



Белорусско-Российский университет

Кафедра «Программное обеспечение информационных технологий»

# Информатика

---

# Базы данных и системы управления базами данных

---

КУТУЗОВ Виктор Владимирович

Республика Беларусь, Могилев, 2023



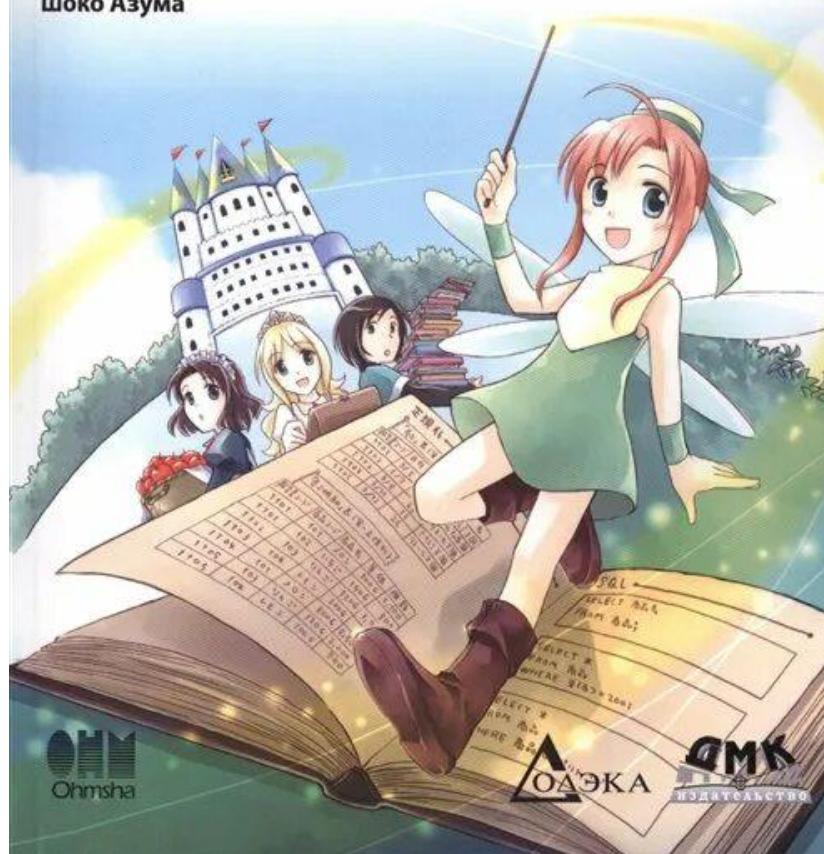
# Литература по Базам данных и системам управления базами данных

# Литература

ЗАНИМАТЕЛЬНОЕ ПРОГРАММИРОВАНИЕ МАНГА

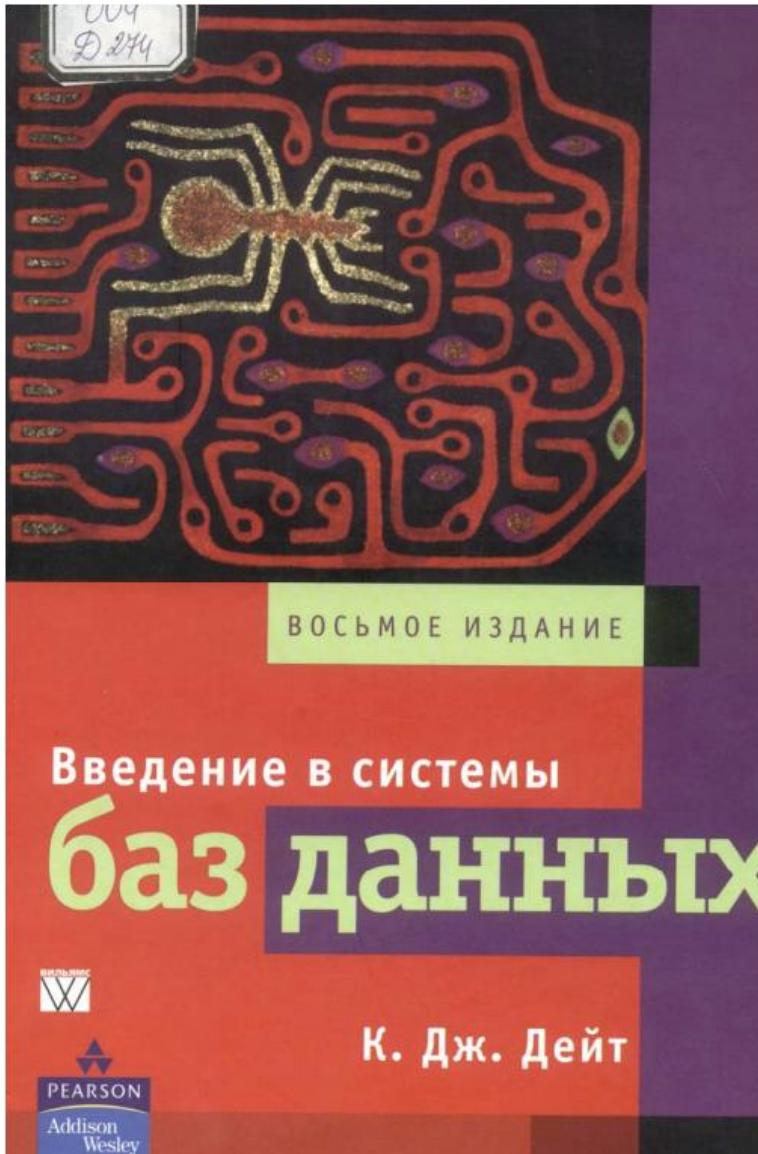
## БАЗЫ ДАННЫХ

Мана Такахаси  
Шоко Азума



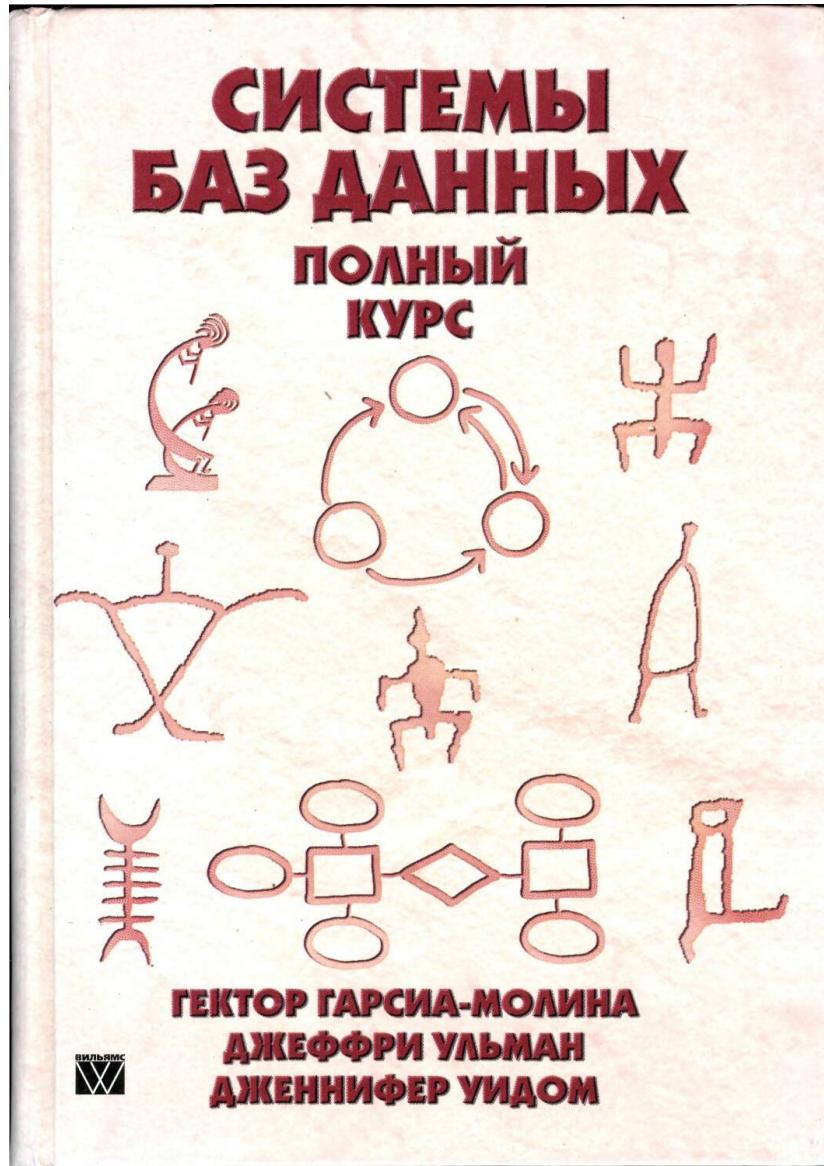
- Такахаси, Мана. Занимательное программирование. **Базы данных.** Манга / Мана Такахаси (автор), Сёко Адзума (худож.) ; пер. Сенниковой Т. И. — М. : ДМК Пресс, 2014. — 238 с.
- У принцессы Руруны и Кейна возникла проблема: в их торгующей фруктами империи царит неразбериха из-за противоречивых данных, и поэтому дыни подменяются яблоками и клубникой, что вызывает большие трудности в работе. И что же им делать? Конечно же, построить реляционную базу данных, и поможет им в этом Тико, чудесная фея баз данных. Она покажет Руруне и Кейну, как создать базу данных, которая поможет управлять продажами, реализацией товара и его экспортом. Вы узнаете, как работает база данных, и поймёте значение таких терминов, как схемы, ключи, нормализация и транзакции.
- Если у вас голова идёт кругом, когда речь заходит о базах данных, или же вы просто заплутали в лабиринте чисел и данных, которые, как вам кажется, неподвластны контролю, присоединяйтесь к Руруне и Кейну.

# Литература



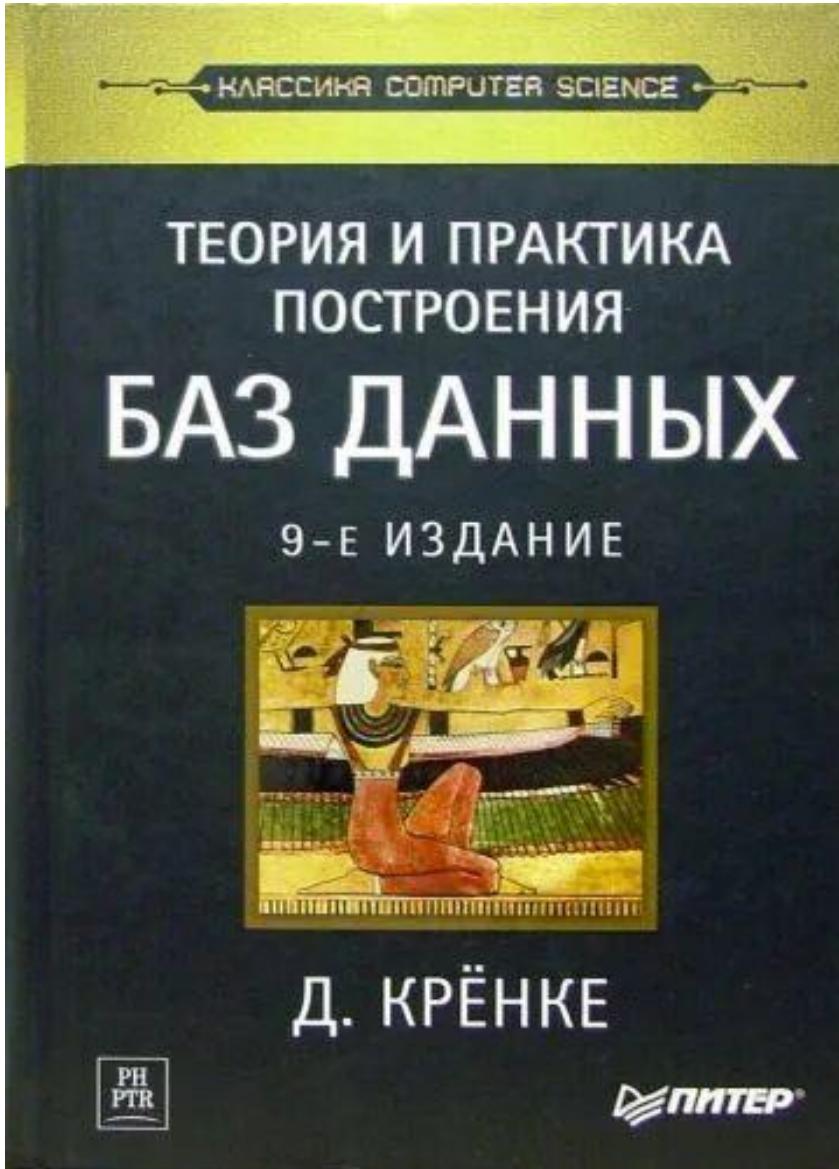
- Дейт, К. Дж. **Введение в системы баз данных**, 8-е издание.: Пер. с англ. — М.: Издательский дом "Вильямс", 2005. — 1328 с.
- Новое издание фундаментального труда Криса Дейта представляет собой исчерпывающее введение в очень обширную в настоящее время теорию систем баз данных. С помощью этой книги читатель сможет приобрести фундаментальные знания в области технологии баз данных, а также ознакомиться с направлениями, по которым рассматриваемая сфера деятельности, вероятно, будет развиваться в будущем. Книга предназначена для использования в основном в качестве учебника, а не справочника, и поэтому, несомненно, вызовет интерес у программистов-профессионалов, научных работников и студентов, изучающих соответствующие курсы в высших учебных заведениях. В ней сделан акцент на усвоении сути и глубоком понимании излагаемого материала, а не просто на его формальном изложении.
- Книга, безусловно, будет полезна всем, кому приходится работать с базами данных или просто пользоваться ими

# Литература



- Гарсиа-Молина, Гектор, Ульман, Джейфри, Д., Уидом, Джениффер **Системы баз данных. Полный курс.** : Пер. с англ. — М. : Издательский дом “Вильямс”, 2003. — 1088 с.
- Книга известного специалиста в области компьютерных наук Дж. Ульмана и его именитых коллег по Станфордскому университету является уникальным учебным и справочным пособием, которое отличается беспрецедентными широтой и глубиной охвата предмета и представляет несомненный интерес для всех, кто по роду своей профессиональной деятельности сталкивается с проблемами проектирования и использования современных систем баз данных.

# Литература



- Крёнке Д. **Теория и практика построения баз данных** 9-е изд. — СПб. Питер 2005 — 859 с.
- В книге Д. Крёнке, выдержавшей уже 9 переизданий, вы найдете традиционно подробный, методически выверенный теоретический и практический материал, посвященный вопросам разработки и использования баз данных.
- В новом издании более глубоко обсуждаются моделирование данных и проектирование баз данных; расширены разделы по SQL и XML; добавлен раздел, знакомящий с ADO NET.
- Книгу отличает большое количество примеров, моделирующих типичные ситуации из практики делового мира.

# Литература



- Болье, А. **Изучаем SQL. Генерация, выборка и обработка данных**, 3-е изд./ Алан Болье; пер. с англ. И.В. Красикова. — Киев.: “Диалектика”, 2021.— 402 с.
- Данная книга отличается широким охватом как тем (от азов SQL до таких сложных вопросов, как аналитические функции и работа с большими базами данных), так и конкретных баз данных (MySQL, Oracle Database, SQL Server) и особенностей реализации тех или иных функциональных возможностей SQL на этих серверах. Книга идеально подходит в качестве учебника для начинающего разработчика в области баз данных. В ней описаны все возможные применения языка SQL и наиболее распространенные серверы баз данных.

# Литература

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»  
(САМАРСКИЙ УНИВЕРСИТЕТ)

A. A. АГАФОНОВ, A. M. БЕЛОВ

## ОСНОВЫ ТЕХНОЛОГИЙ БАЗ ДАННЫХ

Рекомендовано редакционно-издательским советом федерального государственного автономного образовательного учреждения высшего образования «Самарский национальный исследовательский университет имени академика С. П. Королева» в качестве учебного пособия для обучающихся по основной образовательной программе высшего образования по специальности 10.05.03 Информационная безопасность автоматизированных систем

САМАРА

Издательство Самарского университета  
2023

- Агафонов, А. А. **Основы технологий баз данных**: учебное пособие / А.А. Агафонов, А.М. Белов. – Самара: Издательство Самарского университета, 2023. – 304 с.
- Учебное пособие посвящено технологиям проектирования, разработки и внедрения баз данных. В пособии рассматриваются основные понятия реляционных баз данных и реляционной алгебры, вопросы проектирования баз данных. Несколько разделов посвящены языку запросов SQL и его использованию для получения и модификации данных, определению объектов баз данных, таких как таблицы, индексы или представления, а также вопросам использования процедурных расширений языка SQL. Кроме того, рассмотрены NoSQL-решения, предназначенные для хранения денормализованных данных, что может быть более эффективным в распределенной среде, а также принципы и программные инструменты обработки больших данных.

# Литература



- Куликов, С. С. **Реляционные базы данных в примерах. Практическое пособие для программистов и тестировщиков** / С.С. Куликов. - Минск: EPAM Systems, 2023. - 424 с.
- Эта книга посвящена практическому взгляду на реляционную теорию и проектирование реляционных баз данных. Здесь не рассматриваются такие фундаментальные основы, как реляционная алгебра и реляционное счисление, но с множеством примеров и пояснений показаны основные понятия и подходы, необходимые для проектирования баз данных. Этот материал в первую очередь будет полезен тем, кто: никогда не изучал базы данных; когда-то изучал базы данных, но многое забыл; хочет систематизировать имеющиеся знания.
- Все схемы баз данных в этой книге приведены в нотации UML 2.1, созданы с использованием Sparx Enterprise Architect и (если речь идёт об уровнях проектирования, для которых это актуально) ориентированы на MySQL 8.0, Microsoft SQL Server 2019, Oracle 18c. Скорее всего, приведённые решения будут успешно работать на более новых версиях этих СУБД, но не на более старых. Материал книги построен таким образом, что его можно как изучать последовательно, так и использовать как быстрый справочник (на все необходимые пояснения в тексте даны ссылки). В дополнение к тексту данной книги рекомендуется пройти бесплатный онлайн-курс, содержащий серию видео-уроков, тестов и заданий для самоподготовки.
- [https://svyatoslav.biz/relational\\_databases\\_book/](https://svyatoslav.biz/relational_databases_book/)

# Литература



- Куликов, С. С. **Работа с MySQL, MS SQL Server и Oracle в примерах. Практическое пособие для программистов и тестировщиков** (2-е издание)/ С.С. Куликов. - Минск: EPAM Systems, 2023. - 592 с.
- В книге: 3 СУБД, 50 примеров, 129 задач, более 500 запросов с пояснениями и комментариями. От SELECT \* до поиска кратчайшего пути в ориентированном графе; никакой теории, только схемы и код, много кода. Будет полезно тем, кто: когда-то изучал базы данных, но многое забыл; имеет опыт работы с одной СУБД, но хочет быстро переключиться на другую; хочет в предельно сжатые сроки научиться писать типичные SQL-запросы.
- [https://svyatoslav.biz/database\\_book/](https://svyatoslav.biz/database_book/)

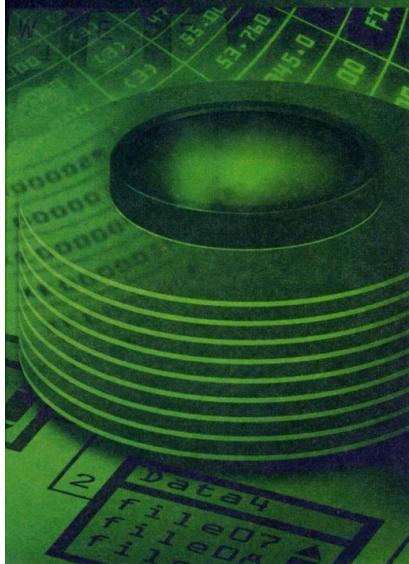
# Литература

Т. Карпова

ПИТЕР

## БАЗЫ ДАННЫХ МОДЕЛИ, РАЗРАБОТКА, РЕАЛИЗАЦИЯ

### УЧЕБНИК



- студентам вузов технических специальностей
- автор книги — преподаватель с многолетним стажем
- теоретический материал в сочетании с примерами

М. П. Малыхина

## БАЗЫ ДАННЫХ: основы, проектирование, использование

- Принципы построения баз данных и СУБД
- Концептуальное проектирование баз данных
- Реляционная модель данных
- Языки программирования баз данных
- Администрирование баз данных
- Организация работы Web-приложений в программном окружении



УЧЕБНОЕ ПОСОБИЕ

bhv®

М.Р.Когаловский

## ЭНЦИКЛОПЕДИЯ ТЕХНОЛОГИЙ БАЗ ДАННЫХ

ЭВОЛЮЦИЯ  
ТЕХНОЛОГИЙ

ТЕХНОЛОГИИ  
И СТАНДАРТЫ

ИНФРАСТРУКТУРА

ТЕРМИНОЛОГИЯ

# Литература

Файлы Теги Ищу Обсуждения О сайте Топ Вопросы и ответы (FAQ) Заказать работу

поиск по сайту

Email Пароль Войти Зарегистрироваться Восстановить пароль FAQ по входу.

Войти через:

Файлы → Академическая и специальная литература → Информатика и вычислительная техника

## Базы данных

Список файлов Последние файлы RSS

Язык SQL <sup>73</sup>

Учебные планы, программы и нормативная документация дисциплин

Учебные программы <sup>6</sup>

Справочные материалы

Энциклопедии <sup>4</sup>

Научные работы

Научные статьи и сборники <sup>8</sup>

Авторефераты и диссертации <sup>14</sup>

Учебно-методические материалы

Методички и практикумы <sup>186</sup>

Учебно-методические комплексы <sup>10</sup>

Лекции <sup>113</sup>

Тесты <sup>7</sup>

- <https://www.twirpx.com/files/science/informatics/db/> - Более двух тысяч книг по Базам данных и СУБД

# Симулятор SQL

KARPOV.COURSES

Программа

Авторы курса

FAQ

ЗАДАТЬ ВОПРОС

БЕСПЛАТНЫЙ КУРС

## СИМУЛЯТОР SQL = ПРАКТИКА НА РЕАЛЬНЫХ ЗАДАЧАХ

Анализ данных на SQL:  
с нуля до продвинутого уровня

НАЧАТЬ ОБУЧЕНИЕ



- <https://karpov.courses/simulator-sql>

# Интерактивный учебник по SQL

The screenshot shows the homepage of an interactive SQL tutorial. At the top, there's a navigation bar with a Russian flag, a British flag, the time '21:26', a 'Содержание' (Content) button, and a 'Вход' (Login) button. The main title 'SQL Задачи и решения' (SQL Tasks and Solutions) is displayed prominently, along with the subtitle 'Учебник. Сергей Моисеенко.' Below the title, a large orange 'SQL' logo is shown. A sidebar on the right contains a banner for 'Решение упражнений sql-ex.ru' with a floral pattern, and a message from Google stating 'ENHANCED BY Google'. A note about PostgreSQL support is present, along with a 'Ещё' (More) link. A yellow sidebar on the right lists recent changes: 'Последние изменения:' followed by several items like 'Функция STRING\_AGG стр. 2', 'Функция TRANSLATE', etc. At the bottom left, there are links for MySQL, PostgreSQL, Oracle, and others, with an ellipsis (...). A large yellow button in the center-right says 'ПЕРЕЙТИ к УЧЕБНИКУ' (GO TO THE BOOK). The URL <http://www.sql-tutorial.ru> is at the bottom.

# Упражнения по SQL

## Упражнения по SQL

Логин:

Пароль:

Вход

Напомнить пароль  
Регистрация

Вход без регистрации

- ▶ Рейтинги
- ▶ Статистика
- ▶ Упражнения по SQL
- ▶ Форумы
- ▶ Сертификация
- ▶ Помощь
- ▶ Оптимизация запросов

Персональная страница  
Разработчики и благодарности

Ищу работу

Ссылки

Отзывы

Поддержать проект SQL-EX.RU



<https://sql-ex.ru>

Language Русский

July 24, 17:00 MSK

Сегодня у нас **661** участников (**74** новых).  
Решено задач на рейтинговом этапе: **24**  
(**7** по **SELECT** и **17** по **DML**),  
на обучающем этапе - **1040**

## Практическое владение языком SQL

Сайт поможет каждому, кто хочет приобрести или повысить свои навыки в написании операторов манипуляции данными языка **SQL**. Цель обучения состоит в том, что вы сами пишете операторы, которые должны вернуть или изменить данные, требуемые заданием. При этом в случае неправильного ответа вы сможете узнать, какие данные возвращает правильный запрос, а также увидеть, что вернул ваш запрос. Кроме того, есть возможность выполнять любые операторы **DML** к имеющимся базам данных, отключив опцию проверки. Упражнения имеют разный уровень сложности (от 1 до 5), который пропущен во втором столбце списка упражнений. Предлагаются упражнения на выборку данных (оператор **SELECT**) и упражнения на модификацию данных (операторы **INSERT**, **UPDATE**, **DELETE** и **MERGE**). По результатам решения задач на сайте ведется рейтинг участников. При этом упражнения на выборку разбиты на три этапа: первый (4 упражнения) без контроля времени на выполнение отдельного задания, второй (начиная с 5 упражнения) - с контролем времени на выполнение каждого задания. На третьем этапе, который называется **оптимизационным** и начинается с задачи 139, требуется не только правильно решить задачу, но и время выполнения запроса должно быть соизмеримым с временем выполнения авторского решения.

Упражнения первого этапа доступны без регистрации, причем задания можно решать в любом порядке. Для выполнения остальных упражнений требуется регистрация. **Регистрация бесплатна**, как и все остальные сервисы сайта. В третьем столбце списка упражнений будут отмечаться ("OK") номера правильно выполненных упражнений для зарегистрировавшихся посетителей. Посетив наш сайт впоследствии, вам не нужно будет вспоминать, какие упражнения вами уже выполнены, а какие - нет. Исключение составляет обучающий этап. Участие в рейтинге обучающего этапа (включая отметки "OK" решенных задач) является платным. Однажды зарегистрировавшись, вы впоследствии вводите логин и пароль, указанные при регистрации. При входе без авторизации система не будет отслеживать ваши успехи. Для авторизованных пользователей доступен форум, на котором можно обсудить решения предложенных упражнений.

Убедительно просим вас **соблюдать правила сайта**.

**ЗАМЕЧАНИЕ:** неправильно сформулированный запрос может вернуть "правильные" данные на текущем состоянии базы данных. Поэтому не стоит удивляться, если результаты

Новости сайта 2023-07-14  
[Все новости](#) [Новости за неделю](#)

На этой неделе сертификаты получили:  
[Последние сертификаты](#)  
[Проверить сертификат](#)

### Именинники:

Kumachev M. ([Ceridan](#))

Дошенко В.Н. ([mcrain](#))

Здоровья и успехов!

Ближайшие дни рождения.



Закажите кружку или  
футболку с логотипом  
сайта.



NEW

[Телеграм-канал SQL-Ex.](#)

На сайте появился новый раздел - Блоги.

Официальная группа sql-ex в VK.com.

На сайте имеются упражнения по оператору **SELECT** (184 упражнения на обучающем этапе и 238 - на рейтинговых) и по другим операторам манипуляции данными - **INSERT**, **UPDATE**, **DELETE** и **MERGE** - (сейчас 55 упражнений). По упражнениям на **SELECT** ведется рейтинг участников. Смотри

[Условия тестирования](#)



Наша книга:  
**"SQL. Задачи и решения"**. Изд-во Питер  
Анализ характерных ошибок при решении задач  
обучающего этапа на сайте.



# Базы данных и системы управления базами данных

# Зачем нужно изучать базы данных?

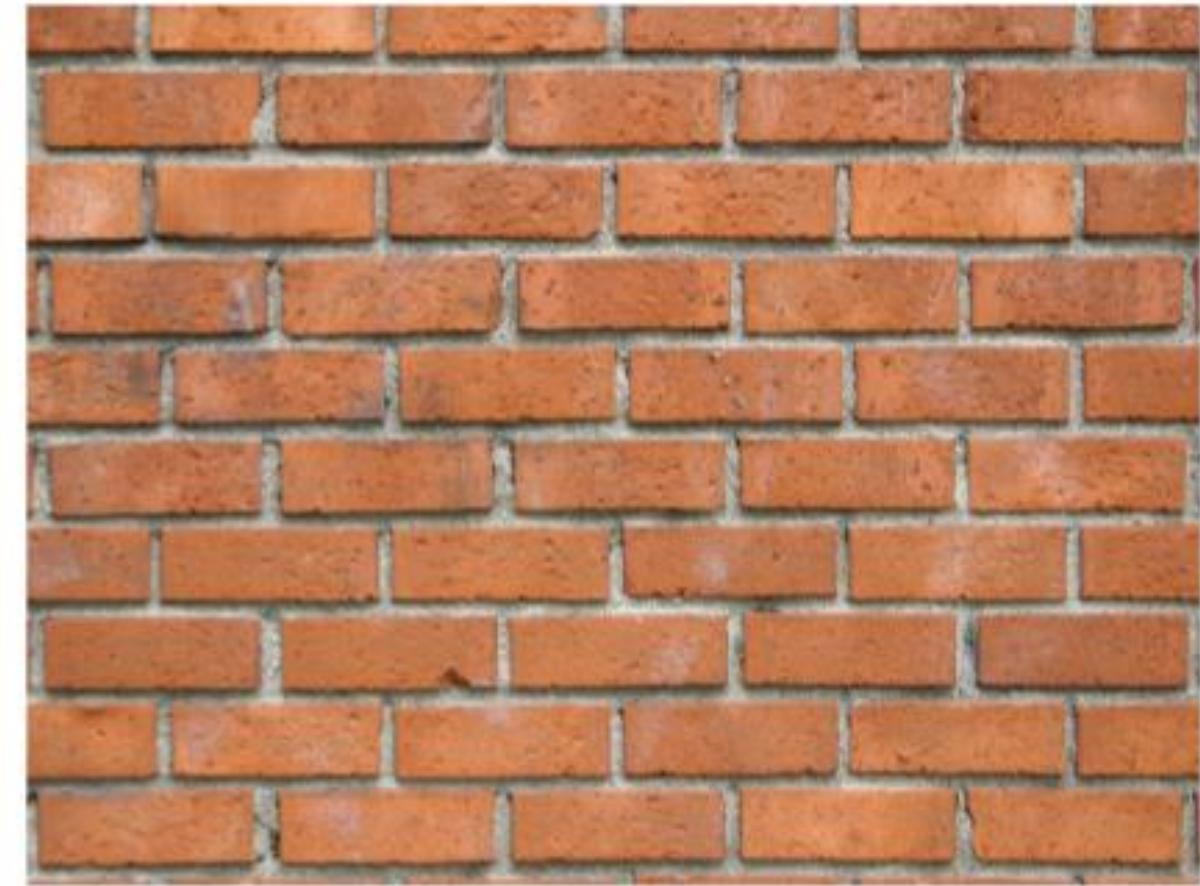
1. Практически в каждом приложении реализована база данных
2. Почти в каждой вакансии упоминается требование к знанию баз данных и SQL запросов (Structured query language — «язык структурированных запросов»)
3. Базы данных позволяют хранить большие объемы данных
4. Возможность анализ накопленных данных

# Профессии, требующие знания БД

- Программист или Разработчик программного обеспечения
- Разработчик баз данных (Database Developer)
- Администратор баз данных
- Аналитик данных (Data Analyst)
- Инженер машинного обучения (Data Engineer)
- Data Scientist
- Тестировщик, QA инженер (Quality Assurance обеспечение качества)
- Project Manager
- DevOps инженер
- и многие другие

# Что такое «База данных»?

- **База данных (БД)** - Это взаимосвязанная информация (данные) об объектах, которая организована специальным образом и хранится на каком-либо носителе.



# База данных

- **База данных** — совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных. [ГОСТ Р ИСО МЭК ТО 10032-2007: (идентичен ISO/IEC TR 10032:2003)]
- **База данных** — организованная в соответствии с определёнными правилами и поддерживаемая в памяти компьютера совокупность данных, характеризующая актуальное состояние некоторой предметной области и используемая для удовлетворения информационных потребностей пользователей. [Когаловский М. Р. Энциклопедия технологий баз данных. 2002.]
- **База данных** — представленная в объективной форме совокупность самостоятельных материалов, систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины (ЭВМ). [Гражданский кодекс РФ, ст. 1260]

# Базы данных (БД)

- С Базой Данных можно делать несколько основных операций: записать данные, прочитать, отредактировать и удалить.
- Принципиально любые данные можно просто записывать в файл, даже не особенно их структурируя. Но это создаст большие проблемы при их чтении, ведь тогда будет сложно понять, где закончилась одна запись и началась другая, в одинаковом ли формате эти записи и т.д.
- База данных заставляет соблюдать некоторую структуру, чтобы обеспечить:
  - удобство, даваемое унификацией
  - эффективный (быстрый) доступ к записям
  - надёжность

# Базы данных (БД)

- **Основные задачи:**
- Обеспечение хранения в БД всей необходимой информации.
- Обеспечение возможности получения данных по всем необходимым запросам.
- Сокращение избыточности и дублирования данных.
- Обеспечение целостности данных (правильности их содержания): исключение противоречий в содержании данных, исключение их потери и т.д.

# **Система управления базой данных (СУБД)**

- **Система управления базой данных (СУБД)** – это программное обеспечение для работы с БД.
- **Система управления базами данных (СУБД)** – это совокупность программ и языковых средств, предназначенных для управления данными в базе данных, ведения базы данных и обеспечения взаимодействия её с прикладными программами [ГОСТ 20886-85].

# Система управления базой данных (СУБД)

- **Функции СУБД:**

- создавать БД
- редактирование БД
- поиск информации в БД
- выполнение несложных расчетов
- вывод отчетов на печать
- и др.

# | Классификация баз данных

По модели данных

По способу хранения данных

По способу доступа к Бд

# Классификация БД - По модели данных

- **Основные типы баз данных:**
- **табличные БД**  
данные в виде одной таблицы
- **сетевые БД**  
набор узлов, в котором каждый может быть связан с каждым.
- **иерархические БД**  
в виде многоуровневой структуры
- **реляционные БД (99,9%)**  
набор взаимосвязанных таблиц

# Простейшие типы баз данных

- К таким базам данных относятся БД, где хранятся данные с простой структурой: например, список разрешенных IP-адресов для доступа к сети, настройки окружения проекта, список подписчиков на рассылку компании и прочее. Они все еще широко распространены.

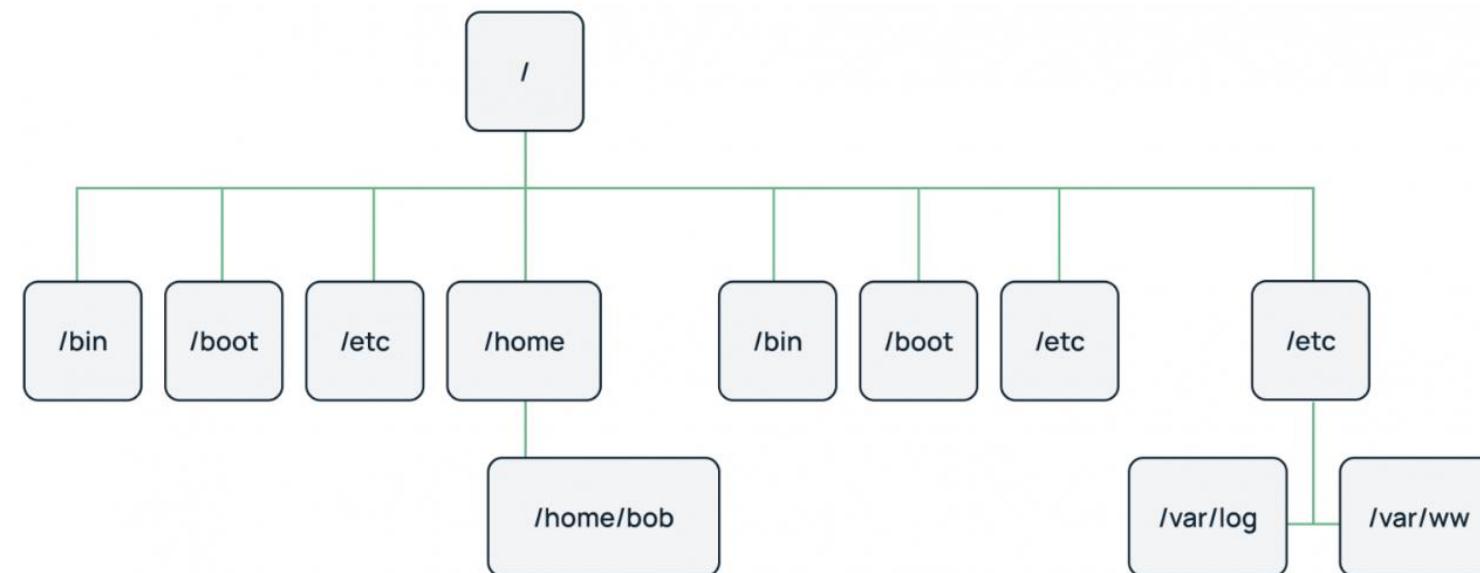
# Текстовые файлы

- Информация об объектах собирается в простых по структуре файлах различных форматов – txt, csv и др. Для разделения полей применяются пробелы, табуляция, запятые, точка с запятой и двоеточие.
- Примеры:** etc/passwd и etc/fstab в Unix-подобных системах, csv-файлы, ini-файлы и др.
- Особенности:**
  - Просто использовать. Для работы с файлами достаточно примитивного текстового редактора.
  - Удобно работать с конфигурационными данными приложений (учетные данные, настройки подключения к удаленным серверам и устройствам, порты и пр.).
- Ограничения:**
  - Сложно установить связи между компонентами данных.
  - Не для всех типов информации.

