## **Что такое «SQL»?**

SQL, Structured query language («язык структурированных запросов») — формальный непроцедурный язык программирования, применяемый для создания, модификации и управления данными в произвольной реляционной базе данных, управляемой соответствующей системой управления базами данных (СУБД).

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Какие существуют операторы SQL?**

****операторы определения данных (Data Definition Language, DDL)****:

* CREATE создает объект БД (базу, таблицу, представление, пользователя и т. д.),
* ALTER изменяет объект, - создает столбец
* DROP удаляет объект;

****операторы манипуляции данными (Data Manipulation Language, DML)****:

* SELECT выбирает данные, удовлетворяющие заданным условиям,
* INSERT добавляет новые данные, - создает строку
* UPDATE изменяет существующие данные, -
* DELETE удаляет данные;

****операторы определения доступа к данным (Data Control Language, DCL)****:

* GRANT предоставляет пользователю (группе) разрешения на определенные операции с объектом,
* REVOKE отзывает ранее выданные разрешения,
* DENY задает запрет, имеющий приоритет над разрешением;

****операторы управления транзакциями (Transaction Control Language, TCL)****:

* COMMIT применяет транзакцию,
* ROLLBACK откатывает все изменения, сделанные в контексте текущей транзакции,
* SAVEPOINT разбивает транзакцию на более мелкие.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Что означает NULL в SQL?**

NULL - специальное значение (псевдозначение), которое может быть записано в поле таблицы базы данных. NULL соответствует понятию «пустое поле», то есть «поле, не содержащее никакого значения».

NULL означает отсутствие, неизвестность информации. Значение NULL не является значением в полном смысле слова: по определению оно означает отсутствие значения и не принадлежит ни одному типу данных. Поэтому NULL не равно ни логическому значению FALSE, ни пустой строке, ни 0. При сравнении NULL с любым значением будет получен результат NULL, а не FALSE и не 0. Более того, NULL не равно NULL! Т.е. значения не будут идентичны.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Что такое «временная таблица»? Для чего она используется?**

****Временная таблица**** - это объект базы данных, который хранится и управляется системой базы данных на временной основе. Они могут быть локальными или глобальными. Используется для сохранения результатов вызова хранимой процедуры, уменьшение числа строк при соединениях, агрегирование данных из различных источников или как замена курсоров и параметризованных представлений.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Что такое «представление» (view) и для чего оно применяется?**

****Представление****, View - виртуальная таблица, представляющая данные одной или более таблиц альтернативным образом.

В действительности представление – всего лишь результат выполнения оператора SELECT, который хранится в структуре памяти, напоминающей SQL таблицу. Они работают в запросах и операторах DML точно также как и основные таблицы, но не содержат никаких собственных данных. Представления значительно расширяют возможности управления данными. Это способ дать публичный доступ к некоторой (но не всей) информации в таблице.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Каков общий синтаксис оператора SELECT?**

SELECT - оператор DML SQL, возвращающий набор данных (выборку) из базы данных, удовлетворяющих заданному условию. Имеет следующую структуру:

SELECT

[DISTINCT | DISTINCTROW | ALL]

select\_expression,...

FROM table\_references

[WHERE where\_definition]

[GROUP BY {unsigned\_integer | column | formula}]

[HAVING where\_definition]

[ORDER BY {unsigned\_integer | column | formula} [ASC | DESC], ...]

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Что такое JOIN?**

****JOIN**** - оператор языка SQL, который является реализацией операции соединения реляционной алгебры. Предназначен для обеспечения выборки данных из двух таблиц и включения этих данных в один результирующий набор.

Особенностями операции соединения являются следующее:

* в схему таблицы-результата входят столбцы обеих исходных таблиц (таблиц-операндов), то есть схема результата является «сцеплением» схем операндов;
* каждая строка таблицы-результата является «сцеплением» строки из одной таблицы-операнда со строкой второй таблицы-операнда;
* при необходимости соединения не двух, а нескольких таблиц, операция соединения применяется несколько раз (последовательно).

SELECT

field\_name [,... n]FROM

Table1

{INNER | {LEFT | RIGHT | FULL} OUTER | CROSS } JOIN

Table2

{ON <condition> | USING (field\_name [,... n])}

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Какие существуют типы JOIN?**

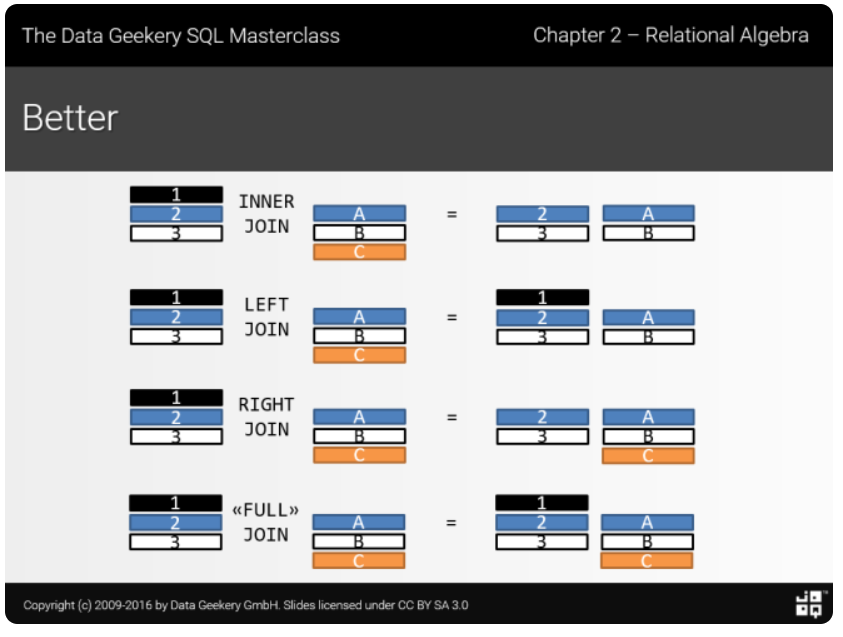
****(INNER) JOIN**** Результатом объединения таблиц являются записи, общие для левой и правой таблиц. Порядок таблиц для оператора не важен, поскольку оператор является симметричным.

****LEFT (OUTER) JOIN**** Производит выбор всех записей первой таблицы и соответствующих им записей второй таблицы. Если записи во второй таблице не найдены, то вместо них подставляется пустой результат (NULL). Порядок таблиц для оператора важен, поскольку оператор не является симметричным.

****RIGHT (OUTER) JOIN**** LEFT JOIN с операндами, расставленными в обратном порядке. Порядок таблиц для оператора важен, поскольку оператор не является симметричным.

****FULL (OUTER) JOIN**** Результатом объединения таблиц являются все записи, которые присутствуют в таблицах. Порядок таблиц для оператора не важен, поскольку оператор является симметричным.

****CROSS JOIN (декартово произведение)**** При выборе каждая строка одной таблицы объединяется с каждой строкой второй таблицы, давая тем самым все возможные сочетания строк двух таблиц. Порядок таблиц для оператора не важен, поскольку оператор является симметричным.

[](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

**в SQL подзапросы — это?**

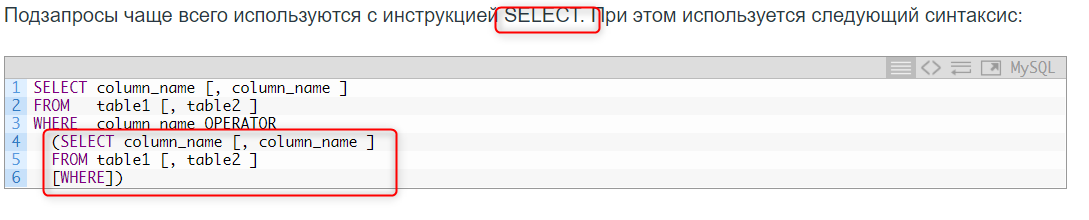
это запрос внутри другого запроса SQL, который вложен в условие WHERE.

Подзапрос используется для возврата данных, которые будут использоваться в основном запросе, в качестве условия для дальнейшей фильтрации данных, подлежащих извлечению.

Подзапросы могут использоваться с инструкциями SELECT, INSERT, UPDATE и DELETE вместе с операторами типа =, <,>,> =, <=, IN, BETWEEN и т. д.

Существует несколько правил, которые применяются к подзапросам:

Подзапросы должны быть заключены в круглые скобки.



**Что лучше использовать JOIN или подзапросы?**

Обычно лучше использовать JOIN, поскольку в большинстве случаев он более понятен и лучше оптимизируется СУБД (но 100% этого гарантировать нельзя). Так же JOIN имеет заметное преимущество над подзапросами в случае, когда список выбора SELECT содержит столбцы более чем из одной таблицы.

# Подзапросы лучше использовать в случаях, когда нужно вычислять агрегатные значения (SUM, MIN, MAX, AVG, COUNT) и использовать их для сравнений во внешних запросах.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Для чего используется оператор HAVING?**

HAVING используется для фильтрации результата GROUP BY по заданным логическим условиям.

## **В чем различие между операторами HAVING и WHERE?**

HAVING используется как WHERE, но в другой части SQL-выражения и, соответственно, на другой стадии формирования ответа.

## **Для чего используется оператор ORDER BY?**

****ORDER BY**** упорядочивает вывод запроса согласно значениям в том или ином количестве выбранных столбцов. Многочисленные столбцы упорядочиваются один внутри другого. Возможно определять возрастание ASC или убывание DESC для каждого столбца. По умолчанию установлено - возрастание.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Для чего используется оператор GROUP BY?**

GROUP BY используется для агрегации записей результата по заданным признакам-атрибутам.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Как GROUP BY обрабатывает значение NULL?**

При использовании GROUP BY все значения NULL считаются равными.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **В чем разница между операторами GROUP BY и DISTINCT?**

DISTINCT указывает, что для вычислений используются только уникальные значения столбца. NULL считается как отдельное значение. GROUP BY создает отдельную группу для всех возможных значений (включая значение NULL).

Если нужно удалить только дубликаты лучше использовать DISTINCT, GROUP BY лучше использовать для определения групп записей, к которым могут применяться агрегатные функции.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Перечислите основные агрегатные функции.**

****Агрегатных функции**** - функции, которые берут группы значений и сводят их к одиночному значению.

SQL предоставляет несколько агрегатных функций:

COUNT - производит подсчет записей, удовлетворяющих условию запроса; SUM - вычисляет арифметическую сумму всех значений колонки; AVG - вычисляет среднее арифметическое всех значений; MAX - определяет наибольшее из всех выбранных значений; MIN - определяет наименьшее из всех выбранных значений.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **В чем разница между COUNT(\*) и COUNT({column})?**

COUNT (\*) подсчитывает количество записей в таблице, не игнорируя значение NULL, поскольку эта функция оперирует записями, а не столбцами.

COUNT ({column}) подсчитывает количество значений в {column}. При подсчете количества значений столбца эта форма функции COUNT не принимает во внимание значение NULL.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Что делает оператор EXISTS?**

EXISTS берет подзапрос, как аргумент, и оценивает его как TRUE, если подзапрос возвращает какие-либо записи и FALSE, если нет.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Для чего используются операторы IN, BETWEEN, LIKE?**

IN - определяет набор значений.

SELECT \* FROM Persons WHERE name IN ('Ivan','Petr','Pavel');

BETWEEN определяет диапазон значений. В отличие от IN, BETWEEN чувствителен к порядку, и первое значение в предложении должно быть первым по алфавитному или числовому порядку.

SELECT \* FROM Persons WHERE age BETWEEN 20 AND 25;

LIKE применим только к полям типа CHAR или VARCHAR, с которыми он используется чтобы находить подстроки. В качестве условия используются символы шаблонизации (wildkards) - специальные символы, которые могут соответствовать чему-нибудь:

\_ замещает любой одиночный символ. Например, 'b\_t' будет соответствовать словам 'bat' или 'bit', но не будет соответствовать 'brat'.

% замещает последовательность любого числа символов. Например '%p%t' будет соответствовать словам 'put', 'posit', или 'opt', но не 'spite'.

SELECT \* FROM UNIVERSITY WHERE NAME LIKE '%o';

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Для чего применяется ключевое слово UNION?**

В языке SQL ключевое слово UNION применяется для объединения результатов двух SQL-запросов в единую таблицу, состоящую из схожих записей. Оба запроса должны возвращать одинаковое число столбцов и совместимые типы данных в соответствующих столбцах. Необходимо отметить, что UNION сам по себе не гарантирует порядок записей. Записи из второго запроса могут оказаться в начале, в конце или вообще перемешаться с записями из первого запроса. В случаях, когда требуется определенный порядок, необходимо использовать ORDER BY.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Какие ограничения на целостность данных существуют в SQL?**

PRIMARY KEY - набор полей (1 или более), значения которых образуют уникальную комбинацию и используются для однозначной идентификации записи в таблице. Для таблицы может быть создано только одно такое ограничение. Данное ограничение используется для обеспечения целостности сущности, которая описана таблицей.

CHECK используется для ограничения множества значений, которые могут быть помещены в данный столбец. Это ограничение используется для обеспечения целостности предметной области, которую описывают таблицы в базе.

UNIQUE обеспечивает отсутствие дубликатов в столбце или наборе столбцов.

FOREIGN KEY защищает от действий, которые могут нарушить связи между таблицами. FOREIGN KEY в одной таблице указывает на PRIMARY KEY в другой. Поэтому данное ограничение нацелено на то, чтобы не было записей FOREIGN KEY, которым не отвечают записи PRIMARY KEY.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Какие отличия между ограничениями PRIMARY и UNIQUE?**

По умолчанию ограничение PRIMARY создает кластерный индекс на столбце, а UNIQUE - некластерный. Другим отличием является то, что PRIMARY не разрешает NULL записей, в то время как UNIQUE разрешает одну (а в некоторых СУБД несколько) NULL запись.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Может ли значение в столбце, на который наложено ограничение FOREIGN KEY, равняться NULL?**

Может, если на данный столбец не наложено ограничение NOT NULL.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Как создать индекс?**

Индекс можно создать либо с помощью выражения CREATE INDEX:

CREATE INDEX index\_name ON table\_name (column\_name)

либо указав ограничение целостности в виде уникального UNIQUE или первичного PRIMARY ключа в операторе создания таблицы CREATE TABLE.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Что делает оператор MERGE?**

MERGE позволяет осуществить слияние данных одной таблицы с данными другой таблицы. При слиянии таблиц проверяется условие, и если оно истинно, то выполняется UPDATE, а если нет - INSERT. При этом изменять поля таблицы в секции UPDATE, по которым идет связывание двух таблиц, нельзя.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **В чем отличие между операторами DELETE и TRUNCATE?**

DELETE - оператор DML, удаляет записи из таблицы, которые удовлетворяют критерию WHERE при этом задействуются триггеры, ограничения и т.д.

TRUNCATE - DDL оператор (удаляет таблицу и создает ее заново. Причем если на эту таблицу есть ссылки FOREGIN KEY или таблица используется в репликации, то пересоздать такую таблицу не получится).

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Что такое «хранимая процедура»?**

****Хранимая процедура**** — объект базы данных, представляющий собой набор SQL-инструкций, который хранится на сервере. Хранимые процедуры очень похожи на обыкновенные процедуры языков высокого уровня, у них могут быть входные и выходные параметры и локальные переменные, в них могут производиться числовые вычисления и операции над символьными данными, результаты которых могут присваиваться переменным и параметрам. В хранимых процедурах могут выполняться стандартные операции с базами данных (как DDL, так и DML). Кроме того, в хранимых процедурах возможны циклы и ветвления, то есть в них могут использоваться инструкции управления процессом исполнения.

Хранимые процедуры позволяют повысить производительность, расширяют возможности программирования и поддерживают функции безопасности данных. В большинстве СУБД при первом запуске хранимой процедуры она компилируется (выполняется синтаксический анализ и генерируется план доступа к данным) и в дальнейшем её обработка осуществляется быстрее.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Что такое «триггер»?**

****Триггер (trigger)**** — это хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено действием по модификации данных: добавлением, удалением или изменением данных в заданной таблице реляционной базы данных. Триггеры применяются для обеспечения целостности данных и реализации сложной бизнес-логики. Триггер запускается сервером автоматически и все производимые им модификации данных рассматриваются как выполняемые в транзакции, в которой выполнено действие, вызвавшее срабатывание триггера. Соответственно, в случае обнаружения ошибки или нарушения целостности данных может произойти откат этой транзакции.

Момент запуска триггера определяется с помощью ключевых слов BEFORE (триггер запускается до выполнения связанного с ним события) или AFTER (после события). В случае, если триггер вызывается до события, он может внести изменения в модифицируемую событием запись. Кроме того, триггеры могут быть привязаны не к таблице, а к представлению (VIEW). В этом случае с их помощью реализуется механизм «обновляемого представления». В этом случае ключевые слова BEFORE и AFTER влияют лишь на последовательность вызова триггеров, так как собственно событие (удаление, вставка или обновление) не происходит.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Что такое «курсор»?**

****Курсор**** — это объект базы данных, который позволяет приложениям работать с записями «по-одной», а не сразу с множеством, как это делается в обычных SQL командах.

Порядок работы с курсором такой:

* Определить курсор (DECLARE)
* Открыть курсор (OPEN)
* Получить запись из курсора (FETCH)
* Обработать запись...
* Закрыть курсор (CLOSE)
* Удалить ссылку курсора (DEALLOCATE). Когда удаляется последняя ссылка курсора, SQL освобождает структуры данных, составляющие курсор.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Опишите разницу типов данных DATETIME и TIMESTAMP.**

DATETIME предназначен для хранения целого числа: YYYYMMDDHHMMSS. И это время не зависит от временной зоны, настроенной на сервере. Размер: 8 байт

TIMESTAMP хранит значение равное количеству секунд, прошедших с полуночи 1 января 1970 года по усреднённому времени Гринвича. При получении из базы отображается с учётом часового пояса. Размер: 4 байта

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Для каких числовых типов недопустимо использовать операции сложения/вычитания?**

В качестве операндов операций сложения и вычитания нельзя использовать числовой тип BIT.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/sql.md" \l "sql)

## **Напишите запрос...**

CREATE TABLE table (

id BIGINT(20) NOT NULL AUTO\_INCREMENT,

created TIMESTAMP NOT NULL DEFAULT 0,

PRIMARY KEY (id)

);

Требуется написать запрос, который вернет максимальное значение id и значение created для этого id:

SELECT id, created FROM table where id = (SELECT MAX(id) FROM table);

CREATE TABLE track\_downloads (

download\_id BIGINT(20) NOT NULL AUTO\_INCREMENT,

track\_id INT NOT NULL,

user\_id BIGINT(20) NOT NULL,

download\_time TIMESTAMP NOT NULL DEFAULT 0,

PRIMARY KEY (download\_id)

);

Напишите SQL-запрос, возвращающий все пары (download\_count, user\_count), удовлетворяющие следующему условию: user\_count — общее ненулевое число пользователей, сделавших ровно download\_count скачиваний 19 ноября 2010 года:

SELECT DISTINCT download\_count, COUNT(\*) AS user\_count FROM (

SELECT COUNT(\*) AS download\_count

FROM track\_downloads WHERE download\_time="2010-11-19"

GROUP BY user\_id) AS download\_countGROUP BY download\_count;