sql syntax

💻 Summary of SQL syntax

Recall

- What are the different types of SQL syntax?

Recall
Overview
  Data Definition Language
    CREATE
    ALTER
    DROP
    RENAME
    TRUNCATE
  Data Manipulation Language
    SELECT
    INSERT
    UPDATE
    DELETE
  Data Control Language
    GRANT
    REVOKE
  Transaction Control Language
    COMMIT
    ROLLBACK

Overview

SQL is the language used to create and manipulate relational databases.

There are a few different programming types:

- Procedural (imperative)
  - The solution to a problem is defined in several distinct steps.
  - C, Java
- Object-oriented
  - Python
- Declarative
  - The desired solution is stated, and the language handles the how, through an optimizer (this does the procedural).
  - SQL
- Functional

SQL is regarded as a declarative programming language. This means that while coding, the focus is on WHAT results you want to obtain, rather than HOW.

SQL syntax comprises of several types of statements that allow you to perform various commands and operations. There are three main components:

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Data Control Language (DCL)
- Transaction Control Language (TCL)

Data Definition Language

This is a syntax. It comprises of a set of keyword (reserved words) statements that allow the user to define or modify data structures and objects, such as tables.

CREATE

This is used for creating entire databases and database objects as tables.

```
CREATE <object_type> <object_name>;
CREATE TABLE <object_name> (<column_name> <data_type>);
```

```
-- Creating a table within a database
CREATE TABLE sales (purchase_number INT);
```

ALTER

This is used to alter existing objects. You can add, remove, or rename elements.

```
ALTER <object_type> <object_name>
<ALTER_TYPE> <object_type> <object_name> <data_type>;

-- Altering the table by adding a new column of the DATE type
ALTER TABLE sales
ADD COLUMN date_of_purchase DATE;
```

DROP

This is used for deleting a database object

```
DROP <object_type> <object_name>;

-- Removes the database object
DROP TABLE customers;
```

RENAME

This allows you to rename an object, such as a table.

```
RENAME <object_type> <object_name> TO <new_object_name>;

-- Renaming the table
RENAME TABLE customers TO customer_data;
```

TRUNCATE

Instead of deleting an entire table through DROP, we can also remove its data and continue to have the table as an object in the database.

```
TRUNCATE <object_type> <object_name>;
```

```
-- Clearing the table
TRUNCATE TABLE customers;
```

Data Manipulation Language

Its statements allow us to manipulate the data in the tables of a database.

SELECT

This is used to retrieve data from database objects, like tables. You can use this to extract specific data from a table. It is very important to master SELECT. Used often with `FROM`.

The `*` delivers the entire content, all records and fields.

```
SELECT <selector> FROM <table_name>;

-- Selecting all data from sales table
SELECT * FROM sales;
```

INSERT

This is used to insert data into tables. This works with the keywords `INTO` and `VALUES`. You have to specify the columns unless you want to add data into all the columns.

```
INSERT <keyword> <table_name> (<column_name(s)>) <keyword> (<new_data>);

-- Inserting data into the sales table, into the respective columns
INSERT INTO sales (purchase_number, date_of_purchase) VALUES (1, '2017-10-11');

-- The sales table only has two tables, so this is an identical statement
INSERT INTO sales VALUES (1, '2017-10-11');
```

UPDATE

This statement allows you to renew existing data in your tables. The syntax is slightly different.

```
-- Example of updating data
UPDATE sales
```

```
SET date_of_purchase = '2017-12-12'
WHERE purchase_number = 1;
```

DELETE

This statement functions similar to the `TRUNCATE` statement. The difference is that you can specify precisely what you want to be removed.

```
-- Remove all records from sales table
DELETE FROM sales;

-- Remove the specific record where purchase_number = 1
DELETE FROM sales
WHERE purchase_number = 1;
```

Data Control Language

This only contains two statements, which allow you to manage the rights users have in a database. Those who have complete rights to a database are referred to as the database administrators. They can grant access to users and also revoke them.

GRANT

This gives certain permissions to users of the database. These permissions will be assigned to the person who has a username registered with that host IP.

```
GRANT <type_of_permission> ON <database_name>.<table_name> TO <username>@<localhost>

-- Creating a mock user
CREATE USER 'mock'@'localhost' IDENTIFIED BY 'pass';

-- Granting only the use of select
GRANT SELECT ON sales.customers TO 'mock'@'localhost';

-- Granting all perms
GRANT ALL ON sales.* TO 'mock'@'localhost';
```

REVOKE

This is used to revoke permissions and privileges of database users. The opposite of `GRANT`, with identical syntax.

```
REVOKE <type_of_permission> ON <database_name>.<table_name> TO <username>@<localhost>

-- Revoking only the use of select
REVOKE SELECT ON sales.customers TO 'mock'@'localhost';

-- Revoking all perms
REVOKE ALL ON sales.* TO 'mock'@'localhost';
```

Transaction Control Language

Not every change made to a database is saved automatically, you have to explicit state that you want a save to happen.

COMMIT

This is related to `INSERT`, `DELETE`, and `UPDATE`. This will save the changes that were made of the database, and allow other users to access the new modified version of the database.

```
-- Committing a change
UPDATE customers
SET last_name = 'Johnson'
WHERE customer_id = 4
COMMIT;
```

ROLLBACK

This will let you take a step back, and undo any changes you have made since the last `COMMIT` and don't want to be saved permanently. Be careful with this, all the changes made after the last committed change will be removed.

```
-- Rolling back a committed change
UPDATE customers
SET last_name = 'Johnson'
WHERE customer_id = 4
COMMIT;

ROLLBACK;
```