Что такое **реляционная базы данных (РБД)**. Слово "реляция" происходит от "relation", то есть "отношение". Это означает, что в РБД существуют механизмы установления связей между таблицами. Делается это с помощью так называемых первичных и внешних ключей.

*Первичный ключ* (сокращенно РК - primary key) - столбец, значения которого во всех строках различны. Первичные ключи могут быть логическими (естественными) и суррогатными (искусственными). Так, для нашей таблицы Пользователи первичным ключом может стать столбец e-mail (ведь теоретически не может быть двух пользователей с одинаковым e-mail). На практике лучше использовать суррогатные ключи, т.к. их применение позволяет абстрагировать ключи от реальных данных. Кроме того, первичные ключи менять нельзя, а что если у пользователя сменится e-mail?

*Суррогатный ключ* представляет собой дополнительное поле в базе данных. Как правило, это порядковый номер записи (хотя вы можете задавать их на свое усмотрение, контролируя, чтобы они были уникальны)

**Реляционные базы данных** состоят из таблиц. Каждая таблица состоит из столбцов (их называют полями или атрибутами) и строк (их называют записями или кортежами). Таблицы в реляционных базах данных обладают рядом свойств. Основными являются следующие:

В таблице не может быть двух одинаковых строк. В математике таблицы, обладающие таким свойством, называют отношениями - по-английски relation, отсюда и название – реляционные.

Столбцы располагаются в определенном порядке, который создается при создании таблицы. В таблице может не быть ни одной строки, но обязательно должен быть хотя бы один столбец.

У каждого столбца есть уникальное имя (в пределах таблицы), и все значения в одном столбце имеют один тип (число, текст, дата...).

На пересечении каждого столбца и строки может находиться только атомарное значение (одно значение, не состоящее из группы значений). Таблицы, удовлетворяющие этому условию, называют нормализованными.

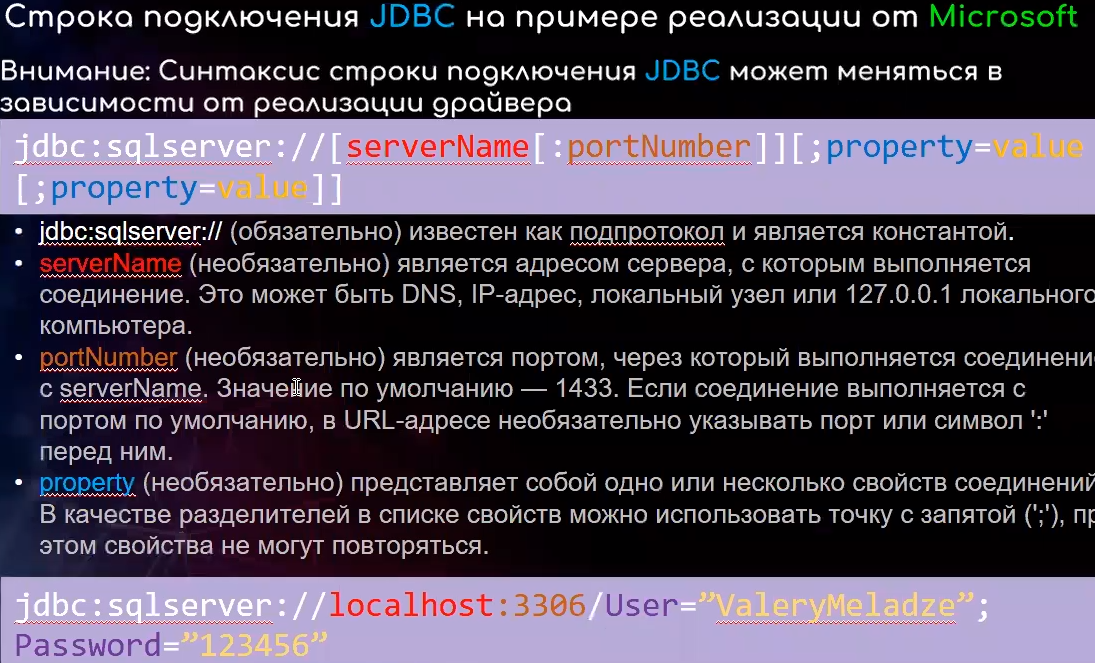
**Например, у нас есть 2 таблицы. В первой таблице – “должности” есть Primary Key с такими записями (Developer, QA, HR), а во второй находится их отдел где они работают и связаны они по Foreign Key. Так вот если мы решим в отдел добавить сотрудника, с той должностью которой нет в первичной таблице, то это сделать будет нельзя. В начале нужно добавить новую должность в таблицу с Primary Key, и только потом добавлять отделы с новой должностью**

