приложений. Они обеспечивают удобное и эффективное хранение больших объемов данных, таких как клиентская информация, продуктовые каталоги, финансовые данные и другие.
2. **Централизация данных:** БД позволяют централизованно хранить данные, что обеспечивает единое и точное представление информации для всех пользователей и систем в организации. Это помогает избежать дублирования данных и обеспечивает согласованность и целостность информации.
3. **Обеспечение доступа к данным:** БД предоставляют механизмы для эффективного доступа к данным. Пользователи и приложения могут выполнять запросы к БД, чтобы получать нужные данные, обновлять их или выполнять аналитические операции. БД также обеспечивают механизмы безопасности, чтобы контролировать доступ пользователей к данным и защищать информацию от несанкционированного доступа.
4. **Поддержка бизнес-процессов:** БД являются основой для реализации бизнес-процессов в информационных системах. Они позволяют организациям автоматизировать и управлять своими операциями, хранить и анализировать данные, принимать решения на основе информации и обеспечивать взаимодействие с клиентами и партнерами.
5. **Поддержка принятия решений:** БД предоставляют данные и аналитические возможности для поддержки принятия решений в организациях. Пользователи могут выполнять сложные запросы и анализировать данные, чтобы выявлять тенденции, прогнозировать результаты, оптимизировать процессы и принимать обоснованные решения.

## Этапы развития БД. Системы распределенного доступа. Настольные СУБД

### 1. Этап распределённого доступа к данным

Базы данных хранились во внешней памяти центральной ЭВМ, пользователями этих баз данных были задачи, **запускаемые в пакетном режиме**.

Интерактивный режим доступа обеспечивался **с помощью консольных терминалов**.

Программы доступа к БД писались на различных языках и запускались как обычные числовые программы. Мощные операционные системы обеспечивали возможность условно параллельного выполнения всего множества задач. Эти системы распределенного доступа - **база данных была централизованной**, хранилась на устройствах внешней памяти одной центральной ЭВМ, а доступ к ней поддерживался от многих пользователей-задач.

### 2. Эпоха Персональных Компьютеров

- Появляется большое количество СУБД, которые стали называться настольными СУБД. (предназначены для работы неподготовленными пользователями).
- Эти СУБД рассчитаны на создание и работу с БД в монопользовательском режиме.
- Встал вопрос экспорта/импорта данных.

- Эти СУБД имели развитый и удобный интерфейс.
- В этих СУБД поддерживался только внешний уровень представления реляционной модели.
- В них отсутствовали средства поддержки ссылочной и структурной целостности БД.
- Настольные СУБД привели к вырождению функций администрирования БД.
- Эти СУБД предъявляли скромные требования к аппаратному обеспечению.

### 3. Этапы развития БД. Распределенные БД. Технология интранет

Появляется множество локальных сетей. Всё больше информации передаётся между компьютерами.

**Транзакция** - последовательность операций над БД, переводящая её из одного непротиворечивого состояния в другое непротиворечивое состояние.

Все СУБД этого этапа обеспечивают **поддержку полной реляционной модели БД**, а именно:

- Структурной целостности (допустимыми в таких БД являются данные, представленные в виде отношений реляционной модели).
- Языковой целостности (языки не ниже стандарта SQL).
- Ссылочной целостности - большинство СУБД рассчитаны на многоплатформенную архитектуру.

Развитие средств администрирования БД с реализацией общей концепции средств защиты данных.

Создаётся **первый стандарт SQL** (1989г.)

Примеры:

- MS Access
- MS SQL Server
- Oracle DB

### Использование технологии доступа к данным Интранет

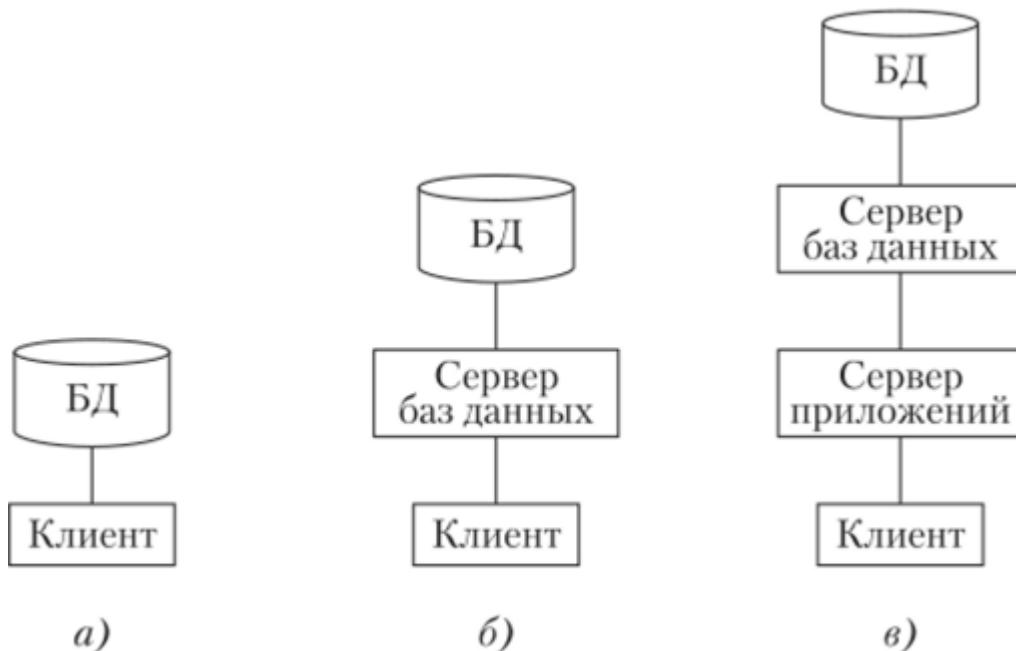
**Интранет** (англ. *Intranet*, также употребляется термин *интрасеть*) — в отличие от Интернета, это внутренняя частная сеть, принадлежащая, как правило, частному лицу, организации или крупному государственному ведомству.

#### **Характерные особенности:**

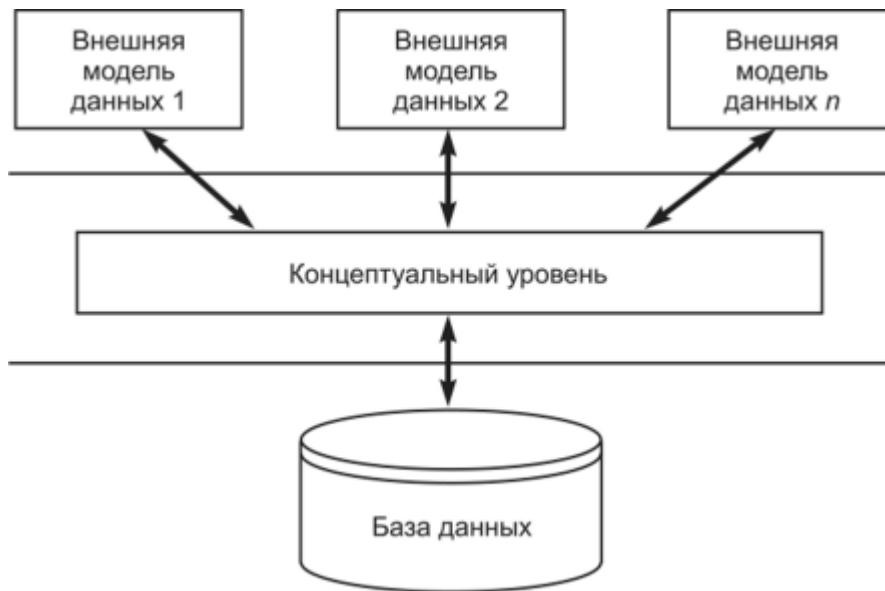
- Нет необходимости использовать специализированное клиентского ПО.
- Для работы с удалённой БД используется браузер.
- В HTML странице встраивается код, который отслеживает все действия пользователя, транслирует их в SQL запросы.

Однако алгоритмически сложные задачи рекомендуется реализовывать в архитектуре клиент-сервер с разработкой специализированного клиентского ПО.

### 4. Архитектура БД



Трёхуровневая архитектура БД (самая стабильная).



**Внешний уровень** — представляет данные для каждого пользователя в удобной для него форме. Он содержит внешние представления базы данных, которые интересны конкретному пользователю. Внешнее представление содержит только те сущности, атрибуты и связи, которые интересны пользователю.

**Концептуальный уровень** — это обобщающее представление БД. Этот уровень описывает то, какие данные хранятся в БД, а также связи, существующие между ними. Этот уровень содержит логическую структуру всей БД (с точки зрения администратора БД). На концептуальном уровне представлены следующие компоненты:

- Все сущности, их атрибуты и связи;
- Накладываемые на данные ограничения;
- Семантическая информация о данных (связанная со значением, смыслом);
- Информация о мерах обеспечения безопасности и поддержки целостности данных.

**Физическая БД** описывает, как данные хранятся на физических носителях. Он оптимизирует производительность и использование дискового пространства. На этом уровне описывается структура данных, организация файлов и детали хранения записей. На физическом уровне хранится следующая информация:

- Распределение дискового пространства для хранения данных и индексов;
- Описание подробностей сохранения записей;
- Сведения о размещении записей;
- Сведения о сжатии данных и выбранных методах их шифрования.

Между **внешним и концептуальным** уровнями существует логическая независимость, что означает, что изменения на концептуальном уровне не должны влиять на внешние представления.

Между **концептуальным и физическим** уровнями существует физическая независимость, что означает, что изменения на физическом уровне не должны влиять на концептуальное представление.

## 5. Процесс прохождения пользовательского запроса

**База метаданных** - база, где хранится вся информация об используемых структурах данных, логической организации данных, правах доступа пользователей и физическом расположении данных.

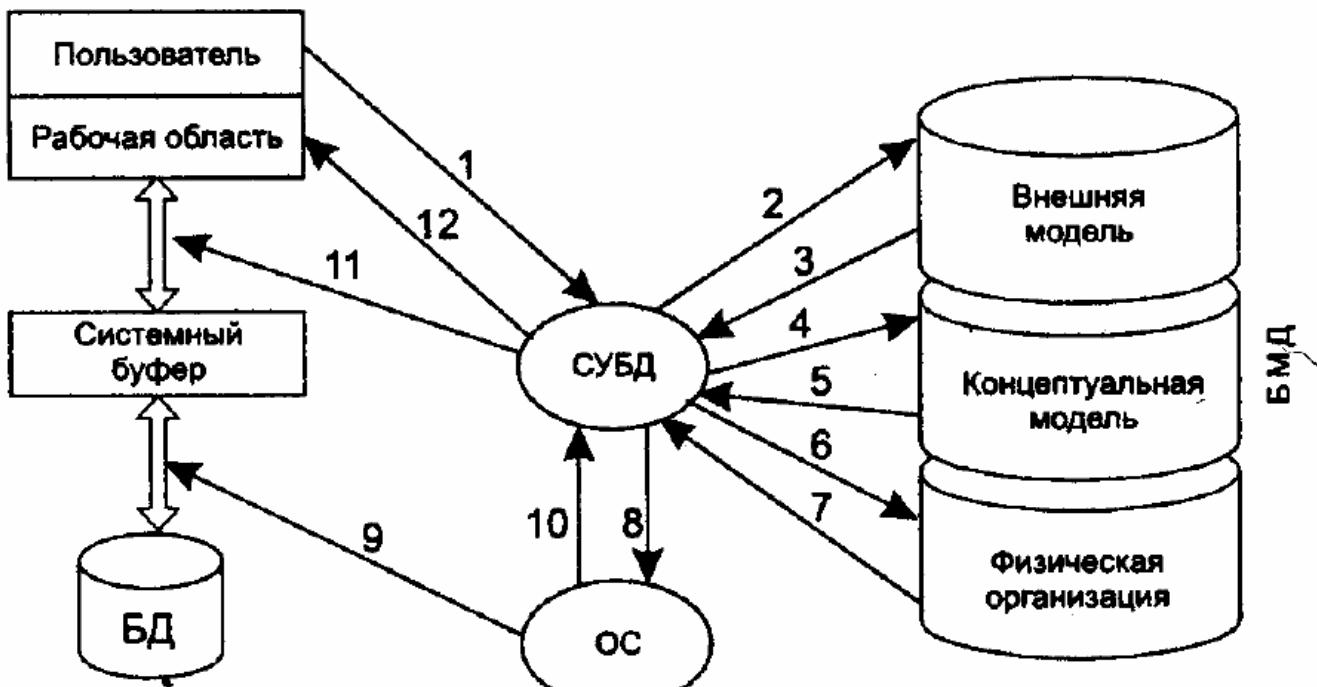


Рис. 1-2. Схема прохождения запроса к БД

Описание:

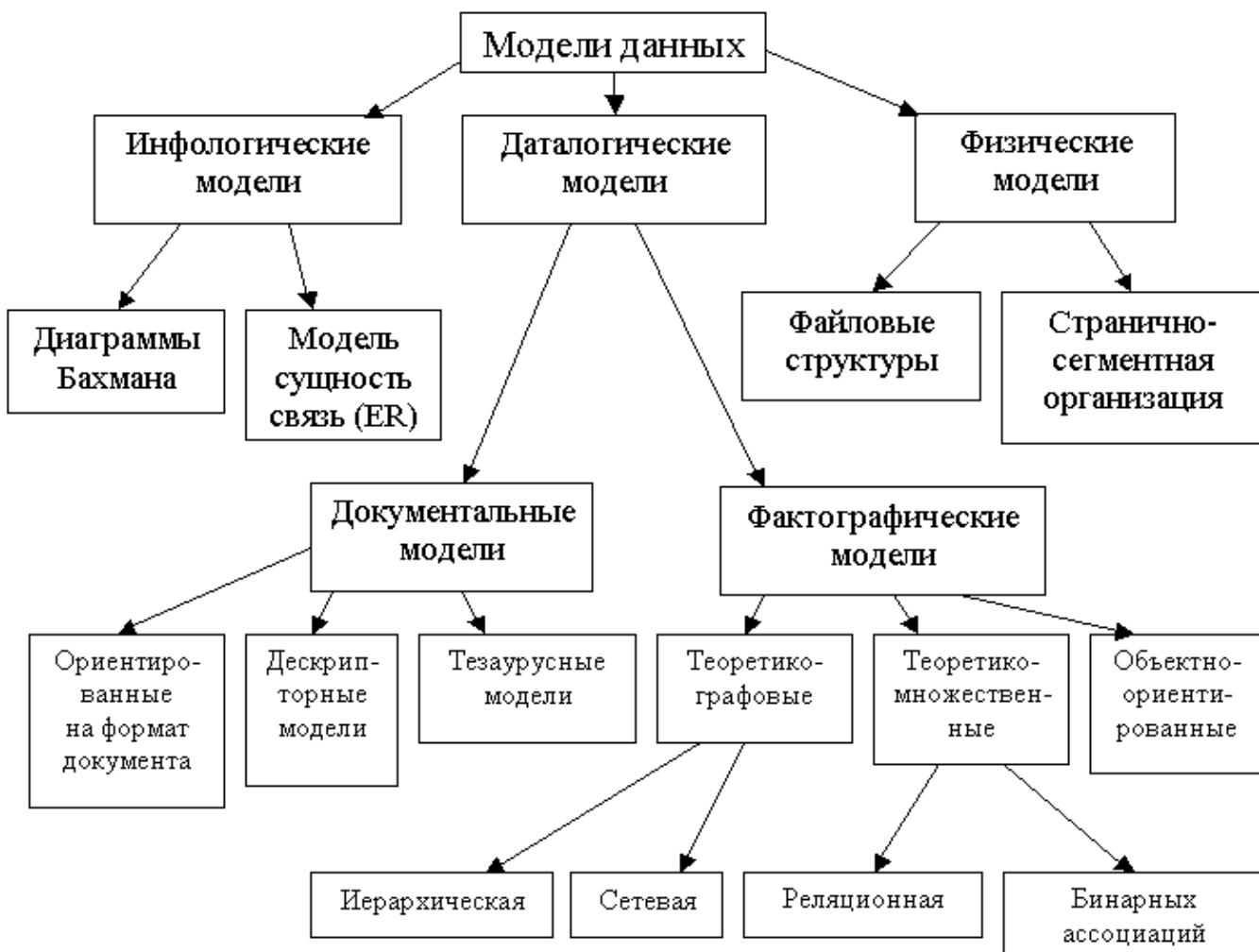
1. Пользователь посылает СУБД запрос на получение данных из БД.
2. Анализ прав пользователя и внешней модели данных, соответствующей данному пользователю, подтверждает или запрещает доступ данного пользователя к запрошенным данным.
3. В случае запрета на доступ к данным СУБД сообщает пользователю об этом (линия 12) и прекращает дальнейший процесс обработки данных, в противном случае СУБД определяет часть концептуальной модели, которая затрагивается запросом пользователя.
4. СУБД запрашивает информацию о части концептуальной модели.

5. СУБД получает информацию о запрошенной части концептуальной модели.
6. СУБД запрашивает информацию о местоположении данных на физическом уровне (файлы или физические адреса).
7. В СУБД возвращается информация о местоположении данных в терминах операционной системы.
8. СУБД вежливо просит операционную систему предоставить необходимые данные, используя средства операционной системы.
9. Операционная система осуществляет перекачку информации из устройств хранения и пересыпает ее в системный буфер.
10. Операционная система оповещает СУБД об окончании пересылки.
11. СУБД выбирает из доставленной информации, находящейся в системном буфере, только то, что нужно пользователю, и пересыпает эти данные в рабочую область пользователя.

## 6. Классификация моделей данных. Документальные модели

**Модель данных** – это некоторая абстракция, которая, будучи приложима к конкретным данным позволяет трактовать их как информацию, т.е. сведения, содержащие не только данные, но и взаимосвязи между ними.

Классификация моделей данных (в соответствии с трехуровневым представлением БД):



- **Инфологические модели** представляют информацию о предметной области независимо от конкретной базы данных. Они описывают объекты, их свойства и связи в понятной и

естественной форме на уровне абстракции информации. Инфологические модели помогают фиксировать и описывать структуру и отношения данных, не привязываясь к конкретной технической реализации или СУБД.

- **Физическая модель** данных (ФМД) – это модель данных, описанная с помощью средств конкретной СУБД. ФМД строится на базе даталогической путем добавления особенностей конкретной СУБД.
- **Даталогическая модель**, отражающая логические взаимосвязи между элементами данных безотносительно их содержания и физической организации. При этом даталогическая модель разрабатывается с учетом конкретной реализации СУБД, также с учетом специфики конкретной предметной области на основе ее инфологической модели.

## Документальные модели

Относятся к даталогическим моделям данных и работают со слабоструктурированной информацией, основанной на свободном формате документа или текста на его естественном языке.

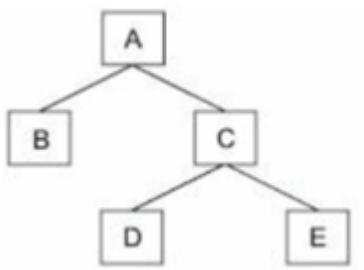
Три вида:

- **Ориентированные на формат документа:** Связаны прежде всего со стандартным общим языком разметки документов, который позволяет организовывать информацию, содержащуюся в документах, и представлять ее в определенном стандартном виде.
- **Дескрипторные модели:** Каждому документу соответствует дескриптор или описатель, который имеет жесткую структуру и описывает документ в соответствии с теми характеристиками, которые требуются для работы с документами.
- **Тезаурусные модели:** Основаны на принципе организации словарей, описывающих языковые выражения и их взаимосвязи. Под тезаурусом понимается иерархический словарь понятий и отношений между ними, что позволяет представлять исходный текст документа в виде системы этих понятий.

## 7. Теоретико-графовые модели данных. Иерархическая модель данных

**Теоретико-графовые модели данных.** Эти модели отражают совокупность объектов реального мира в виде графа взаимосвязанных информационных объектов.

### Иерархическая модель



Является наиболее простой из всех даталогических моделей. В реальном мире очень многие связи соответствуют иерархии, когда один объект выступает как родительский, а с ним может быть связано множество подчиненных объектов. Иерархия проста и естественна в отображении взаимосвязи между классами объектов.

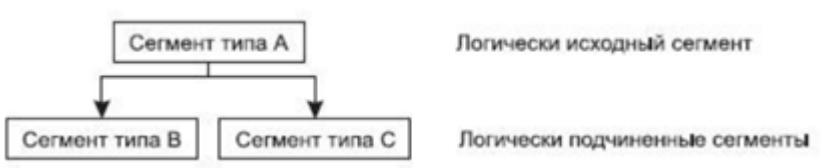
Эти модели представляются в виде графов взаимосвязанных информационных объектов.

## Основные информационные единицы иерархической модели

- **Поле данных** – это минимальная неделимая единица данных, доступная пользователю с помощью СУБД. Разработчик сам выбирает что будет выступать в качестве поля. Примеры: адрес.
- **Сегмент** – совокупность полей, который называют записью. В иерархической модели выделяют: тип сегмента и экземпляр сегмента (разница между ними тип переменной и сама переменная).
- **Тип сегмента** – поименованная совокупность типов элементов данных, в него входящих. Каждый тип сегмента в рамках иерархической модели образует некоторый набор однородных записей.
- **Экземпляр сегмента** – образуется из конкретных значений полей или элементов данных, в него входящих. Пример: Группа (№ группы, староста) 5831, Свистунов Ю. 5836, Лунгу С.

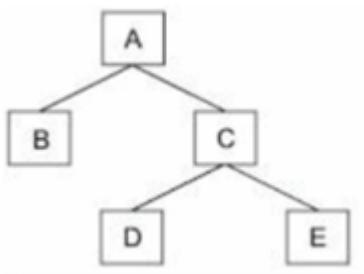
Для **возможности различия отдельных записей** в данном наборе каждый тип сегмента должен иметь ключ или набор ключевых атрибутов (полей, элементов данных) - набор элементов данных, однозначно определяющих экземпляр сегмента.

Схема иерархической БД представляет собой совокупность отдельных деревьев, каждое дерево в рамках модели называется физической базой данных.



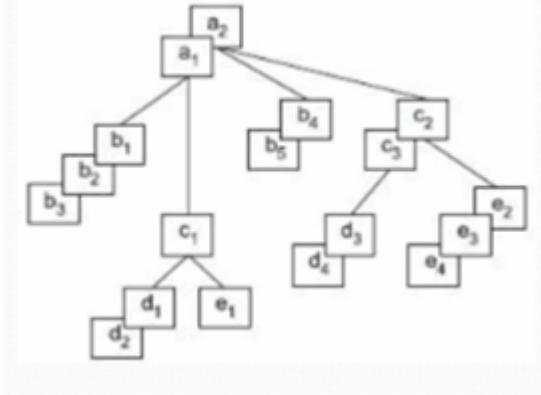
Каждая физическая БД удовлетворяет следующим **иерархическим ограничениям**:

- В каждом физическом БД существует один корневой сегмент
- Каждый логически исходный сегмент может быть связан с произвольным числом логически подчиненных сегментов
- Каждый логический сегмент может быть связан только с одним логически исходным (родительским) сегментом



Каждый тип сегмента может иметь множество соответствующих ему экземпляров. Между экземплярами сегментов также существуют **иерархические связи**.

Экземпляры-потомки одного типа, связанные с одним экземпляром, **называются близнецами**. Набор всех экземпляров сегментов, подчинённых одному экземпляру корневого сегмента, называют физической записью.

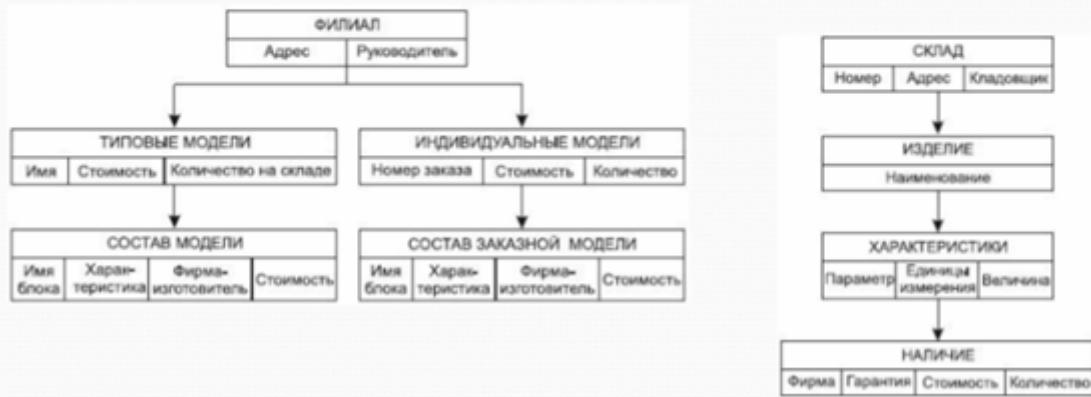


Пример:

### Пример иерархической БД

#### Физическая БД "Филиалы"

#### Физическая модель "Склады"



## 8. Теоретико-графовые модели данных. Сетевая модель

### Сетевая модель данных

- **Элемент данных** - минимальная информационная единица, доступная пользователю с использованием СУБД.
- **Агрегат данных** соответствует следующему уровню обобщения в модели. Имеет имя, и в системе допустимо обращение к агрегату по имени. В модели определены агрегаты двух типов:
  - *Вектор*. Пример вектора: Адрес (Город, улица, дом, корпус, квартира)
  - *Повторяющаяся группа*. Это совокупность векторов 12 штук (для примера).

Пример: Зарплата (Месяц, сумма (оклад, бонус, премия, штраф))

- **Запись** - совокупность агрегатов или элементов данных, моделирующая некоторый класс объектов реального мира. Для записи вводятся понятия типа записи и экземпляра записи (аналогия с сегментом).
- **Набор данных** - двухуровневый граф, связывающий отношением "один-ко-многим" два типа записи. Набор фактически отражает иерархическую связь между двумя типами записей.

Родительский тип связи в данном наборе называется **владельцем набора**, а дочерний тип записи - **членом набора**.

Для двух типов записи может быть задано любое количество наборов, которые их связывают. Фактически наличие подобных возможностей позволяет промоделировать отношение "**многие-ко-многим**" между двумя объектами реального мира, что выгодно отличает сетевую модель от иерархической.

Существенным ограничением набора является то, что один и тот же тип записи **не может быть одновременно и владельцем, и членом набора**.



