**База данных** - (БД) оргонизационная структура, преднозначенная для хранения, изменения и извлечения взаимосвязанной информации, преимущественно больших обьёмов.

**Реляционная бд -** это тип БД, в которой данные организованы в виде связанных таблиц, также называемых отношениями. «Отношение» переводится на английский язык как relation, отсюда и появилось название «реляционная база данных»

**Система управления базами данных** - (СУБД), комплекс програмых средств, необходимых для создания структуры новой базы, её наполнения, редоктирования содержимого и отображения информации. СУБД основывается на модели базы данных - это специальные структуры предназначенные для работы с данными.

Основными элементами реляционной базы данных являются:

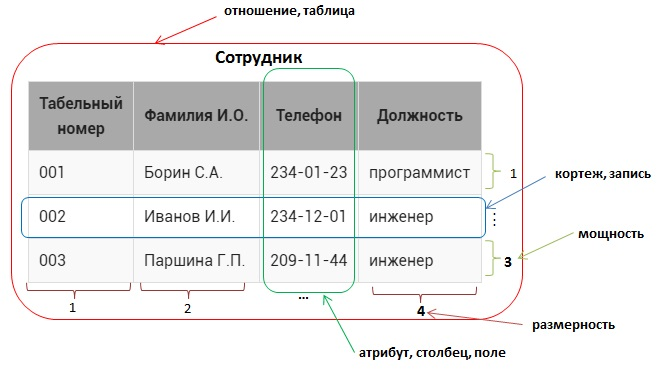
**таблицы**, (отношения)

**столбцы (**атрибуты)

**строки** (кортежи)

а в математических понятиях это: отношения, атрибуты и кортежи. Также часто, строки называют записями, столбцы называют колонками, а пересечение записи и колонки называют ячейкой.

Содержание строки и названия столбцов должны быть уникальны в пределах одной базы данных.



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** – 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}.

**Типы данных**

Важно понимать, что можно создавать базы для любых типов данных: текстов, дат, времени, событий, цифр. В зависимости от типа информации реляционные базы данных делят на типы. Каждый тип данных (атрибут) имеет свое обозначение:

-INTEGER- данные из целых чисел;

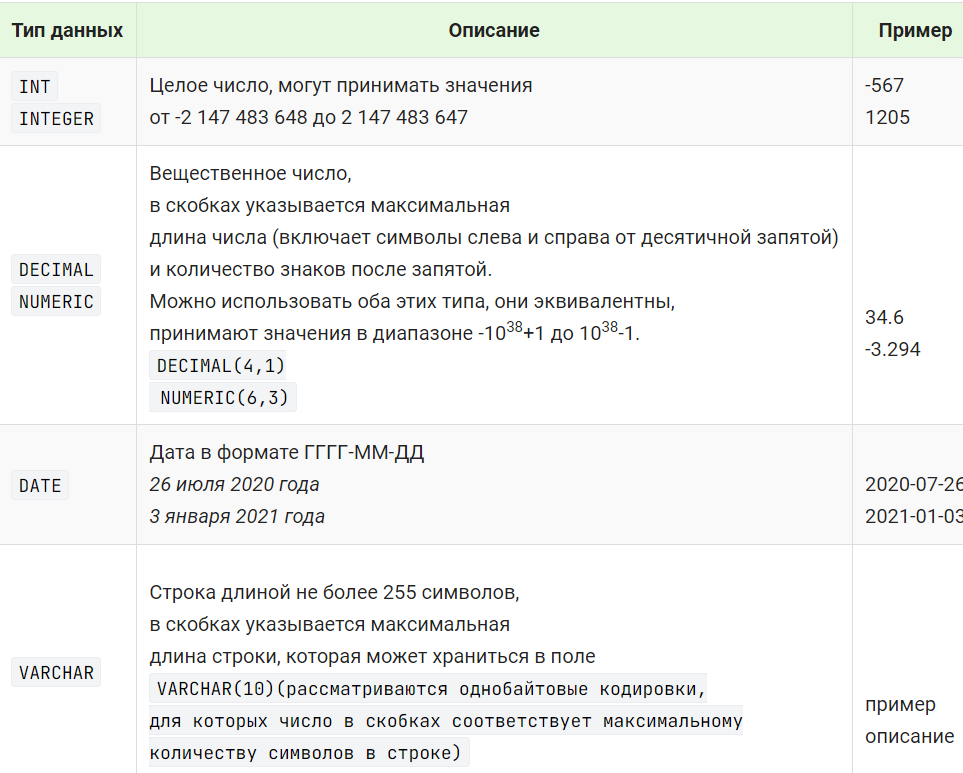
-FLOAT – данные из дробных чисел, так называемые данные с плавающей точкой;