**Connection URL**

**JDBC** (англ. **Java DataBase Connectivity** — *соединение с базами данных на***Java**) — позволяет **Java** приложениям получить доступ к данным различных СУБД.

Если говорить коротко, то JDBC позволяет делать три простые вещи:

1. Установить соединение с базой данных
2. Посылать запросы и изменять состояние базы данных
3. Обрабатывать результаты запросов



**Языки в SQL**

Data Query Language (DQL) – язык для выборки из таблицы (SELECT)

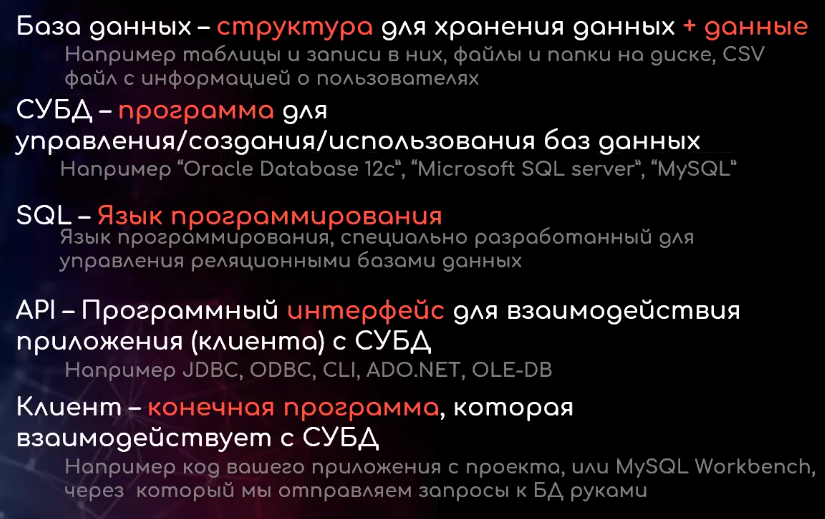
Data Definition Language (DDL) – язык описания данных (CREATE, ALTER, DROP) – добавление столбцов в таблицу

Data Manipulation Language (DML) – язык манипулирования данными (INSERT, UPDATE, DELETE) – вставка значений в столбцы

Data Control Language (DCL) – язык управления доступом к данным (описывает какие пользователи могут изменять таблицы) (GRANT, REVOKE)

Для доступа к БД нужно развернуть локальный сервер (логично, так как БД работает только в связке с сервером) - WAMP

Для mac нужен MAMP – mac, apache, MySQL, php



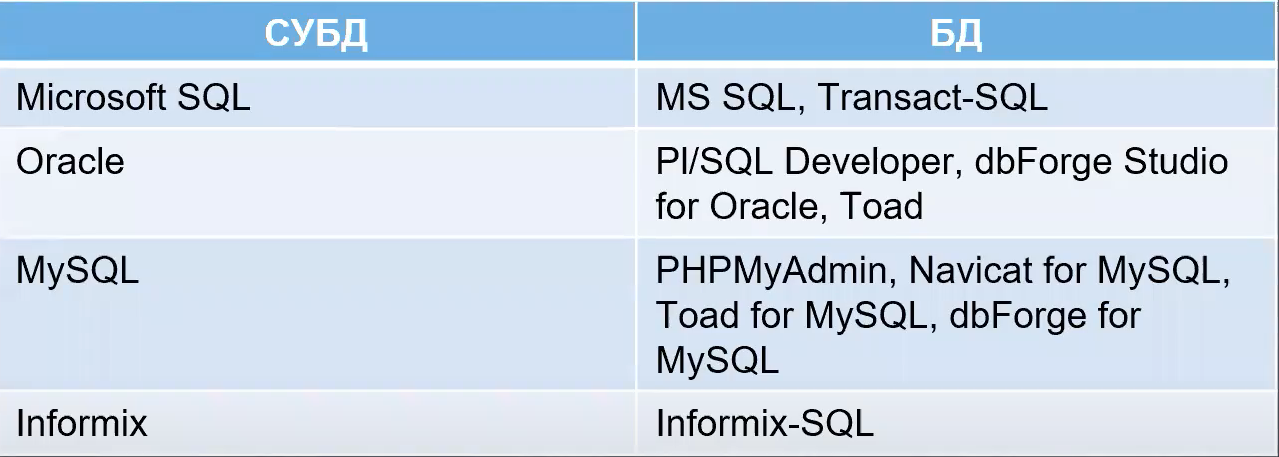
- Клиент для доступа к данным (php Admin, DBeaver, MySQL Workbench…)