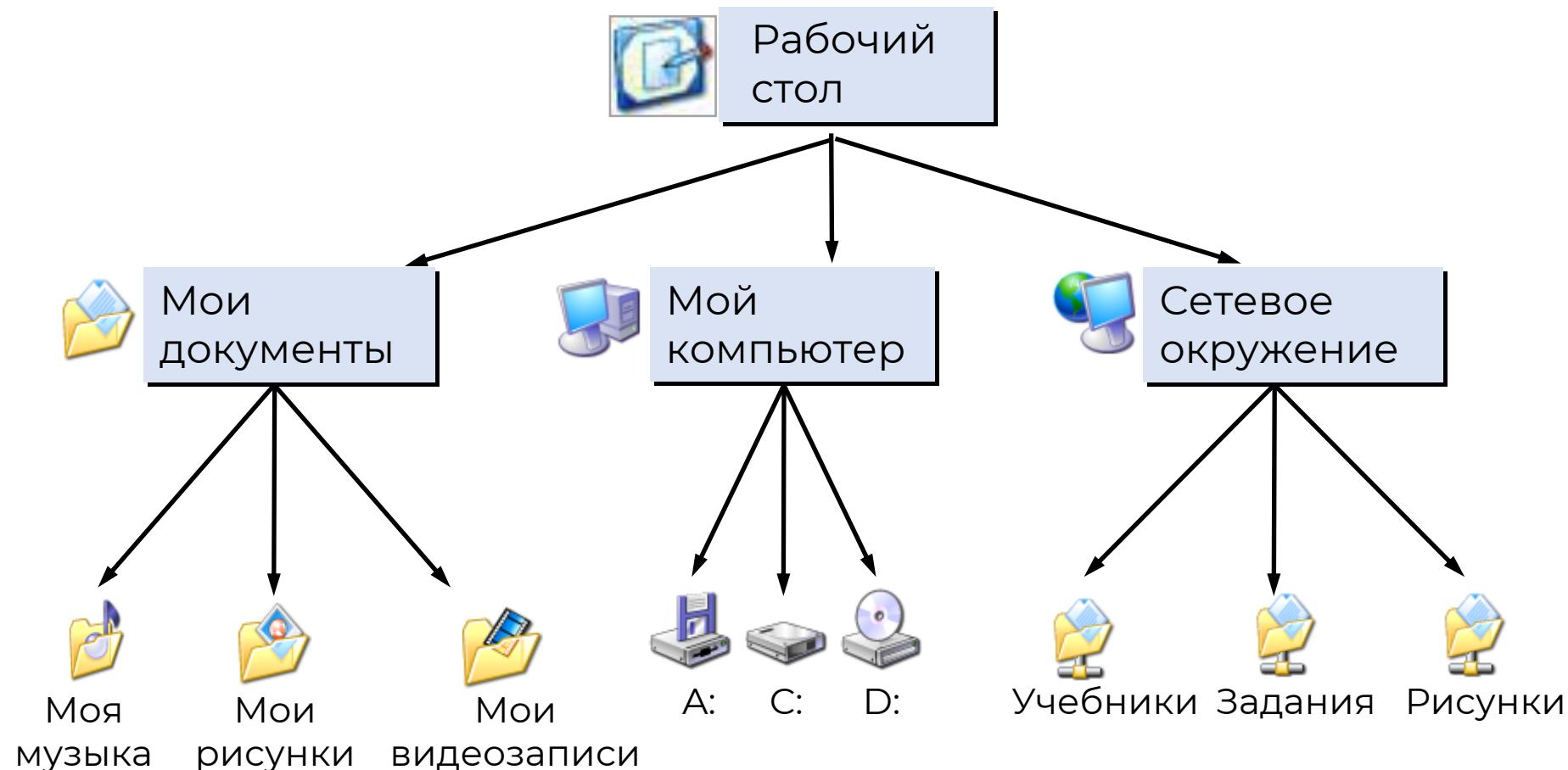
```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
```

# Иерархические базы данных

- В отличие от текстовых файлов здесь между хранимыми объектами устанавливаются связи. Объекты делятся на родителей (основные классы или категории объектов) и потомков (экземпляры этих классов или категорий). При этом у каждого потомка может быть не более одного родителя.
- Графическим представлением такой базы данных является древовидная структура.

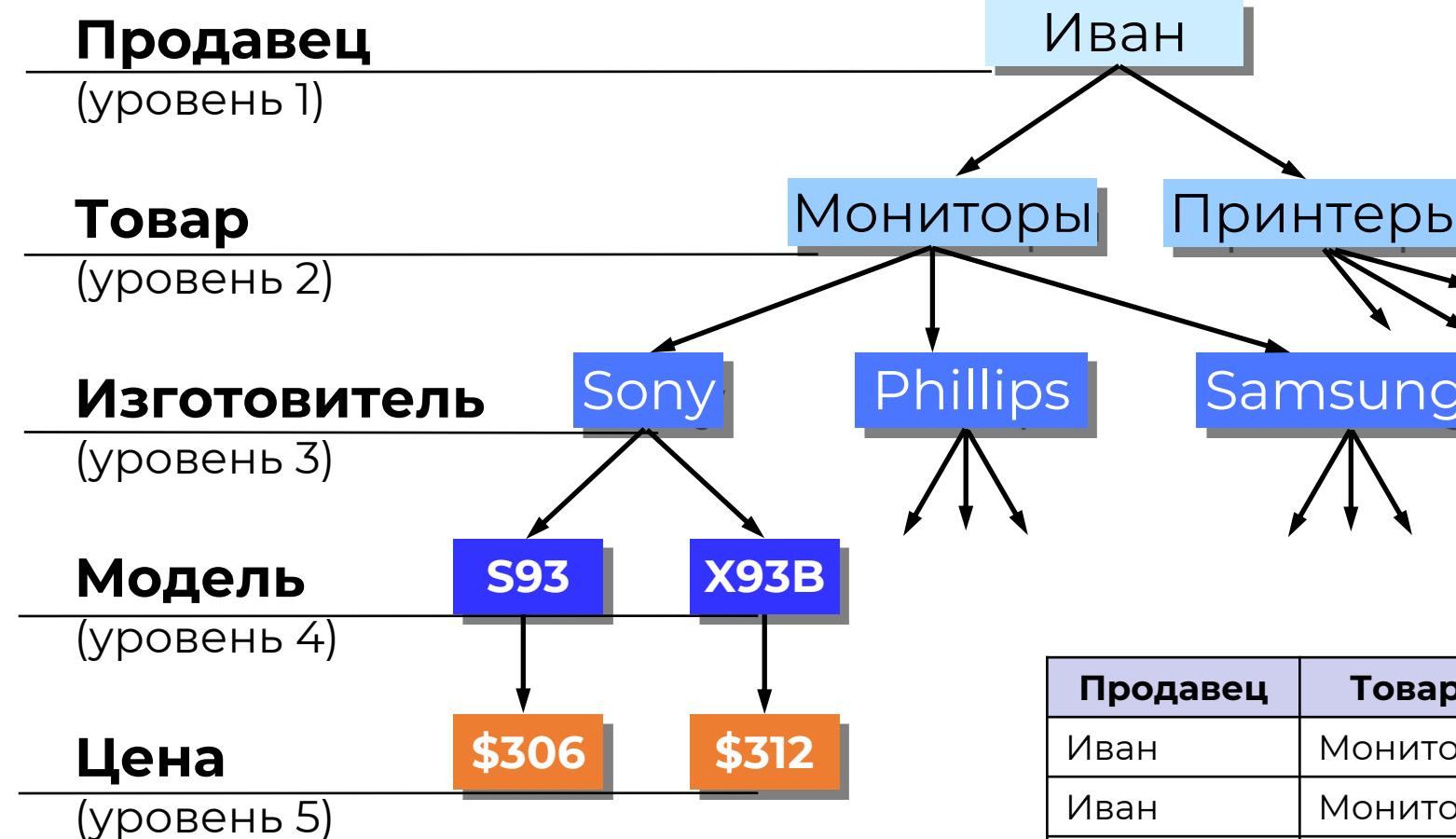


# Иерархические базы данных



# Иерархические базы данных

Прайс-лист:



Приведение к табличной форме:

Продавец	Товар	Изготовитель	Модель	Цена
Иван	Монитор	Sony	S93	\$306
Иван	Монитор	Sony	X93B	\$312
Иван	Монитор	Phillips	190 B5 CG	\$318
Иван	Монитор	Samsung	SyncMaster 193P	\$452
...				

# Иерархические базы данных

- **Примеры:** Организация файловых систем; DNS и LDAP-соединения.
- **Особенности:**
  - Отношения между объектами реализованы в виде физических указателей. Например, в файловой системе путь к папке или файлу строится из имен корневых и вложенных каталогов;
  - Моделирование отношений вложенности и подчиненности.
  - **Ограничения:** Технология иерархической организации не предполагает связи «многие-ко-многим», а значит, система хранения данных довольно ограничена.

# Сетевые базы данных

- Эта технология развивает иерархический подход за счет моделирования сложных отношений между объектами. Здесь потомки могут иметь более одного родителя, однако ограничения иерархического подхода сохраняются.
- Пример: IDMS — специализированная СУБД для мейнфреймов.



# Реляционные базы данных

- Самый популярный и простой тип баз данных — реляционные базы данных.
- Каждая такая база данных представляет из себя набор табличек.
- **Таблица — это и есть та структура, ограничивающая формат данных.** Идея таблицы в том, что каждая запись представляет из себя одну строку, при этом набор колонок фиксирован. Таким образом в каждой строке приведён один и тот же набор данных, ни больше, ни меньше.
- Например, в профиле пользователя всегда есть: имя, фамилия, дата рождения, информация о себе, email, и URL-адрес аватарки. Даже если информация о себе или аватар отсутствуют, в таблице они будут приведены в виде пустой ячейки. Структура таблиц и связей между ними называется схемой (schema).

# Реляционная модель данных

Целое	Строка		Целое		Типы данных
номер	имя	должность	деньги		Домены
Табельный номер	Имя	Должность	Оклад	Премия	Атрибуты
2934	Иванов	Инженер	112	40	
2935	Петров	Вед. Инженер	144	50	
2936	Сидоров	Бухгалтер	92	35	

Отношение

Ключ

Типы данных

Домены

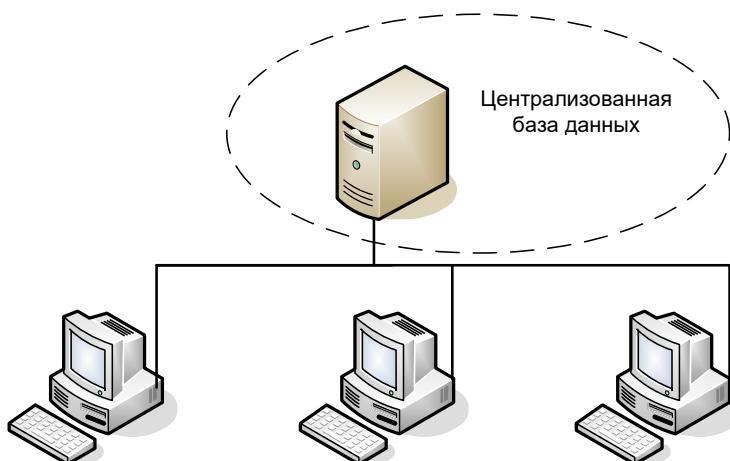
Атрибуты

Кортежи

# Классификация БД - по способу хранения данных

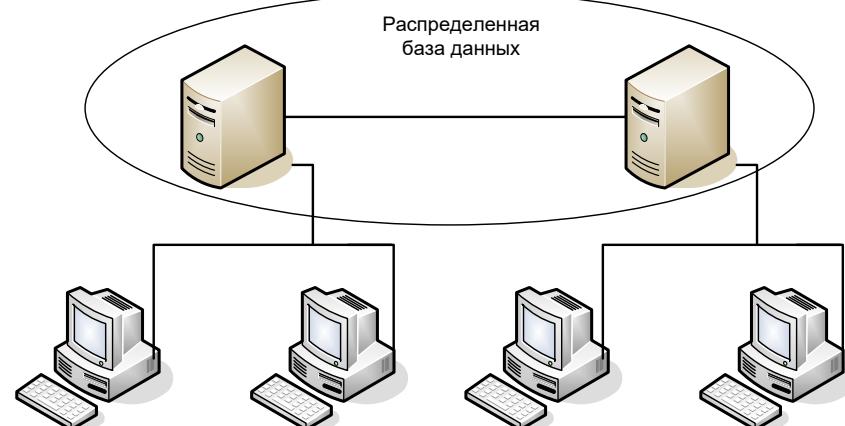
## Централизованные

- БД хранится на одном сервере



## Распределенные

- составные части единой БД хранятся на нескольких серверах, объединенных в сеть



# Классификация БД - по способу доступа к БД

## Базы данных

### Локальные

БД, СУБД и клиентские программы установлены на рабочей станции (PC)

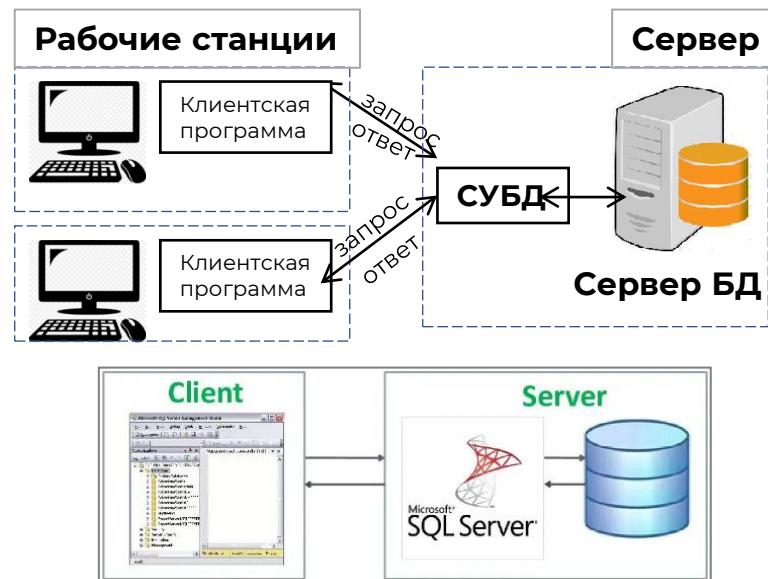
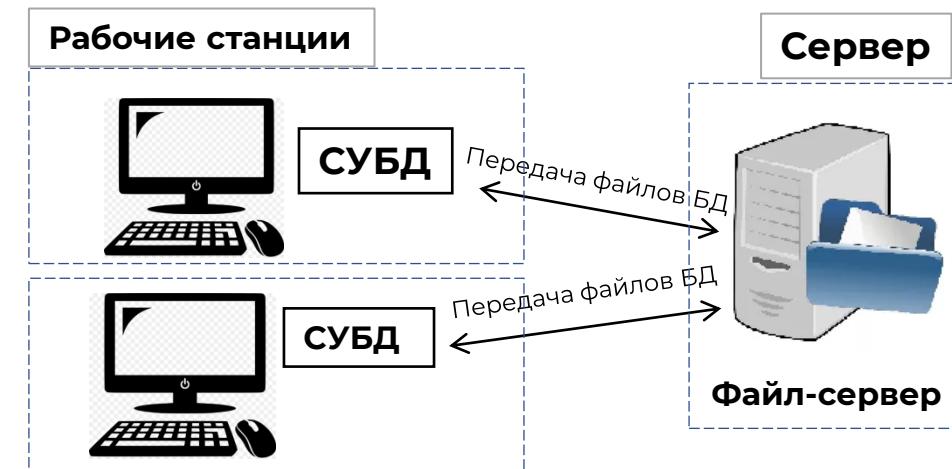
### Удаленные (сетевые)

#### Файл-серверные

БД находится на сервере сети (файловом сервере), а СУБД и клиентские программы на рабочей станции

#### Клиент-серверные

БД и СУБД находятся на сервере (сервер БД), а клиентские программы на рабочих станциях. С рабочей станции (клиента) отправляются запросы на сервер (используется специальный язык запросов SQL), полученные результаты выводятся на экране рабочей станции (клиенте)





# Примеры баз данных



# Устаревшие БД

- dBase
- FoxPro
- Paradox
- Clipper
- и др.



Paradox®

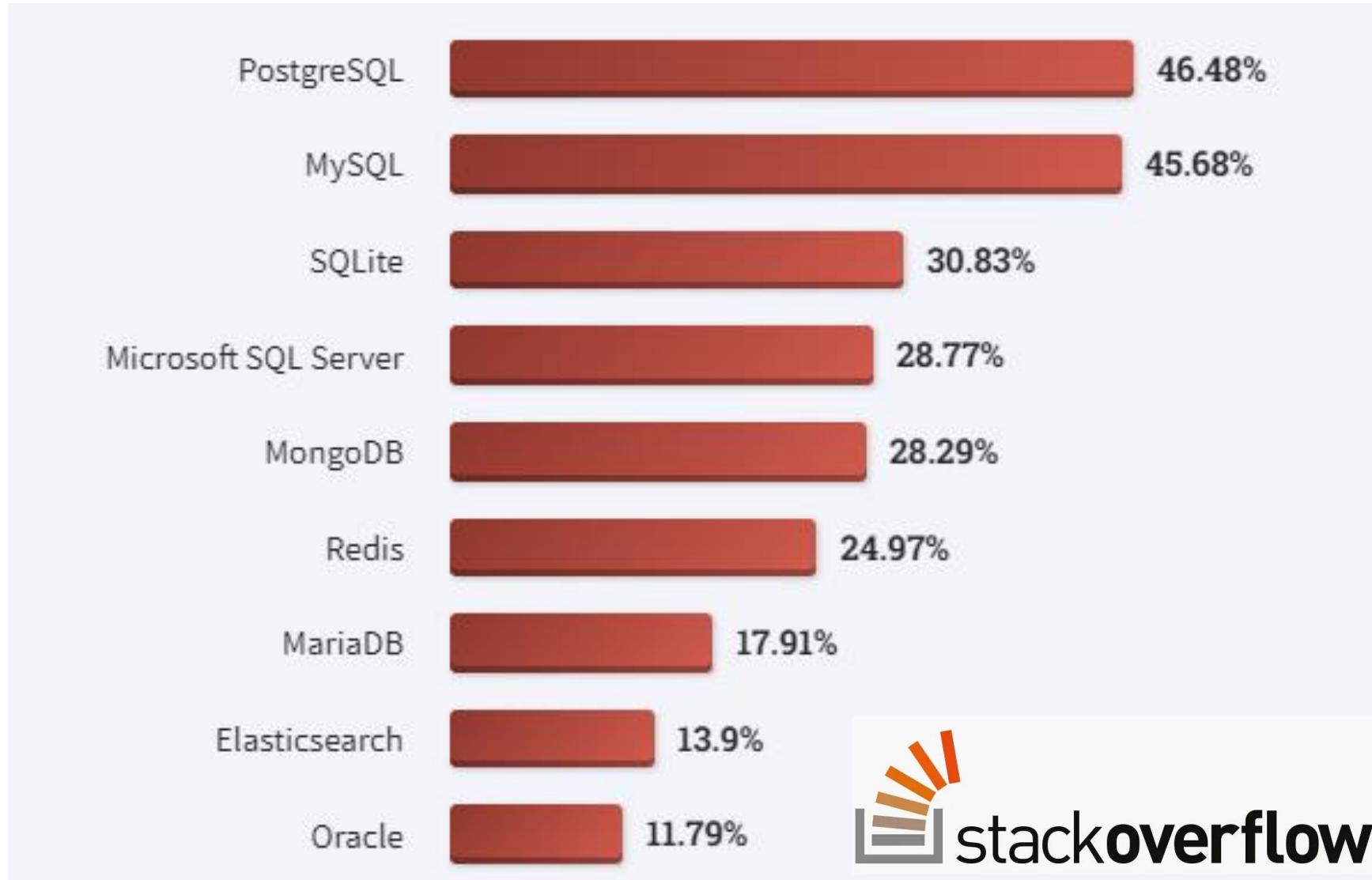


Microsoft®  
**Visual FoxPro**

**CLIPPER**

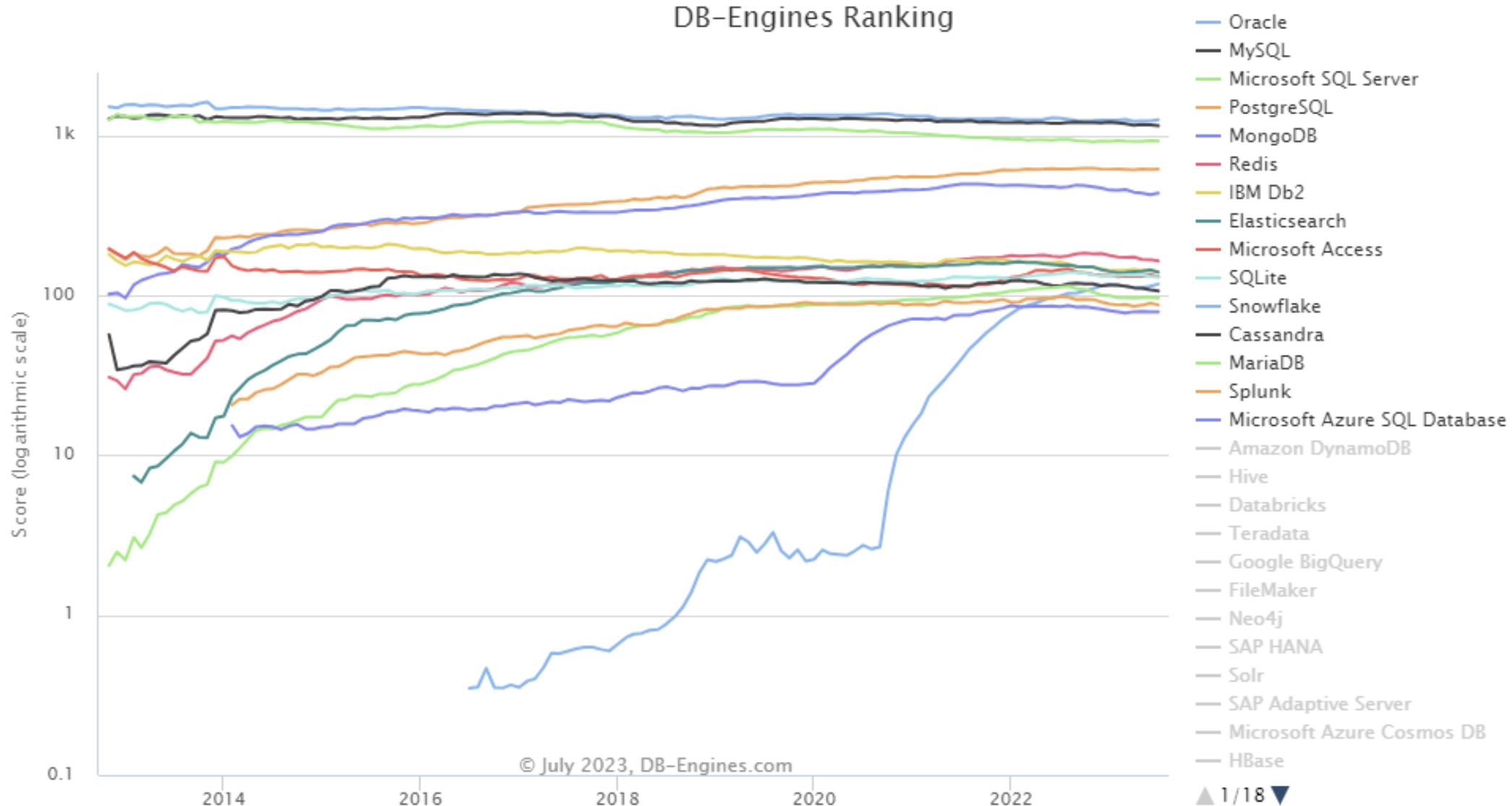
**dBASE™**

# | Наиболее популярные БД в 2022 году



<https://survey.stackoverflow.co/2022/#most-popular-technologies-database-profile>

# DB-Engines Ranking - Trend Popularity 2023



[https://db-engines.com/en/ranking\\_trend](https://db-engines.com/en/ranking_trend)

# DB-Engines Ranking - Trend Popularity 2023

419 systems in ranking, July 2023

Rank			DBMS	Database Model	Score		
Jul 2023	Jun 2023	Jul 2022			Jul 2023	Jun 2023	Jul 2022
1.	1.	1.	Oracle	Relational, Multi-model	1256.01	+24.54	-24.28
2.	2.	2.	MySQL	Relational, Multi-model	1150.35	-13.59	-44.53
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	921.60	-8.47	-20.53
4.	4.	4.	PostgreSQL	Relational, Multi-model	617.83	+5.01	+1.96
5.	5.	5.	MongoDB	Document, Multi-model	435.49	+10.13	-37.49
6.	6.	6.	Redis	Key-value, Multi-model	163.76	-3.59	-9.86
7.	7.	7.	IBM Db2	Relational, Multi-model	139.81	-5.07	-21.40
8.	8.	8.	Elasticsearch	Search engine, Multi-model	139.59	-4.16	-14.74
9.	9.	9.	Microsoft Access	Relational	130.72	-3.73	-14.37
10.	10.	10.	SQLite	Relational	130.20	-1.02	-6.48
11.	11.	13.	Snowflake	Relational	117.69	+3.55	+18.53
12.	12.	11.	Cassandra	Wide column	106.53	-2.03	-7.88
13.	13.	12.	MariaDB	Relational, Multi-model	96.10	-1.21	-16.42
14.	14.	14.	Splunk	Search engine	87.12	-2.34	-11.09
15.	16.	15.	Microsoft Azure SQL Database	Relational, Multi-model	78.96	-0.01	-5.94
16.	15.	16.	Amazon DynamoDB	Multi-model	78.81	-1.10	-5.13

# DB-Engines Ranking of Relational DBMS 2023

include secondary database models

168 systems in ranking, July 2023

Rank				Database Model	Score		
	Jul 2023	Jun 2023	Jul 2022		Jul 2023	Jun 2023	Jul 2022
1.	1.	1.	1.	Oracle	Relational, Multi-model	1256.01	+24.54 -24.28
2.	2.	2.	2.	MySQL	Relational, Multi-model	1150.35	-13.59 -44.53
3.	3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	921.60	-8.47 -20.53
4.	4.	4.	4.	PostgreSQL	Relational, Multi-model	617.83	+5.01 +1.96
5.	5.	5.	5.	IBM Db2	Relational, Multi-model	139.81	-5.07 -21.40
6.	6.	6.	6.	Microsoft Access	Relational	130.72	-3.73 -14.37
7.	7.	7.	7.	SQLite	Relational	130.20	-1.02 -6.48
8.	8.	↑ 9.	9.	Snowflake	Relational	117.69	+3.55 +18.53
9.	9.	↓ 8.	8.	MariaDB	Relational, Multi-model	96.10	-1.21 -16.42
10.	10.	10.	10.	Microsoft Azure SQL Database	Relational, Multi-model	78.96	-0.01 -5.94
11.	11.	11.	11.	Hive	Relational	72.87	-2.65 -6.61
12.	12.	↑ 14.	14.	Databricks	Multi-model	68.47	+2.65 +17.25
13.	13.	↓ 12.	12.	Teradata	Relational, Multi-model	60.25	-2.39 -10.67
14.	14.	↑ 16.	16.	Google BigQuery	Relational	55.42	+0.78 +6.53
15.	15.	15.	15.	FileMaker	Relational	53.32	-1.06 +2.12
16.	16.	↓ 13.	13.	SAP HANA	Relational, Multi-model	50.72	-0.69 -3.77
17.	17.	17.	17.	SAP Adaptive Server	Relational, Multi-model	42.87	-0.88 -2.58
18.	↑ 19.	↑ 22.	22.	Microsoft Azure Synapse Analytics	Relational	27.62	+2.69 +6.00
19.	↓ 18.	↓ 18.	18.	Firebird	Relational	26.16	-0.33 +0.59

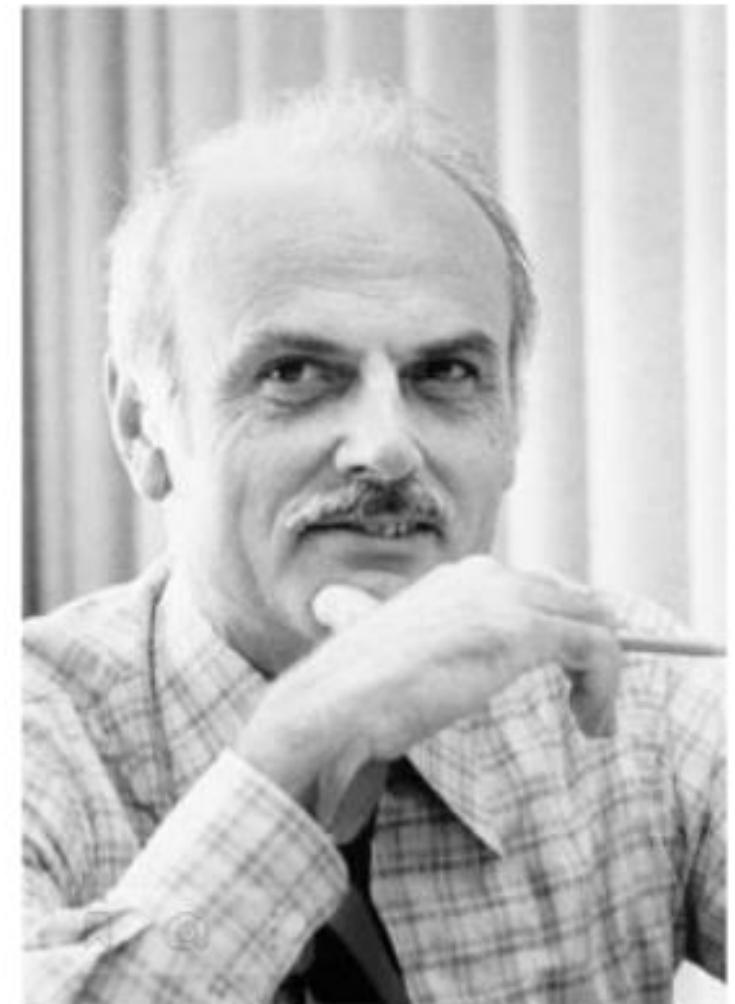
<https://db-engines.com/en/ranking/relational+dbms>



# Реляционные базы данных

# Реляционный подход к организации Бд

- Принято считать, что реляционный подход к организации баз данных был заложен в конце 1960-х гг. Эдгаром Коддом.
- В последние десятилетия этот подход является наиболее распространенным (с оговоркой, что в называемых в обиходе реляционными системах баз данных, основанных на языке SQL, в действительности нарушаются некоторые важные принципы классического реляционного подхода).



Эдгар Франк «Тед» Кодд

# Реляционный подход к организации Бд

- **Достоинствами реляционного подхода принято считать следующие свойства:**
- реляционный подход основывается на небольшом числе интуитивно понятных абстракций, на основе которых возможно простое моделирование наиболее распространенных предметных областей;
- эти абстракции могут быть точно и формально определены;
- теоретическим базисом реляционного подхода к организации баз данных служит простой и мощный математический аппарат теории множеств и математической логики;
- реляционный подход обеспечивает возможность ненавигационного манипулирования данными без необходимости знания конкретной физической организации баз данных во внешней памяти.

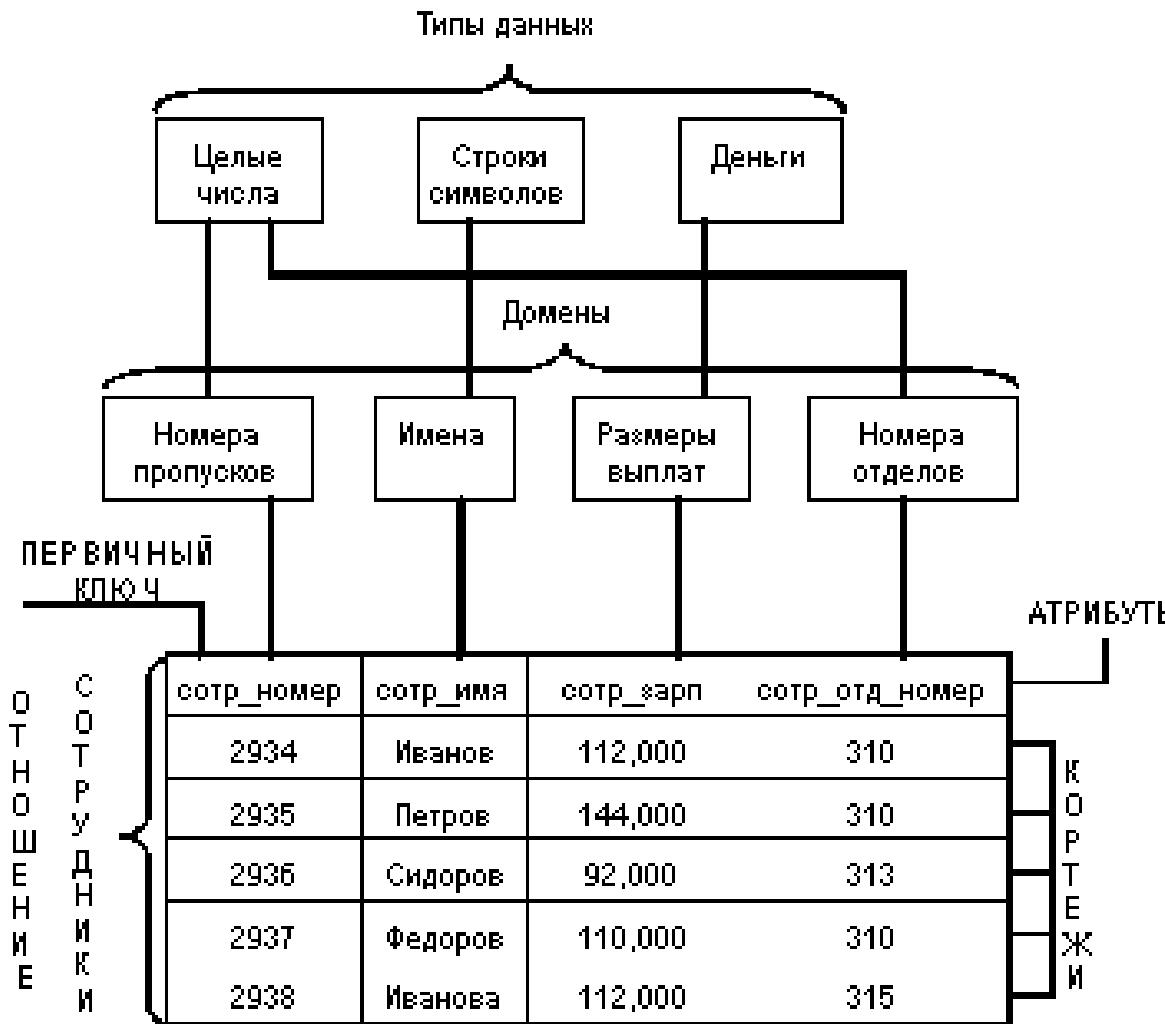
# Реляционная модель данных

- **Реляционная модель данных (РМД)**
- Логическая модель данных, прикладная теория построения баз данных, которая является приложением к задачам обработки данных таких разделов математики как теории множеств и логика первого порядка.
- **Реляционная модель данных включает следующие компоненты:**
- **Структурный аспект (составляющая)** - Данные в базе воспринимаются пользователем, как таблицы (и никак иначе).
- **Аспект (составляющая) целостности** - Отношения (таблицы) отвечают определенным условиям целостности. РМД поддерживает декларативные ограничения целостности уровня домена (типа данных), уровня отношения и уровня базы данных.
- **Аспект (составляющая) обработки (манипулирования)** - РМД поддерживает операторы манипулирования отношениями (реляционная алгебра, реляционное исчисление).

