Learning Objectives

Learners will be able to...

- Create table structure
- Add and use correct data type and default values
- Choose field constraints
- Delete table and field of table

info

Make Sure You Know

Learners should understand the concept of a database, what it is used for, and its importance in organizing and storing data efficiently.

Learners should be familiar with basic SQL queries like SELECT, INSERT, UPDATE, and DELETE. They should also understand how to interact with databases using SQL commands.

CREATE TABLE statement

You can create a new empty table in the current database using the CREATE TABLE statement.

The table will be owned by the user who issued this command. The table name to be specified after CREATE TABLE must be different from the names of other tables in the same database.

Syntax:

```
CREATE TABLE [IF NOT EXISTS] table_name (
   column1 datatype(length) column_constraint,
   column2 datatype(length) column_constraint,
   column3 datatype(length) column_constraint,
   table_constraints
);
```

If **IF NOT EXISTS** clause specified - the query will not fail, if a table with the same name already exists, all statements will be ignored.

Column names, their types and restrictions are listed in parentheses, restrictions specific to the entire table can be listed after all columns.

For example, creating a table of actors might look like this.

```
CREATE TABLE actor (
  actor_id integer PRIMARY KEY,
  first_name varchar(45) NOT NULL,
  last_name varchar(45) NOT NULL,
  last_update timestamp without time zone DEFAULT now() NOT NULL
);
```

Additionally, when creating a table, default values and relationships between tables that already exist in the database can be set.

Data Types

As the data stored in the database continues to expand rapidly, the optimization of data storage becomes a crucial undertaking. The size of the data has a significant impact not only on the storage requirements and associated costs but also on the performance of database queries. The critical element that determines the storage size is the data type selected for each column in the database.

For example, let's take a type like UUID. It can be stored as a string, and then it will occupy 36 bytes in the database. But in digital representation, as a UUID type, it will only take 16 bytes! Thus, on each entry, we can only save 20 bytes by using the correct type. It doesn't seem like much, but at 50 million records, that would already be a 1 terabyte savings, which is significant.

It is recommended to choose a data type that allows you to implement business logic and takes up a minimum amount of space.

You should try to adhere to the following rules:

- If the data is numeric, favor SMALLINT, INTEGER, BIGINT, or DECIMAL data types. DECFLOAT and FLOAT are also options for very large numbers.
- If the data is character, use CHAR or VARCHAR data types.
- If the data is date and time, use DATE, TIME, and TIMESTAMP data types.
- If the data is multimedia, use GRAPHIC, VARGRAPHIC, BLOB, CLOB, or DBCLOB data types.

DEFAULT values

The DEFAULT constraint is used when no value was passed to the column specified with the default constraint. For this column the data will be automatically populated with the default value.

The DEFAULT constraint is used to set a default value for a column.

For example, if we want to set default values for last name and first name, it might look like this. Thus, in the absence of data, we will get the first and last name - John Doe.

```
first_name varchar(45) DEFAULT 'John' NOT NULL,
last_name varchar(45) DEFAULT 'Doe' NOT NULL
```

Another commonly used case is setting the creation time of the entry.

```
last_update timestamp without time zone DEFAULT now()
```

The `last_update` field will be set to the current time(value of `now()` function) if no additional data has been passed.

This method can also be used to generate unique sequential values. Each of the DBMS can have differences, in PostgreSQL sequences are used for this.

```
actor_id integer DEFAULT nextval('actor_actor_id_seq'::regclass)
```

In addition, the previously described UUID type can also be used as a unique value.

```
actor_id uuid DEFAULT gen_random_uuid ()
```

Constraints

Constraints are rules that we can apply to the type of data in a table. With their help, we can control the data integrity and business logic of the application.

Constraints can be at the column or table level. Column-level restrictions apply to a column, while table-level restrictions apply to the entire table.

NOT NULL

Ensures that a column cannot have a NULL value.

```
last_name varchar(45) NOT NULL
```

UNIQUE

Ensures that all values in a column are different.

```
email varchar(100) UNIQUE
```

PRIMARY KEY

A primary key is a field which can uniquely identify each row in a table and cannot be null. And this constraint is used to specify a field in a table as primary key.

```
actor_id integer PRIMARY KEY,
```

info

PRIMARY KEY

PRIMARY KEY is combination of UNIQUE plus NOT NULL constraints. But the main difference of PRIMARY KEY is that it can be specified only **once**, when UNIQUE plus NOT NULL can be used **many times**.

FOREIGN KEY

This constraint is used for relationships between tables, prevents actions that would destroy links between tables.

CHECK

This constraint helps to check the values of a column or columns against a certain condition - a predicate.

```
age integer CHECK (age >= 18)
```

Validation conditions can also be compound and use multiple fields. Such checks are located at the end of the table description.

```
CHECK (age >= 21 AND city='Warwick')
```

Delete Tables

If a table needs to be deleted, then the delete operator is used - **DROP TABLE**.

```
DROP TABLE table_name;
```

important

Warning

Be careful when dropping tables, this action can lead to data loss.

Attempting to drop a table that does not exist will result in an error.

```
DROP TABLE table_name;
ERROR:  table "table_name" does not exist
```

If we want to avoid this behavior, we can use the **IF EXISTS** clause.

```
DROP TABLE IF EXISTS table_name;
```