

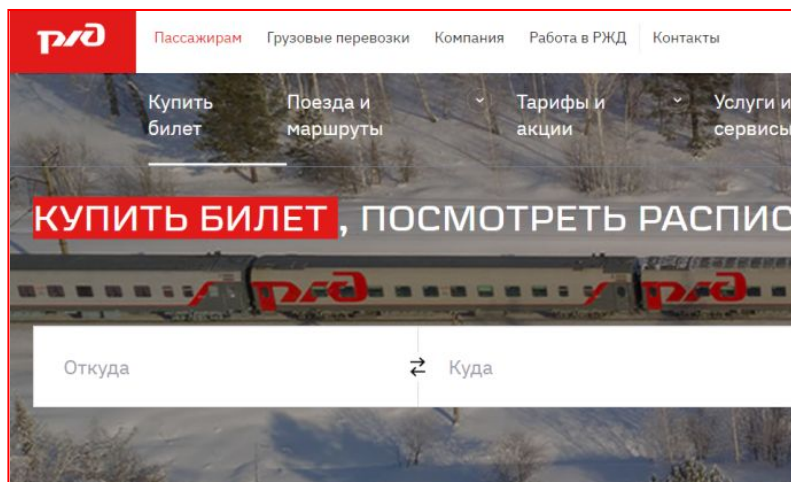
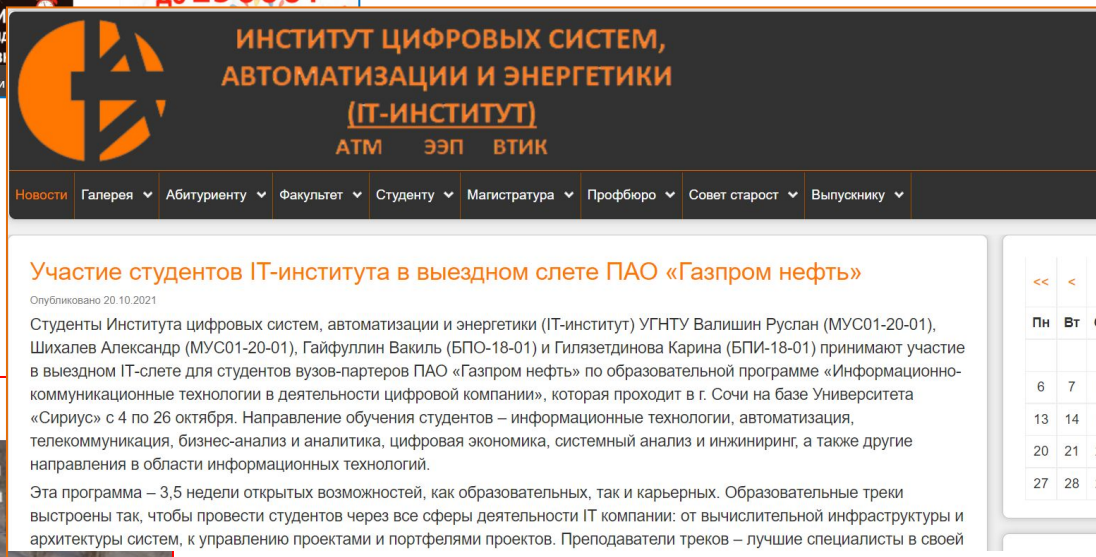
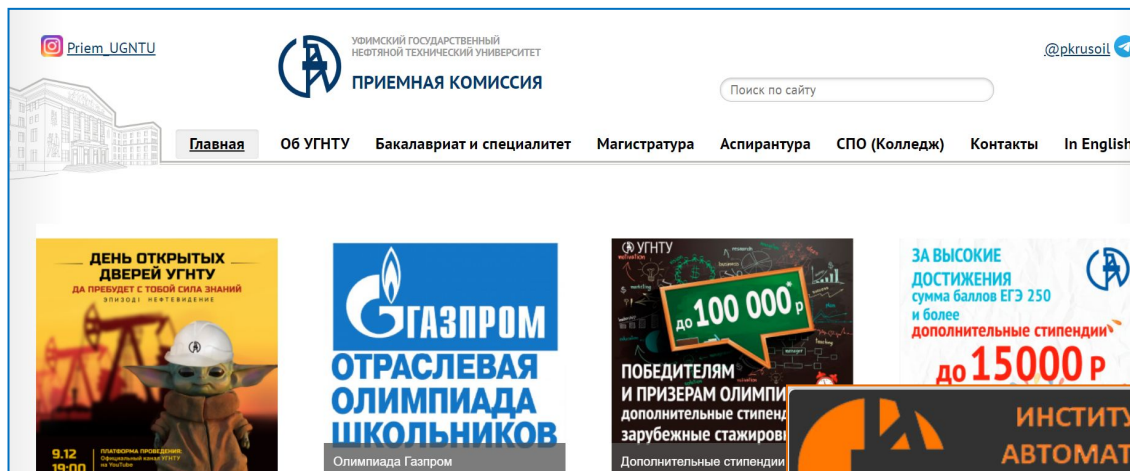


# Базы данных



1) Практически в каждом приложении реализована БД

1) Практически в каждом приложении реализована БД



## 2) Почти в каждой вакансии упоминается SQL (Structured query language — «язык структурированных запросов»)

Будьте первыми

Аналитик данных

Газпромбанк ✓

Уфа

Работать с отчетностью 2 линии поддержки: определение структуры потока обращений клиентов, отслеживание отклонений KPI, поиск закономерностей и зон роста.

...Query, Power Pivot, Power BI, SSAS Tabular, T-SQL (MS SQL Server) / SQL (Oracle). Понимает методы и подходы к анализу...

Откликнуться

1 декабря

Отклик без резюме

Будьте первыми

Старший аналитик (Regular Middle)

от 126 500 руб.

Можно работать из дома

Группа Компаний Астрал ✓

Уфа

Осуществлять сбор требований от Заказчиков/экспертов предметных областей. Проектировать интерфейс медицинской системы, формировать требования по интеграционному взаимодействию между системами на...

Знание реляционных БД, опыт написания простых SQL-запросов. Понимание структур XML, JSON и т.д. Опыт работы с REST на...

Откликнуться

1 декабря

Будьте первыми

Главный технолог / Бизнес-аналитик IT

АО МОСКОВСКИЙ ОБЛАСТНОЙ БАНК ✓

Уфа

Формирование требований по доработкам контрольных процедур на этапе формирования витрины CRE. Ведение базы ошибок, возникающих при отправке данных в БКИ...

Знание ПО Credit Registry. Аналитический склад ума и умение алгоритмизировать процессы. Хорошее владение PL/SQL, Power...

Откликнуться

1 декабря

Отклик без резюме

Junior Web-разработчик / Инженер-программист

ООО Витасмарт ✓

Уфа

Сопровождение системы учёта лиц, застрахованных по ОМС (C#.net, JS, Oracle). Доработка уже существующих и создание новых приложений...

SQL (Oracle). PL/SQL. Языки программирования C#, JS, html (css, jQuery). Будут плюсом: навыки (опыт работы) в WEB...

Откликнуться

Показать контакты

1 декабря

Отклик без резюме

Будьте первыми

Разработчик программного обеспечения (Python)

ООО Фродекс ✓

Уфа

Компания "Фродекс" - разработчик системы обнаружения мошеннических платежей и противодействия отмыванию денег. Каждый день мы проверяем сотни тысяч...

Понимание микросервисной архитектуры и serverless-подхода. Опыт работы с SQL и NoSQL базами данных (PostgreSQL, Redis).

Откликнуться

Показать контакты

1 декабря

PL/pgSQL-разработчик со знанием frontend (middle), удалённо

от 100 000 руб.

Можно работать из дома

ООО БизнесПомощь ✓

Уфа

В настоящее время мы разрабатываем многофункциональную CRM-платформу для эффективной работы с нашими клиентами и партнёрами, ведения внутренней документации и...

Рабочий стек и инструменты: PostgreSQL, PgAdmin, pl/pgsql, SQL, JSON и JSONB, Bootstrap, собственная админка, Postman. Мы рассчитываем, что вы...

Откликнуться

Показать контакты

1 декабря

QA Engineer

80 000 – 140 000 руб.

Можно работать из дома

ООО Финном Технологии ✓

Уфа

Тестировать в стартапе в области финтеха. Выполнять функциональное и регрессионное тестирование. Разрабатывать тестовые сценарии и пополнять тестовую документацию.

Иметь опыт в написании тест-кейсов. Базовые знания клиент-серверных приложений. SQL на базовом уровне для анализа ошибок и данных.

Работодатель сейчас онлайн

Откликнуться

1 декабря



3) Базы данных позволяют хранить большие объемы данных

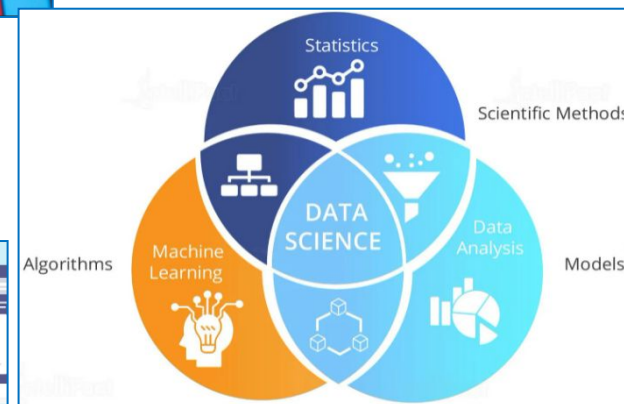
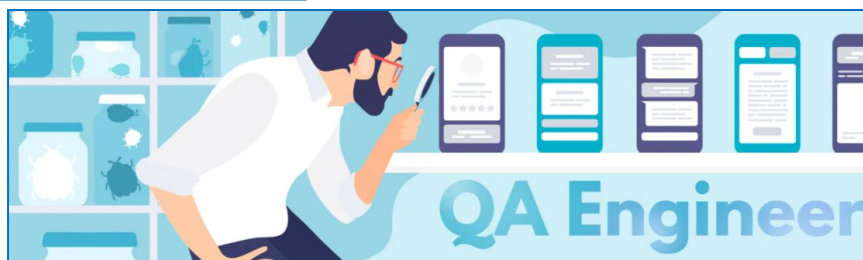


4) Возможность анализ накопленных данных.



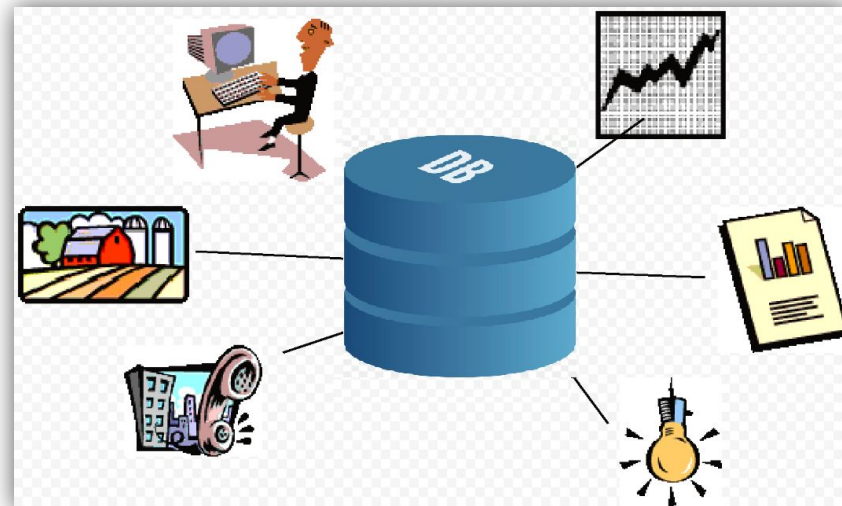
# Профессии, требующие знания БД

1. Разработчик программного обеспечения.
2. Аналитик данных (Data Analyst).
3. Data Scientist.
4. QA инженер (Quality Assurance обеспечение качества ).
5. Project Manager.
6. DevOps.



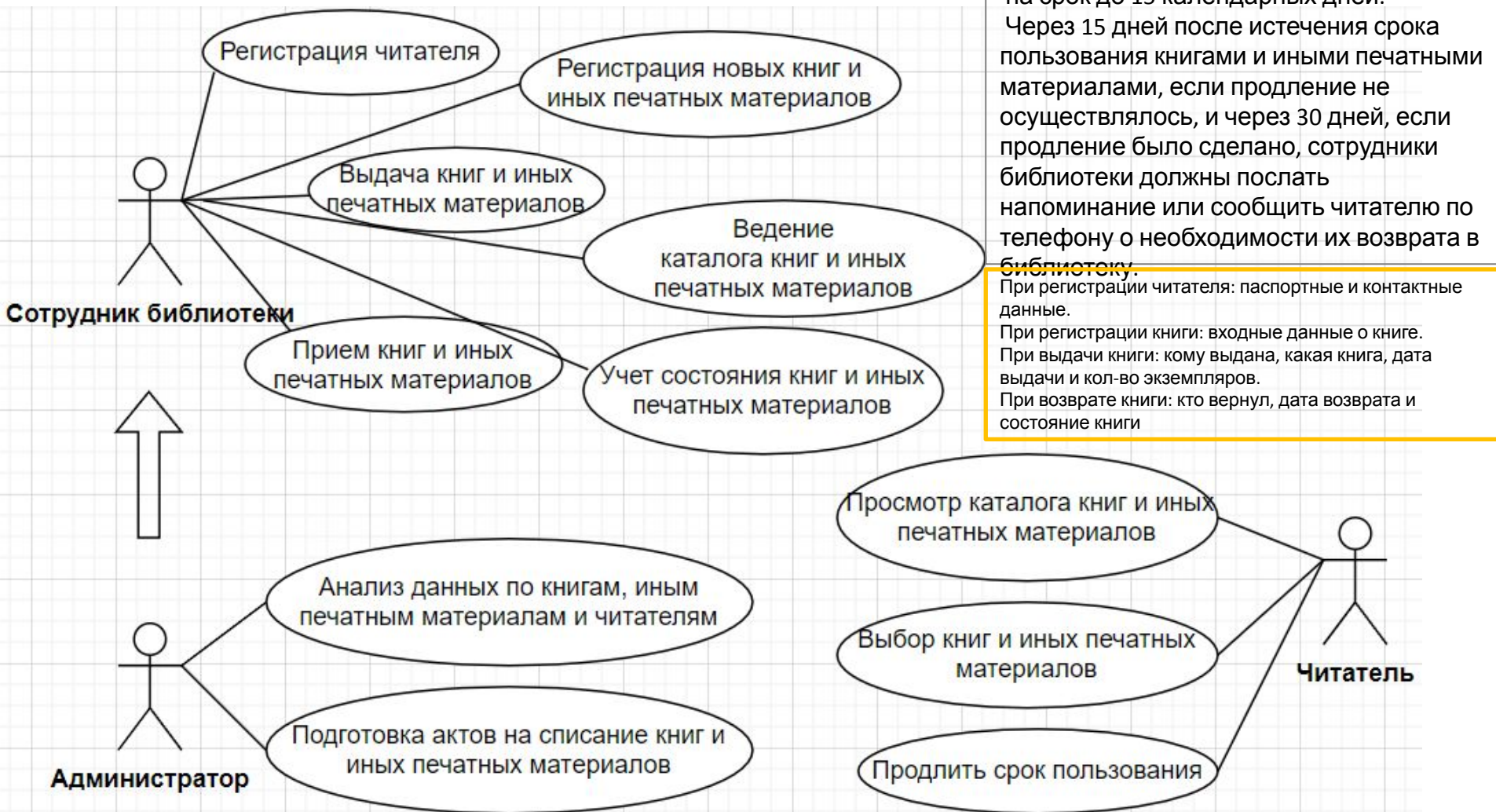
# Основные понятия БД

- данные;
- предметная область;
- бизнес – правила.





# области «Муниципальная библиотека»

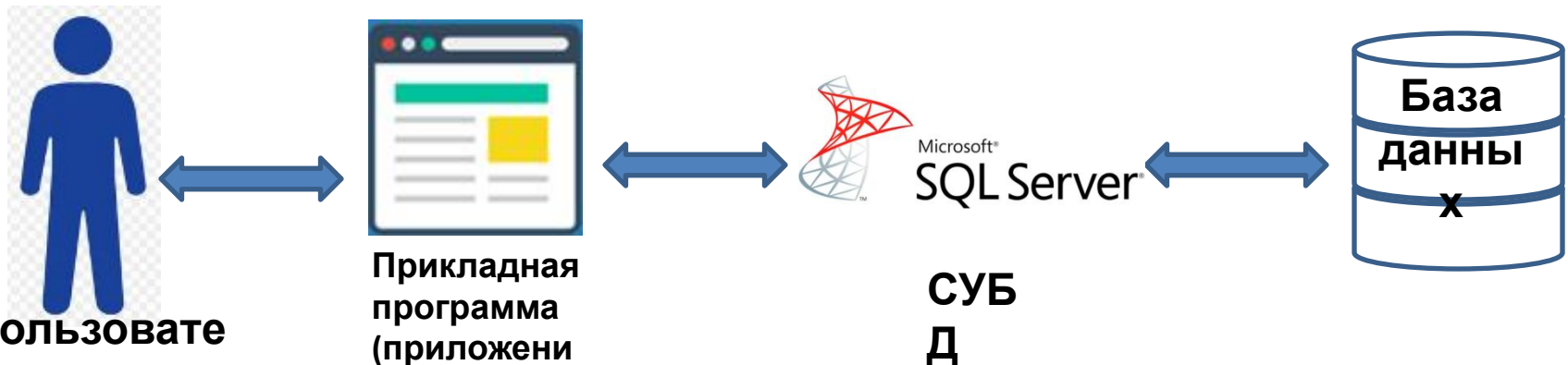


# Основные термины

**База данных (БД) (Database, BD)** – это организованная совокупность данных о некоторой предметной области, предназначенная для длительного хранения и постоянного применения.



**Система управления базой данных (СУБД) (Database Management System, DBMS)** – это программное обеспечение для работы с БД, т.е. совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями.





# Задачи, которые решает СУБД

1. Надежное хранение данных.
2. Быстрый поиск нужной информации.
3. Многопользовательский доступ.
4. Разграничение прав доступа.
5. Доступ к базе данных по сети.
6. Понятный для работы с данными язык SQL (Structured Query Language).



