A COMPARISON OF SQLITE AND POSTGRESQL PERFORMANCE ON A SERIES OF QUERIES INVOLVING JOIN, GROUP BY, WHERE, AND HAVING CLAUSES

Nathan Akerhielm
3 December 2023

I. Abstract

The aim of this paper is to compare PostgreSQL and SQLite utilizing a series of SQL queries involving JOIN, GROUP BY, WHERE, and HAVING clauses. A thorough literature review yielded comparisons between the two systems, but not a side-by-side experiment involving queries. Therefore, the work in this paper is an important step to better understand performance. A benchmark study was conducted on a set of small, medium, and large-sized databases, which were tested with simple, moderate, and complex queries. Two-sampled t-tests were performed to compare mean runtimes and boxplots aided in the comparison between the two systems.

II. Introduction

Businesses use relational database systems to both organize and structure their data, and to gain valuable information to generate reports and forecasts. These firms seek to maximize their profits and efficiency, and one manner in which they can do so is through efficient database performance. Two relational databases that are popular within the business world are SQLite and PostgreSQL. SQLite was created in 2000 with the goal of allowing "the program to operate without having to install a database management system, but rather use a standalone, embedded database" (Farmer 2021). PostgreSQL, on the other hand, is older, as it was developed in 1985, with the goal creating of a database that used the fewest features needed to support the most common data types. PostgreSQL is server based, which means it communicates by sending and receiving data in a client-server model using TCP/IP, whereas SQLite is server-less, allowing any process that accesses the database to read and write to the database disk file directly (Farmer 2021).

In terms of database performance, the simplest measure is query response time (i.e. how long it takes the relational database to execute a query) (Nichter 2022). Timing starts when PostgreSQL or SQLite receive the query and ends when it has sent the results to the user. This study will determine superior performance between the two databases systems based on shortest average query response time.

III. Literature Review

The literature review yielded several papers that examine the performance of relational database systems. One such paper by Wodyk and Skublewska-Paszkowska (2020) studied MySQL, SQL Server, and PostgreSQL using a web application framework. They worked with datasets with anywhere from 1 to 20,000 records. Their main finding was that in databases with up to 1,000 records, SQLite performed the better than PostgreSQL, but PostgreSQL performed faster on the larger datasets.

Furthermore, Drake and Ostezer (2022) provided a high-level overview of PostgreSQL, MySQL, and SQLite. They list the advantages of SQLite to be its user-friendliness and portability while its drawbacks are that it only allows one write operation to occur at a time, and it doesn't work well with databases over 1TB. PostgreSQL's advantages are its compatibility with other programming languages and concurrency, but its drawbacks are complex replication and its speed is slower than MySQL (Ostezer and Drake 2022).

IV. Methods

This paper seeks to build upon the aforementioned researchers' work through a benchmark study in which database size and query complexity were the independent variables and runtime in milliseconds was the dependent variable. The database sizes ranged from small (1,000

records), medium (10,000), and large (100,000), and the query complexity similarly varied from simple (SELECT *), moderate (JOIN, GROUP BY, and WHERE clauses), and complex (JOIN, GROUP BY, HAVING, WHERE, and ORDER BY clauses). The two sample databases, Customers and Orders, were created using the Faker library in Python, and an example of records from each are found in the Tables 1 and 2, respectively. The columns of the Customers are: Customer_ID, Order_ID, Customer_Name, Address, Email, Age and Credit_Score. The columns of Orders are: Order_ID, Customer_ID, Date, Quantity, Price, and Sales_Amount.

