**Теория**

**Что такое СУБД?**

Допустим, есть большая база данных, которой пользуются многие сотрудники: кто-то ищет информацию, а кто-то изменяет или даже удаляет её. Чтобы правильно обрабатывать все эти запросы, нужно специальное программное обеспечение, и именно такое ПО получило название системы управления базами данных (СУБД).

**Какие типы СУБД в соответствии с моделями данных вы знаете?**

Этот вопрос по SQL предполагает не просто назвать, но и дать краткое описание каждому типу.Существует [несколько типов СУБД](https://tproger.ru/translations/sql-nosql-database-models):

1. Реляционные, которые поддерживают установку связей между таблицами с помощью первичных и внешних ключей. Пример — MySQL.Flat File — базы данных с двумерными файлами, в которых содержатся записи одного типа и отсутствует связь с другими файлами, как в реляционных. Пример — Excel.Иерархические подразумевают наличие записей, связанных друг с другом по принципу отношений один-к-одному или один-ко-многим. А вот для отношений многие-ко-многим следует использовать реляционную модель. Пример — Adabas.Сетевые похожи на иерархические, но в этом случае «ребёнок» может иметь несколько «родителей» и наоборот. Примеры — IDS и IDMS.Объектно-ориентированные СУБД работают с базами данных, которые состоят из объектов, используемых в ООП. Объекты группируются в классы и называются экземплярами, а классы в свою очередь взаимодействуют через методы. Пример — Versant.Объектно-реляционные обладают преимуществами реляционной и объектно-ориентированной моделей. Пример — IBM Db2.Многомерная модель является разновидностью реляционной и использует многомерные структуры. Часто представляется в виде кубов данных. Пример — Oracle Essbase.Гибридные состоят из двух и более типов баз данных. Используются в том случае, если одного типа недостаточно для обработки всех запросов. Пример — Altibase HDВ.

**Что такое первичный ключ?**

Первичный ключ или PRIMARY KEY предназначен для однозначной идентификации каждой записи в таблице и является строго уникальным (UNIQUE): две записи таблицы не могут иметь одинаковые значения первичного ключа. Нулевые значения (NULL) в PRIMARY KEY не допускаются. Если в качестве PRIMARY KEY используется несколько полей, их называют составным ключом.Пример:

CREATE TABLE USERS ( id INT NOT NULL, name VARCHAR (20) NOT NULL, PRIMARY KEY (id));

Здесь в качестве первичного ключа используется поле id.

**Когда используется PRIMARY KEY?**

PRIMARY KEY — это первичный ключ, который используется в качестве основного ключа и может быть использован для связи с дочерней таблицей, содержащей внешний ключ.

**А что такое внешний ключ?**

Внешний ключ или FOREIGN KEY также является атрибутом ограничения и обеспечивает связь двух таблиц. По сути, это поле или несколько полей, которые ссылаются на PRIMARY KEY в родительской таблице.Пример использования:

CREATE TABLE order ( order\_id INT NOT NULL, user\_id INT, PRIMARY KEY (order\_id), FOREIGN KEY (user\_id) REFERENCES users(id));

В данном случае внешний ключ, привязанный к полю user\_id в таблице order, ссылается на первичный ключ id в таблице users, и именно по этим полям происходит связывание двух таблиц.

**Какие ещё ограничения вы знаете, как они работают и указываются?**

SQL-ограничения (constraints) указываются при создании или изменении таблицы. Это правила для ограничения типа данных, которые могут храниться в таблице. Действие с данными не будет выполнено, если нарушаются установленные ограничения.

* UNIQUE — гарантирует уникальность значений в столбце;
* NOT NULL — значение не может быть NULL;
* INDEX — создаёт индексы в таблице для быстрого поиска/запросов;
* CHECK — значения столбца должны соответствовать заданным условиям;
* DEFAULT — предоставляет столбцу значения по умолчанию&

**Для чего используется ключевое слово ORDER BY?**

Для сортировки данных в порядке возрастания (ASC) или убывания (DESC).Пример использования:

SELECT \* FROM user ORDER BY name DESC;

