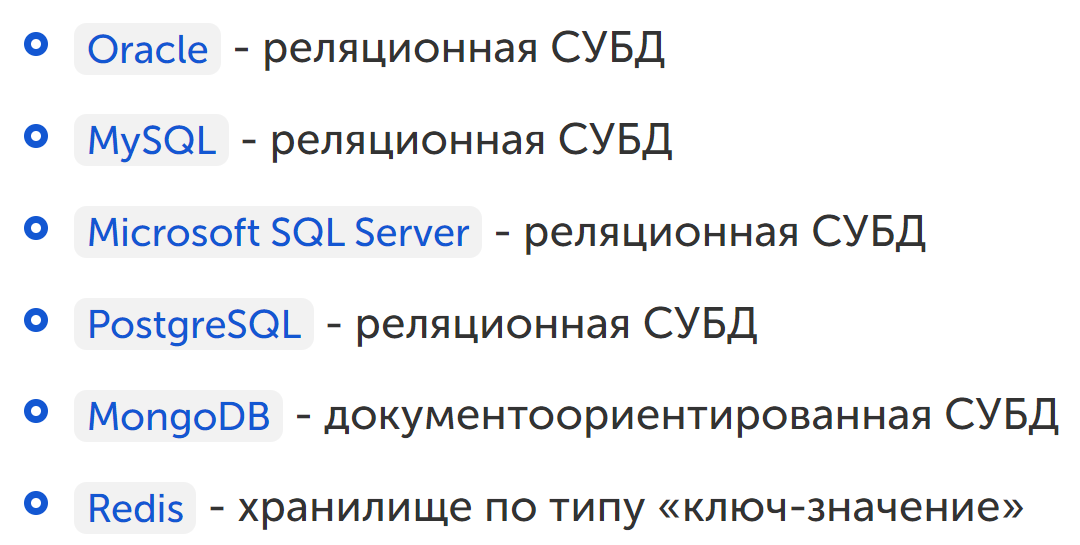
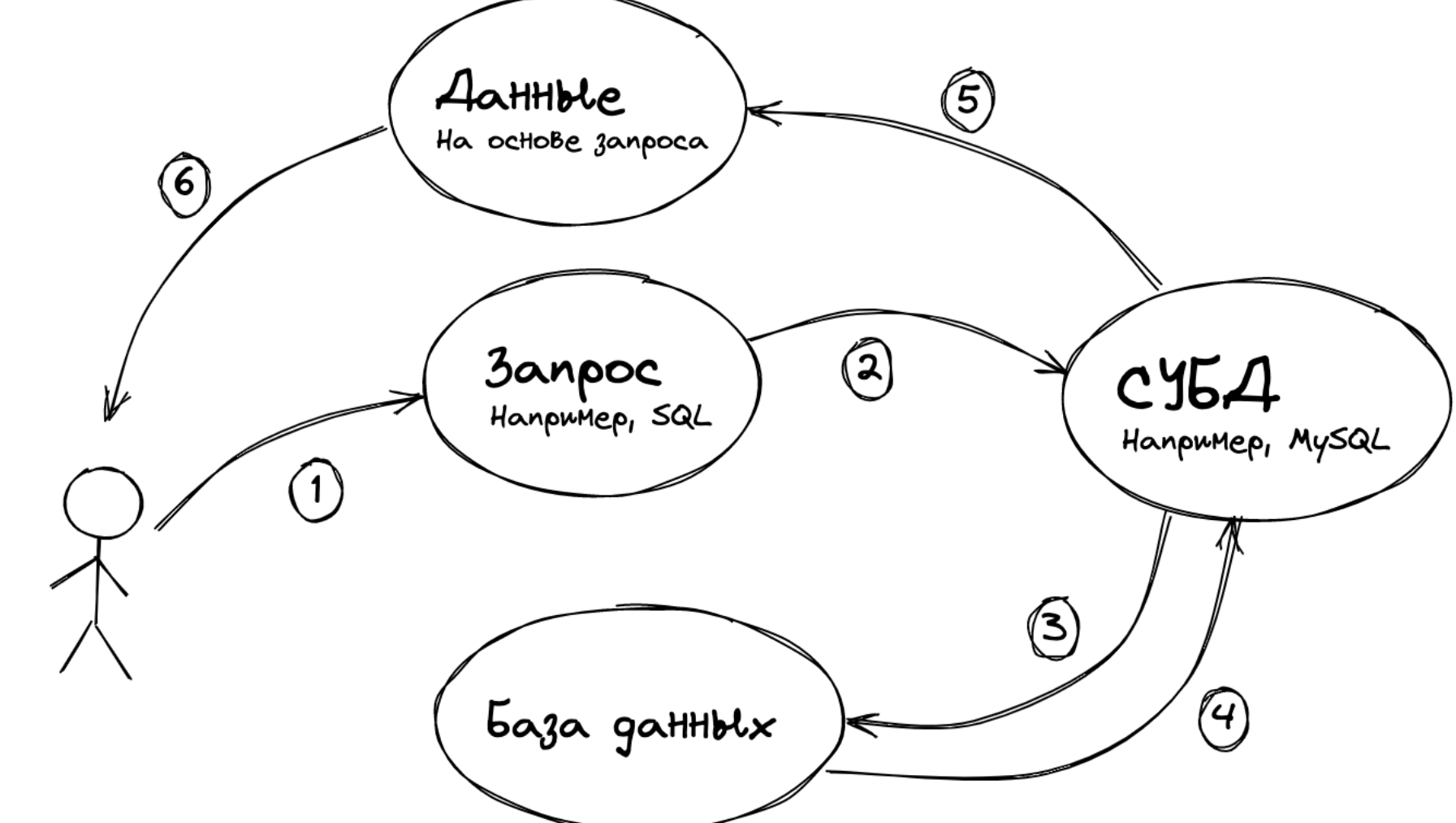