Пример сетевой БД

Схема БД "Торговая фирма"

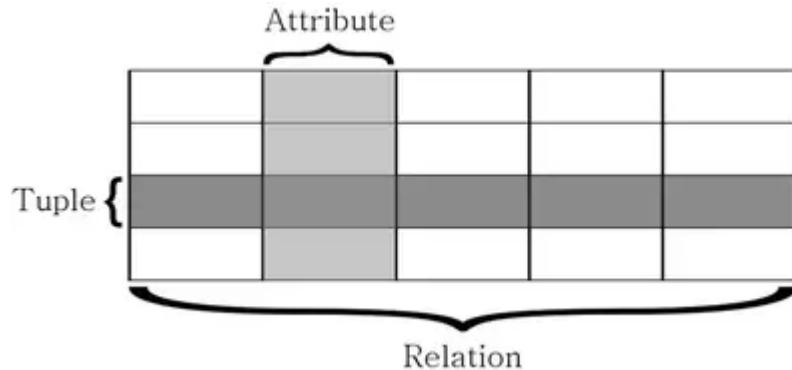


## Реляционная модель. Основные определения

**Теоретико-множественные** модели основаны на теории множеств, опираются на свойства множеств и операции, которые производятся над множествами. Эти модели наиболее перспективны для создания БД.

### Реляционная модель данных

Основной структурой данных в модели является отношение.



**Опр. N-арным отношением R** называют подмножество декартова произведения  $D_1 \times D_2 \times \dots \times D_n$  множеств  $D_1, D_2, \dots, D_n$  ( $n > 1$ ), необязательно различных. Исходные множества  $D_1, D_2, \dots, D_n$  называют в модели доменами.

$$R \subseteq D_1 \times D_2 \times \dots \times D_n$$

Полное декартово произведение - набор

всевозможных сочетаний из  $n$  элементов каждое, где каждый элемент берётся из своего домена.

**Пример** - Декартовая прямоугольная система координат.

Отношение R моделирует реальную ситуацию, и оно может содержать, допустим, только 5 строк, которые соответствуют результатам сессии (Крылов экзамен по Базам данных ещё не сдавал):

Отношение может быть представлено в виде таблицы, столбцы которой соответствуют вхождениям доменов в отношение, а строки - наборам из  $n$  значений, взятых из исходных доменов, которые расположены в строго определённом порядке в соответствии с заголовком. Такие наборы из  $n$  значений часто называют  $n$ -ками.

R		
Фамилия	Дисциплина	Оценка
Иванов	Теория автоматов	4
Иванов	Базы данных	3
Крылов	Теория автоматов	5
Степанов	Теория автоматов	5
Степанов	Базы данных	4

Таблица обладает рядом специфических свойств:

- В таблице нет двух одинаковых строк
- Таблица имеет столбцы, соответствующие атрибутам отношения
- Каждый атрибут в отношении имеет уникальное имя
- Порядок строк в таблице произвольный

**Атрибут** - вхождение домена в отношение.

**Кортеж** - строки отношения.

**Степень (ранг) отношения** - количество атрибутов в отношении.

В отношении **не может быть двух одинаковых кортежей**. Два отношения, отличающиеся только порядком строк или порядком столбцов, будут интерпретироваться в рамках реляционной модели как одинаковые.

Реляционная модель представляет **базу данных в виде множества взаимосвязанных отношений**.

В этой модели поддерживаются иерархические связи между отношениями. В каждой связи **одно отношение может выступать как основное, а другое отношение выступает в роли подчинённого**.

Для поддержки этих связей оба отношения должны содержать наборы атрибутов, по которым они связаны. В основном отношении это первичный ключ отношения (**PRIMARY KEY**), который однозначно определяет кортеж основного отношения. В подчинённом отношении для моделирования связей должен присутствовать набор атрибутов, соответствующий первичному ключу основного отношения. Данный набор атрибутов в подчинённом отношении принято называть внешним ключом (**FOREIGN KEY**).



## 10. Реляционная алгебра. Основные операции: объединение, вычитание, декартово произведение

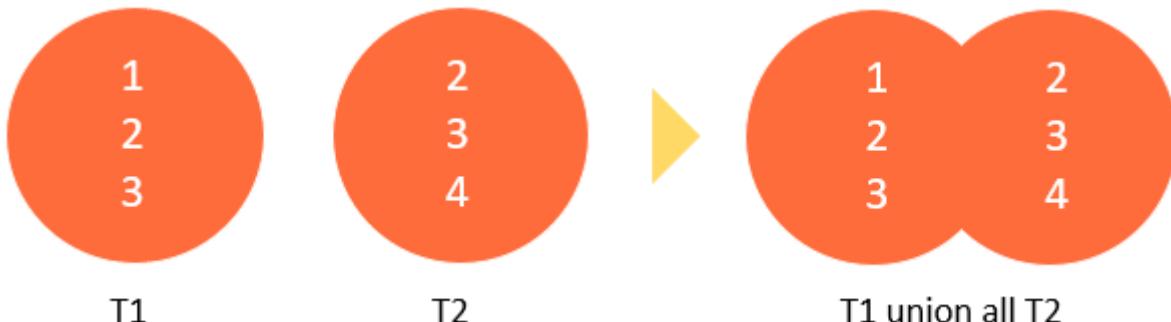
Реляционная алгебра и язык запросов реляционной алгебры

В реляционной алгебре определяются **5 основных операций** над отношениями, на основе которых можно сконструировать язык запросов (язык манипулирования данными - ЯМД) реляционной алгебры. Результатом операции также является отношение.

**Запрос** - операция над отношениями, результатом которой также является отношение.

### 1. Объединение

**Объединение (Union)** отношений R и S даёт множество кортежей, которые принадлежат R или S, или им обоим.  $REZ = R \cup S$

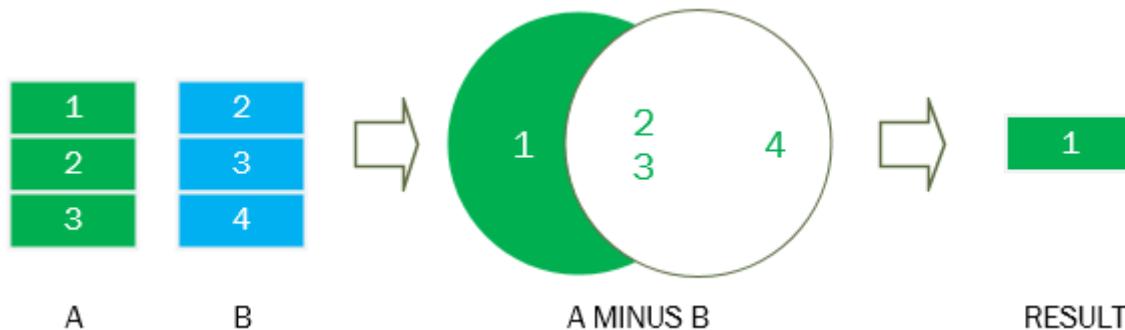


Имена атрибутов отношений-операндов могут быть различными. Соответствующие значения компонент кортежей должны принадлежать одному домену.

$R = \underline{PN}, \underline{PIM}, \underline{ST}, \underline{GOROD}$	$S = \underline{PN}, \underline{PIM}, \underline{ST}, \underline{GOROD}$
П1 Иванов 20 Москва	П1 Иванов 20 Москва
П2 Петров 40 Самара	П3 Кротов 10 Москва

## 2. Вычитание

**Вычитание (MINUS)** - в результирующее отношение включаются кортежи из R, не принадлежащие S, и  $K_{rez} = K_r = K_s$ .



Операция вычитания используется для удаления ненужных кортежей отношения.

## 3. Декартово произведение

**Декартово произведение (TIMES)** - результирующее отношение состоит из кортежей, Первые  $K_r$  компонент которых - кортежи из R, а последние  $K_s$  компонент - кортежи из S и  $K_{rez} = K_r + K_s$  ("склеивание" кортежей операндов)

Декартово произведение			
Отношение A		Отношение B	
Имя	Факультет	Фамилия	Стипендия
Вася	Математический	Иванов	Есть
Коля	Физический	Петров	Нет

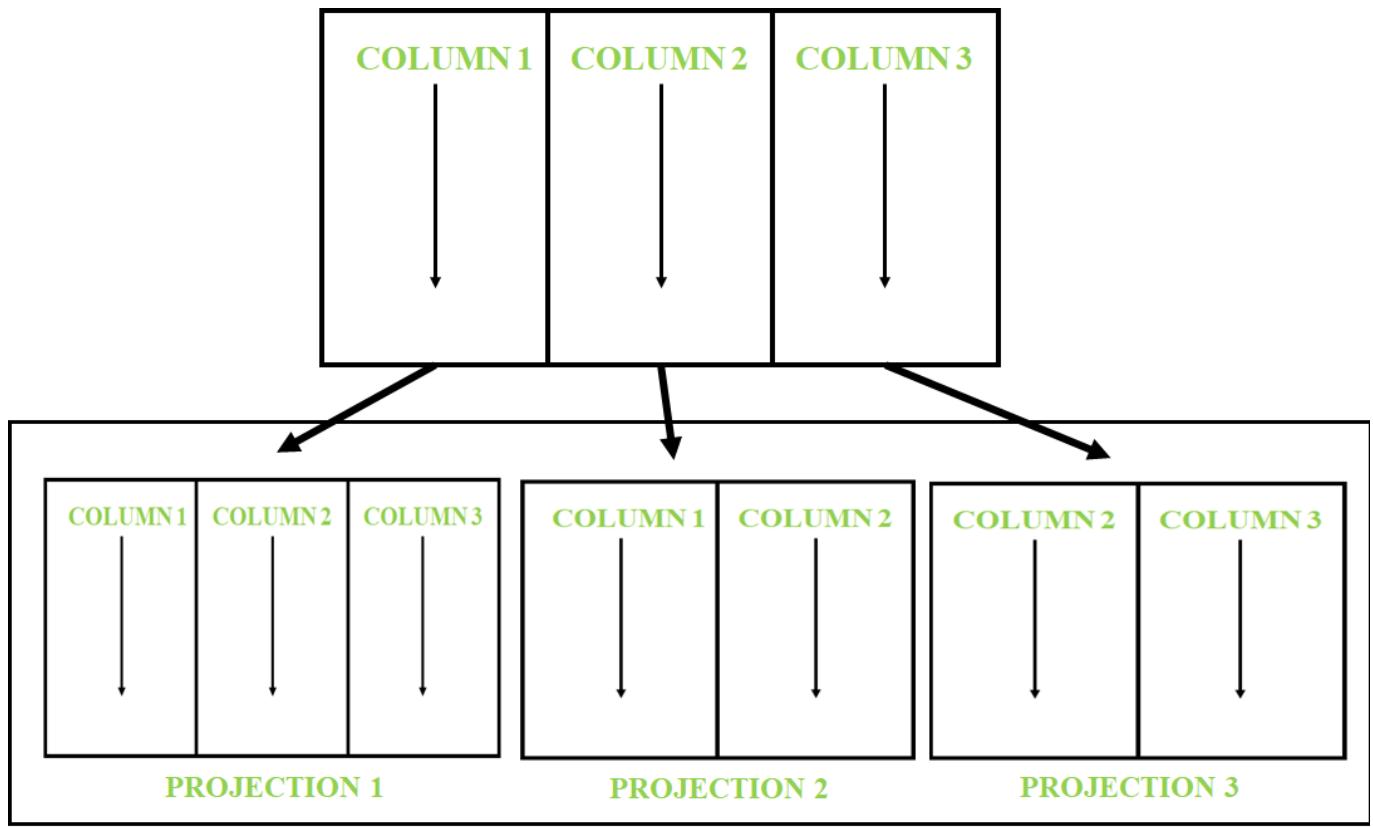
Отношение A x B			
Имя	Факультет	Фамилия	Стипендия
Вася	Математический	Иванов	Есть
Коля	Физический	Петров	Нет
Коля	Физический	Иванов	Есть
Вася	Математический	Петров	Нет

## 11. Реляционная алгебра. Основные операции: проекция, селекция

Продолжение предыдущего билета.

