Week 4: SQL, Tables and Migrations

1. SQL

Syntax for creating tables ->

CREATE TABLE flights (

Id INTEGER PRIMARY KEY AUTOINCREMENT,

Origin TEXT NOT NULL,

DESTINATION TEXT NOT NULL,

DURATION INTEGER NOT NULL

);

Types of Constraints

* Check
* Default
* Not Null
* Primary Key
* Unique

INSERT INTO flights (origin, destination, duration) VALUES (“New York”, “London”, 415);

1. SQL Queries
2. Retrieve Data

SELECT \* FROM flights -> \* means everything

SELECT \* FROM flights WHERE id = 3; or SELECT \* FROM flights WHERE origin = “New York”

SELECT \* FROM flights WHERE duration > 500 AND destination = “Paris”;

SELECT \* FROM flights WHERE origin IN (“New York”, “Lima”);

**More Complicated Searches in SQL**

What if you want to find a partial match for example you want origin in flights where it contains the letter a

SELECT \* FROM flights WHERE origin LIKE “%a%” -> % is a wildcard character in SQL where it means match any character. The LIKE command tells SQL we are not looking for an exact match

1. Update Data

UPDATE flights SET duration = 430 WHERE origin = “New York” AND destination = “London”;

1. DELETE Data

DELETE FROM flights WHERE origin = “Tokyo”;

1. Useful Clauses

* LIMIT
* ORDER BY
* GROUP BY
* HAVING

1. Foreign Keys

Tables in SQL have a tendency to spiral out of control and increase in size because we have too much data. Therefore, impt for us to design the right tables to have in any application. However, these tables have to be linked to one another and that is through foreign keys.

Foreign keys are any column that contains a column to another table. This column would be the foreign key. Splitting tables therefore make our data more organized and joining them would make it easy to get the right data

1. Relationships
2. One to Many

1 to many relationship is used to describe the relationship where 1 row in a SQL table can be linked to many rows in another table.

1. Many to Many

Many to many refers to the idea where many rows in 1 table can be linked to many rows in another table.

1. Example

How to implement Many To Many

If you have a table called people storing first name, last name and id

Another table called locations containing id, destination and airport names

Another table called flights, which contains origin, destination and id. Note that origin and destination here can be foreign keys linking to locations table

A new table called passengers which contains 2 columns, person\_id and flight\_id. It is an association table which associates rows from 1 table to rows form another table. This is called a join table.

Summary

* Table for airports (location) -> Storing location details
* Table for flights -> Storing flight details. Any flight maps to 2 different airports one for origin one for destination. Any airport might appear for multiple flights. This is an example of a 1 to many relationship. Where 1 row of airports can be seen in multiple rows of flights.
* Table of persons -> Storing passenger details. Each row is meant to represent 1 person
* Table of passengers -> Storing relationship between persons and flights i.e. tracking which passengers goes to what flight. This is called an association table as it will contain only 2 columns, person\_id and flight\_id. This is an example of a many to many relationship between person table and flight table because 1 person can take many flights.

1. JOIN

The above design in summary demonstrates why JOIN is impt as it now helps you visualize multiple tables together. Data can be held in multiple different tables and therefore the use of a single table is unintuitive. Look at the passengers table where it only contains person\_id and flight\_id.

1. Syntax

Assuming that we only have 2 tables, flights and passengers and every passenger is only linked to 1 flight.

SELECT first, origin, destination FROM flights JOIN passengers ON passengers.flight\_id = flights.id;

The output of a JOIN statement is a completely new table. To create that table we would need to first indicate the columns of the first table that we want in the new table. That would be first, origin and destination.

Next, we would need to show how these 2 tables are related to one another. In our case, the flight\_id column in passenger means the same as flights.id in flight table.

1. Types of Join

**Inner JOIN**

SQL will table 2 tables and compare them and will only match them if their columns are a match.

**Left Outer Join**

**Right Outer Join**

**Full outer join**

1. Index

Sometimes we would need to make queries of our databases more efficient because we have to lookup databases often. We use index to achieve this. An index of a database works the same way an index in the library does. For example, if library organizes books by their title alphabetically, then the titles act as their index. In SQL, index acts the same way. We have to define 1 column in the table to be an index and when we search the column via this specific index, that query will be faster.

Note: Creating an index by a column requires more memory as it is creating a unique data structure to create this. The exact reasons of how Index works are not explained in CS50 must find yourself.

Syntax

CREATE INDEX name\_index ON passengers (last) -> This means create an index called name\_index on the passengers last name.

1. SQL Injection Attacks

When dealing with databases we must be cognizant of how hostile actors can attack them. Most commonly this is via SQL injection attacks. This occurs when a form posts something and the backend uses those inputs to execute a SQL query. Hackers could try to comment out authentication via - - as that stands for commenting out things in SQL.

Best way to avoid this is to just use an ORM and let the ORM handle security like with Django.

1. Race Conditions

Basically, occurs when a multi-threaded application try to query the same database simultaneously with different threads. To solve we can either lock the database or do something else.

1. How Django use Models
2. Migrations

Django gives you the ability to model SQL tables as Python classes within the models.py file. However, when you create python classes, nothing is created within the databases yet. We still need 1 more step to change these Python classes into SQL tables. This process is called migrations.

Command for creating migrations are:

python manage.py makemigrations -> This tells Django to create instructions to implement the changes done to models.py. I.E. now there are changes to models.py please make those changes legible to SQL

Once you do this, steps for migrations can be found in migrations.py

Python manage.py migrate -> actually starts the migrations. The default without any settings is that a SQLlite3 db file is created within the models.py where changes was made.

1. Interacting with SQlite3 files

We can enter the Django shell to write Python code to interact with SQLite3 files that have been created. Alternatively, we can use a GUI like DB browser or write actual SQL commands in Python. However, Django shell contains an in-built ORM that would effectively change Python code into SQL commands using Django’s own library.

Enter Django file with command

Python manage.py shell