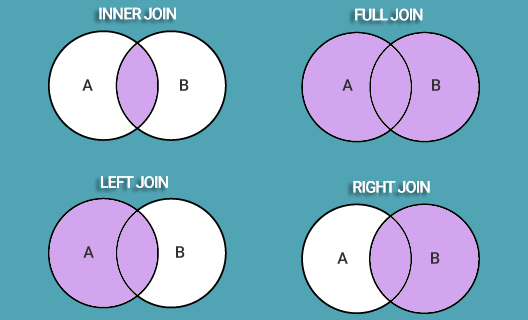
Выбираются пользователи, которые будут отсортированы по имени в порядке убывания. Дополните ответ на этот вопрос по SQL тем, что без указания DESC данные были бы отсортированы по умолчанию — в порядке возрастания:

SELECT \* FROM user ORDER BY name;

**Назовите четыре основных типа соединения в SQL**

Чтобы объединить две таблицы в одну, следует использовать оператор JOIN. Соединение таблиц может быть внутренним (INNER) или внешним (OUTER), причём внешнее соединение может быть левым (LEFT), правым (RIGHT) или полным (FULL).

* INNER JOIN — получение записей с одинаковыми значениями в обеих таблицах, т.е. получение пересечения таблиц.
* FULL OUTER JOIN — объединяет записи из обеих таблиц (если условие объединения равно true) и дополняет их всеми записями из обеих таблиц, которые не имеют совпадений. Для записей, которые не имеют совпадений из другой таблицы, недостающее поле будет иметь значение NULL.
* LEFT JOIN — возвращает все записи, удовлетворяющие условию объединения, плюс все оставшиеся записи из внешней (левой) таблицы, которые не удовлетворяют условию объединения.
* RIGHT JOIN — работает точно так же, как и левое объединение, только в качестве внешней таблицы будет использоваться правая.



Рассмотрим пример соединения SQL таблиц с использованием INNER JOIN. Следующий запрос выбирает все заказы с информацией о клиенте:

SELECT Orders.OrderID, Customers.CustomerNameFROM OrdersINNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

**А что такое Self JOIN?**

Такой вопрос тоже может прозвучать на собеседовании по SQL. Это выражение используется для того, чтобы таблица объединилась сама с собой, словно это две разные таблицы. Чтобы такое реализовать, одна из таких «таблиц» временно переименовывается. Например, следующий SQL-запрос объединяет клиентов из одного города:

SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.CityFROM Customers A, Customers BWHERE A.CustomerID <> B.CustomerIDAND A.City = B.CityORDER BY A.City;

**Для чего нужен оператор UNION?**

Он используется для объединения полученных данных из двух или более запросов, которые должны иметь одинаковое количество столбцов с одинаковыми типами данных и расположенных в том же порядке. Пример использования:

SELECT column(s) FROM first\_tableUNIONSELECT column(s) FROM second\_table;

**Как работают подстановочные знаки?**

Это специальные символы, которые нужны для замены каких-либо знаков в запросе. Они используются вместе с оператором LIKE, с помощью которого можно отфильтровать запрашиваемые данные.

**Какими бывают подстановочные знаки?**

* % — заменить ноль или более символов;
* \_ — заменить один символ.

Примеры:

SELECT \* FROM user WHERE name LIKE '%test%';

Данный запрос позволяет найти данные всех пользователей, имена которых содержат в себе «test».

SELECT \* FROM user WHERE name LIKE 't\_est';

А в этом случае имена искомых пользователей начинаются на «t», после содержат какой-либо символ и «est» в конце.

**Что делают псевдонимы Aliases?**

SQL-псевдонимы нужны для того, чтобы дать временное имя таблице или столбцу. Это нужно, когда в запросе есть таблицы или столбцы с неоднозначными именами. В этом случае для удобства в составлении запроса используются псевдонимы. SQL-псевдоним существует только на время запроса.Пример:

SELECT very\_long\_column\_name AS alias\_nameFROM table;

**Для чего нужен оператор INSERT INTO SELECT?**

Данный оператор копирует данные из одной таблицы и вставляет их в другую, при этом типы данных в обеих таблицах должны соответствовать.Пример использования:

INSERT INTO second\_tableSELECT \* FROM first\_tableWHERE condition;

**Что такое нормализация и денормализация?**

Нормализация отношений в SQL призвана [организовать информацию в базе данных](https://tproger.ru/translations/db-scheme-design-guide) таким образом, чтобы она не занимала много места и с ней было удобно работать. Это удаление избыточных данных, устранение дублей, идентификация наборов связанных данных через PRIMARY KEY, etc.Соответственно, денормализация является обратным процессом, который вносит в нормализованную таблицу избыточные данные.Подробнее о пяти нормальных формах и форме Бойса-Кодда можно узнать [из данного видеокурса](https://www.youtube.com/watch?v=nsXV4PGMmrk&list=PLtULzFUr0kI5YKdBKKAzt7AFjjoMnGX_a).