## 4. Проекция (Project)

**Проекция (Project)** даёт настроить вертикальное подмножество отношения. Применяется к одному конкретному отношению.



**Пример:**

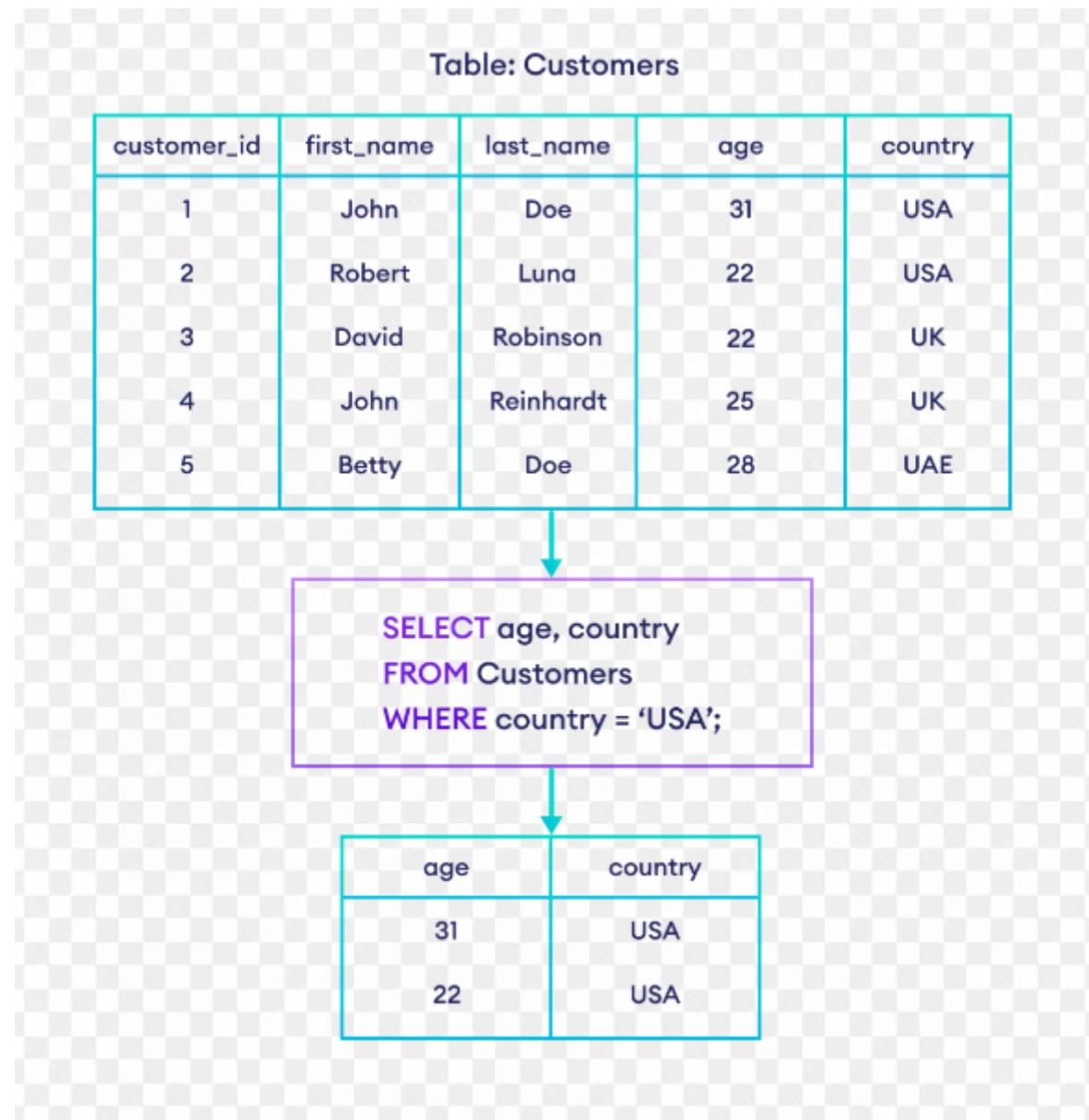
Пусть  $R = \underline{PN}, \underline{PIM}, ST, GOROD$   
 П1 Иванов 20 Москва  
 П2 Петров 40 Самара  
 П3 Кротов 10 Москва

Ответ на запрос: “Найти все города, в которых размещены поставщики”  
 ( $GOROD? \rightarrow *$ ) может быть получен как результат выполнения операции проекции R на атрибут GOROD:

$REZ = \pi_{GOROD} (R) = \underline{GOROD}$   
 Москва  
 Самара

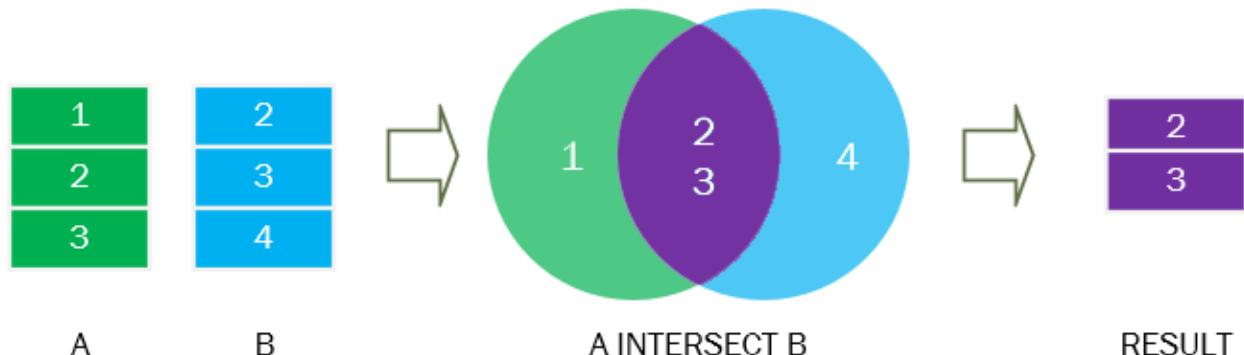
## 5. Выбор(SELECT)

**Селекция или выбор (SELECT)** даёт возможность построить горизонтальное подмножество отношения. Отбор кортежей может производиться с использованием операций. (арифметических, логических и т.д.)



## 12. Реляционная алгебра. Дополнительные операции: пересечение, деление, соединение

1. **Пересечение (INTERSECTION)** ( $REZ = R \cap S$ ): отношений  $R$  и  $S$  даёт множество кортежей, которые принадлежат как  $R$ , так и  $S$ . Эта операция осуществляется при  $Kr = Kr = Ks$ .



2. **Деление (DIVISION)** ( $REZ = R \div S$ ): Деление выполнимо, если делитель  $S \neq \emptyset$ , и атрибуты делителя образуют некоторое подмножество атрибутов делимого  $R$ , и  $Kr > Ks$ . Аргументы частного,

делимого и делителя связаны отношением Krez = Kr - Ks.

**R**

ColA	ColB
F	1
F	2
F	3
E	1
E	3
S	1
S	2

**S**

ColB
1
2

$$R \div S =$$

ColA
F
S

**Пример:**

$$R = \begin{array}{cccc} A & B & C & D \end{array} \quad \text{и} \quad S = \begin{array}{cc} C & D \end{array}$$

a	b	c	d
a	b	e	f
b	c	e	f
e	d	c	d
e	d	e	f
a	b	d	e
m	n	c	d

$\cdot \begin{array}{l} c \\ d \end{array} \} u1$

$\begin{array}{l} e \\ f \end{array} \} u2$

$$REZ = R \div S = \begin{array}{cc} A & B \\ a & b \\ e & d \\ \leftarrow t \rightarrow \end{array}$$

Повторяющиеся кортежи из результата исключаются.

$$\text{Для того же } R \text{ и } S = \begin{array}{cc} C & D \\ c & d \end{array} \quad REZ = R \div S = \begin{array}{cc} A & B \\ a & b \\ c & d \\ m & n \\ \leftarrow t \rightarrow \leftarrow u \rightarrow \end{array}$$

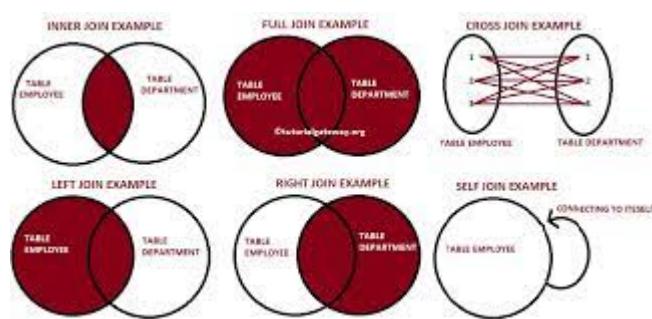
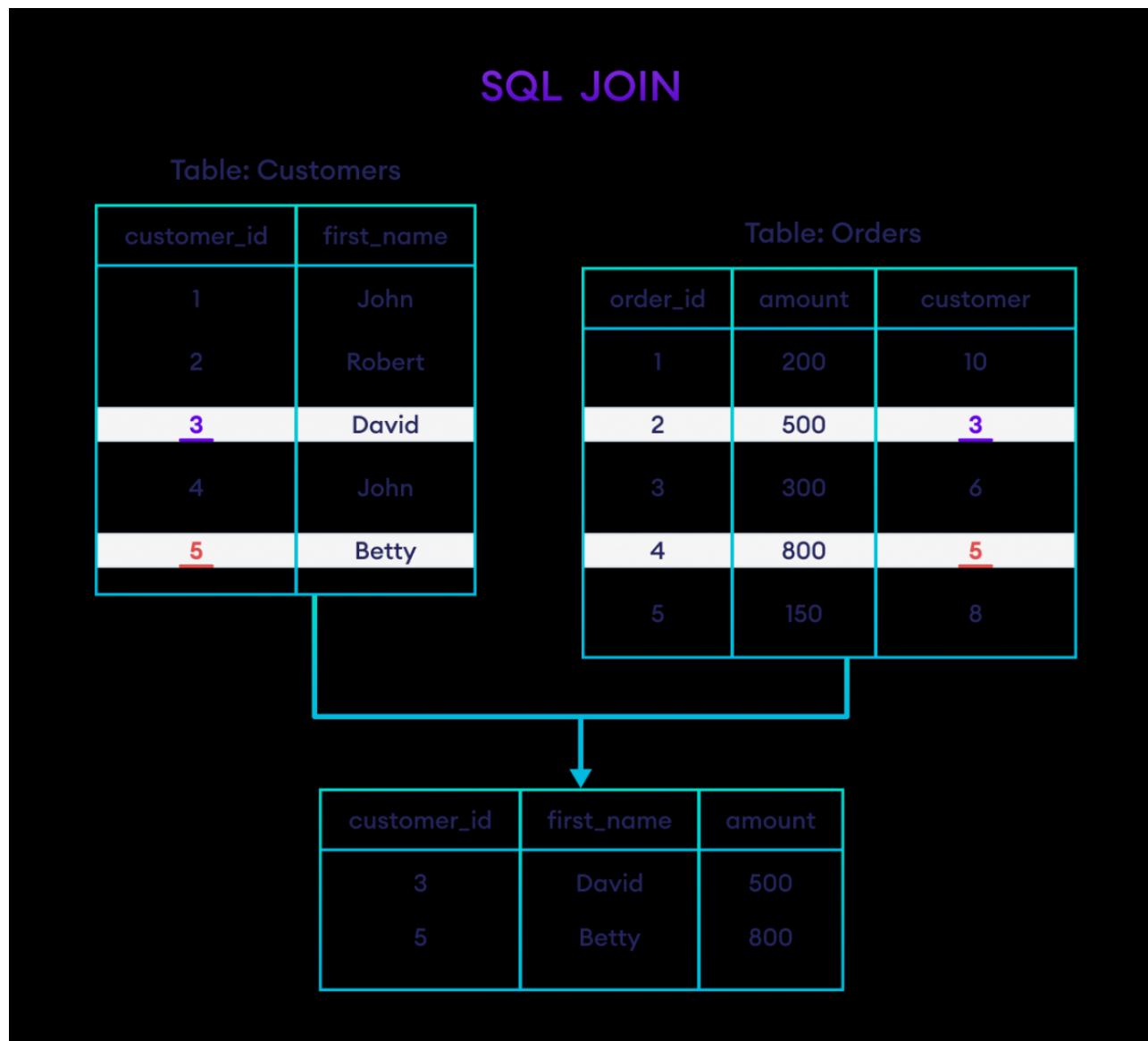
3. **Соединение (JOIN)** ( $REZ = R \bowtie S$ ): отношения осуществляется с использованием операций отношения ( $<\leq = \# \geq >$ ).

**Пример:**

$$R = \begin{array}{ccc} A & B & C \\ 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{array}$$

$$S = \begin{array}{c} D \quad E \\ 3 \quad 1 \\ 6 \quad 2 \end{array}$$

$$REZ = R \bowtie S = \begin{array}{ccccc} A & B & C & D & E \\ 1 & 2 & 3 & 3 & 1 \\ 1 & 2 & 3 & 6 & 2 \\ 4 & 5 & 6 & 6 & 2 \end{array}$$



4. **Естественное соединение (JOIN)** - тоже, что и 3.

**Шаг 1.** Вычисляется декартово произведение  $R \times S$ . Например, для

$R = \begin{array}{lll} A & B & C \end{array}$	$S = \begin{array}{lll} B & C & D \end{array}$	$R \times S = \begin{array}{llllll} A & R.B & R.C & S.B & S.C & D \end{array}$
a b c	b c d	a b c b c d (+)
d b c	b c e	a b c b c e (+)
b b f		d b c b c d (+)
c a d		d b c b c e (+)
		b b f b c d
		b b f b c e
		c a d b c d
		c a d b c e

**Шаг 2.** Из декартова произведения отбираем кортежи, у которых равны значения общих атрибутов. В примере:

$$R.B = S.B \text{ и } R.C = S.C$$

Такие кортежи отмечены (+).