Например, нужно вывести фамилии и имена сотрудников проживающих в городе Уфа из таблицы Сотрудники:

SQLQuery1.sql -...OX\admin (53))\*

```
SELECT Surname, Name, City  
FROM Cooperator  
WHERE City = 'Уфа'
```

Результаты   Сообщения

	Surname	Name	City
1	Иванов	Егор	Уфа
2	Егорова	Анастасия	Уфа
3	Федоровская	Анита	Уфа



# Информационная система

**Информационная система** – это система, реализующая автоматизированный сбор, хранение, поиск, извлечение и модификацию данных и включающая технические средства обработки данных, программное обеспечение и соответствующий персонал.



Информационная система = БД + СУБД

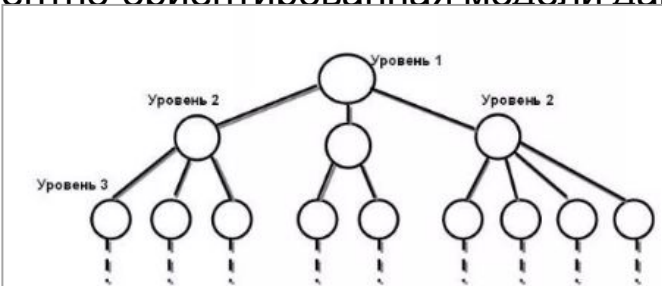
# Классификация баз данных

## 1) По модели данных

**Модель данных** – это метод (принцип) логической организации данных, используемый СУБД.

По способу установления связей между данными исторически сложились 3-и классические модели: **иерархическая, сетевая, реляционная**.

Далее появились постреляционная, многомерная, объектно-ориентированная, объектно-реляционная, документно-ориентированная модели данных и др.



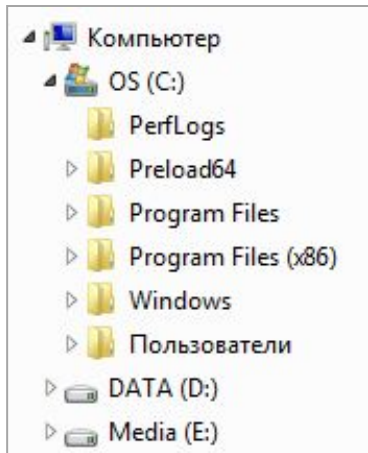
Графическое представление иерархической модели данных

Имя поля 1	Имя поля 2	Имя поля 3	Имя поля 4

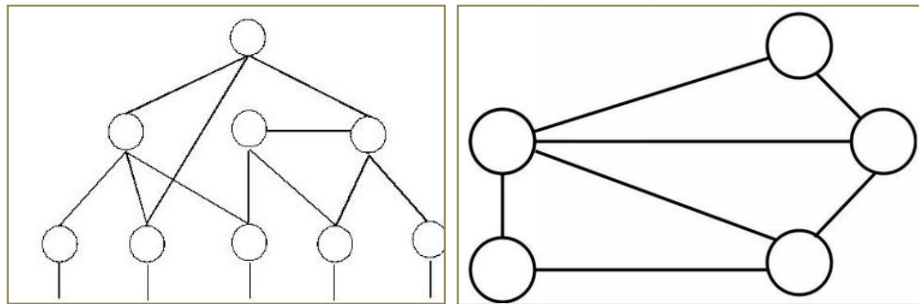
Запись (указывает на строку)

Поле (указывает на столбец)

Структура таблицы реляционной БД



Иерархическая модель схожа по принципу построения с файловой системой

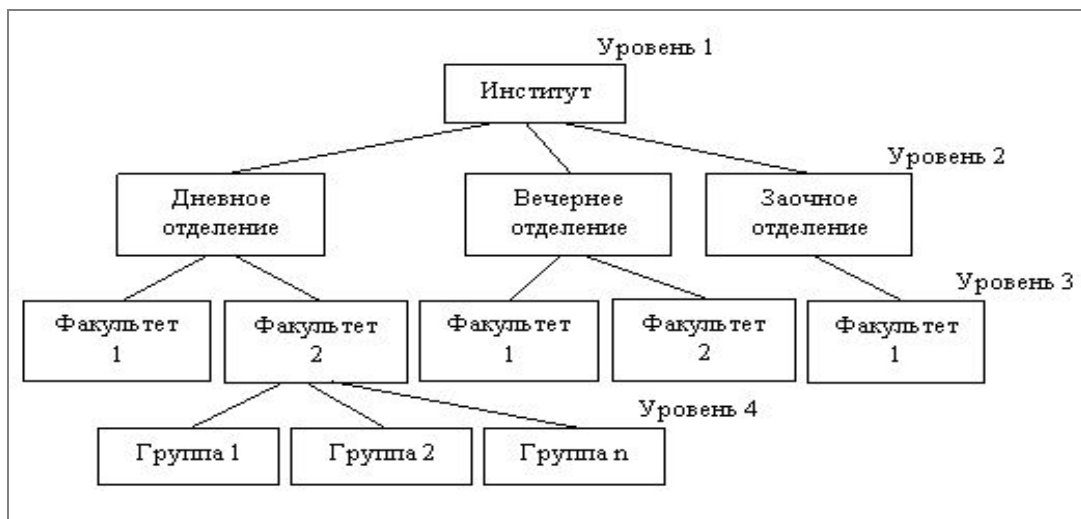
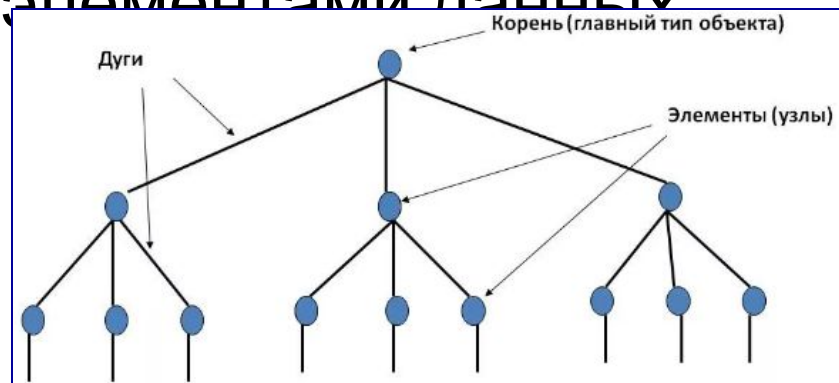


Графические представления сетевой модели данных



# Иерархическая модель данных

**Иерархическая модель** данных имеет форму дерева с дугами-связями и узлами-элементами данных



# Сетевая модель данных

**Сетевую модель** данных можно рассматривать как расширенную версию иерархической модели.

Основное различие между иерархической и сетевой моделью состоит в том, что в сетевой модели запись может иметь связи со многими другими записями, а не только с одной родительской.



# Реляционная модель данных

**Реляционная база данных** – это набор простых таблиц (отношений, сущностей), между которыми установлены связи с помощью ключей.

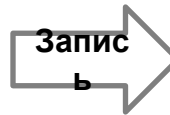
- Edgar Frank Codd.
- Основные концепции модели опубликована в 1970 г.
- Модель основывается на понятии «отношения» (Relation).

**Запись** - это строка таблицы.

**Поле** - это столбец таблицы.

**Имя поля** содержит название столбца вынесенное в заголовок.

**Поле** (имя + тип (свойства: размер, формат и др.))



	Имя поля		

таблица «Предприятие»

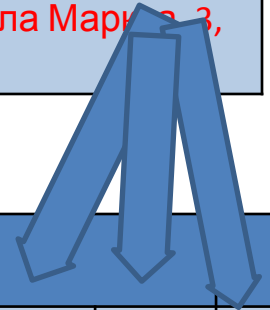
Код	Наименование	Отрасль	Вид деятельности	Дата основания
1	Уфимское моторостроительное производственное объединение	авиадвигателестроение	производство авиационных двигателей	01.01.1925
2	Туймазинский картонно-бумажный комбинат	целлюлозно-бумажная	производство и продажа бумаги	01.01.1962
3	Учалинский горно-обогатительный комбинат	горное дело	добыча и производства	01.01.1961



# Атомарные значения полей

Фрагмент БД ВУЗа, таблица «Студент»

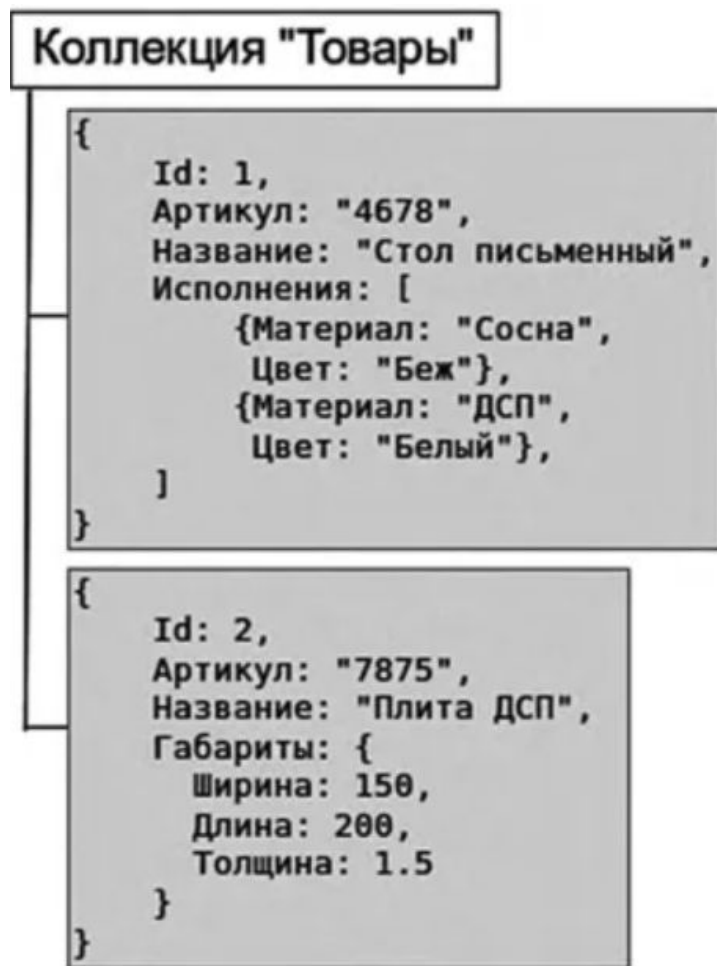
Код	Фамилия	Имя	Отчество	Год рождения	Пол	Индекс	Город	Адрес
1	Иванов	Иван	Иванович	2001	мужской	450075	Уфа	бульвар Славы, 2, кв.12
2	Сидоров	Петр	Петрович	2001	мужской	450064	Уфа	Мира, 7, кв.10
3	Синицын	Инна	Петровна	2002	женский	450008	Уфа	Карла Маркса, 3, кв.4



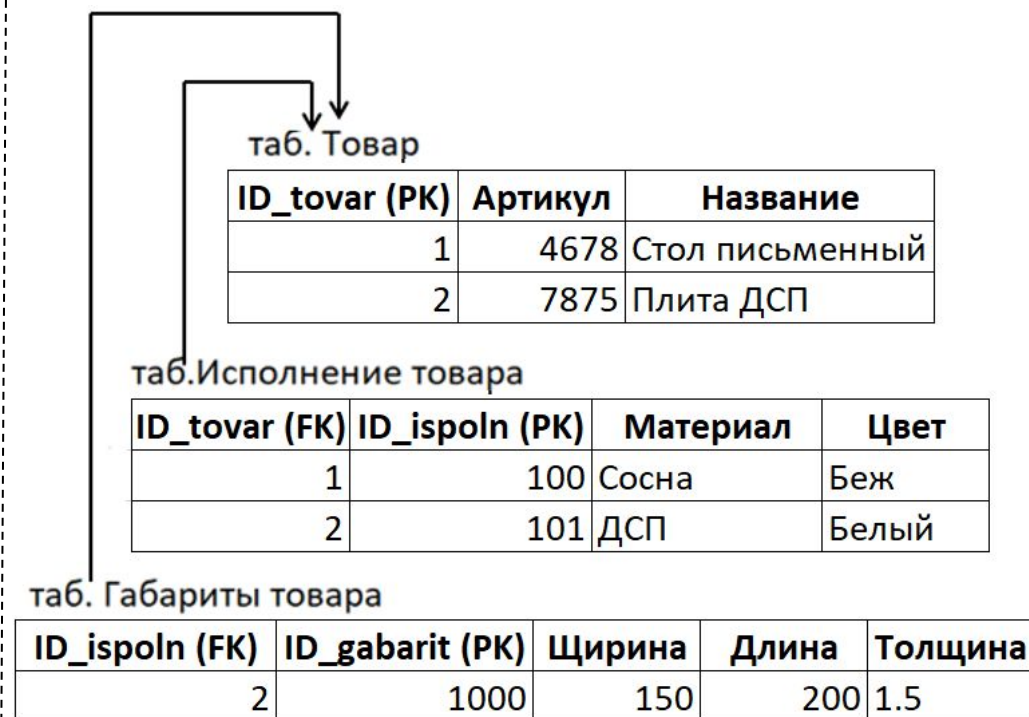
Фрагмент БД Издательства, таблица «Подписчик»

Код	Фамилия	Имя	Отчество	Год рождения	Пол	Индекс	Город	Улица	Дом	Квартира
1	Иванов	Иван	Иванович	2001	мужской	450075	Уфа	бульвар Славы	2	12
2	Сидоров	Петр	Петрович	2001	мужской	450064	Уфа	Мира	7	10
3	Синицына	Инна	Петровна	2002	женский	450008	Уфа	Карла Маркса	3	4

# Документно-ориентированная модель данных



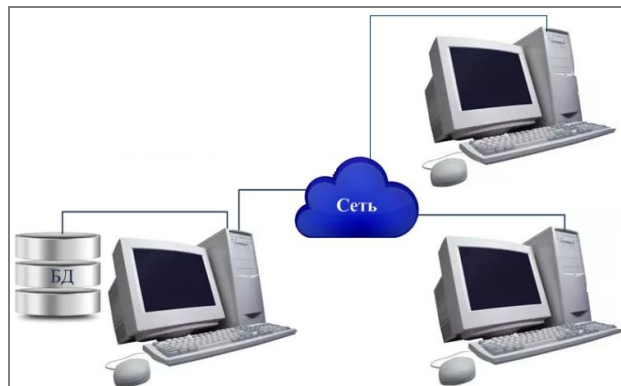
## Реляционная модель данных



## 2) по способу хранения данных

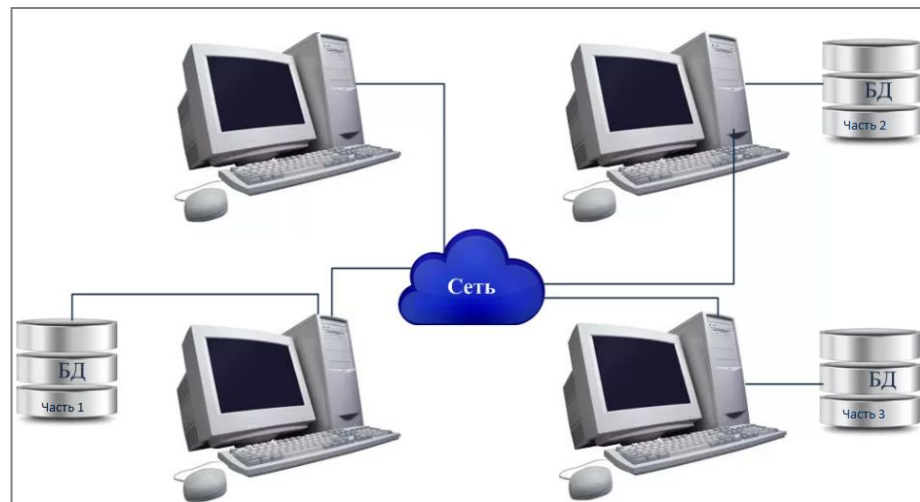
### Базы данных

**Централизованные**  
(БД хранится на одном компьютере)



**Распределенные**

(составные части единой БД хранятся на нескольких компьютерах, объединенных в сеть)





### 3) по способу доступа к БД

## Базы данных

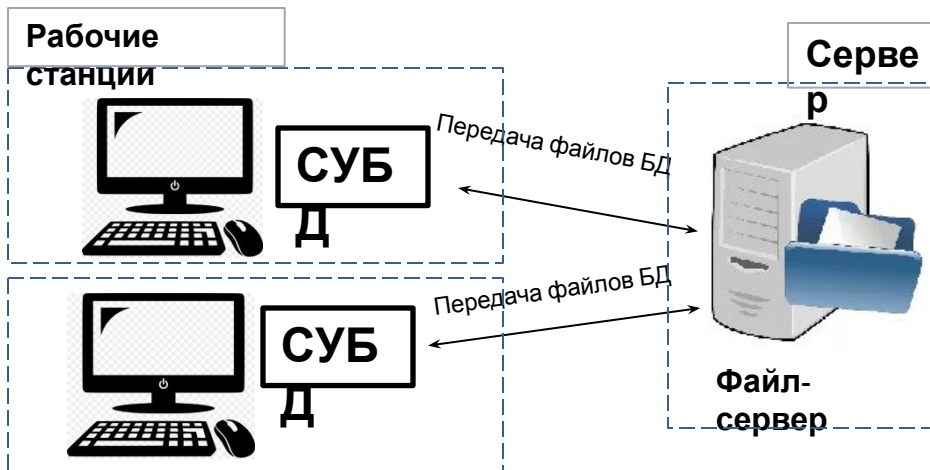
### Локальные

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



### Файл –серверные

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

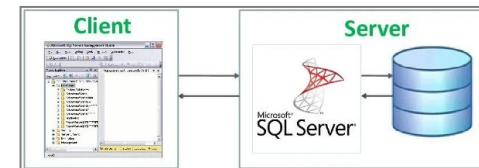
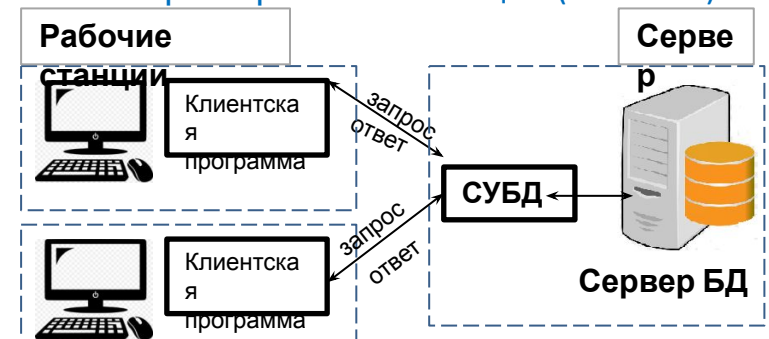