**База данных** — это набор данных, хранящиеся в структурированном виде.

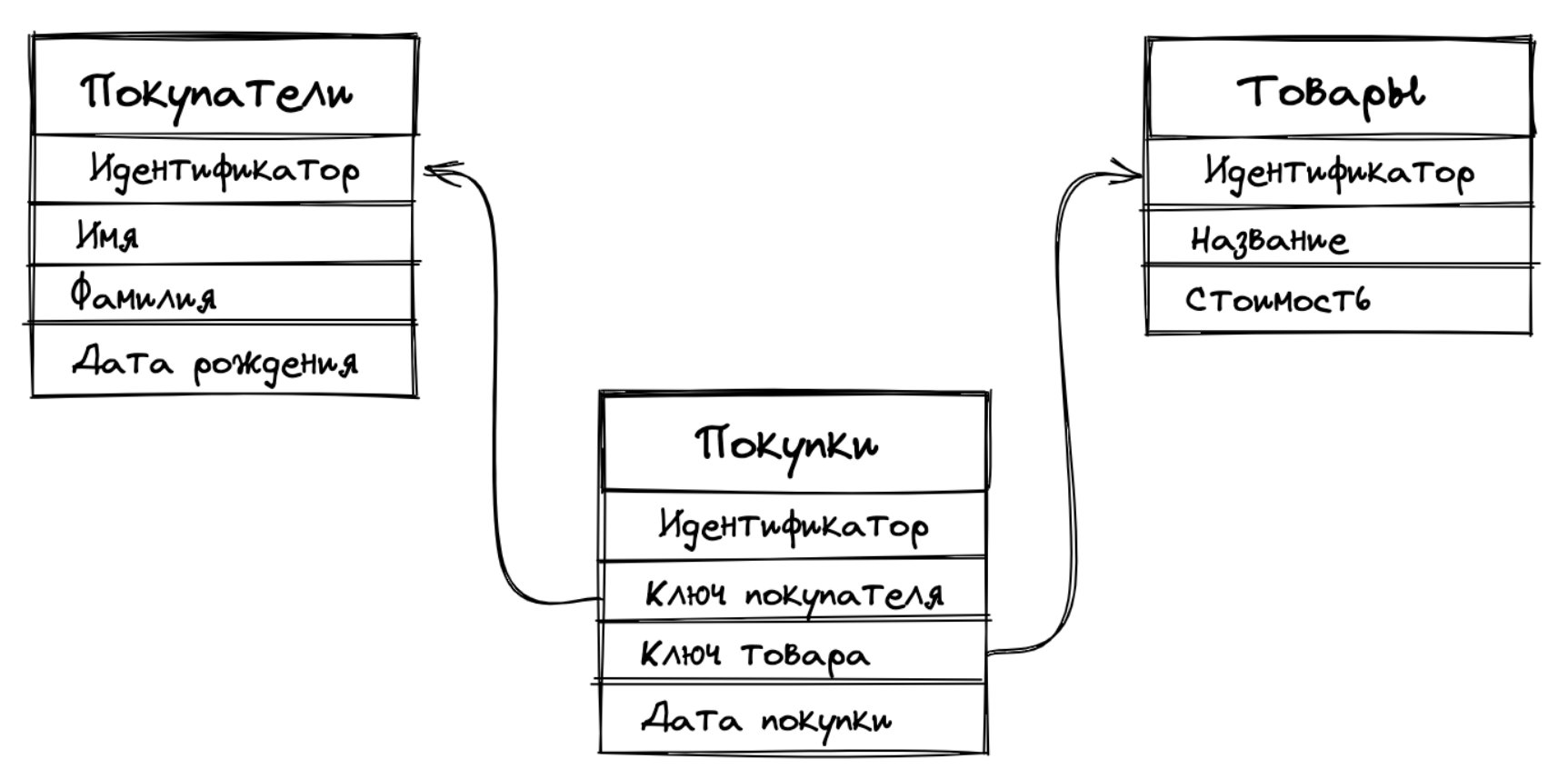
**Система управления базами данных** — это совокупность языковых и программных средств, которая осуществляет доступ к данным, позволяет их создавать, менять и удалять, обеспечивает безопасность данных и т.д.





Данные в реляционных структурах организованы в виде набора таблиц, называемых отношениями, состоящих из столбцов и строк.

Каждая строка в таблице может быть помечена уникальным идентификатором, называемым **первичным ключом**, а строки из нескольких таблиц могут быть связаны с помощью внешних ключей.