-CHAR, VARCHAR – текстовые типы данных (символьные);

-LOGICAL – логический тип данных (да/нет);

-DATE/TIME – временные данные.

Это основные типы данных, которых на самом деле гораздо больше. Причем, каждый язык программирования имеет свой набор системных атрибутов (типов данных).



Рекомендации по выбору типов данных для полей таблицы.

Выбирайте минимальный тип данных исходя из максимального значения поля. Например, если максимальный текст, который может быть записан в поле, имеет длину 25 символов, значит нужно использовать тип VARCHAR(25).

Для описания ключевого поля используйте описание INT PRIMARY KEY AUTO\_INCREMENT. Это значит, что в поле будут заноситься различные целые числа, при этом они будут автоматически генерироваться (каждая следующая строка будет иметь значение ключа на 1 больше предыдущего).

Определим тип данных для каждого поля таблицы **book**:

**book\_id** - ключевой столбец, целое число, которое должно генерироваться автоматически - INT PRIMARY KEY AUTO\_INCREMENT;

**title** - строка текста, ее длина выбирается в зависимости от данных, которые предполагается хранить в поле, предположим, что название книги не превышает 50 символов - VARCHAR(50);

**author** - строка текста - VARCHAR(30);

**price** - для описание денежного значения используется числовой тип данных с двумя знаками после запятой - DECIMAL(8,2);

**amount** - целое число - INT.

**Первичный ключ**

Каждая строка (запись) в БД должна быть уникальна, т.е. таблица должна иметь ключ по которому можно идентифицировать каждую строку (другими словами каждая строка должна иметь уникальный идентификатор, его называют **первичным ключом**). Обозначается первичный ключ, как primary key.

Primary key не позволяет создавать одинаковых записей (строк) в таблице;

Primary key обеспечивают логическую связь между таблицами одной базы данных (для реляционных БД).

**Ключ внешний** (Foreign key)

Foreign key, кратко FK. Обеспечивает однозначную логическую связь, между таблицами одной БД.

Пример:

Две таблицы данных: Люди и Номера телефонов.

Таблица: Люди

primary key Имя

1 Зайцев

2 Белкин

3 Волков

Таблица: Номера телефонов

primary key телефон foreign key

1 12345 1

2 54321 1

3 678910 2

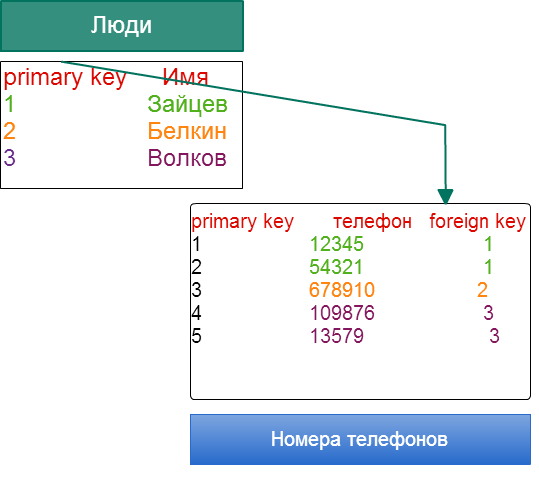
4 109876 3

5 13579 3

В таблице Номера телефонов PK уникален. FK этой таблицы является PK таблицы Люди. Связь между номерами телефонов и людьми обеспечивает FK таблицы телефонов. То есть:

У Зайцева два телефона; 12345, 54321

У Волкова два телефона; 109876, 13579

У Белкина один телефон. 678910

Другими словами primary key из таблицы ‘люди’ стал foreign key в таблице ‘номера телефонов’