### Удаленные (сетевые)

### Клиент-серверные

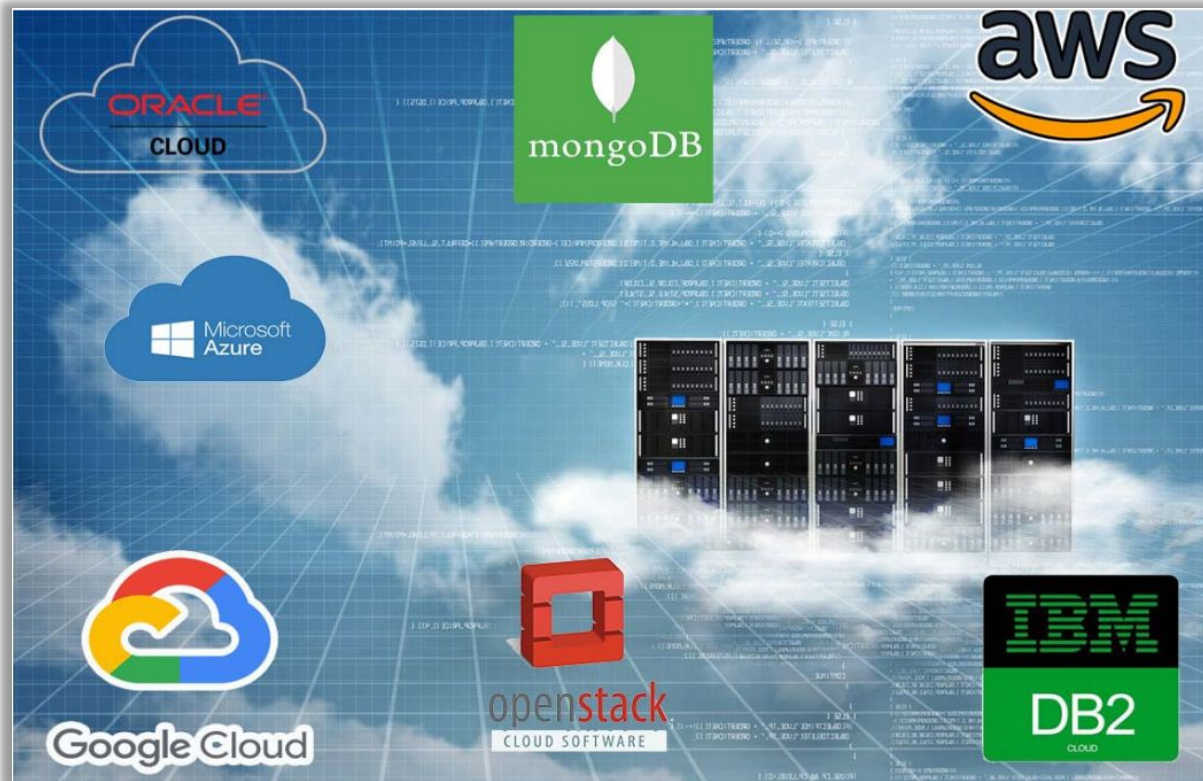
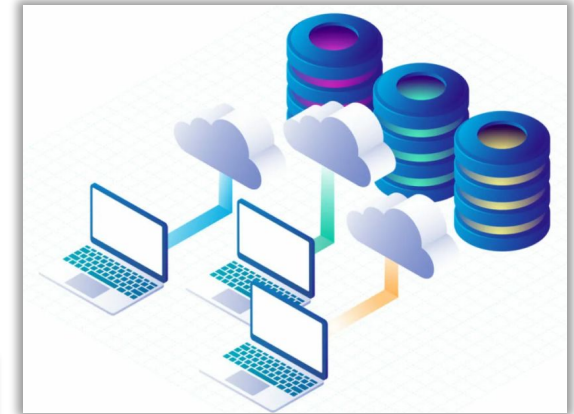
(БД и СУБД находятся на сервере (сервер БД), а клиентские программы на рабочих станциях).

С рабочей станции (клиента) отправляются запросы на сервер (используется специальный язык запросов SQL), полученные результаты выводятся на экране рабочей станции (клиенте)



# Облачные платформы

Облачные платформы предоставляют возможность разработки, выполнения приложений и хранения данных на серверах, расположенных в распределенных дата-центрах.



# Главные законы об информации и информационной безопасности

149-ФЗ Об информационной безопасности — устанавливает основные права и обязанности, касающиеся информации и информационной безопасности.

152-ФЗ — описывает правила работы с персональными данными.

98-ФЗ — определяет, что относится к коммерческой тайне компаний.

68-ФЗ — дает определение электронной подписи и описывает, как и когда ее можно применять, какой юридической силой она обладает.

187-ФЗ — описывает правила защиты IT-инфраструктуры на предприятиях, работающих в сферах, критически важных для государства. К таким сферам относится здравоохранение, наука, оборона, связь, транспорт, энергетика, банки и некоторая промышленность.

**Федеральная служба по  
техническому и  
экспортному контролю**

Государственный орган



составляет

Государственный реестр  
сертифицированных средств  
защиты информации

# Программа курса «Базы данных»

освоите язык запросов доступа к базам данных;

сможете обеспечить целостность данных в базах данных;

научитесь оптимизировать запросы;

научитесь проектировать базы данных;

научитесь создавать приложение для работы с базами данных.



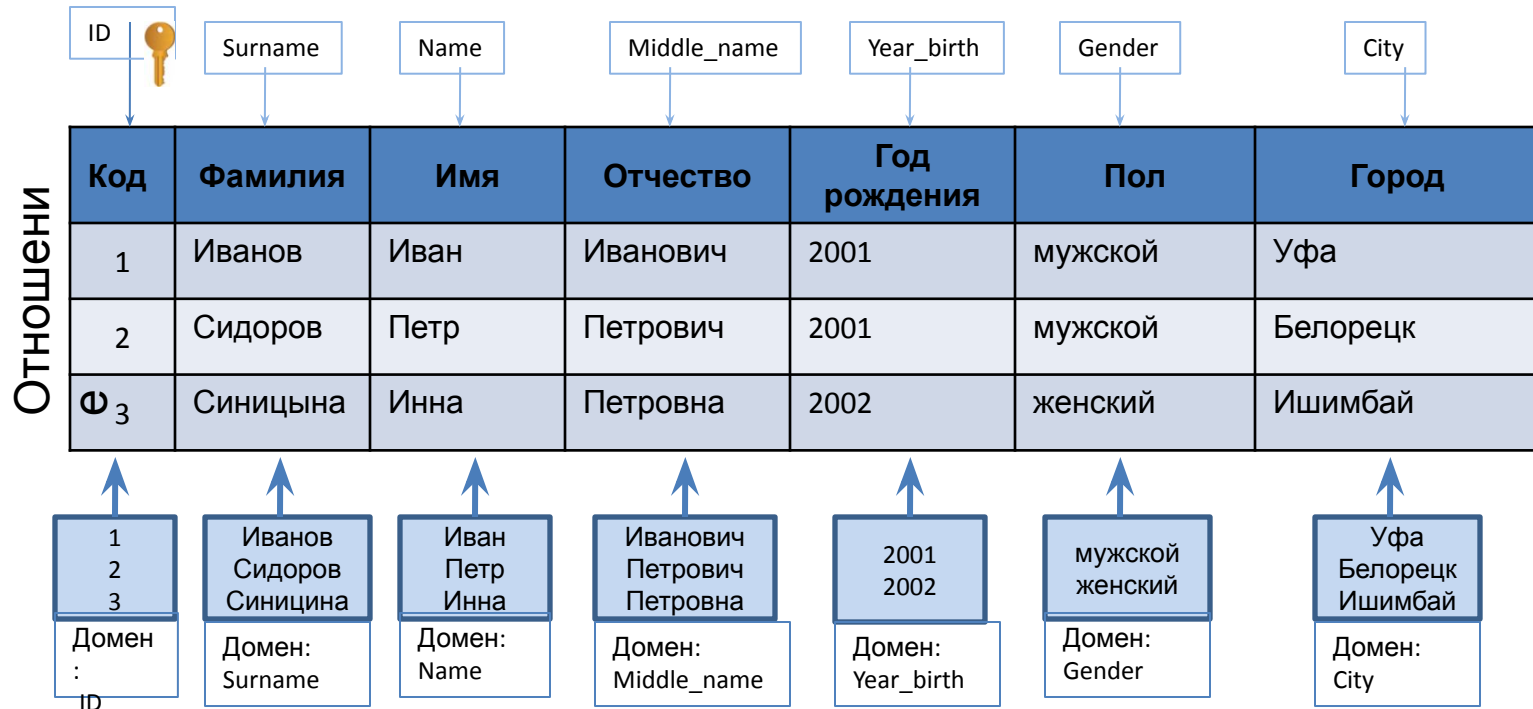
# Примеры СУБД

1. MS Access, MS SQL Server от компании Microsoft Corporation.
2. Oracle, MySQL от компании Oracle Corporation.
3. PostgreSQL от компании PostgreSQL Global Development Group
  - а) Postgres Pro от российской компании Postgres Professional.
4. и др.



# Основные элементы реляционной БД

Термины реляционной модели	Термины «табличные» и языка SQL	Термины обработки данных
Отношение	Таблица	Файл
Кортеж	Строка	Запись
Атрибут	Столбец	Поле
Домен	Множество допустимых значений	Базовый или пользовательский тип данных (с условиями)
Мощность (кардинальность) отношения	Количество строк	Количество записей
Степень отношения	Количество столбцов	Количество полей



# Ключи

**Первичный ключ (сокращенно РК - *Primary Key*)** – это поле (или совокупность полей), значения которого не могут повторяться.



таблица «Студент»						
Фамилия	Имя	Отчество	Дата рождения	Пол	Номер зачетной книжки	Адрес
Иванов	Иван	Иванович	01.01.2001	мужской	12345678	450075, Уфа, б-р Славы, 2
Сидоров	Петр	Петрович	02.02.2001	мужской	89123456	450900 д. Жилино, Семейная, 7
Синицына	Инна	Петровна	03.03.2002	женский	56789012	450008 Уфа, пл. Карла Маркса, 3



**Первичный  
ключ**

# Простой, составной ключ

**Простой** первичный ключ состоит из одного поля.

**Составной** первичный ключ состоит из более чем одного поля.

таблица «Студент»						
Фамилия	Имя	Отчество	Дата рождения	Пол	Номер зачетной книжки	Адрес
Иванов	Иван	Иванович	01.01.2001	мужской	12345678	450075, Уфа, б-р Славы, 2
Сидоров	Петр	Петрович	02.02.2001	мужской	89123456	450900 д. Жилино, Семейная, 7
Синицын а	Инна	Петровна	03.03.2002	женский	56789012	450008 Уфа, пл. Карла Маркса, 3

Составной первичный  
ключ

Простой первичный  
ключ

таблица «Студент»					
Фамилия	Имя	Отчество	Дата рождения	Пол	Адрес
Иванов	Иван	Иванович	01.01.2001	мужской	450075, Уфа, б-р Славы, 2
Сидоров	Петр	Петрович	02.02.2001	мужской	450900 д. Жилино, Семейная, 7
Синицын а	Инна	Петровна	03.03.2002	женский	450008 Уфа, пл. Карла Маркса, 3

# Ключи по способу задания

**Логический (естественный) первичный ключ** – поле, данные в котором логически связаны с информационным содержанием записи и уникально ее идентифицируют. **Логический первичный**

**ключ**




таблица «Пользователь»		
Имя	E-mail	Пароль
Иван	ivan@mail.ru	A@s123
Семен	semen@bk.ru	@A345s
Инна	Inna@yandex.ru	f5@As

**Суррогатный (искусственный) первичный ключ** - поле добавленное искусственным путем для однозначной идентификации записей.

**Суррогатный первичный**

**ключ**




таблица «Пользователь»			
Код пользователя	Имя	E-mail	Пароль
5	Иван	ivan@mail.ru	A@s123
6	Семен	semen@bk.ru	@A345s
7	Инна	Inna@yandex.ru	f5@As



# Ключи

**Внешний ключ (сокращенно FK - Foreign Key)** – не ключевое поле (совокупность полей) таблицы, связанное с первичным ключом другой таблицы.

Первичный ключ (РК) →

Главная таблица ↓

Код города	Название	Год основания	Площадь, км <sup>2</sup>
1	Уфа	1574	708
2	Москва	1147	2 561
3	Калининград	1255	224
NULL	Самара	1586	541

Ошибка!  
Значение первичного ключа не может быть NULL

Первичный ключ (РК) ↓

Внешний ключ (FK) ↓

Подчиненная таблица ↓

Код улицы	Код города	Название
100	1	бульвар Славы
101	1	Карла Маркса
102	3	Карла Маркса
103	2	Арбат
104	2	Тверская
105	3	Янтарная
106	4	Свердловская

Ошибка! Нет значения РК в главной таблице

# Ключи

**Ключ** или **потенциальный ключ (Candidate key)** - простой или составной ключ, который уникально идентифицирует каждую запись таблицы.

Если отношение имеет более одного потенциального ключа, то один из них рассматривается как первичный, остальные являются **альтернативными (Alternative key)**.

таблица «Пользователь»				
Код пользовател я	Имя	Логин	E-mail	Пароль
5	Иван	Ivan	ivan@mail.ru	A@s123
6	Семен	Semen	semen@bk.ru	@A345s
7	Инна	Inna	Inna@yandex.ru	f5@As

Потенциальные ключи:

- 1) Код пользователя (например, первичный ключ).
- 2) Логин.
- 3) E-mail.
- 4) Имя + Пароль (если не задано условие на уникальность пароля).

Не потенциальные ключи (т.к. не отвечают требованию минимальности):

- 5) Имя + Логин.
- 6) Имя + E-mail.

# Ограничения целостности данных

Целостностью данных можно назвать механизм поддержания соответствия базы данных предметной области.

Ограничения целостности – это требования предназначенные для предупреждения добавления в базу невозможных, невероятных данных, т.е. обеспечивают защиту от возможных ошибок пользователя.

Базовые требования обеспечения целостности:

1) **Ссылочную целостность (или целостность ссылок)** обеспечивается системой первичных и внешних ключей. Правило: значение внешнего ключа подчиненной таблицы должно соответствовать одному из значений первичного ключа главной таблицы или иметь значение *NULL*.

2) **Целостность сущностей**. Правило: любая таблица (отношение) должна иметь первичный ключ. Поле первичного ключа не должно содержать пустые значения (*NULL*).

3) **Целостность домена<sup>1</sup>**. Правило: все значения некоторого поля должны принадлежат множеству допустимых значений.

Целостность домена обеспечивается заданием условий на значения (CHECK), запретом пустых значений (NOT NULL), заданием значения по умолчанию (DEFAULT), хранимыми процедурами, триггерами.

Например, при создании структуры таблицы можно сразу указать возможное количество цифр в поле Номер телефона.

<sup>1</sup>Домен (в реляционной модели данных) – множество допустимых значений поля.

# Задачи

1) Какие поля в таб.1 и таб.2 могут быть первичными ключами?

Определите названия ключей по типу и по способу задания.

Дата	Время	Температура воды	Температура воздуха	Скорость ветра	Волны
01.06.2019	9:00	19	29	8	2
26.07.2019	9:00	22	30	7	2
23.08.2019	10:00	24	36	9	3
23.08.2019	14:00	26	38	2	1

Фамилия	Имя	Адрес	Телефон
Иванов	Иван	г.Уфа, Айская,11	9271112233
Евдокимов	Петр	г.Сибай, Ленина,2	9167779900
Петров	Иван	г.Уфа, Мира,10	9064455612

# Задачи

2) Какие из приведенных ниже полей могут стать простым естественным первичным ключом?

- фамилия;
- имя;
- номер и серия паспорта;
- номер дома;
- город проживания;
- адрес электронной почты;
- дата выполнения работы;
- марка стиральной машины.

3) Сколько строительных компаний в городе Москва?

4) Сколько строительных компаний в городе Уфа?

таблица

«Город»

Код города	Название
1	Пермь
2	Москва
3	Челябинск
4	Уфа

таблица «Строительная

Код	Название	Код города
100	ЖилСтрой	3
101	Новострой	2
102	Трест №3	4
103	ИнвестСтрой	1
104	Кооператив №1	2
105	Трест№1	4
106	СтройКапитал	4



# Задачи

- 5) Определите какие материалы отправлены в каждый из городов?
- 6) Сколько единиц огнеупорных кирпичей отправлено в каждый город?
- 7) Посчитайте общую стоимость материалов, отправленных в каждый город.