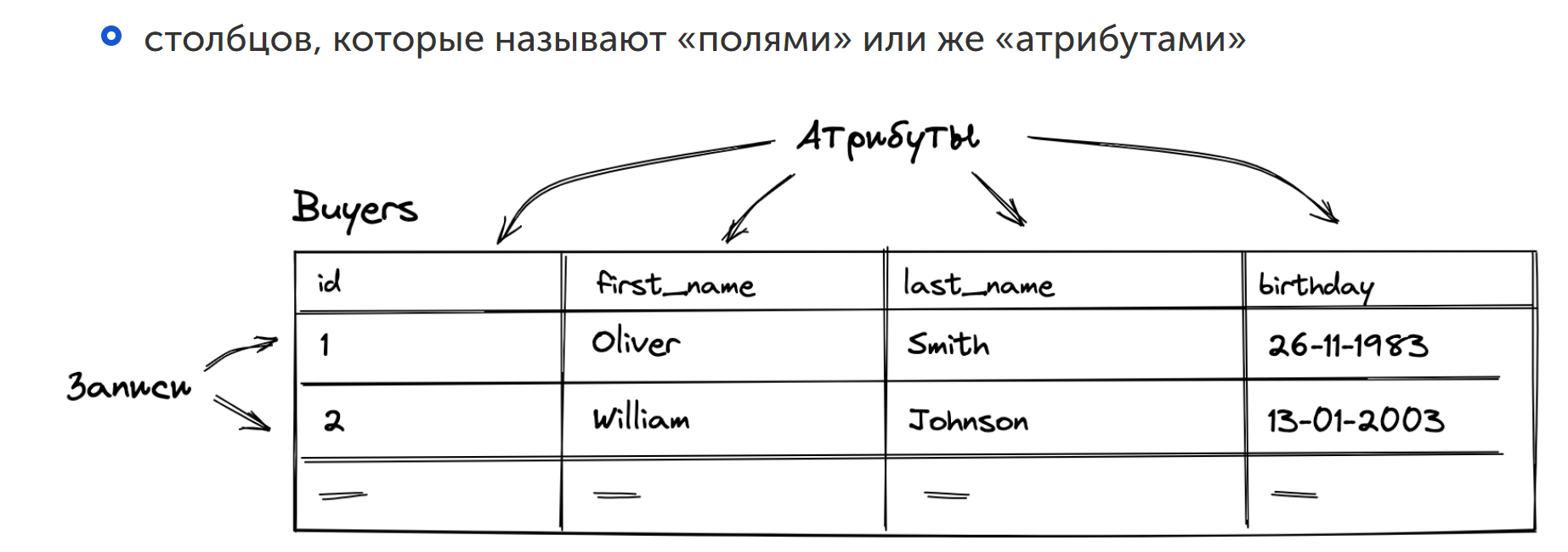


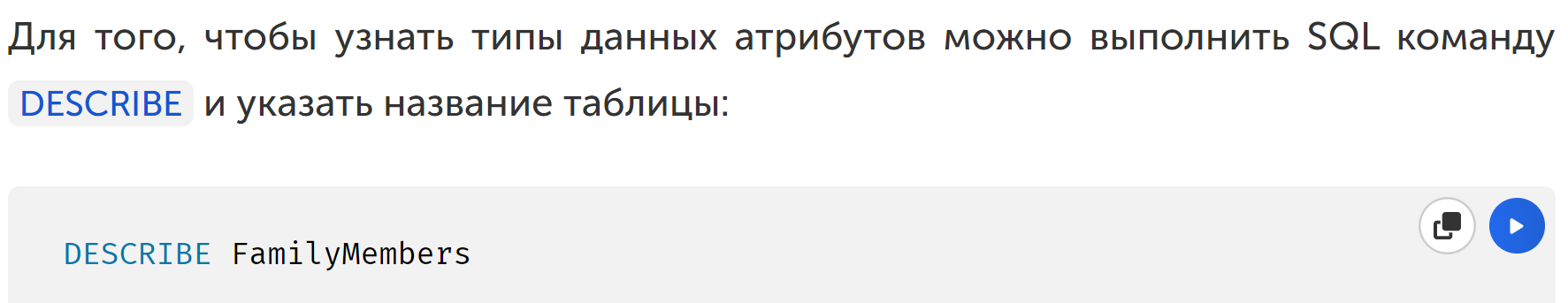
В таблице может не быть ни одной строчки, но обязательно должен быть хотя бы один столбец.

**Key-value базы данных** – это тип баз данных, которые хранят данные как совокупность пар «ключ-значение», в которых ключ служит уникальным идентификатором.

**Документно-ориентированные базы данных** – это тип баз данных, направленный на хранение и запрос данных в виде документов, подобном JSON.

В отличие от других баз данных, документо-ориентированные оперируют «документами», сгруппированными по коллекциям. Документ представляет собой набор атрибутов (ключ и соответствующее ему значение). Значения могут быть, как и простыми типами данных (строки, числа или даты), так и более сложными, такими как вложенные объекты, массивы и ссылки на другие документы.