- API нужен что бы подключить БД к приложению. Что бы “подружить” backend язык приложения (Java, C#, python,…) с БД, которая работает на SQL языке

- Сервер базы данных

- СУБД (MySQL, PostgreSQL,…)

*Диалекты языка SQL в СУБД*:



Возможности СУБД

- Пользовательский интерфейс

- Управление конкурентным доступом (если 2 пользователя хотят купить один и тот же товар, когда его в наличие 1 штука) – такие дилемы решает БД

- Копирование и восстановление данных

- Целостность данных

- Оптимизация запросов

- Физическая и логическая независимость данных

**ТРАНЗАКЦИИ**

Транзакции – это набор запросов, которые должны выполниться все. Если выполнился только 1 запрос, то он отменяется. (нужно при оплате)

COMMIT – фиксирует транзакцию. ROLLBACK – отменяет транзакцию

START TRANSACTION;

UPDATE account SET balance = balance – 15000

WHERE account\_number = 1234567;

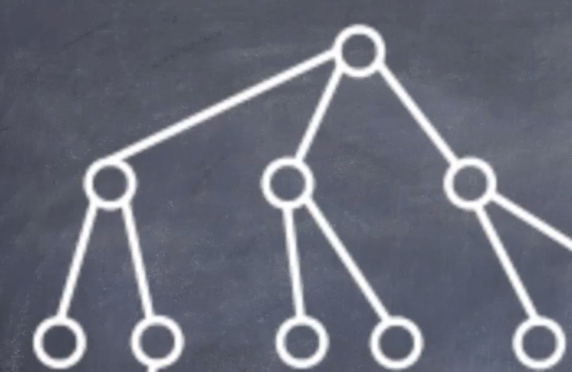
UPDATE account SET balance = balance + 15000

WHERE account\_number = 9876543;

COMMIT;