таблица

«Город» Код	Название
1	Пермь
2	Москва
3	Челябинск
4	Уфа

таблица «Строительная

Код	Название	Код города
100	ЖилСтрой	3
101	Новострой	2
102	Трест №3	4
103	ИнвестСтрой	1
104	Кооператив №1	2
105	Трест№1	4
106	СтройКапитал	4

таблица

«Материал» Артикул	Название	Цена за единицу, руб
1234	Кирпич огнеупорный	10
4352	Кирпич облицовочный	8
1265	Опора бруса	100
763	Панель	1200
8412	Плитка настенная	50

таблица

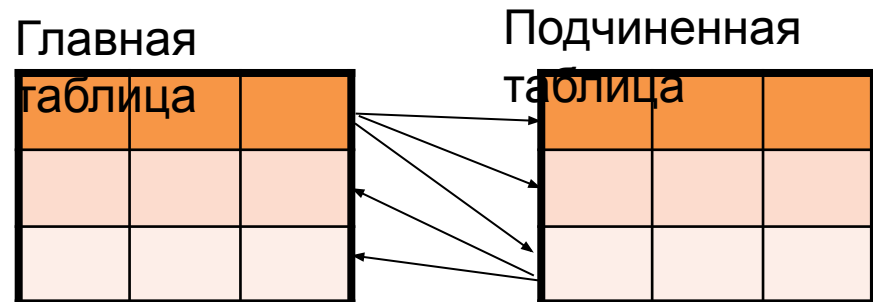
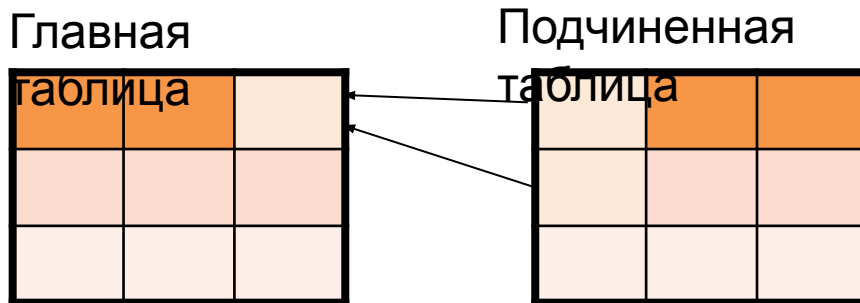
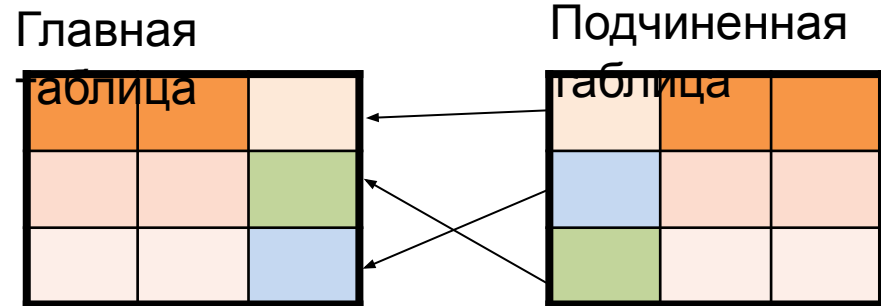
«Накладная» Накладная	Код компании	Артикул	Количество, ед.
912	101	1234	1000
751	100	1234	2000
784	102	8412	1000
132	100	763	100
542	104	1234	500
321	101	1265	100

# Виды связей между реляционными таблицами

**Реляционная база данных** — это совокупность двумерных взаимосвязанных таблиц.

Виды связей между таблицами:

- 1) Один к одному (1:1, 1–1).
- 2) Один ко многим (1:M, 1 –  $\infty$  ).
- 3) Многие ко многим (M:M,  $\infty$  –  $\infty$ ).



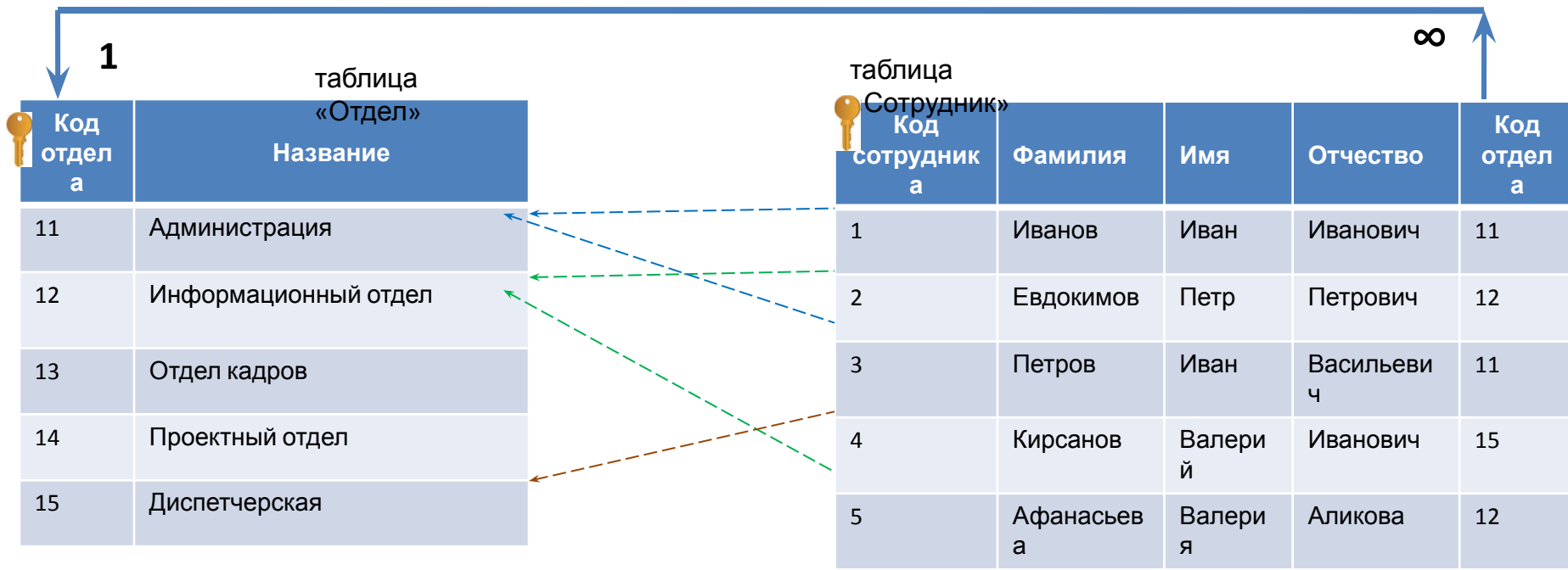
# Вид связи один к одному

Связь **один к одному** означает, что одной записи в главной таблице соответствует только одна запись в подчиненной.



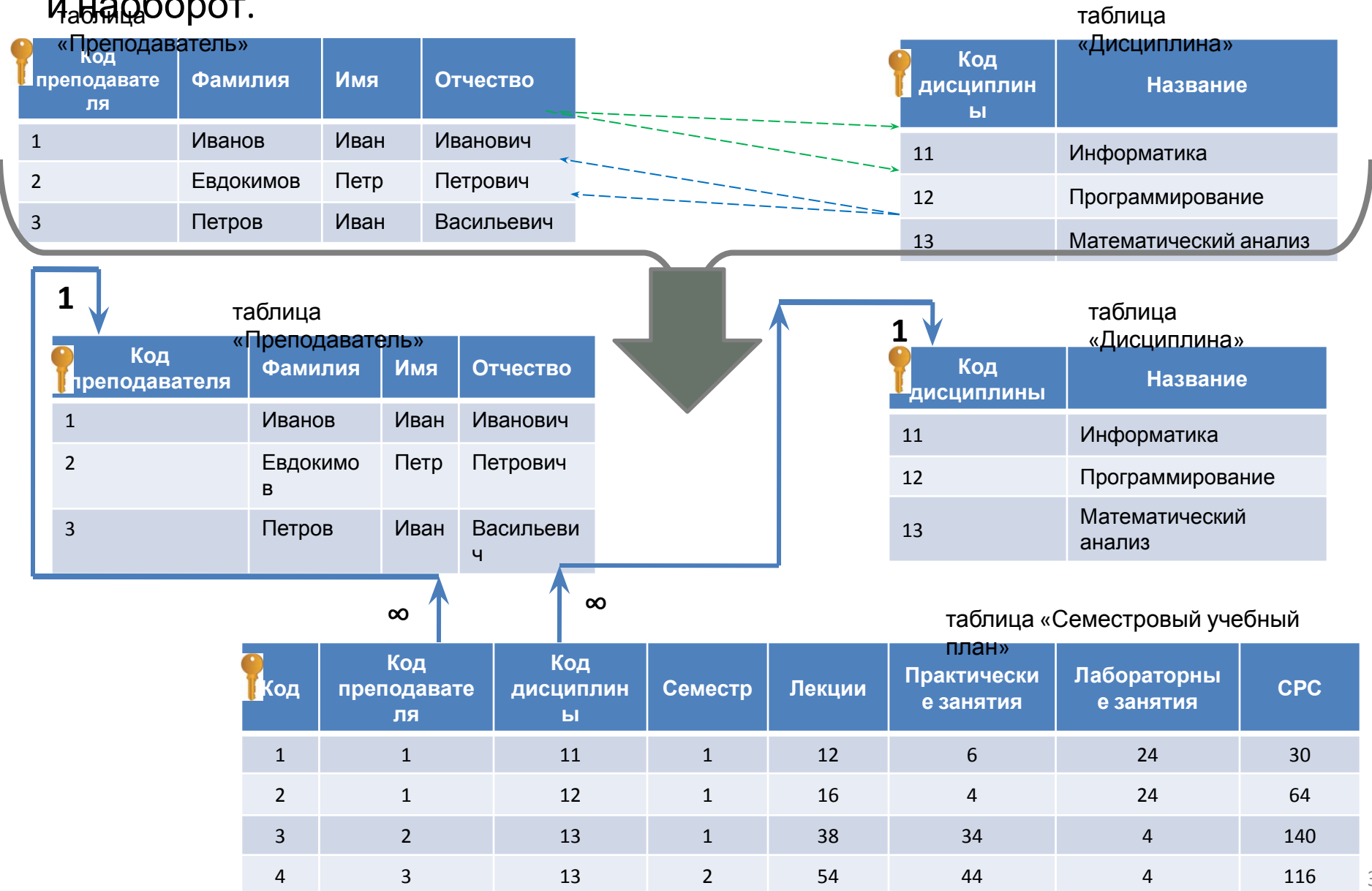
# Вид связи один ко многим

Связь **один ко многим** означает, что одной записи в главной таблице может соответствовать множество записей в подчиненной таблице.



# Вид связи многие ко многим

Связь **многие ко многим** означает, что одной записи в главной таблице может соответствовать множество записей в подчиненной, и наоборот.

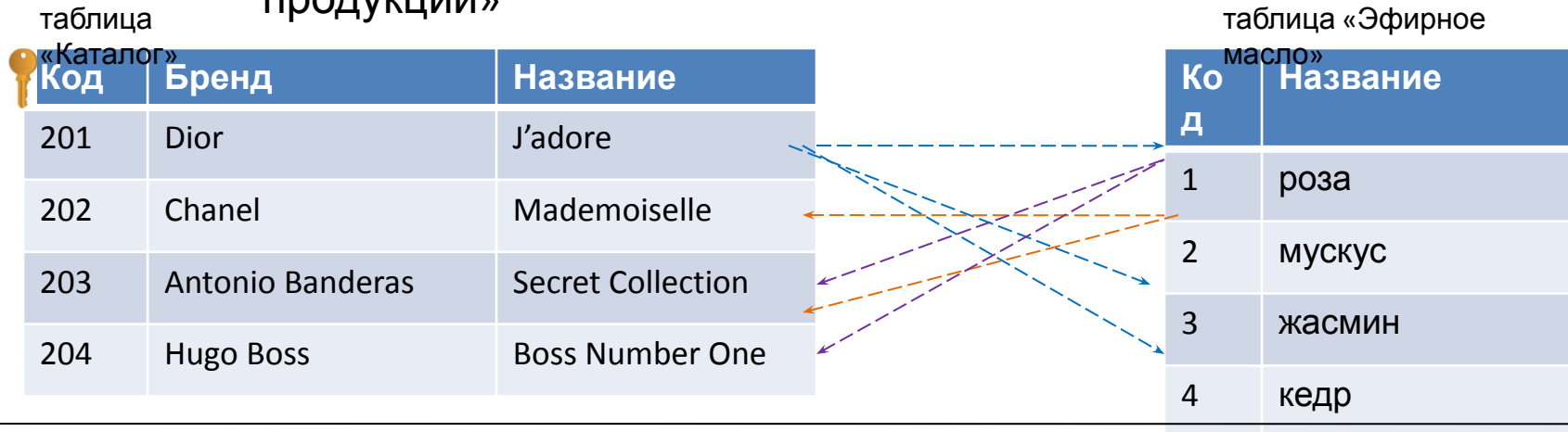




# Задачи

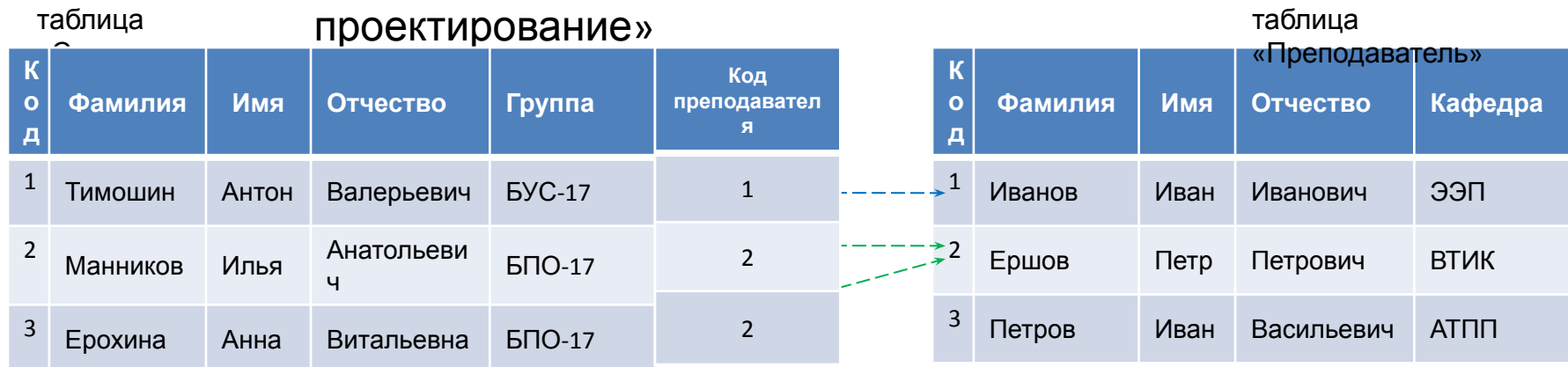
1) Какой вид связи между таблицей 1 и таблицей 2?

## Фрагмент БД «Информация по парфюмерной продукции»



2) Какой вид связи между таблицей 1 и таблицей 2? Какая из двух таблиц главная, какая подчинённая?

## Фрагмент БД «Дипломное проектирование»



# Задачи

3) Какой вид связи между таблицей 1 и таблицей 2? Какая из двух таблиц главная, какая подчинённая?

## Фрагмент БД «Административные единицы»

таблица

Код	Название	Площадь
1	Уфа	707
2	Ишимбай	108,52
3	Сибай	157

таблица «Территориальное деление»

Код города	Код	Название
1	101	Демский
1	102	Калининский
2	103	Шахтау
2	104	Уральский

4) Какой вид связи между таблицей 1 и таблицей 2? Какая из двух таблиц главная, какая подчинённая?

## Фрагмент БД «География»

таблица

Код	Название	Площадь
1	Россия	146 748 590
2	Италия	301 340
3	Испания	505 990

таблица «Столица»

Код страны	Код	Название	Площадь
1	101	Москва	2 561
2	102	Рим	1 287
3	103	Мадрид	607

# Поддержка целостности сущностей и целостности ссылок.

Синтаксис :

1) < поле и тип данных> **PRIMARY KEY**

2) **FOREIGN KEY** < поле> **REFERENCES** < главная таблица> [< первичный ключ>]

Пример, фрагмент БД «Дипломное проектирование»

Diploma_supervisor_id	Surname	Name	Middle_name	Department
1	Иванов	Иван	Иванович	ЭЭП
2	Ершов	Петр	Петрович	ВТИК
3	Петров	Иван	Васильевич	АТПП

```
CREATE TABLE Diploma_supervisor
(Diploma_supervisor_Id INTEGER PRIMARY KEY IDENTITY(1,1),
Surname VARCHAR(20) NOT NULL,
Name VARCHAR(20) NOT NULL,
Middle_name VARCHAR(20),
Department VARCHAR(10))
```

table. Student

Student_id	Surname	Name	Middle_name	Diploma_supervisor_id
1	Тимошин	Антон	Валерьевич	1
2	Манников	Илья	Анатольевич	2
3	Ерохина	Анна	Витальевна	2

**CREATE TABLE** Student

```
(Student_id INTEGER PRIMARY KEY ,
Surname VARCHAR(20) NOT NULL,
Name VARCHAR(20) NOT NULL,
Middle_name VARCHAR(20),
Diploma_supervisor_id INTEGER,
CONSTRAINT Key_ foreign
```

```
FOREIGN KEY (Diploma_supervisor_id) REFERENCES Diploma_supervisor (Diploma_supervisor_id))
```

**ALTER TABLE** Student

```
ADD CONSTRAINT Key_ foreign FOREIGN KEY(Diploma_supervisor_id) REFERENCES Diploma_supervisor (Diploma_supervisor_id)
```

# Поддержка целостности данных при использовании команд UPDATE и DELETE

FOREIGN KEY ( ) REFERENCES [ [ ]

[ **ON DELETE** { **NO ACTION** | SET NULL | CASCADE | SET DEFAULT }

[ ON UPDATE { NO ACTION | SET NULL | CASCADE | SET DEFAULT } ]

table. Diploma\_supervisor

Diploma_supervisor_id	Surname	Name	Middle_name	Department
1	Иванов	Иван	Иванович	ЭЭП
2	Ершов	Петр	Петрович	ВТИК
3	Петров	Иван	Васильевич	АТПП



Ошибка

table. Student

Student_id	Surname	Name	Middle_name	Diploma_supervisor_id
1	Тимошин	Антон	Валерьевич	1
2	Манников	Илья	Анатольевич	2
3	Ерохина	Анна	Витальевна	2

CREATE TABLE **Student**

(Student\_id INTEGER **PRIMARY KEY**,

Surname VARCHAR(20) **NOT NULL**,

Name VARCHAR(20) **NOT NULL**,

Middle\_name VARCHAR(20),

Diploma\_supervisor\_id INTEGER,

**CONSTRAINT** Key\_foreign

**FOREIGN KEY**(Diploma\_supervisor\_id) **REFERENCES** Diploma\_supervisor(Diploma\_supervisor\_id) **ON DELETE NO ACTION**)

# Поддержка целостности данных при использовании команд UPDATE и DELETE

FOREIGN KEY ( ) REFERENCES [ [ ]

[ **ON DELETE** {NO ACTION | **SET NULL** | CASCADE | SET DEFAULT}]

[ON UPDATE {NO ACTION | SET NULL | CASCADE | SET DEFAULT}]

table. Diploma\_supervisor

Diploma_supervisor_id	Surname	Name	Middle_name	Department
1	Иванов	Иван	Иванович	ЭЭП
2	Ершов	Петр	Петрович	ВТИК
3	Петров	Иван	Васильевич	АТПП

table. Student

Student_id	Surname	Name	Middle_name	Diploma_supervisor_id
1	Тимошин	Антон	Валерьевич	1
2	Манников	Илья	Анатолеви ч	<i>NULL</i>
3	Ерохина	Анна	Витальевна	<i>NULL</i>

