Evans Mutwiri

6 April 2022

Title: Version Control

[GitHub Link](https://github.com/EvansMutwiri/my--internship.git)

Structured Query Language

A database is a systematic collection of data. Databases support electronic storage and manipulation of data. They make data management easy.

Some of the most popular types of database types are:

* Distributed database - has no central storage.
* Relational Database
* Object oriented databases - postgresql is an example
* Centralized database
* Open-source database
* Cloud database
* NoSQL databases

There are 5 main components:

1. Hardware
2. Software
3. Data
4. Procedure
5. Database Access Language

Data access language is used to access data to and from the database, enter new data, update existing data or retrieve required data from Database Management System.

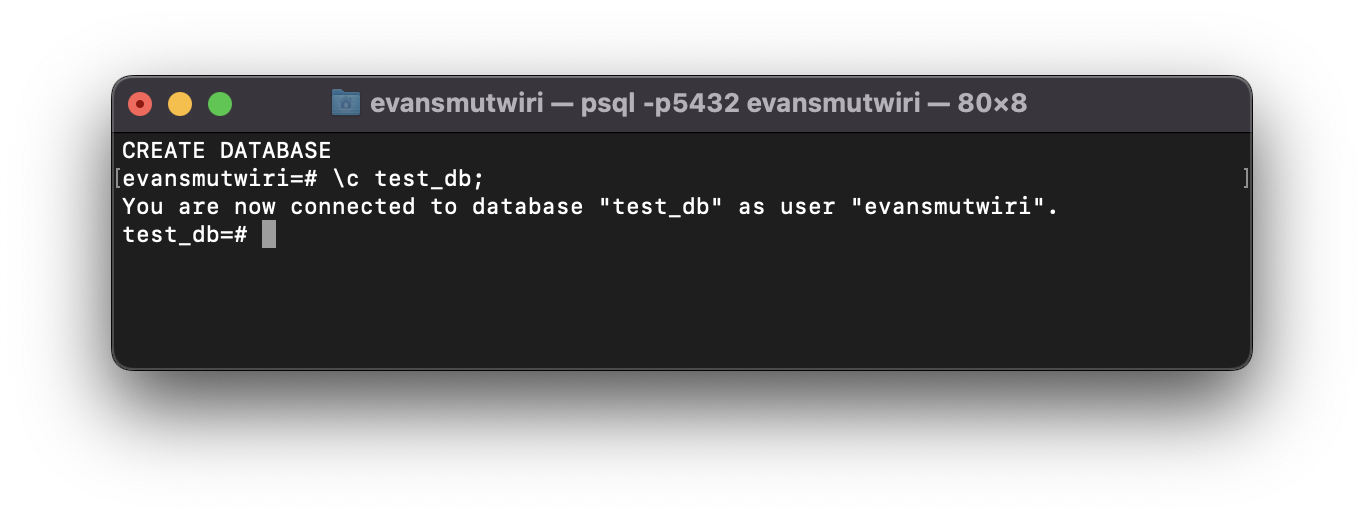
PostgreSQL and MySQL are *relational database management systems* (RDBMS). Data is stored in relations(tables). Each table is a named collection of *rows*. Each row of a given table has the same set of named *columns*, and each column is of a specific data type. Tables are grouped into databases, and a collection of databases managed by a single PostgreSQL server instance constitutes a database *cluster*.

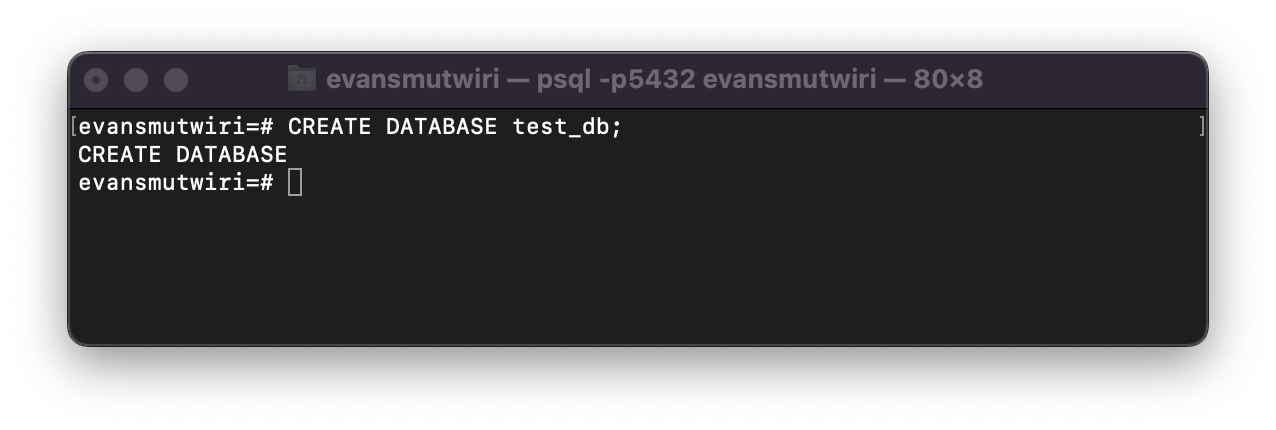
*Structured Query language* (SQL) is the standard for dealing with relational databases. SQL can be used to insert, search, update, and delete database records and other operations such as optimizing and maintenance of dbs.