# Реляционная модель данных

- **Реляционная модель данных** – данные представлены посредством строк в таблицах.
- В теории баз данных эти таблицы называют **отношениями (relations)** – поэтому и базы данных называются **реляционными**. Отношение – это математический термин.
- При определении свойств таких отношений используется теория множеств.
- В терминах данной теории строки таблицы будут называться **кортежами (tuples)**, а колонки – **атрибутами**.
- Отношение имеет заголовок, который состоит из атрибутов, и тело, состоящее из кортежей. Количество атрибутов называется **степенью отношения**, а количество кортежей – **кардинальным числом**.

# Соотношение основных понятий реляционного подхода



## Основные понятия реляционной модели

- 1) отношение (таблица);
- 2) схема отношения (таблицы);
- 2) атрибут;
- 3) первичный ключ;
- 4) кортеж;
- 5) тип данных;
- 6) домен (domain).

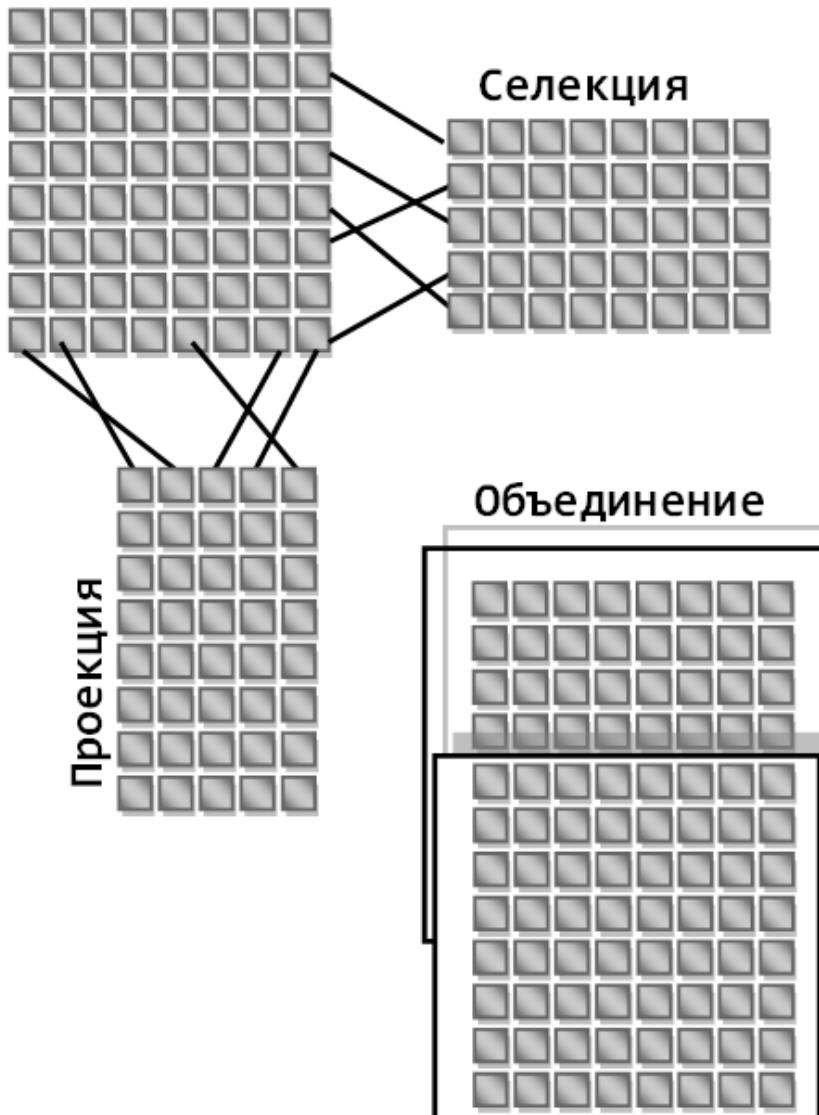
# Терминология

- **Домен** - тип данных, то есть допустимое множество значений.
- **Кортеж** - множество пар {имя атрибута, значение}, которое содержит одно вхождение каждого имени атрибута, принадлежащего схеме отношения.
- **Отношение** - множество кортежей (не упорядоченное).
- **Целостность базы данных** - соответствие имеющейся в базе данных информации её внутренней логике, структуре и всем явно заданным правилам.

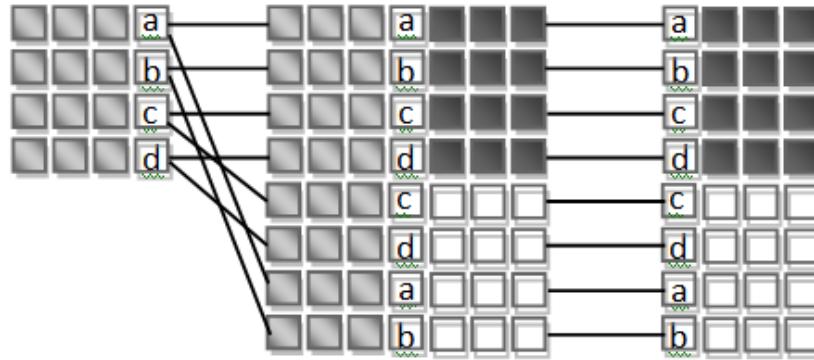
Формальные термины	Альтернатива 1	Альтернатива 2
Relation (отношение)	Table (таблица)	File (файл)
Tuple (кортеж)	Row (строка)	Record (запись)
Attribute (атрибут)	Column (столбец)	Field (поле)

# Реляционная алгебра

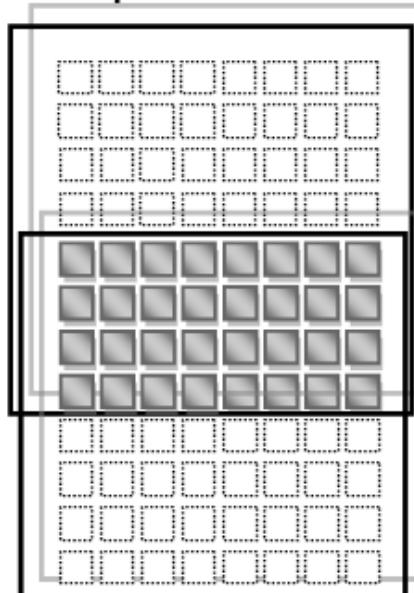
- Выборка
- Проекция
- Объединение
- Пересечение
- Разность
- Произведение
- Деление
- Соединение



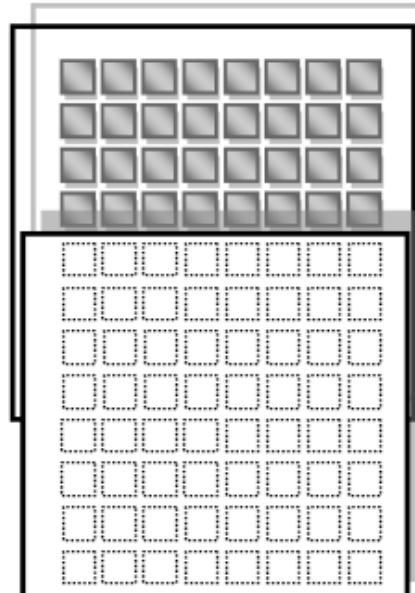
Естественное соединение



Пересечение



Разность



# Реляционная алгебра – теоретический базис SQL



**SQL** (Structured Query Language — «язык структурированных запросов») — формальный непроцедурный язык программирования, применяемый для создания, модификации и управления данными в реляционной БД, управляемой СУБД.

**Реляционная алгебра** — это язык операций, выполняемых над отношениями — таблицами реляционной базы данных. Операции реляционной алгебры позволяют на основе одного или нескольких отношений создавать другое отношение без изменения самих исходных отношений.

**Теория множеств** — раздел математики, в котором изучаются общие свойства множеств — совокупностей элементов произвольной природы, обладающих каким-либо общим свойством.

# Реляционная алгебра

- **Основная идея** реляционной алгебры состоит в том, что коль скоро отношения являются множествами, средства манипулирования отношениями могут базироваться на традиционных теоретико-множественных операциях, дополненных некоторыми специальными операциями, специфичными для реляционных баз данных.
- Существует много подходов к определению реляционной алгебры, которые различаются наборами операций и способами их интерпретации, но, в принципе, являются более или менее равносильными.
- Набор основных алгебраических операций состоит из восьми операций, которые делятся на два класса – теоретико-множественные операции и специальные реляционные операции.

# Общая интерпретация реляционных операций

- Если не вдаваться в некоторые тонкости, которые мы рассмотрим в следующих разделах, то почти для всех операций предложенного выше набора имеется очевидная и простая интерпретация.
  - **При выполнении операции объединения ( UNION )** двух отношений с одинаковыми заголовками производится отношение, включающее все кортежи, которые входят хотя бы в одно из отношений-операндов.
  - **Операция пересечения ( INTERSECT )** двух отношений с одинаковыми заголовками производит отношение, включающее все кортежи, которые входят в оба отношения-операнда.
  - **Отношение, являющееся разностью ( MINUS )** двух отношений с одинаковыми заголовками, включает все кортежи, входящие в отношение-первый operand, такие, что ни один из них не входит в отношение, которое является вторым operandом.
  - **При выполнении декартова произведения ( TIMES )** двух отношений, пересечение заголовков которых пусто, производится отношение, кортежи которого производятся путем объединения кортежей первого и второго operandов.
  - **Результатом ограничения ( WHERE )** отношения по некоторому условию является отношение, включающее кортежи отношения-operandса, удовлетворяющее этому условию.

# Общая интерпретация реляционных операций

- Если не вдаваться в некоторые тонкости, которые мы рассмотрим в следующих разделах, то почти для всех операций предложенного выше набора имеется очевидная и простая интерпретация.
  - **При выполнении проекции ( PROJECT )** отношения на заданное подмножество множества его атрибутов производится отношение, кортежи которого являются соответствующими подмножествами кортежей отношения-операнда.
  - **При соединении ( JOIN )** двух отношений по некоторому условию образуется результирующее отношение, кортежи которого производятся путем объединения кортежей первого и второго отношений и удовлетворяют этому условию.
  - **У операции реляционного деления ( DIVIDE BY )** два операнда – бинарное и унарное отношения. Результирующее отношение состоит из унарных кортежей, включающих значения первого атрибута кортежей первого операнда таких, что множество значений второго атрибута (при фиксированном значении первого атрибута) включает множество значений второго операнда.
  - **Операция переименования ( RENAME )** производит отношение, тело которого совпадает с телом операнда, но имена атрибутов изменены.
  - **Операция присваивания ( := )** позволяет сохранить результат вычисления реляционного выражения в существующем отношении БД.

# Общая интерпретация реляционных операций

- Поскольку результатом любой реляционной операции (кроме операции присваивания, которая не вырабатывает значения) является некое отношение, можно образовывать реляционные выражения, в которых вместо отношения-операнда некоторой реляционной операции находится вложенное реляционное выражение. В построении реляционного выражения могут участвовать все реляционные операции, кроме операции присваивания.
- Вычислительная интерпретация реляционного выражения диктуется установленными приоритетами операций:
- **RENAME >= WHERE = PROJECT >= TIMES = JOIN = INTERSECT = DIVIDE BY >= UNION = MINUS**



# Проектирование базы данных

# Жизненный цикл Базы Данных

## Процедуры, выполняемые на этапах жизненного цикла БД



# Проектирование БД

- Проектирование БД
  - – Концептуальное проектирование

• **Концептуальная (инфологическая) модель** - словесное описание предметной области. Наиболее наглядным является использование специальных графических нотаций (соглашений). Модель строится с использованием стандартных языковых средств, обычно графических, например ER-диаграмм (диаграмм «Сущность-связь»). Модель строится без ориентации на какую-либо конкретную СУБД.

- – Логическое проектирование

• **Логическая (даталогическая) модель** является прототипом создаваемой БД. Все объекты, выделенные при исследовании предметной области и их взаимосвязи, отражаются в структуры типа сущность-связь **с привязкой к конкретному типу БД**.

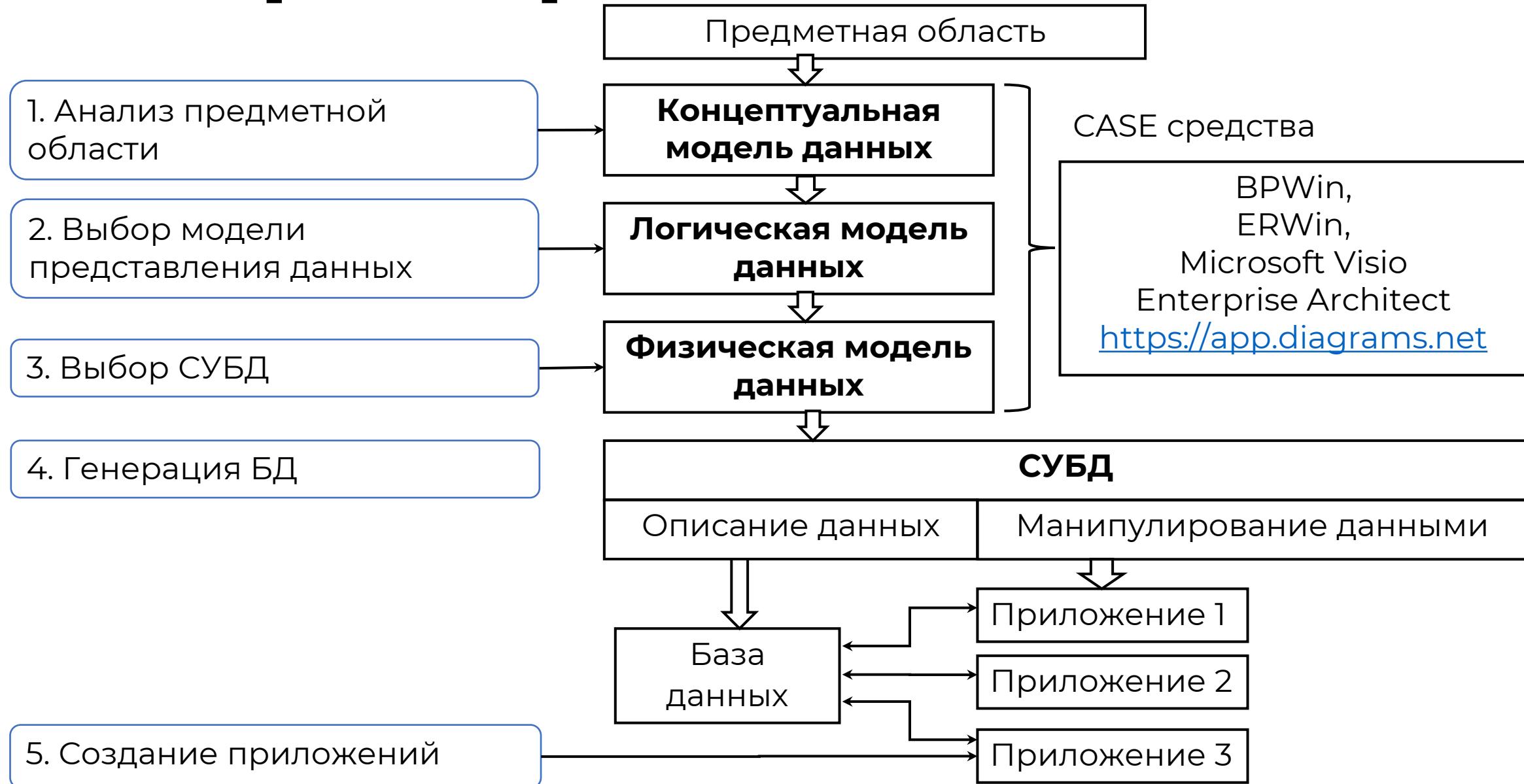
• **Нормализация реляционной БД** – процесс приведения таблиц РБД к строгой форме путем их последовательного преобразования к состоянию, в котором они удовлетворяют условиям первой, второй и третьей нормальных форм. В процессе нормализации происходит последовательное улучшение логической модели данных с тем, чтобы обеспечить ее устойчивость к операциям добавления, удаления и изменения данных.

- – Физическое проектирование

• **Физическая модель** создается с учетом конкретной СУБД и должна учитывать все ее особенности. К таким особенностям могут относиться правила именования таблиц и атрибутов, создание связей между таблицами, поддерживаемые типы данных.



# Этапы проектирования БД



# Этапы проектирования

- **Внешнее представление (внешняя схема) данных**
- С совокупностью требований со стороны конкретной функции, выполняемой пользователем.
- **Концептуальная схема**
- Полная совокупность всех требований к данным, полученной из пользовательских представлений о реальном мире.
- **Внутренняя схема**
- Сама база данных.



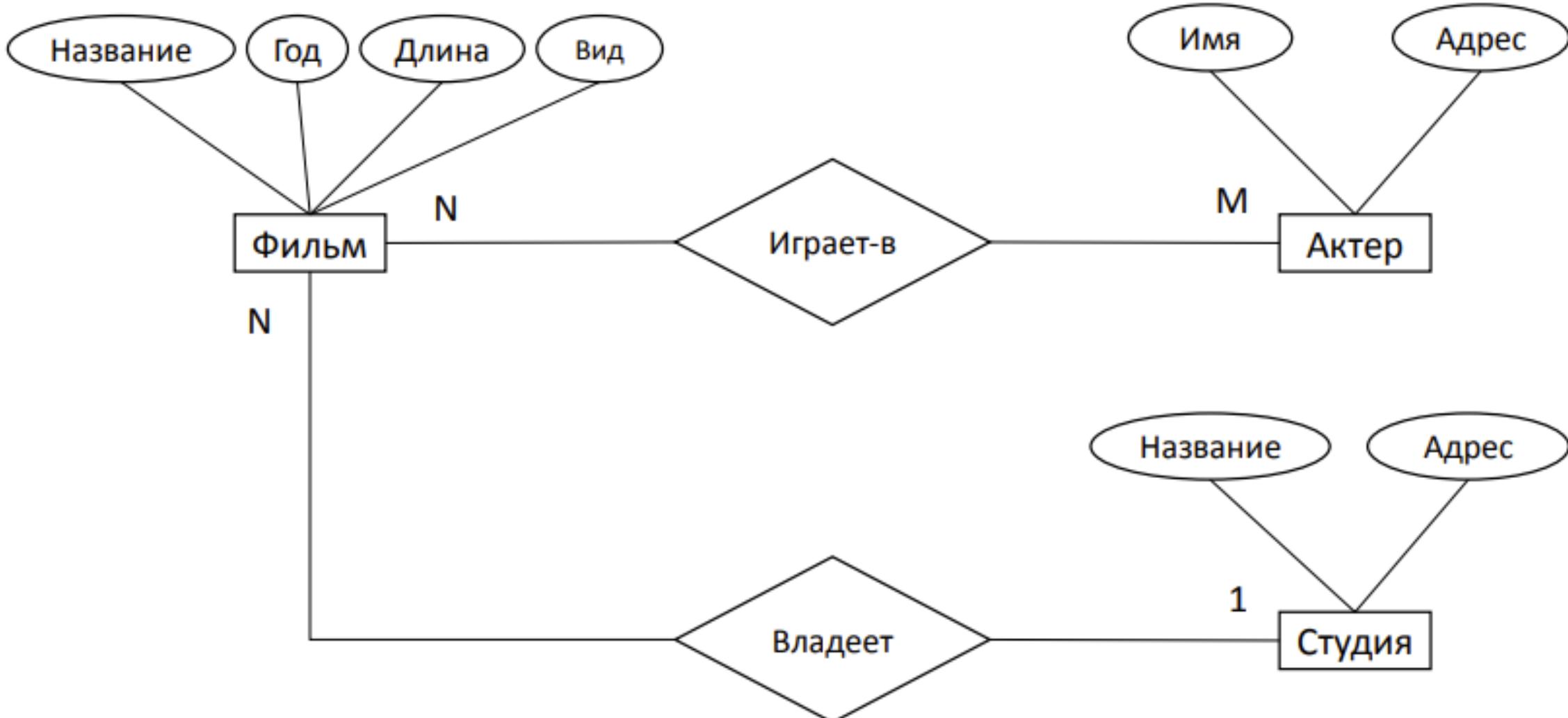
# Концептуальное проектирование БД

- Концептуальное (инфологическое) проектирование
- Построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции.

- сущности
- атрибуты
- связи



# Пример концептуальной модели «сущность-связь»



# Пример концептуальной модели «сущность-связь»



# Концептуальное проектирование БД

- Или инфологическое,
- Семантическое моделирование. Связано со смысловым содержанием данных, независимо от их представления в ЭВМ
- **На этом этапе создаются подробные модели пользовательских представлений данных предметной области.** Затем они интегрируются в концептуальную модель, которая фиксирует все элементы корпоративных данных подлежащих загрузке в БД
- Проектирование сложных баз данных с большим количеством атрибутов осуществляется использованием, так называемого, нисходящего подхода. Этот подход начинается с разработки моделей данных, которые содержат несколько высокоуровневых сущностей и связей, затем работа продолжается в виде серии нисходящих уточнений низкоуровневых сущностей, связей и относящихся к ним атрибутов.
- Нисходящий подход демонстрируется в концепции модели "сущность — связь" (Entity-Relationship model — ER-модель) — самой популярной технологии высокоуровневого моделирования данных, предложенной П. Ченом.

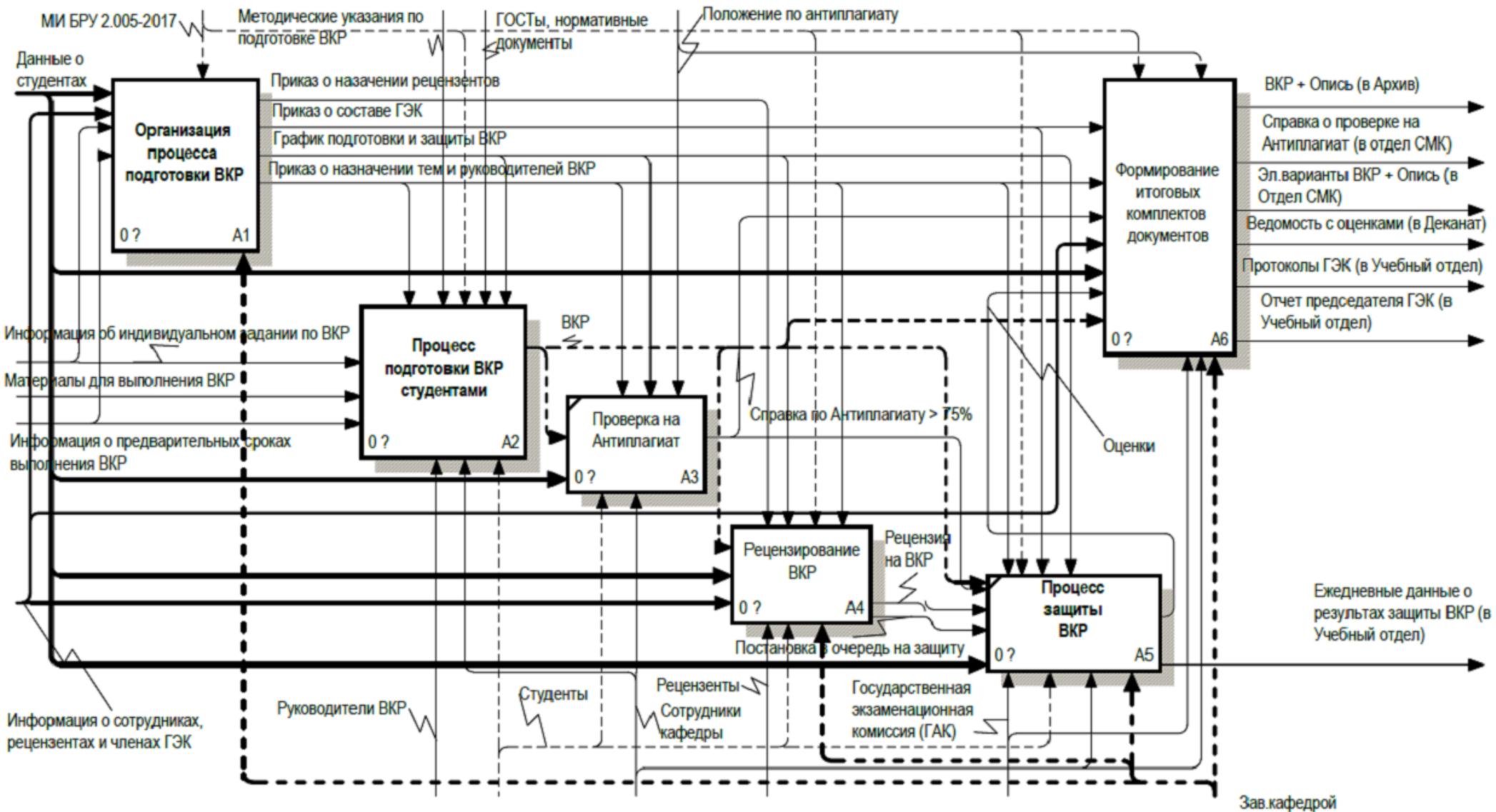
# Концептуальное проектирование БД

- Результатом этого этапа проектирования является построение первичной информационной структуры базы данных, которая называется концептуальной схемой базы данных или инфологической моделью.
- Особенность инфологических моделей
  - 1. Семантическая (смысловая) наполняемость
  - 2. Не зависимость от конкретной СУБД
- Для описания инфологических моделей существует несколько типов такого рода моделей, например:
  - семантическая модель Хаммера – Мак-Леона
  - функциональная модель Шипмана
  - сущностная модель Чена (ER-модель)
  - UML - диаграммы
  - и др.
- На базе этих моделей строятся системы автоматизированного проектирования, так наз. CASE- системы.
  - На базе модели Чена созданы ERWin, POWER DESIGNER и др.
  - На базе модели UML создана Sparx Systems Enterprise Architect, RATIONAL ROSE, PARADIGM PLUS, SELECT ENTERPRISE и др.

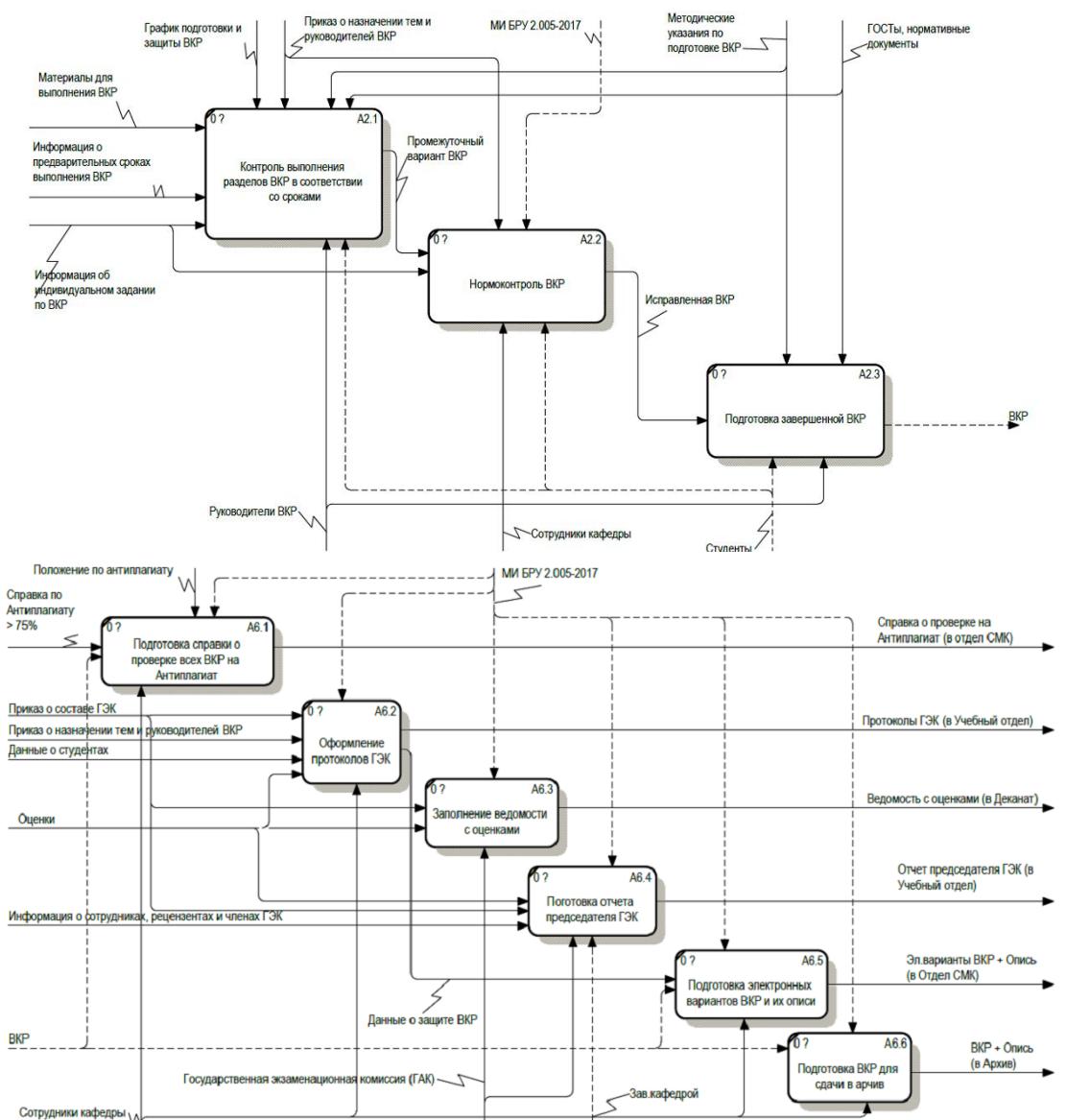
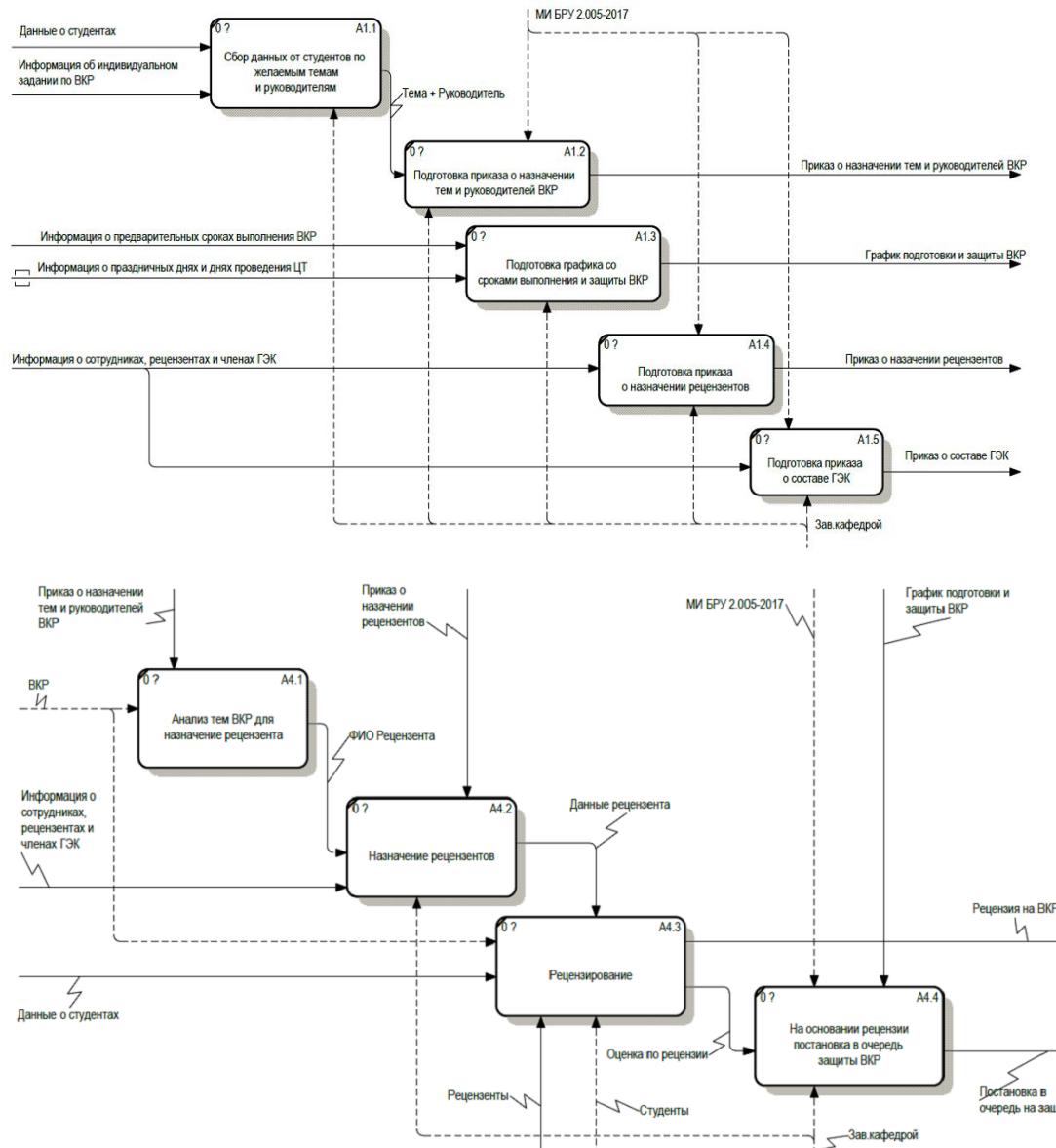
# Пример Контекстной диаграммы (IDEFO)



# Пример декомпозиции Контекстной диаграммы



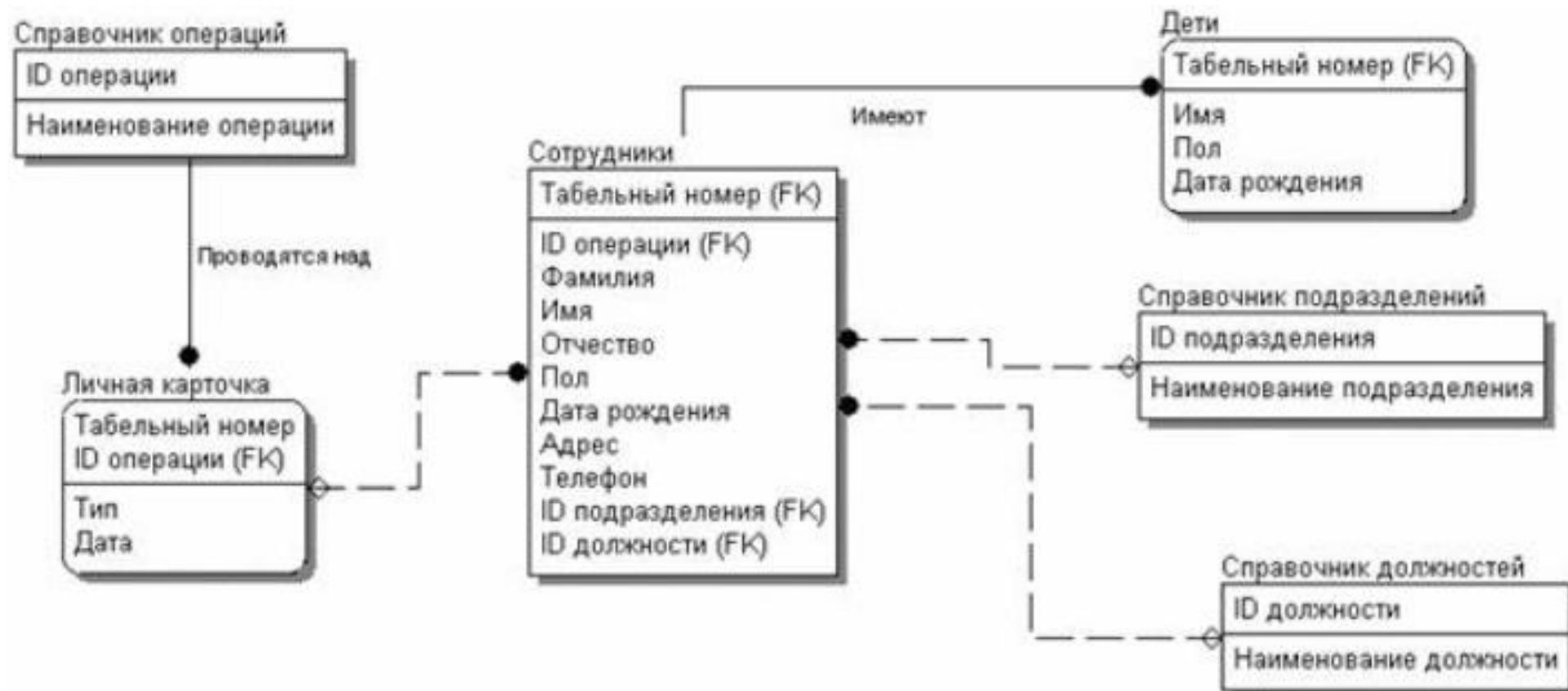
# Пример декомпозиции процессов в нотации DFD



# Логическое проектирование

- **Логическое (даталогическое) проектирование**
- Создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных.

