JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 8, No. 3, September 2023, Pp. 1052-1060

ANALISIS PERBANDINGAN PERFORMA DATABASE DUCKDB DAN SQLITE PADA PENGOLAHAN BIG DATA

Farid A. Nugraha[1)], Yerymia A. Susetyo[2)]
1. Universitas Kristen Satya Wacana, Indonesia
2. Universitas Kristen Satya Wacana, Indonesia

Article Info
Kata Kunci: Basis Data; DuckDB; Perbandingan; SQLite;

Keywords: *Comparison; Database; DuckDB; SQLite;*

Article history:
Received 26 June 2023
Revised 10 July 2023
Accepted 24 July 2023
Available online 1 September 2023

DOI :
https://doi.org/10.29100/jipi.v8i3.4032

* Corresponding author.
Corresponding Author
E-mail address:
672019202@student.uksw.edu

ABSTRAK
Data memiliki peran sangat penting pada zaman ini karena dengan data setiap perusahaan dapat mengambil keputusan dengan lebih baik. Namun data yang ada tentunya akan semakin besar dan kompleks seiring berjalannya waktu. Akibatnya adalah waktu pengolahan data menjadi lebih lambat dan dapat menghambat proses bisnis. Pemilihan *database* yang tepat sangat penting karena dapat mempengaruhi performa suatu aplikasi. Saat ini *database* memiliki banyak jenis diantaranya yaitu DuckDB dan SQLite di mana kedua *database* tersebut adalah *database* yang tepat untuk menangani *big data*. Untuk membandingkan dua *database* tersebut tahapan-tahapan metode yang penulis gunakan yaitu identifikasi kebutuhan perangkat, persiapan *dataset*, perancangan skema pengujian, implementasi dan pengujian, dan analisis hasil. Pada penelitian ini, *query* yang diuji antara lain *insert*, *update*, *delete*, *select*, *sum*, *count*, *max*, dan *average*. Data yang digunakan merupakan data *sales* dengan jumlah 6.362.620 data. Dari pengujian yang dilakukan SQLite unggul dalam mengeksekusi *query insert*, *update* semua kolom, *delete*, dan *select*. Sementara itu, DuckDB unggul dalam mengeksekusi *query* yang menggunakan fungsi agregat dan *update* dua buah kolom. Dengan hasil tersebut dapat ditarik kesimpulan bahwa SQLite cocok digunakan untuk melakukan proses transaksi. Sedangkan DuckDB cocok digunakan untuk melakukan proses analisis.

ABSTRACT
Data has a very important role in this era because with data every company can make better decisions. However, the existing data will certainly get bigger and more complex as time goes by. The result is that data processing time becomes slower and can hamper business processes. Choosing the right database is very important because it can affect the performance of an application. Currently, there are many types of databases including DuckDB and SQLite where both databases are the proper databases to handle big data. To compare the two databases, the stages of the method that the author uses are identification of device requirements, preparation of datasets, design of test schemes, implementation and testing, and analysis of results. The queries tested include insert, update, delete, select, sum, count, max, and average. The data used is sales data with a total of 6,362,620 datas. From the tests carried out SQLite is better at executing insert, updating many columns, delete, and select queries. Meanwhile, DuckDB is better at executing queries that use the aggregate functions and update two columns. With these results, it can be concluded that SQLite is suitable for transaction processing. While DuckDB is suitable for the analysis process.

I. PENDAHULUAN

Di era seperti ini, seluruh aspek kehidupan berjalan beriringan dengan teknologi. Teknologi saat ini sudah sangat berkembang dan hadir di seluruh aspek kehidupan seperti *Internet of Things* (IOT), layanan *cloud*, *market place*, dan sebagainya. Hal ini membuat manusia sangat bergantung pada teknologi karena hampir semua kebutuhan dapat terselesaikan mulai dari pemenuhan kebutuhan sehari-hari, hiburan, dan sosialisasi [1]. Dengan semakin banyaknya orang yang menggunakan teknologi data yang tersimpan tentu akan semakin besar dan kompleks. Kumpulan data yang sangat cepat berubah, berukuran sangat besar, dan kompleks dapat diartikan sebagai *big data* [2]. Karakteristik umum pada *big data* biasa disebut dengan 3V yaitu *variety*, *volume*, dan *velocity*

[3]. *Variety* artinya data yang tersimpan memiliki format yang berbeda-beda seperti formulir data, foto, *file* dokumen, hingga video. *Volume* artinya data yang tersimpan memiliki jumlah yang sangat besar. Sedangkan *velocity* memiliki arti banyaknya data yang masuk atau berubah setiap waktunya.

