ASSIGNMENT 4: PERFORMANCE ANALYSIS OF PSYCOPG2 AND RPOSTGRES WITH MONTE CARLO SIMULATIONS

Michael Mistarz and Alex McCorriston
MSDS 460: Decision Analytics
Northwestern University
Dr. Thomas Miller
May 19, 2024

1.0 Abstract

This study investigates the performance differences between Python's psycopg2 connector and R's RPostgres connector for interacting with PostgreSQL. Using Monte Carlo simulations, we analyzed processing times across CRUD (Create, Read, Update, Delete) operations. We created a PostgreSQL database and populated it with realistic synthetic data using the faker library in Python and the charlatan library in R, along with random data from normal and uniform probability distributions. Each simulation involved 200 iterations of CRUD operations on randomly sampled data. Results show that Python significantly outperforms R in all CRUD operations and data types, as confirmed by nonparametric statistical tests. These findings provide valuable insights for companies selecting an optimal backend programming language for efficient PostgreSQL database management. However, the study's limitations include focusing only on Python and R and using synthetic data, which may not fully capture real-world variability.

2.0 Introduction

Modern companies rely on robust databases to manage massive volumes of information and enable advanced analytics. PostgreSQL, an open-source relational database management system (RDBMS), is a popular choice for this purpose (White n.d.). Alongside selecting a database, companies must choose an optimal backend programming language for efficient server-side data management and processing. PostgreSQL supports various languages such as Python, C++, Java, and more (ScaleGrid 2019). This study investigates the performance differences between two database connectors, psycopg2 in Python and RPostgres in R, when interacting with PostgreSQL. Python and R are widely used in data science and analytics, making their performance in database interactions particularly relevant. Using Monte Carlo simulations, we analyze processing times for CRUD operations to provide insights for selecting the most suitable backend language based on processing speed performance.

3.0 Literature Review

Existing research typically compares the processing times of different database systems rather than database connectors. For example, Truică et al. (2015) found that NoSQL databases (MongoDB, CouchDB, Couchbase) generally outperform RDBMS (Microsoft SQL Server, MySQL, PostgreSQL) in CRUD operations. Another study by Amindotb (2021) compared MySQL, PostgreSQL, MongoDB, and ArangoDB, finding that PostgreSQL excels in updating and deleting operations, while MongoDB is more efficient for creating and reading operations. Finally, a benchmark study that did directly compare database connectors, psycopg2 and psycopg3, found that psycopg3 generally outperformed psycopg2, especially in create and read operations (Tariq 2024). Given this scarcity of studies on PostgreSQL database connectors, this study aims to contribute to this body of research by thoroughly comparing psycopg2 and RPostgres. Analyzing these connectors can provide valuable insights for companies that use Python and R environments, aiding them in selecting the optimal backend language for their PostgreSQL projects.

4.0 Methods

This study adopts a 2x4 factorial design, systematically varying two independent variables: the choice of programming language (R with RPostgres and Python with psycopg2) and the type of CRUD operation. Our initial step was to establish a PostgreSQL database named “ExCompany”, comprising a single table, “employees", with eight columns: employee_id (primary key), first_name, last_name, age, rating, json_contact_info, bjson_contact_info, and address. These columns include integer, text, float, JSON, JSONB, and geometry data types. Next, we connected to our database using psycopg2 for Python and RPostgres for R. To populate the database, we developed a function that used the faker library in Python and the charlatan library in R to generate realistic fake data for the first_name and last_name columns. For employee_id, we ensured uniqueness by seeding a random 9-digit number between 100000000 and 99999999 for each row. Additionally, age and rating were sampled from normal distributions with means of 35 and 3, and standard deviations of 10 and 1, respectively. Address data was drawn from uniform distributions within Chicago’s latitude and

longitude bounds. The json_contact_info and bjson_contact_info columns were populated with email addresses in the format "first_name.last_name@company.com" and phone numbers with Chicago's area code (312), 555, followed by a random 4-digit extension.