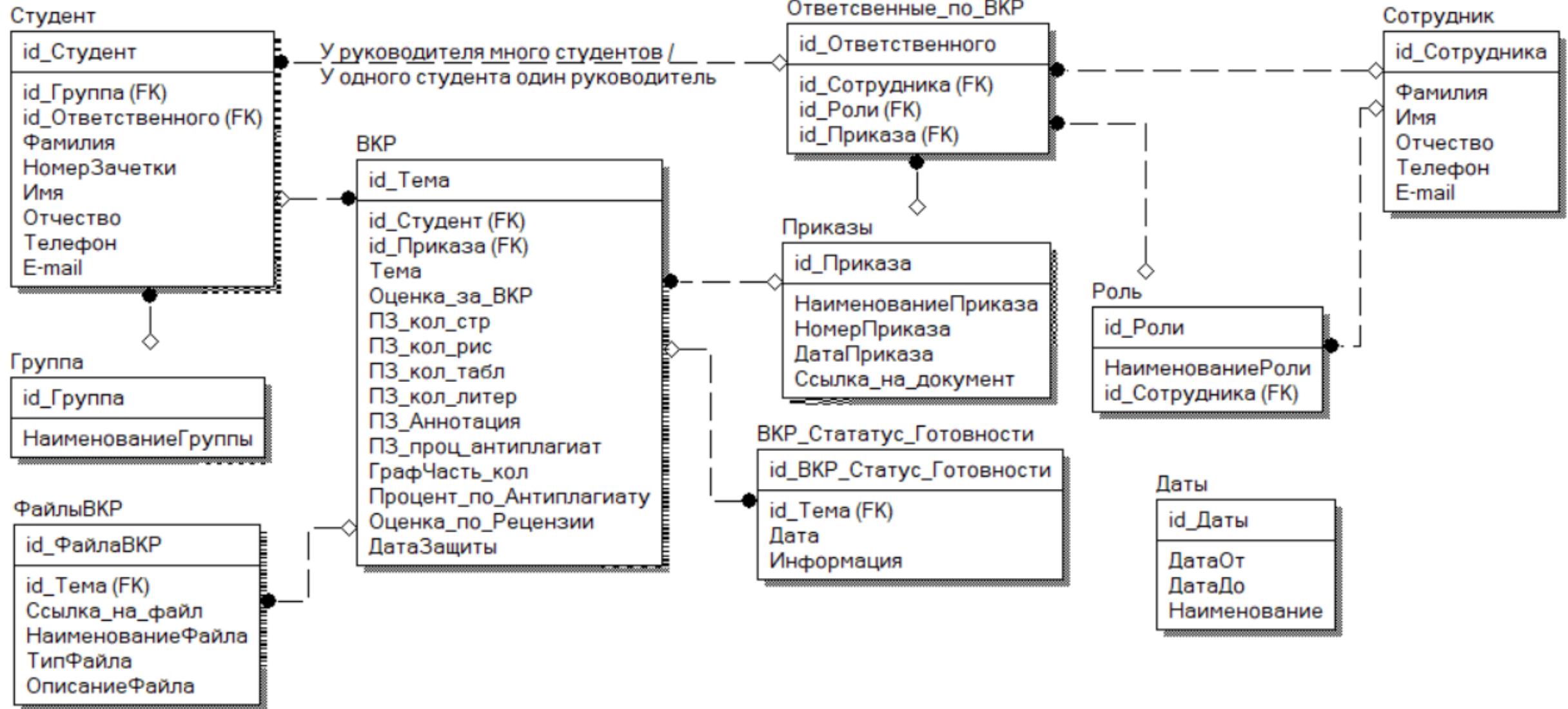
- записи
- элементы данных
- связи между записями



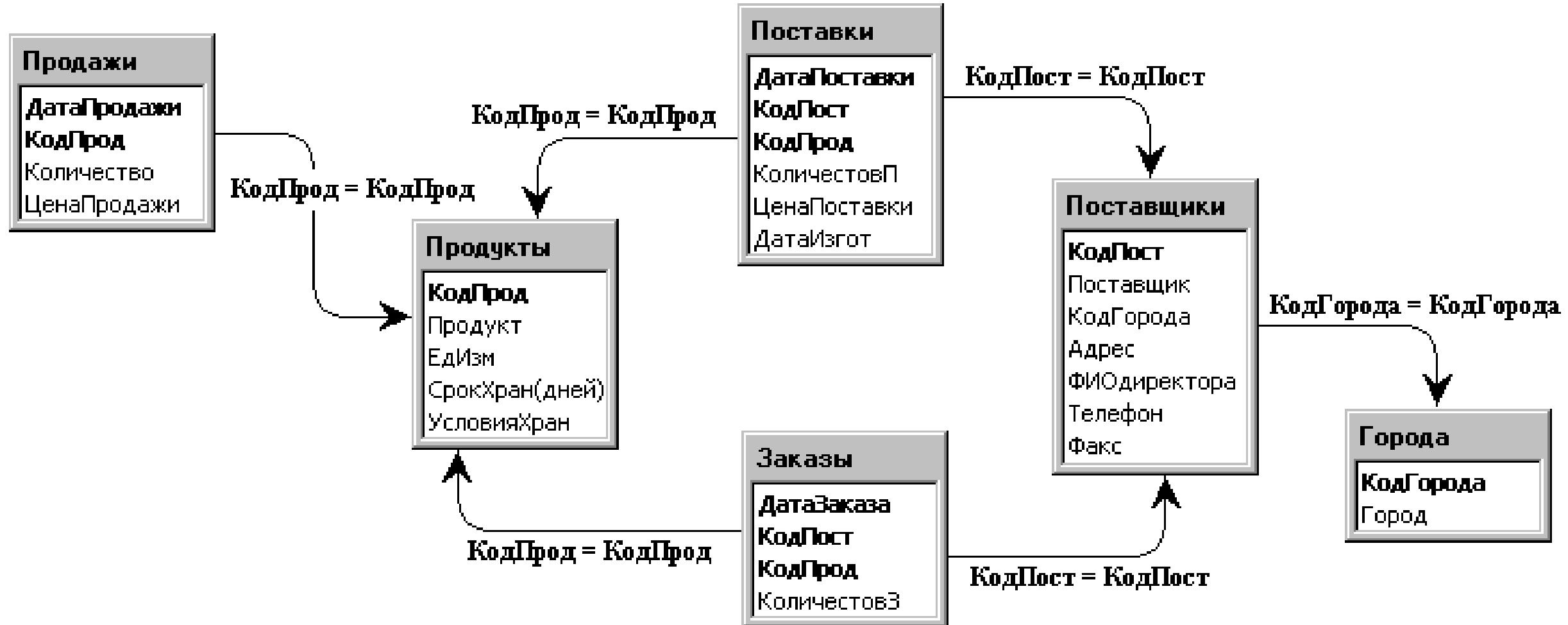
# Преобразование концептуальной модели в логическую

- Создать логическую или концептуальную модель можно с помощью CASE средств:
  - Oracle Designer (Oracle),
  - PowerDesigner (Sybase),
  - AllFusion Data Modeler или **ERwin** (Computer Associates),
  - AllFusion Process Modeler или **BPWin** (Computer Associates),
  - **Visio** Enterprise Edition (MS),
  - Sparx Systems **Enterprise Architect**,
  - и др.

# Пример ER-диаграммы логического уровня



# Пример



# Физическое проектирование

- **Физическое проектирование**

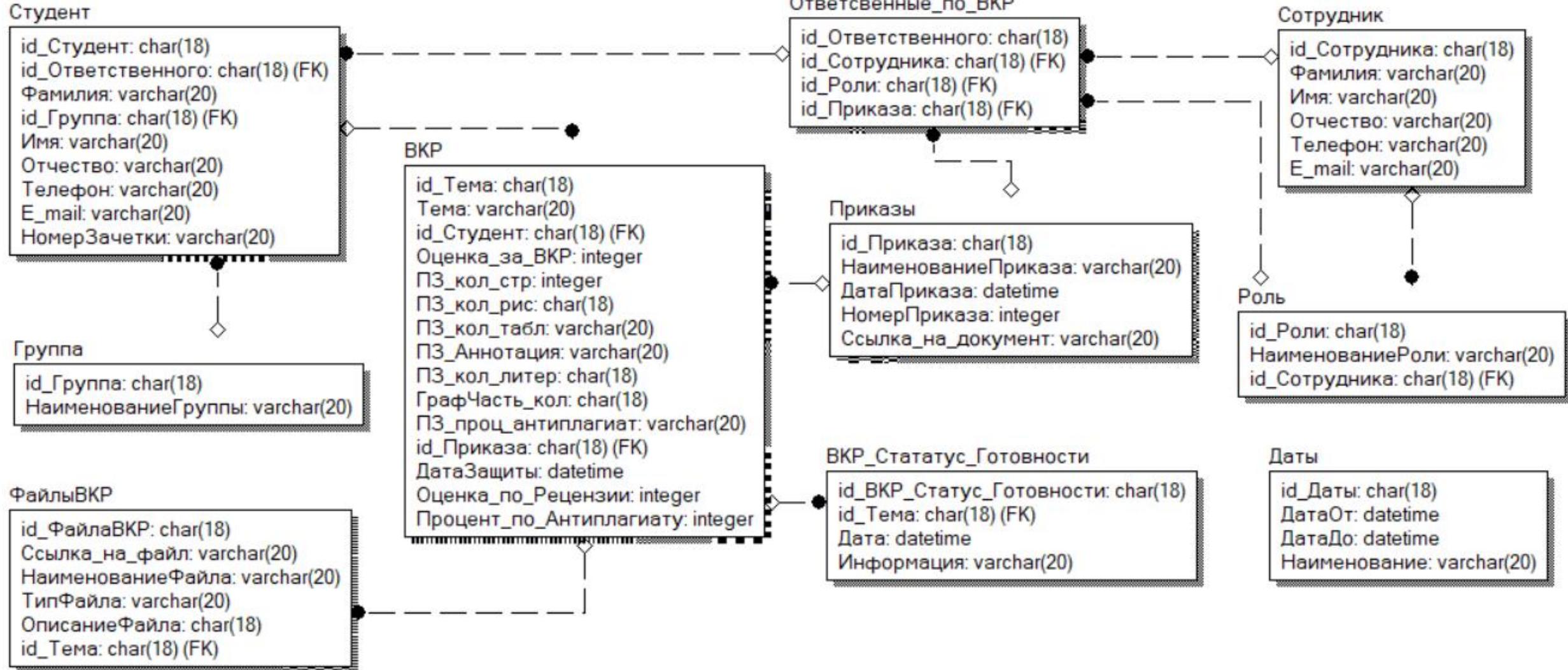
- Создание схемы базы данных для конкретной СУБД.



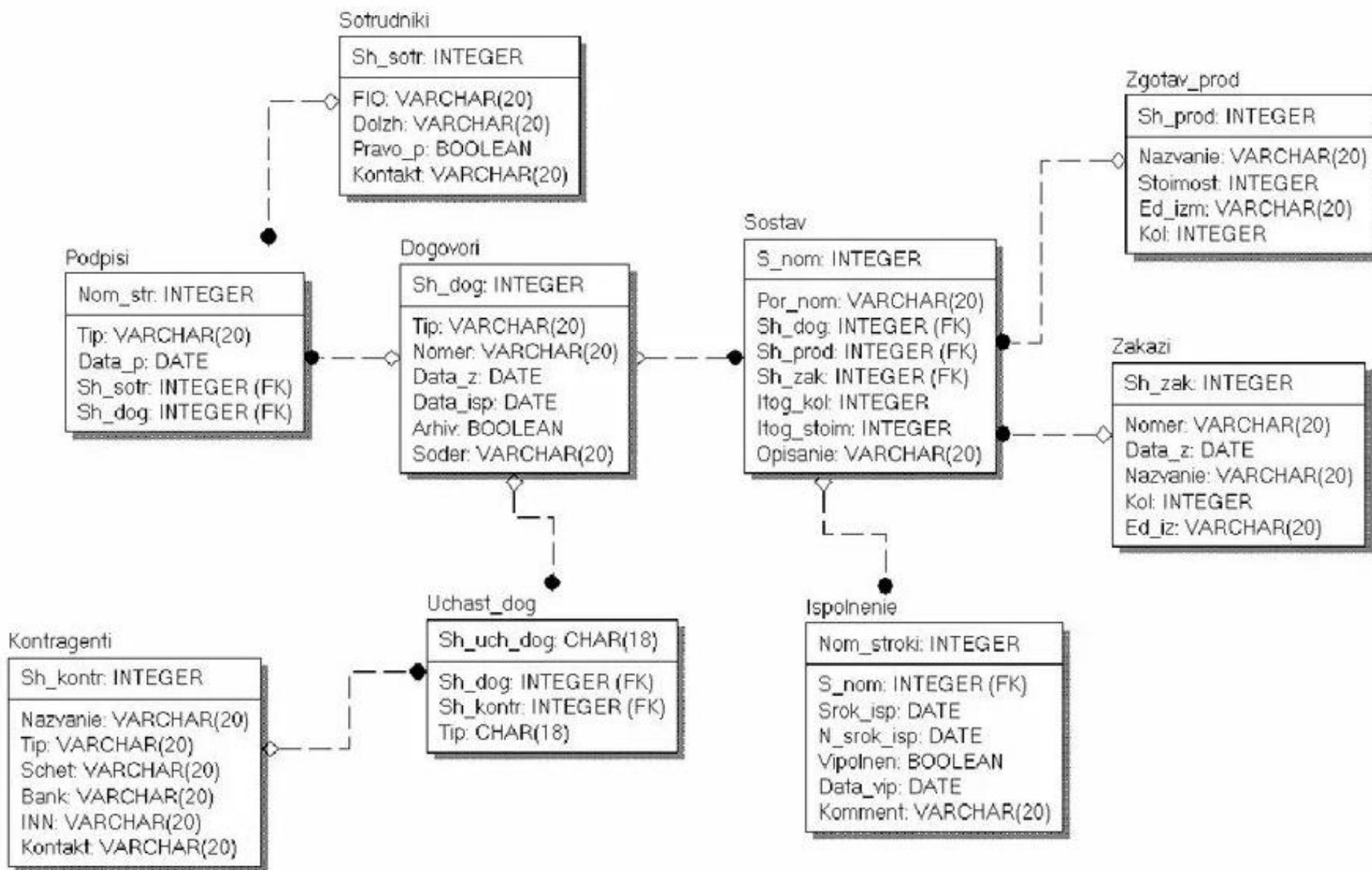
- группирование данных
- индексы
- методы доступа



# Пример ER-диаграммы физического уровня



# Пример физической модели БД



# Почему проект Бд может быть плохим?

Блюдо	Вид	Дата	Продукт	Калорийность	Вес (г)	Поставщик	Город	Страна	Цена (\$)
Лобио	Закуска	01.09.2012	Фасоль	307	200	"Хуанхэ"	Пекин	Китай	0.37
Лобио	Закуска	01.09.2012	Лук	45	40	"Наталка"	Киев	Украина	0.52
Лобио	Закуска	01.09.2012	Масло	742	30	"Лайма"	Рига	Латвия	1.55
Лобио	Закуска	01.09.2012	Зелень	18	10	"Даугава"	Рига	Латвия	0.99
Борщ	Суп	01.09.2012	Мясо	166	80	"Наталка"	Киев	Украина	2.18
Борщ	Суп	01.09.2012	Лук	45	30	"Наталка"	Киев	Украина	0.52
Борщ	Суп	01.09.2012	Томаты	24	40	"Полесье"	Киев	Украина	0.45
Борщ	Суп	01.09.2012	Рис	334	50	"Хуанхэ"	Пекин	Китай	0.44
Борщ	Суп	01.09.2012	Масло	742	15	"Полесье"	Киев	Украина	1.62
Борщ	Суп	01.09.2012	Зелень	18	15	"Наталка"	Киев	Украина	0.88

1. Избыточность
2. Потенциальная противоречивость  
(аномалии обновления)

3. Аномалии включения
4. Аномалии удаления



# Нормализация и денормализация реляционных баз данных

# Пример структуры таблицы

Заголовок	Дата	Автор	E-Mail	Текст	Оценка	Ответы
Руслан и Людмила	01.01.1820	Пушкин А.С.	sukin_syn@mail.ru	...	500	120
Война и Мир	01.03.1869	Толстой Л.Н.	tolstoy@inbox.ru	...	100	345
Отцы и Дети	15.08.1862	Иван Тургенев	nedobobov@mail.ru	...	256	18
Повести Белкина	12.06.1830	Пушкин А.С.	sukin_syn@mail.ru	...	400	96
Муму	10.11.1852	Иван Тургенев	nedobobov@mail.ru	...	312	46

# Первичный ключ

- Первичный ключ выбирается исходя из соображений удобства и сохранения уникальности. Если такого ключа нет, то имеет смысл добавить специальное поле.

TopicID	Заголовок	Дата	Автор	E-Mail	Текст	Оценка	Ответы
5	Руслан и Людмила	01.01.1820	Пушкин А.С.	sukin_syn@mail.ru	...	500	120
8	Война и Мир	01.03.1869	Толстой Л.Н.	tolstoy@inbox.ru	...	100	345
12	Отцы и Дети	15.08.1862	Иван Тургенев	nedobobov@mail.ru	...	256	18
43	Повести Белкина	12.06.1830	Пушкин А.С.	sukin_syn@mail.ru	...	400	96
16	Муму	10.11.1852	Иван Тургенев	nedobobov@mail.ru	...	312	46

# Справочник пользователей

- **Нормальная форма**
- Свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных.
- Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

E-Mail	Пароль				
<input type="text"/>	<input type="password"/>				
Имя пользователя	<input type="text"/>				
<input type="text"/>	дата регистрации	mm	/ dd	/ уууу	<input type="button" value="календарь"/>
Привилегии	Доступные разделы				
<input type="checkbox"/> Администратор	<input type="checkbox"/> Новости				
<input type="checkbox"/> Модератор	<input type="checkbox"/> Юмор				

E-Mail	Пароль	Имя	Дата	Привилегии	Разделы
tolstoy@mail.ru	*****	Толстой Л.Н.	09.09.1828	Админ, Модератор	Новости, Флуд
saltykov@inbox.ru	*****	Николай Щедрин	27.01.1826	Модератор	Флуд, Юмор
fmd@mail.ru	*****	Федор Михайлович	11.11.1821	Игрок	E-Mail
sukin_syn@mail.ru	*****	Пушкин А.С.	06.06.1799	Админ	Поэзия

# Нормализация

- Основная цель нормализации – устранение избыточности данных.
  - Первая нормальная форма (1НФ, 1NF)
  - Вторая нормальная форма (2НФ, 2NF)
  - Третья нормальная форма (3НФ, 3NF)
  - Нормальная форма Бойса — Кодда (НФБК, BCNF)
  - Четвёртая нормальная форма (4НФ, 4NF)
  - Пятая нормальная форма (5НФ, 5NF)
  - Доменно-ключевая нормальная форма (ДКНФ, DKNF).
- Основные свойства нормальных форм состоят в следующем:
  - каждая следующая нормальная форма в некотором смысле лучше предыдущей нормальной формы;
  - при переходе к следующей нормальной форме свойства предыдущих нормальных форм сохраняются.

# 1-ая нормальная форма

- Переменная отношения находится в первой нормальной форме (1НФ) тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов.

E-Mail	Пароль	Имя	Дата	Привилегии	Разделы
tolstoy@mail.ru	*****	Толстой Л.Н.	09.09.1828	Админ, Модератор	Новости, Флуд
saltykov@inbox.ru	*****	Николай Щедрин	27.01.1826	Модератор	Флуд, Юмор
fmd@mail.ru	*****	Федор Михайлович	11.11.1821	Игрок	E-Mail
sukin_syn@mail.ru	*****	Пушкин А.С.	06.06.1799	Админ	Поэзия

**Кортеж** - множество пар {имя атрибута, значение}, которое содержит одно вхождение каждого имени атрибута, принадлежащего схеме отношения. (строка таблицы)

**Атрибут** - свойство некоторой сущности. Часто называется полем таблицы.

# 1-ая нормальная форма

- Переменная отношения находится в первой нормальной форме (1НФ) тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов.

E-Mail	Пароль	Имя	Дата	Привилегии	Разделы
tolstoy@mail.ru	*****	Толстой Л.Н.	09.09.1828	Админ	Новости
tolstoy@mail.ru	*****	Толстой Л.Н.	09.09.1828	Админ	Флуд
tolstoy@mail.ru	*****	Толстой Л.Н.	09.09.1828	Модератор	Новости
tolstoy@mail.ru	*****	Толстой Л.Н.	09.09.1828	Модератор	Флуд
saltykov@inbox.ru	*****	Николай Щедрин	27.01.1826	Модератор	Флуд
saltykov@inbox.ru	*****	Николай Щедрин	27.01.1826	Модератор	Юмор
fmd@mail.ru	*****	Федор Михайлович	11.11.1821	Игрок	E-Mail
sukin_syn@mail.ru	*****	Пушкин А.С.	06.06.1799	Админ	Поэзия

# 2-ая нормальная форма

- Переменная отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме, и каждый неключевой атрибут неприводимо зависит от ее потенциального ключа.

E-Mail	Пароль	Имя	Дата	Привилегии	Разделы
tolstoy@mail.ru	*****	Толстой Л.Н.	09.09.1828	Админ	Новости
tolstoy@mail.ru	*****	Толстой Л.Н.	09.09.1828	Админ	Флуд
tolstoy@mail.ru	*****	Толстой Л.Н.	09.09.1828	Модератор	Новости
tolstoy@mail.ru	*****	Толстой Л.Н.	09.09.1828	Модератор	Флуд
saltykov@inbox.ru	*****	Николай Щедрин	27.01.1826	Модератор	Флуд
saltykov@inbox.ru	*****	Николай Щедрин	27.01.1826	Модератор	Юмор
fmd@mail.ru	*****	Федор Михайлович	11.11.1821	Игрок	E-Mail
sukin_syn@mail.ru	*****	Пушкин А.С.	06.06.1799	Админ	Поэзия

# 2-ая нормальная форма

- Переменная отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме, и каждый неключевой атрибут неприводимо зависит от ее потенциального ключа.

E-Mail	Пароль	Имя	Дата
tolstoy@mail.ru	*****	Толстой Л.Н.	09.09.1828
saltykov@inbox.ru	*****	Николай Щедрин	27.01.1826
fmd@mail.ru	*****	Федор Михайлович	11.11.1821
sukin_syn@mail.ru	*****	Пушкин А.С.	06.06.1799

E-Mail	Привилегии	Разделы
tolstoy@mail.ru	Админ	Новости
tolstoy@mail.ru	Админ	Флуд
tolstoy@mail.ru	Модератор	Новости
tolstoy@mail.ru	Модератор	Флуд
saltykov@inbox.ru	Модератор	Флуд
saltykov@inbox.ru	Модератор	Юмор
fmd@mail.ru	Игрок	E-Mail
sukin_syn@mail.ru	Админ	Поэзия

# 3-я нормальная форма

- Переменная отношения находится в третьей нормальной форме, когда она находится во второй нормальной форме, и отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых.

TopicID	Заголовок	Дата	Автор	E-Mail	Текст	Оценка	Ответы
5	Руслан и Людмила	01.01.1820	Пушкин А.С.	sukin_syn@mail.ru	...	500	120
8	Война и Мир	01.03.1869	Толстой Л.Н.	tolstoy@inbox.ru	...	100	345
12	Отцы и Дети	15.08.1862	Иван Тургенев	nedobobov@mail.ru	...	256	18
43	Повести Белкина	12.06.1830	Пушкин А.С.	sukin_syn@mail.ru	...	400	96
16	Муму	10.11.1852	Иван Тургенев	nedobobov@mail.ru	...	312	46

# 3-я нормальная форма

- Переменная отношения находится в третьей нормальной форме, когда она находится во второй нормальной форме, и отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых.

E-Mail	Автор
sukin_syn@mail.ru	Пушкин А.С.
tolstoy@inbox.ru	Толстой Л.Н.
nedobobov@mail.ru	Иван Тургенев

TopicID	Заголовок	Дата	E-Mail	Текст	Оценка	Ответы
5	Руслан и Людмила	01.01.1820	sukin_syn@mail.ru	...	500	120
8	Война и Мир	01.03.1869	tolstoy@inbox.ru	...	100	345
12	Отцы и Дети	15.08.1862	nedobobov@mail.ru	...	256	18
43	Повести Белкина	12.06.1830	sukin_syn@mail.ru	...	400	96
16	Муму	10.11.1852	nedobobov@mail.ru	...	312	46

# 4-ая нормальная форма

- Переменная отношения находится в четвёртой нормальной форме, если она находится в третьей нормальной форме и не содержит нетривиальных многозначных зависимостей.

E-Mail	Привилегии	Разделы
tolstoy@mail.ru	Админ	Новости
tolstoy@mail.ru	Админ	Флуд
tolstoy@mail.ru	Модератор	Новости
tolstoy@mail.ru	Модератор	Флуд
saltykov@inbox.ru	Модератор	Флуд
saltykov@inbox.ru	Модератор	Юмор
fmd@mail.ru	Игрок	E-Mail
sukin_syn@mail.ru	Админ	Поэзия

# 4-ая нормальная форма

- Переменная отношения находится в четвёртой нормальной форме, если она находится в третьей нормальной форме и не содержит нетривиальных многозначных зависимостей.

E-Mail	Привилегии
tolstoy@mail.ru	Админ
tolstoy@mail.ru	Модератор
saltykov@inbox.ru	Модератор
fmd@mail.ru	Игрок
sukin_syn@mail.ru	Админ

E-Mail	Разделы
tolstoy@mail.ru	Новости
tolstoy@mail.ru	Флуд
saltykov@inbox.ru	Флуд
saltykov@inbox.ru	Юмор
fmd@mail.ru	E-Mail
sukin_syn@mail.ru	Поэзия

# Реляционная часть

- ОДИН-К-ОДНОМУ (1:1)
- ОДИН-КО-МНОГИМ (1:М)
- МНОГИЕ-К-ОДНОМУ (М:1)
- МНОГИЕ-КО-МНОГИМ (М:Н)

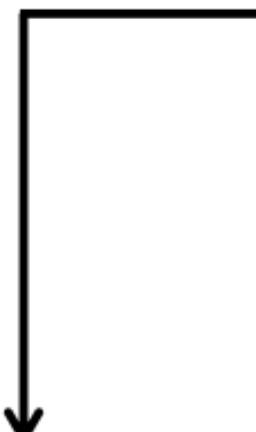


# Примеры связей

## Связь «Один ко многим»

Таблица «Кафедры»

Код кафедры (PK)	Название
100	Кафедра фундаментальной информатики
101	Кафедра систем автоматизированного проектирования
102	Кафедра математического анализа, алгебры и геометрии



Табельный номер (PK)	Код кафедры (FK)	Фамилия
1234	100	Иванов
1237	102	Петров
3456	100	Сидоров

Таблица «Сотрудники»

# Примеры связей

Таблица «Дисциплины»

Код дисциплины (PK)	Название
100	Базы данных
101	ООП
102	Языки программирования

Таблица «Студенты»

Код студента (PK)	Фамилия
200	Антонов
202	Белов
210	Волков

**Связь «Многие ко многим»**

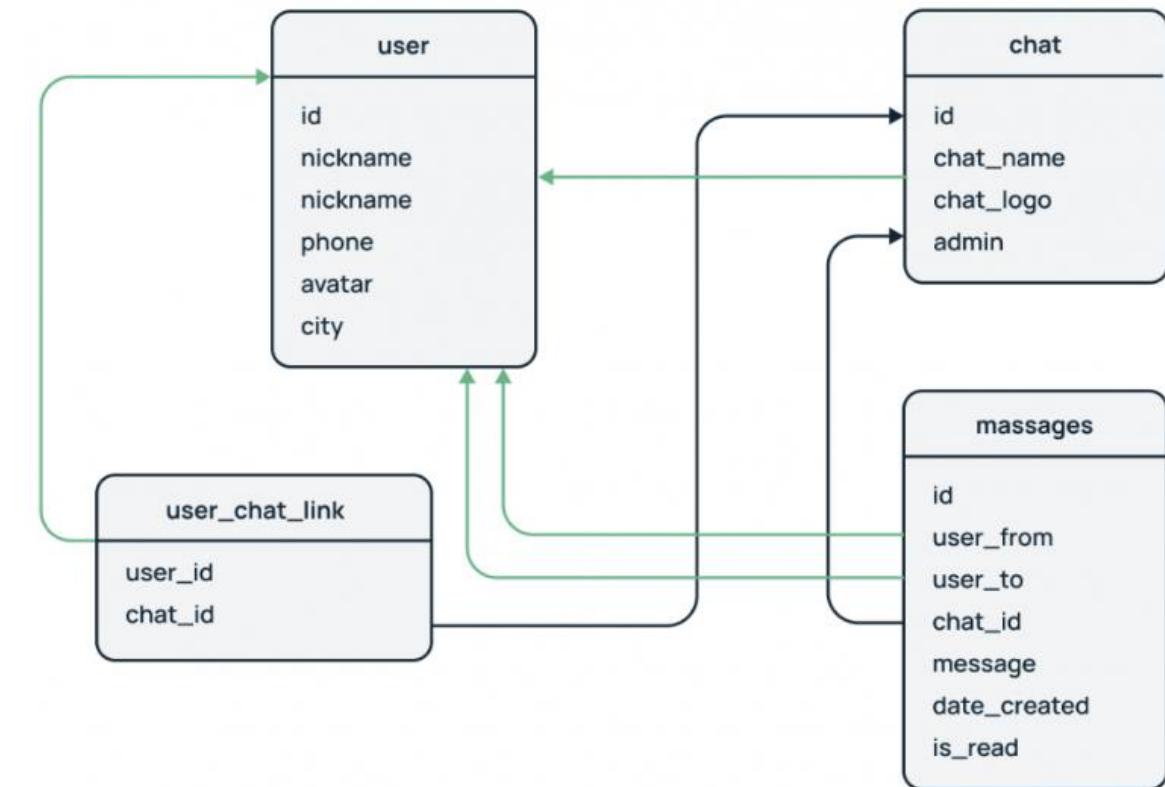
Код дисциплины (FK)	Код студента (FK)
100	200
101	200
100	202

# Денормализация

- Помимо нормализации, в реляционных БД существует и обратный процесс — **дениормализация**.
- Он направлен на перенос наиболее часто используемых полей из внешних таблиц во внутренние.

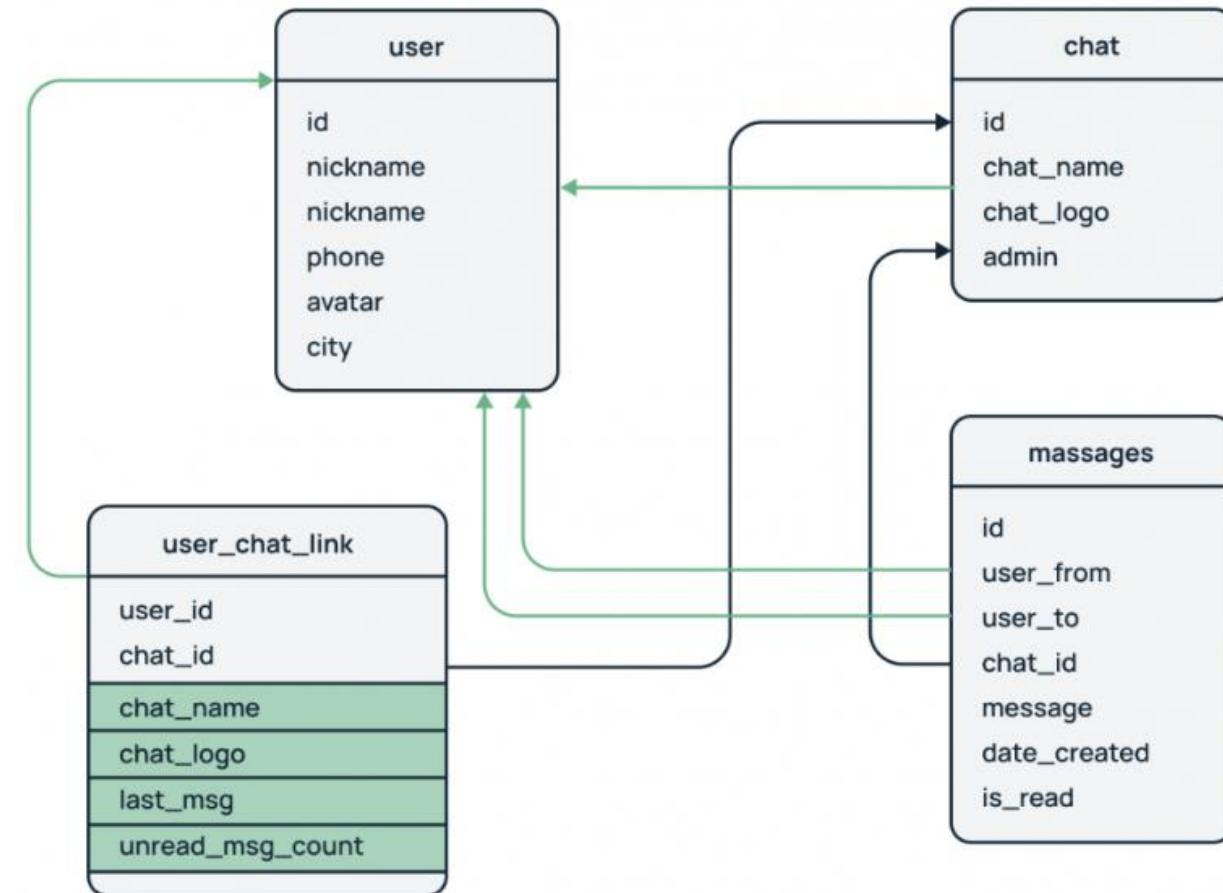
# Денормализация

- Рассмотрим это на примере мессенджера.
- Пользователь (user) оставляет сообщения (messages) в чатах (chat). Структура данных такова, что сообщения связаны с пользователем и чатом через внешние ключи (user\_from и user\_to, а также chat\_id в таблице сообщений; user\_id и chat\_id в таблице user\_chat\_link). Поскольку схема нормализована, то различные запросы на выборку, подсчет и агрегацию статистики по чатам, пользователям и сообщениям необходимо выполнять с помощью присоединения внешних таблиц.
- На относительно небольших объемах данных эти запросы будут отрабатывать быстро, а с увеличением размера базы – замедляться. Причина кроется в механизме присоединения. Он основан на построчном сравнении двух и более таблиц по условию соединения — например, равенство chat\_id в messages и id в chat. А это дает нагрузку на сервер базы данных, которая с ростом ее размера только увеличивается. Для оптимизации такого рода запросов существует механизм денормализации.