In order to determine the number of trials (n) to run for each study, the following formula was utilized (Allen 2011): $$n = 2 * \frac{t_{1-\frac{\alpha}{2}} - t_{\beta}}{(\frac{\mu_1 - \mu_2}{\sigma})^2}$$

In the formula, ß is the probability of Type II error, α is the probability of Type I error , $t_{1-\alpha/2}$ is the t-test statistic for the desired level of Type I error, $t_{ß}$ is the t-test statistic for the desired level of Type II error, σ is the population standard deviation , and $\mu_1 - \mu_2$ is the difference between the two population means. The standard values for α and ß of 0.05 and 0.80, respectively, were utilized. Using R, we found that that $t_{1-\alpha/2}$=12.71 and $t_{ß}$=1.38. Furthermore, since the population standard deviation is unknown, the commonly accepted formula of σ=Range/4 was used. Letting the Wodyk study represent the closest study we have to be representative of the population, we find that Range=3. Finally, $\mu_1 - \mu_2$ =0.25 using the Wodyk study. Plugging in these values we obtain the following $2*(12.71-1.38)/(.25/.75)^2$=204 trials.

After completion of the trials, a two-sample t-test was conducted to compare the difference in mean runtimes. Since nine experiments were conducted instead of just one, a correction was required. The Bonferroni Correction is the process by which the desired level of alpha α is adjusted for a family of statistical tests so that we can properly determine whether or not to reject

the null hypothesis (Frost 2023). The formula is $\boldsymbol{\alpha_{new}} = \alpha_{original} / n$, where $\alpha_{original}$ is the original α and n is the number of experiments, so we compute $\boldsymbol{\alpha_{new}} = 0.05/9 = 0.006$.

V. Results

Overall, the boxplot in Figure 1 shows that PostgreSQL performs faster than SQLite, and Table 3 shows PostgreSQL has a lower mean run time by about 27 milliseconds. Clearly, as database size and complexity increase from small to large, and simple to complex, runtimes increase, as seen in Figures 2 and 3 and respectively. The boxplots in Figure 4 indicate that SQLite performed faster on the simple queries whereas PostgreSQL was superior on moderate and complex queries. The graphical interpretation is confirmed by the results of the t-test in Table 4, as we seen statistically significant results in all nine tests.

This finding is due to the fact that SQLite has a simple design and consequently can handle simple operations quickly, but SQLite can only perform one write operation at a time whereas PostgreSQL can handle multiple simultaneous operations. Thus, when operations increase in complexity, SQLite operates much slower than PostgreSQL (Farmer 2021).

VI. Conclusions

The results indicate that individuals should SQLite for simple queries, but otherwise PostgreSQL performs best. This paper helps the field by providing a side-by-side runtime comparison of two of the most population relational database systems. In future work, complexity in queries can include subqueries and common table expressions to further validate the results. Furthermore, replication of this study can enhance its legitimacy. Overall, given that businesses professionals require more than simple queries, they should use PostgreSQL in order to maximize efficiency and reduce costs.

Appendix

	Customer_ID	Customer_Name	Address	Email	Age	Credit_Score
0	100000	Melissa Miranda	3732 Ramos Estates\nPort Brookeville, AR 73899	victor88@example.com	85	454
1	100001	Debbie Reid	81378 Kathleen Roads\nSummerhaven, TX 93807	millercole@example.com	57	566
2	100002	Shelby Berry	40843 Martin Street\nBrendamouth, MD 22238	michaelpratt@example.net	82	403
3	100003	Jason Chaney	9464 Hernandez Via\nMurrayland, DC 32475	amythomas@example.org	41	635
4	100004	Francis Dodson	9011 Cynthia Glen\nWest Laurie, FL 36582	denniscurtis@example.com	73	473

Table 1: Customers Dataset First Five Rows

	Order_ID	Customer_ID	Date	Quantity	Price	Sales_Amount
0	DACD7042	258984	2019-01-26	12	1197.75	14373.00
1	AAEC1887	414831	2020-12-03	6	203.22	1219.32
2	EAEC4494	265445	2018-03-26	8	1016.04	8128.32
3	CBCE4882	399305	2023-09-11	2	1927.07	3854.14
4	AAED2789	569052	2023-01-25	2	55.47	110.94

Table 2: Orders Dataset First Five Rows

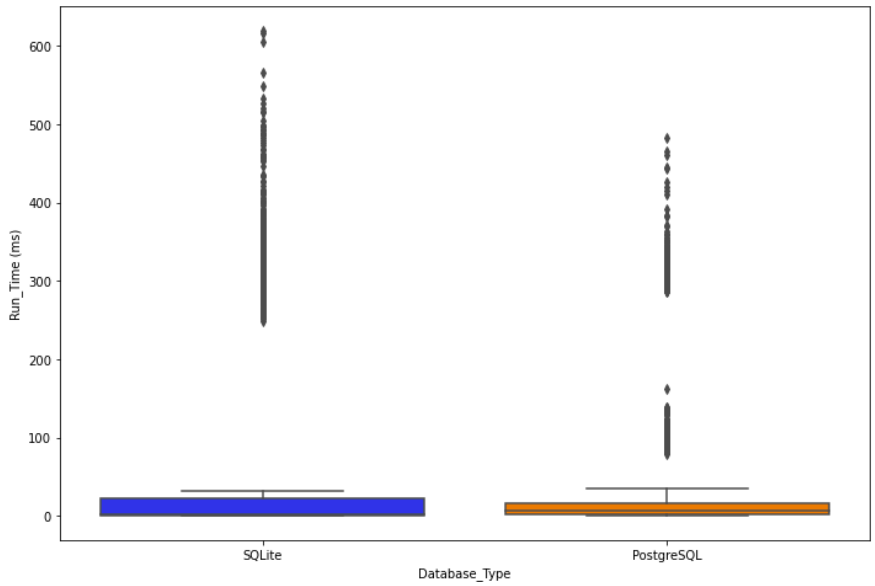

Figure 1: SQLite vs. PostgreSQL Runtime

	Database_Type	count	mean	std	min	25%	50%	75%	max
0	PostgreSQL	1836.0	51.771222	100.973046	-0.033	1.51275	6.2805	16.87025	482.564
1	SQLite	1836.0	78.456675	139.598069	0.020	0.03200	2.0970	22.42850	619.209