CREATE TABLE **Student**

(Student\_id            INTEGER **PRIMARY KEY**,  
Surname                VARCHAR(20) **NOT NULL**,  
Name                    VARCHAR(20) **NOT NULL**,  
Middle\_name            VARCHAR(20),  
Diploma\_supervisor\_id        INTEGER,

**CONSTRAINT** Key\_foreign

**FOREIGN KEY**(Diploma\_supervisor\_id) **REFERENCES** Diploma\_supervisor(Diploma\_supervisor\_id) **ON DELETE SET NULL**)

# Поддержка целостности данных при использовании команд UPDATE и DELETE

FOREIGN KEY ( ) REFERENCES [ [ ]

[ **ON DELETE** {NO ACTION | SET NULL| **CASCADE** | SET DEFAULT}]

[ON UPDATE {NO ACTION | SET NULL| CASCADE | SET DEFAULT}]

table. Department

Dept_id	Department	Telephone
100	Администрация	12345
101	Информационный отдел	54321
102	Отдел кадров	23431
<del>103</del>	<del>Проектный отдел</del>	<del>45673</del>

table. Cooperator

Cooperator_id	Surname	Name	Middle_name	Dept_id
1	Иванов	Иван	Иванович	100
2	Сидоров	Петр	Петрович	101
3	Синицына	Инна	Петровна	102
<del>4</del>	<del>Егоров</del>	<del>Валерий</del>	<del>Игнатьевич</del>	<del>103</del>
<del>5</del>	<del>Воронина</del>	<del>Наталья</del>	<del>Игоревна</del>	<del>103</del>

CREATE TABLE **Cooperator**

(Cooperator\_id            INTEGER **PRIMARY KEY**,

Surname                 VARCHAR(20) **NOT NULL**,

Name                     VARCHAR(20) **NOT NULL**,

Middle\_name            VARCHAR(20),

Dept\_id                  INTEGER,

**CONSTRAINT** Key\_ foreign

**FOREIGN KEY**(Dept\_id ) **REFERENCES** Department(Dept\_id ) **ON DELETE CASCADE**)



# Поддержка целостности данных при использовании команд UPDATE и DELETE

FOREIGN KEY ( ) REFERENCES [ [ ]

[ **ON DELETE** {NO ACTION | SET NULL| CASCADE | **SET DEFAULT**}]

[ON UPDATE {NO ACTION | SET NULL| CASCADE | SET DEFAULT}]

table. Department

Dept_id	Department	Telephone
100	Администрация	12345
101	Информационный отдел	54321
102	Отдел кадров	23431
103	Проектный отдел	45673
1000	Резерв	11108

table. Cooperator

Cooperator_id	Surname	Name	Middle_name	Dept_id
1	Иванов	Иван	Иванович	100
2	Сидоров	Петр	Петрович	101
3	Синицына	Инна	Петровна	102
4	Егоров	Валерий	Игнатьевич	1000
5	Воронина	Наталья	Игоревна	1080

CREATE TABLE **Cooperator**

(Cooperator\_id            INTEGER **PRIMARY KEY**,

Surname                VARCHAR(20) **NOT NULL**,

Name                    VARCHAR(20) **NOT NULL**,

Middle\_name            VARCHAR(20),

Dept\_id                 INTEGER **DEFAULT 1000**,

**CONSTRAINT** Key\_ foreign

**FOREIGN KEY**(Dept\_id ) **REFERENCES** Department(Dept\_id ) **ON DELETE SET DEFAULT**)

# Язык SQL



SQL  
Stuctured Query Language

Язык SQL, предназначенный для взаимодействия с данными в БД.

Появился в середине 70-х (первые публикации 1974 г.).

Первый принятый стандарт SQL/86 разработан Американским национальным институтом стандартов (ANSI) и одобрен Международной организацией по стандартизации (ISO) в 1986г.

Последняя редакция стандарта: SQL:2016

В каждой СУБД применяется свой диалект языка.

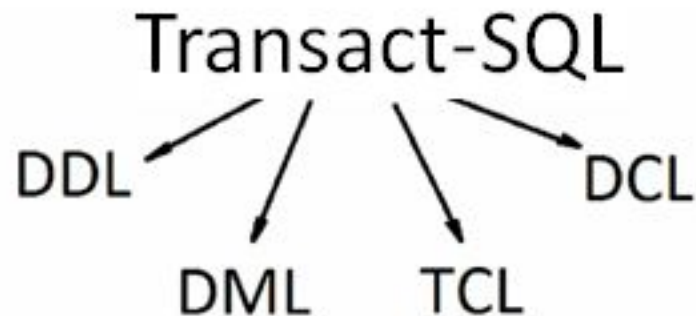


# Диалекты языка SQL

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



# Команды



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

# Команды языка определения данных (DDL - Data Definition Language)

Команда	Смысл	Действие
CREATE DATABASE	Создать БД	Создает новую базу данных, определив основные параметр для нее
ALTER DATEBASE	Изменить БД	Изменяет набор основных объектов в базе данных, ограничений, касающихся всей базу данных
DROP DATEBASEE	Удалить БД	Удаляет существующую БД, только в том случае, если есть права на выполнение данного действия
CREATE TABLE	Создать таблицу	Создает новую таблицу
ALTER TABLE	Изменить таблицу	Изменяет структуру существующей таблицы или ограничения целостности
DROP TABLE	Удалить таблицу	Удаляет таблицу из БД
TRUNCATE TABLE	Удалить данные из таблицы	Удаляет данные в таблице, но сохраняет ее структуру и индексы
CREATE VIEW	Создать представление	Создает виртуальную таблицу, соответствующую некоторому SQL-запросу
ALTER VIEW	Изменить представление	Изменяет ранее созданное представление
CREATE INDEX	Создать индекс	Создает индекс для некоторой таблицы с целью обеспечения быстрого доступа по атрибутам, входящим в индекс
ALTER INDEX	Изменить индекс	Изменяет существующий индекс таблицы или представления
DROP INDEX	Удалить индекс	Удаляет ранее созданный индекс
CREATE PROCEDURE	Создать процедуру	Создает хранимую процедуру и сохраняет как объект базы данных
ALTER PROCEDURE	Изменить процедуру	Изменяет ранее созданную процедуру
DROP PROCEDURE	Удалить процедуру	Удаляет процедуру из БД
CREATE FUNCTION	Создать функцию	Создает определенную пользователем функцию и сохраняет как объект базы данных
ALTER FUNCTION	Изменить функцию	Изменяет существующую функцию
DROP FUNCTION	Удалить функцию	Удаляет функцию из БД
CREATE TRIGGER	Создать триггер	Создает триггер, как предварительно определенное действие или последовательность действий, автоматически осуществляемых при выполнении операций обновления, добавления или удаления данных.
ALTER TRIGGER	Изменить триггер	Изменяет определение триггера
DROP TRIGGER	Удалить триггер	Удаляет ранее созданный триггер

# Примеры применения команд DDL

Создание таблицы «Плата за электроэнергию»

```
CREATE TABLE Rent_for_light  
(Id INT PRIMARY KEY,  
Region VARCHAR(20) ,  
N_Month TINYINT CHECK (N_Month>0 AND N_Month<13),  
Unit_cost REAL /* плата за единицу электрической энергии, кВт*/  
)
```

Table. Rent\_for\_light

Id	Region	N_Month	Unit_cost
NULL	NULL	NULL	NULL

```
ALTER TABLE Rent_for_light  
ADD Regional_coefficient REAL
```

Table. Rent\_for\_light

Id	Region	N_Month	Unit_cost	Regional_coefficient
NULL	NULL	NULL	NULL	NULL

```
TRUNCATE TABLE Rent_for_light
```

```
DROP TABLE Rent_for_light
```



## Команды языка манипулирования данными (DML - Data Manipulation Language)

Команда	Смысл	Действие
INSERT	Вставить строку	Вставляет одну или несколько строк в базовую таблицу. Допустимы модификации команды, при которых сразу несколько строк могут быть перенесены из одной таблицы или запроса в базовую таблицу
UPDATE	Обновить строку	Обновляет значения одного или нескольких столбцов в одной или нескольких строках, соответствующих условиям фильтрации
DELETE	Удалить строку	Удаляет одну или несколько строк, соответствующих условиям фильтрации, из базовой таблицы. Применение команды согласуется с принципами поддержки целостности, поэтому эта команда не всегда может быть выполнена корректно, даже если синтаксически она записана правильно
SELECT	Выбрать строки	Команда, позволяющая сформировать результирующее отношение, соответствующее запросу

# Примеры применения команд DML

**INSERT INTO** Rent\_for\_light

**VALUES** ('Республика Башкортостан', 1, 20, 0.1), ('Республика Татарстан', 1, 25, 0.1)

Id	Region	N_Month	Unit_cost	Regional_coefficient
1	Республика Башкортостан	1	20	0.1
2	Республика Татарстан	1	25	0.1

**UPDATE** Rent\_for\_light

**SET** Regional\_coefficient = Regional\_coefficient \*2

**WHERE** Region ='Республика Татарстан'

Id	Region	N_Month	Unit_cost	Regional_coefficient
1	Республика Башкортостан	1	20	0.1
2	Республика Татарстан	1	25	0.2

**SELECT** Region, Unit\_cost

**FROM** Rent\_for\_light

Region	Unit_cost
Республика Башкортостан	20
Республика Татарстан	25

**DELETE FROM** Rent\_for\_light

**WHERE** Regional\_coefficient>0.1

Id	Region	N_Month	Unit_cost	Regional_coefficient
1	Республика Башкортостан	1	20	0.1

# Синтаксис SELECT

**SELECT** *column\_name1, column\_name2, ...*

поля для  
вывода

**FROM** *table\_name*

таблица, данные  
которой нужно  
вывести

**WHERE** *condition*

условие поиска, возвращает только  
строки , соответствующие этому  
условию

Id	Region	N_Month	Unit_cost	Regional_coefficient
1	Республика Башкортостан	1	20	0.1
2	Республика Татарстан	1	25	0.2

```
SELECT Region , N_Month, Unit_cost , Regional_coefficient
FROM Rent_for_light
WHERE Regional_coefficient>0.1
```

Region	N_Month	Unit_cost	Regional_coefficient
Республика Татарстан	1	25	0.2

# Транзакция

**Транзакция** – это последовательность операций с данными, выполняющаяся как единое целое.

Транзакции повышают надежность баз данных.



# Пример транзакции

таблица «Счет в

банке»

Код 	Номер счета	Баланс	Код_клиента (FK)
1	101	1000	1
2	102	2000	2
3	109	500	4

Нужно перевести от одного клиента банка (с номером счета 101) другому клиенту банка (с номером счета 109) денежную сумму в размере 500 руб.

1) **UPDATE** Bank\_account  
    **SET** Balance = Balance - 500  
    **WHERE** Number\_account = 101

2) **UPDATE** Bank\_account  
    **SET** Balance = Balance + 500  
    **WHERE** Number\_account = 109

# Транзакции и целостность баз данных

Количество операций, входящих в транзакцию, может быть от одной и более. Разработчик решает, какие команды должны выполняться как одна транзакция, а какие могут быть разбиты на несколько последовательно выполняемых транзакций.

Транзакция обладает следующими важными свойствами (**ACID**), которые гарантируют правильность и надежность работы системы:

1) **A**tomicity (атомарность).

Каждая транзакция в БД должна быть выполнена полностью либо не выполнена совсем. Не допускается частичное выполнение.



2) **C**onsistency (согласованность).

Должно быть согласованное состояние БД до и после выполнения транзакции.

3) **I**solation (изолированность).

Результаты транзакции не должны быть видны другим транзакциям, пока она не завершится.



4) **D**urability (устойчивость, долговечность).

Изменения, внесенные в БД в результате выполнения транзакции должны



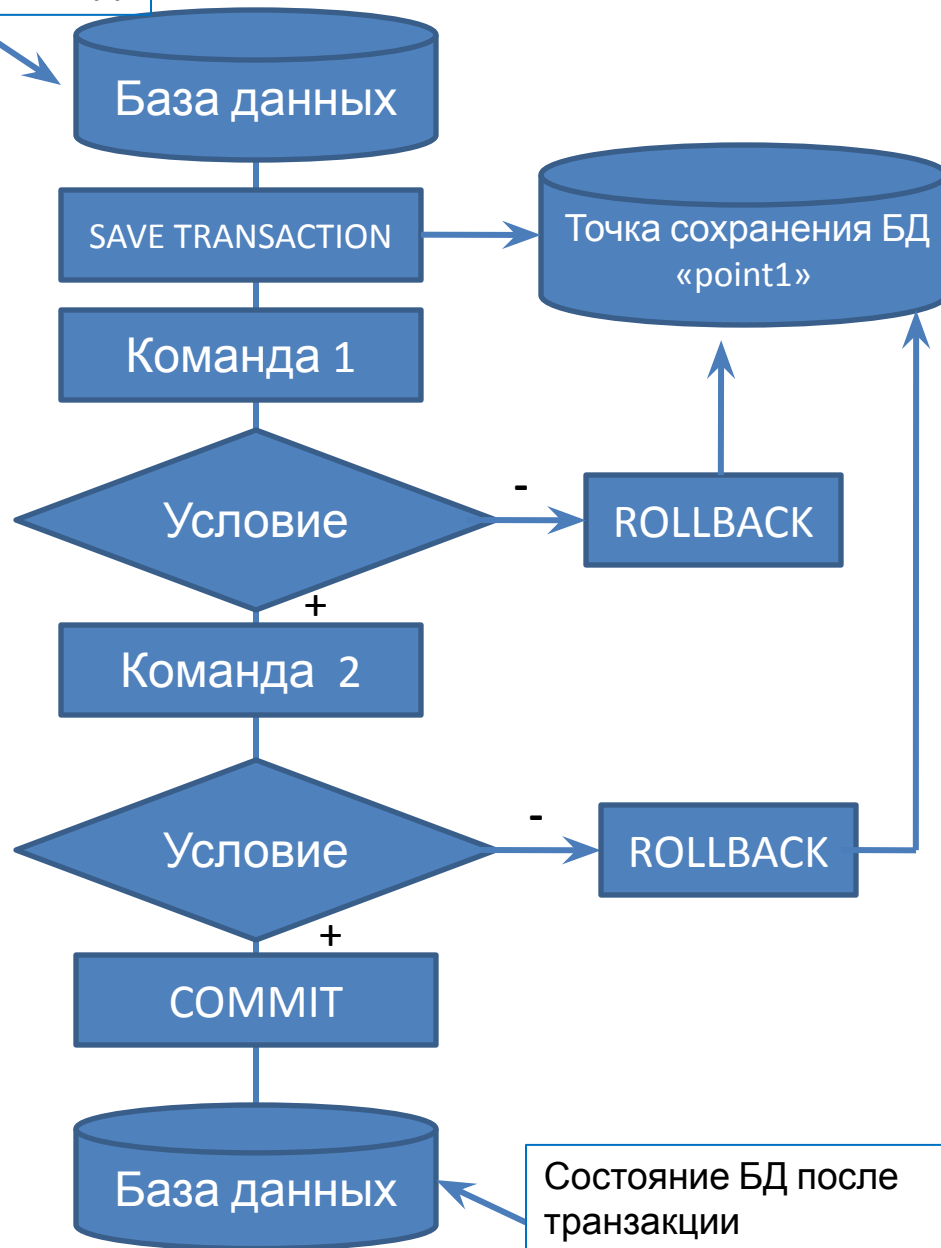


## Команды языка управления транзакциями (TCL - Transaction Control Language)

Команда	Смысл	Действие
SAVE TRANSACTION	Сохранить промежуточную точку выполнения транзакции	Сохранить промежуточное состояние БД, пометить его для того, чтобы можно было в дальнейшем к нему вернуться
COMMIT	Завершить транзакцию	Завершить комплексную взаимосвязанную обработку информации, объединенную в транзакцию
ROLLBACK	Прерывание транзакции	Отменить изменения, проведенные в ходе выполнения транзакции

# Примеры применения команд TCL

Начальное  
состояние БД



```
BEGIN TRANSACTION  
SAVE TRANSACTION point1  
  UPDATE Bank_account  
    SET Balance= Balance - 500  
    WHERE Number_account = 101  
  IF (@@error != 0)  
    ROLLBACK TRANSACTION point1  
  UPDATE Bank_account  
    SET Balance= Balance + 500  
    WHERE Number_account = 109  
  IF (@@error != 0)  
    ROLLBACK TRANSACTION point1  
COMMIT
```

Состояние БД после  
транзакции  
(конечное состояние)

# Команды языка управления данными (DCL - Data Control Language)

Команда	Смысл	Действие
GRANT	Предоставить права	Предоставить права доступа на ряд действий над некоторым объектом БД
DENY	Запретить доступ	Запретить доступ к объектам базы данных
REVOKE	Лишить права	Лишить прав доступа к некоторому объекту или некоторым действиям над объектом

# Примеры применения команд DCL

**GRANT SELECT ON Student TO User2;**

**DENY CREATE DATABASE, CREATE TABLE TO User2;**

**REVOKE SELECT ON Student TO User2.**

# Значение NULL

Значение NULL - универсальное значение, не зависящее от типа данных поля. Данное значение

свидетельствует об отсутствии значения у поля, это не то же самое, что число «0».

Поле с значением NULL - это поле, которое было оставлено пустым во время создания записи!

таблица «Информация об

Код абитуриента	Фамилия	Имя	Отчество	Дата рождения	Номер телефона	...
101	Синицына	Инна	Петровна	01.01.2000	12345678910	
102	Егоров	Валерий	Игнатьевич	01.02.2001	NULL	
103	Воронина	Наталья	Игоревна	11.02.2001	34567890123	

таблица «Результат экзамена»

Код	Код абитуриента	Наименование экзамена	Результат	Номер свидетельства
1	101	ЕГЭ Математика	89	1234567
2	102	ЕГЭ Математика	90	2334556
3	102	ЕГЭ Физика	NULL	NULL
4	101	ЕГЭ Физика	NULL	NULL
5	101	ЕГЭ Русский язык	NULL	NULL

# Использование значения NULL в условиях поиска

**IS NULL** – предикат, применяется для выявления равенства значения некоторого поля неопределенному значению (NULL).

**IS NOT NULL**– предикат, применяется для выявления неравенства значения некоторого поля неопределенному значению (NULL).