Berdasarkan penelitian yang dilakukan oleh Ari Fadli dan kawan-kawan, hasil penelitian menunjukkan bahwa semakin besar data yang tersimpan akan meningkatkan waktu eksekusi. Meningkatnya waktu eksekusi yang dijalankan terjadi baik pada proses *create, read, update,* maupun *delete* [4]. Dalam proses bisnis, waktu adalah hal yang sangat penting dalam pengambilan keputusan karena semakin banyak waktu yang terpakai untuk suatu pekerjaan dapat menaikkan biaya perusahaan menjadi lebih mahal [5]. Pemilihan dan pemakaian *database* juga sangat penting untuk proses bisnis karena *database* dapat berfungsi sebagai alat untuk memudahkan perusahaan atau organisasi dalam proses pengambilan keputusan [6] Selain itu, pemilihan *database* juga berpengaruh besar pada performa dan dapat mempercepat pemanfaatan kembali data-data yang sebelumnya telah tersimpan [7-8].

Database yang cocok digunakan untuk mengatasi *big data* di antaranya yaitu DuckDB dan SQLite. DuckDB merupakan *embeddable database* yang berorientasi kolom (*columnar*). DuckDB efisien untuk mengolah *big data* karena berorientasi kolom yang artinya DuckDB membaca data dari kolom ke kolom. Sementara itu, SQLite merupakan *embeddable database* relasional yang berorientasi baris. Salah satu fitur SQLite yaitu dapat menggabungkan tabel-tabel pada *database* berbeda karena SQLite bisa mengakses *file database* secara bersamaan [9].

Seiring berjalannya waktu data yang tersimpan di dalam *database* setiap perusahaan pasti akan terus membesar dan menjadi kompleks. Akibatnya adalah waktu pengolahan data menjadi lebih lambat dan dapat menghambat proses bisnis. Ada banyak faktor yang dapat meningkatkan performa aplikasi salah satunya yaitu pemilihan *database*. Saat ini *database* memiliki banyak jenis sesuai dengan kebutuhan sehingga pada penelitian ini penulis akan melakukan analisis perbandingan performa dari DuckDB dan SQLite. Kedua *database* tersebut memiliki perbedaan dalam mengolah data karena SQLite merupakan *row oriented* dan DuckDB merupakan *column oriented.* Dalam penelitian ini penulis membandingkan fitur, cara kerja, dan performa kedua database tersebut dalam mengeksekusi berbagai *query* pada kasus *big data* dengan menggunakan jupyter notebook. Untuk membandingkan dua *database* tersebut tahapan-tahapan metode yang penulis gunakan yaitu identifikasi kebutuhan perangkat, persiapan *dataset*, perancangan skema pengujian, implementasi dan pengujian, dan analisis hasil.

