



Databases

Enseignant : Diviyan Kalainathan

Date : 07/06/2023

Summary

01 – SQL

02 – PostgreSQL vs SQLite

03 – NoSQL ?

04 – Object Relational Mappings (ORMs)

05 – Project

Small notes before starting

- Let's all code in Python & SQL !
- We will be using GitHub for sharing projects and slides: <https://github.com/finish-off-school/course-databases>
- Projects are to be submitted via pull requests to this repo !
- Content is taken from Dan Suciu @ University of Washington and Brad Urani and Sathish Ravikumar

Any questions: diviyan@fentech.ai / @diviyan on discord

What is this course ?

Practice with databases, reminders, optimizations

- Practice makes perfect: you need to code (a lot?) to make good code/ optimized requests
(while aiming for good practice and optimized code)
- We're going to go quickly through an SQL course (ask me directly for any questions) for a reminder
- We're going to see the specificities of PostgreSQL, and why it is better for production-level (and the limitations of SQLite)
- We're going to see rapidly what is NoSQL though MongoDB
- We're going to use SQL DBs using Python SQLAlchemy

01 – SQL

Setup the exercise DB

- Go to <https://github.com/finish-off-school/course-databases> and download the restaurants.db in the /data folder
- Execute the command in the folder:
 - ``sqlite3 restaurants.db``
- You are ready to make SQL requests !
- Using python: (Optional)
 - `import sqlite3`
 - `client = sqlite3.connect("restaurants.db")`
 - `client.execute("SELECT ... ")`

Exercice 1

SELECT

- Find the `placeID` that have a score `0` in food_rating !

useful commands :

- ``tables``
- ``schema table_name``

Exercise 2

SELECT

- Find the `placeID` that have a score `0` in food_rating !

useful commands :

- ``tables``
- ``schema table_name``

02 — SQLite vs PostgreSQL

SQLite

- Open Source RDBMS (Relational Database Management System), server-less: linked to a file
- Easy to setup: just a link to a file
- Written in C, High performance, portability Readability
- Lightweight
- Available Types: NULL, INTEGER, REAL, TEXT, BLOB

LIMITATIONS:

- Limited concurrency : 1 connection allowed in write, many in read
- Does not scale for huge applications (lots of traffic)
- No user management
- No security

PostgreSQL

- Open Source RDBMS (Relational Database Management System)
- Client-Server Model
- Written in C, complex system
- Many data
- Focus on security and production-level DB

LIMITATIONS:

- Less efficient (more memory consumption) than SQLite
- Might be complex to setup

What to choose ?

If complex project with multiple connections/users : Use Postgres !

Otherwise: use SQLite !

PostgreSQL: Some details

- **User Management is made with roles and schemas:**
 - A schema is kind of a `folder` in which you put tables
 - `role` = `user`
 - A role can be granted specific permissions to a given schema

Exercise 6

Setup a PostgreSQL DB that :

- Has 2 roles and two schemas (one for each role)
- Has one table called `user_table` in each schema
- Each role has all rights to its schema, but can only read the other (but not write into) the other schema

useful links:

- <https://www.postgresqltutorial.com/postgresql-administration/postgresql-schema/>
- <https://www.google.com>

03 — NoSQL

Exercise 7

- Setup a MongoDB and try inserting the SQLite `restaurant` table into a `restaurant` collection in a `restaurants` DB
- Launch your mongoDB with the command :
``docker run --rm -p 27017:27017 mongo:jammy``
- in python: use pymongo package: `pip install pymongo`
- `import pymongo`
`client = pymongo.MongoClient()`

04 – ORMs

Exercise 8

- Setup a SQLAlchemy on the SQLite DB `restaurants` and map the `ratings` table
- Perform the following operations:
 - Insert a few ratings
 - select ratings with all 3 ratings == 2
 - delete ratings with service_rating == 0

05 — Project

Extension to the card-game (Complexity)

- The project of this course is rather lightweight :
 - The goal is to just map the hands of the players to SQLite tables `player 1` and `player2` with an SQLAlchemy
 - Use the SQLAlchemy package to interact with the hands of the players. Query, Delete, Add !
- Requirements:
 - Due to : 20/06/2023 23:59
 - Pull request to the github repo with your code
 - Use SQLAlchemy and SQLite

Good Luck !