- 1) **SELECT** *column\_names*  
    **FROM** *table\_name*  
    **WHERE** *column\_name* **IS NULL**;
- 2) **SELECT** *column\_names*  
    **FROM** *table\_name*  
    **WHERE** *column\_name* **IS NOT NULL**;

Пример 1:

```
SELECT *  
  FROM Information  
  WHERE Telephone IS NULL;
```

Пример 2:

```
SELECT *  
  FROM Information  
  WHERE Telephone IS NOT NULL;
```

**Ошибка:** .... **WHERE** Telephone = **NULL** или .... **WHERE** Telephone = **NOT NULL**

т.к. любая операция сравнения с NULL (даже с самим с собой «NULL = NULL»), в результате сравнения выдает значение UNKNOWN (неизвестность).



Оператор SQL состоит из:

- 1) зарезервированных слов;
- 2) пользовательских названий.

Пользовательские названия в Transact SQL должны состоять из символов латинского алфавита, цифр или символа «\_», начинаться с буквы и не могут содержать пробелы.

Возможно включение других символов (@, #, \$ в СУБД SQL Server и #, \$ в СУБД Oracle)

Для обращения к таблице или полю таблицы можно указать составное имя:

**Название\_БД.имя\_владельца.название\_таблицы или название\_таблицы.  
название\_поля**

Каждая из этих характеристик отделяется от предыдущей точкой:

***database.dbowner.table\_name.column\_name;***

Промежуточные значения – имя владельца можно не указывать, если это не приводит к конфликтам имен.

**Выбор базы данных для  
использования:**

USE <название БД>

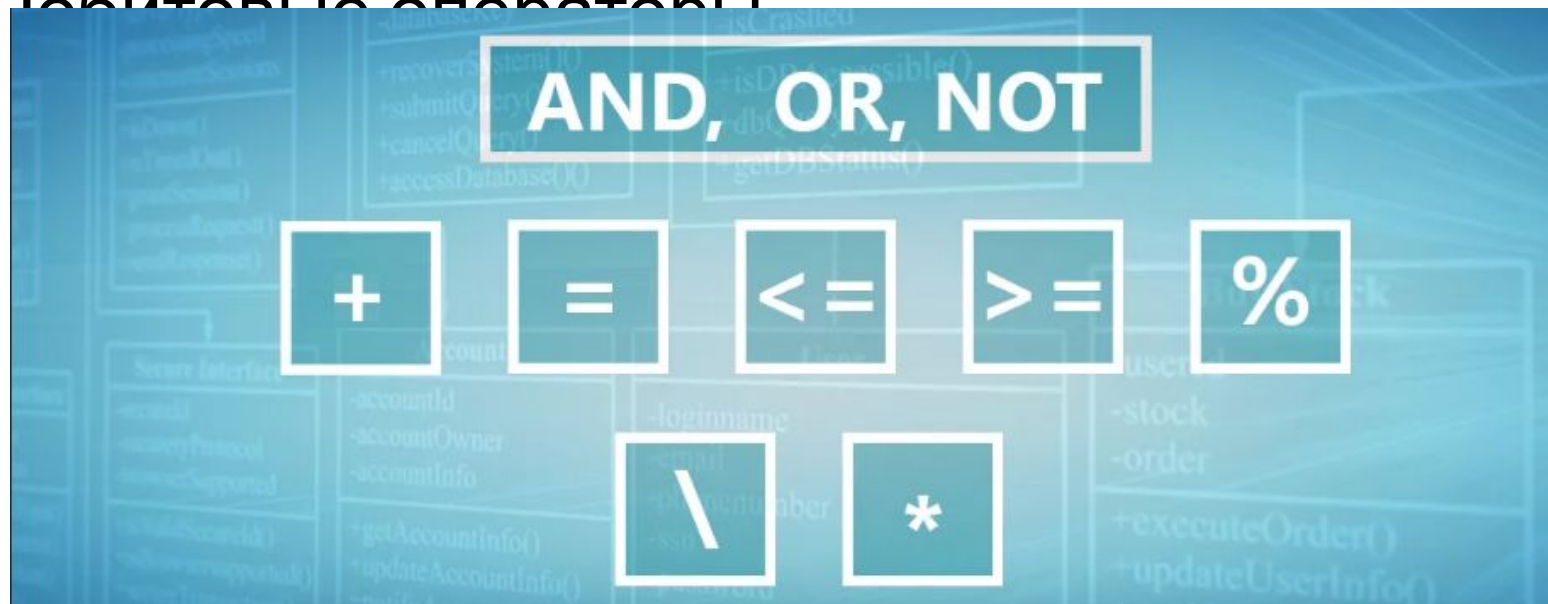
Например, use Sudent\_Ivanov

**Комментарии** в языке Transact - SQL:

1. /\*Текст комментария\*/ –для записи многострочных комментариев.
2. --Текст комментария –для однострочных комментариев.

# Операторы:

1. Арифметические операторы.
2. Операторы присваивания.
3. Операторы сравнения.
4. Логические операторы.
5. Унарные операторы.
6. Побитовые операторы.



# Арифметические операторы

Арифметические операторы выполняют математические операции над двумя значениями одного или разных типов данных.

Результатом выполнения любой арифметической операции со значением **NULL** будет **NULL**.

Арифметический оператор	Действие
+	Сложение
-	Вычитание
/	Деление
*	Умножение
%	Остаток от деления. Возвращает остаток от деления в виде целого числа

# Арифметические операторы. Сложение

Table. Student

ID	Surname	Name	Stipend	...
1	Иванов	Иван	3000	
2	Сидоров	Петр	2500	
3	Синицына	Инна	5000	

**Select** Surname, Name, Stipend+1000  
**From** Student

Surname	Name	Отсутствует имя столбца
Иванов	Иван	4000
Сидоров	Петр	3500
Синицына	Инна	6000

Table. Student

ID	Surname	Name	Stipend	Increase	...
1	Иванов	Иван	3000	1000,50	
2	Сидоров	Петр	2500	NULL	
3	Синицына	Инна	5000	2000	

**Select** Surname, Name, Stipend+ Increase  
**From** Student

Surname	Name	Отсутствует имя столбца
Иванов	Иван	4000,50
Сидоров	Петр	NULL
Синицына	Инна	7000

# Операторы присваивания

Оператор присваивания «=» присваивает значение переменной.

В качестве оператора для присваивания псевдонимов таблицам или заголовкам столбцов применяется ключевое слово **AS** (alias).

Table. Student

ID	Surname	Name	Stipend	...
1	Иванов	Иван	3000	
2	Сидоров	Петр	2500	
3	Синицына	Инна	5000	

```
Select Surname AS Фамилия, Name AS Имя, Stipend AS  
Стипендия  
From Student
```

Фамилия	Имя	Стипендия
Иванов	Иван	3000
Сидоров	Петр	2500
Синицына	Инна	5000

Результат выполненного запроса

Table. Student

ID	Surname	Name	Stipend	...
1	Иванов	Иван	3000	
2	Сидоров	Петр	2500	
3	Синицына	Инна	5000	

```
UPDATE Student  
SET Stipend=1000
```

ID	Surname	Name	Stipend	...
1	Иванов	Иван	1000	
2	Сидоров	Петр	1000	
3	Синицын а	Инна	1000	

Базовая таблица после обновления

# Операторы сравнения

**Операторы сравнения** проверяют равенство или неравенство двух выражений. Результатом операции является булево значение – TRUE или FALSE.

СУБД сверяет все значения выбранного столбца с заданным и, если результат сравнения возвращает TRUE – выводит результат.

Оператор	Описание
=	если левый аргумент равен правому
>	если левый аргумент больше правого
<	если левый аргумент меньше правого
>=	если левый аргумент больше или равен правому
<=	если левый аргумент меньше или равен правому
<>	если левый аргумент не равен правому

# Операторы сравнения

Table. Student

ID	Surname	Name	Stipend	...
1	Иванов	Иван	3000	
2	Сидоров	Петр	2500	
3	Синицына	Инна	5000	

```
Select Surname AS [Фамилия студента], Name AS [Имя студента]  
From Student  
WHERE Stipend >3000
```

Фамилия студента	Имя студента
Синицына	Инна

Table. Student

ID	Surname	Name	Stipend	...
1	Иванов	Иван	3000	
2	Сидоров	Петр	2500	
3	Синицына	Инна	5000	

```
Select Surname AS [Фамилия студента], Name AS [Имя студента]  
From Student  
WHERE Stipend >=3000
```

Фамилия студента	Имя студента
Иванов	Иван
Синицына	Инна

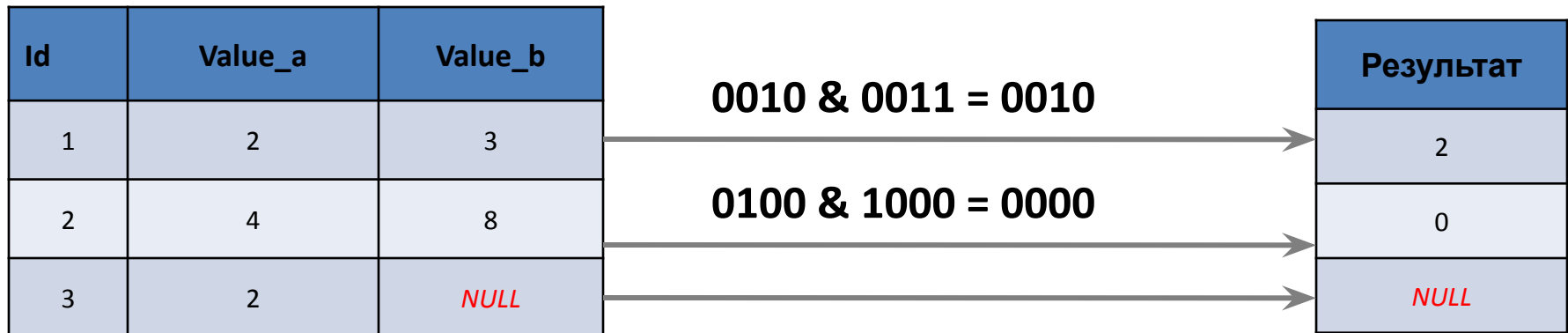
# Побитовые операторы

Побитовые операторы выполняют побитовые действия над двумя выражениями с любым типом данных, относящихся к целочисленному.

Оператор	Название	Описание
&	Побитовое И	Если оба бита в определённой позиции равны 1, результат равен 1
	Побитовое ИЛИ	Если хотя бы один бит в определённой позиции равен 1, результат равен 1
^	Побитовое исключающее ИЛИ	При разных значениях бит в определённой позиции результат равен 1.
~	Побитовое НЕ	Меняет значение бита в каждой позиции на противоположное.

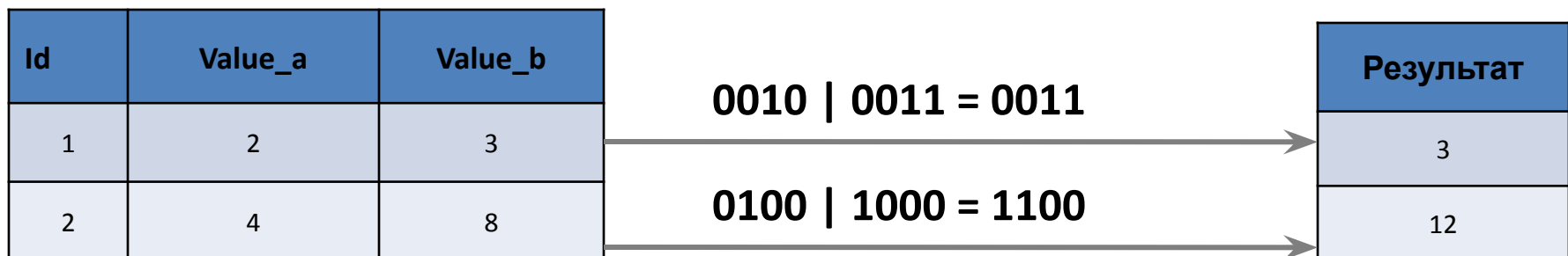


# Побитовое И



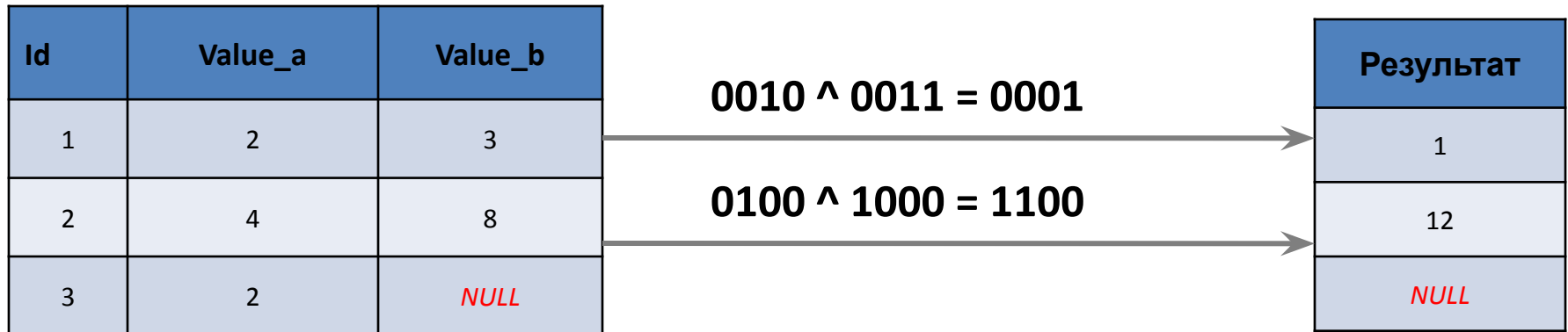
**SELECT** Value\_a & Value\_b AS Результат  
**FROM** Table1

# Побитовое ИЛИ



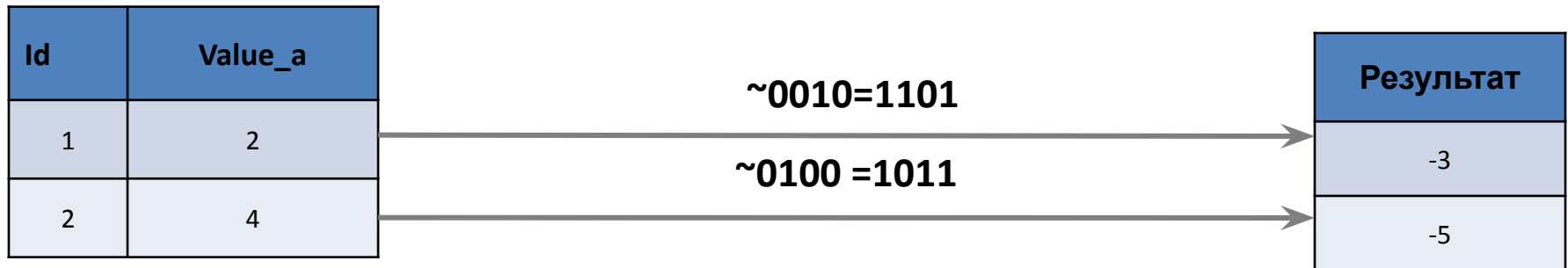
**SELECT** Value\_a | Value\_b AS Результат  
**FROM** Table1

# Побитовое «исключающее ИЛИ»



**SELECT** Value\_a  $\wedge$  Value\_b **AS** Результат  
**FROM** Table1

# Побитовое НЕ



**SELECT**  $\sim$ Value\_a **AS** Результат  
**FROM** Table1

# Логические операторы

**Логические операторы** проверяют истину некоторого условия. Логические операторы возвращают булево значение **TRUE** или **FALSE**.

Логический оператор	Описание
AND	TRUE, если оба булевых выражения дают результат TRUE
OR	TRUE, если любое булево выражение равно TRUE
NOT	Обращает значение любого другого булева оператора
IN	TRUE, если операнд равен одному выражению из списка или одной или несколькими строкам, возвращаемым подзапросом
LIKE	TRUE, если операнд совпадает с шаблоном
BETWEEN	TRUE, если операнд находится внутри диапазона
EXISTS	TRUE, если подзапрос возвращает хотя бы одну строку
ALL	TRUE, если весь набор сравнений дает результат TRUE
ANY	TRUE, если хотя бы одно сравнение из набора дает результат TRUE
SOME	TRUE, если несколько сравнений из набора дают результат TRUE

# Оператор AND (И)

Table. Student

ID	Surname	Name	Stipend	City	...
1	Иванов	Иван	NULL	Уфа	
2	Сидоров	Петр	2500	Ишимбай	
3	Синицын а	Инна	5000	Уфа	
4	Федоров а	Мария	0	Сибай	

Результат

Фамилия	Имя	Стипендия
Синицына	Инна	5000

```
Select Surname AS Фамилия, Name AS Имя, Stipend AS Стипендия  
From Student  
WHERE Stipend>0 AND City ='Уфа'
```

# Оператор OR (ИЛИ)

Table. Student

ID	Surname	Name	Stipend	City	...
1	Иванов	Иван	NULL	Уфа	
2	Сидоров	Петр	2500	Ишимбай	
3	Синицын а	Инна	5000	Уфа	
4	Федоров а	Мария	0	Сибай	

Результат

Фамили я	Имя	Стипенди я	Город
Иванов	Иван	NULL	Уфа
Синицын а	Инна	5000	Уфа
Федоров а	Мария	0	Сибай

```
Select Surname AS Фамилия, Name AS Имя, Stipend AS Стипендия, City AS  
Город  
From Student  
WHERE City = 'Сибай' OR City = 'Уфа'
```

# Оператор NOT (НЕ)

Table. Student

ID	Surname	Name	Stipend	City	...
1	Иванов	Иван	NULL	Уфа	
2	Сидоров	Петр	2500	Ишимбай	
3	Синицын а	Инна	5000	Уфа	
4	Федоров а	Мария	0	Сибай	

Результат

Фамилия	Имя	Стипендия	Город
Сидоров	Петр	2500	Ишимбай
Федоров а	Мария	0	Сибай

```
Select Surname AS Фамилия, Name AS Имя, Stipend AS Стипендия, City AS Город
From Student
WHERE NOT City = 'Уфа'
```

# Оператор IN

**Операторы IN** (равен любому из списка) и **NOT IN** (не равен ни одному из списка) используются для сравнения проверяемого значения поля с заданным списком. Этот список значений указывается в скобках справа от оператора IN.