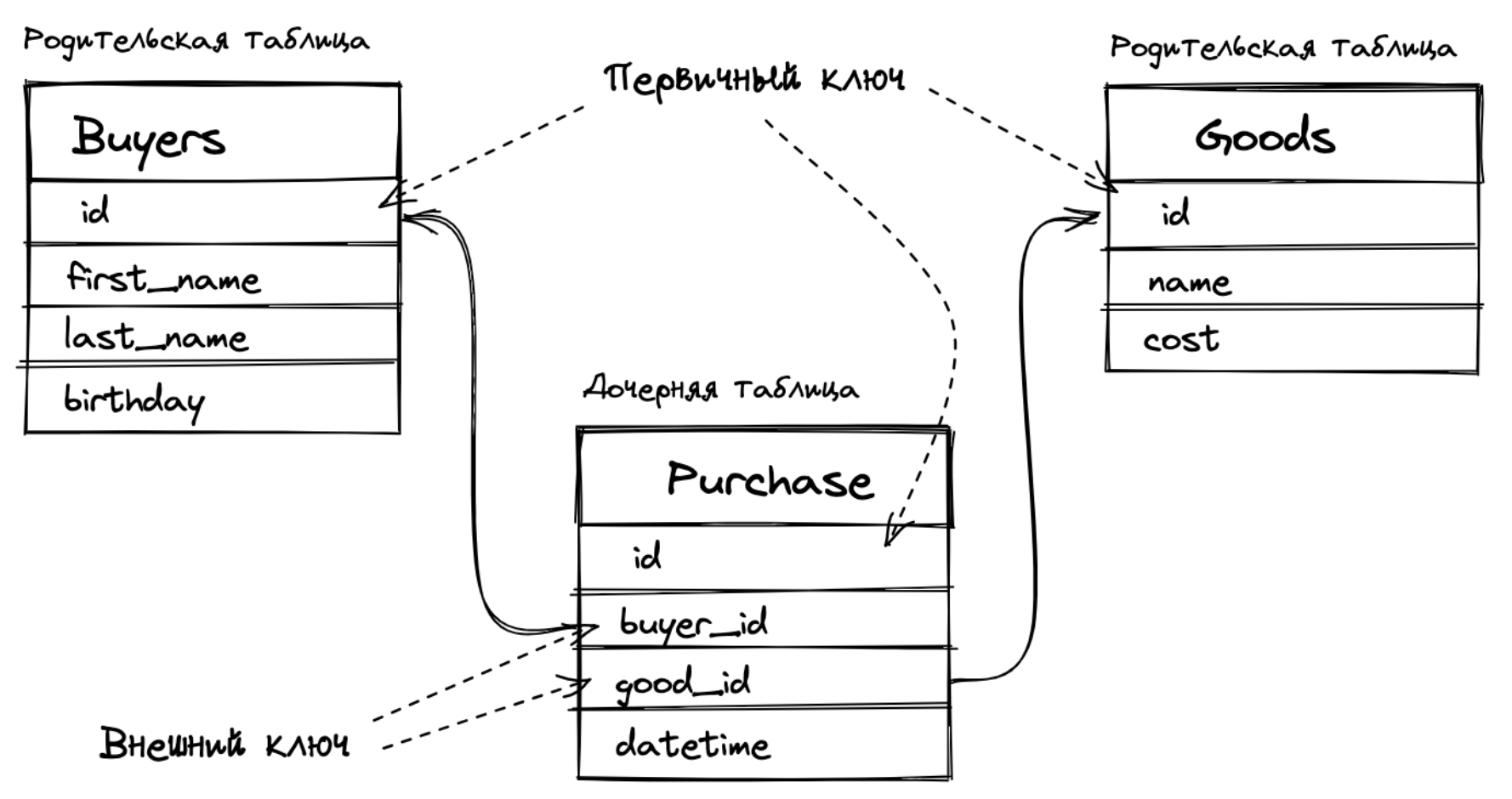




**Ключевое поле** (первичный ключ) – это поле (или набор полей), значение которого однозначно определяет запись в таблице.

**Внешний ключ** – это поле (или набор полей) в одной таблице, которое ссылается на первичный ключ в другой таблице.

Таблица с внешним ключом называется **дочерней таблицей**, а таблица с первичным ключом называется **ссылочной** или **родительской таблицей**.



Если наличие первичного ключа – это обязательное требование для каждой таблицы в реляционной базе данных, то правило внешнего ключа – нет.

Если внешний ключ не определен, то всё также будет работать, но СУБД не будет проверять, что, например, при создании записи в таблице Purchase в полях buyer\_id и good\_id лежат значения, которые определены в соответствующих таблицах в поле id.

**SQL** — язык структурированных запросов (SQL, Structured Query Language), который используется в качестве эффективного способа сохранения данных, поиска их частей, обновления, извлечения и удаления из базы данных.

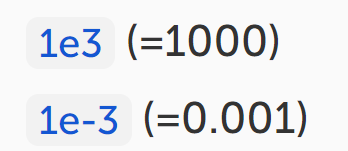
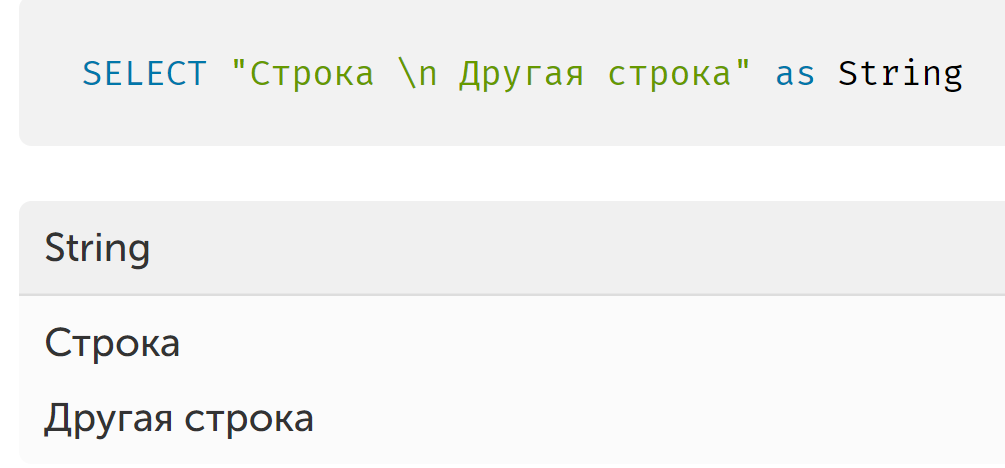
Диалекты SQL – это расширения языка SQL для конкретных СУБД.