# Денормализация

- В таблицу связи пользователя и чата `user_chat_link` добавлены дублирующие поля имени чата (`chat_name`) и аватара (`chat_logo`). Также туда выводятся последнее сообщение (`last_msg`) и количество непрочитанных сообщений (`unread_msg_count`).
- Теперь для получения указанных выше полей и проведения аналитики по ним можно использовать таблицу `user_chat_link` без необходимости соединения с таблицей сообщений. Тем не менее, такой подход имеет ограничения.
- За счет дополнительных полей оптимизируются запросы на чтение и агрегацию данных, однако ценой этого является вынужденная избыточность и усложнение бизнес-логики приложения. В частности, усложняется написание запросов изменения данных (`update` и `delete`), а также модификации структуры базы (`create`).
- Использование денормализации должно быть тщательно осмыслено. Нужно быть уверенным в том, что нормализованная структура, оптимизированные запросы и правильно настроенные индексы более не способны удовлетворять критерию быстродействия.





# Язык SQL кратко

# SQL (Structured Query Language)

- Structured Query Language
- Язык структурированных запросов

SQL

# SQL (Structured Query Language)

- **SQL** – непроцедурный язык взаимодействия приложений и пользователей с реляционными СУБД, состоящий из набора стандартных команд на английском языке.
- Язык запросов, который специально разрабатывался для удобного управления данными.
- SQL может использоваться как интерактивный (для выполнения запросов) и как встроенный (для построения прикладных программ).
- **Базовый вариант SQL содержит порядка 40 команд (часто еще называемых запросами или операторами)** для выполнения различных действий внутри СУБД.
- SQL – язык очень большого объема. Его спецификация содержит свыше 2000 страниц, не считая больше 300 страниц исправлений. В тоже время в стандарте SQL есть противоречия.
- Стандарты языка SQL: SQL-86, SQL-89, SQL-92 (SQL2), SQL:1999 (SQL3), SQL:2003, SQL:2006, SQL:2008, SQL:2011, SQL:2016.

# Стандарт SQL:2023

- В 2023 году Международная организация по стандартизации (ISO) утвердила и опубликовала международный стандарт **SQL:2023 (ISO/IEC 9075)**, который определяет девятую редакцию спецификации языка SQL, применяемого для манипуляции данными в реляционных СУБД.

The screenshot shows the ISO website interface. At the top, there is a navigation bar with links for 'Standards', 'About us', 'News', 'Taking part', and 'Store'. A search bar and a shopping cart icon are also present. Below the navigation, the breadcrumb trail shows 'ICS < 35 < 35.060'. The main content area displays the standard details:

**ISO/IEC 9075-1:2023**  
Information technology — Database languages SQL — Part 1: Framework (SQL/Framework)

**General information**

Status : Published	Publication date : 2023-06
Edition : 6	Number of pages : 74
Technical Committee : ISO/IEC JTC 1/SC 32 Data management and interchange	
ICS : 35.060 Languages used in information technology	

**Preview**

**SUSTAINABLE DEVELOPMENT GOALS**  
This standard contributes to the following Sustainable Development Goal:

**Buy this standard**

Format: PDF   Language: English

CHF 187

<https://www.iso.org/standard/76583.html>

<https://peter.eisentraut.org/blog/2023/04/04/sql-2023-is-finished-here-is-whats-new>

# Стандарты SQL 2023

- **На текущий момент в магазине ISO опубликованы и доступны для покупки 11 частей стандарта SQL:2023 на английском языке в PDF-формате:**
  - ISO/IEC 9075-1:2023 Information technology — Database languages SQL — Part 1: Framework (SQL/Framework)
  - ISO/IEC 9075-2:2023 Information technology — Database languages SQL — Part 2: Foundation (SQL/Foundation)
  - ISO/IEC 9075-3:2023 Information technology — Database languages SQL — Part 3: Call-Level Interface (SQL/CLI)
  - ISO/IEC 9075-4:2023 Information technology — Database languages SQL — Part 4: Persistent stored modules (SQL/PSM)
  - ISO/IEC 9075-9:2023 Information technology — Database languages SQL — Part 9: Management of External Data (SQL/MED)
  - ISO/IEC 9075-10:2023 Information technology — Database languages SQL — Part 10: Object language bindings (SQL/OLB)
  - ISO/IEC 9075-11:2023 Information technology — Database languages SQL — Part 11: Information and definition schemas (SQL/Schemata)
  - ISO/IEC 9075-13:2023 Information technology — Database languages SQL — Part 13: SQL Routines and types using the Java TM programming language (SQL/JRT)
  - ISO/IEC 9075-14:2023 Information technology — Database languages SQL — Part 14: XML-Related Specifications (SQL/XML)
  - ISO/IEC 9075-15:2023 Information technology — Database languages SQL — Part 15: Multidimensional arrays (SQL/MDA)
  - ISO/IEC 9075-16:2023 Information technology — Database languages SQL — Part 16: Property Graph Queries (SQL/PGQ)

# Диалекты языка SQL (расширения SQL)

- **Transact-SQL (или T-SQL)** — СУБД MS SQL Server (Microsoft).
- **Jet SQL** — СУБД Access (Microsoft).
- **PL/SQL** (Procedural Language/SQL) — СУБД Oracle (Oracle).
- **PL/pgSQL** (Procedural Language/postgreSQL) — СУБД PostgreSQL (PostgreSQL Global Development Group).
- и др.



# SQL (Structured Query Language)

- В языке SQL вместо терминов отношение и переменная отношения используется термин **таблица**, а вместо терминов кортеж и атрибут — **строка** и **столбец**.

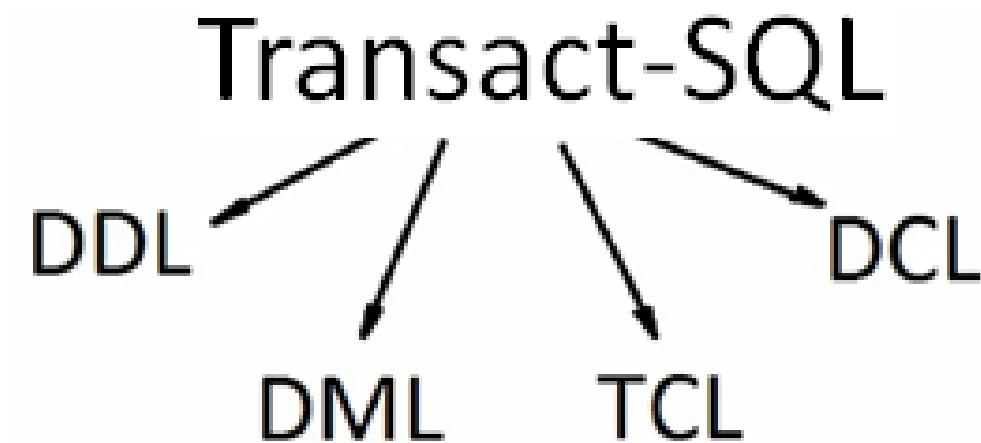


# CREATE READ (RETRIEVE) UPDATE DELETE

CRUD — сокращенное именование 4-х базовых функций, используемых при работе с хранилищами данных: создание (Create); чтение (Read); редактирование (Update); удаление (Delete).

# Команды SQL

<b>Команды языка определения данных (DDL - Data Definition Language)</b>	CREATE, ALTER, DROP, TRUNCATE
<b>Команды языка манипулирования данными (DML - Data Manipulation Language)</b>	INSERT, UPDATE, DELETE, SELECT
<b>Команды языка управления транзакциями (TCL - Transaction Control Language)</b>	SAVE TRANSACTION, COMMIT, ROLLBACK
<b>Команды языка управления данными (DCL - Data Control Language)</b>	GRANT, REVOKE, DENY



# Операторы SQL

- Операторы DDL - определения объектов базы данных
  - **CREATE DATABASE** - создать базу данных
  - **DROP DATABASE** - удалить базы данных
  - **CREATE TABLE** - создать таблицу
  - **ALTER TABLE** - изменить таблицу
  - **DROP TABLE** - удалить таблицу
  - **CREATE DOMAIN** - создать домен
  - **ALTER DOMAIN** - изменить домен
  - **DROP DOMAIN** - удалить домен
  - **CREATE VIEW** - создать представление
  - **DROP VIEW** - удалить представление

# Операторы SQL

- Операторы DML - манипулирования данными
- **SELECT** - отобрать строки из таблиц
- **INSERT** - добавить строки в таблицу
- **UPDATE** - изменить строки в таблице
- **DELETE** - удалить строки в таблице

# Язык SQL

- **звездочка (\*)** - для обозначения "все";
- **квадратные скобки ([])** – конструкции, заключенные в эти скобки, являются необязательными (т.е. могут быть опущены);
- **фигурные скобки {}** – конструкции, заключенные в эти скобки, должны рассматриваться как целые синтаксические единицы;
- **многоточие (...)** – указывает на то, что непосредственно предшествующая ему синтаксическая единица факультативно может повторяться один или более раз;
- **прямая черта ()** – означает наличие выбора из двух или более возможностей.
- **точка с запятой ;** – завершающий элемент предложений SQL;
- **запятая ,** – используется для разделения элементов списков;
- **пробелы ()** – могут вводиться для повышения наглядности между любыми синтаксическими конструкциями предложений SQL;
- **прописные жирные латинские буквы и символы** – используются для написания конструкций языка SQL;
- **строчные буквы** – используются для написания конструкций, которые должны заменяться конкретными значениями, выбранными пользователем;

# SQL : DML

- Есть четыре основных типа запросов данных в SQL, которые относятся к так называемому языку манипулирования данными (Data Manipulation Language или DML):
  - **SELECT** – выбрать строки из таблиц;
  - **INSERT** – добавить строки в таблицу;
  - **UPDATE** – изменить строки в таблице;
  - **DELETE** – удалить строки в таблице;
- Каждый из этих запросов имеет различные операторы и функции, которые используются для того, чтобы произвести какие-то действия с данными.
- **Запрос SELECT чаще всего используется в БД и имеет самое большое количество опций .**

# SELECT

- Для выборки данных используется команда **SELECT**.
- **SELECT [DISTINCT] <список столбцов>**
- **FROM <имя таблицы> [JOIN <имя таблицы> ON <условия связывания>]**
- **[WHERE <условия выборки>]**
- **[GROUP BY <список столбцов для группировки> [HAVING <условия выборки групп>] ]**
- **[ORDER BY <список столбцов для сортировки>]**

# Секция FROM

- Перечень таблиц, из которых производится выборка данных, указывается в секции **FROM**. Выборка возможна как из одной таблицы, так и из нескольких логически взаимосвязанных.
- Логическая взаимосвязь осуществляется с помощью подсекции **JOIN**.
- На каждую логическую связь пишется отдельная подсекция.
- Внутри подсекции указывается условие связи двух таблиц (обычно по условию равенства первичных и вторичных ключей).

# Выборка (A WHERE c)

## Персоны

Имя	Возраст	Вес
Harry	34	80
Donald	29	70
Helena	54	54
Peter	34	80

$\sigma_{\text{Возраст} \geq 34}(\text{Персоны})$

Имя	Возраст	Вес
Harry	34	80
Helena	54	54
Peter	34	80

SELECT \* FROM "Персоны" WHERE "Возраст" >= 34

# Проекция (PROJECT A {x, y,... z})

## Персоны

Имя	Возраст	Вес
Harry	34	80
Donald	29	70
Helena	54	54
Peter	34	80

$\Pi_{\text{Возраст}, \text{Вес}}(\text{Персоны})$

Возраст	Вес
29	70
54	54
34	80

SELECT DISTINCT "Возраст", "Вес" FROM "Персоны"

# Объединение (A UNION B)

## Персоны

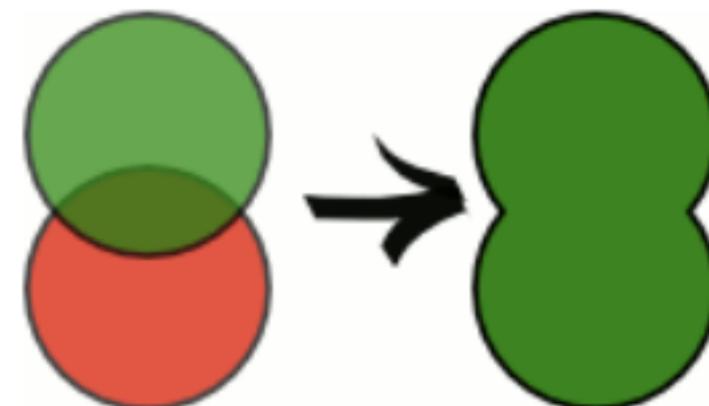
Имя	Возраст	Вес
Harry	34	80
Donald	29	70
Helena	54	54
Peter	34	80

## Персонажи

Имя	Возраст	Вес
Daffy	24	19
Donald	29	70
Scrooge	81	27

Персоны  $\cup$  Персонажи

Имя	Возраст	Вес
Harry	34	80
Donald	29	70
Helena	54	54
Peter	34	80
Daffy	24	19
Scrooge	81	27



# Объединение (A UNION B)

Персоны

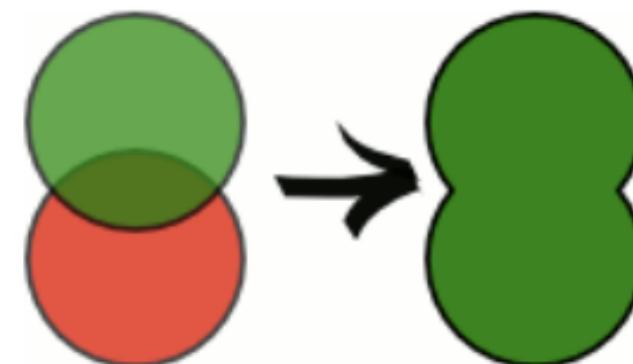
Имя	Возраст	Вес
Harry	34	80
Donald	29	70
Helena	54	54
Peter	34	80

Персонажи

Имя	Возраст	Вес
Daffy	24	19
Donald	29	70
Scrooge	81	27

Персоны  $\cup$  Персонажи

Имя	Возраст	Вес
Harry	34	80
Donald	29	70
Helena	54	54
Peter	34	80
Daffy	24	19
Scrooge	81	27



SELECT \* FROM "Персоны" UNION SELECT \* FROM "Персонажи"

# Пересечение (A INTERSECT B)

Персоны

Имя	Возраст	Вес
Harry	34	80
Donald	29	70
Helena	54	54
Peter	34	80

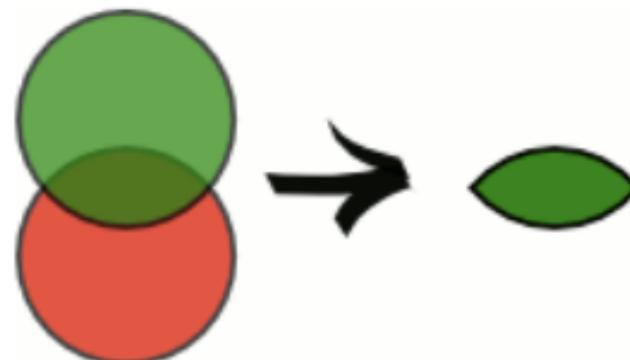
Персонажи

Имя	Возраст	Вес
Daffy	24	19
Donald	29	70
Scrooge	81	27

Персоны  $\cap$  Персонажи

Имя	Возраст	Вес
Donald	29	70

```
SELECT * FROM "Персоны"  
NATURAL JOIN "Персонажи"
```



# Разность (A MINUS B)

Персоны

Имя	Возраст	Вес
Harry	34	80
Donald	29	70
Helena	54	54
Peter	34	80

Персонажи

Имя	Возраст	Вес
Daffy	24	19
Donald	29	70
Scrooge	81	27

Персоны \ Персонажи

Имя	Возраст	Вес
Harry	34	80
Helena	54	54
Peter	34	80

```
SELECT * FROM "Персоны"
NATURAL LEFT JOIN "Персонажи"
WHERE "Персонажи" IS NULL
```



# Произведение (A TIMES B)

## Мультфильмы

Код_Мульта	Название_Мульта
1	The Simpsons
2	Family Guy
3	Duck Tales

## Каналы

Код_Канала	Название_Канала
1	СТС
2	2x2

## Мультфильмы × Каналы

Код_Мульта	Название_Мульта	Код_Канала	Название_Канала
1	The Simpsons	1	СТС
2	Family Guy	1	СТС
3	Duck Tales	1	СТС
1	The Simpsons	2	2x2
2	Family Guy	2	2x2
3	Duck Tales	2	2x2

# Произведение (A TIMES B)

## Мультфильмы

Код_Мульта	Название_Мульта
1	The Simpsons
2	Family Guy
3	Duck Tales

## Каналы

Код_Канала	Название_Канала
1	СТС
2	2x2

Мультфильмы × Каналы

Код_Мульта	Название_Мульта	Код_Канала	Название_Канала
1	The Simpsons	1	СТС
2	Family Guy	1	СТС
3	Duck Tales	1	СТС
1	The Simpsons	2	2x2
2	Family Guy	2	2x2
3	Duck Tales	2	2x2

SELECT \* FROM "Персоны", "Персонажи"

# Деление (A DIVIDE BY B)

## Мультфильмы

Код_Мульта	Название_Мульта	Название_Канала
1	The Simpsons	RenTV
1	The Simpsons	2x2
1	The Simpsons	CTC
2	Family Guy	RenTV
2	Family Guy	2x2
3	Duck Tales	CTC
3	Duck Tales	2x2

## Каналы

Название_Канала
RenTV
2x2

Мультфильмы ÷ Каналы

Код_Мульта	Название_Мульта
1	The Simpsons
2	Family Guy

# Деление (A DIVIDE BY B)

## Мультфильмы

Код_Мульта	Название_Мульта	Название_Канала
1	The Simpsons	RenTV
1	The Simpsons	2x2
1	The Simpsons	СТС
2	Family Guy	RenTV
2	Family Guy	2x2
3	Duck Tales	СТС
3	Duck Tales	2x2

## Каналы

Название_Канала
RenTV
2x2

## Мультфильмы ÷ Каналы

Код_Мульта	Название_Мульта
1	The Simpsons
2	Family Guy

```
SELECT "Код_Мульта", "Название_Мульта"
FROM "Мультфильмы"
JOIN "Каналы" USING ("Название_Канала")
GROUP BY "Код_Мульта", "Название_Мульта"
HAVING COUNT(DISTINCT "Название_Канала") = (
    SELECT COUNT(DISTINCT "Название_Канала") FROM "Каналы"
)
```

# Соединение ((A TIMES B) WHERE P)

Мультфильмы

Код_Мульта	Название_Мульта	Название_Канала
1	The Simpsons	2x2
2	Family Guy	2x2
3	Duck Tales	RenTV

Каналы

Код_Канала	Частота
RenTV	3.1415
2x2	783.25

Мультфильмы▷Каналы

Код_Мульта	Название_Мульта	Название_Канала	Код_Канала	Частота
1	The Simpsons	2x2	2x2	783.25
2	Family Guy	2x2	2x2	783.25
3	Duck Tales	RenTV	RenTV	3.1415

```
SELECT *
FROM "Мультфильмы"
JOIN "Каналы" ON ("Название_Канала" = "Код_Канала")
```

# Первичный ключ

- **Потенциальный ключ**
- В реляционной модели данных - подмножество атрибутов отношения, удовлетворяющее требованиям уникальности и минимальности (несократимости).
- **Уникальность** означает, что не существует двух кортежей данного отношения, в которых значения этого подмножества атрибутов совпадают (равны).
- **Минимальность** (несократимость) означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, удовлетворяющее условию уникальности. Иными словами, если из потенциального ключа убрать любой атрибут, он утратит свойство уникальности.
- **Первичный ключ (англ, primary key)**
- В реляционной модели данных один из потенциальных ключей отношения, выбранный в качестве основного ключа (или ключа по умолчанию).
- Если в отношении имеется единственный потенциальный ключ, он является и первичным ключом. Если потенциальных ключей несколько, один из них выбирается в качестве первичного, а другие называют «альтернативными».

# Суррогатный ключ

Даже при наличии естественного ключа добавление суррогатного в большинстве случаев оправдано.

UserID	E-Mail	Пароль	Имя	Дата
12	tolstoy@mail.ru	*****	Толстой Л.Н.	09.09.1828
48	saltykov@inbox.ru	*****	Николай Щедрин	27.01.1826
543	fmd@mail.ru	*****	Федор Михайлович	11.11.1821
5	sukin_syn@mail.ru	*****	Пушкин А.С.	06.06.1799

UserID	Привилегии
12	Админ
12	Модератор
48	Модератор
543	Игрок
5	Админ

UserID	Разделы
12	Новости
12	Флуд
48	Флуд
48	Юмор
543	E-Mail
5	Поэзия

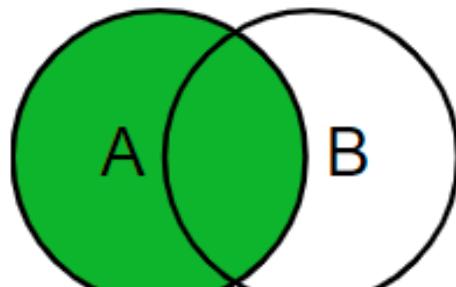
# Секция JOIN

- **SELECT** Table1.Field1, Table2.Field2
- **FROM** Table1
- **JOIN** Table2
- **ON** Table2.ID1 =Table1.ID1
- **AND** Table2.ID2 =Table1.ID2
- **AND** ....

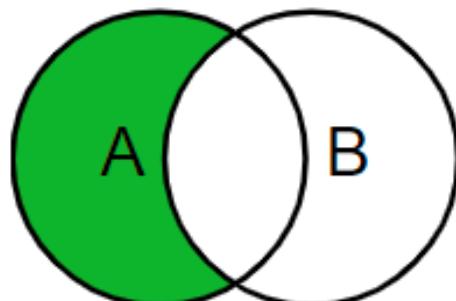
# Секция JOIN

Тип	Результат
<b>JOIN</b>	В результирующем наборе присутствуют только записи, значения связанных полей в которых совпадают.
<b>LEFT JOIN</b>	В результирующем наборе присутствуют все записи из Table1 и соответствующие им записи из Table2. Если соответствия нет, поля из Table2 будут пустыми
<b>RIGHT JOIN</b>	В результирующем наборе присутствуют все записи из Table2 и соответствующие им записи из Table1. Если соответствия нет, поля из Table1 будут пустыми
<b>FULL JOIN</b>	В результирующем наборе присутствуют все записи из Table1 и соответствующие им записи из Table2. Если соответствия нет – поля из Table2 будут пустыми. Записи из Table2, которым не нашлось пары в Table1, тоже будут присутствовать в результирующем наборе. В этом случае поля из Table1 будут пустыми.
<b>CROSS JOIN</b>	Результирующий набор содержит все варианты комбинации строк из Table1 и Table2. Условие соединения при этом не указывается.

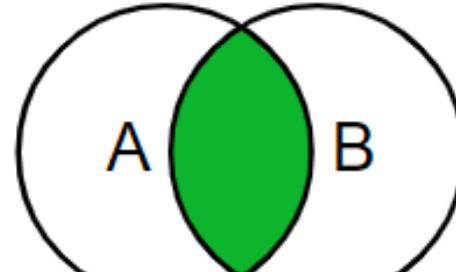
# SQL JOINS



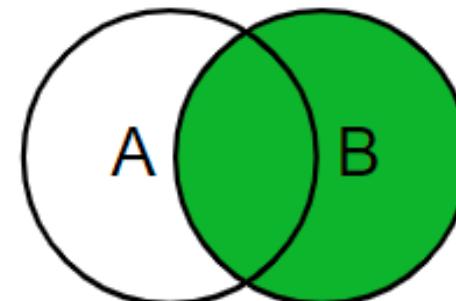
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key
```



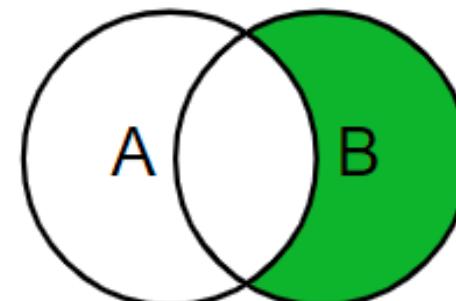
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key  
WHERE B.Key IS NULL
```



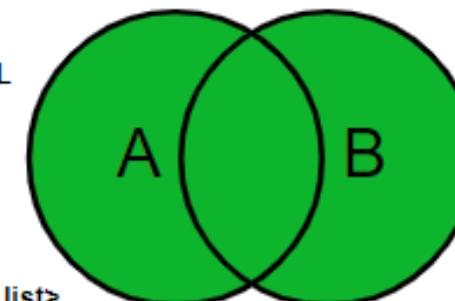
```
SELECT <select_list>  
FROM TableA A  
INNER JOIN TableB B  
ON A.Key = B.Key
```



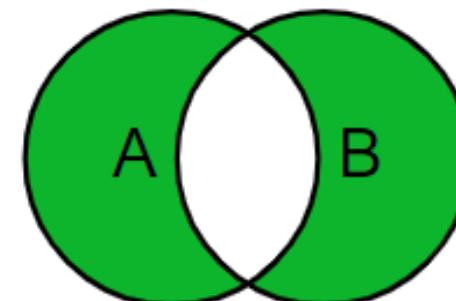
```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key
```



```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL  
OR B.Key IS NULL
```

# Секция JOIN

Table1

Key1	Field1
1	A
2	B
3	C

Table2

Key2	Field2
1	AAA
2	BBB
2	CCC
4	DDD

- **SELECT** Table1.Field1, Table2.Field2
- **FROM** Table1
- **JOIN** Table2 **ON** Table1.Key1 = Table2.Key2

A	AAA
B	BBB
B	CCC

# Запрос INSERT

- **INSERT** — оператор языка SQL, который позволяет добавить строки в таблицу, заполняя их значениями. Значения можно вставлять перечислением с помощью слова `values` и перечислив их в круглых скобках через запятую или оператором `select`.

**INSERT INTO** имя\_таблицы (столбец1, столбец2, столбец3, ...)  
**VALUES** (значение1, значение2, значение3, ...);

**INSERT INTO** имя\_таблицы  
**VALUES** (значение1, значение2, значение3, ...);

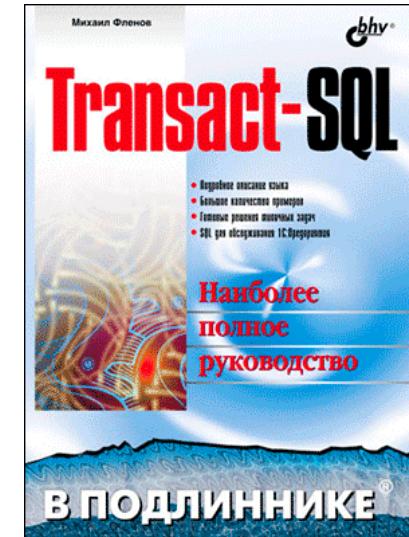
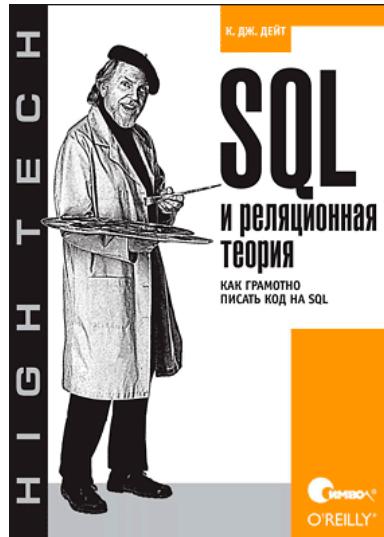
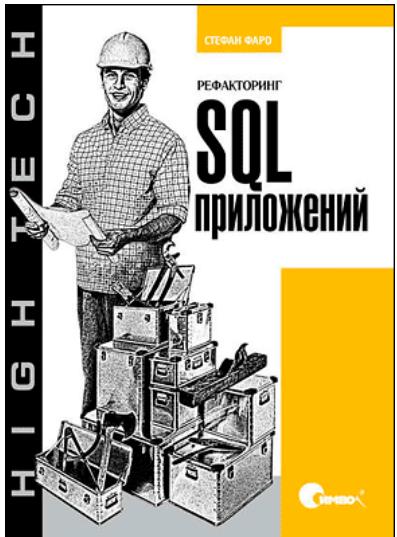
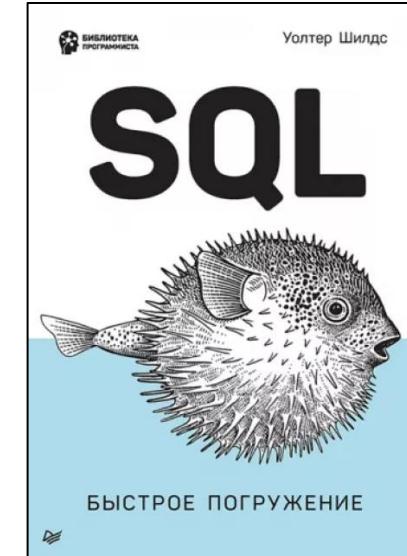
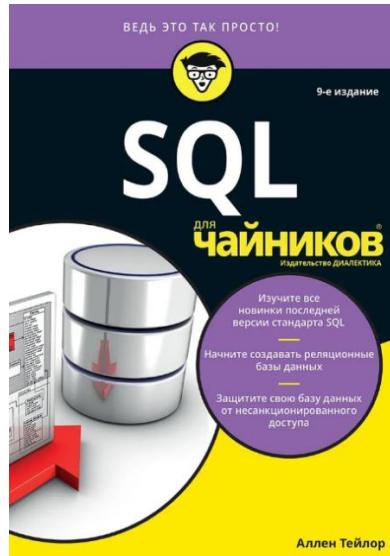
# Запрос UPDATE

- **UPDATE** используется для того, чтобы изменить существующие значения или освободить поле в строке, поэтому новые значения должны соответствовать существующему типу данных и обеспечивать приемлемые значения. Если нет желания изменить значения во всех строках, то нужно использовать условие **WHERE**.
- **UPDATE** имя\_таблицы **SET** column1 = 'data1', column2 = 'data2'  
**WHERE** column3 = 'data3';

# Запрос **DELETE**

- **Запрос DELETE** полностью удаляет строку из базы данных. Если вы хотите удалить одно единственное поле, то нужно использовать запрос UPDATE и установить для этого поля значение, которое будет являться аналогом NULL в вашей программе. Будьте внимательны, и ограничивайте ваш запрос DELETE условием WHERE, иначе вы можете потерять все содержимое таблицы.
- **DELETE FROM** имя\_таблицы **WHERE** column1 = 'data1';

# Подробнее про SQL в книгах





# NoSQL и нереляционные базы данных



# Нереляционные базы данных

- Все преимущества и недостатки реляционных БД основаны на жесткой структуризации и типизации сведений об объектах.
- С одной стороны, можно оптимизировать хранение и индексирование данных за счет нормализации или же денормализации.
- С другой — сложно организовать хранение и обработку плохо структурированных (например, объекты кэша) или вовсе не структурированных данных (например, данные из нескольких источников).
- Для борьбы с этими ограничениями было разработано семейство нереляционных БД.

# | NoSQL (Not Only SQL)

- **SQL (Structured Query Language)** — это язык структурированных запросов, используемый для управления и манипулирования реляционными базами данных. SQL-базы данных применяются там, где необходимо хранить и управлять данными структурированной природы, например, информацией о продуктах, покупателях и оформленных заказах в магазине.
- **NoSQL (Not Only SQL)** — это широкий термин, который относится к нереляционным моделям баз данных, которые используют различные структуры для хранения данных: документы, ключ-значение, столбцовые и графовые БД. NoSQL-базы данных применяются, когда необходимо хранить данные неструктурированной природы, например, большие объёмы текстовых данных, изображения и видео.

# NoSQL

- **NoSQL-базы данных можно сравнить с большим и сложным пазлом, в котором каждая часть представляет отдельный фрагмент информации.**
- Информация может быть представлена не только в виде текста, но и в виде изображений, звуковых файлов, видеоматериалов и т. д. Каждый фрагмент может быть различного размера и формы, и для того, чтобы получить полную картину, необходимо собрать и объединить все фрагменты.
- Это подобно работе с NoSQL-базами данных, где каждый элемент может быть различного типа и формата, и для того, чтобы получить всю необходимую информацию, следует использовать различные методы и инструменты запросов. Как и в пазле, каждый элемент данных имеет своё место и собирается в единую картину по мере необходимости.

# NoSQL

- **NoSQL-базы данных имеют более гибкую модель данных, которая не требует таблиц и связей, как в SQL-базах.**
- Как правило, данные в NoSQL-базах хранятся в документах, коллекциях или графах.
- **Документ** — это структурированный контейнер для хранения данных в формате пар ключ-значение, где пары могут иметь разные типы данных.
- **Коллекция** — это группа документов, связанных между собой.
- **Граф** — это набор вершин и связей между ними.
- NoSQL-базы данных используют специальные языки запросов, которые позволяют пользователям запрашивать и манипулировать данными

# Основные отличия между SQL и NoSQL

- Схема данных:** SQL-база данных имеет строгую схему данных, которая определяет типы данных и связи между таблицами. В NoSQL-базах данных нет строгой схемы данных.
- Масштабируемость:** SQL-базы данных имеют ограничения на масштабируемость, из-за чего они могут быть неэффективны в обработке большого количества данных. NoSQL-базы данных обладают большой масштабируемостью, из-за чего они могут обрабатывать большие объемы данных.
- Гибкость запросов:** SQL имеет очень мощный язык запросов, что делает его лучшим выбором для сложных запросов, связанных с большим количеством таблиц. С другой стороны, NoSQL имеет простой язык запросов, который хорошо подходит для запросов, связанных с большим количеством данных.
- Скорость обработки:** Несмотря на то, что SQL обычно работает медленнее, чем NoSQL, его мощный язык запросов позволяет быстро обрабатывать сложные запросы. С другой стороны, NoSQL работает очень быстро с неструктурированными данными в больших объемах.

# Особенности SQL и NoSQL

- **Особенности SQL:**

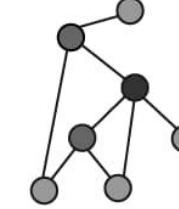
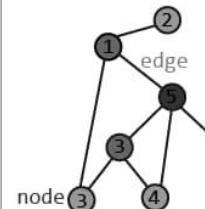
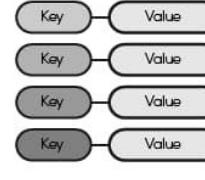
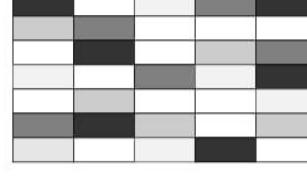
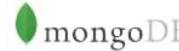
- SQL-базы данных имеют жёстко определённый формат хранения данных, что делает их наиболее подходящим выбором для представления сложных связанных данных;
- SQL имеет мощный язык запросов, который позволяет обрабатывать сложные запросы;
- SQL-базы данных обычно требуют больших затрат на обслуживание и настройку.

- **Особенности NoSQL:**

- NoSQL-базы данных хранят данные в форме документов, что делает их лучшим выбором для хранения неструктурированных данных, таких как данные о социальных сетях и блогах;
- NoSQL имеет простой язык запросов, который позволяет быстро обрабатывать запросы на огромные объёмы неструктурированных данных;
- NoSQL-базы данных позволяют быстрее масштабировать и расширять базу данных.

# | NoSQL (не только SQL)

## NoSQL DATABASE TYPES

Document	Graph	Key-Value	Wide-Column																							
 <pre>{ "user":{ "id":"143", "name":"improgrammer", "city":"New York" } }</pre>	 	 <table border="1"><tr><td>143</td><td>John Smith</td></tr><tr><td>178</td><td>Oduar park</td></tr><tr><td>285</td><td>New York</td></tr><tr><td>129</td><td>(0 856 1489)</td></tr></table>	143	John Smith	178	Oduar park	285	New York	129	(0 856 1489)	 <table border="1"><tr><td>1</td><td>Fruit</td><td>A Foo</td><td>B Baz</td><td></td></tr><tr><td>2</td><td>City</td><td>E DC</td><td>D PLA</td><td>G FLD</td></tr><tr><td>3</td><td>State</td><td>A NZ</td><td>C CL</td><td></td></tr></table>	1	Fruit	A Foo	B Baz		2	City	E DC	D PLA	G FLD	3	State	A NZ	C CL	
143	John Smith																									
178	Oduar park																									
285	New York																									
129	(0 856 1489)																									
1	Fruit	A Foo	B Baz																							
2	City	E DC	D PLA	G FLD																						
3	State	A NZ	C CL																							
  	 	   	  																							

# Базы данных «Ключ-значение»

- Это простейшая разновидность нереляционных БД. Данные хранятся в виде словаря, где указателем выступает ключ.