Table. Student

ID	Surname	Name	Stipend	City	...
1	Иванов	Иван	NULL	Уфа	
2	Сидоров	Петр	2500	Ишимбай	
3	Синицын а	Инна	5000	Уфа	
4	Федоров а	Мария	0	Сибай	

Результат

Фамилия	Имя	Стипендия	Город
Иванов	Иван	NULL	Уфа
Синицын а	Инна	5000	Уфа
Федоров а	Мария	0	Сибай

```
Select Surname AS Фамилия, Name AS Имя, Stipend AS Стипендия, City AS Город
From Student
WHERE City IN ('Уфа', 'Сибай')
```

# Оператор NOT IN

**Операторы IN** (равен любому из списка) и **NOT IN** (не равен ни одному из списка) используются для сравнения проверяемого значения поля с заданным списком. Этот список значений указывается в скобках справа от оператора IN.

Table. Student

ID	Surname	Name	Stipend	City	...
1	Иванов	Иван	NULL	Уфа	
2	Сидоров	Петр	2500	Ишимбай	
3	Синицын а	Инна	5000	Уфа	
4	Федоров а	Мария	0	Сибай	

Результат

Фамилия	Имя	Стипенди я	Город
Сидоров	Петр	2500	Ишимба й

```
Select Surname AS Фамилия, Name AS Имя, Stipend AS Стипендия, City AS Город
From Student
WHERE NOT City IN ('Уфа', 'Сибай')
```



# Оператор LIKE

Оператор **LIKE** просматривает строковые значения полей с целью определения, входит ли заданная в операторе подстрока (образец поиска) в символьную строку-значение проверяемого поля.

Можно применять шаблон искомого образца строки, использующий следующие символы:

- символ подчеркивания «**\_**», определяет возможность наличия в указанном месте одного любого символа;
- символ «**%**» допускает присутствие в указанном месте проверяемой строки последовательности любых символов произвольной длины.

Если необходимо включить в образец символы «**\_**» или «**%**» для этого с помощью ключевого слова **ESCAPE** нужно определить так называемый escape-символ, чаще для этой цели применяют символы "@" или "~".

Table. Student

ID	Surname	Name	Stipend	City	...
1	Иванов	Иван	NULL	Уфа	
2	Сидоров	Петр	2500	Ишимбай	
3	Синицын а	Инна	5000	Уфа	
4	Федоров а	Мария	0	Сибай	

Результат

Фамилия	Имя	Стипендия
Синицына	Инна	5000

```
Select Surname AS 'Фамилия', Name AS 'Имя', Stipend AS 'Стипендия'  
From Student  
WHERE Surname LIKE 'C%' AND Surname NOT LIKE '%@_%' ESCAPE '@'
```

# Оператор BETWEEN

**Оператор BETWEEN** используется для проверки условия вхождения значения поля в заданный интервал, то есть вместо списка значений атрибута этот оператор задает границы его изменения.

Table. Student

ID	Surname	Name	Stipend	City	Birthday	...
1	Иванов	Иван	NULL	Уфа	01.01.2004	
2	Сидоров	Петр	2500	Ишимбай	20.01.2003	
3	Синицына	Инна	5000	Уфа	3.01.2003	
4	Федорова	Мария	0	Сибай	02.02.2004	

Результат

Surname	Name	City	Дата рождения
Иванов	Иван	Уфа	01.01.2004
Федорова	Мария	Сибай	02.02.2004

```
Select Surname AS Фамилия, Name AS Имя, City AS Город, Birthday AS [Дата рождения]
From Student
WHERE Birthday BETWEEN '01.01.2004' AND '31.12.2004'
```

Table. Student

ID	Surname	Name	Stipend	City	Birthday	...
1	Иванов	Иван	NULL	Уфа	01.01.2004	
2	Сидоров	Петр	2500	Ишимбай	20.01.2003	
3	Синицына	Инна	5000	Уфа	3.03.2003	
4	Федорова	Мария	0	Сибай	02.02.2004	

Результат

Surname	Name	City	Дата рождения
Сидоров	Петр	Ишимбай	20.01.2003

```
Select Surname AS Фамилия, Name AS Имя, City AS Город, Birthday AS [Дата рождения]
From Student
WHERE Birthday BETWEEN '01.01.2003' AND '31.01.2003'
```

# Унарные операторы

Оператор	Действие
+	Числовое значение становится положительным
-	Числовое значение становится
~	Поразрядное НЕ. Возвращает двоичное дополнение числа

# Примеры, унарные операторы

Id	Value_a		Результат
1	2	→	-2
2	4	→	-4

**SELECT** -Value\_a **AS** Результат  
**FROM** Table1

Id	Value_a		Результат
1	2	→	-3
2	4	→	-5

**SELECT** ~Value\_a **AS** Результат  
**FROM** Table1

# Приоритет операторов

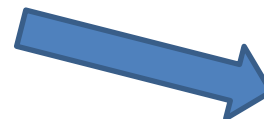
1. () – выражения в скобках.
2. +, -, ~ – унарные операторы.
3. \*, /, % – арифметические операторы.
4. +, - – арифметические операторы.
5. =, >, <, >=, <=, <> – операторы сравнения.
6. ^ (побитное исключающее ИЛИ), & (побитное И), | (побитное ИЛИ).
7. NOT.
8. AND.
9. ALL, ANY, SOME, BETWEEN, IN, LIKE, OR.
10. = – присваивание значения переменной.

Id	a	b
1	2	3
2	4	8

**Пример.** Вычислим сумму

```
SELECT a + b * 2 AS ИТОГО  
FROM Table1
```

```
SELECT ( a + b ) * 2 AS  
ИТОГО  
FROM Table1
```



ИТОГ О
8
20

ИТОГ О
10
24

# Задачи

Table. Product

ID	Product_name	Manufacturer	Price	Number	Date_of_delivery
1	Мяч	Torneo	1000	10	11.01.2021
2	Лыжи	Fischer	5900	1	23.12.2020
3	Коньки	Nordway	1499	5	10.12.2020
4	Лыжи	Salomon	5000	8	1.12.2020
5	Сноуборд	Termit	8900	4	18.11.2020
6	Лыжи	Madshus	3000	3	10.11.2020

Какие данные будут получены в результате выполнения запросов?

1) SELECT \*

FROM Product

WHERE Price<5000 OR NOT(Date\_of\_delivery<'20.12.2020')

2) SELECT Product\_name, Price\*Number

FROM Product

WHERE Date\_of\_delivery BETWEEN '20.12.2020' AND '01.02.2021'

3) SELECT \*

FROM Product

WHERE NOT (Number>5) AND Product\_name LIKE '\_ы%'

4) SELECT Product\_name, Price\*Number

FROM Product

WHERE (Price\*Number)>6000 AND Number IN(1,2,3)

# Типы данных MS SQL Server

## Числовые типы данных:

**BIT**: хранит значение 0 или 1. Фактически является аналогом булевого типа в языках программирования. Занимает 1 байт.

**TINYINT**: хранит числа от 0 до 255. Занимает 1 байт. Хорошо подходит для хранения небольших чисел.

**SMALLINT**: хранит числа от -32 768 до 32 767. Занимает 2 байта

**INT**: хранит числа от -2 147 483 648 до 2 147 483 647. Занимает 4 байта. Наиболее используемый тип для хранения чисел.

**BIGINT**: хранит очень большие числа от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807, которые занимают в памяти 8 байт.

**DECIMAL ( p , s ) ] и NUMERIC ( p , s ) ]**: числа с фиксированной точностью и масштабом. При использовании максимальной точности числа могут принимать значения в диапазоне от  $-10^{38}+1$  до  $10^{38}-1$ .

Синонимами типа DECIMAL по стандарту ISO является тип DEC(p, s).

Тип NUMERIC функционально эквивалентен типу DECIMAL.

**p (точность)** – максимальное общее число хранимых десятичных разрядов. Это число включает символы слева и справа от десятичной запятой. Точность должна быть значением в диапазоне от 1 до максимум 38. Точность по умолчанию составляет 18.

**s (масштаб)** – максимальное число хранимых десятичных разрядов справа от десятичной запятой. Это число отнимается от p для определения максимального количества цифр слева от десятичной запятой. Масштаб должен иметь значение от 0 до p и может быть указан только при заданной точности. По умолчанию масштаб принимает значение 0, поэтому  $0 \leq s \leq p$ . Максимальный размер хранилища зависит от точности.

**SMALLMONEY**: хранит дробные значения от -214 748.3648 до 214 748.3647. Предназначено для хранения денежных величин. Занимает 4 байта. Эквивалентен типу DECIMAL(10,4).

**MONEY**: хранит дробные значения от -922 337 203 685 477.5808 до 922 337 203 685 477.5807.

Представляет денежные величины и занимает 8 байт. Эквивалентен типу DECIMAL(19,4).

**FLOAT**: хранит числа от  $-1.79E+308$  до  $1.79E+308$ . Занимает от 4 до 8 байт в зависимости от дробной части.

Может иметь форму определения в виде FLOAT(n), где n представляет число бит, которые используются для хранения десятичной части числа (мантиссы). По умолчанию n = 53.

**REAL**: хранит числа от  $-3.40E+38$  до  $3.40E+38$ . Занимает 4 байта. Эквивалентен типу FLOAT(24).

# Типы данных MS SQL Server

## Типы данных, представляющие дату и время:

**DATE:** ГГГГ-ММ-ДД. Хранит даты от 1 января 0001 года до 31 декабря 9999 года. Занимает 3 байта.

**DATETIME:** хранит даты и время от 01/01/1753 до 31/12/9999. Занимает 8 байт.

**DATETIME2:** ГГГГ-ММ-ДД чч:мм:сс[.доли секунды], хранит даты и время в диапазоне от 01/01/0001 00:00:00.0000000 до 31/12/9999 23:59:59.9999999.

Занимает от 6 до 8 байт в зависимости от точности времени.

Может иметь форму DATETIME2(n), где n представляет количество цифр от 0 до 7 в дробной части секунд.

**SMALLDATETIME:** хранит даты и время в диапазоне от 01/01/1900 до 06/06/2079, то есть ближайшие даты. Занимает от 4 байта.

**DATETIMEOFFSET:** хранит даты и время в диапазоне от 0001-01-01 до 9999-12-31. Сохраняет детальную информацию о времени с точностью до 100 наносекунд. Занимает 10 байт.

**TIME:** хранит время в диапазоне от 00:00:00.0000000 до 23:59:59.9999999. Занимает от 3 до 5 байт.

Может иметь форму TIME(n), где n представляет количество цифр от 0 до 7 в дробной части секунд



# Типы данных MS SQL Server

## Строковые типы данных:

**CHAR:** хранит строку длиной от 1 до 8 000 символов. На каждый символ выделяет по 1 байту. Не подходит для многих языков, так как хранит символы не в кодировке Unicode.

Количество символов, которое может хранить столбец, передается в скобках. Например, для столбца с типом CHAR(10) будет выделено 10 байт. И если мы сохраним в столбце строку менее 10 символов, то она будет дополнена пробелами.

**VARCHAR:** хранит строку. На каждый символ выделяется 1 байт. Можно указать конкретную длину для столбца - от 1 до 8 000 символов, например, VARCHAR(10). Если строка должна иметь больше 8000 символов, то задается размер MAX, а на хранение строки может выделяться до 2 Гб: VARCHAR(MAX).

Не подходит для многих языков, так как хранит символы не в кодировке Unicode.

В отличие от типа CHAR если в столбец с типом VARCHAR(10) будет сохранена строка в 5 символов, то в столбце будет сохранено именно пять символов.

**NCHAR:** хранит строку в кодировке Unicode длиной от 1 до 4 000 символов. На каждый символ выделяется 2 байта. Например, NCHAR(15)

**NVARCHAR:** хранит строку в кодировке Unicode. На каждый символ выделяется 2 байта.

Можно задать конкретный размер от 1 до 4 000 символов: . Если строка должна иметь больше 4000 символов, то задается размер MAX, а на хранение строки может выделяться до 2 Гб.

Еще два типа **TEXT** и **NTTEXT** являются устаревшими и поэтому их не рекомендуется использовать. Вместо них применяются VARCHAR и NVARCHAR соответственно

# Типы данных MS SQL Server

## Бинарные типы данных:

**BINARY**: хранит бинарные данные в виде последовательности от 1 до 8 000 байт.

**VARBINARY**: хранит бинарные данные в виде последовательности от 1 до 8 000 байт, либо до  $2^{31}-1$  байт при использовании значения MAX (VARBINARY(MAX)).

Еще один бинарный тип - тип IMAGE является устаревшим, и вместо него рекомендуется применять тип VARBINARY.

## Остальные типы данных:

**UNIQUEIDENTIFIER**: уникальный идентификатор GUID (по сути строка с уникальным значением), который занимает 16 байт.

**TIMESTAMP**: некоторое число, которое хранит номер версии строки в таблице. Занимает 8 байт. В новых версиях СУБД заменен на rowversion.

**CURSOR**: представляет набор строк.

**HIERARCHYID**: представляет позицию в иерархии.

**SQL\_VARIANT**: может хранить данные любого другого типа данных T-SQL.

**XML**: хранит документы XML или фрагменты документов XML. Занимает в памяти до 2 Гб.

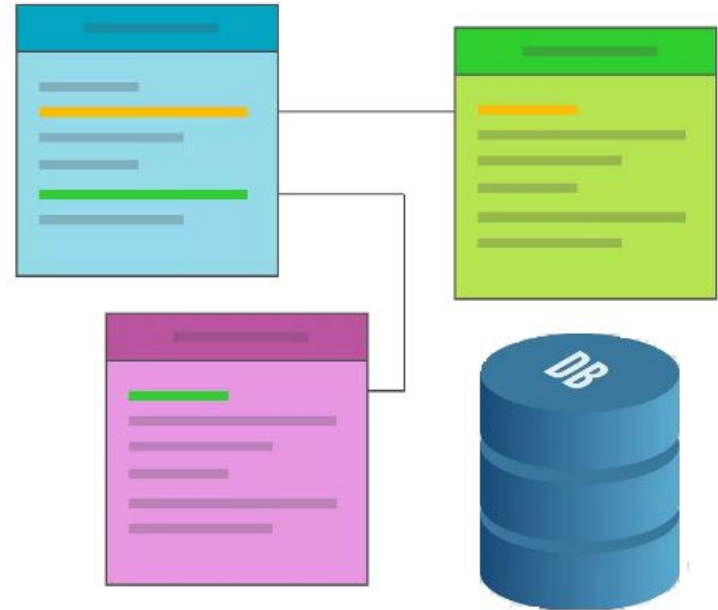
**TABLE**: представляет определение таблицы.

**GEOGRAPHY**: хранит географические данные, такие как широта и долгота.

**GEOMETRY**: хранит координаты местонахождения на плоскости.

# Проектирование баз данных

**Проектирование баз данных** — процесс разработки схемы базы данных и определения необходимых ограничений целостности.



## **Основные задачи:**

- 1) Сохранить необходимые данные о конкретной предметной области.
- 2) Получить данные по всем необходимым запросам.
- 3) Сократить избыточность дублирования данных.
- 4) Обеспечить целостности данных.

# Проблемы, возникающие при проектировании БД

Нужно добавить новый отдел, а сотрудников пока не набрали=> **аномалия добавления.**

Т.к. чтобы добавить новый отдел без сотрудника нужно будет присвоить значение NULL в соответствующей строке поля *Табельный номер сотрудника*, но так как поле *Табельный номер сотрудника* является первичным ключом отношения, СУБД отклонит попытку добавления такой записи.

При изменении названия отдела или номера телефона => **аномалия модификации.**

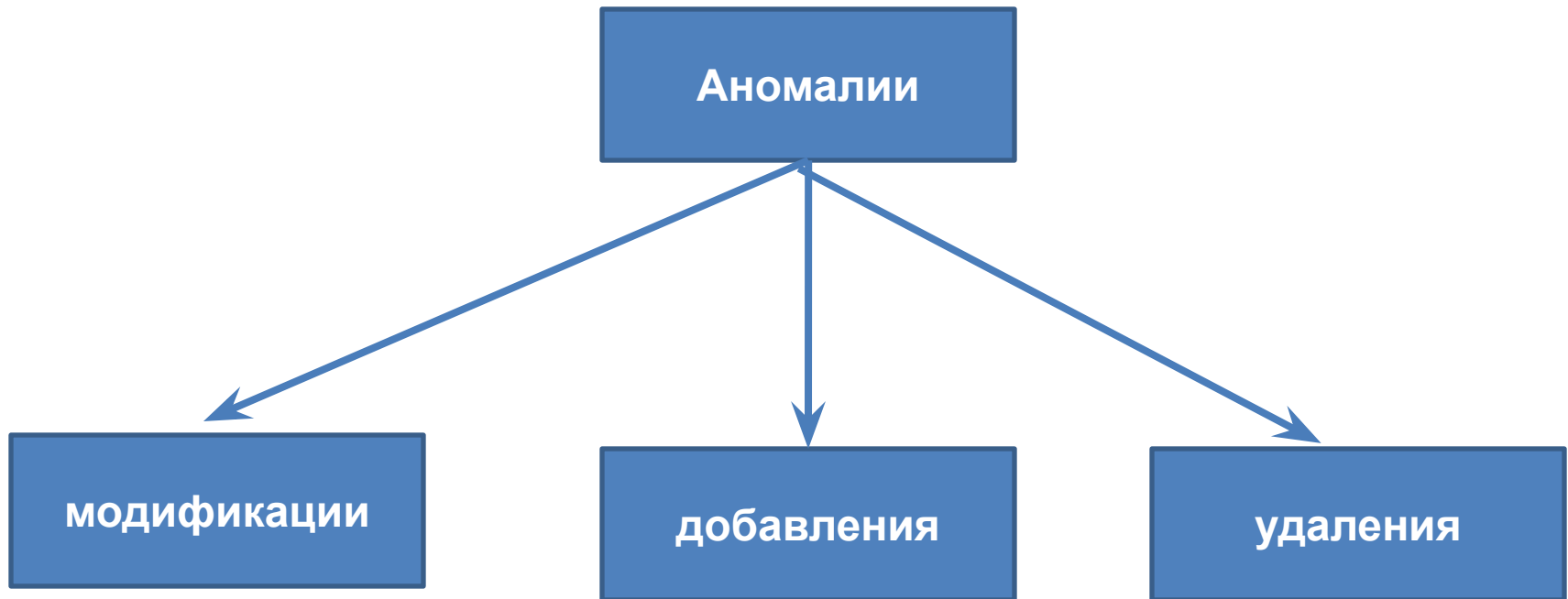
Если в отделе работает всего один сотрудник и он увольняется => **аномалия удаления.**

Таблица «Сотрудник

Табельный номер сотрудника (РК)	Фамилия	Имя	Отчество	Отдел	Телефон отдела
1	Иванов	Иван	Иванович	Администрация	12345
2	Сидоров	Петр	Петрович	Информационный отдел	54321
3	Синицына	Инна	Петровна	Отдел кадров	23431
4	Егоров	Валерий	Игнатьевич	Проектный отдел	45673
5	Воронина	Наталья	Игоревна	Отдел кадров	23431

# Аномалии в таблицах БД

При неправильно спроектированной схеме реляционной БД могут возникнуть аномалии при выполнении операций модификации, добавления, удаления данных.