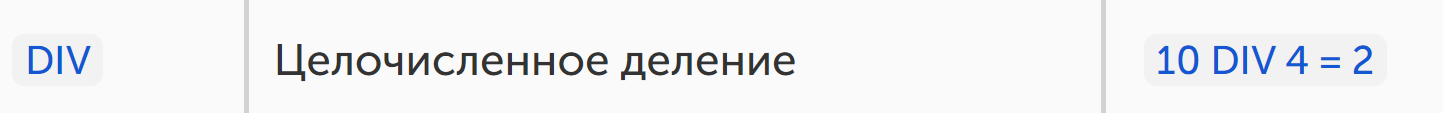
Вот некоторые из них:

* T-SQL – диалект Microsoft SQL Server
* PL/SQL – диалект Oracle Database
* PL/pgSQL – диалект PostgreSQL

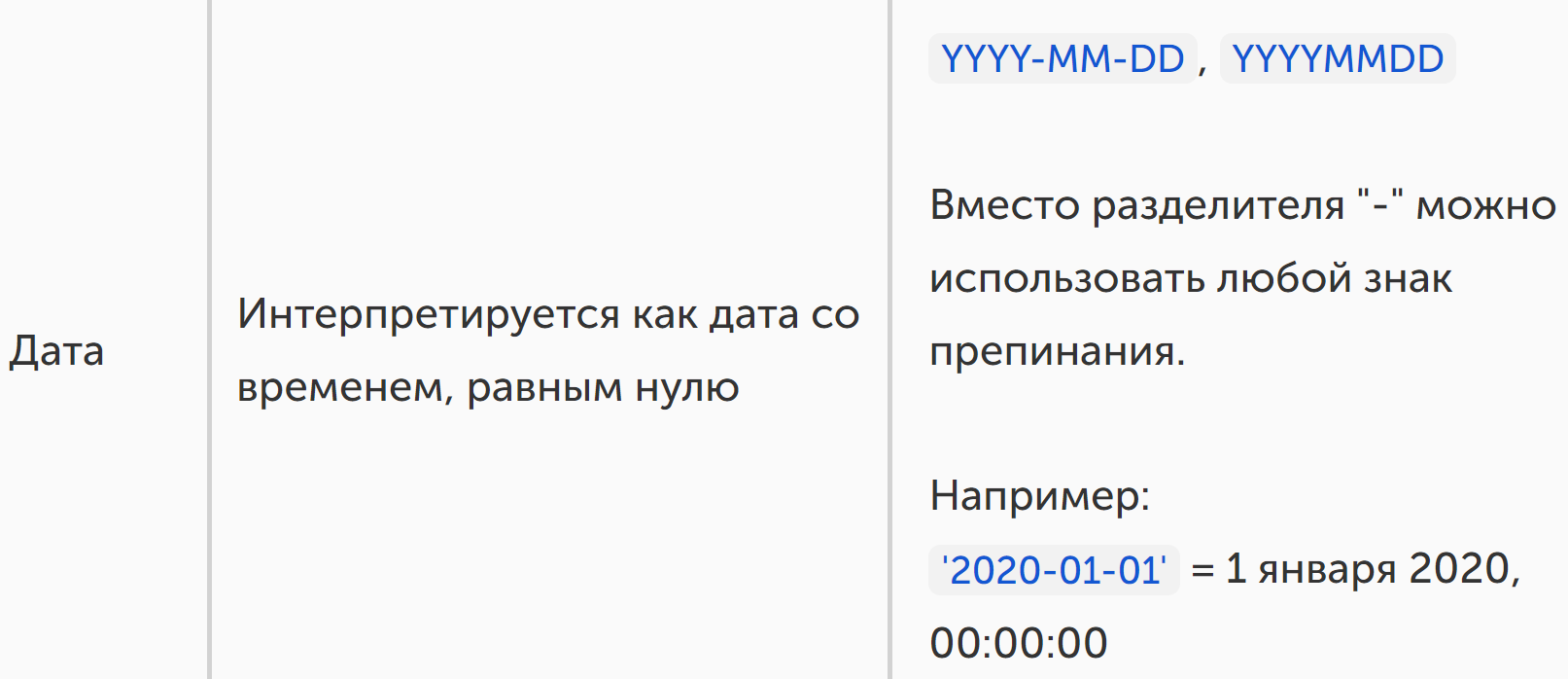
В случае, если мы хотим вывести какие-то столбцы таблицы, но чтобы в итоговой выборке они были названы иначе, мы можем использовать псевдонимы (их также называют алиасами).