**Объясните разницу между командами DELETE и TRUNCATE”**

Команда DELETE — это DML-операция, которая удаляет записи из таблицы, соответствующие заданному условию:

DELETE FROM table\_name WHERE condition;

При этом создаются логи удаления, то есть операцию можно отменить.А вот команда TRUNCATE — это DDL-операция, которая полностью пересоздаёт таблицу, и отменить такое удаление невозможно:

TRUNCATE TABLE table\_name;

**Чем VARCHAR отличается от NVARCHAR?**

Главное отличие в том, что VARCHAR хранит значения в формате ASCII, где символ занимает один байт, а NVARCHAR хранит значения в формате Unicode, где символ «весит» 2 байта. Тип VARCHAR следует использовать, если вы уверены, что в значениях не будет Unicode-символов. Например, VARCHAR можно применить к адресам электронной почты, состоящих из ASCII-символов.

**Практика**

**Как выбрать записи с нечётными Id?**

Один из самых распространённых вопросов на собеседовании. SQL запрос для выбора записей с нечётными id должен выглядеть следующим образом:

SELECT \* FROM sample WHERE id % 2 != 0;

Если остаток от деления id на 2 равен нулю, перед нами чётное значение, и наоборот.

**Как найти дубли в поле email?**

SELECT email, COUNT(email)FROM customersGROUP BY emailHAVING COUNT(email) > 1;

Функция COUNT() возвращает количество строк из поля email. Оператор HAVING работает почти так же, как и WHERE, вот только применяется не для всех столбцов, а для набора, созданного оператором GROUP BY.

**При выборке из таблицы прибавьте к дате 1 день**

SELECT DATE\_ADD(date, 1 DAY) as new\_date FROM table;

Функция DATE\_ADD() прибавляет к дате заданный промежуток времени. Синтаксис выглядит следующим образом:

SELECT DATE\_ADD(дата, INTERVAL что\_прибавить) FROM имя\_таблицы WHERE условие;

**Выберите только уникальные имена**

SELECT DISTINCT name FROM users;

SELECT DISTINCT возвращает разные значения, даже если в выбранном столбце есть дубли.

**Найдите в таблице среднюю зарплату работников**

SELECT AVG(salary) FROM workers;

Функция AVG() применяется только к числовым типам данных и возвращает среднее значение по столбцу.

**А теперь получите список сотрудников с зарплатой выше средней**

SELECT \* FROM workersWHERE salary > (SELECT AVG (salary) FROM workers);

**Даны таблицы workers и departments. Найдите все департаменты без единого сотрудника**

SELECT department\_nameFROM workers wRIGHT JOIN departments d ON (w.department\_id = d.department\_id)WHERE first\_name IS NULL;

**Замените в таблице зарплату работника на 1000, если она равна 900, и на 1500 в остальных случаях**

Замена значений — одна из наиболее часто встречаемых задач по SQL на собеседованиях. Решить её несложно:

UPDATE table SET salary =CASEWHEN salary = 900 THEN 1000ELSE 1500END;

Оператор UPDATE используется для изменения существующих записей. Но ответы на подобные вопросы с собеседований по SQL должны быть более развёрнутыми. Уточните,что после UPDATE следует указать, какие записи должны быть обновлены. В противном случае обновятся все записи в таблице.В нашем примере условие задаётся через оператор CASE: если текущая зарплата равна 900, изменяем её на 1000, в остальных случаях — на 1500.

**При выборке из таблицы пользователей создайте поле, которое будет включать в себя и имена, и зарплату**

SELECT CONCAT(name, salary) AS new\_field FROM users;

Функция CONCAT() используется для конкатенации (объединения) строк, неявно преобразуя при этом любые типы данных в строки.

**Переименуйте таблицу**

ALTER TABLE first\_table RENAME second\_table;

С помощью оператора ALTER TABLE можно добавлять, удалять, изменять столбцы, а также изменять название таблицы.