**Managing Databases**

* **Selecting Database**: Once you get connected to the server, it is required to select a database to work with, because there might be more than one database available in the server.

\c db\_name; and use db\_name; commands are used to connect to a database in postgres and MySQL respectively.



* **Creating Database:** use the CREATE DATABASE statement. For example CREATE DATABASE db\_name;
* **Dropping Database:** use the DROP DATABASE statement. For example DROP DATABASE db\_name;
* **Create Table:** You can create a new table by specifying the table name, along with all column names and their types: 

It is possible to write each statement in its line because postgres will not recognize a command until you add a semicolon;

-- introduces comments up to the end of the line.

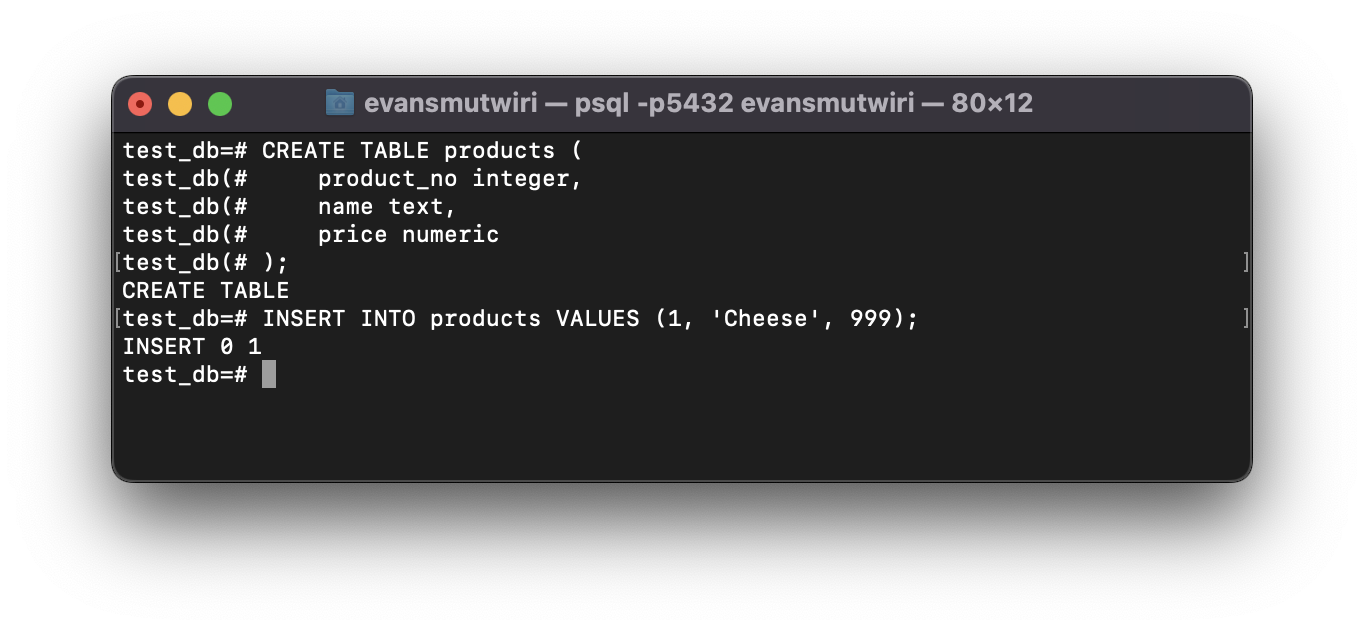
**Data types:**

The example shows a few data types available in postgres, varchar(80) string of maximum 80 characters, int for integer numbers, real for floating point numbers, date for dates.