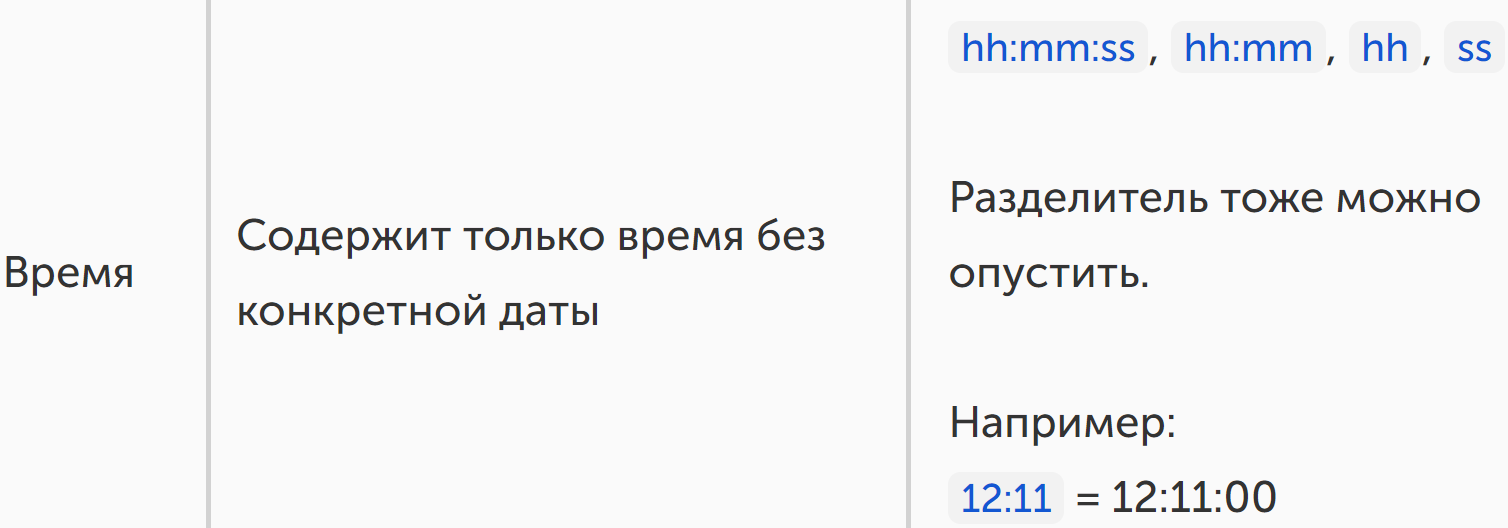
Их синтаксис достаточно простой, мы должны использовать оператор AS.













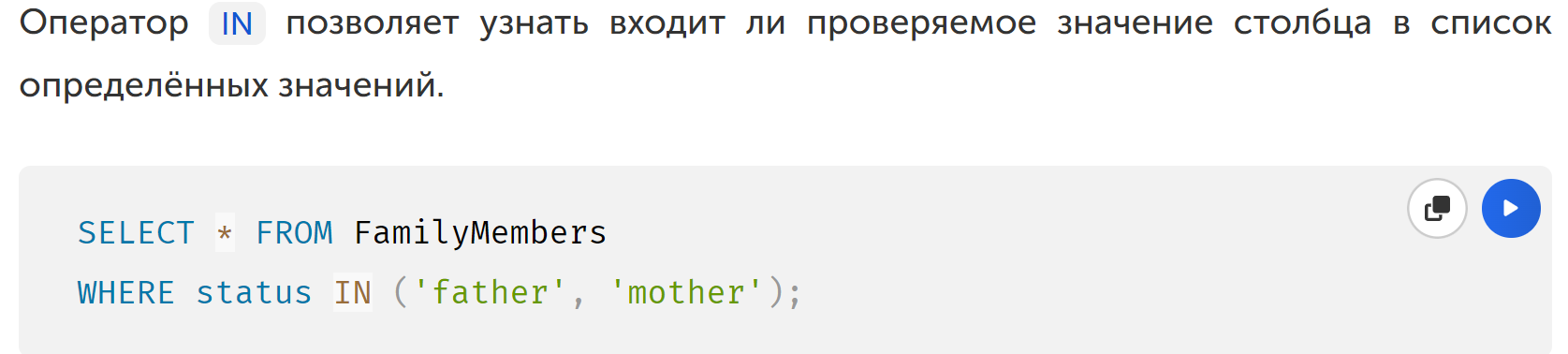
Чтобы при выборке избежать дублирования, есть оператор DISTINCT.

