5-5-2024

Comparative Analysis of Database Handling for a Star Map Application

China Gunter
University of Alabama in Huntsville

Jo Leyendecker

Comparative Analysis of Database Handling for a Star Map Application

by

China Gunter and Jo Leyendecker

An Honors Capstone

submitted in partial fulfillment of the requirements

for the Honors Diploma

to

The Honors College

of

The University of Alabama in Huntsville

May 5, 2024

Honors Capstone Project Director: Dr. Dan Schrimpsher

Jodi Leyendecker

China Gunter 5/5/2024

Student (signature) Date

5/4/2024

Project Director (signature) Date

Department Chair (signature) Date

Sean Lane Digitally signed by Sean M Lane
Date: 2024.05.07 10:28:34 -05'00'

Honors College Dean (signature) Date

HONORS COLLEGE
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Honors College

Frank Franz Hall

+1 (256) 824-6450 (voice)

+1 (256) 824-7339 (fax)

honors@uah.edu

Honors Thesis Copyright Permission

This form must be signed by the student and submitted with the final manuscript.

In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

_China Gunter_____

Student Name (printed)

Student Signature

5/5/2024

Date

HONORS COLLEGE
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Honors College

Frank Franz Hall

+1 (256) 824-6450 (voice)

+1 (256) 824-7339 (fax)

honors@uah.edu

Honors Thesis Copyright Permission

This form must be signed by the student and submitted with the final manuscript.

In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

_Jo Leyendecker_____

Student Name (printed)

Joeli Leyendecker

Student Signature

5/5/2024

Date

I. Introduction

The Astronomicon, our senior software design project, is an application that uses star data to generate a map of the sky at a given time, date, and location. To design our project's backend, we primarily used Python, a popular and flexible high-level object-oriented programming language. We chose to use it partially due to its popularity in data science, since one of the most vital parts of this project was ensuring the functionality of database handling. Python has various importable libraries for interfacing with popular database handlers. For this project, the sqlite3 library for SQLite and psycopg2 library for PostgreSQL were implemented and compared. We found that the more heavy-duty PostgreSQL database was significantly more complex to set up and implement without providing any observable increase in project performance, which we expected due to the size of our project's database.

As computer scientists going into the field, we will likely have to handle data management in some form fairly often. Modern businesses are reliant on quick, effective, and reliable data management, and with thorough, organized data, businesses can generate accurate business intelligence, manage user information, track financial data, and more. Everything is reliant on file systems and data organization, so it is our responsibility to learn about and keep up with data management. Our project used and recorded a large amount of data containing information on stars, their locations, movements, and other relevant features, so we had to implement a database to keep track of and pull from this information. As such, we turned to SQLite, "the most used database engine in the world" (*What is SQLite?*), and its Python library variants. The basic Python SQLite library sqlite3, being suitable for local databases, was our first choice for the scope of our original project. In addition, for comparison and learning purposes,

we chose to rewrite the project implementing psycopg2, a Python PostgreSQL library. (Stedman and Vaughan)

II. Context

Before our findings can be discussed, some context must be provided. While both libraries in question are written in C and implement SQL or SQL-adjacent code, they serve different purposes for software developers. SQLite is a more straightforward SQL implementation designed for smaller and less data-manipulation-intensive projects compared to PostgreSQL, which sacrifices ease of use for better data manipulation and thread management.

Python's sqlite3 library "provides a lightweight disk-based database" (*SQLITE3 - DB-API 2.0 interface for SQLite databases*) and is considered to be fairly simple and straightforward to use. It stores the database locally, eliminating the need for a server, and, if necessary, it can be written to be easily ported "to a larger database such as PostgreSQL or Oracle" (*SQLITE3 - DB-API 2.0 interface for SQLite databases*). Due to its simplicity and convenience, as of the 2.5x version, it comes already installed with Python (*Python database tutorial*). Thus, it was an obvious first choice for database handling in our project.