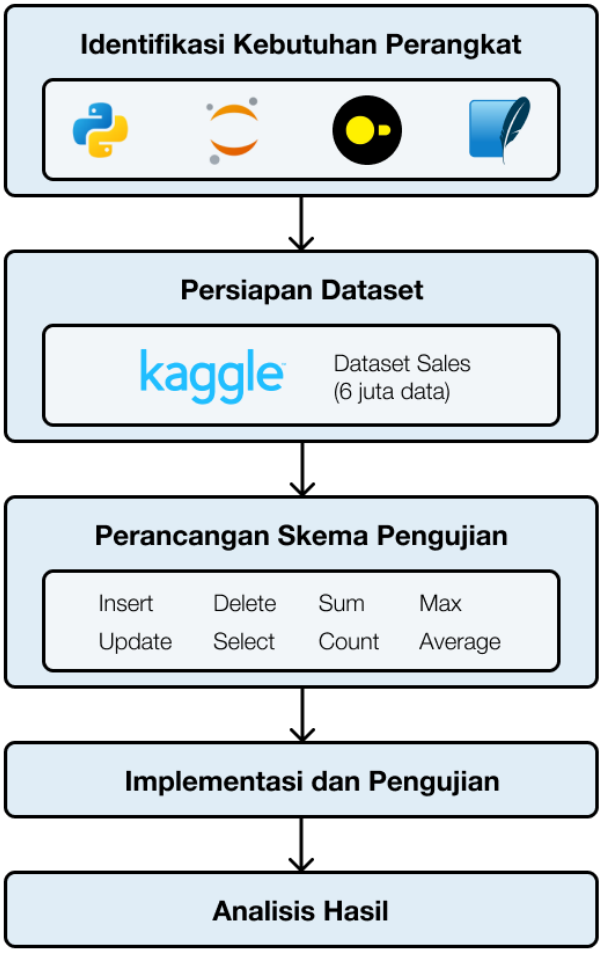
Penelitian mengenai perbandingan performa antara dua *database* sebelumnya pernah dilakukan oleh Dibyo dan kawan-kawan dengan judul "Perbandingan Performa Antara MongoDB dan MariaDB". Pada penelitian tersebut *database* yang dibandingkan yaitu SQL *database* MariaDB dengan NoSQL *database* MongoDB. Tujuan dari penelitian tersebut adalah membuktikan *database* mana yang memiliki performa yang lebih baik. Dalam pengujiannya peneliti menggunakan sebuah tabel tunggal yang jumlah datanya beragam dari 100 hingga 10.000 data. Hasil yang didapat menunjukkan bahwa *response time* pada NoSQL *database* unggul pada proses penambahan data, penghapusan data, pengubahan data, dan menampilkan data dengan satu kriteria yang mudah. Sementara itu, *response time* SQL *database* lebih unggul saat menampilkan data dengan banyak kriteria yang kompleks [10].

Penelitian lain mengenai analisis performa *database* berorientasi kolom sebelumnya sudah pernah dilakukan oleh Aaron dan kawan-kawan dengan judul "Implementasi ETL dan Perbandingan Performa *Column-Oriented Database* dan *Relational Database* sebagai *Data Warehouse*". Pada penelitian tersebut peneliti mengimplementasikan *non-relational database* sebagai *data warehouse* dengan menggunakan *relational data warehouse* sebagai sumber datanya. Dalam penerapannya penetili menggunakan Cassandra yang merupakan *non-relational database* bertipe *column-oriented.* Hasil dari penelitian ini yaitu *column-oriented database* membutuhkan perhatian lebih, meskipun begitu *column-oriented database* mendapatkan hasil yang lebih baik [11].

Perbandingan dua *relational database* sebelumnya sudah pernah dilakukan pada penelitian berjudul "Studi Perbandingan Performansi Antara MySQL dan PostgreSQL" yang ditulis oleh Ardian dan Maryanah. Pada penelitian tersebut *database* yang dibandingkan yaitu MySQL dan PostgreSQL keduanya merupakan *database relational* yang datanya terstruktur. Penelitian ini dilakukan dengan menggunakan aplikasi Adminer 4.7.7 yaitu sebuah alat manajemen *database.* Dalam penelitian ini dilakukan perbandingan sebanyak tiga kali dengan masing-masing *database* berisikan data sebanyak 50.000, 100.000, dan 1.000.000 data. Hasil dari penelitian tersebut menunjukkan bahwa PostgreSQL dapat menjadi solusi yang lebih baik karena mendapatkan *response time* yang lebih cepat saat proses menampilkan seluruh data, menampilkan hasil fungsi *count*(), dan menampilkan data dari dua buah tabel menggunakan *join* [12].