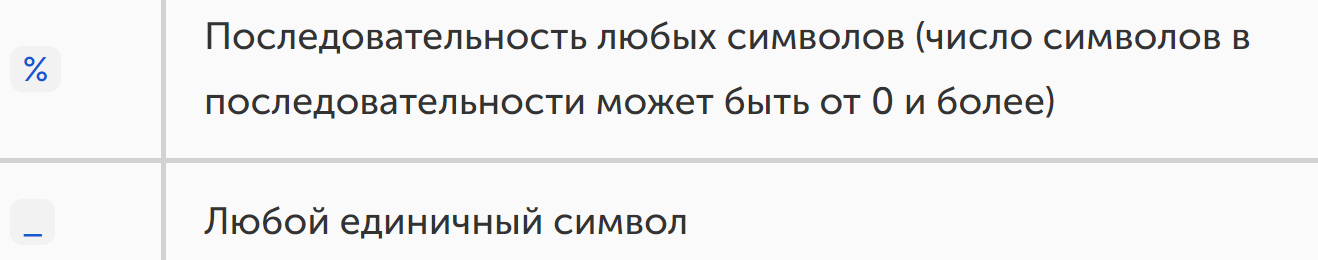


При использовании оператора DISTINCT для двух и более колонок будут удаляться записи, которые имеют одинаковые значения по всем полям.

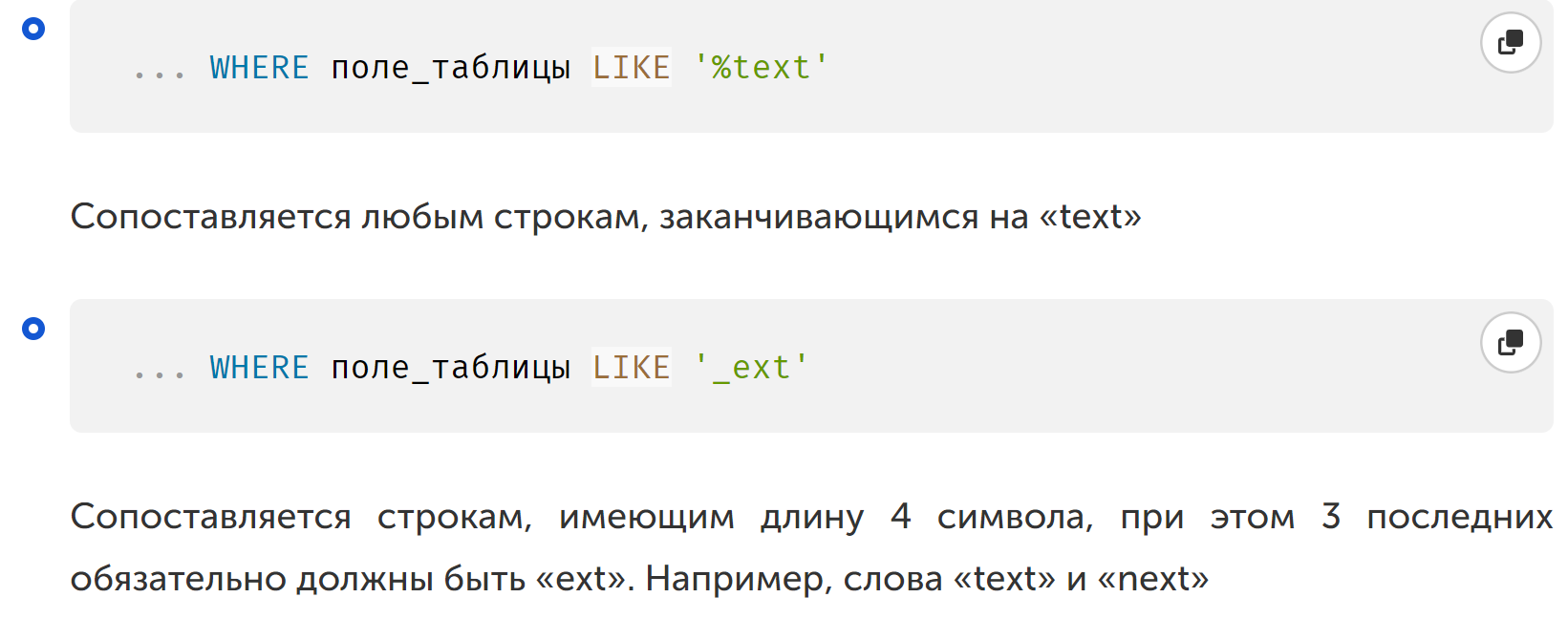
Ситуация, когда требуется сделать выборку по определённому условию, встречается очень часто. Для этого в операторе SELECT существует оператор WHERE, после которого следуют условия для ограничения строк.