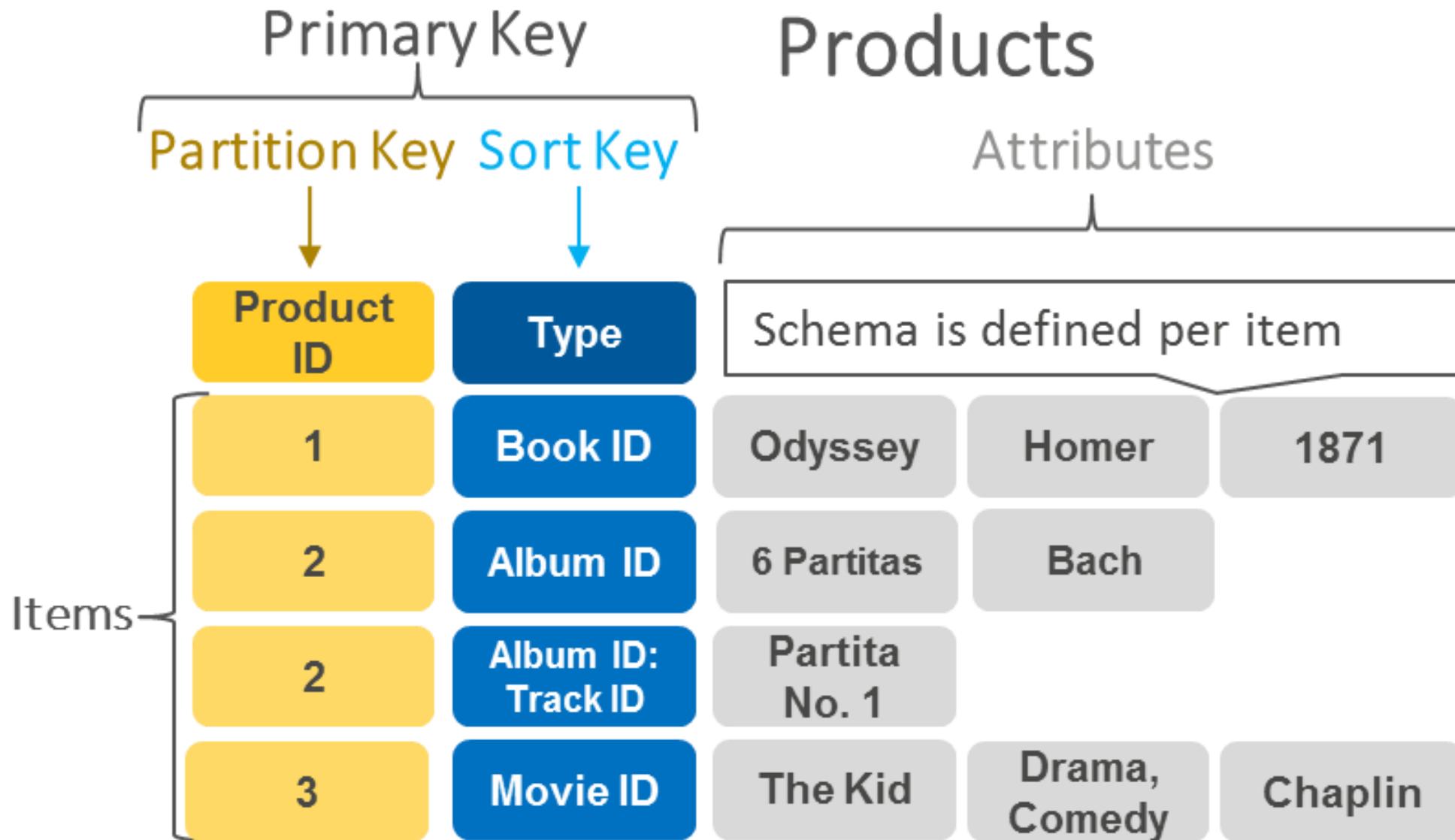
ключ	значение
user_id:	f5badc33-5bd7-4b65-a737-b5304675f476
color:	blue
repetitions:	3
text:	hello world
data:	{...}

- Особенности:**
- Хранение и обработка разных по типу и содержанию данных: в одном хранилище под разными ключами могут находиться файлы, строки, текст, числа, JSON-объекты и другие типы данных.
- Высокая скорость доступа к данным за счет адресного хранения.
- Легкое масштабирование. Можно создать правила шардирования по определенным ключам – например, сессии пользователей разных сайтов хранятся в различных сегментах БД.
- Ограничения:** Поскольку подход не предполагает жесткой типизации и структуризации данных, то контроль их валидности, а также нейминг ключей отдаются на откуп разработчику.
- Примеры:** Amazon, DynamoDB, Redis, Riak, LevelDB, различные хранилища кэша – например, Memcached и пр.

# Базы данных «Ключ-значение»

- Самый простой вариант баз данных NoSQL вращается вокруг концепции ассоциативных массивов, другими словами, он просто связывает данный ключ с записью любого типа из простой строки или JSON в виде файлы.
- Базы данных «ключ-значение» организованы в разделы или корзины, которые могут содержать одну или несколько сущностей, и каждая запись представлена уникальным ключом строки. Записи, в свою очередь, содержат одно или несколько полей.
- Концепция ключей разделов и строк - один из наиболее важных аспектов NoSQL, поскольку он позволяет перемещать логические разделы (определяемые ключом раздела) по физическим разделам (узлам) в соответствии с их рабочей нагрузкой.
- В то же время это одно из основных препятствий для новичков: выбор неправильного ключа приведет к катастрофе, если ваши запросы не воспользуются выбранным ключом, а после того, как вы его выбрали, пути назад уже не будет.

# Базы данных «Ключ-значение»



# Базы данных «Ключ-значение»

Keys

Values

2398239

{ "name": "Michał", "Age": "31"}

2398240

"Lorem ipsum dolor sit amet"

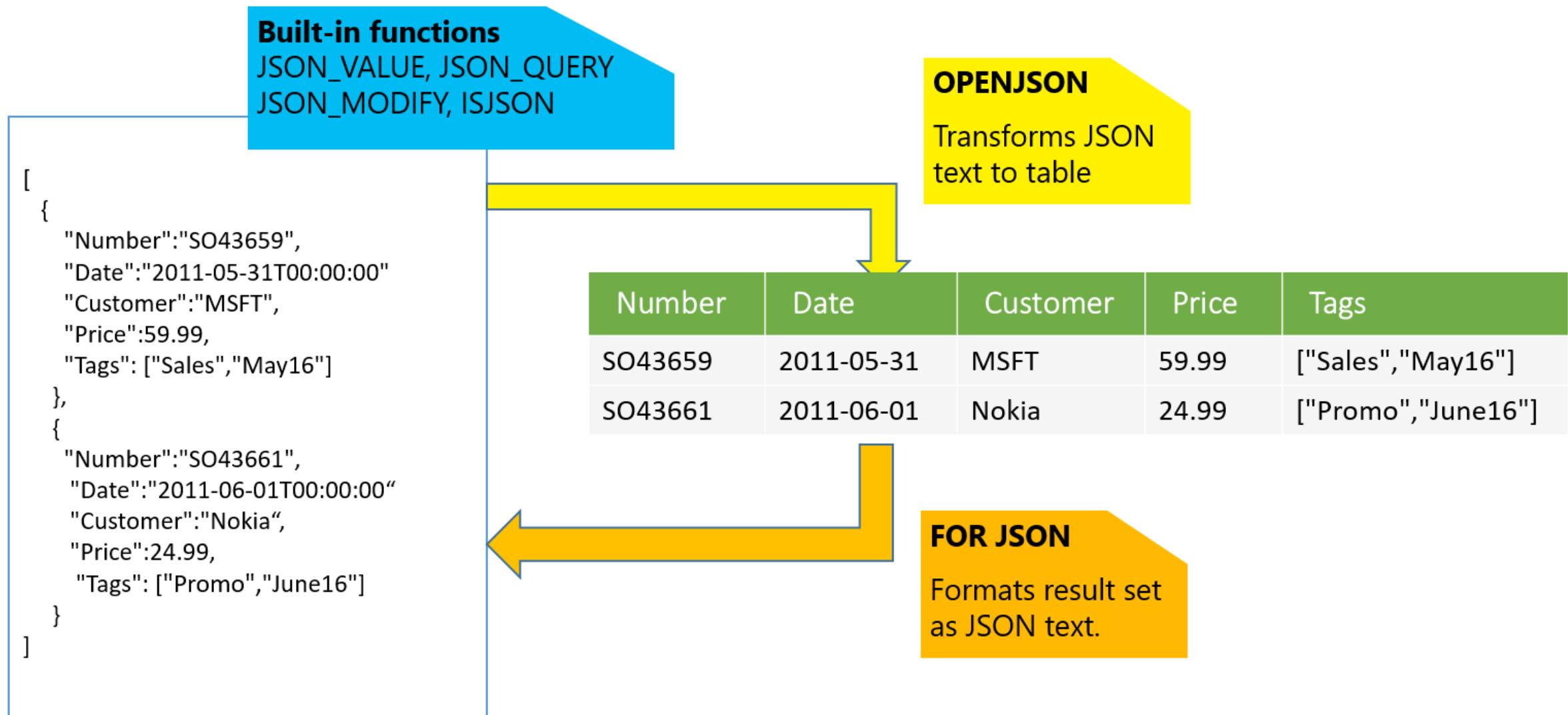
2398241

{ "name": "Marlon Brando", "Profession": "Actor"}

2398242

42

# Базы данных «Ключ-значение»



# Базы данных «Ключ-значение»

- **Примеры:** Amazon, DynamoDB, Redis, Riak, LevelDB, различные хранилища кэша – например, Memcached и пр.



Amazon  
DynamoDB



# Документоориентированные БД

- В отличие от баз типа «Ключ-значение» данные здесь хранятся в структурированных форматах – **XML, JSON, BSON**. Тем не менее, сохраняется адресный доступ к данным по ключу. При этом содержимое документа может иметь различный набор свойств.
- Например, каталог профилей пользователей: один в качестве предпочтений указал любимое блюдо, а другой – видеогру. Поскольку эти сведения нельзя хранить в одном поле ввиду логической и структурной разобщенности, они записываются в отдельные свойства отдельных документов. При необходимости можно добавить в документы новые свойства, не нарушив при этом общей целостности данных.
- **Особенности:**
  - хорошо подходят для быстрой разработки систем и сервисов, работающих с по-разному структуризованными данными,
  - легко масштабируются и меняют структуру при необходимости.
- **Примеры:** MongoDB, RethinkDB, CouchDB, DocumentDB.

# Документоориентированные БД

```
ID: завтрак
```

```
{  
    "type": "тост",  
    "хлеб": "цельнозерновой",  
    "намазка": [  
        "масло",  
        "джем"  
    ]  
}
```

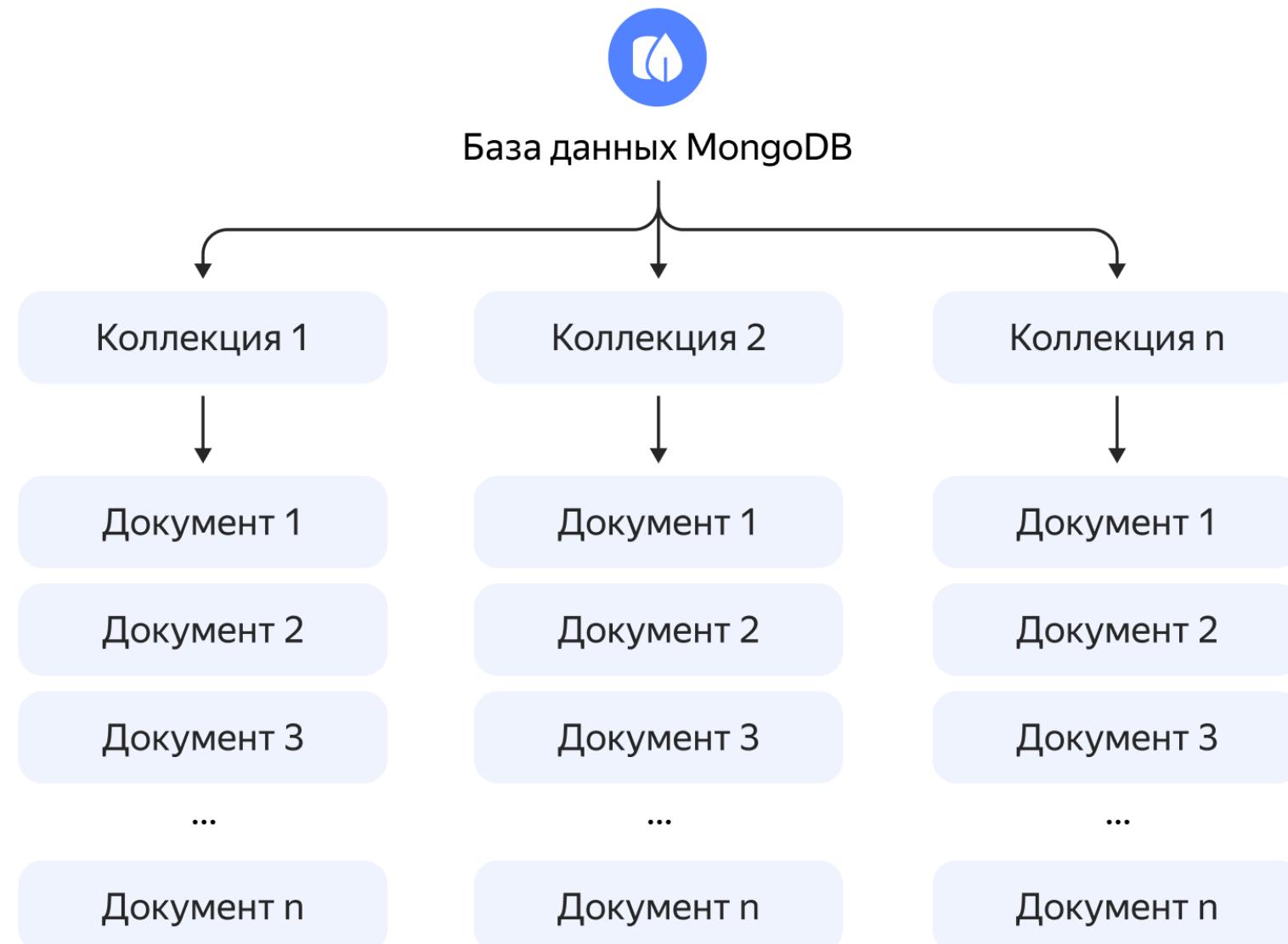
```
ID: ланч
```

```
{  
    "тип": "салат",  
    "вегетарианский": нет,  
    "ингредиенты": [  
        "шпинат",  
        "помидоры",  
        "огурцы",  
        "морковь",  
        "заправка": [  
            "оливковое масло",  
            "уксус",  
            "мед",  
            "лимон",  
            "соль",  
            "перец",  
        ],  
        "тунец",  
        "греческие орехи",  
    ],  
    "рейтинг": "5 звезд",  
    "ресторан": "Skylight Diner"  
}
```

```
ID: обед
```

```
{  
    "тип": "пицца",  
    "размер": "большой",  
    "топпинги": [  
        "пепперони",  
        "томаты",  
        "сосиски",  
    ],  
    "цена": 9.00,  
    "порезанная": да  
}
```

# Документоориентированные БД



# | Документоориентированные БД

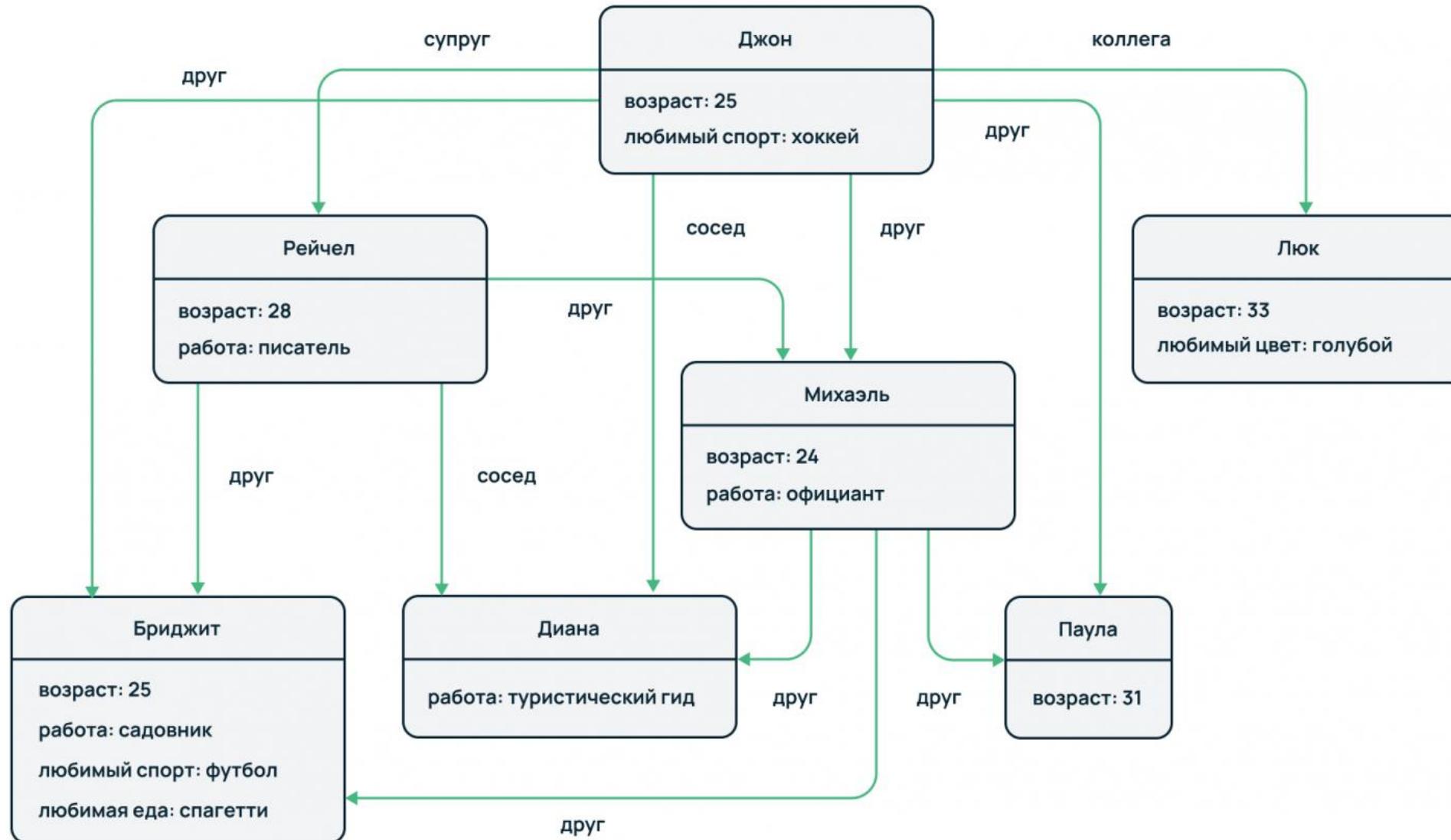
- Примеры: MongoDB, RethinkDB, CouchDB, DocumentDB.



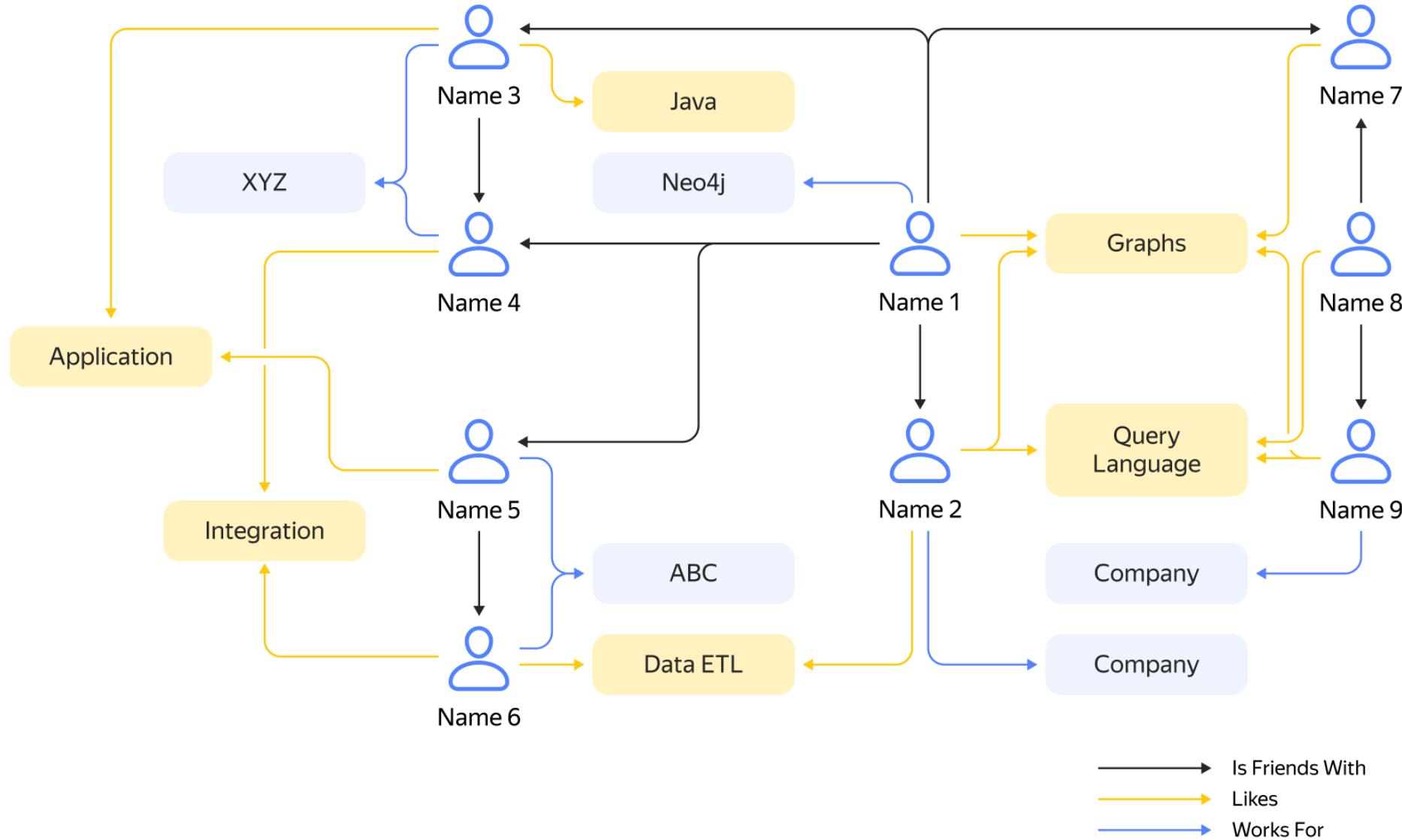
# Графовые базы данных

- Это семейство баз предназначено для моделирования сложных отношений с помощью теории графов, где связями выступают ребра графа, а сами объекты – это узлы или вершины.
- Такой подход может пригодиться при анализе профилей пользователей социальных сетей. Один пользователь подписан на обновления второго, другой пользователь подписан на определенное сообщество и так далее. Также технология может использоваться при анализе экономической активности контрагентов для выявления различных схем мошенничества. Например, можно отследить использование определенных счетов, карт или реквизитов контрагентов в различных операциях.
- **Особенности:** высокая производительность, поскольку обход ребер и вершин значительно быстрее анализа множества внешних и внутренних таблиц и их соединения по условию отбора в реляционных БД.
- **Примеры:** Neo4J, JanusGraph, Dgraph, OrientDB.

# Графовые базы данных



# Графовые базы данных



# Графовые базы данных

- Примеры: Neo4J, JanusGraph, Dgraph, OrientDB.

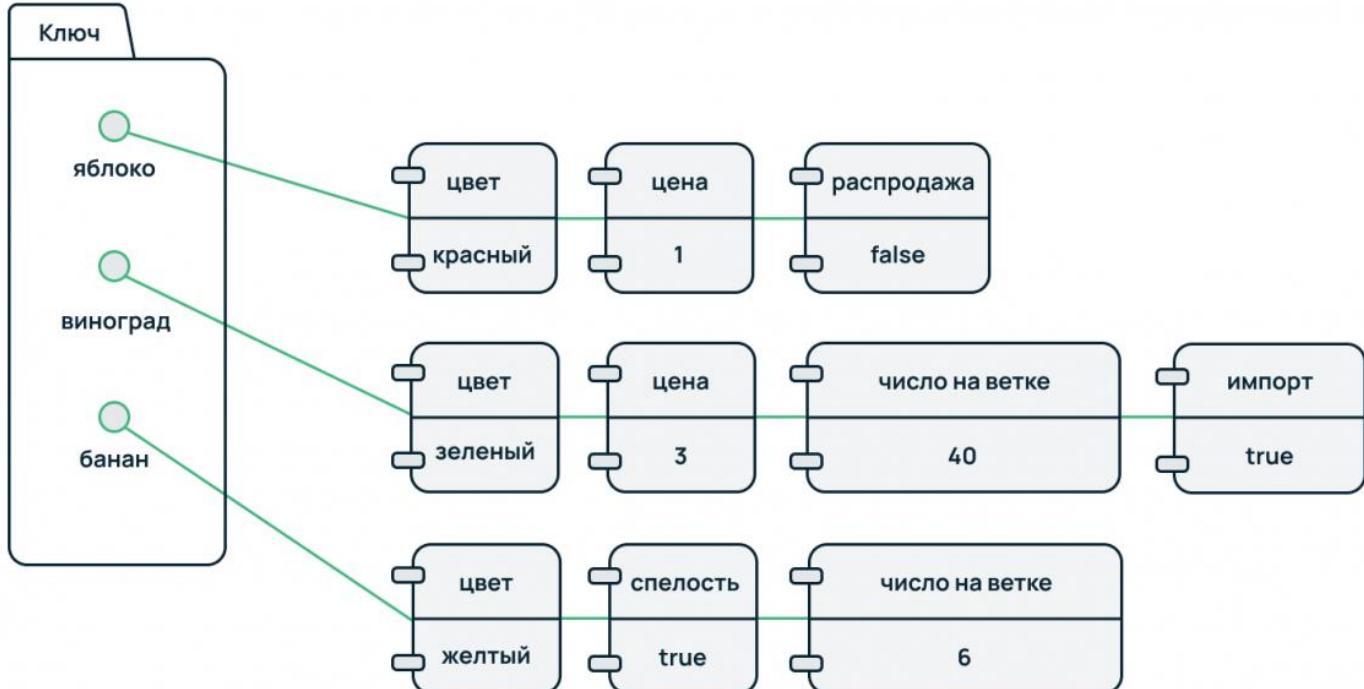


# Колоночные базы данных

- Как можно понять из названия, записи в таких базах хранятся не по строкам, а по столбцам (колонкам). Вместо таблиц здесь используются колоночные семейства. Они содержат ключи, указывающие на формат строки записи информации об объекте. Каждая строка имеет свой набор свойств, что позволяет хранить в рамках одного семейства разно структурированные данные.
- Технология активно используется при построении аналитических систем и сервисов, работающих с большими объемами данных.
- **Особенности:**
- С группировкой свойств по колонкам при запросе индексируется меньший объем данных, что обеспечивает высокую скорость его выполнения.
- Широкие возможности масштабирования и модификации структуры — так, при добавлении новых колонок не придется их жестко формализовывать, как в случае с реляционными базами.
- **Примеры:** Cassandra, HBase, ClickHouse.

# Колоночные базы данных

Семейство столбцов: Фрукты

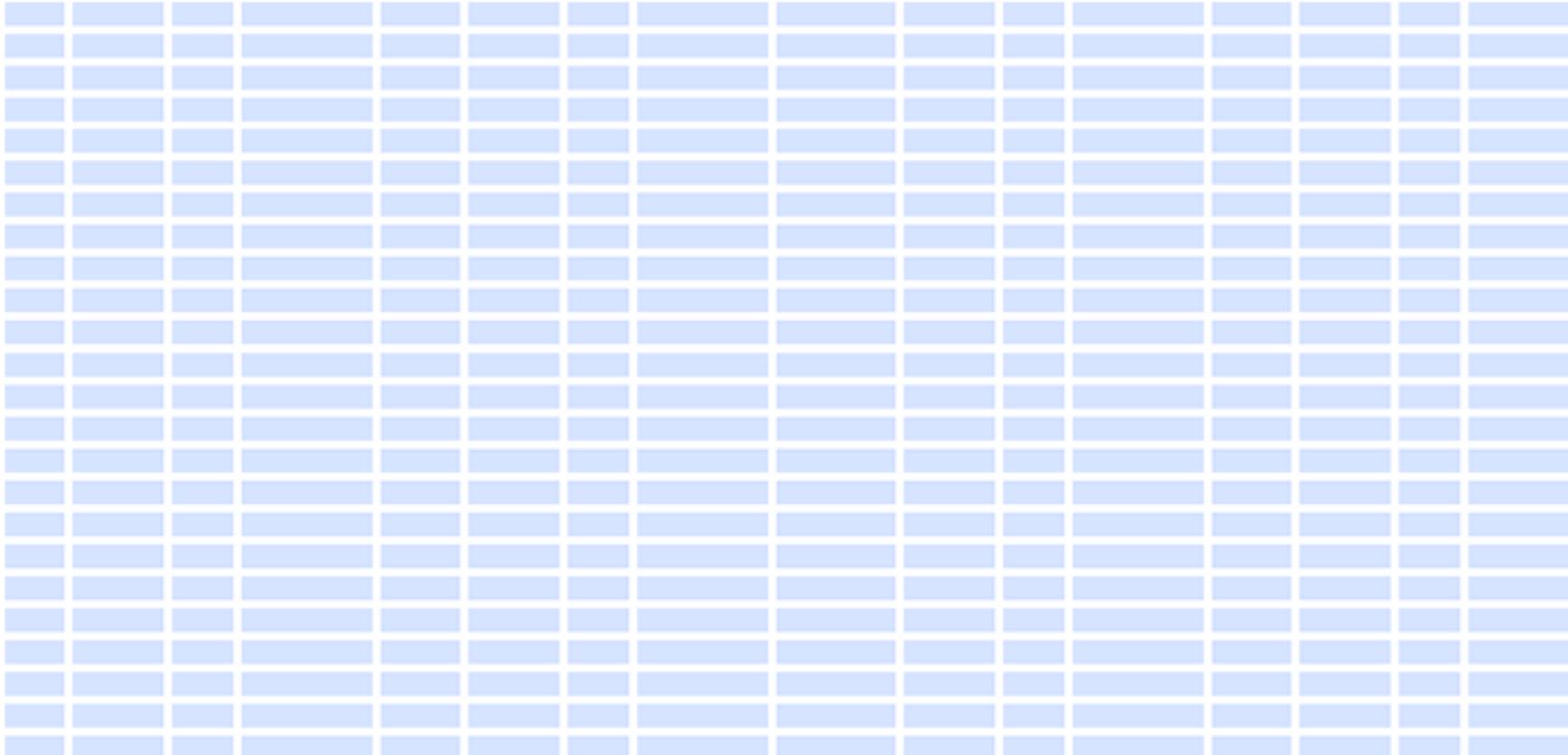


На рисунке приведен пример колоночного хранения информации о фруктах. Известно три типа фруктов: яблоки, виноград, бананы. Все они объединены в семейство фруктов.

У каждого фрукта индивидуальный набор свойств. Для яблок это цвет, цена и наличие. У винограда это цвет, цена, число ягод в связке и происхождение (импортный или нет). У бананов же это цвет, цена, число в связке и зрелость.

Чтобы получить детальную сводку по одному типу фруктов, достаточно в запросе указать его идентификатор. При этом можно построить аналитический запрос по общим для всего семейства признакам – например, посчитать число фруктов с группировкой по цвету, вычислить среднюю цену на все фрукты в магазине и т.д.

# Колоночные базы данных



Колоночные  
(столбцовые)  
СУБД

# Колоночные базы данных

- Примеры: Cassandra, HBase, ClickHouse.



# Базы данных временных рядов

- Данный тип БД можно использовать при необходимости отслеживания исторической динамики по ряду показателей. Здесь данные группируются по временным меткам. Базы временных рядов чаще ориентированы на запись, чем на построение сложных аналитических запросов.
- **Особенности:** Можно обрабатывать постоянный поток входных данных.
- Ограничения: Производительность зависит от объема поступающей информации, количества отслеживаемых метрик, а также временного лага между записью новых данных и запросами на чтение
- **Примеры БД:** OpenTSDB, Prometheus, InfluxDB, TimescaleDB

# Базы данных временных рядов

Время	Температура CPU	Значение нагрузки	Утилизация памяти %
2019-10-31T03:48:05+00:00	37	0.85	92
2019-10-31T03:48:10+00:00	42	0.87	90
2019-10-31T03:48:15+00:00	33	0.74	87
2019-10-31T03:48:20+00:00	34	0.72	77
2019-10-31T03:48:25+00:00	40	0.88	81
2019-10-31T03:48:30+00:00	42	0.89	82
2019-10-31T03:48:35+00:00	41	0.88	82

На рисунке выше приведен пример использования такой БД для отслеживания состояния ПК во времени по ряду показателей – температуре процессора, загрузке системы и потреблению оперативной памяти.