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 8, No. 3, September 2023, Pp. 1052-1060

II. METODE PENELITIAN



Gambar 1. Skema Tahapan Penelitian

Terdapat lima tahapan pada penelitian ini dimulai dari identifikasi kebutuhan perangkat hingga analisis hasil. Skema tahapan penelitian dapat dilihat pada Gambar 1. Detail pada setiap tahapan penelitian adalah sebagai berikut:

A. Identifikasi Kebutuhan Perangkat

Pada tahap ini dilakukan identifikasi perangkat lunak yang dibutuhkan untuk melakukan pengujian sesuai dengan spesifikasi yang dibutuhkan dari DuckDB dan SQLite. Teknologi yang digunakan antara lain:

1) Python sebagai bahasa pemrograman yang digunakan untuk pengujian *database* dan memvisualisasikan hasil pengujian dengan *library* matplotlib.
2) Jupyter Notebook sebagai *tool* untuk menulis kode, melakukan perhitungan, dan visualisasi data.
3) DuckDB sebagai *database* yang berjenis *column oriented.* DuckDB merupakan *database* yang mirip dengan SQLite di mana keduanya didesain sebagai *database* yang *embedded.* DuckDB merupakan *database* yang berorientasi kolom. Artinya DuckDB membaca data dari kolom ke kolom. Hal ini membuat DuckDB lebih cocok untuk digunakan dalam analisis data yang sangat banyak karena biasanya dalam analisis hanya membutuhkan beberapa kolom saja. Fitur-fitur yang disediakan oleh DuckDB antara lain mudah untuk di-*install*, tidak memerlukan manajemen *server*, format penyimpanan pada satu *file*, dan proses analisis yang cepat.
4) SQLite sebagai *database* yang berjenis *row oriented.* SQLite adalah sistem manajemen *database* yang sangat banyak digunakan terutama pada *smartphone* android. SQLite tergolong *database* yang sangat ringan karena hanya membutuhkan sebagian kecil memori namun dapat memberikan performa yang sangat cepat [13]. SQLite merupakan sebuah *embedded database* yang artinya SQLite tertanam langsung pada aplikasi sehingga cocok digunakan untuk penyimpanan *end user.* Fitur-fitur yang disediakan oleh SQLite antara lain ringan, tidak memerlukan proses instalasi, bersifat *open source,* dan dapat digunakan pada berbagai macam *platform.*

B. Persiapan Dataset

Dataset yang digunakan oleh penulis berasal dari sebuah *website* penyedia data yang gratis untuk digunakan. *Dataset* yang penulis gunakan berisi informasi transaksi dari sebuah perusahaan *finance.* Jumlah kolom yang

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 8, No. 3, September 2023, Pp. 1052-1060

terdapat pada *dataset* ini yaitu sembilan kolom yang terdiri dari *step, type, amount, nameOrig, oldbalanceORg, newbalanceOrig, nameDest, oldbalanceDest, newbalanceDest, isFraud*, dan *isFlaggedFraud*. *Dataset* yang penulis gunakan memiliki jumlah data sebanyak 6.362.620 data.

C. Perancangan Skema Pengujian

SQL atau *Structured Query Language* merupakan sebuah bahasa yang dapat digunakan oleh pengguna untuk berkomunikasi dengan *database.* Dengan SQL pengguna dapat melakukan manajemen data seperti membuat tabel, menambahkan data, menghapus data, membuat relasi antar tabel, dan sebagainya [14]. Perintah SQL dasar pada umumnya dibagi menjadi lima berdasarkan jenisnya yaitu DDL, DML, DQL, DCL, dan TCL. SQL juga menyediakan fungsi agregat yaitu fungsi-fungsi yang dapat membantu pengguna dalam melakukan pengolahan data pada kolom tertentu seperti COUNT, SUM, AVG, dan MAX. [15]. Skema pengujian yang penulis lakukan pada penelitian ini sebagai berikut:

TABEL I
KODE PROGRAM QUERY DATA

Kode Program

```
# query insert
INSERT INTO sales VALUES (1, 'PAYMENT', 5000, 'C123123123', '10000',
'5000', 'M32132131', 0, 0, 0, 1)

# query update
UPDATE fraud SET isFraud = 1, isFlaggedFraud = 0

UPDATE sales SET amount = 12345.67, nameOrig = 'C999888777666', oldbalance-
Org = '98765.54', newbalanceOrig = '55555.55', nameDest = 'M12345678910',
oldbalanceDest = 1.2, newbalanceDest = 2.3, isSales = 0, isFlaggedSales = 1
WHERE type = 'CASH_IN'

# query delete
DELETE FROM fraud WHERE type = 'DEBIT'

# query select
SELECT * FROM fraud

# query sum
SELECT SUM(amount) FROM fraud

# query count
SELECT COUNT(amount) FROM fraud GROUP BY type

# query max
SELECT MAX(amount) FROM fraud GROUP BY type

#query average
SELECT AVG(amount) FROM fraud WHERE type = 'CASH_OUT'
```

