SQL basics

Data definition language (DDL)

Data definition language

alx

The sublanguage responsible for defining how data are structured in a database in SQL is called the **data definition language (DDL)**.

The commands that are used to **build, amend, or remove** SQL tables are contained in the data definition language.

These commands include **CREATE TABLE, ALTER TABLE, TRUNCATE TABLE, and DROP TABLE.**

Database schemas and tables

alx

Tables are the fundamental building blocks of a database schema and **store data in rows and columns**.
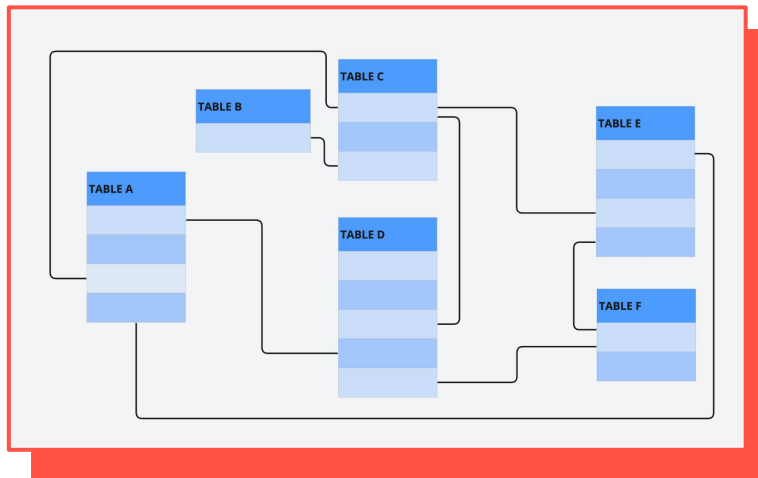
A database schema is a **logical container that houses these tables** and provides a framework for classifying, ordering, and arranging them in **relation to one another**.

TABLE A

col1	col2	col3
x	34	s
y	73	m
z	22	l
w	12	m

TABLE B

col1	col2	col3
x	34	s
y	73	m
z	22	l
w	12	m

TABLE A
TABLE B
TABLE C
TABLE D
TABLE E
TABLE F

Database schemas and DDL

A SQL sublanguage known as **data definition language**, or **DDL**, is used to **create, modify, or remove** SQL tables from the database schema.

```
CREATE DATABASE united_nations;
USE united_nations;

CREATE TABLE united_nations.Access_to_Basic_
    Region VARCHAR(32),


ALTER TABLE Access_to_Basic_Services
MODIFY COLUMN Country_name VARCHAR(37);


    DROP TABLE Access_to_Basic_Services;
    DROP DATABASE united_nations;
```

CREATE DATABASE

alx

The CREATE DATABASE statement is used to **create a new SQL database**.

```
CREATE DATABASE database_name;
USE database_name;
```

```
1   CREATE DATABASE united_nations;
2   USE united_nations;
```

Creating a database typically **requires appropriate permissions or privileges** depending on the database management system we are working with.

1. Creates a database named united_nations.

2. Selects the united_nations database. All subsequent SQL operations will be performed inside this database.

CREATE TABLE

The CREATE TABLE statement is used to **create new** tables. It specifies the structure of the table, defining the columns and their data types.

Syntax

```
CREATE TABLE table_name (
column1 datatype,
Column2 datatype [Constraint],
....
);
```

1. Creates a table inside the united_nations database named Access_to_Basic_Services.

2. If the "**USE database_name"** function wasn't executed, the **database name is entered before the table name.**

3. Inside the brackets, it **defines the name of each column** and its **data type,** separated by a

Example

```
1   CREATE TABLE united_nations.Access_to_Basic_Services(
2       Region VARCHAR(32),
3       Sub_region VARCHAR(25),
4       Country_name INTEGER NOT NULL
5       Time_period INTEGER  NOT NULL,
6       Pct_managed_drinking_water_services NUMERIC(5,2),
7       Pct_managed_sanitation_services NUMERIC(5,2),
8       Est_population_in_millions NUMERIC(11,6),
9       Est_gdp_in_billions NUMERIC(8,2),
10      Land_area NUMERIC(10,2),
11      Pct_unemployment NUMERIC(5,2)
12  );
```

4. After the data type, we can insert an optional **constraint** that allows us to **enforce rules** on the **type of data** the column can have, e.g., **NOT NULL.**

Constraints

When creating a table in SQL, we can apply various **constraints** to columns to **enforce data integrity** and **define rules** for the values stored in those columns. Here are some commonly used constraints in SQL:

NOT NULL

This constraint ensures that a column **cannot contain NULL values**. It enforces the requirement for the column to have a non-null value for each row.

UNIQUE

This constraint ensures that the values in a column (or a combination of columns) are **unique across the table**. It **prevents duplicate values** from being inserted into the column(s).

PRIMARY KEY

The PRIMARY KEY constraint **uniquely identifies each row in a column,** combining the `NOT NULL` **and** `UNIQUE` constraints. The primary key column values are unique and cannot be null.

FOREIGN KEY

This constraint **establishes a relationship between two tables** based on a column. It ensures that the values in the **primary key column** in the first table **correspond** to the values in the **foreign key column** in the second table.

ALTER TABLE

The ALTER TABLE statement is used to **modify the structure of an existing database object**, such as adding, modifying, or deleting columns in a table.

Syntax

To add a column
```
ALTER TABLE table_name
ADD column_name datatype;
```

To delete a column
```
ALTER TABLE table_name
DROP COLUMN column_name;
```

To rename a column
```
ALTER TABLE table_name
RENAME COLUMN old_name to new_name;
```

To change the data type of a column
```
ALTER TABLE table_name
MODIFY COLUMN column_name datatype;
```

Examples

```
1    -- Add column Gini_index with datatype FLOAT
2    ALTER TABLE Access_to_Basic_Services
3    ADD Gini_index FLOAT;
4
5    -- Drop column Gini_index
6    ALTER TABLE Access_to_Basic_Services
7    DROP COLUMN Gini_index;
```

```
1    ALTER TABLE Access_to_Basic_Services
2    MODIFY COLUMN Country_name VARCHAR(37);
```

TRUNCATE TABLE

The TRUNCATE TABLE statement is used to **remove all data from a table**, effectively resetting it to an empty state. This operation is faster than deleting individual rows.

Syntax

```
TRUNCATE TABLE table_name;
```

*If the "USE database_name" function wasn't executed, the database name is entered before the table name.

Example

```
1    TRUNCATE TABLE united_nations.Access_to_Basic_Services;
```

As soon as the TRUNCATE TABLE statement is executed, the data are **permanently wiped from the table** and **cannot be recovered; hence,** it is important to use this command with caution. Appropriate backups of the data are required.

Removes all the content of the `Access_to_Basic_Services` table without deleting the table itself.

DROP TABLE and DROP DATABASE

The DROP statements are used to **remove entire database objects**, such as tables or schemas, from the database.

Syntax

```
DROP TABLE table_name;

DROP DATABASE database_name;
```

Example

```
1    DROP TABLE Access_to_Basic_Services;
2    DROP DATABASE united_nations;
```

It is important to exercise caution when using the `DROP TABLE` or `DROP DATABASE` statements, as they **permanently delete the table or database**, and they cannot be recovered.

Deletes the `Access_to_Basic_Services` table and then deletes the `united_nations` database as well.