# Базы данных временных рядов

- Примеры БД: OpenTSDB, Prometheus, InfluxDB, TimescaleDB





# Microsoft Access

## Реляционная БД

# Microsoft Access

- Microsoft Access – это система управления реляционными базами данных
- Access работает со следующими объектами:
  - Таблицами;
  - Формами;
  - Запросами;
  - Отчетами;
  - Макросами;
  - Модулями.
- Все объекты содержатся в одном файле с расширением \*.accdb или \*.mdb

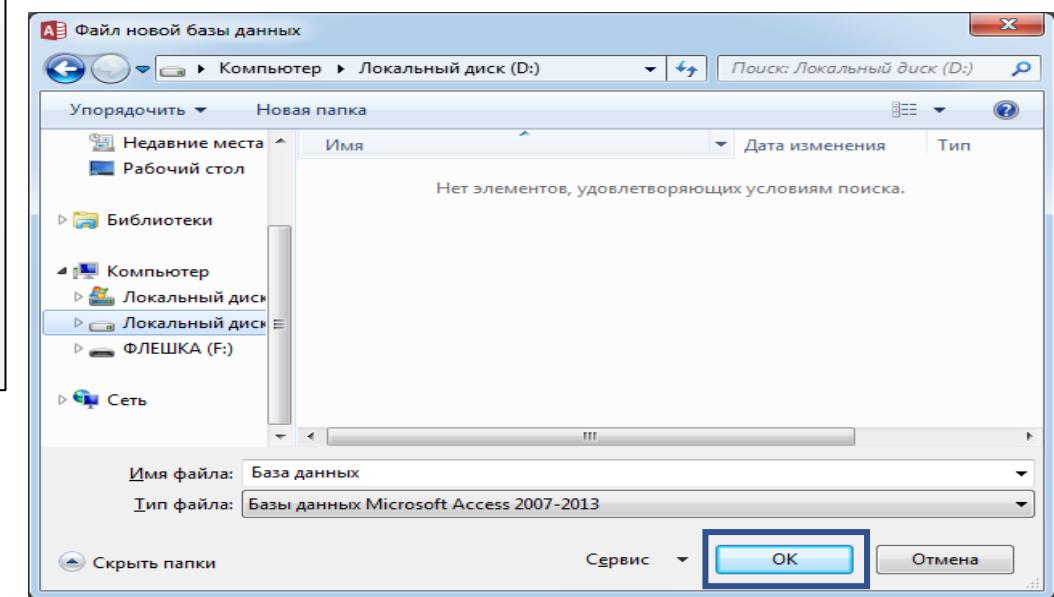
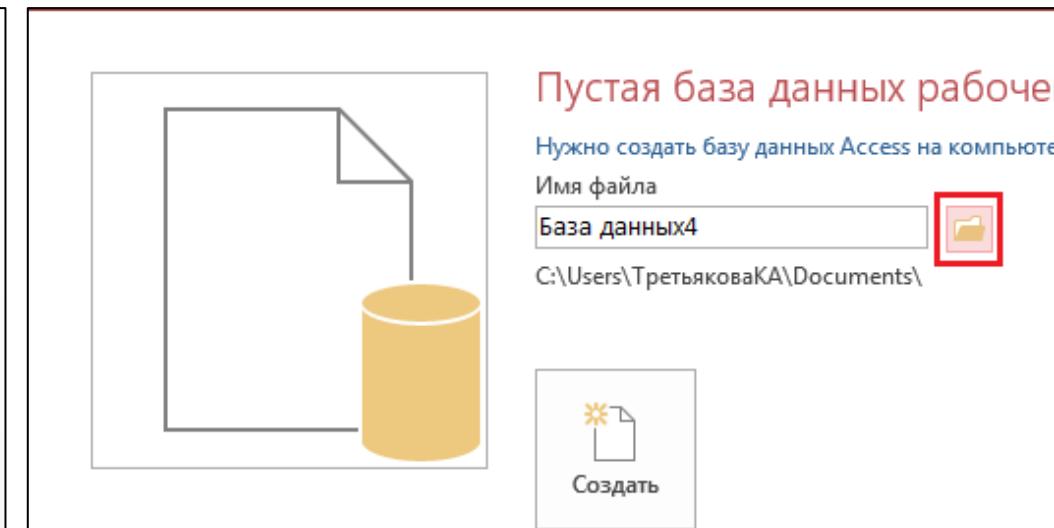
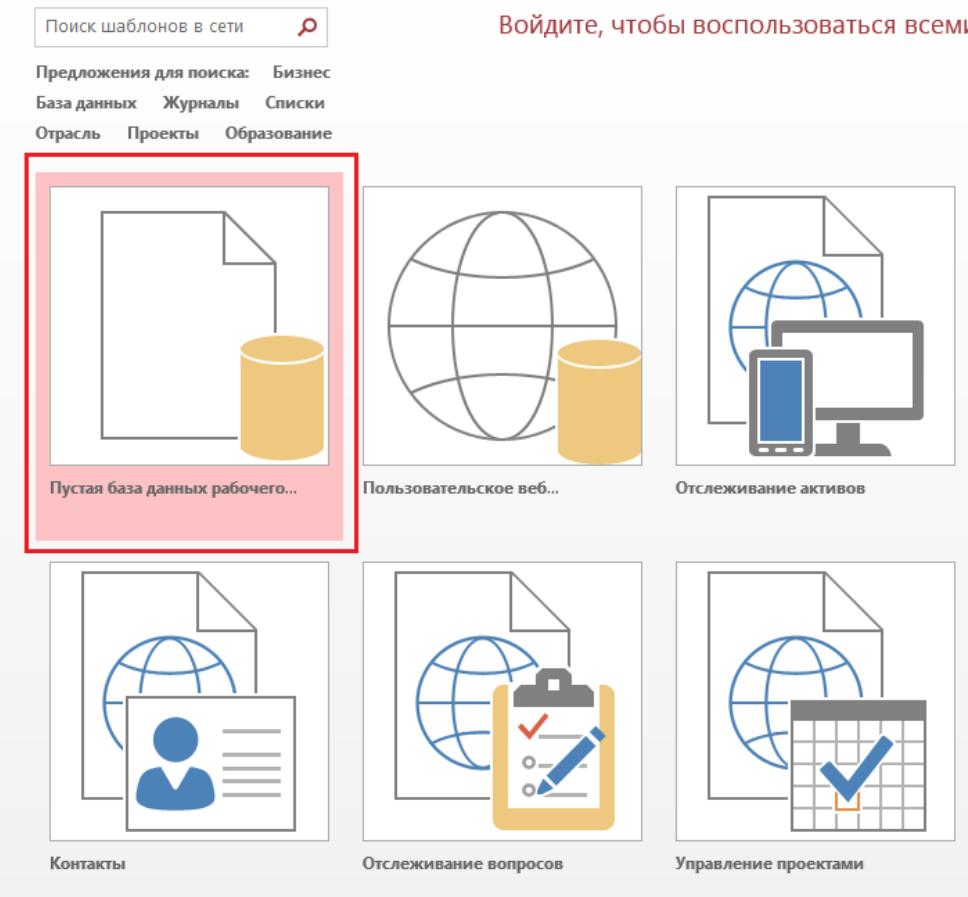
# Интерфейс Microsoft Access

Access

Последние

В последнее время вы не открывали файлы.  
Чтобы найти файл, щелкните "Открыть другие  
Файлы".

Открыть другие Файлы



# Интерфейс Microsoft Access

The screenshot shows the Microsoft Access application window. The ribbon tabs at the top include: ФАЙЛ, ГЛАВНАЯ, СОЗДАНИЕ, ВНЕШНИЕ ДАННЫЕ, РАБОТА С БАЗАМИ ДАННЫХ, РАБОТА С ТАБЛИЦАМИ (highlighted), ПОЛЯ, and ТАБЛИЦА. The status bar at the bottom indicates 'Режим таблицы' (Table View).

The main area displays a table named 'Таблица1' (Table1). The first column is labeled 'Код' (Code) and the second column is labeled '(№)' (No.). A tooltip 'Щелкните для добавления' (Click to add) is visible over the second column header. The table has one row with a single cell containing a red asterisk (\*) and a blue placeholder '(№)'.

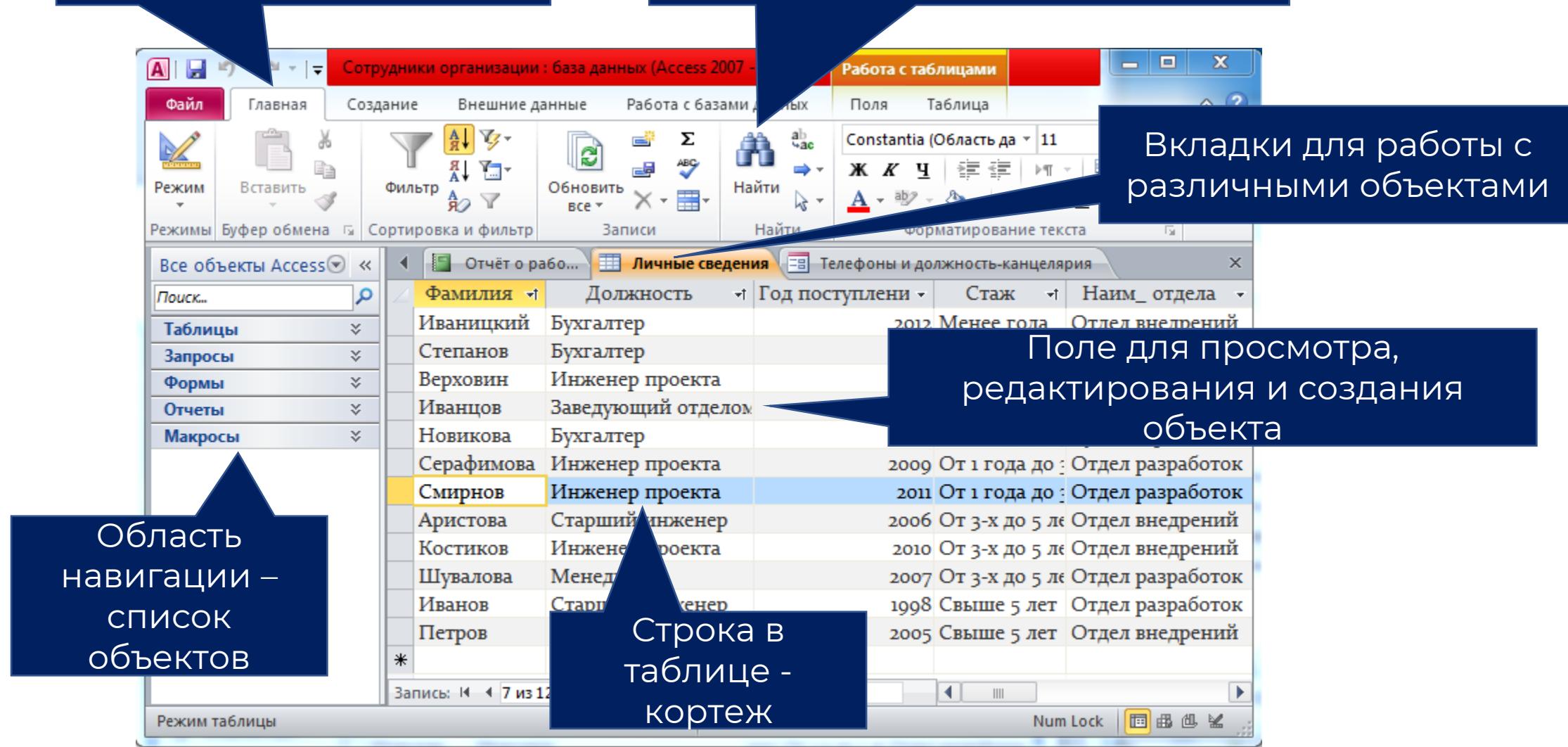
On the right side, there is a floating window titled 'Список полей' (List of fields) which contains the message: 'Нет полей, доступных для добавления в текущее представление.' (There are no fields available for adding to the current view).

The status bar at the bottom also shows 'NUM LOCK' and other standard keyboard indicator lights.

# Интерфейс Microsoft Access

Вкладки для выбора режима работы

Панель инструментов при работе с вкладкой «Работа с таблицами»



# Типы полей

- Поля могут содержать данные следующих основных типов:

Тип данных	Использование
<b>Короткий текст ("Текстовый")</b>	Буквенно-цифровые данные (имена, названия и т. д.) до 255 символов.
<b>Длинный текст ("Поле МЕМО")</b>	Большие объемы буквенно-цифровые данные: предложений и абзацев. До 1 Гб, но отображаются первые 64000 символов.
<b>Число</b>	Числовые данные.
<b>Дата/время</b>	Значения даты и времени.
<b>Денежный</b>	Денежные данные, хранящие до 4 десятичных знаков после запятой.
<b>Счетчик</b>	Целые числа, которые задаются автоматически при вводе каждой новой записей. Эти числа не могут быть изменены пользователем
<b>Логический</b>	Access хранит числовое значение 1 и 0 (нуль) (истина/ложь)
<b>Объект OLE</b>	Изображения, графики или объекты другого приложения Windows.
<b>Гиперссылка</b>	Адрес ссылки на документ или файл в Интернете, локальной сети или на локальном компьютере.
<b>Вложение</b>	Можно вложить файлы, например рисунки, документы, электронные таблицы и диаграммы.
<b>Вычисляемый</b>	Можно создать выражение, использующее данных из одного или нескольких полей.

# Типы полей

Имя поля	Тип данных
	Короткий текст
	Длинный текст
	Числовой
	Дата и время
	Денежный
	Счетчик
	Логический
	Поле объекта OLE
	Гиперссылка
	Вычисляемый

Свойства поля

Общие	Подстановка
Размер поля	255
Формат поля	
Маска ввода	
Подпись	
Значение по умолчанию	
Правило проверки	
Сообщение об ошибке	
Обязательное поле	Нет

- Поле каждого типа имеет свой набор свойств.
- Наиболее важными свойствами полей являются:
  - **размер поля** — определяет максимальную длину текстового или числового поля;
  - **формат поля** — устанавливает формат данных;
  - **обязательное поле** — указывает на то, что данное поле обязательно надо заполнить.

# Объекты

**ОТЧЕТ** - это объект БД, предназначенный для форматирования, вычисления, печати и обобщения выбранных данных. Отчет можно просматривать на экране

**ФОРМА**. Этот объект предназначен для ввода и вывода данных, а также для управления работой приложения. Внешний вид данных, извлекаемых из таблиц или запросов, определяется формами. С помощью форм можно запускать макросы или процедуры Visual Basic.



**ЗАПРОС**. Это объект, обеспечивающий настраиваемый вывод данных из одной или нескольких таблиц. Имеется возможность создавать запросы на выборку, обновление, вставку и удаление данных.

**СТРАНИЦЫ**. Объекты БД, которые позволяют публиковать данные на веб-страницах в корпоративной сети организации.

# Объекты

**МАКРОС.** Этот объект представляет собой структурированное описание одного или нескольких действий, которые необходимо выполнить в качестве реакции на определенные события.



**МОДУЛЬ.** Это объект, содержащий пользовательские процедуры, написанные на языке Visual Basic. Модули обеспечивают выполнение функций из любого места приложения, или могут быть связаны с конкретной формой.

# Объекты

**Таблица является базовым объектом.**

**Таблица** - это объект, используемый для непосредственного хранения данных. Каждая таблица содержит сведения об определенном предмете.

**Вся информация находится именно в таблицах.** Все остальные объекты – производные, и являются правилами, по которым преобразуется информация из таблиц.

**Любая таблица может быть представлена в двух видах:**

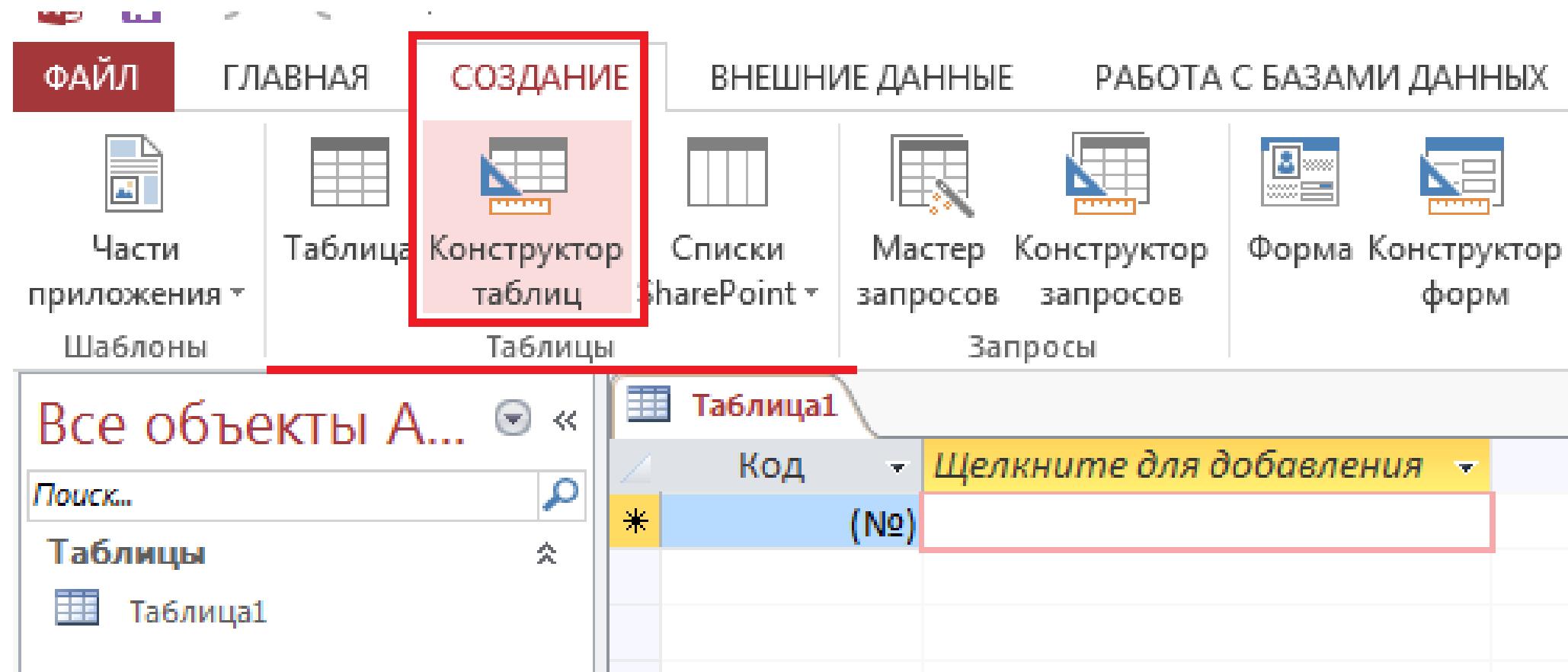
**В режиме конструктора.** В этом режиме для каждого поля указывается название, тип и выбирается ключевое поле.

**В оперативном режиме.** В этом режиме выполняется ввод, редактирование и просмотр записей таблицы.



# Создание таблицы

- **Таблица** - Базовый объект БД, все остальные объекты создаются на основе существующих таблиц (производные объекты).



# Создание таблицы

- Создаем структуру таблицы
- Для этого: Вкладка Главная-Режим-Конструктор

The screenshot shows the Microsoft Access ribbon with the 'ГЛАВНАЯ' tab highlighted in red. The 'КОНСТРУКТОР' tab is also highlighted in orange. The 'Режим таблицы' button is selected in the 'Режим' group. A table is open in the 'Конструктор' view, showing columns for 'Имя поля' (Field Name) and 'Тип данных' (Data Type). The first row has 'Личный номер' in 'Имя поля' and 'Числовой' in 'Тип данных'. Other rows show 'Номер приказа' (Short Text), 'Дата приказа' (Date/Time), 'Должность' (Short Text), and 'Зарплата' (Currency).

Имя поля	Тип данных
Личный номер	Числовой
Номер приказа	Короткий текст
Дата приказа	Дата и время
Должность	Короткий текст
Зарплата	Денежный

# Создание таблицы

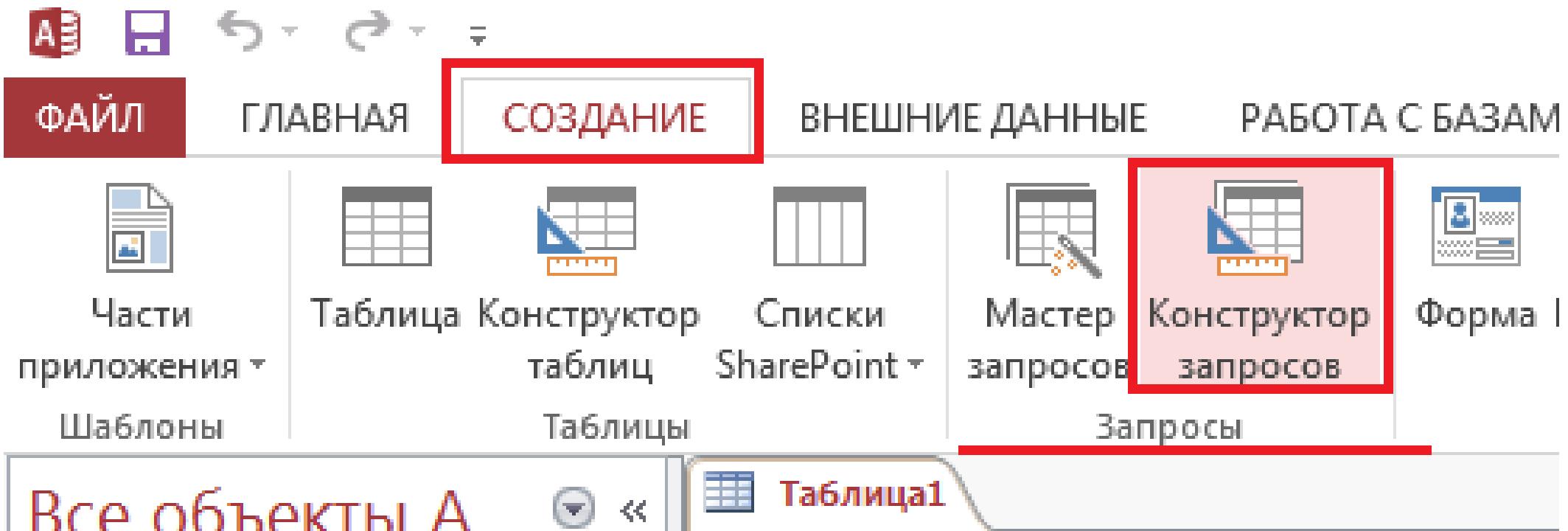
- Вводим информацию в таблицу
- Для этого: Вкладка Главная-Режим таблицы

The screenshot shows the Microsoft Access ribbon with the 'ГЛАВНАЯ' (Home) tab selected. A red box highlights the 'Режим' (Mode) button in the 'ФАЙЛ' (File) tab group. Below the ribbon, a context menu is open under 'Режим таблицы' (Table View), which includes options like 'Режим таблицы' (Table View), 'Конструктор' (Designer), 'Анкета' (Form), 'Назначен...' (Assigned...), 'Запросы' (Queries), and 'Запрос1' (Query1). To the right of the ribbon, there's a 'СОЗДАНИЕ' (Create) tab with various tools for creating tables and queries. The main workspace displays a table named 'Назначения' (Assignments) with the following data:

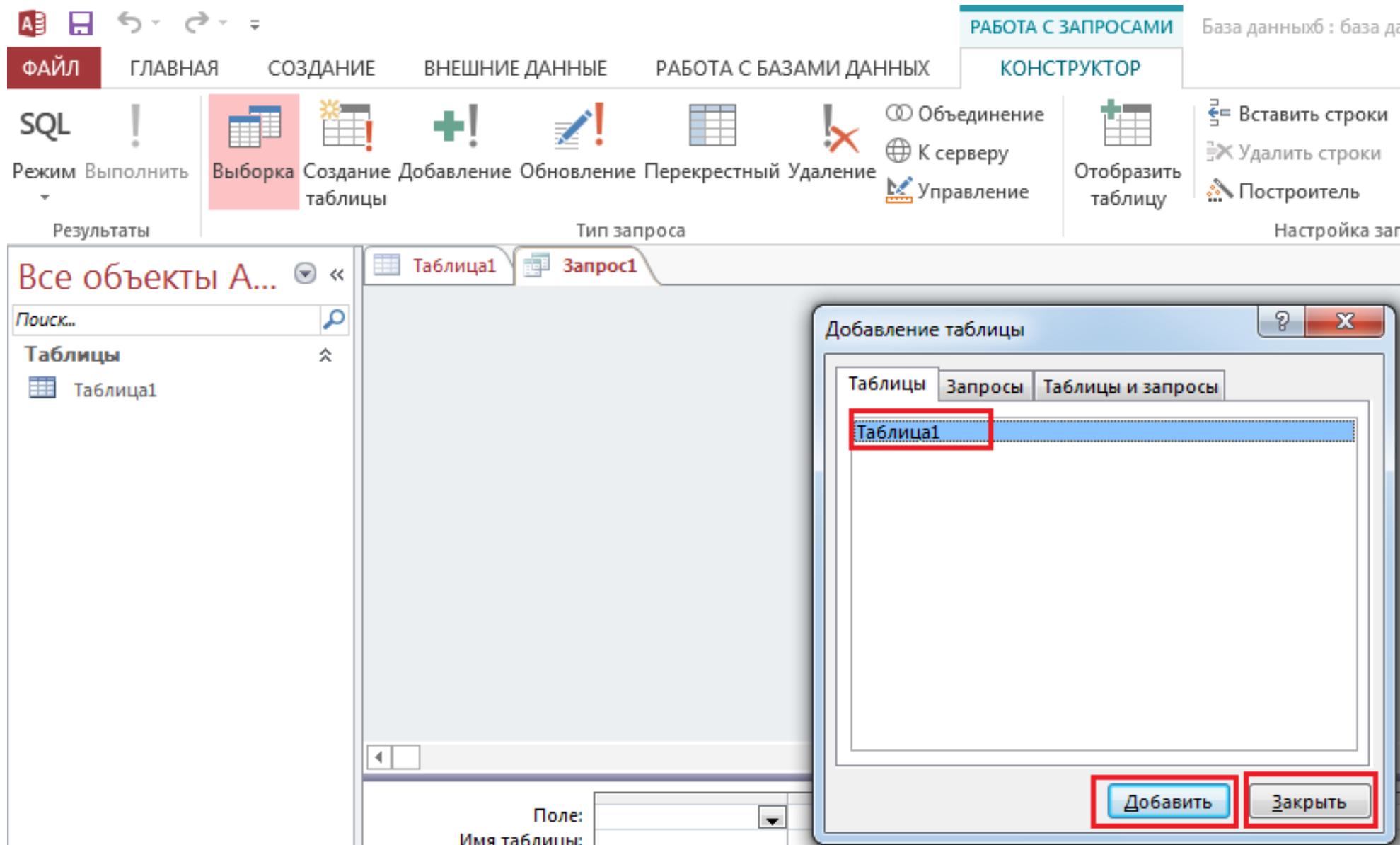
Номер приказа	Дата приказа	Должность	Зарплата
1 423	11.01.2000	Директор	5 000,00 ₽
2 424	15.02.2000	Инженер	2 000,00 ₽
2 425	11.12.2000	Старший инженер	2 500,00 ₽
4 427	12.01.2001	Бухгалтер	2 000,00 ₽
4 428	13.01.2002	Главный бухгалтер	2 500,00 ₽

# Создание запросов

- Запросы
- Отбор данных на основании заданных условий.



# Создание запросов

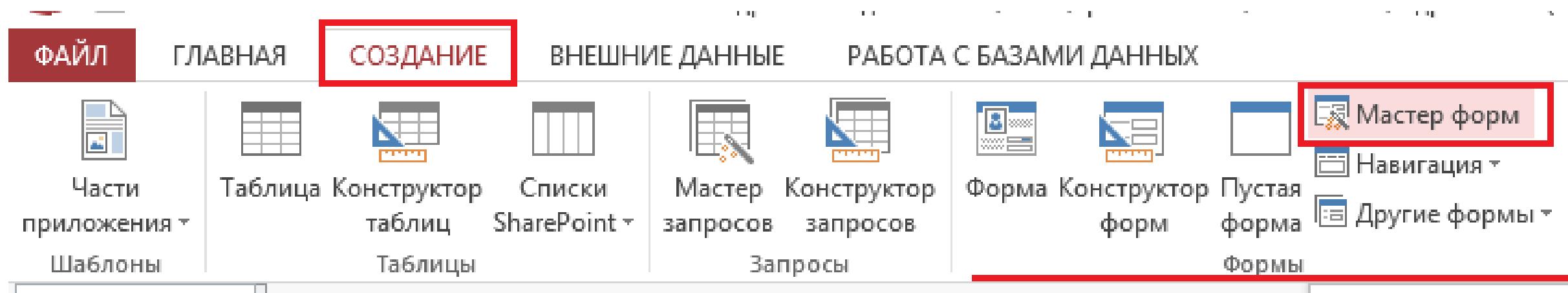


# Создание запросов

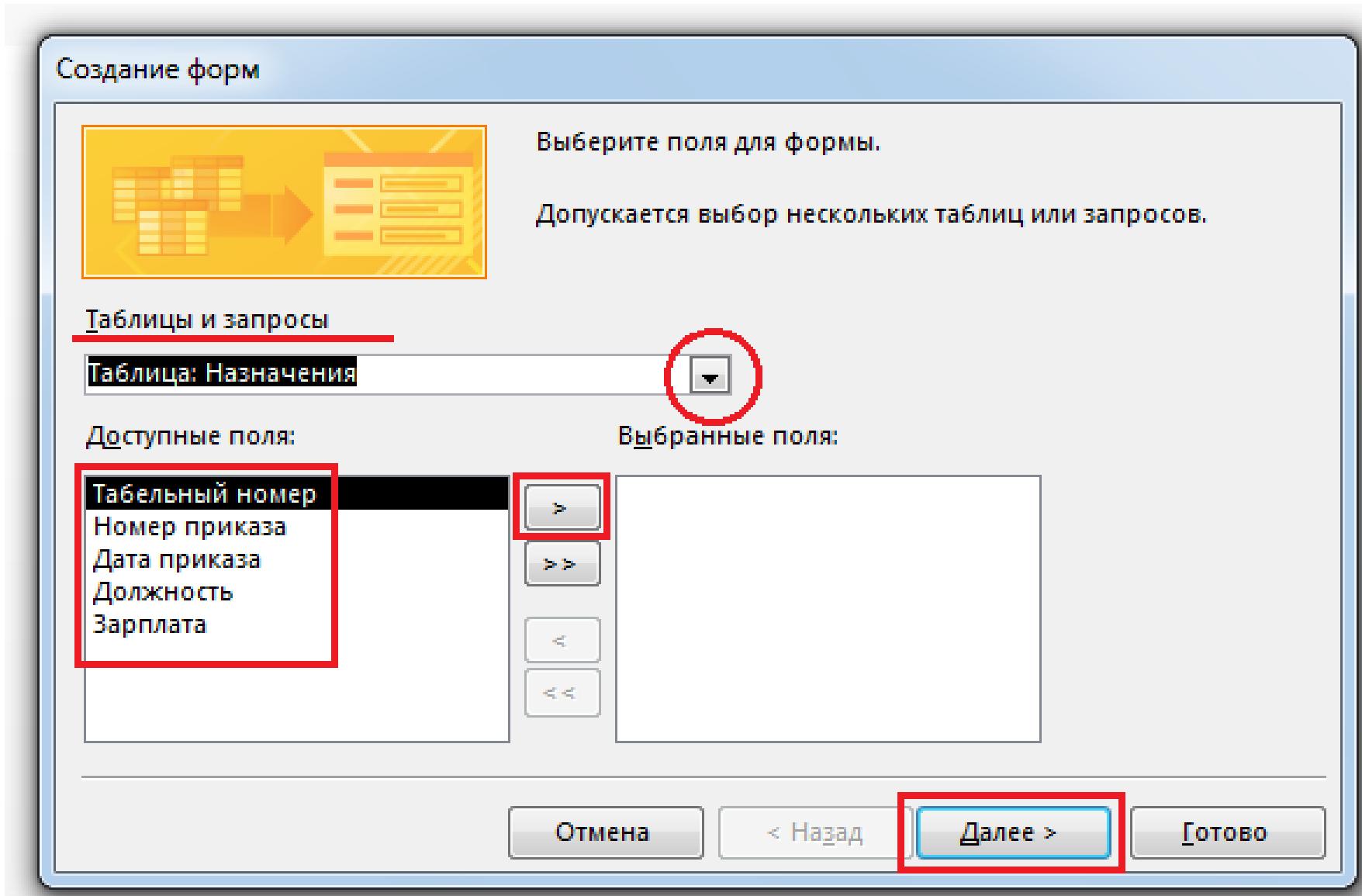
The screenshot shows the Microsoft Access application window with the 'Конструктор' (Designer) tab selected in the ribbon. The ribbon also includes tabs for 'Файл' (File), 'Главная' (Home), 'Создание' (Create), 'Внешние данные' (External Data), and 'Работа с базами данных' (Work with Databases). The 'Конструктор' tab has several icons: 'Выборка' (Query), 'Создание' (Create), 'Добавление' (Add), 'Обновление' (Update), 'Перекрестный' (Cross-Tab), 'Удаление' (Delete), 'Объединение' (Union), 'К серверу' (To Server), and 'Управление' (Manage). Below the ribbon, there are sections for 'Режим Выполнить' (Run Mode) and 'Тип запроса' (Query Type). The left pane displays a list of tables ('Анкета', 'Назначен...') and queries ('Запрос1', 'Запрос2', 'Запрос3', 'Запрос4', 'Запрос5', 'Запрос6', 'Запрос7', 'Запрос8', 'Запрос9'). The main workspace shows a query named 'Запрос1' with a title 'Анкета' containing fields: Табельный номер, ФИО, Дата рождения, Пол, Адрес, and Телефон. Below the query is a parameter sheet with fields: Поле: ФИО, Имя таблицы: Анкета, Сортировка: по возрастанию, Вывод на экран: или:, and Условие отбора: (checkboxes for each field). The status bar at the bottom indicates 'Белорусско-Российский университет Кафедра «Программное обеспечение информационных технологий»'.

# Создание форм

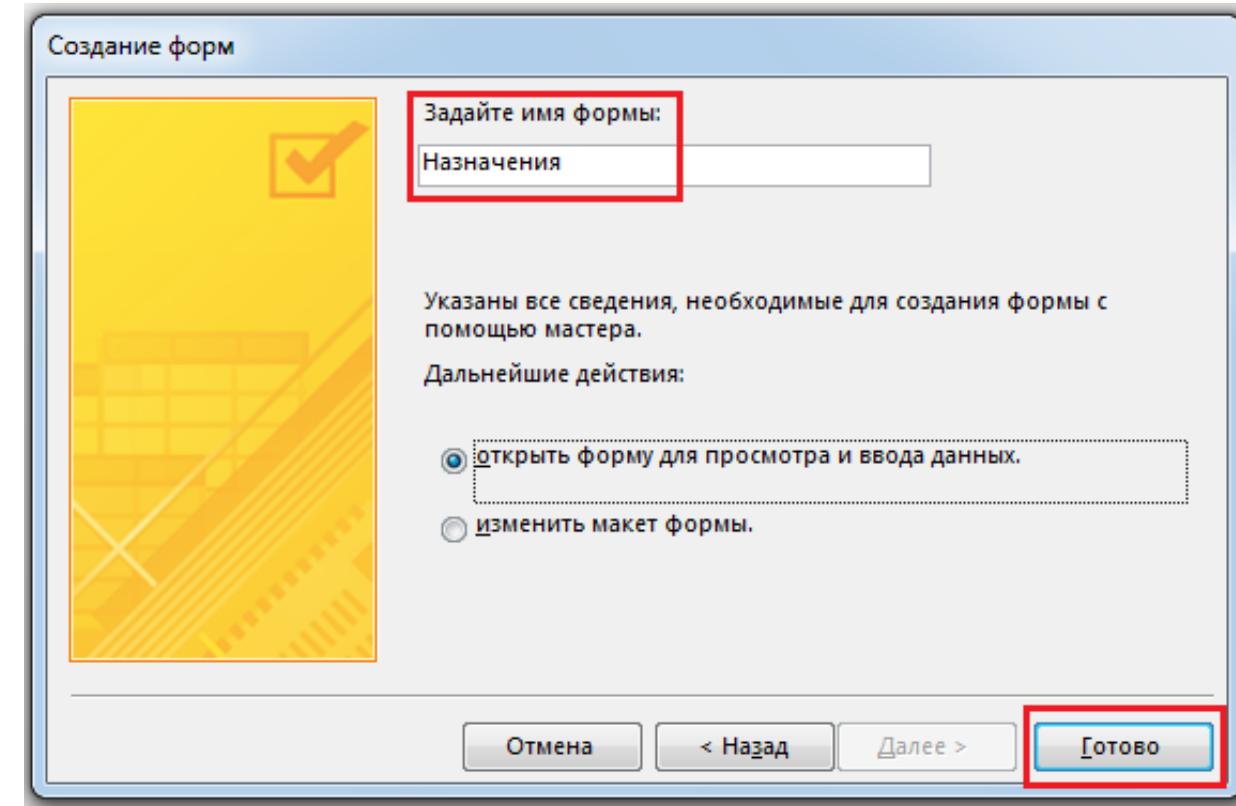
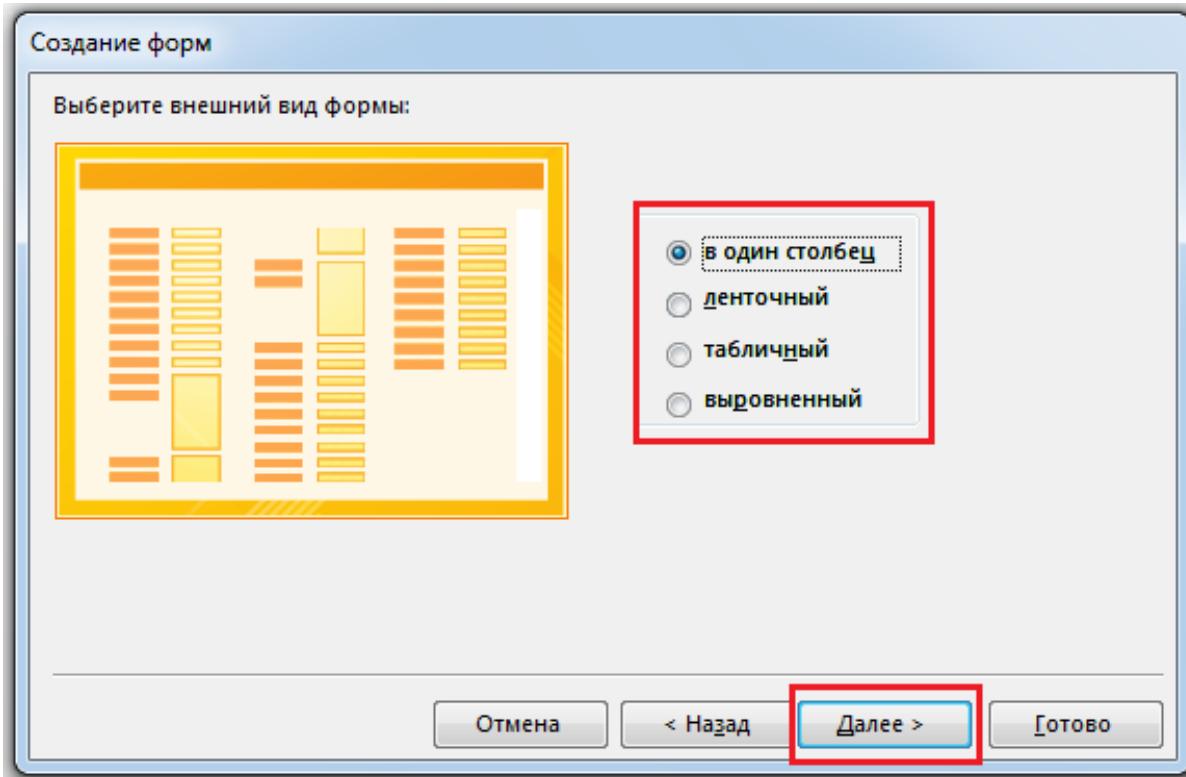
- Формы
- Формы позволяют отображать данные, содержащиеся в таблицах или запросах, в более удобном для восприятия виде.