**Шаг 3.** Перенесем отмеченные кортежи в результатирующее отношение REZ, исключая повторяющиеся кортежи и используя общие атрибуты один раз. Тогда

$$REZ = R \bowtie S = \begin{array}{lll} A & B & C & D \end{array}$$

a	b	c	d
a	b	c	e
d	b	c	d
d	b	c	e

## 13. Функциональные зависимости атрибутов

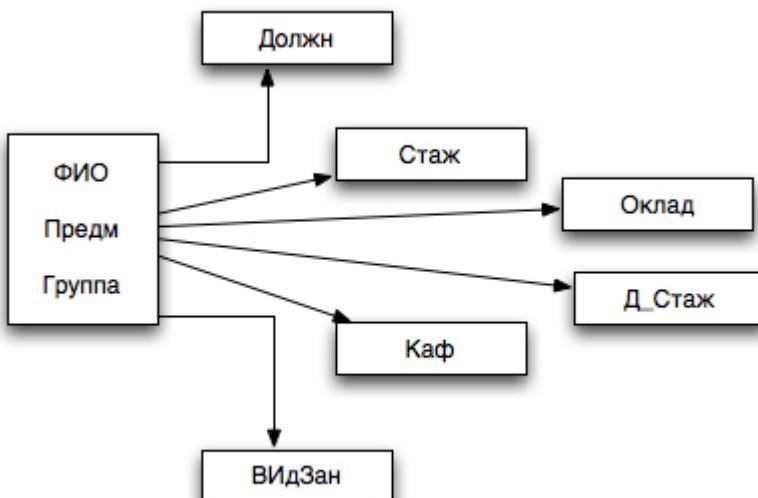
**Нормализация** – это процесс, направленный на снижение избыточности информации в реляционных базах данных.

В основу нормализации положена концепция *функциональных зависимостей*.

Нормализация отношений модели предметной области позволяет создать логическую модель реляционной БД.

Функциональная зависимость между атрибутами

**Функциональной зависимостью** набора атрибутов  $B$  отношения  $R$  от набора атрибутов  $A$  того же отношения, обозначаемый как  $R.A \rightarrow R.B$  или  $A \rightarrow B$ , называется такое отношение проекций  $R[A]$  и  $R[B]$ , при котором в каждый момент времени любому элементу проекции  $R[A]$  соответствует только один элемент проекции  $R[B]$ , входящий вместе с ним в какой-либо кортеж отношения  $R$ .



ГРАФИК_ПОЛЕТОВ (Пилот, Рейс, Дата_вылета, Время_вылета)				
Ивано в	100	8.07	10:20	Известно, что:
Ивано в	102	9.07	13:30	1) каждому рейсу соответствует определенное время вылета;
Исаев в	90	7.07	6:00	2) для каждого пилота, даты и времени вылета возможен только один рейс;
Исаев в	100	11.07	10:20	3) на определенный день и рейс назначается определенный пилот.
Исаев в	103	10.07	19:30	
Петро в	100	12.07	10:20	
Петро в	102	11.07	13:30	
Фроло в	90	8.07	6:00	

Продолжение примера				
ГРАФИК_ПОЛЕТОВ (Пилот, Рейс, Дата_вылета, Время_вылета)				
➤ "Время_вылета" функционально зависит от "Рейс":				
				$\text{Рейс} \rightarrow \text{Время\_вылета};$
➤ "Рейс" функционально зависит от ("Пилот", "Дата_вылета", "Время_вылета"):				
				$(\text{Пилот}, \text{Дата\_вылета}, \text{Время\_вылета}) \rightarrow \text{Рейс};$
➤ "Пилот" функционально зависит от ("Рейс", "Дата_вылета"):				
				$(\text{Рейс}, \text{Дата\_вылета}) \rightarrow \text{Пилот}.$

Т-зависимости				
Пусть $X, Y, Z$ - атрибуты отношения $R$ .				
• Если имеются ФЗ $F: X \rightarrow Y$ и $F: Y \rightarrow Z$ ,				
• но отсутствуют ФЗ $F: Z \rightarrow Y$ и $F: Y \rightarrow X$ ,				
то $Z$ <u>транзитивно</u> зависит от $X$ .				
Такие ФЗ называются транзитивными ( <b>Т-зависимостями</b> ).				
Т- зависимость возникает в случае, если неключевой атрибут зависит от другого неключевого атрибута.				

Грубо говоря, нормализация это выявление функциональной зависимости.

## Три Аксиомы Армстронга

Применяя эти аксиомы, из заданного множества функциональных зависимостей можно вывести все возможные функциональные зависимости.

- **Рефлексивность** Если В является подмножеством А, то можно сказать, что В функционально зависит от А ( $A \rightarrow B$ )
- **Дополнение** Если В функционально зависит от А (). Если  $A \rightarrow B$  то  $A.C \rightarrow B.C$
- **Транзитивность** Если  $A \rightarrow C \rightarrow B$ , то  $A \rightarrow B$

**Возможным ключом** отношения называется набор атрибутов отношения, который полностью, и **однозначно определяет значения всех остальных** атрибутов отношения (то есть возможный ключ - набор атрибутов, однозначно определяющий кортеж отношения, и при этом при удалении любого атрибута из этого набора, его свойство однозначной идентификации кортежа теряется).

В общем случае в отношении, может быть, несколько возможных ключей. Среди возможных ключей отношения обычно выбирают один, который **считается главным** и который называют **первичным ключом отношения**

**Неключевым атрибутом** называется любой атрибут отношения, не входящий в состав **ни одного возможного ключа отношения**.

**Взаимно-независимые атрибуты** - такие атрибуты, которые не зависят функционально один от другого.

## 14. Этапы проектирования реляционной БД

Этот билет коррелирует с 13. Функциональные зависимости атрибутов, 15. Системный анализ предметной области. Пример. и Нормальными формами.

Процесс проектирования базы данных (БД) включает переход от **неформального описания структуры предметной области к формализованному описанию объектов** предметной области в терминах выбранной модели данных. Проектирование реляционных БД на основе принципов нормализации

**Инфологическая модель** ориентирована на пользователя и определяет информационные объекты, их атрибуты и связи между ними.

**Логическое проектирование** включает создание схемы БД на основе конкретной модели данных, например, реляционной модели.

Важным шагом является выбор **эффективного размещения БД** на внешних носителях для оптимальной работы приложения.

Результирующий проект реляционной БД представляет собой набор взаимосвязанных отношений с определенными атрибутами, первичными ключами и дополнительными свойствами, связанными с поддержкой целостности данных.

## 15. Системный анализ предметной области. Пример

С точки зрения проектирования БД в рамках системного анализа необходимо провести подробное словесное описание объектов предметной области и **реальных связей**, которые присутствуют между описываемыми объектами.

Системный анализ должен заканчиваться:

- подробным описанием информации об объектах предметной области, которая требуется для решения конкретных задач, и которая должна храниться в БД
- формулировкой конкретных задач, которые будут решаться с использованием данной БД с кратким описанием алгоритмов их решения
- описанием выходных документов, которые должны генерироваться в системе
- описанием входных документов, которые служат основанием для заполнения данными БД

Пример

TL:DR

Пусть требуется разработать информационную систему для автоматизации учёта получения и выдачи книг в библиотеке. Система должна предусматривать режимы ведения системного каталога, отражающего перечень областей знаний, по которым имеются книги в библиотеке. Внутри библиотеки области знаний в систематическом каталоге могут иметь уникальный внутренний номер и полное наименование.

Каждая книга может содержать сведения из нескольких областей знаний. Каждая книга в библиотеке может присутствовать в нескольких экземплярах.

Каждая книга, хранящаяся в библиотеке, характеризуется следующими параметрами:

- Уникальный шифр (серийный номер)
- Название
- Фамилии авторов (могут отсутствовать)
- Место издания (город)
- Издательство
- Год издания
- Количество страниц
- Стоимость книги
- Количество экземпляров книги в библиотеке

Книги могут иметь одинаковые названия, но они различаются по своему уникальному шифру (ISBN - International Standard Book Number). В библиотеке ведётся картотека читателей.

На каждого читателя в картотеку заносятся следующие сведения:

- ФИО
- Домашний адрес
- Телефон (будем считать, что у нас 2 телефона - рабочий и домашний)
- Дата рождения

Каждому читателю присваивается уникальный номер читательского билета. Каждая книга в библиотеке может присутствовать в нескольких экземплярах. Каждый экземпляр имеет следующие характеристики:

- Уникальный инвентарный номер

- Шифр книги, который совпадает с уникальным шифром из описания книг
- Место размещения в библиотеке

В случае выдачи экземпляра книги читателю в библиотеке хранится специальный вкладыш, в котором должны быть записаны следующие сведения:

- Номер билета читателя, который взял книгу
- Дата выдачи книги
- Дата возврата

Предусмотреть следующие ограничения на информацию в системе:

1. Книга может не иметь ни одного автора
2. В библиотеке должны быть записаны читатели не моложе 17 лет
3. В библиотеке присутствуют книги, изданные начиная с 1960 по текущий год
4. Каждый читатель может держать на руках не более 5 книг
5. Каждый читатель при регистрации в библиотеке должен дать телефон для связи: он может быть рабочим или домашним
6. Каждая область знаний может содержать ссылки на множество книг, но каждая книга может относиться к различным областям знаний

С данной системой должны работать следующие группы пользователей:

- Библиотекари
- Читатели
- Администрация библиотеки

Читатель должен иметь возможность решать следующие задачи:

1. Просматривать системный каталог
2. По выбранной области знаний получить полный перечень книг, которые числятся в библиотеке
3. Для выбранной книги получить инвентарный номер свободного экземпляра книги или сообщение о том, что свободных экземпляров книги нет
4. Для выбранного автора получить список книг, которые числятся в библиотеке

Библиотекарь должен иметь возможность решать следующие задачи:

1. Принимать новые книги и регистрировать их в библиотеке.
2. Относить книги к одной или нескольким областям знаний.
3. Проводить назначение новых инвентарных номеров вновь принятым книгам, и запоминать место размещения каждого экземпляра.
4. Проводить списание старых и не пользующихся спросом книг. Списывать можно только книги, ни один экземпляр которых не находится у читателей. Списание проводится по специальному акту списания, который утверждается администрацией библиотеки.
5. Вести учёт выданных книг читателям, при этом предполагается два режима работы: Выдача книг читателю и приём от него возвращаемых им книг обратно в библиотеку.
6. Проводить списание утерянных читателем книг по специальному акту списания, подписанному администрацией библиотеки.
7. Проводить закрытие абонемента читателя, если читатель хочет выписаться из библиотеки и не является её должником.

Администрация библиотеки должна иметь возможность получать сведения о:

- должниках - читателях библиотеки
- сведения о книгах, которые не являются популярными
- Сведения о стоимости конкретной книги, для того чтобы установить возможность возмещения стоимости утерянной книги

## 16. Инфологическое проектирование. Модель «сущность – связь»

В настоящий момент **ER-модель** ("сущность-связь") стала фактическим стандартом при инфологическом моделировании БД. Большинство современных CASE-средств содержат инструментальные средства для описания данных в формализме этой модели. Кроме того, разработаны методы автоматического преобразования **проекта БД из ER-модели в реляционную**.

В основе ER-модели лежат следующие базовые понятия:

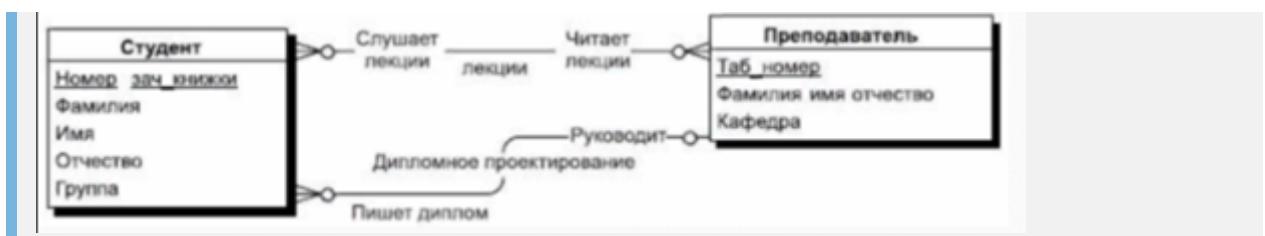
- **Сущность**, с помощью которой моделируется класс однотипных объектов. Сущность имеет имя, уникальное в пределах моделируемой системы. Предполагается, что в системе существует множество экземпляров данной сущности.
- Объект, которому соответствует понятие сущности, имеет свой набор **атрибутов** - характеристик, определяющих свойства данного представителя класса. При этом набор атрибутов должен быть таким, чтобы можно было различать конкретные экземпляры сущности.

Набор атрибутов, **однозначно идентифицирующий** конкретный экземпляр сущности, называют **ключевым**.

- Между сущностями могут быть установлены **связи** - бинарные ассоциации, показывающие, каким образом сущности соотносятся или взаимодействуют между собой. Связь может существовать между двумя разными сущностями или между сущностью и ею самой (рекурсивная связь). Она показывает, как связаны экземпляры сущностей между собой.

