CRUD in SQL.

Objectives:

By the end of this chapter, you should be able to:

- Compare and contrast DML and DDL
- Use DDL to perform CRUD on tables, columns and databases
- Use DML to perform CRUD on records in a table

DDL vs DML

When working with SQL, the commands you will be using fall into two major categories:

DDL - Data Definition Language - this refers to the SQL syntax and commands around creating, modifying and deleting **tables**, **columns** and **databases**.

DML - Data Manipulation Language - this refers to the SQL syntax and commands around creating, reading, modifying and deleting **rows**.

Let's first focus on DDL.

Creating a table (DDL)

Open up postgres by typing `psql` in your terminal. Next enter the following:

```
CREATE TABLE users (id SERIAL PRIMARY KEY,
                    first_name TEXT,
                    last_name TEXT);
```

In the example above, `users` is the name of the table we are creating. The `id`, `first_name`, and `last_name` are all columns in the `users` table. `SERIAL` and `TEXT` are examples of data types, which we will talk about in detail next. `PRIMARY KEY` is a constraint placed on the column.

`\d+ users` should show our newly-created `users` table with 3 columns: `id`, `first_name`, `last_name`

id	first_name	last_name

Data Types in Postgres

In relational databases like postgres, you must specify the type of data that you plan to store in a column. Here are the types in postgres:

- **SERIAL** - auto incrementing integer, perfect for IDs
- **TEXT** - pieces of text (equally as space efficient as VARCHAR)

- **VARCHAR** - a variable number of characters
- **CHAR** - a fixed number of characters
- **BOOLEAN** - a boolean
- **INTEGER** - an integer
- **REAL** - a floating point number, e.g., 3.141593
- **DECIMAL**, **NUMERIC** - floating point numbers that have user specified percision.
- **MONEY** - floating point numbers use for money
- **ARRAY** - an array (array types are rarely used)

For more on the difference between text, varchar, and char, check out this StackOverflow post.

Constraints

Constraints are certain database table restrictions that are either implicitly or explicitly created via the database schema. Constraints are a key part of DDL. Violating a constraint will throw a database error and (usually) abort the intended operation.

Common constraints are:

- **not null** - when creating or updating a record, a column value cannot be made NULL. This constraint is requiring information; for example, a table of college students cannot have NULL for their college major.

```
CREATE TABLE college_students (
    id SERIAL PRIMARY KEY,
    last_name VARCHAR(50),
    first_name VARCHAR(50),
    major VARCHAR(50) NOT NULL
);
```

- **unique** - a column value for a record must be unique in its table. For example, consider a table of users with a phone_number column: no two users may have the same phone number, so a unique constraint is placed on that column.

```
CREATE TABLE phonebook (
    id SERIAL PRIMARY KEY,
    last_name VARCHAR(50),
    first_name VARCHAR(50),
    phone_number VARCHAR(7) UNIQUE
);
```

- **primary key** - an identifier for a record which has both unique and not-null constraints. Primary keys are used internally by the RDBMS to reference rows. Good examples of primary keys include: drivers license number, social security number, auto-generated unique IDs such as a UUID, etc. Bad examples are things like email address, full name, or phone number.

```
CREATE TABLE users (id SERIAL PRIMARY KEY,
                    first_name TEXT,
                    last_name TEXT);
```

- **foreign key** - *we'll talk about foreign keys later when we talk about joins.*
- **check** - an expression is provided that must evaluate truthy for the operation to proceed. For example, if you have a table of products for an online shopping site, you might put a check constraint on products to have price > 0.

```
CREATE TABLE products (
    product_no SERIAL PRIMARY KEY,
    name TEXT,
    price NUMERIC CHECK (price > 0)
);
```

For more info about constraints and proper syntax in PostgreSQL, check out the PostgreSQL docs

Adding a column in a table

```
ALTER TABLE users ADD COLUMN favorite_number INTEGER;
```

`\d+ users` and we should see favorite_number

id first_name last_name favorite_number

Removing a column in a table

```
ALTER TABLE users DROP COLUMN favorite_number;
```

`\d+ users` and favorite_number should no longer exist

id first_name last_name

Renaming columns in a table

```
ALTER TABLE users ADD COLUMN jobb TEXT;
```

id first_name last_name jobb

```
ALTER TABLE users RENAME COLUMN jobb TO job;
```

`\d+ users` and we should see 'job' spelled correctly now

id first_name last_name job

Changing the datatype of a column in a table

```
ALTER TABLE users ADD COLUMN favorite_number TEXT;
ALTER TABLE users ALTER COLUMN favorite_number SET DATA TYPE VARCHAR(100);
```

`\d+ users` and we should see a different data type for our column

Adding a constraint to a table

```
ALTER TABLE users ADD CONSTRAINT favorite_number NOT NULL;
```

`\d+ users` and we should see favorite_number is not nullable.

Creating and Modifying data (DML)

When we perform CRUD operations on our rows (not columns, tables or databases) we are using DML or Data Manipulation Language.

CRUD	SQL
Create	INSERT
Read	SELECT
Update	UPDATE
Delete	DELETE

Let's get started with reading data from our tables using `SELECT`.

SELECT

```
--to select all rows and columns--
SELECT * FROM users;

--to select specific columns--
SELECT id, first_name FROM users;

--to select specific columns and rows--
SELECT id, first_name FROM users WHERE id=1;
```

INSERT

To insert or add data to a table - we use the INSERT command

```
--start with the INSERT INTO commands and then specify a table(column1,
column2, ...) and VALUES for each column.
INSERT INTO users(first_name, last_name) VALUES ('Elie', 'Schoppik');
```

UPDATE

To update a row or multiple rows we use the `UPDATE` command.

```
UPDATE users SET first_name = 'Elie'; -- will update all users
UPDATE users SET first_name = 'Elie' WHERE id = 1; -- will update a user with
an id of 1
```

DELETE

To delete a row or multiple rows we use the `DELETE FROM` command.

```
DELETE FROM users; -- will delete all users
DELETE FROM users WHERE id=1; -- will delete a user with an id of 1
```