We then proceeded to create functions for inserting, reading, updating, and deleting records. A total of 28 CRUD functions were developed, encompassing a set for each data type and a set for querying the full dataset. To conduct Monte Carlo simulations for the insert queries, we generated 2,000 rows of fake data, randomly sampling 200 rows per iteration, executing the query, and repeating the process for 200 iterations. For read, update, and delete queries, we initially inserted 2,000 rows of fake data into the "employees" table. Subsequently, we randomly selected 200 rows from this dataset to execute our queries, repeating this process over 200 simulations. We truncated the table for each insert query to keep the number of rows consistent. For the delete queries, we rolled back each execution to keep 2,000 rows of data in the table. The resulting data from each simulation were aggregated into an Excel file for subsequent analysis.

5.0 Results

To analyze the resulting data, we used histograms to visualize the processing times for the full dataset and bar charts to depict the median processing times for each data type. These visual analyses were followed by nonparametric tests to determine statistical significance. The histograms, in Figures 1 and 2 of the Appendix, provide a visual comparison of the distribution of CRUD operation times for the full data for both Python and R. Python consistently demonstrated narrower distributions with lower processing times across all CRUD operations compared to R. This observation was further supported by creating bar charts of median CRUD processing times, in Figure 3 of the Appendix, for the full data and for each individual data type. Python clearly outperformed R in all categories.

To statistically validate these observations, we initially conducted a two-way ANOVA to examine the effects of programming language and CRUD operation, as well as their interaction on

processing time (Soetewey 2020). However, due to non-normal residuals, which violate an assumption of ANOVA testing, nonparametric tests were conducted instead for further analysis (Soetewey 2020; Soetewey 2022). The Mann-Whitney U test confirmed a significant difference in processing times between Python and R ($W = 44792$, $p < 0.05$), with Python being faster. The Kruskal-Wallis test revealed significant differences across CRUD operations ($\chi^2(3) = 245.11$, $p < 0.05$) and data types ($\chi^2(6) = 526.4$, $p < 0.05$), indicating that both the type of CRUD operation and the data type significantly influence processing times. Further analysis with Dunn's test identified significant pairwise differences among a majority of the CRUD operations and data types. However, there were no significant differences between read and update operations and among certain data type pairs, such as BJSON vs. float, geometry vs. integer, and float vs. text. These insights can inform database design and optimization, highlighting areas where performance is consistent and allowing for targeted improvements where significant differences exist.

6.0 Conclusion

Our analysis of CRUD operation processing times using Monte Carlo simulations shows that Python with the psycopg2 connector significantly outperforms R with the RPostgres connector when interacting with PostgreSQL across all CRUD operations and data types, with lower processing times confirmed by statistical validation through nonparametric tests. These findings highlight the crucial role of programming language choice, CRUD operation type, and data type in determining processing efficiency. This provides valuable insights for companies in selecting an optimal backend programming language for efficient database management and processing. However, the study's limitations include focusing only on Python and R and using synthetic data, which may not fully capture real-world variability. Future research could broaden the scope to include more languages, connectors, databases, real-world data, and different statistical tests to uncover additional insights.