Python's psycopg2 library does for the PostgreSQL language what the sqlite3 library does for SQLite. Psycopg2 is "the most popular PostgreSQL database adapter for the Python programming language" (*PSYCOPG 2.9.9 documentation*). It was created for thread-heavy programs, specializing in thread management and manipulation surrounding database handling. Unlike SQLite, PostgreSQL stores its databases on a server that must be set up by the programmer and ported in to access or edit the database. Because of this, psycopg2 made for a potentially interesting contrast with sqlite3. (Olumide)

Both libraries, sqlite3 and psycopg2, use very similar commands and syntaxes, making converting Python code from using one to using the other a relatively painless process. Despite these similarities in wording, SQLite and PostgreSQL are designed to serve very different purposes. SQLite creates isolated database files to store the data on the user's computer, while PostgreSQL requires the programmer to host a server on which to store a logical database. SQLite is meant for small-scale projects, ideally only hosted on one machine, while PostgreSQL is designed for large, high-traffic projects that will be used on several machines and databases that will be updated and edited frequently. PostgreSQL also supports far more data types and implements more SQL commands, giving it an advantage over SQLite when used in projects that require a large, complex database with diverse data sets while making it more difficult for a programmer to pick up. As our project worked with a relatively small database, a .csv file of stars and their locations, we expected that SQLite would outperform PostgreSQL.

III. Research and Findings

Our original project was written using sqlite3, but we were curious to see if psycopg2 would perform significantly differently and how. To see which of the two libraries we selected performed better, we rewrote our database handler file using both sqlite3 and psycopg2 and compared them. Since the syntaxes of sqlite3 and psycopg2 are quite similar, the code itself did not change much between the two versions of the file. However, the setup of the actual database, though trivial in SQLite due to storing the database locally and generating a .db file in the program folders, was far more complicated with PostgreSQL. To run the program with a PostgreSQL database, a server had to be created and hosted, and a port had to be set up so that the program could access and interact with the database. This implementation took longer than expected due to PostgreSQL's comparatively worse ease of use, and this led to delays in the

development of the alternate database handler. This extra effort, while presumably useful for the larger databases and thread-heavy programs that PostgreSQL is designed for, did not pay off for the relatively small and straightforward database in our project. Implementing the two databases showed no significant difference in load time between them, despite the higher complexity of the PostgreSQL database. This wasn't unexpected; due to the nature of the original project, we had expected that sqlite3 would perform better than psycopg2, as the former was created for simple local databases such as the one we used.

IV. Conclusion

Overall, while researching multiple different ways to implement database handling was useful for our understanding of the subject, our project's performance did not tangibly benefit from using PostgreSQL as an alternative to SQLite. As previously stated, this was expected, as the simplicity of SQLite for smaller databases was what led us to using it for the project in the first place. Though it did not improve the performance of our project, it was worthwhile to learn about the basic implementation of PostgreSQL, as it will likely be relevant in more thread-heavy projects in our future careers. Both SQLite and PostgreSQL serve important, distinct purposes in database handling, and implementing them both in a project more clearly aligned with the goals of one helped spotlight the necessity of choosing the right language and library for aspects of a project.

Sources

Olumide, Shittu. “How to Use PostgreSQL in Python.” *freeCodeCamp.Org*, freeCodeCamp, 8 June 2023, www.freecodecamp.org/news/postgresql-in-python/.

“PSYCOPG 2.9.9 Documentation.” *Psycopg*, The Psycopg Team, 2021, www.psycopg.org/docs/.

“Python Database Tutorial.” *GeeksforGeeks*, GeeksforGeeks, 15 Mar. 2023, www.geeksforgeeks.org/python-database-tutorial/.

“SQLITE3 - DB-API 2.0 Interface for SQLite Databases.” *Python Documentation*, Python Software Foundation, docs.python.org/3/library/sqlite3.html. Accessed 27 Apr. 2024.

Stedman, Craig, and Jack Vaughan. “What Is Data Management and Why Is It Important?” *Data Management*, TechTarget, Dec. 2022, www.techtarget.com/searchdatamanagement/definition/data-management.

“What Is SQLite?” *SQLite*, SQLite Consortium, 15 Apr. 2024, www.sqlite.org/.