Tabel I berisikan kode program *query* data yang akan diuji pada SQLite dan DuckDB. *Query* yang diuji antara lain *insert, update, delete, select* semua data, *sum, count, max,* dan *average.*

D. Implementasi dan Pengujian

Pada tahapan implementasi dan pengujian dilakukan dengan menggunakan bahasa pemrograman python melalui jupyter notebook. *Database* diuji melalui komputer lokal dan keduanya diakses secara *embedded* (tanpa *server*). *Embedded database* merupakan sebuah *database* yang terintegrasi atau menempel pada aplikasi yang dikembangkan [16]. Setiap skema pengujian dilakukan masing-masing tiga kali percobaan untuk mendapatkan hasil yang lebih akurat.

E. Analisis Hasil

Pada tahap ini hasil implementasi dan pengujian dianalisis untuk menemukan kelemahan dan kelebihan pada masing-masing *database*.

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 8, No. 3, September 2023, Pp. 1052-1060

III. HASIL DAN PEMBAHASAN

TABEL II
KODE PROGRAM QUERY DATA

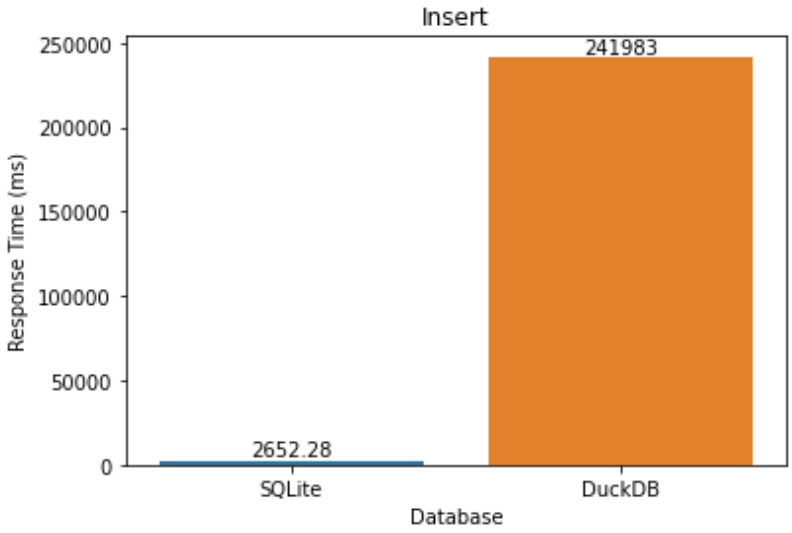
Kode Program

```
import duckdb
import sqlite3

# DuckDB
conn_duckdb = duckdb.connect('sales_duckdb.db')
conn_duckdb.execute("Kode Query")

# SQLite
conn_sqlite = sqlite3.connect('sales_sqlite.db')
cursor_sqlite = conn_sqlite.cursor()
cursor_sqlite.execute("Kode Query")
```