Appendix

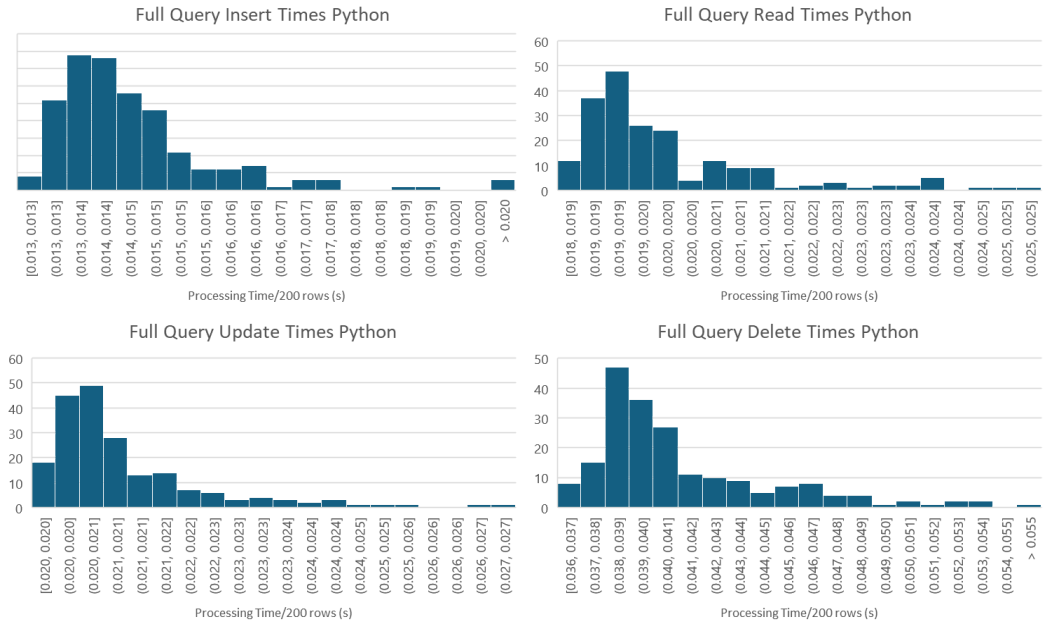

Figure 1. Full Query CRUD Operation Times Histograms for Python During Monte Carlo Simulation (200 Iterations, 200 Rows Sampled from 2,000 Rows of Fake Generated Data).

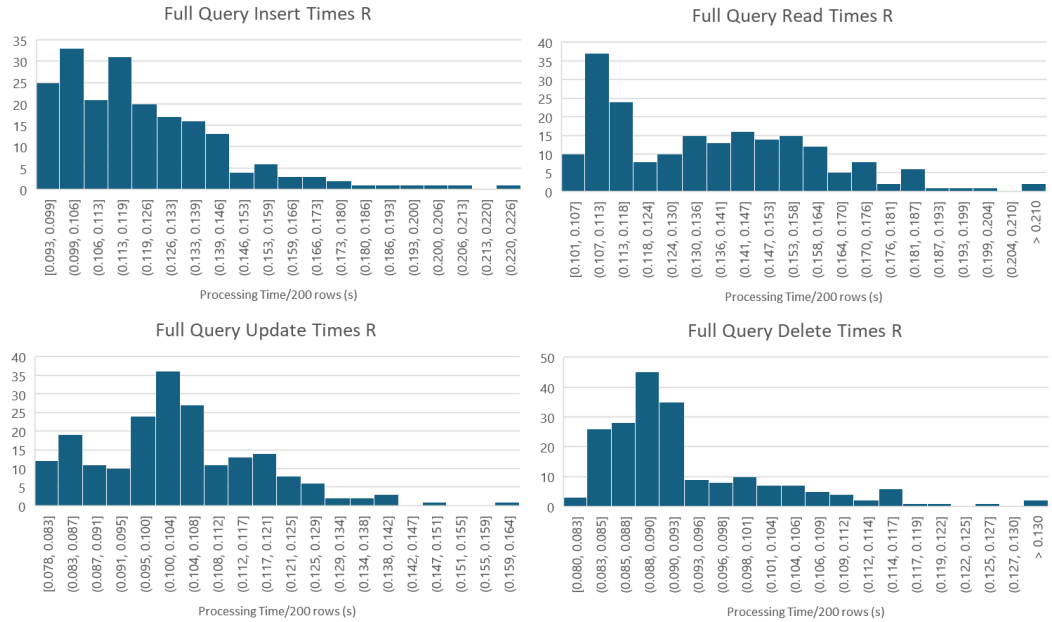

Figure 2. Full Query CRUD Operation Times Histograms for R During Monte Carlo Simulation (200 Iterations, 200 Rows Sampled from 2,000 Rows of Fake Generated Data).