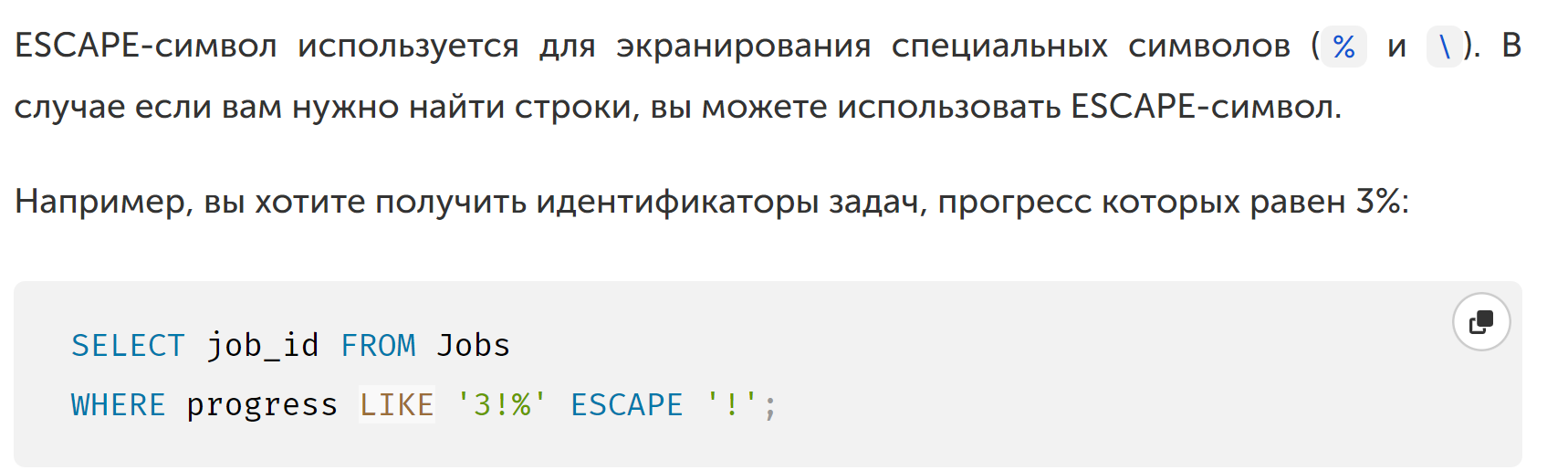
Оператор IS NULL позволяет узнать равно ли проверяемое значение NULL, т.е. пустое ли значение.





Оператор LIKE используется при условных запросах, когда мы хотим узнать соответствует ли строка определённому шаблону.

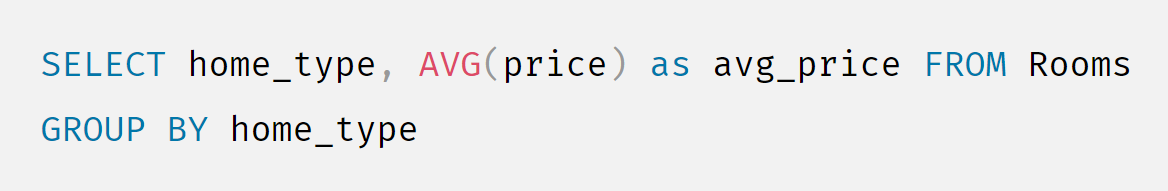




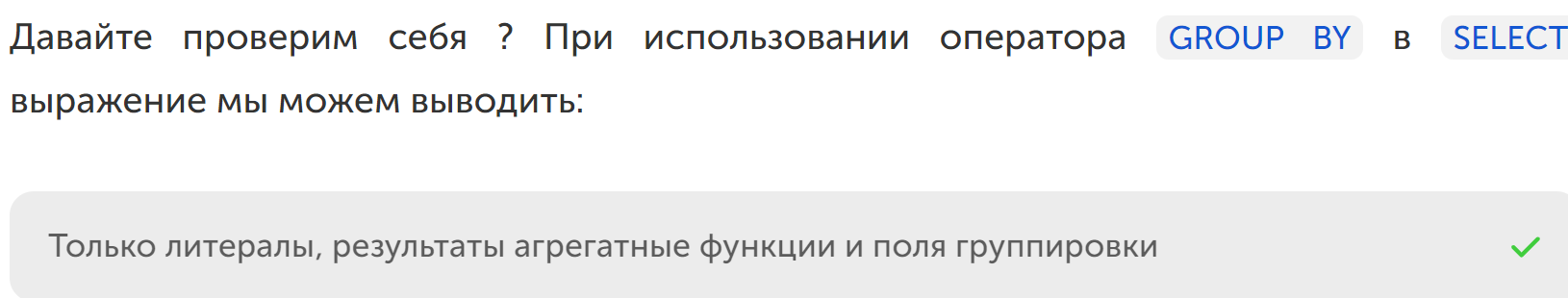
Для упорядочивания записей используется конструкция ORDER BY.

Для сортировки результатов по двум или более столбцам их следует указывать через запятую.

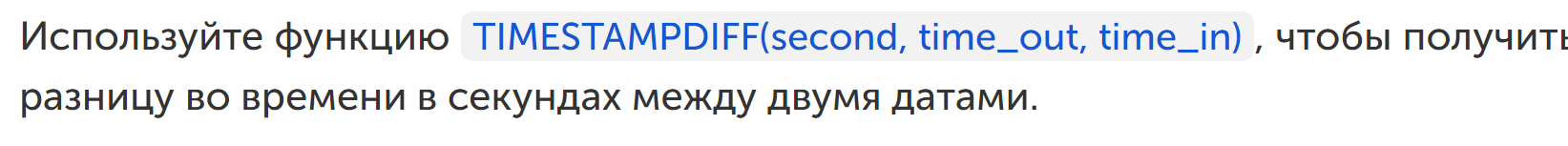
Следует иметь в виду, что для GROUP BY все значения NULL трактуются как равные, т.е. при группировке по полю, содержащему NULL-значения, все такие строки попадут в одну группу.



Так выполненный запрос сначала разбивает все записи из таблицы Rooms на 3 группы, опираясь на поле home\_type. Далее, для каждой группы суммирует все значения, взятые из поля price у каждой записи, входящей в текущую группу, и затем полученный результат делится на количество записей в данной группе.



**Агрегатная функция** – это функция, которая выполняет вычисление на наборе значений и возвращает одиночное значение.



Для фильтрации групп мы должны использовать оператор HAVING:

