SQL

SQL - что такое SQL и для чего используют команды SQL?

SQL — это структурированный язык запросов, созданный для того, чтобы получать из базы данных необходимую информацию. Если описать схему работы SQL простыми словами, то специалист формирует запрос и направляет его в базу. Та в свою очередь обрабатывает эту информацию, «понимает», что именно нужно специалисту, и отправляет ответ.

Данные хранятся в виде таблиц, они структурированы и разложены по строкам и столбцам, чтобы ими легче было оперировать. Такой способ хранения информации называют реляционными базами данных (от англ. *relation* — «отношения»). Название указывает на то, что объекты в такой базе связаны определенными отношениями.

SQL — это не язык программирования, поэтому написать приложение или сайт с его помощью не получится, но при этом внутренняя работа сайта (бэкенд) невозможна без запросов. Поиск информации в Google — это тоже модель использования SQL. Пользователь задает параметры, которые его интересуют, и отправляет запрос на сервер, затем происходит магия и в поисковой выдаче появляются результаты, соответствующие именно этому запросу.

Б**азовые запросы SQL: CRUD и Select**

CRUD (CREATE - READ - UPDATE - DELETE) - это основные операции программирования СУБД, а именно СОЗДАНИЕ - ЧТЕНИЕ - ОБНОВЛЕНИЕ - УДАЛЕНИЕ данных таблицы, которые, в свою очередь отвечают за четыре основные функции: ВСТАВИТЬ - ВЫБРАТЬ - РЕДАКТИРОВАТЬ - УДАЛИТЬ.

CRUD постоянно используется для всего, что связано с базами данных и проектированием баз данных. Разработчики программного обеспечения ничего не могут сделать без операций CRUD. Например, при разработке веб-сайтов используется REST (передача репрезентативного состояния), который является надмножеством CRUD, используемого для ресурсов HTTP.

С другой стороны, CRUD не менее важен для конечных пользователей. Без него такие вещи, как регистрация на веб-сайтах, создание блогов или закладок, были бы невозможны. Большинство приложений, которые мы используем, позволяют нам добавлять или создавать новые записи, искать существующие, вносить в них изменения или удалять их.

Create позволяет добавлять новые строки в вашу таблицу.

Read - функция чтения похожа на функцию поиска, поскольку позволяет извлекать определенные записи и считывать их значения.

Update - это то, как мы изменяем существующую запись в таблице. Мы можем использовать это для изменения существующих записей в базе данных. При выполнении UPDATE вам необходимо определить целевую таблицу и столбцы, которые необходимо обновить. Вам также понадобятся связанные значения, а иногда и строки.

Рекомендуется ограничить количество строк, так как это помогает избежать проблем с параллелизмом.

Чтобы обновить существующую запись, используйте следующее:

```
UPDATE table_name
 SET column1 = value1, column2 = value2, ...
 WHERE condition;
```

Допустим, мы хотим обновить название и цену товара. Мы бы использовали:

```
UPDATE menu
 SET item_name = 'chocolate croissant', price = 2.5
 WHERE item_id = 1;
```

Это обновит таблицу, так что предыдущая запись с id 1теперь будет заменена шоколадным круассаном с price 2.5.

Delete - используется для удаления записи из таблицы. SQL имеет встроенную функцию удаления для одновременного удаления одной или нескольких записей из базы данных. Некоторые приложения реляционных баз данных могут разрешать жесткое удаление (безвозвратное удаление) или мягкое удаление (обновление статуса строки).

Команда удаления выглядит следующим образом:

```
DELETE FROM table_name WHERE condition;
```

Если мы хотим удалить один элемент из таблицы, мы используем:

> DELETE FROM menu WHERE item_name='bread';

Это приведет к удалению строки с хлебным предметом из таблицы. Если вы хотите удалить все записи из таблицы, вы можете использовать:

> DELETE FROM menu;

Select – позволяет производить выборку данных из таблиц и преобразовывать к нужному виду полученные результаты.

Результатом выполнения оператора `SELECT` является таблица. К этой таблице может быть снова применен оператор `SELECT` и т.д., то есть такие операторы могут быть вложены друг в друга. Вложенные операторы `SELECT` называют подзапросами.

Синтаксис оператора `SELECT` использует следующие основные предложения:

```
SELECT <список столбцов>
FROM <список таблиц>
[WHERE <условие выбора строк>]
[GROUP BY <условие группировки>]
[HAVING <условие выбора групп>]
[ORDER BY <условие сортировки>]
```

Кратко пояснить смысл предложений оператора `SELECT` можно следующим образом:

- `SELECT` - выбрать данные из указанных столбцов и (если необходимо) выполнить перед выводом их преобразование в соответствии с указанными выражениями и (или) функциями
- `FROM` - из перечисленных таблиц, в которых расположены эти столбцы
- `WHERE` - где строки из указанных таблиц должны удовлетворять указанному перечню условий отбора строк
- `GROUP BY` - группируя по указанному перечню столбцов с тем, чтобы получить для каждой группы единственное значение
- `HAVING` - имея в результате лишь те группы, которые удовлетворяют указанному перечню условий отбора групп
- `ORDER BY` - сортируя по указанному перечню столбцов

Как видно из синтаксиса рассматриваемого оператора, обязательными являются только два первых предложения: `SELECT` и `FROM`.

№	Команда	Пояснение
1	SHOW DATABASES	SQL - команда, которая отвечает за просмотр доступных БД
2	CREATE DATABASE	Команда для создания новой базы данных.
3	USE	С помощью этой SQL-команды `USE <database_name>` выбирается база данных, необходимая для дальнейшей работы с ней.
4	SOURCE	А `SOURCE <file.sql>` позволит выполнить сразу несколько SQL-команд, содержащихся в файле с расширением .sql.
5	DROP DATABASE	Стандартная SQL-команда для удаления целой базы данных.
6	SHOW TABLES	С помощью этой несложной команды можно увидеть все таблицы, которые доступны в базе данных.

№	Команда	Пояснение
7	CREATE TABLE	SQL-команда для создания новой таблицы: CREATE TABLE <table_name1> (<col_name1> <col_type1>, <col_name2><col_type2>, <col_name3><col_type3> PRIMARY KEY(<col_name1>), FOREIGN KEY(<col_name2>) REFERENCES <table_name2>(<col_name2>)); Ограничения целостности при использовании CREATE TABLE Может понадобиться создать ограничения для определённых столбцов в таблице. При создании таблицы можно задать следующие ограничения: - ячейка таблицы не может иметь значение NULL; - первичный ключ — PRIMARY KEY(col_name1, col_name2, …); - внешний ключ — FOREIGN KEY(col_namex1, …, col_namexn) REFERENCES table_name(col_namex1, …, col_namexn). Можно задать больше одного первичного ключа. В этом случае получится составной первичный ключ. Пример: Создайте таблицу «instructor»: CREATE TABLE instructor (ID CHAR(5), name VARCHAR(20) NOT NULL, dept_name VARCHAR(20), salary NUMERIC(8,2), PRIMARY KEY (ID), FOREIGN KEY (dept_name) REFERENCES department(dept_name));
8	DESCRIBE	С помощью `DESCRIBE <table_name>` можно просмотреть различные сведения (тип значений, является ключом или нет) о столбцах таблицы.
9	INSERT	Команда INSERT INTO <table_name> в SQL отвечает за добавление данных в таблицу: INSERT INTO <table_name> (<col_name1>, <col_name2>, <col_name3>, …) VALUES (<value1>, <value2>, <value3>, …); При добавлении данных в каждый столбец таблицы не требуется указывать названия столбцов. INSERT INTO <table_name> VALUES (<value1>, <value2>, <value3>, …);

№	Команда	Пояснение
10	UPDATE	SQL-команда для обновления данных таблицы: UPDATE <table_name> SET <col_name1> = <value1>, <col_name2> = <value2>, ... WHERE <condition>;
11	DELETE	SQL-команда `DELETE FROM <table_name>` используется для удаления данных из таблицы.
12	DROP TABLE	Можно удалить всю таблицу целиком
13	SELECT	К одной из основных команд, которые позволяют работать непосредственно с данными, относится SELECT для получения данных из выбранной таблицы: SELECT <col_name1>, <col_name2>, … FROM <table_name>; Следующей командой можно вывести все данные из таблицы: SELECT * FROM <table_name>;
14	SELECT DISTINCT	В столбцах таблицы могут содержаться повторяющиеся данные. Используйте SELECT DISTINCT для получения только неповторяющихся данных. SELECT DISTINCT <col_name1>, <col_name2>, … FROM <table_name>;
15	WHERE	Можно использовать ключевое слово WHERE в SELECT для указания условий в запросе: SELECT <col_name1>, <col_name2>, … FROM <table_name> WHERE <condition>; В запросе можно задавать следующие условия: - сравнение текста; - сравнение численных значений; - логические операции AND (и), OR (или) и NOT (отрицание). Пример: попробуйте выполнить следующие команды. Обратите внимание на условия, заданные в WHERE: SELECT * FROM course WHERE dept_name=’Comp. Sci.’; SELECT * FROM course WHERE credits>3; SELECT * FROM course WHERE dept_name='Comp. Sci.' AND credits>3;

№	Команда	Пояснение
16	GROUP BY	Оператор GROUP BY часто используется с агрегатными функциями, такими как COUNT, MAX, MIN, SUM и AVG, для группировки выходных значений. SELECT <col_name1>, <col_name2>, … FROM <table_name> GROUP BY <col_namex>; Пример: выведем количество курсов для каждого факультета: SELECT COUNT(course_id), dept_name FROM course GROUP BY dept_name;
17	HAVING	Ключевое слово HAVING было добавлено в SQL по той причине, что WHERE не может использоваться для работы с агрегатными функциями. SELECT <col_name1>, <col_name2>, ... FROM <table_name> GROUP BY <column_namex> HAVING <condition> Пример: выведем список факультетов, у которых более одного курса: SELECT COUNT(course_id), dept_name FROM course GROUP BY dept_name HAVING COUNT(course_id)>1;
18	ORDER BY	Используется для сортировки результатов запроса по убыванию или возрастанию. ORDER BY отсортирует по возрастанию, если не будет указан способ сортировки ASC или DESC. SELECT <col_name1>, <col_name2>, … FROM <table_name> ORDER BY <col_name1>, <col_name2>, … ASC|DESC; Пример: выведем список курсов по возрастанию и убыванию количества кредитов: SELECT * FROM course ORDER BY credits; SELECT * FROM course ORDER BY credits DESC;
19	BETWEEN	Используется для выбора значений данных из определенного промежутка. Могут быть использованы числовые и текстовые значения, а также даты SELECT <col_name1>, <col_name2>, … FROM <table_name> WHERE <col_namex> BETWEEN <value1> AND <value2>; Пример: Выведем список инструкторов, чья ЗП больше 50К, но меньше 100К SELECT * FROM instructor WHERE salary BETWEEN 50000 AND 100000;

№	Команда	Пояснение
20	LIKE	Оператор LIKE используется в WHERE, чтобы задать шаблон поиска похожего значения. Есть два свободных оператора, которые используются в LIKE: % (ни одного, один или несколько символов); _ (один символ). SELECT <col_name1>, <col_name2>, … FROM <table_name> WHERE <col_namex> LIKE <pattern>; Пример: Выведем список курсов, в имени которых содержится “to”, и список курсов, названия которых начинается с “CS-”: SELECT * FROM course WHERE title LIKE ‘%to%’; SELECT * FROM course WHERE course_id LIKE 'CS-___';
21	IN	С помощью IN можно указать несколько значений для оператора WHERE: SELECT <col_name1>, <col_name2>, … FROM <table_name> WHERE <col_namen> IN (<value1>, <value2>, …); Пример: Выведем список студентов с направлений Comp. Sci., Physics и Elec. Eng.: SELECT * FROM student WHERE dept_name IN (‘Comp. Sci.’, ‘Physics’, ‘Elec. Eng.’);
22	JOIN	используется для связи двух или более таблиц с помощью общих атрибутов внутри них. Существуют различные способы объединения в SQL: ЛЕВОСТОРОННЕЕ ОБЪЕДИНЕНИЕ - все строки из A, даже если их нет в B; ВНУТРЕННЕЕ ОБЪЕДИНЕНИЕ - строки, содержащиеся и в A, и в B; ПРАВОСТОРОННЕЕ ОБЪЕДИНЕНИЕ - все строки из B, даже если их нет в A. SELECT <col_name1>, <col_name2>, … FROM <table_name1> JOIN <table_name2> ON <table_name1.col_namex> = <table2.col_namex>; Пример: выберем список всех обязательных курсов и детали о них SELECT prereq.course_id, title, dept_name, credits, prereq_id FROM prereq LEFT OUTER JOIN course ON prereq.course_id=course.course_id;

№	Команда	Пояснение
23	VIEW	это виртуальная таблица SQL, созданная в результате выполнения выражения. Она содержит строки и столбцы и очень похожа на обычную SQL-таблицу. `VIEW` всегда показывает самую свежую информацию из базы данных. Создание CREATE VIEW <view_name> AS SELECT <col_name1>, <col_name2>, … FROM <table_name> WHERE <condition>; Удаление DROP VIEW <view_name>;
24	Агрегатные функции	Это не совсем основные команды SQL, однако знать их тоже желательно. Агрегатные функции используются для получения совокупного результата, относящегося к рассматриваемым данным: COUNT(col_name) — возвращает количество строк; SUM(col_name) — возвращает сумму значений в данном столбце; AVG(col_name) — возвращает среднее значение данного столбца; MIN(col_name) — возвращает наименьшее значение данного столбца; MAX(col_name) — возвращает наибольшее значение данного столбца.
25	Вложенные подзапросы	это SQL запросы, которые включают выражения SELECT, FROM и WHERE, вложенные в другой запрос Пример Найдем курсы, которые преподавались осенью 2009 и весной 2010 годов SELECT DISTINCT course_id FROM section WHERE semester = ‘Fall’ AND year= 2009 AND course_id IN (SELECT course_id FROM selection WHERE semester = ‘Spring’ AND year = 2010);