Table 3: Summary Statistics of PostgreSQL vs. SQLite Runtime

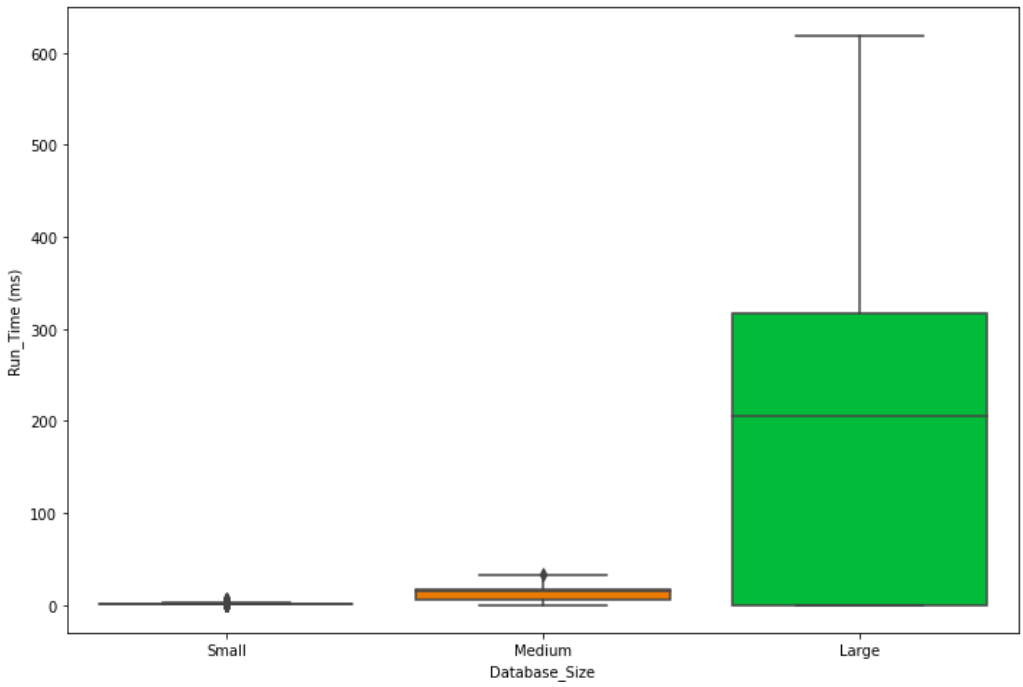

Figure 2: Runtime by Database Size

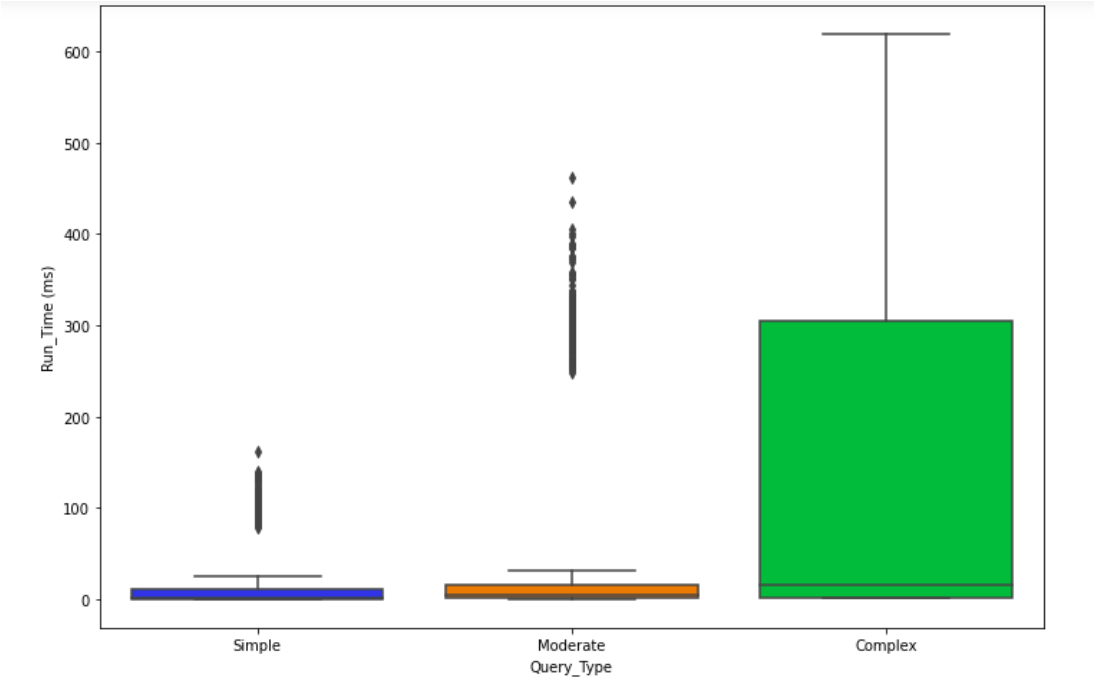

Figure 3: Runtimes by Query Type

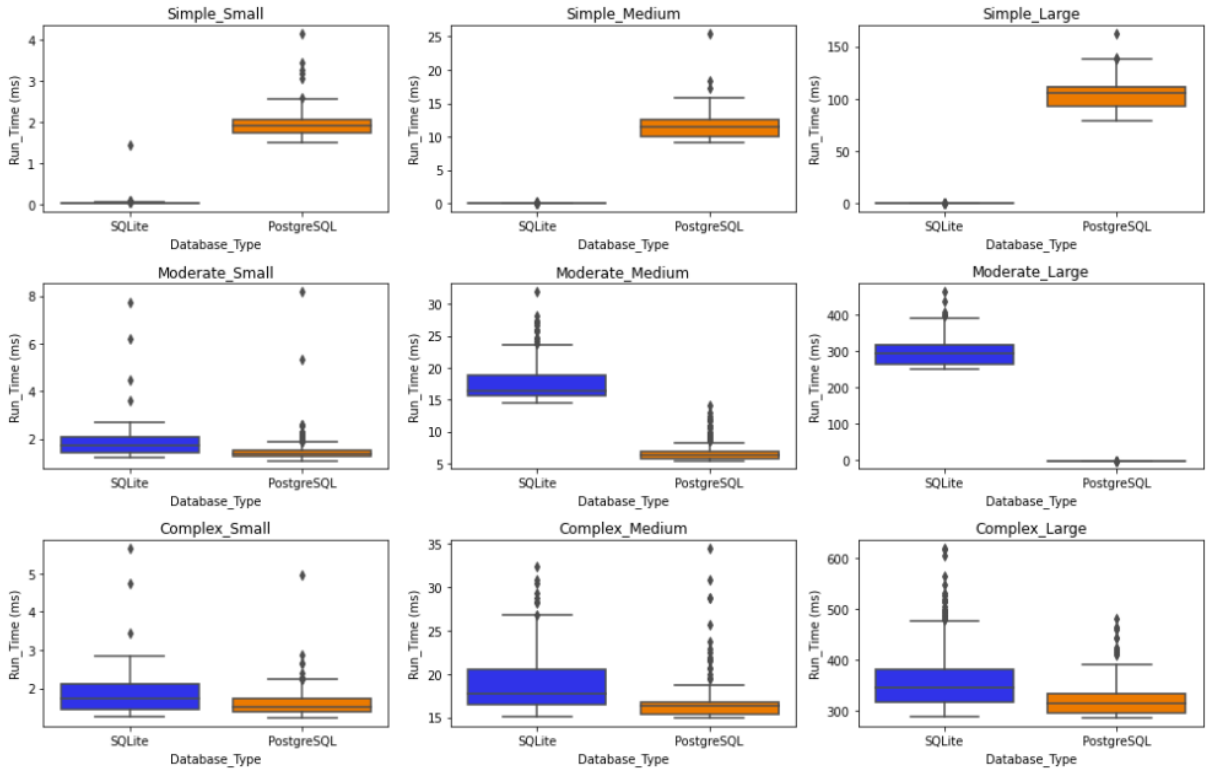

Figure 4: Runtime by Query Type, Database Type, and Database Size

	Experiment_Type	T_Test_Statistics	Unadjusted_P_Values	Statistically_Significant_Difference
0	Simple_Small	80.883671	1.651087e-176	True
1	Simple_Medium	85.817429	1.650979e-161	True
2	Simple_Large	113.538950	1.197725e-185	True
3	Moderate_Small	-5.098551	5.275563e-07	True
4	Moderate_Medium	-44.799409	6.085582e-134	True
5	Moderate_Large	-114.479068	2.305922e-186	True
6	Complex_Small	-4.261941	2.575904e-05	True
7	Complex_Medium	-7.246656	2.533745e-12	True
8	Complex_Large	-8.595113	4.886729e-16	True

Table 4: Two-Sample T-Test Results

References

Allen, John C. “Sample Size Calculation for Two Independent Groups: A Useful Rule of Thumb.” *Proceedings of Singapore Healthcare* 20, no. 2 (2011): 138–40. https://doi.org/10.1177/201010581102000213.

Farmer, Kyle. “Databases: Sqlite and PostgreSQL.” Medium, March 23, 2021. https://medium.com/swlh/databases-sqlite-and-postgresql-7ad8aaec82b9.

Frost, Jim. “What Is the Bonferroni Correction and How to Use It.” Statistics By Jim, July 17, 2023. https://statisticsbyjim.com/hypothesis-testing/bonferroni-correction/.

Nichter, Daniel. *Efficient MySQL performance: Best practices and techniques*. S.l.: O’REILLY MEDIA, 2022.

Ostezer, and Mark Drake. “SQLite vs MySQL VS PostgreSQL: A Comparison of Relational Database Management Systems.” DigitalOcean, March 9, 2022. https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems.

Wodyk, Rafał, and Maria Skublewska-Paszkowska. 2020. “Performance Comparison of Relational Databases SQL Server, MySQL and PostgreSQL Using a Web Application and the Laravel Framework.” Journal of Computer Sciences Institute 17: 358–64. https://doi.org/10.35784/jcsi.2279