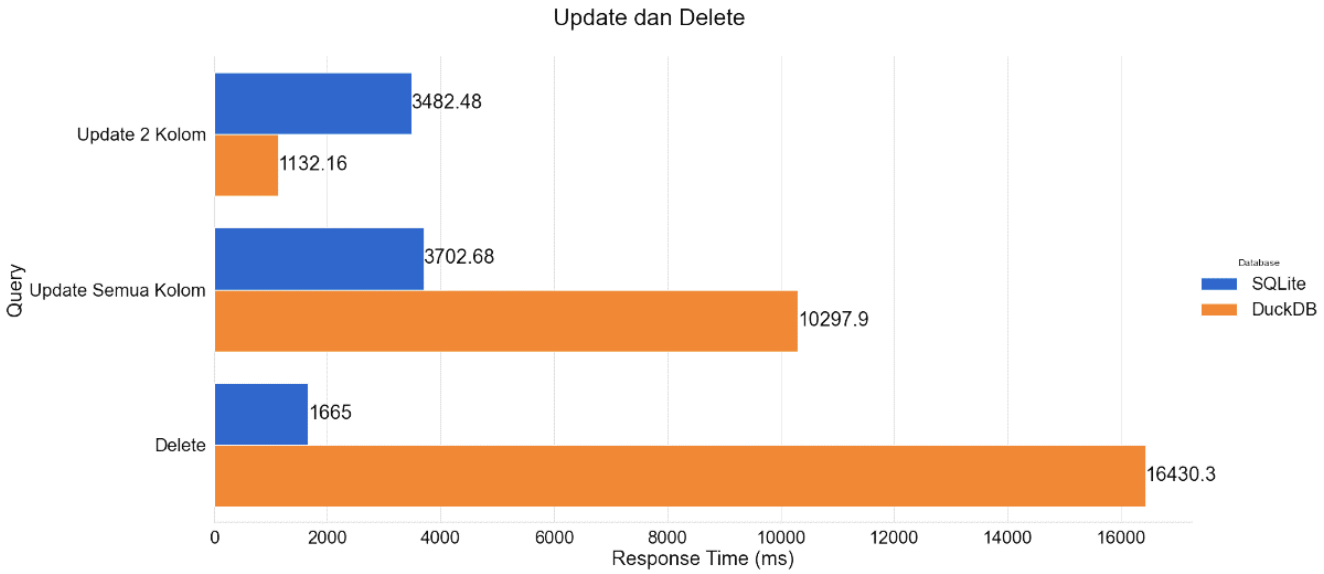
Pembuatan koneksi dari python ke SQLite dan DuckDB mudah dilakukan karena kedua *database* ini bersifat *embedded*. Pertama yang perlu dilakukan adalah *import library* DuckDB dan SQLite, selanjutnya buat koneksi dengan memanggil *method* connect(). Untuk mengeksekusi sebuah *query* pada SQLite perlu membuat sebuah *cursor* terlebih dahulu sedangkan DuckDB tidak. Kode program untuk melakukan koneksi dapat dilihat pada Tabel II.



Gambar 2. Hasil Pengujian *Insert Data*

Pada pengujian pertama dilakukan untuk mengetahui *response time* masing-masing *database* dalam mengeksekusi *query insert* sebanyak 100.000 data. Hasil pengujian dapat dilihat pada Gambar 2. Dari pengujian yang dilakukan didapatkan total *response time* dari *database* SQLite adalah 2.652,28 ms dan DuckDB adalah 241.983 ms. Sedangkan hasil rata-rata *response time* satu kali eksekusi dari *database* SQLite yaitu 0,0265 ms dan DuckDB yaitu 2,4198 ms. Dengan hasil tersebut SQLite membutuhkan waktu lebih cepat dalam melakukan *insert* data dibandingkan DuckDB.

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 8, No. 3, September 2023, Pp. 1052-1060