# Создание форм



# Создание форм



# Создание форм

Назначения

Табельный номер	<input type="text"/> 1
Номер приказа	423
Дата приказа	11.01.2000
Должность	Директор
Зарплата	5 000,00 ₽

Запись: 1 из 5 ▶ ▶ ⏪ ⏩ Нет фильтра Поиск

# Создание форм

The screenshot shows the Microsoft Access ribbon with the 'ГЛАВНАЯ' (Home) tab selected, indicated by a red box. The 'Режим' (Mode) button in the 'Работа с базами данных' (Work with databases) group is also highlighted with a red box. A context menu is open over the 'Режим' button, with 'Конструктор' (Constructor) highlighted and also enclosed in a red box.

Below the ribbon, the main workspace displays a form titled 'Значения' (Values). The form contains fields for 'Табельный номер' (Employee number), 'Номер приказа' (Decree number), 'Дата приказа' (Date of decree), 'Должность' (Position), 'Зарплата' (Salary), and a calculated field 'Итог' (Total) which shows '5 000,00 ₽'.

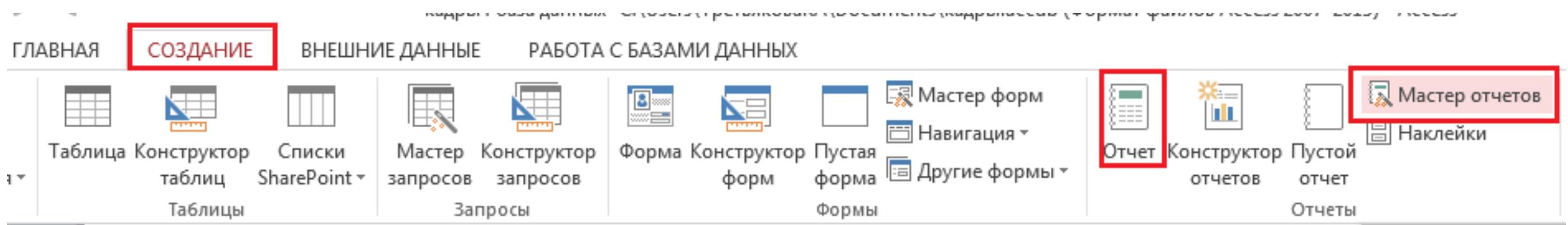
Поле	Значение
Табельный номер	1
Номер приказа	423
Дата приказа	11.01.2000
Должность	Директор
Зарплата	5 000,00 ₽
Итог	5 000,00 ₽

# Создание форм

The screenshot shows the Microsoft Access ribbon interface with the 'Инструменты конструктора форм' (Tools) tab selected, indicated by a red box. Below the ribbon, the 'Конструктор' (Constructor) tab is also highlighted in purple. The main area displays a data grid representing a form titled 'Назначения' (Assignments). The grid contains several fields: 'Табельный номер' (Employee number), 'Номер приказа' (Decree number), 'Дата приказа' (Date of decree), 'Должность' (Position), and 'Зарплата' (Salary). The 'Номер приказа' field is currently selected, showing its value 'Номер приказа'. The 'Элементы управления' (Controls) section of the ribbon shows various icons for adding controls like images, titles, and dates.

# Создание отчетов

- Отчеты
- Отчеты предназначены для печати данных, содержащихся в таблицах и запросах, в красиво оформленном виде.



# Создание отчетов

Инструменты конструктора отчетов

ФАЙЛ ГЛАВНАЯ СОЗДАНИЕ ВНЕШНИЕ ДАННЫЕ РАБОТА С БАЗАМИ ДАННЫХ КОНСТРУКТОР УПОРЯДОЧИТЬ ФОРМАТ ПАРАМЕТРЫ СТРАНИЦЫ

Режим  
Темы  
Шрифты  
Группировка  
Без подробностей  
Итоги  
аб  
Aa  
xxx  
XYZ  
Добавить изображение  
Номера страниц  
Эмблема  
Заголовок  
Дата и время  
Добавить пол

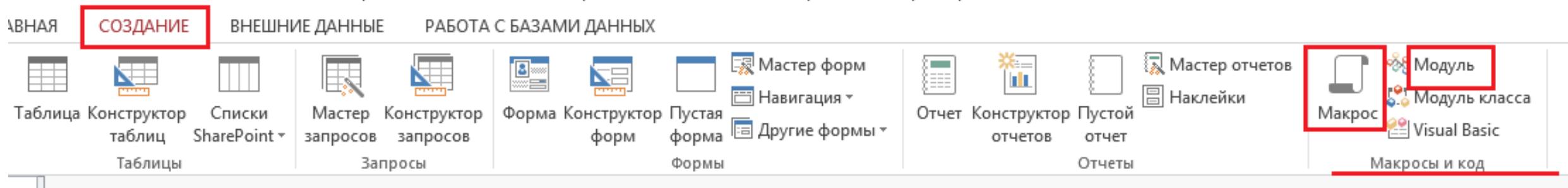
Представление отчета  
Предварительный просмотр  
Режим макета  
**Конструктор**  
Запрос2  
Запрос3  
Запрос4  
Запрос5

Группировка и итоги  
Элементы управления  
Колонтитулы  
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

Заголовок  
ФИО  
Адрес  
Область данных  
Табельный №  
ФИО  
Адрес

# Объекты

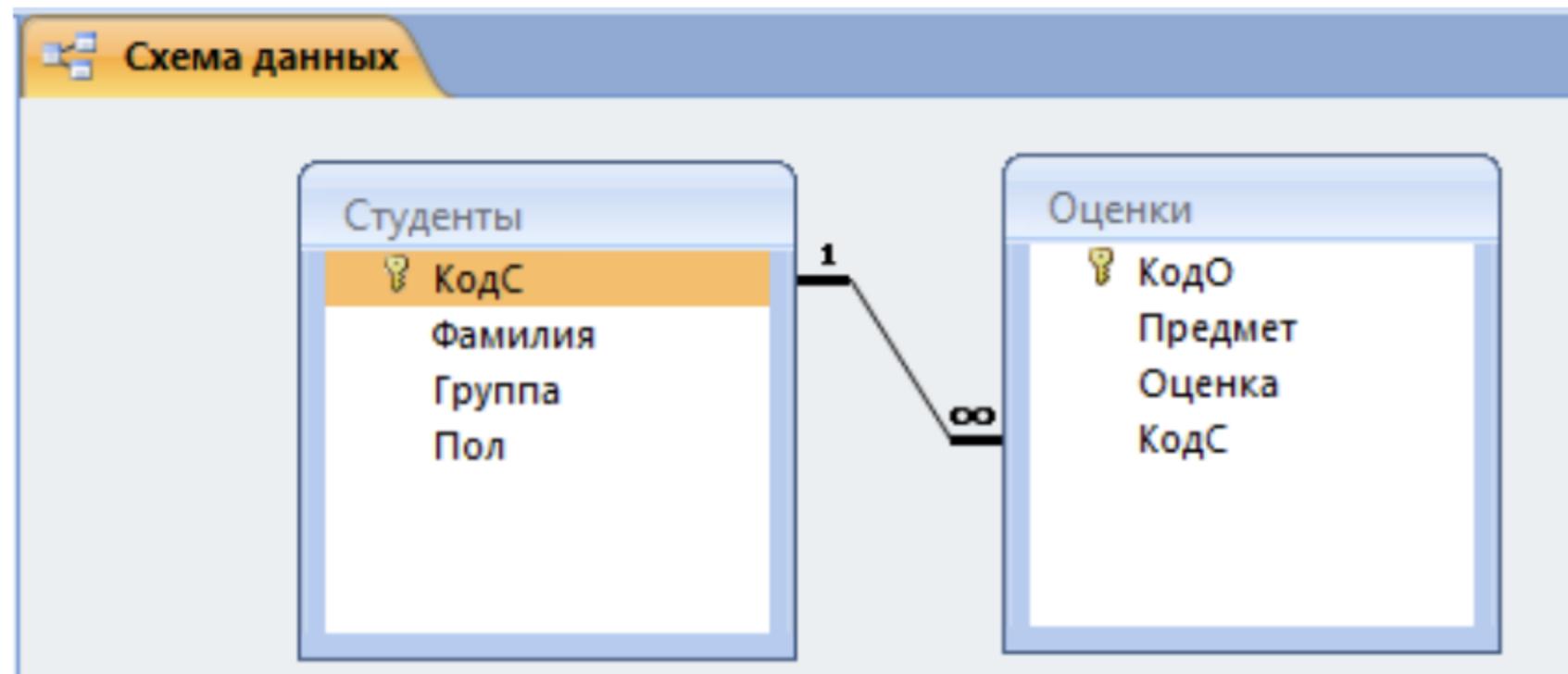
- **Макросы**
- Макросы служат для автоматизации повторяющихся операций.
- **Модули**
- Модули служат для автоматизации работы с БД. Модули еще называют процедурами обработки событий и пишутся на языке VBA.





# Пример создания запросов к базе данных Access на SQL

- Пример. Создание запросов к базе данных на SQL
- Схема данных:



# Таблицы

Студенты			
КодС	Фамилия	Группа	Пол
1	Иванов	801	М
2	Петров	801	М
3	Сидорова	801	Ж
4	Кузнецов	803	М
5	Малышева	803	Ж

Оценки				
КодО	Предмет	Оценка	КодС	
1	Информатика	4	1	
2	Информатика	3	2	
3	Алгебра	5	2	
4	Алгебра	4	5	
5	Алгебра	3	4	
6	Информатика	4	4	
7	Алгебра	2	3	
8	Информатика	3	5	

# Оператор SELECT

- Этот оператор реализует все операции реляционной алгебры.
- Синтаксис SELECT:

<b>SELECT</b> (ALL   DISTINCT)(<список_полей> *)	проекция
<b>FROM</b> <список_таблиц>	декартово произведение
<b>WHERE</b> <условие_выборки/соединения>	фильтрация/ условное соединение
<b>GROUP BY</b> <поля_группировки>	группировка
<b>HAVING</b> <условие_на_группу>	фильтрация групп
<b>ORDER BY</b> <поля_сортировки>	сортировка

# Оператор **SELECT**

- Пояснения:
  1. **ALL** – в результат включаются все строки запроса (в том числе одинаковые; по умолчанию);  
**DISTINCT** – в результат включаются только различные строки.
  2. \* означает выбор всех полей таблиц; иначе – только тех, которые указаны в разделе **SELECT**.
  3. В разделе **FROM** задается перечень исходных таблиц, над которыми выполняется декартово произведение.
  4. В разделе **WHERE** задаются условия запроса: либо условие выборки, либо условие соединения.

# Оператор **SELECT**

- Пояснения:
  5. В разделе **GROUP BY** задается список полей, по значениям которых выполняется группировка записей.
  6. В разделе **HAVING** задаются условия, накладываемые на каждую группу записей.
  7. В разделе **ORDER BY** перечисляются поля, определяющие порядок сортировки записей в результате.
  8. Имена полей – составные:  
**ИмяТаблицы.ИмяПоля**
  9. В разделе **SELECT** можно задавать псевдонимы полей (**AS**); также **AS** можно использовать в разделе **FROM** для задания псевдонимов таблиц.

# Условия (предикаты) в разделе WHERE

- Сравнения  $=$ ,  $<>$ ,  $<$ ,  $<=$ ,  $>$ ,  $>=$
- **Between A and B** – значение между А и В
- **Not Between A and B** – значение не между А и В
- **In (множество)** – принадлежность множеству
- **Not In (множество)** – непринадлежность множеству
- **Like “шаблон”** – сравнение с шаблоном (образцом):
  - \_ - любой одиночный символ
  - % - любая последовательность символов
  - другие символы обозначают сами себя
- **Is Null** – сравнение с неопределенным значением
- **Not Is Null** – отрицание неопределенного значения

01. Список всех студентов (копия таблицы Студенты)

```
SELECT *  
FROM Студенты;
```

	КодС	Фамилия	Группа	Пол
	1	Иванов	801	М
	2	Петров	801	М
	3	Сидорова	801	Ж
	4	Кузнецов	803	М
	5	Малышева	803	Ж

02. Фамилии  
студентов

SELECT Студенты.Фамилия  
FROM Студенты;

Запрос2	
	Фамилия
	Иванов
	Петров
	Сидорова
	Кузнецов
	Малышева

03. Список  
студентов в  
алфавитном по-  
рядке

```
SELECT Студенты.Фамилия  
FROM Студенты  
ORDER BY Студенты.Фамилия;
```

Запрос3	
	Фамилия
	Иванов
	Кузнецов
	Малышева
	Петров
	Сидорова

#### 04. Список студентов группы 801

```
SELECT Студенты.Фамилия, Студенты.Группа  
FROM Студенты  
WHERE Студенты.Группа="801";
```

Фамилия	Группа
Иванов	801
Петров	801
Сидорова	801

## 05. Список оценок

```
SELECT Оценки.Предмет, Оценки.Оценка,  
Оценки.КодС  
FROM Оценки  
ORDER BY Оценки.Предмет;
```

Предмет	Оценка	КодС
Алгебра	2	3
Алгебра	3	4
Алгебра	4	5
Алгебра	5	2
Информатика	3	5
Информатика	4	4
Информатика	3	2
Информатика	4	1

Хотелось бы  
видеть фамилии!  
Но они в таблице  
Студенты

## 06. Список студентов и их оценок

```
SELECT *  
FROM Студенты , Оценки ; !!!
```

КодО	Предмет	Оценка	Оценки.Код	Студен	Фамилия	Группа	Пол
1	Информатика	4	1	1	Иванов	801	М
1	Информатика	4	1	2	Петров	801	М
1	Информатика	4	1	3	Сидорова	801	Ж
1	Информатика	4	1	4	Кузнецов	803	М
1	Информатика	4	1	5	Малышева	803	Ж
2	Информатика	3	2	1	Иванов	801	М
2	Информатика	3	2	2	Петров	801	М
2	Информатика	3	2	3	Сидорова	801	Ж
2	Информатика	3	2	4	Кузнецов	803	М
2	Информатика	3	2	5	Малышева	803	Ж
3	Алгебра	5	2	1	Иванов	801	М
3	Алгебра	5	2	2	Петров	801	М
3	Алгебра	5	2	3	Сидорова	801	Ж
3	Алгебра	5	2	4	Кузнецов	803	М
3	Алгебра	5	2	5	Малышева	803	Ж
4	Алгебра	4	5	1	Иванов	801	М
4	Алгебра	4	5	2	Петров	801	М
4	Алгебра	4	5	3	Сидорова	801	Ж
4	Алгебра	4	5	4	Кузнецов	803	М
4	Алгебра	4	5	5	Малышева	803	Ж
5	Алгебра	3	4	1	Иванов	801	М
5	Алгебра	3	4	2	Петров	801	М
5	Алгебра	3	4	3	Сидорова	801	Ж
5	Алгебра	3	4	4	Кузнецов	803	М
5	Алгебра	3	4	5	Малышева	803	Ж
6	Информатика	4	4	1	Иванов	801	М
6	Информатика	4	4	2	Петров	801	М
6	Информатика	4	4	3	Сидорова	801	Ж

Не вышло, т.к. не  
было указано, как  
соединяются  
записи двух таблиц  
(каждая с каждой)



## 07. Список студентов и их оценок (еще попытка)

```
SELECT Студенты.Фамилия, Оценки.Предмет, Оценки.Оценка  
FROM Студенты,Оценки  
WHERE Студенты.КодС = Оценки.КодС;
```

Фамилия	Предмет	Оценка
Иванов	Информатика	4
Петров	Информатика	3
Петров	Алгебра	5
Сидорова	Алгебра	2
Кузнецов	Алгебра	3
Кузнецов	Информатика	4
Малышева	Алгебра	4
Малышева	Информатика	3

Теперь  
правильно

## 08. Список предметов

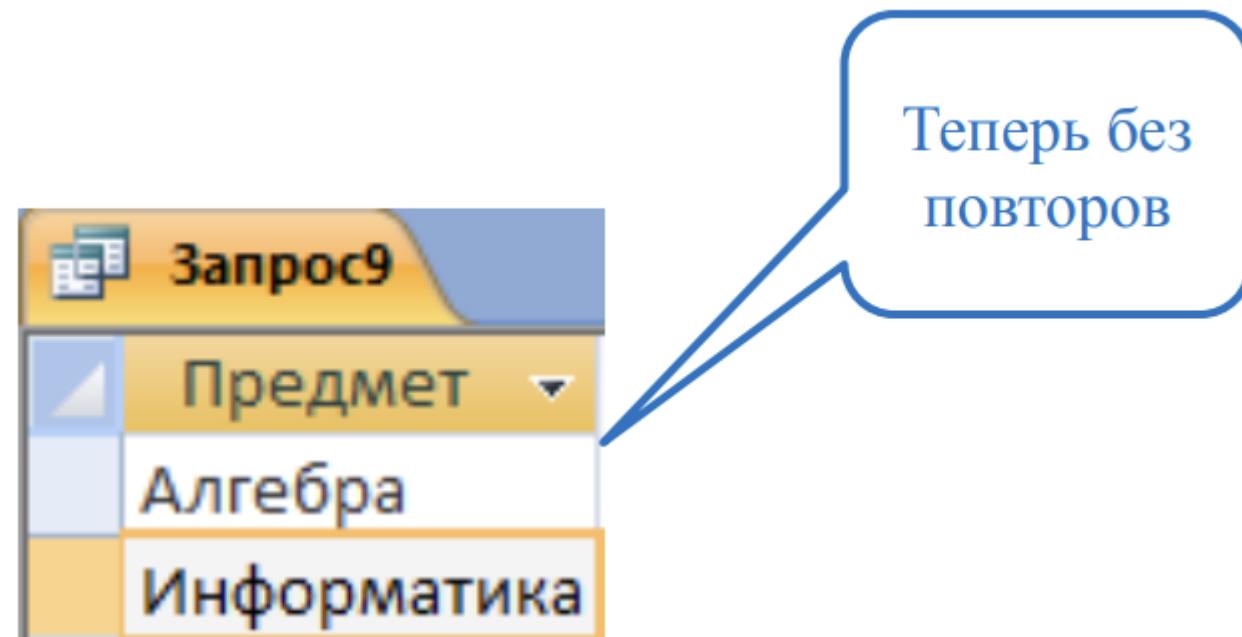
```
SELECT Оценки.Предмет  
FROM Оценки;
```

Предмет
Информатика
Информатика
Алгебра
Алгебра
Алгебра
Информатика
Алгебра
Информатика

Но здесь  
есть  
повторы!

## 09. Список предметов (без повторов)

```
SELECT DISTINCT Оценки.Предмет  
FROM Оценки;
```



## 10. Все оценки студента Петрова

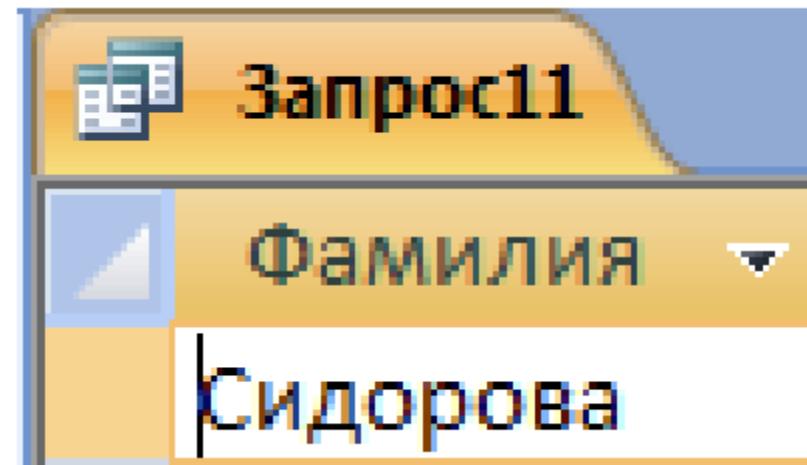
```
SELECT Студенты.Фамилия, Оценки.Предмет, Оценки.Оценка  
FROM Студенты, Оценки  
WHERE (Студенты.КодС = Оценки.КодС)  
AND (Студенты.Фамилия="Петров");
```

The screenshot shows a Microsoft Access query results table titled "Запрос10". The table has three columns: "Фамилия" (Family Name), "Предмет" (Subject), and "Оценка" (Grade). There are two rows of data. The first row shows "Петров" in the "Фамилия" column, "Информатика" in the "Предмет" column, and the grade "3" in the "Оценка" column. The second row shows "Петров" again in the "Фамилия" column, "Алгебра" in the "Предмет" column, and the grade "5" in the "Оценка" column.

Фамилия	Предмет	Оценка
Петров	Информатика	3
Петров	Алгебра	5

11. Список студентов, у которых есть хотя бы одна двойка

```
SELECT DISTINCT Студенты.Фамилия  
FROM Студенты , Оценки  
WHERE (Студенты.КодС = Оценки.КодС)  
AND (Оценки.Оценка=2);
```



## 12. Переименование столбца

```
SELECT Студенты.Фамилия, Студенты.Группа AS НомерГруппы  
FROM Студенты;
```

Фамилия	НомерГруппы
Иванов	801
Петров	801
Сидорова	801
Кузнецов	803
Малышева	803

# Группировка и агрегатные функции

- **Группировка** – разбиение всего множества записей таблицы на группы, в которые собираются записи, имеющие одинаковые значения полей группировки.
- Если группировать записи по полю «Пол», то они будут разбиты на две группы записей («М» и «Ж»).
- Если группировать записи по полю «Группа», то они будут разбиты на две группы записей (801 и 803).

Студенты				
	КодС	Фамилия	Группа	Пол
1	1	Иванов	801	М
2	2	Петров	801	М
3	3	Сидорова	801	Ж
4	4	Кузнецов	803	М
5	5	Малышева	803	Ж

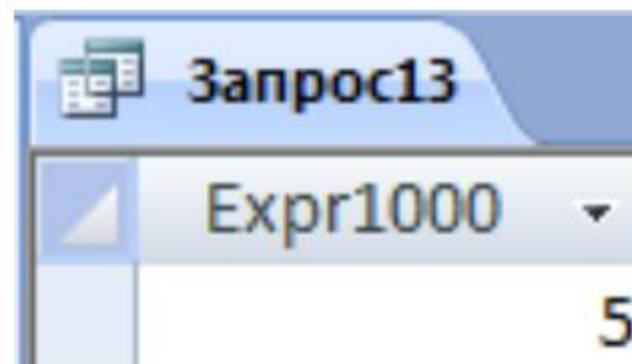
# Агрегатная функция

- Агрегатная функция – вычисляет обобщенное групповое значение для всех записей каждой группы.
- Агрегатные функции используются подобно именам полей в SELECT
- Аргументами агрегатных функций являются имена полей (либо \* для COUNT)
- При использовании группировки в разделе SELECT можно использовать только имена полей группировки либо агрегатные функции.
- Можно применять агрегатные функции и без группировки, тогда вся таблица рассматривается как одна группа.

ФУНКЦИЯ	РЕЗУЛЬТАТ
COUNT()	Количество записей группы / непустых значений поля
SUM()	Сумма
AVG()	Среднее арифметическое
MIN()	Наименьшее
MAX()	Наибольшее

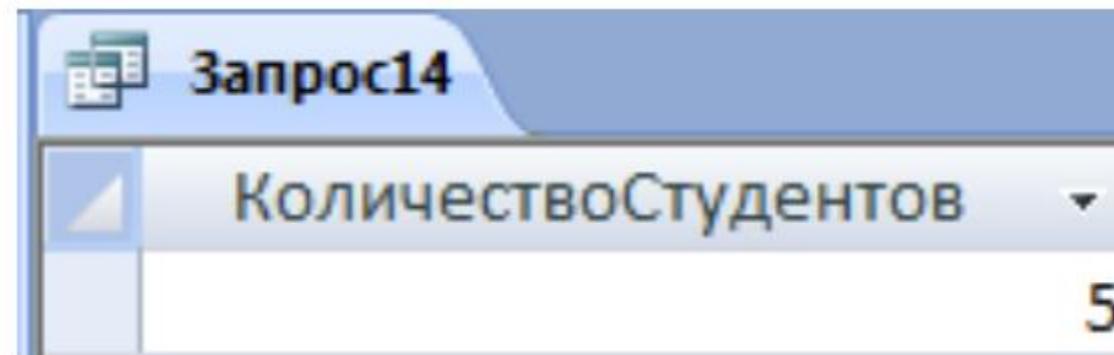
### 13. Общее количество студентов

```
SELECT COUNT(*)  
FROM Студенты;
```



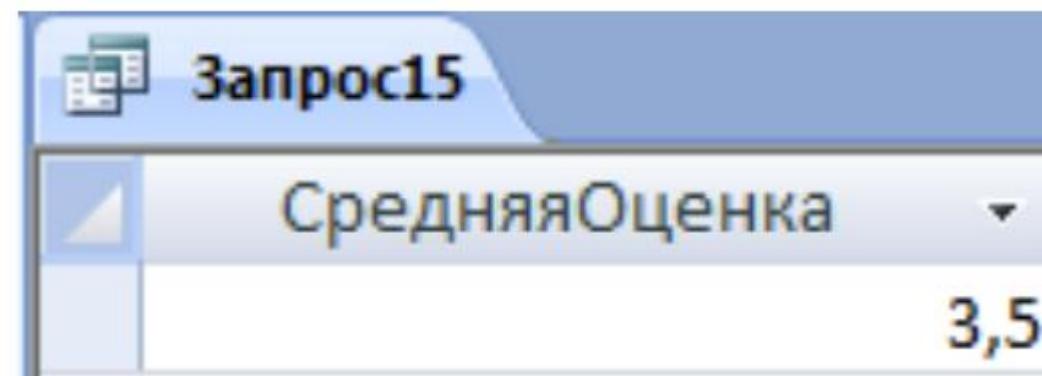
14. Общее количество студентов (заголовок столбца)

```
SELECT COUNT(*) AS КоличествоСтудентов  
FROM Студенты;
```



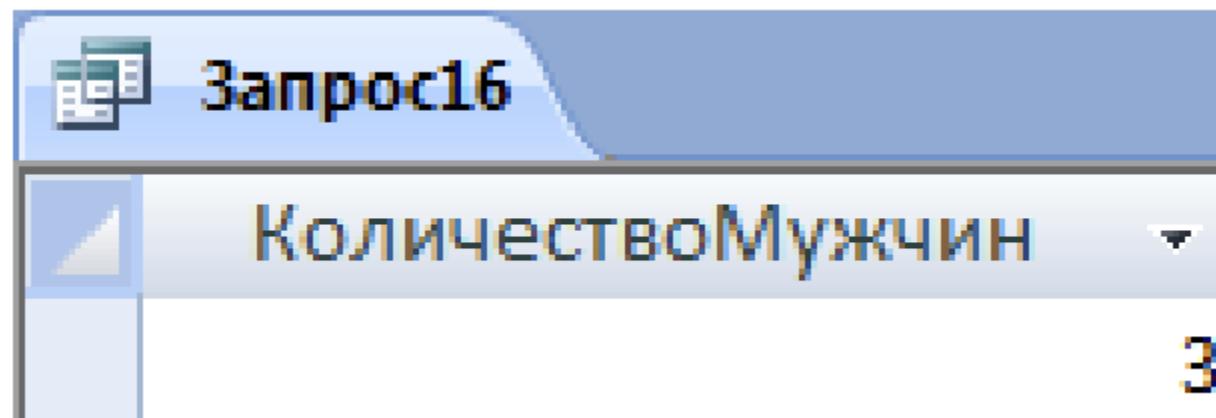
15. Средняя оценка всех студентов

```
SELECT AVG(Оценка) AS СредняяОценка  
FROM Оценки;
```



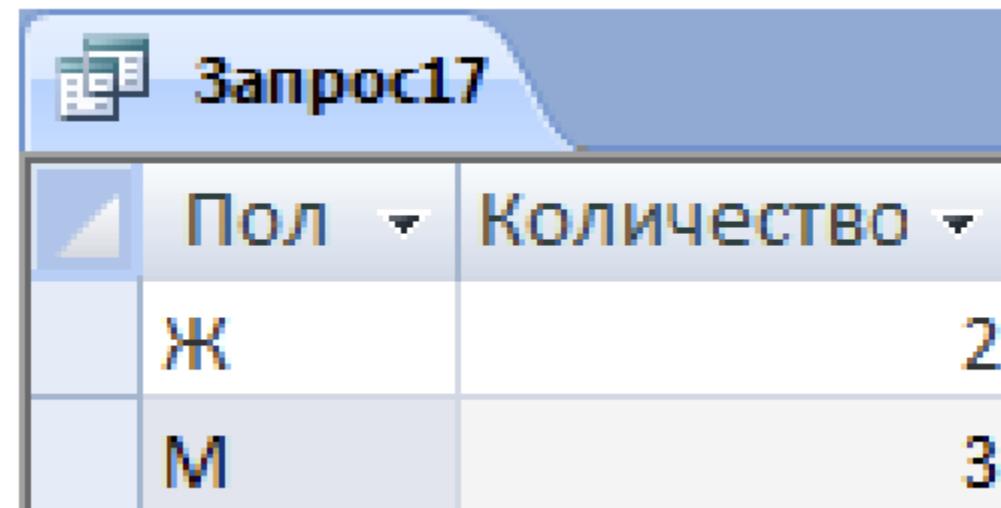
## 16. Количество студентов-мужчин

```
SELECT COUNT(*) AS КоличествоМужчин  
FROM Студенты  
WHERE Пол="М";
```



## 17. Количество студентов каждого пола

```
SELECT Пол, COUNT(*) AS Количество  
FROM Студенты  
GROUP BY Пол;
```



The screenshot shows a Microsoft Access query window titled "Запрос17". The table has two columns: "Пол" (Gender) and "Количество" (Count). There are two rows: one for female ("Ж") with a count of 2, and one for male ("М") with a count of 3.

Пол	Количество
Ж	2
М	3

## 18. Средняя оценка в каждой группе

```
SELECT Студенты.Группа, Avg(Оценки.Оценка) AS  
    СредняяОценка  
FROM Студенты, Оценки  
WHERE Студенты.КодС=Оценки.КодС  
GROUP BY Студенты.Группа;
```

The screenshot shows a Microsoft Access query results window titled "Запрос18". The columns are "Группа" (Group) and "СредняяОценка" (Average Grade). The data shows two rows: Group 801 with an average grade of 3,5, and Group 803 with an average grade of 3,5.

Группа	СредняяОценка
801	3,5
803	3,5

## 19. Средняя оценка каждого студента

```
SELECT MAX(Студенты.Фамилия) AS Фамилия,  
       AVG(Оценки.Оценка) AS СредняяОценка  
  FROM Студенты, Оценки  
 WHERE Студенты.КодС = Оценки.КодС  
 GROUP BY Студенты.КодС;
```

Фамилия	СредняяОценка
Иванов	4
Петров	4
Сидорова	2
Кузнецов	3,5
Малышева	3,5



# Информатика

Тема: Базы данных и системы управления базами  
данных

**Благодарю  
за внимание**

**КУТУЗОВ** Виктор Владимирович

# Список использованных источников

1. Рабочая программа дисциплины «Информатика» / Кутузов В.В. – Могилев : Белорусско-Российский университет, 2023
2. Фотографии и картинки взяты с сайтов Яндекс.Картинки и Гугл.Картинки, иконки с flaticon.com
3. Навроцкий Артем - Лекции по СУБД для Технопарка  
<https://github.com/bozaro/tech-db-lectures>  
<https://bozaro.github.io/tech-db-lectures/>  
<https://tech-db-lectures.bozaro.ru/master/>  
<https://www.youtube.com/playlist?list=PLrCZzMib1e9oOFQbuOgjKYbRUoA8zGKnj>
4. ЕРАМ - Базы данных и язык SQL  
<https://ppt-online.org/6544>
5. Виды баз данных  
<https://selectel.ru/blog/databases-types/>
6. Ультимативная дорожная карта для изучения SQL и баз данных в 2023 году + источники для знаний  
<https://habr.com/ru/articles/725414/>
7. Как изучать SQL в 2023 году  
<https://habr.com/ru/amp/publications/725166/>
8. Нормализация форм БД  
<http://surokIxj.bget.ru/sibgau/БД%20Нормализация%20форм.pdf>
9. Язык SQL. Вводная лекция. Введение в теорию баз данных (Е. П. Моргунов)  
<https://edu.postgrespro.ru/sqlprimer/sqlprimer-2018-msu-00.pdf>

# Список использованных источников

10. НОУ «ИНТУИТ» Введение в реляционные базы данных. Лекция 2: Введение в реляционную модель данных  
<https://intuit.ru/studies/courses/74/74/lecture/27903?page=1>
11. НОУ «ИНТУИТ» Введение в реляционные базы данных. Лекция 3: Базисные средства манипулирования реляционными данными: реляционная алгебра Кодда  
<https://intuit.ru/studies/courses/74/74/lecture/27905?page=1>
12. Сравнение SQL- и NoSQL-баз данных  
<https://habr.com/ru/companies/ruvds/articles/727474/>
13. Кубил В. Н. Лекции по дисциплине «Базы данных»  
<https://ppt-online.org/1077001>
14. Введение в базы данных  
<https://github.com/maxchv/itstep/blob/master/databases/presentation/01.intro.rst>
15. Принципы логического проектирования БД  
<https://theslide.ru/uncategorized/principy-logicheskogo-proektirovaniya-bd>
16. Архитектура базы данных  
<https://en.ppt-online.org/256428>
17. Базы данных. Язык SQL, его структура, стандарты  
<https://ppt-online.org/317286>
18. Курс «Основы баз данных и SQL» Глава 7 (продолжение II). Реляционная алгебра. Язык SQL  
[https://java-online-course.github.io/course/materials/db\\_basics/presentations/ОИТ%20=%20Лекция%20076%20=%20Язык%20SQL.pdf](https://java-online-course.github.io/course/materials/db_basics/presentations/ОИТ%20=%20Лекция%20076%20=%20Язык%20SQL.pdf)

# Список использованных источников

19. Учебник. Сергей Моисеенко. SQL Задачи и решения. Интерактивный учебник по SQL  
<http://www.sql-tutorial.ru>
20. Практическое владение языком SQL  
<https://sql-ex.ru>
21. Онлайн Курс обучения программированию на Java  
<https://java-online-course.github.io/course/about>
22. Онлайн Курс «Основы баз данных и SQL»  
[https://java-online-course.github.io/course/materials/db\\_basics/db\\_basics](https://java-online-course.github.io/course/materials/db_basics/db_basics)
23. SQL Tutorial  
<https://www.w3schools.com/sql/default.asp>
24. Обработка информации средствами Microsoft Access  
[https://www.fond21veka.ru/publication/?download\\_file=67918](https://www.fond21veka.ru/publication/?download_file=67918)
25. Учебник по языку SQL (DDL, DML) на примере диалекта MS SQL Server. Часть пятая  
<https://habr.com/ru/articles/256169/>
26. Попов А. Короткий вводный курс в базы данных: основные понятия, модели и механизмы.  
<https://andpop.ru/courses/dbms.shtml>