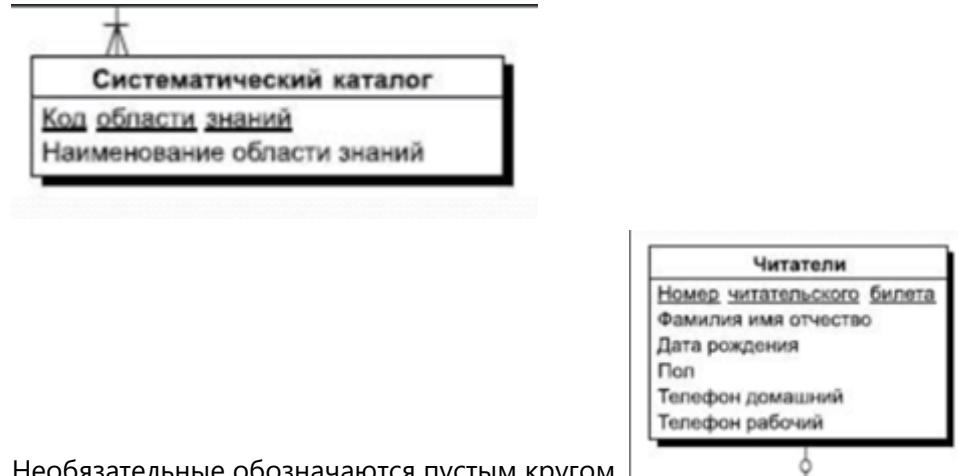
Связи делятся на три типа по множественности:

- **один-к-одному (1:1)** означает, что экземпляр одной сущности связан только с одним экземпляром другой сущности.
- **один-ко-многим (1: M)** означает, что один экземпляр сущности, расположенный слева по связи, может быть связан с несколькими экземплярами сущности, расположенными справа по связи
- **многие-ко-многим (M: M)** означает, что один экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, и наоборот, один экземпляр второй сущности может быть связан с несколькими экземплярами первой сущности.



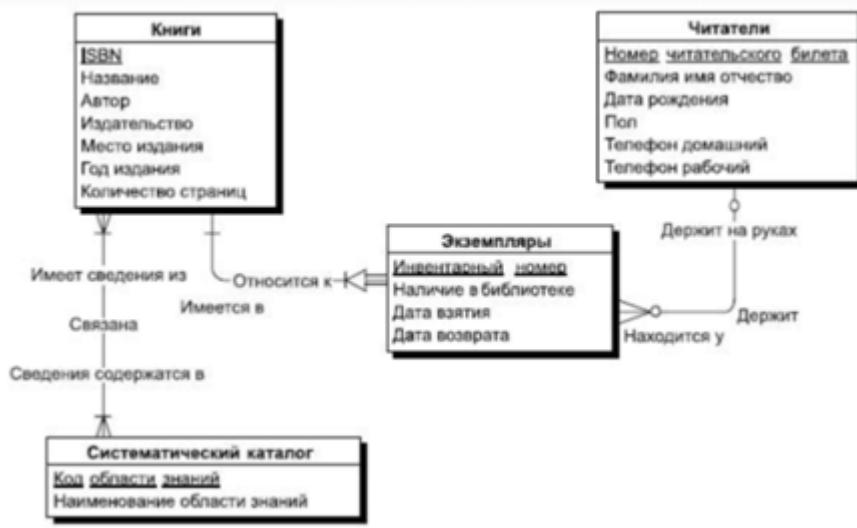
Связи бывают **обязательными и необязательными**:

- Обязательные обозначаются перпендикулярной чертой.



- Необязательные обозначаются пустым кругом.

Пример ER модели:



ER-модель допускает **принцип категоризации сущности**: сущность может быть представлена в виде нескольких своих подтипов-сущностей, каждая из которых может иметь общие атрибуты и отношения, которые определяются однажды, на верхнем уровне, и наследуются на нижнем уровне. При этом все подтипы одной сущности рассматривать как взаимоисключающие.

Сущность, на основе которой строятся подтипы, называется **суперсущность (супертип)**. Любой экземпляр супертипа должен относиться к конкретному типу.

Далее пойдут \*\*нормальные формы\*\*, знать очень важно!

## 17. Даталогическое проектирование. **Первая нормальная форма**

**Нормализация** – это метод проектирования базы данных, который позволяет привести базу данных к минимальной избыточности.

**Избыточность данных** создает предпосылки для появления различных аномалий, снижает производительность, и делает управление данными не гибким и не очень удобным.

### Ненормализованная форма (НФ)

Ненормализованная форма (НФ) – это форма организации данных в реляционной базе данных, в которой таблицы содержат повторяющиеся группы данных и/или непростые значения в столбцах.

Пример таблицы в ненормализованной форме:

CustomerID	CustomerName	OrderID	OrderDate	ProductID	ProductName	Quantity
1	John	101	2021-01-01	1	Apple	3
1	John	101	2021-01-01	2	Banana	2
2	Lisa	102	2021-01-02	1	Apple	1
2	Lisa	102	2021-01-02	3	Orange	4

В этом примере таблица содержит повторяющиеся значения для каждого заказа. Для каждого заказа существует дублирование информации о клиенте. Также каждая строка содержит информацию о конкретном продукте, включая его название, но название продукта также повторяется для каждого заказа, где этот продукт присутствует.

Это неэффективное использование хранения данных, так как дублирование может привести к проблемам, таким как сложности обновления информации и непоследовательность данных.

Основная цель нормализации данных состоит в том, чтобы избавиться от повторяющихся данных и устраниить аномалии, возникающие при обновлении и вставке данных.

### Первая нормальная форма (1НФ)

**Первая нормальная форма (1НФ)** – это первый уровень нормализации данных в реляционной базе данных. В 1НФ все значения в таблице являются атомарными, то есть они не могут быть разделены на более мелкие части.

Критерии для достижения 1НФ:

1. Каждая ячейка в таблице должна содержать только одно значение.
2. Уникальный идентификатор должен быть определен для каждой строки (обычно это первичный ключ).

Пример таблицы в ненормализованной форме:

CustomerID	CustomerName	OrderID	OrderDate	ProductID	ProductName	Quantity
1	John	101	2021-01-01	1, 2	Apple, Banana	3, 2
2	Lisa	102	2021-01-02	1, 3	Apple, Orange	1, 4

Пример таблицы в первой нормальной форме:

**Таблица Customers:**

CustomerID	CustomerName
1	John
2	Lisa

**Таблица Orders:**

OrderID	CustomerID	OrderDate
101	1	2021-01-01
102	2	2021-01-02

**Таблица OrderItems:**

OrderID	ProductID	ProductName	Quantity
101	1	Apple	3
101	2	Banana	2
102	1	Apple	1
102	3	Orange	4

В этом примере изначальная таблица была разделена на три таблицы в первой нормальной форме. Таблица Customers содержит информацию о клиентах, таблица Orders содержит информацию о заказах, а таблица OrderItems содержит информацию о продуктах, связанных с каждым заказом.

В результате получается структура данных, где значения столбцов являются атомарными, и каждая таблица имеет уникальный идентификатор (первичный ключ), что соответствует критериям 1НФ.

## 18. Даталогическое проектирование. Вторая нормальная форма

Вторая нормальная форма (2НФ) - это второй уровень нормализации данных в реляционной базе данных. В 2НФ таблица должна быть в 1НФ, и все неключевые атрибуты должны полностью зависеть от первичного ключа.

Критерии для достижения 2НФ:

1. Таблица должна быть в 1НФ.
2. Все неключевые атрибуты должны полностью зависеть от каждой части составного первичного ключа.

Пример таблицы в первой нормальной форме:

**Таблица Orders:**

OrderID	CustomerID	OrderDate	ProductID	ProductName	Quantity
101	1	2021-01-01	1	Apple	3

OrderID	CustomerID	OrderDate	ProductID	ProductName	Quantity
101	1	2021-01-01	2	Banana	2
102	2	2021-01-02	1	Apple	1
102	2	2021-01-02	3	Orange	4

Пример таблицы во второй нормальной форме:

#### Таблица Customers:

CustomerID	CustomerName
1	John
2	Lisa

#### Таблица Orders:

OrderID	CustomerID	OrderDate
101	1	2021-01-01
102	2	2021-01-02

#### Таблица Products:

ProductID	ProductName
1	Apple
2	Banana
3	Orange

#### Таблица OrderItems:

OrderID	ProductID	Quantity
101	1	3
101	2	2
102	1	1
102	3	4

В этом примере таблица Orders была разделена на три таблицы: Customers, Orders и OrderItems.

Таблица Customers содержит информацию о клиентах, таблица Orders содержит информацию о заказах, а таблица OrderItems содержит информацию о продуктах и их количестве в каждом заказе.

Теперь все атрибуты полностью зависят от первичного ключа. Каждая таблица содержит только те атрибуты, которые непосредственно связаны с соответствующими сущностями, что соответствует критериям 2НФ.

## 19. Даталогическое проектирование. Третья нормальная форма

### Третья нормальная форма (3НФ)

---

Третья нормальная форма (3НФ) - это третий уровень нормализации данных в реляционной базе данных. В 3НФ таблица должна быть в 2НФ, и все неключевые атрибуты должны зависеть только от первичного ключа, а не от других неключевых атрибутов.

Критерии для достижения 3НФ:

1. Таблица должна быть в 2НФ.
2. Все неключевые атрибуты должны зависеть только от каждого отдельного первичного ключа, а не от других неключевых атрибутов.

Пример таблицы во второй нормальной форме:

**Таблица Customers:**

CustomerID	CustomerName	CustomerAddress
1	John	123 Main St
2	Lisa	456 Elm St

**Таблица Orders:**

OrderID	CustomerID	OrderDate
101	1	2021-01-01
102	2	2021-01-02

**Таблица Products:**

ProductID	ProductName	ProductPrice
1	Apple	1.99
2	Banana	0.99
3	Orange	1.49

**Таблица OrderItems:**

OrderID	ProductID	Quantity
101	1	3
101	2	2
102	1	1
102	3	4

Пример таблицы в третьей нормальной форме:

**Таблица Customers:**

CustomerID	CustomerName	CustomerAddress
1	John	123 Main St
2	Lisa	456 Elm St

**Таблица Orders:**

OrderID	CustomerID	OrderDate
101	1	2021-01-01
102	2	2021-01-02

**Таблица Products:**

ProductID	ProductName
1	Apple
2	Banana
3	Orange

**Таблица OrderItems:**

OrderID	ProductID	Quantity
101	1	3
101	2	2
102	1	1
102	3	4

**Таблица ProductPrices:**

ProductID	ProductPrice
1	1.99
2	0.99
3	1.49

В этом примере таблица Products была разделена на две таблицы: Products и ProductPrices. Таблица Products содержит только атрибуты, связанные с продуктами, а таблица ProductPrices содержит информацию о ценах на продукты.

Теперь все неключевые атрибуты зависят только от первичного ключа каждой таблицы, а не от других неключевых атрибутов, что соответствует критериям ЗНФ. Это способствует избежанию аномалий и обеспечивает более эффективную структуру данных.

## 20. Даталогическое проектирование. **Нормальная форма Бойса-Кодда**

**Нормальная форма Бойса-Кодда (НФБК)** - это четвёртый уровень нормализации данных в реляционной базе данных. В НФБК таблица должна быть в третьей нормальной форме (ЗНФ) и не должна иметь зависимостей между неключевыми атрибутами.

Критерии для достижения НФБК:

1. Таблица должна быть в ЗНФ.
2. Не должно быть транзитивных зависимостей между неключевыми атрибутами.

Пример таблицы в третьей нормальной форме:

**Таблица Customers:**

CustomerID	CustomerName	CustomerAddress
1	John	123 Main St
2	Lisa	456 Elm St

**Таблица Orders:**

OrderID	CustomerID	OrderDate
101	1	2021-01-01
102	2	2021-01-02

**Таблица Products:**

ProductID	ProductName
1	Apple
2	Banana
3	Orange

**Таблица OrderItems:**

OrderID	ProductID	Quantity
101	1	3
101	2	2
102	1	1

<b>OrderID</b>	<b>ProductID</b>	<b>Quantity</b>
----------------	------------------	-----------------

102	3	4
-----	---	---

Пример таблицы в нормальной форме Бойса-Кодда:

**Таблица Customers:**

<b>CustomerID</b>	<b>CustomerName</b>	<b>CustomerAddress</b>
1	John	123 Main St
2	Lisa	456 Elm St

**Таблица Orders:**

<b>OrderID</b>	<b>CustomerID</b>	<b>OrderDate</b>
101	1	2021-01-01
102	2	2021-01-02

**Таблица Products:**

<b>ProductID</b>	<b>ProductName</b>
1	Apple
2	Banana
3	Orange

**Таблица OrderItems:**

<b>OrderID</b>	<b>OrderItemID</b>	<b>ProductID</b>	<b>Quantity</b>
101	1	1	3
101	2	2	2
102	1	1	1
102	2	3	4

В этом примере таблица OrderItems была дополнена новым атрибутом OrderItemID, который служит первичным ключом для таблицы OrderItems. Такая структура позволяет избежать транзитивных зависимостей между неключевыми атрибутами, что соответствует критериям НФБК.

Всего существует 7 нормальных форм, но эти 4 являются фундаментальными.

**Для примера 4:**

**Четвёртая нормальная форма (4НФ)** - это четвёртый уровень нормализации данных в реляционной базе данных. В 4НФ таблица должна быть в третьей нормальной форме (3НФ) и не должна иметь многозначных зависимостей.

Критерии для достижения 4НФ:

1. Таблица должна быть в 3НФ.
2. Не должно быть многозначных зависимостей.

Многозначная зависимость - это ситуация, когда один или более атрибутов в таблице зависят от других атрибутов, образуя множество возможных значений для каждого набора ключевых атрибутов.