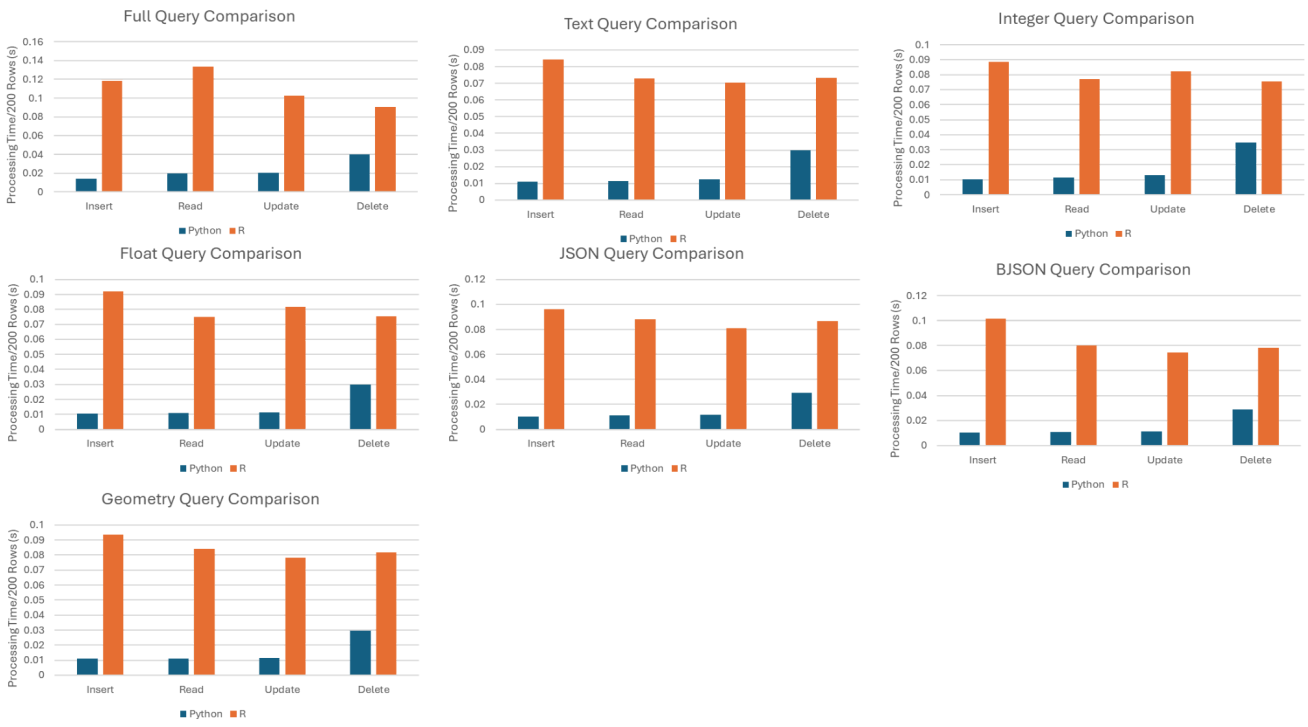

Figure 3. Median Processing Times for Various Data Types Across CRUD Operations for Python and R During Monte Carlo Simulation (200 Iterations, 200 Rows Sampled from 2,000 Rows of Generated Data).

test	W	p.value
Mann-Whitney U Test	44792	0

test	chi.squared	df	p.value
Kruskal-Wallis Test for Operation	245.1113	3	7.467062e-53

test	chi.squared	df	p.value
Kruskal-Wallis Test for Data Type	526.3994	6	1.724527e-110

Figure 4. Non-Parametric Test Results: Mann-Whitney U Test for Processing Time Differences Between Python and R (top), and Kruskal-Wallis Test for Processing Time Differences Across CRUD Operations (middle) and Data Types (bottom).