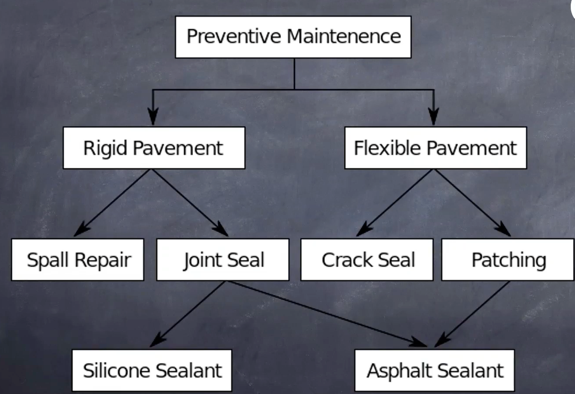
**Модели данных:**

* *Иерархическая*



У 1 предка много потомков (связь 1 ко многим) – например Проводник Windows

* *Сетевая*

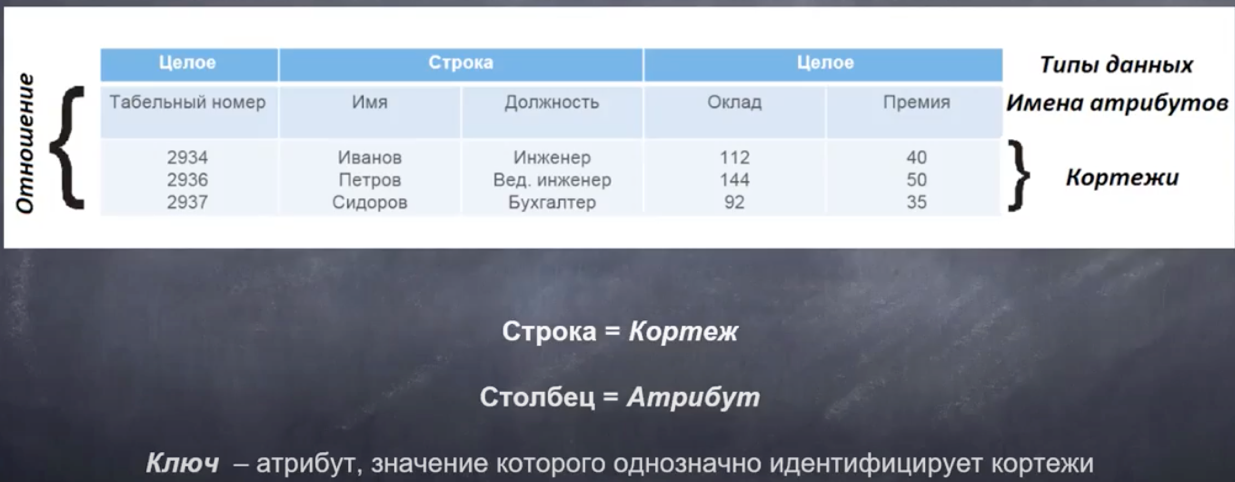


У потомков можем быть множество предков

* *Реляционная*



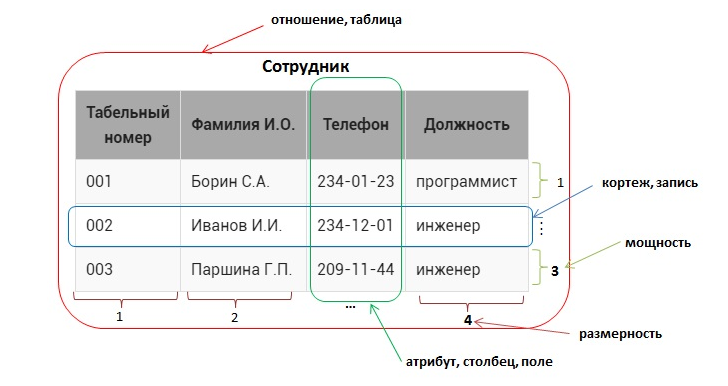
1. Каждое значение атрибута должно быть атомарно (разделяем ФИО отдельно на Имя Фамилия Отчество)
2. В столбцах данные должны быть 1 типа (например в столбце Имя должны быть только строки)
3. Записи в таблице должны быть уникальны (для этого используется id)
4. Столбцы должны иметь уникальное имя



* *Big Data, NoSQL*



**ОТНОШЕНИЯ**

****

На примере таблицы **Сотрудник** рассмотрим **терминологию реляционных баз данных:**

* ***отношение***  – это структура данных целиком, набор записей (в обычном понимании – таблица) , в  примере –это **Сотрудник**;
* ***кортеж*** – это каждая строка , содержащая данные (более распространенный термин – запись ), например, <001, Борин С.А, 234-01-23, программист>, все кортежи в отношении должны быть различны;
* ***мощность*** – число кортежей в таблице (проще говоря, число записей), в данном случае 3, мощность отношения может быть любой (от 0 до бесконечности), порядок следования кортежей - неважен;
* ***атрибут*** (**поле**)– это столбец в таблице (более распространенный термин – поле ), в примере – **Табельный номер, Фамилия И.О., Телефон, Должность**)
* ***размерность*** – это число атрибутов в таблице, в данном случае – 4;
* размерность отношения должна быть больше 0, порядок следования атрибутов существенен;
* ***домен атрибута*** – это допустимые значения (неповторяющиеся), которые можно занести в поле , например для атрибута **Должность** домен – {инженер, программист}.