Пример таблицы в третьей нормальной форме:

**Таблица Students:**

StudentID	StudentName	Course	Instructor
1	John	Math	Smith
2	Lisa	Science	Johnson
3	Mark	Math	Smith
4	Emily	History	Brown

Пример таблицы в четвёртой нормальной форме:

**Таблица Students:**

StudentID	StudentName
1	John
2	Lisa
3	Mark
4	Emily

**Таблица Courses:**

CourseID	Course
1	Math
2	Science
3	History

**Таблица Instructors:**

InstructorID	Instructor
1	Smith

InstructorID	Instructor
2	Johnson
3	Brown

**Таблица StudentCourses:**

StudentID	CourseID
1	1
2	2
3	1
4	3

В этом примере таблица Students была разделена на несколько таблиц: Students, Courses и Instructors. Затем была создана отдельная таблица StudentCourses для связи студентов с курсами, удалив тем самым многозначные зависимости. Каждая таблица содержит только уникальные значения атрибутов, что соответствует критериям 4НФ.

---

## 21. Преобразование ER-модели в реляционную модель данных

- Каждой сущности ставится в соответствие **отношение реляционной** модели данных. При этом имена сущности и отношения могут быть различными (имена отношений могут быть ограничены требованиями конкретной СУБД).
- Каждый **атрибут сущности** становится атрибутом соответствующего отношения. Переименование атрибутов должно происходить в соответствии с теми же правилами, что и переименование отношений в п.1. Для каждого атрибута задаётся конкретный допустимый в СУБД тип данных и допустимость или недопустимость NULL значений для него.
- Первичный ключ** сущности становится PRIMARY KEY соответствующего отношения. Атрибуты, входящие в первичный ключ отношения, автоматически получают свойство обязательности (NOT NULL). В каждое отношение, соответствующее **подчинённой сущности**, добавляется набор атрибутов основной сущности, являющейся первичным ключом основной сущности. В отношении, соответствующем подчинённой сущности, этот набор атрибутов становится **внешним ключом** (FOREIGN KEY).

СОТРУДНИК	
Табельный номер	
Фамилия	
Имя	
Отчество	
Количество детей	

EMPLOYEE	
T_NUM	int
NAME	varchar(30)
F_NAME	varchar(30)
L_NAME	varchar(30)
COUNT_CH	varchar(30)

- Для моделирования необязательного типа связи на физическом уровне у атрибутов, соответствующих внешнему ключу, устанавливается свойство допустимости неопределённых

значений (признак NULL). При обязательном типе связи атрибуты получают свойство отсутствия неопределённых значений (признак NOT NULL).

5. Для отражения категоризации сущностей при переходе к реляционной модели возможны несколько вариантов представления:

- Возможно создать только одно отношение для всех подтипов одного супертипа. В него включают все атрибуты всех подтипов. Однако тогда для ряда экземпляров ряд атрибутов не будет иметь смысла. Достоинство - создаётся всего одно отношение.
- При втором способе для каждого подтипа и для супертипа создаются свои отдельные отношения. Недостаток - создаётся много отношений, однако работа ведётся только со значимыми атрибутами подтипа. Кроме того, для возможности переходов к подтипам от супертипа необходимо в супертип включить идентификатор связи.

Необходимо указать тип дескриптора - взаимоисключающий (M/E, mutually exclusive) или нет. Тип M/E означает, что один экземпляр сущности супертипа связан только с одним экземпляром сущности подтипа и для каждого экземпляра сущности супертипа существует потомок. Также необходимо указать, что наследуется в подтипы - только идентификатор супертипа или все атрибуты супертипа.

### Разрешение связей типа "многие-ко-многим"

Используется специальный механизм преобразования - вводится специальное дополнительное связующее отношение, которое связано с каждым исходным связью "один-ко-многим", атрибутами этого отношения являются первичные ключи связываемых отношений.

При этом каждый из атрибутов нового отношения является внешним ключом (FOREIGN KEY), а вместе они образуют первичный ключ (PRIMARY KEY) нового связующего отношения.

Теория нормализации применима к модели "сущность-связь". Поэтому нормализацию можно проводить и на уровне инфологической (семантической) модели и смысл её аналогичен нормализации реляционной модели.

## 22. Язык SQL. Типы данных

**SQL (Structured Query Language)** – структурированный язык запросов – стандартный язык запросов по работе с реляционными БД.

### Структура SQL:

- Операторы определения данных - Data Definition Language (DDL)
- Операторы манипулирования данными Data Manipulation Language (DML) (базируется на операциях реляционной алгебры)
- Язык запросов Data Query Language (DQL) (базируются на языках реляционной алгебры)
- Средства управления транзакциями
- Средства администрирования данных

В коммерческих СУБД набор основных операторов расширен. В большинство СУБД включены операторы определения и запуска хранимых процедур и операторы определения триггеров.

## Типы данных:

### Типы данных

В языке SQL/89 поддерживаются следующие типы данных:

- CHARACTER(n) или CHAR(n) — символьные строки постоянной длины в n символов.
- NUMERIC[(n,m)] — точные числа, здесь n — общее количество цифр в числе, m — количество цифр слева от десятичной точки.
- DECIMAL[(n,m)] — точные числа, здесь n — общее количество цифр в числе, m — количество цифр слева от десятичной точки.
- DEC[(n,m)] — то же, что и DECIMAL[(n,m)].
- INTEGER или INT — целые числа.
- SMALLINT — целые числа меньшего диапазона. В большинстве реализаций тип данных INTEGER соответствует целым числам, хранимым в четырех байтах, а SMALLINT — соответствует целым числам, хранимым в двух байтах.
- FLOAT[(n)] — числа большой точности, хранимые в форме с плавающей точкой. Здесь n — число байтов, резервируемое под хранение одного числа.
- REAL — вещественный тип чисел, который соответствует числам с плавающей точкой, меньшей точности, чем FLOAT.
- DOUBLE PRECISION специфицирует тип данных с определенной в реализации точностью большей, чем определенная в реализации точность для REAL.

В стандарте SQL92 добавлены следующие типы данных:

- VARCHAR(n) — строки символов переменной длины.
- NCHAR(N) — строки локализованных символов постоянной длины.
- NCHAR VARYING(n) — строки локализованных символов переменной длины.
- BIT(n) — строка битов постоянной длины.
- BIT VARYING(n) — строка битов переменной длины.
- DATE — календарная дата.
- TIMESTAMP(точность) — дата и время.
- INTERVAL — временной интервал.

Числовые константы могут задаваться так:

213.314    612.716    +551.702

Константы с плавающей запятой задаются так:

2.9E-4    -134.235E7    0.54267E18

Строковые константы должны быть заключены в кавычки:

'Крылов Ю.Д.'    'Санкт-Петербург'

В СУБД также могут существовать и специальные **системные константы**. Стандарт SQL/89 определяет только одну системную константу USER – имя, под которым пользователь подключился к БД.

## 23. Язык SQL. Операторы определения данных DDL

## Язык описания данных (*Data Definition Language*)

Оператор	Смысл	Действие
<b>CREATE TABLE</b>	Создать таблицу	Создает новую таблицу в БД
<b>DROP TABLE</b>	Удалить таблицу	Удаляет таблицу из БД
<b>ALTER TABLE</b>	Изменить таблицу	Изменяет структуру существующей таблицы или ограничения целостности, задаваемые для данной таблицы
<b>CREATE VIEW</b>	Создать представление	Создает виртуальную таблицу, соответствующую некоторому SQL-запросу
<b>ALTER VIEW</b>	Изменить представление	Изменяет ранее созданное представление
<b>DROP VIEW</b>	Удалить представление	Удаляет ранее созданное представление
<b>CREATE INDEX</b>	Создать индекс	Создает индекс для некоторой таблицы для обеспечения быстрого доступа по атрибутам, входящим в индекс
<b>DROP INDEX</b>	Удалить индекс	Удаляет ранее созданный индекс

Операторы DDL с заданием ограничений целостности

**Целостность** - одно из основополагающих понятий в технологии БД. Это соответствие информационной модели предметной области, хранимой в БД, объектам реального мира и их взаимосвязи в любой момент времени.

Любое значимое изменение в предметной области должно отражаться в БД, и при этом должна сохраняться однозначная интерпретация информационной модели в терминах предметной области.

**Поддержка целостности** в реляционной модели данных включает в себя 4 аспекта:

1. Поддержка структурной целостности - реляционная СУБД должна допускать работу только с однородными структурами данных типа "реляционное отношение"
2. Поддержка языковой целостности - реляционная СУБД должна обеспечивать языки описания и манипулирования данными не ниже стандарта SQL
3. Поддержка ссылочной целостности - обеспечение одного из заданных принципов взаимосвязи между экземплярами кортежей взаимосвязанных отношений: каскадное изменение и удаление.
4. Семантическая целостность - (используется, когда предыдущие 3 не выполняются) может быть обеспечена 2 путями: декларативный и процедурный.
  - Декларативный связан с наличием механизмов в рамках СУБД, которые обеспечивают проверку и выполнения ряда декларативных правил-ограничений. Например, ограничение целостности атрибута. Мы его по умолчанию задаем, можем задать обязательность (необязательность), пределы. Декларативные ограничения – немедленно проверяемые.
  - Процедурные ограничения – откладываемые, они осуществляются с помощью транзакций и триггеров.

Эти виды целостности определяют правила работы СУБД с РБД. **Каждая СУБД должна уметь это делать**, а разработчики должны это учитывать при построении РБД.

Набор операторов языка SQL принято называть **скриптом**.

Тогда скрипт для некоторых таблиц БД "Библиотека" будет выглядеть следующим образом:

```
CREATE TABLE BOOKS
(
    ISBN (название) varchar (120) (тип данных) NOT NULL (доп.ограничение столбца),
    PRIMARY KEY (ограничение уникальности столбца),
    TITLE varchar (120) NOT NULL,
    YEAR_PUBL smallint DEFAULT (Year (Now)) (значение по умолчанию),
    PUBLISHER varchar (20) NULL,
    PAGES smallint CHECK (PAGES >=5 AND PAGES <= 1000) (условие проверки на
    допустимость),
)
```

## Оператор модификации таблиц

### ALTER TABLE

```
ALTER TABLE table_name
ADD column_name datatype;
```

- позволяет выполнить следующие операции изменения для схемы таблиц:
- Добавить новый столбец в уже существующую заполненную таблицу
- Изменить значение по умолчанию для какого-либо столбца
- Удалить столбец из существующей таблицы
- Добавить или удалить первичный ключ таблицы
- Добавить или удалить внешний ключ таблицы
- Добавить или удалить условие уникальности
- Добавить или удалить условие проверки для любого столбца или для таблицы в целом

### DROP TABLE

Синтаксис: **DROP TABLE** <имя таблицы> [**CASCADE** | **RESTRICT**]

- **CASCADE** – при удалении таблицы будут удаляться все связанные с ней объекты.
- **RESTRICT** – все объекты, связанные с таблицей, не будут уничтожены.

Удаление таблицы, не должно нарушать целостности БД. Удаляем в порядке обратном созданию.

```
DROP DATABASE testDB;
```

## Язык SQL. Операторы манипулирования данными DML

## Язык манипулирования данными (*Data Manipulation Language*)

Оператор	Смысл	Действие
<b>DELETE</b>	Удалить строки	Удаляет одну или несколько строк, соответствующих условиям фильтрации, из базовой таблицы. Применение оператора согласуется с принципами поддержки целостности, поэтому этот оператор не всегда может быть выполнен корректно, даже если синтаксически он записан правильно
<b>INSERT</b>	Вставить строку	Вставляет одну строку в базовую таблицу. Допустимы модификации оператора, при которых сразу несколько строк могут быть перенесены из одной таблицы или запроса в базовую таблицу
<b>UPDATE</b>	Обновить строку	Обновляет значения одного или нескольких столбцов в одной или нескольких строках, соответствующих условиям фильтрации

- **DELETE** - statement is used to delete existing records in a table.

```
DELETE FROM table_name WHERE condition;
```

- **INSERT INTO** - statement is used to insert new records in a table.

```
INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode,
Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006',
'Norway');
```

- **UPDATE** - statement is used to modify the existing records in a table.

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

## 25. Язык SQL. Оператор выбора DQL

### SELECT

Единственный оператор выбора **SELECT** реализует все операции реляционной алгебры. Один и тот же запрос может быть реализован несколькими способами.

```
SELECT [ALL|DISTINCT](<Список полей>|*)
```

```
FROM <Список таблиц>

[WHERE<Предикат-условие выборки или соединения>]

[GROUP BY<Список полей результата>]

[HAVING<Предикат-условие для группы>]

[ORDER BY<Список полей, по которым упорядочить вывод>]
```

```
SELECT column1, column2, ...
FROM table_name;
```

The **HAVING** clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5
ORDER BY COUNT(CustomerID) DESC;
```

## Агрегатные функции

Функция	Результат
COUNT	Количество строк или непустых значений полей, которые выбрал запрос
SUM	Сумма всех выбранных значений данного поля
AVG	Среднеарифметическое значение всех выбранных значений данного поля
MIN	Наименьшее из всех выбранных значений данного поля
MAX	Наибольшее из всех выбранных значений данного поля

```
SELECT R1. Оценка, R1. Дисциплина, COUNT (*), AVG(Оценка)
FROM R1, R2
WHERE R1. ФИО = R2. ФИО
AND R1. Оценка IS NOT NULL
AND R1. Оценка > 2
GROUP BY R1. Оценка R1. Дисциплина
```

Дисциплина	COUNT(*)	AVR(Оценка)
Базы данных	6	3.83
Теория информации	3	3.67
Сети и телекоммуникации	3	4.66
Иностранный язык	4	4.25

## 26. Язык SQL. Средства администрирования данных

### Средства администрирования данных

Оператор	Смысл	Действие
<b>ALTER DATABASE</b>	Изменить БД	Изменить набор основных объектов в базе данных, ограничений, касающихся всей базы данных
<b>ALTER DBAREA</b>	Изменить область хранения БД	Изменить ранее созданную область хранения
<b>ALTER PASSWORD</b>	Изменить пароль	Изменить пароль для всей базы данных
<b>CREATE DATABASE</b>	Создать БД	Создать новую базу данных, определив основные параметры для нее
<b>CREATE DBAREA</b>	Создать область хранения	Создать новую область хранения и сделать ее доступной для размещения данных
<b>DROP DATABASE</b>	Удалить БД	Удалить существующую базу данных (только в том случае, когда вы имеете право выполнить это действие)
<b>DROP DBAREA</b>	Удалить область хранения БД	Удалить существующую область хранения (если в ней на настоящий момент не располагаются активные данные)
<b>GRANT</b>	Предоставить права	Предоставить права доступа на ряд действий над некоторым объектом БД
<b>REVOKE</b>	Лишить прав	Лишить прав доступа к некоторому объекту или некоторым действиям над объектом

The **CREATE TABLE** statement is used to create a new table in a database.