Comparison	Z	P.unadj	P.adj
create - delete	-9.685776	3.465649e-22	1.386260e-21
create - read	4.349540	1.364233e-05	4.092700e-05
delete - read	14.035317	9.477224e-45	5.686335e-44
create - update	3.295985	9.807716e-04	1.961543e-03
delete - update	12.981762	1.552675e-38	7.763375e-38
read - update	-1.053555	2.920868e-01	2.920868e-01

Comparison	Z	P.unadj	P.adj
bjson - float	-0.3204369	7.486371e-01	1.000000e+00
bjson - full	-18.3151353	6.267093e-75	1.253419e-73
float - full	-17.9946984	2.143860e-72	4.073334e-71
bjson - geometry	-3.4633583	5.334775e-04	5.334775e-03
float - geometry	-3.1429214	1.672708e-03	1.338166e-02
full - geometry	14.8517770	6.774867e-50	1.151727e-48
bjson - integer	-3.2163946	1.298122e-03	1.168310e-02
float - integer	-2.8959577	3.780034e-03	2.646024e-02
full - integer	15.0987407	1.650555e-51	2.970999e-50
geometry - integer	0.2469637	8.049363e-01	8.049363e-01
bjson - json	-4.6729558	2.968957e-06	3.859644e-05
float - json	-4.3525189	1.345823e-05	1.480405e-04
full - json	13.6421795	2.247419e-42	3.595870e-41
geometry - json	-1.2095975	2.264334e-01	6.793001e-01
integer - json	-1.4565612	1.452375e-01	7.261877e-01
bjson - text	1.2404581	2.148060e-01	8.592240e-01
float - text	1.5608950	1.185485e-01	7.112912e-01
full - text	19.5555934	3.696792e-85	7.763262e-84
geometry - text	4.7038164	2.553429e-06	3.574801e-05
integer - text	4.4568527	8.317172e-06	9.980607e-05
json - text	5.9134139	3.350887e-09	5.026331e-08

Figure 5. Dunn's Test Results for Pairwise Comparisons of CRUD Operations and Data Types, Highlighting Significant Differences in Processing Times.

References

Amindotb. 2021. “Databases Benchmark.” Medium. https://medium.com/@amindotb/databases-benchmark-b5b8941239f.

ScaleGrid. 2019. “Most Popular PostgreSQL Providers & Deployments in Enterprise.” https://scalegrid.io/blog/postgresql-trends-most-popular-cloud-providers-languages-vacuum-query-management-strategies-deployment-types-in-enterprise/#:~:text=Most%20Used%20Languages%20with%20PostgreSQL&text=The%20supported%20programming%20languages%20for,languages%20through%20its%20available%20extensions.

Soetewey, Antoine. 2020. "ANOVA in R." Stats and R. https://statsandr.com/blog/anova-in-r/#variable-type.

Soetewey, Antoine. 2020. "Wilcoxon test in R: how to compare 2 groups under the non-normality assumption?" Stats and R. https://statsandr.com/blog/wilcoxon-test-in-r-how-to-compare-2-groups-under-the-non-normality-assumption/.

Soetewey, Antoine. 2022. "Kruskal-Wallis test, or the nonparametric version of the ANOVA." Stats and R. https://statsandr.com/blog/kruskal-wallis-test-nonparametric-version-anova/.

Tariq, Semab. 2024. “Psycopg2 Vs Psycopg3 Performance Benchmark.” Timescale Blog. https://www.timescale.com/blog/psycopg2-vs-psycopg3-performance-benchmark/.

Truică, Ciprian-Octavian, Florin Rădulescu, Alexandru Boicea, and Ion Bucur. 2015. “Performance Evaluation for CRUD Operations in Asynchronously Replicated Document Oriented Database.” *2015 20th International Conference on Control Systems and Computer Science*: 191-196. https://doi.org/10.1109/cscs.2015.32.

White, Emma. n.d. “The Most Popular Databases in 2024.” BairesDev. Accessed on May 17, 2024. https://www.bairesdev.com/blog/most-popular-databases/.