Gambar 3. Hasil Pengujian *Update* dan *Delete Data*

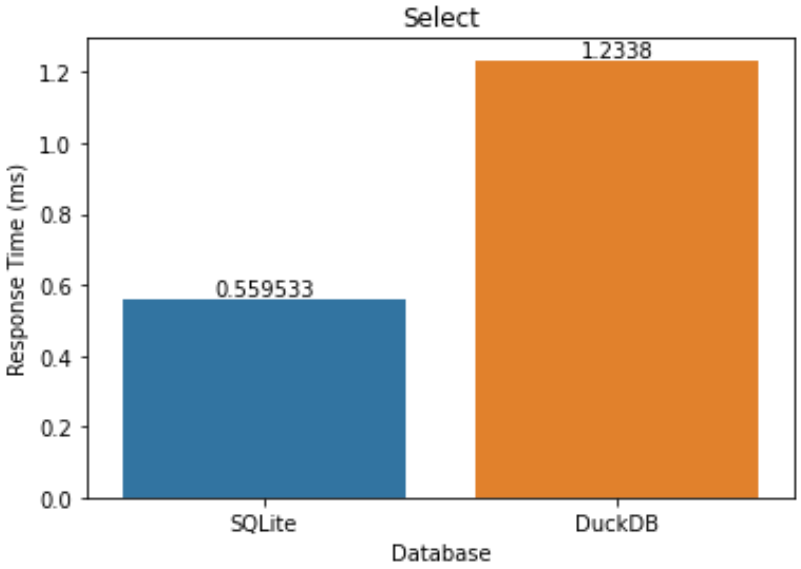
Pada pengujian selanjutnya dilakukan untuk mengetahui *response time* masing-masing *database* dalam mengeksekusi *query update*. Terdapat dua *query update* yang diuji, *query* pertama bertujuan untuk melakukan *update* pada dua buah kolom dan *query* kedua bertujuan untuk melakukan *update* pada semua kolom pada tabel. Hasil pengujian dapat dilihat pada Gambar 3. Dari tiga kali pengujian pada *query update* dua buah kolom didapatkan rata-rata *response time* dari *database* SQLite adalah 3.482,48 ms dan DuckDB adalah 1.132,16 ms. Untuk tiga kali pengujian pada *update* semua kolom didapatkan rata-rata *response time* dari *database* SQLite adalah 3.702,68 ms dan DuckDB adalah 10.297,9 ms. Dengan hasil tersebut DuckDB membutuhkan waktu lebih cepat dalam melakukan *update data* dibandingkan SQLite ketika hanya ada dua buah kolom yang di-*update*. Namun ketika semua kolom di-*update* SQLite membutuhkan waktu yang lebih cepat dibandingkan dengan DuckDB.

Pada pengujian lainnya dilakukan untuk mengetahui *response time* masing-masing *database* dalam mengeksekusi *query delete*. Hasil pengujian dapat dilihat pada Gambar 3. Dari tiga kali pengujian didapatkan rata-rata *response time* dari *database* SQLite memperoleh 1.665 ms dan DuckDB memperoleh 16.430,3 ms. Dengan hasil tersebut SQLite membutuhkan waktu lebih cepat dalam melakukan *delete data* dibandingkan DuckDB.

Berdasarkan pengujian yang telah dilakukan pada *query insert, update,* dan *delete* SQLite mendapatkan *response time* yang lebih cepat dibandingkan dengan DuckDB. SQLite mendapatkan hasil yang lebih baik karena SQLite menggunakan teknik *write ahead logging* (WAL) untuk menangangi operasi *write data*. Teknik ini memungkinkan SQLite untuk melakukan penulisan data pada *file log* secara terpisah sebelum menulis ke *database* utama, sehingga memungkinkan SQLite untuk melakukan beberapa operasi *write* secara bersamaan. Hal ini mengurangi *overhead I/O* dan meningkatkan kinerja *insert, update,* dan *delete*.