```
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

**PASSWORD** используется системой аутентификации в MySQL для генерации хэшированного пароля из строки пароля обычного текста с использованием более мощных методов хеширования.

Команда **GRANT** используется для назначения привилегий пользователям.

Пример: передача прав на SELECT STUDENT

```
GRANT SELECT ON Student TO P2;
```

## 27. Защита информации в базах данных

В современных СУБД поддерживается один из двух подходов к вопросу обеспечения безопасности данных: **избирательный подход** и **обязательный подход**.

В обоих подходах единицей данных или «объектом данных», для которых должна быть создана система безопасности, может быть как вся база данных целиком, так и любой объект внутри базы данных.

Необходимо поддерживать два фундаментальных принципа:

- **Проверка полномочий** – каждому пользователю соответствует набор действий, которые он может выполнять по отношению к определенным объектам.
- **Проверка подлинности** – достоверное подтверждение того, что пользователь, пытающийся выполнить санкционированное действие, действительно тот, за кого он себя выдает.

## 28. Реализация реляционной модели данных в СУБД.

Реляционная модель данных является основой для большинства современных систем управления базами данных (СУБД). Она представляет данные в виде таблиц, состоящих из строк и столбцов, где каждая строка представляет отдельную запись, а каждый столбец представляет отдельное поле данных.

1. Каждой сущности ставится в соответствие отношение реляционной модели данных. При этом имена сущности и отношения могут быть различными (имена отношений могут быть ограничены требованиями конкретной СУБД).
2. Каждый атрибут сущности становится атрибутом соответствующего отношения. Переименование атрибутов должно происходить в соответствии с теми же правилами, что и переименование отношений в п.1. Для каждого атрибута задается конкретный допустимый в СУБД тип данных и обязательность или необязательность данного атрибута (то есть допустимость или недопустимость NULL значений для него).
3. Первичный ключ сущности становится PRIMARY KEY соответствующего отношения. Атрибуты, входящие в первичный ключ отношения, автоматически получают свойство обязательности (NOT NULL).
4. В каждое отношение, соответствующее подчиненной сущности, добавляется набор атрибутов основной сущности, являющейся первичным ключом основной сущности. В отношении, соответствующем подчиненной сущности, этот набор атрибутов становится внешним ключом (FOREIGN KEY).
5. Для моделирования необязательного типа связи на физическом уровне у атрибутов, соответствующих внешнему ключу, устанавливается свойство допустимости неопределенных значений (признак NULL). При обязательном типе связи атрибуты получают свойство отсутствия неопределенных значений (признак NOT NULL).
6. Для отражения категоризации сущностей при переходе к реляционной модели возможны несколько вариантов представления.
  - Возможно создать только одно отношение для всех подтипов одного супертипа. В него включают все атрибуты всех подтипов. Однако тогда для ряда экземпляров ряд атрибутов

не будет иметь смысла. Достоинство - создается всего одно отношение.

- При втором способе для каждого подтипа и для супертипа создаются свои отдельные отношения. Недостаток - создается много отношений, однако работа ведется только со значимыми атрибутами подтипа. Кроме того, для возможности переходов к подтипам от супертипа необходимо в супертип включить идентификатор связи.

Необходимо указать тип дискриминатора - взаимоисключающий (M/E, mutually exclusive) или нет. Тип М/Е означает, что один экземпляр сущности супертипа связан только с одним экземпляром сущности подтипа и для каждого экземпляра сущности супертипа существует потомок. Также необходимо указать, что наследуется в подтипы - только идентификатор супертипа или все атрибуты супертипа.

## 29. Распределенная обработка данных. Клиент-серверная архитектура

Распределенная обработка данных

# Распределенная обработка данных

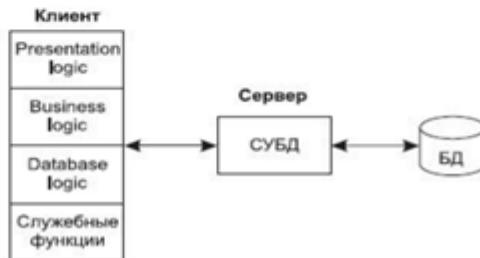


Клиент-серверная архитектура

## Модели «клиент—сервер»

Основной принцип технологии «клиент—сервер» заключается в разделении функций стандартного интерактивного приложения на 5 групп:

- 1) функции ввода и отображения данных (*Presentation Logic*);
- 2) прикладные функции, определяющие основные алгоритмы решения задач приложения (*Business Logic*);
- 3) функции обработки данных внутри приложения (*Database Logic*);
- 4) функции управления информационными ресурсами (*Database Manager System*);
- 5) служебные функции, играющие роль связок между функциями первых четырех групп.



30. Файл-серверная архитектура. Модель удаленного управления данными

## Двухуровневые модели

### Модель удаленного управления данными



- **Достоинства:**

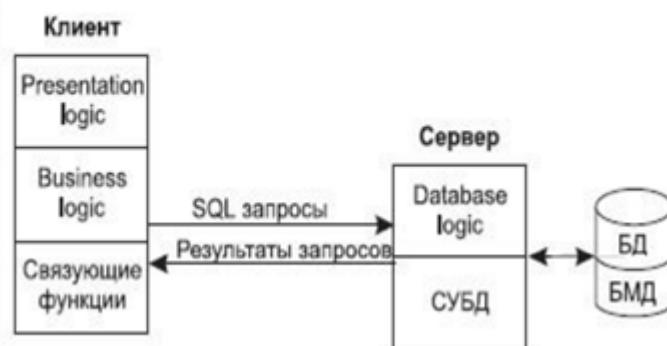
- многопользовательский режим работы с данными
- удобство централизованного управления доступом
- низкая стоимость разработки
- высокая скорость разработки
- невысокая стоимость обновления и изменения ПО

- **Недостатки:**

- высокий сетевой трафик
- низкая производительность
- плохая возможность подключения новых клиентов
- ненадежность системы

## 31. Двухуровневая клиент-серверная архитектура. Модель удаленного доступа к данным

### *Модель удаленного доступа к данным*



- **Достоинства:**

- существенная разгрузка сервера БД
- загрузка сервера операциями обработки данных, запросов и транзакций
- резкое уменьшение загрузки сети
- унификация интерфейса «клиент-сервер» - язык SQL становится стандартом

- **Недостатки:**

- излишнее дублирование кода приложений
- сложное администрирование
- высокая стоимость оборудования
- бизнес логика приложений осталась в клиентском ПО

## 32. Двухуровневая клиент-серверная архитектура. Модель сервера баз данных

## Модель сервера баз данных



Основу данной модели составляют

- механизм хранимых процедур как средство программирования SQL-сервера,
- механизм триггеров как механизм отслеживания текущего состояния информационного хранилища ,
- механизм ограничений на пользовательские типы данных (механизм поддержки доменной структуры).

- **Достоинства:**

- резкое уменьшение трафика обмена информацией между клиентом и сервером
- использование механизма триггеров для централизованного контроля
- сервер является активным
- существенное уменьшение дублирования алгоритмов обработки данных в разных клиентских приложениях

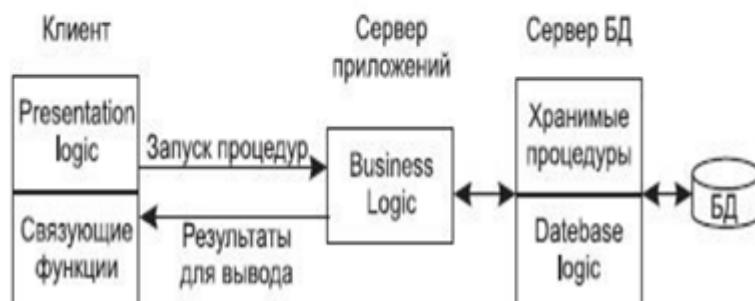
- **Недостатки:**

- очень большая загрузка сервера
- неработоспособность сервера может сделать неработоспособной всю вычислительную сеть
- сложное администрирование

## 33. Трехуровневая клиент-серверная архитектура. Модель сервера приложений

### Трехуровневые модели

#### Модель сервера приложений



- **Достоинства:**

- клиентское ПО не нуждается в администрировании
- масштабируемость
- конфигурируемость
- высокая безопасность и надежность
- низкие требования к скорости канала между терминалами и сервером приложений
- низкие требования к производительности и техническим характеристикам терминалов

- **Недостатки:**

- сложность администрирования и обслуживания
- более высокая сложность создания приложений
- высокие требования к производительности серверов приложений и сервера базы данных
- высокие требования к скорости канала (сети) между сервером базы данных и серверами приложений

## 34. Трехуровневая клиент-серверная архитектура. Модель доступа через Internet/Intranet

### ***Модель доступа через Internet/Intranet***

Клиент – компьютер , оснащенный Веб-браузером.

На сервере приложений - бизнес-логика и логика обработки данных в виде бизнес-объектов.

#### **Особенности:**

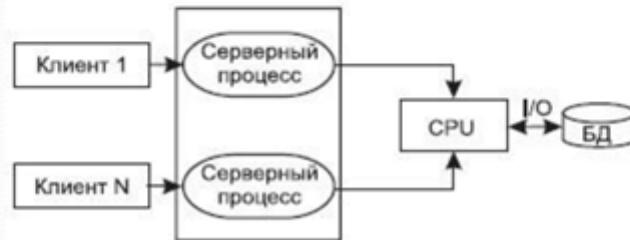
- отсутствие необходимости использовать дополнительное ПО на стороне клиента
- возможность подключения практически неограниченного количества клиентов
- централизованное место хранения данных
- недоступность при отсутствии работоспособности сервера или каналов связи
- достаточно низкая скорость Веб-сервера и каналов передачи данных
- нет существенных ограничений относительно объема данных

## 35. Варианты моделей серверов баз данных

# Модели серверов баз данных

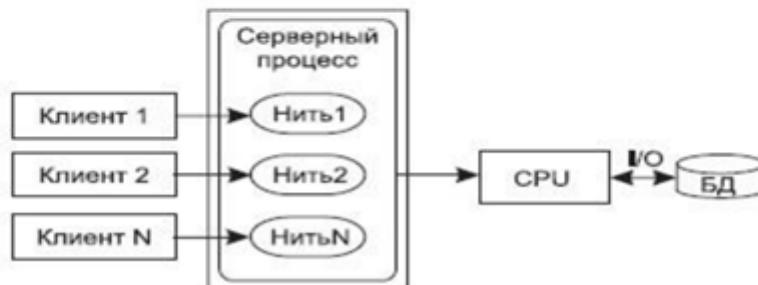
**Нулевой этап** развития серверов БД - управление данными (функция сервера) и взаимодействие с пользователем были совмещены в одной программе.

Модель «**один-к-одному**» - сервер обслуживал запросы только одного пользователя (клиента), и для обслуживания нескольких клиентов нужно было запустить эквивалентное число серверов.

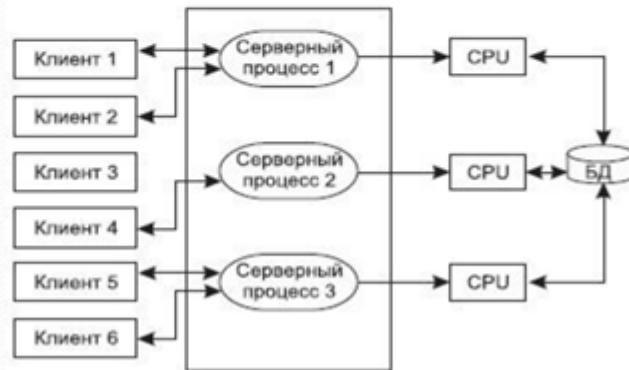


Модель **многопотоковая односерверная** - система с выделенным сервером, который способен обрабатывать запросы от многих клиентов. Сервер единственный обладает монополией на управление данными и взаимодействует одновременно со многими клиентами.

Логически каждый клиент связан с сервером отдельной нитью («*thread*»), или потоком, по которому пересылаются запросы.



Модель **многонитевая мультисерверная** - возможность запуска нескольких серверов базы данных, в том числе и на различных процессорах, при этом каждый из серверов должен быть многопоточным.



## 36. Типы параллелизма в многопотоковой архитектуре

### Типы параллелизма