# Решение проблемы

Декомпозиция

таблица «Сотрудник отдела»

Табельный номер сотрудника (РК)	Фамилия	Имя	Отчество	Отдел	Телефон отдела
1	Иванов	Иван	Иванович	Администрация	12345
2	Сидоров	Петр	Петрович	Информационный отдел	54321
3	Синицына	Инна	Петровна	Отдел кадров	23431
4	Егоров	Валерий	Игнатьевич	Проектный отдел	45673
5	Воронина	Наталья	Игоревна	Отдел кадров	23431

таблица «Сотрудник»

Код (РК)	Табельный номер сотрудника	Фамилия	Имя	Отчество	Код отдела (FK)
1	1	Иванов	Иван	Иванович	100
2	2	Сидоров	Петр	Петрович	101
3	3	Синицына	Инна	Петровна	102
4	4	Егоров	Валерий	Игнатьевич	103
5	5	Воронина	Наталья	Игоревна	102

таблица «Отдел»

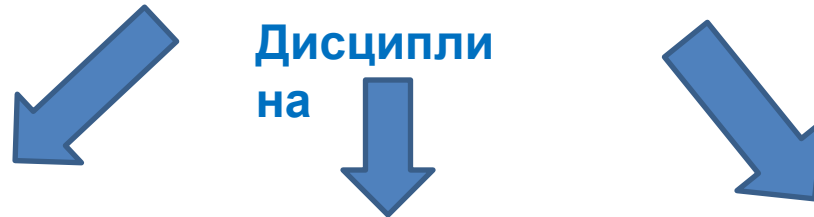
Код отдела (РК)	Отдел	Телефон отдела
100	Администрация	12345
101	Информационный отдел	54321
102	Отдел кадров	23431
103	Проектный отдел	45673

# Проектирование баз данных

**Нормализация** – это процесс преобразования отношения в состояние, обеспечивающее лучшие условия выборки, добавления, изменения и удаления данных.

**Цель:** устранение избыточности данных в базе данных.

Не допускается наличие в таблице полей, названия которых входят в одно множество допустимых значений => не эффективная таблица.



Ко д	Семестр	Математика, часы	Физика, часы	Физическая культура, часы
11	1	60	46	42
12	2	62	36	32
13	3	54	56	40

Пример, таблица «Учебный план ВУЗа»

# Проектирование баз данных

таблица «Учебный план ВУЗа»

Код	Семестр	Математика, часы	Физика, часы	Физическая культура, часы
11	1	60	46	42
12	2	62	36	32
13	3	54	56	40



Код	Дисциплина	Часы	Семестр
11	Математика	60	1
12	Физика	62	1
13	Физическая культура	64	1
14	Математика	46	2
15	Физика	36	2
16	Математика	56	3
17	Физика	42	3
18	Физическая культура	32	2
19	Физическая культура	40	3

**Вопрос:**  
какие  
аномалии  
могут  
возникнуть  
в данной  
таблице?

Появляется избыточность данных => что можно предпринять? (см. следующий слайд)



Код	Дисциплина	Часы	Семестр
11	Математика	60	1
12	Физика	62	1
13	Физическая культура	64	1
14	Математика	46	2
15	Физика	36	2
16	Математика	56	3
17	Физика	42	3
18	Физическая культура	32	2
19	Физическая культура	40	3

таблица  
«Учебный  
план  
ВУЗа»

Вопросы:

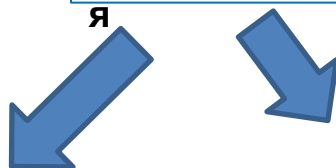
1) Сколько таблиц будет получено в результате декомпозиции. С какими полями?

2) Как эти таблицы будут связаны между собой?

Код (РК)	Дисциплина
11	Математика
12	Физика
13	Физическая культура

Декомпозици

я



Код (РК)	Код Дисциплины (FK)	Часы	Семестр
1	11	60	1
2	12	62	1
3	13	64	1
4	11	46	2
5	12	36	2
6	11	56	3
7	12	42	3
8	13	32	2
9	13	40	3

таблица

«Дисциплина»

таблица «Учебный план  
ВУЗа»

# Подход NoSQL

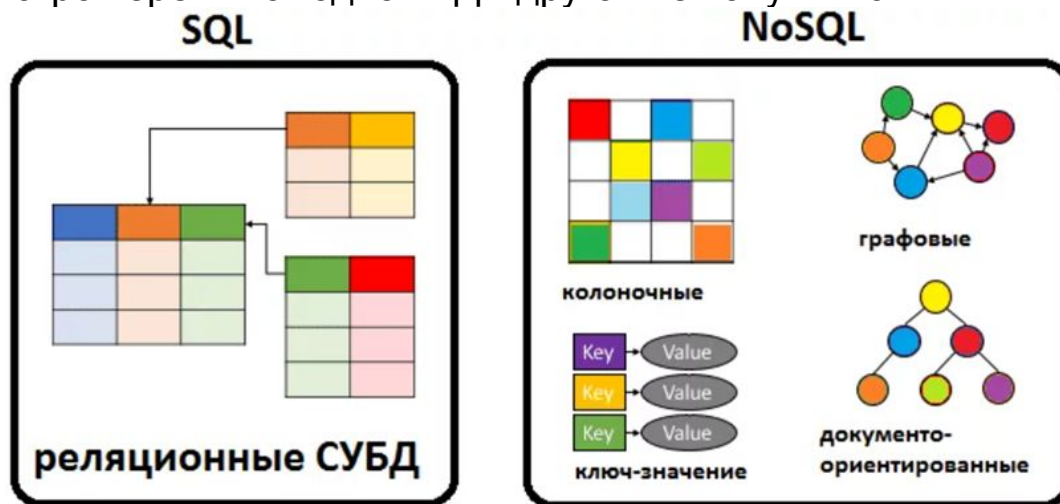
**Термин NoSQL** обозначает нереляционные базы данных, которые хранят данные в формате, отличном от реляционных таблиц.

**Термин NoSQL** принято переводить, как «не только SQL», поскольку некоторые из этих баз данных поддерживают запросы, совместимые с SQL.

В базах данных NoSQL применяются модели данных, оптимизированные под решение конкретных задач.

Базы данных NoSQL из-за высокой масштабируемости и высокой доступности используются в веб-приложениях реального времени, больших данных, в онлайн-играх, в проектах Интернет вещей, социальных сетях, поисковых системах, приложениях для онлайн-рекламы.

У каждой БД NoSQL есть свои правила для работы с данными, а также языки, соответственно быстро перейти от одной БД к другой не получится.



**Big Data** или **большие данные** — это структурированные или неструктурированные массивы данных большого объема. Их обрабатывают при помощи специальных автоматизированных инструментов, чтобы использовать для статистики, анализа, прогнозов и принятия решений.

**Интернет вещей** (IoT, Internet of Things) — объединение разных устройств в общую сеть, в которой они могут собирать информацию, обрабатывать ее и обмениваться данными между собой, с человеком и серверами в дата-центре или облаке.

# Модель данных «Ключ-значение»

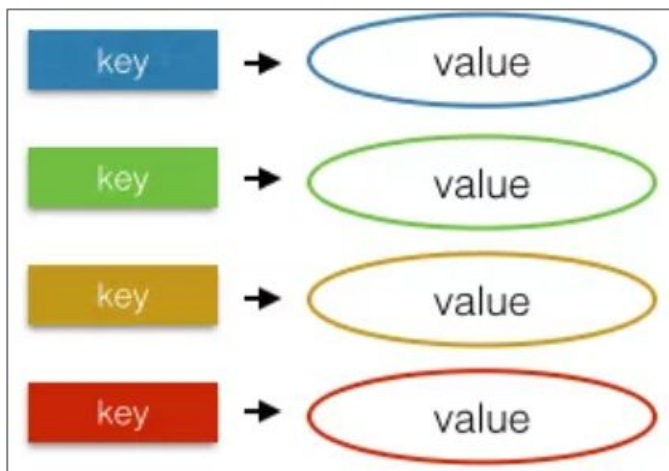
Большинство БД поддерживают только самые простые операции запроса, вставки и удаления. Чтобы частично или полностью изменить значение, приложение всегда перезаписывает существующее значение целиком. В большинстве реализаций атомарной операцией считается чтение или запись одного значения. Запись больших значений занимает относительно долгое время.

Поиск по значениям отсутствует, есть только по ключу/ключам. Все сведения о схеме поддерживаются и применяются на уровне приложения.

Главные плюсы: масштабируемость, простота.

Основные недостатки: не поддерживаются связи между объектами, в основном запросы только с поиском по ключу.

Возможно применить для хранения изображений, сессий, счетчиков посещений или просмотров, в игровых и рекламных приложениях, в проектах интернет вещей и т.д.



Key	Value (Opaque)
User:2:friends	{23, 76, 233, 11}
User:2:settings	Theme: dark, cookies: false
User:3:friends	[234, 3466, 86, 55]

в ключе могут храниться данные и настройки учетной записи

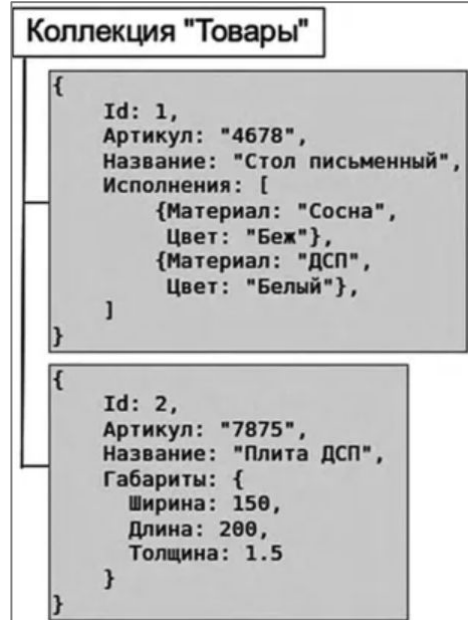
# Документно-ориентированная модель данных

Возможно реализовать большую вложенность и сложность структуры данных, чем в БД «ключ-значение» (например, документ вложенный в документ, вложенный в документ). Данные хранятся в Json документах.

Основные достоинства: возможно хранение слабоструктурированных объектов; быстрое выполнение запросов.

Основные недостатки: при запросе выводиться весь документ, даже если необходимо было какое-то одно значение, что сказывается на производительности; медленный процесс обновления данных (данные могут храниться распределено на нескольких серверах); возможно дублирование данных.

Возможно применение для каталогов, пользовательские профили, в CMS-системах, издательском деле и документальном поиске.



**CMS** (Content Management System) — это система управления, движок, платформа или конструктор, который позволяет управлять содержимым сайта.

# Колоночная модель данных

Основная идея колоночной модели данных — это хранение данных не по строкам, как в реляционных таблицах, а по колонкам. Это означает, что с точки зрения SQL-клиента данные представлены как обычно в виде таблиц, но физически эти таблицы являются совокупностью колонок, каждая из которых по сути представляет собой таблицу из одного поля.

Достоинства: возможность хранить большое количество данных с большим количеством атрибутов, скорость выполнения запросов.

Недостатки: скорость выполнения сложных запросов на больших объемах данных, скорость изменение структуры таблиц с данными.

Колоночные СУБД применяются как правило в аналитических системах.

CustomerID	Column Family: Identity
001	First name: Mu Bae Last name: Min
002	First name: Francisco Last name: Vila Nova Suffix: Jr.
003	First name: Lena Last name: Adamczyk Title: Dr.

CustomerID	Column Family: Contact Info
001	Phone number: 555-0100 Email: someone@example.com
002	Email: vilanova@contoso.com
003	Phone number: 555-0120

← Семейство колонок

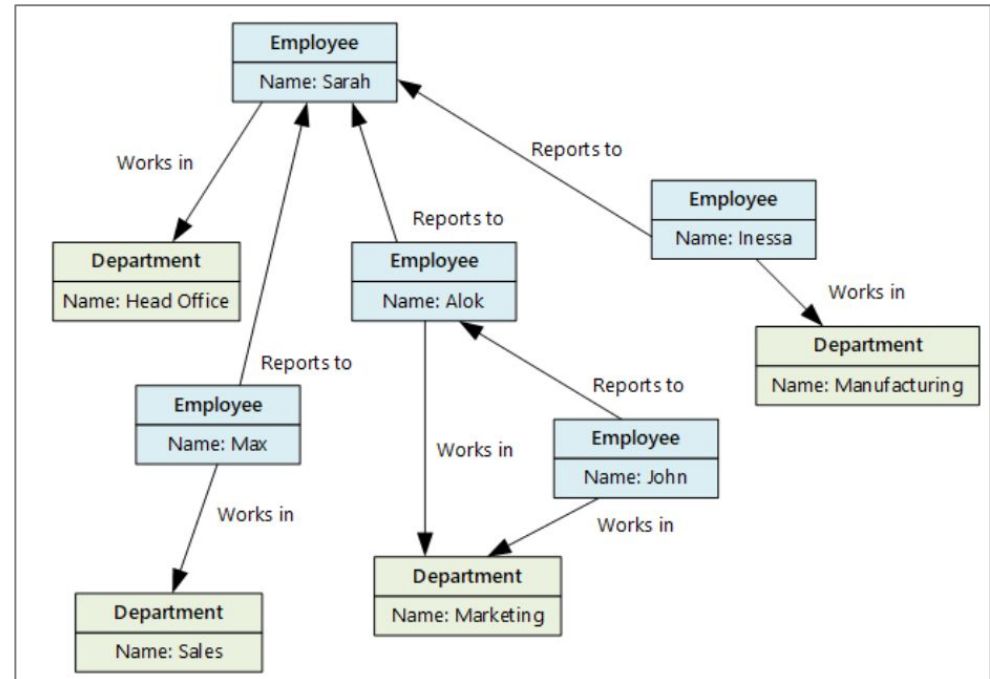
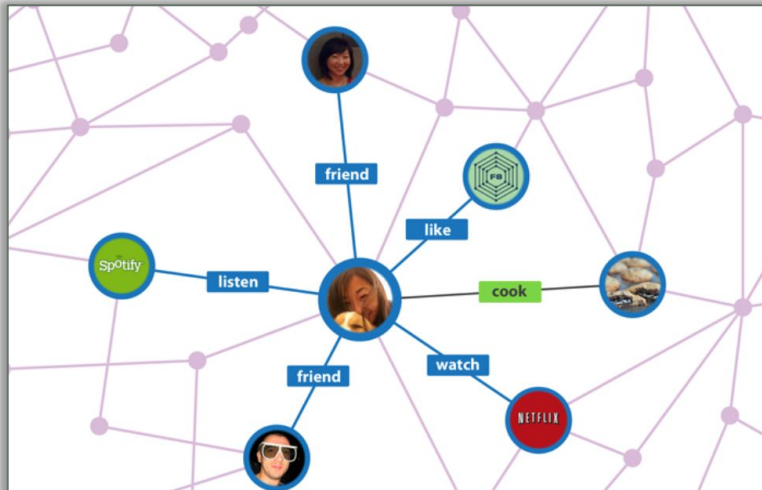
↘ Реляционная таблица

CustomerID	First name	Last name	Suffix	Title	Phone Number	Email
001	Mu Bae	Min			555-0100	someone@example.com
002	Francisco	Vila Nova	Jr.			
003	Lena	Adamczyk		Dr.	555-0120	vilanova@contoso.com

# Графовая модель данных

**Графовая модель данных** основана на узлах и рёбрах, представляющих взаимосвязанные данные (например, отношения между людьми в социальной сети), они упрощают хранение и навигацию по сложным отношениям.

Возможно применение в задачах, ориентированных на связи: социальные сети, выявление мошенничества, маршруты общественного транспорта, дорожные карты и т.п.



Узлы: сотрудники и отделы.

Рёбра: определяют отношения подчинения и отдел, в котором работает каждый сотрудник.

Стрелки: показывают направление связей.



# Основные черты

Традиционные реляционные СУБД основаны на принципах ACID:

Atomicity - атомарность

Consistency - согласованность

Isolation – изолированность

Durability - надежность

СУБД с подходом NoSQL основаны на принципах BASE:

Basic Availability - базовая доступность — каждый запрос гарантированно завершается (успешно или безуспешно).

Soft State - гибкое состояние — состояние системы может изменяться со временем, даже без ввода новых данных, для достижения согласования данных.

Eventual Consistency - согласованность в конечном счете — данные могут быть некоторое время рассогласованы, но приходят к согласованию через некоторое время.



# Примеры NoSQL СУБД

1. Документно-ориентированные: CouchDB (Couchbase), MongoDB (MongoDB).
2. Колоночные: Cassandra (Apache Software Foundation), ClickHouse (Яндекс).
3. Графовые: Neo4j (Neo Technology), OrientDB (Orient Technologies LTD).
4. Ключ – значение: Tarantool (VK), Redis (Redis Labs), Oracle NOSQL Database (Oracle), Amazon DynamoDB (Amazon).





# Источники информации

- К.Дж. Дейт. Введение в системы баз данных. Восьмое издание. – М.: Вильямс, 2005. – 1328 С.
- Голицина О.Л. Базы данных. - Изд. «ФОРУМ», 2009. - 400 с.
- Дунаев В.В. Базы данных. Язык SQL для студента. - СПб.: БХВ-Петербург, 2007. - 320 с.
- Советов Б.Я. Базы данных. – Изд. «Высшая школа», 2007. - 463 с.
- Харрингтон Д. Проектирование объектно-ориентированных баз данных. Год 2007. "Лань" Электронная библиотечная система
- Техническая документация Microsoft:  
<https://docs.microsoft.com/ru-ru/>
- Интерактивные учебник по SQL: <http://www.sql-tutorial.ru/ru/>
- Национальная библиотека им. Н. Э. Баумана:  
<https://ru.bmstu.wiki>
- К.Ю. Поляков и т.д.