Other data types include as specified by SQL: bigint, bit, bit varying, boolean, char, character varying, character, varchar, date, double precision, integer, interval, numeric, decimal, real, smallint, time (with or without time zone), timestamp (with or without time zone), xml.

**Drop Table:** when a table is no longer useful DROP TABLE is used to delete it.

Data manipulation

1. To create a new row, use the INSERT command. The command requires the table name and column values. The data values are listed in the order in which the columns appear in the table, separated by commas.Although the above syntax is allowed it would be wise to list columns explicitly first so that the values can align to them. Column values can be missing, they will be filled with default values but the row must be complete.
2. Updating data - ie modification of data already inside the database. You can update individual rows, all the rows in a table, or a subset of all rows. Each column can be updated separately; the other columns are not affected.

To update existing rows, use the UPDATE command. This requires three pieces of information:

* The name of the table and column to update
* The new value of the column
* Which row(s) to update - you specify which conditions a row must meet in order to be updated. Example:

UPDATE products SET price = 10 WHERE price = 5;

This might cause zero, one or many rows to be updated. It is not an error if the update attempt does not match any rows.

Syntax: first is the UPDATE keyword, then the table name, SET keyword, equal sign, and a new column value. The value doesn't need to be a constant eg. to increase the price by 10%

UPDATE products SET price = price \* 1.10;

When the WHERE keyword is used only values that match the condition are updated. The equals sign in SET is an assignment while the one in WHERE clause is a comparison. More than one value can be updated in the SET clause.

UPDATE mytable SET a = 5, b = 3, c = 1 WHERE a > 0;

1. Deleting data - just as adding data is only possible in whole rows, you can only remove entire rows from the table. If you have the primary key you can specify the exact row to remove or remove a row that matches specified conditions, or you can remove all rows at once. Use the DELETE key word.

DELETE FROM products WHERE price = 10;

Or remove all rows!

DELETE FROM products;

1. **Queries** - The process of retrieving or the command to retrieve data from a database is called a query. The SELECT command is used to specify queries. General syntax:

Assuming there exists a table called table1 this selects all the rows and columns from table1.

SELECT \* FROM table1;

Table expressions: A table expression computes a table. It contains a FROM clause that is optionally followed by WHERE, GROUP BY, and HAVING clauses. These optional clauses specify a pipeline of successive transformations performed on the table derived from the FROM clause.

**Joined Tables:**

A joined table is a table derived from two other (real or derived) tables according to the rules of the particular join type. Inner, outer, and cross-joins are available. The general syntax of a joined table is

***T1*** ***join\_type*** ***T2*** [ ***join\_condition*** ]

Joins of all types can be chained together, or nested: either or both ***T1*** and ***T2*** can be joined tables. Parentheses can be used around JOIN clauses to control the join order. In the absence of parentheses, JOIN clauses nest left-to-right.

Join Types:

1. Cross join

***T1*** CROSS JOIN ***T2***

1. Qualified joins

* INNER JOIN - For each row R1 of T1, the joined table has a row for each row in T2 that satisfies the join condition with R1.
* LEFT OUTER JOIN - First, an inner join is performed. Then for each row in T1 that does not satisfy the join condition with any of T2, a joined row is added with null values in columns of T2. Thus, the joined table always has at least one row for each row in T1.
* RIGHT OUTER JOIN - First, an inner join is performed. Then, for each row in T2 that does not satisfy the join condition with any row in T1, a joined row is added with null values in columns of T1. This is the converse of a left join: the result table will always have a row for each row in T2.
* FULL OUTER JOIN - First, an inner join is performed. Then, for each row in T1 that does not satisfy the join condition with any row in T2, a joined row is added with null values in columns of T2. Also, for each row of T2 that does not satisfy the join condition with any row in T1, a joined row with null values in the columns of T1 is added.

**References**

<https://www.w3schools.com/sql/default.asp>

<https://www.freecodecamp.org/news/sql-and-databases-full-course/>

<https://www.postgresql.org/docs/>

<https://www.tutorialspoint.com/postgresql/postgresql_data_types.htm>

<https://www.guru99.com/postgresql-tutorial.html>