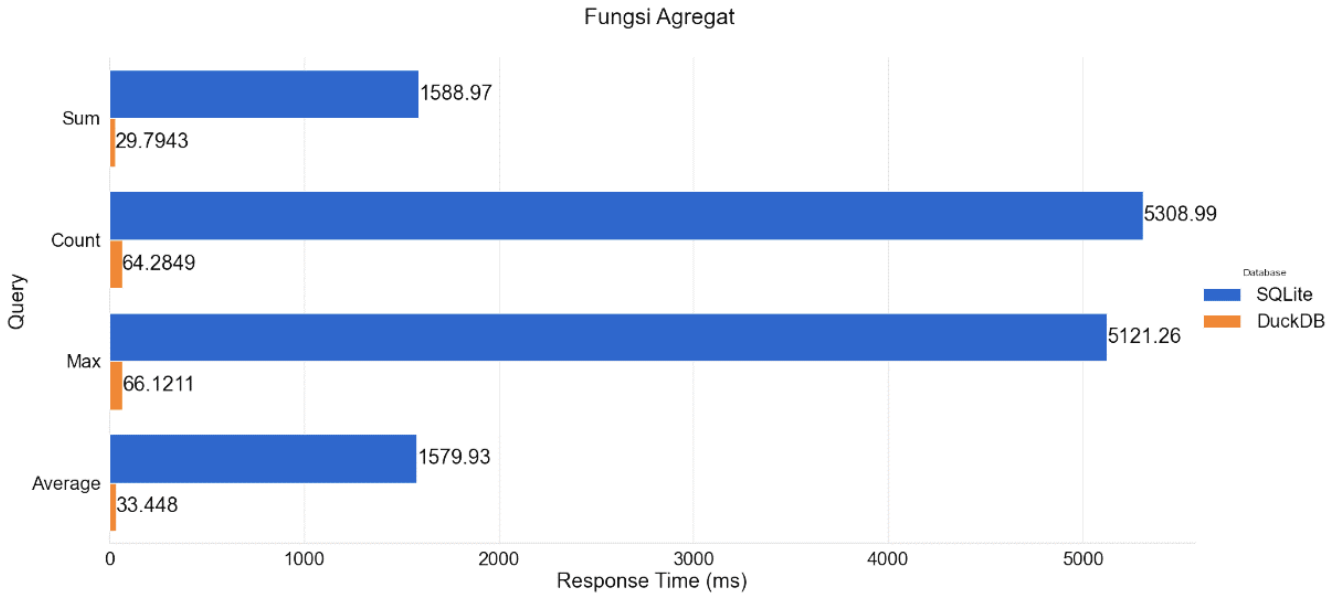
Sementara itu, DuckDB lebih lambat dalam mengeksekusi *query insert, update,* dan *delete* karena DuckDB merupakan *database column oriented.* Ketika terdapat baris data baru yang ditambahkan, setiap kolom pada baris tersebut harus ditambahkan ke blok yang sesuai. Hal tersebut membuat performa *insert* data pada DuckDB lebih lambat terutama apabila tabel memiliki banyak kolom. Masalah yang sama juga terjadi ketika melakukan *update* dan *delete data.* Seperti yang terlihat pada Gambar 3, DuckDB dapat mengeksekusi *query update* dengan lebih cepat ketika hanya dua buah kolom yang diubah. Namun ketika semua kolom diubah performa DuckDB menurun hampir 10 kali lipat. Untuk menghapus satu baris data DuckDB perlu menghapus setiap data pada semua kolom, akibatnya performa akan menurun ketika data yang disimpan sangat besar.

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 8, No. 3, September 2023, Pp. 1052-1060



Gambar 4. Hasil Pengujian *Select Data*

Pada pengujian selanjutnya dilakukan untuk mengetahui *response time* masing-masing *database* dalam mengeksekusi *query select*. Hasil pengujian dapat dilihat pada Gambar 4. Dari tiga kali pengujian didapatkan rata-rata *response time* dari *database* SQLite adalah 0,5595 ms dan DuckDB adalah 1,2338 ms. Dengan hasil tersebut SQLite membutuhkan waktu lebih cepat dalam melakukan *select* seluruh data dibandingkan DuckDB. Hasil tersebut bisa didapatkan karena SQLite menggunakan teknik *indexing* yang efektif untuk mempercepat *query select*. SQLite menggunakan indeks *B-tree* yang dikombinasikan dengan optimasi *query* seperti *query planner* dan *query optimizer*. Indeks *B-tree* memungkinkan SQLite untuk mempercepat pencarian data di dalam *database* dan memungkinkan SQLite untuk memproses *query select* dengan cepat.



Gambar 5. Hasil Pengujian Fungsi Agregat

Pada pengujian selanjutnya dilakukan untuk mengetahui *response time* masing-masing *database* dalam mengeksekusi *query* dengan fungsi agregat. Beberapa fungsi agregat yang diuji yaitu *sum*, *count*, *max*, dan *average.* Hasil pengujian dapat dilihat pada Gambar 5. Dari tiga kali pengujian *query sum* didapatkan rata-rata *response time database* SQLite adalah 1.588,97 ms dan DuckDB adalah 29,7943 ms. Pada *query count* SQLite mendapatkan 5.308,99 ms dan DuckDB mendapatkan 64,2849 ms. Pada *query max* SQLite mendapatkan 5.121,26 ms dan DuckDB mendapatkan 66,1211 ms. Pada *query average* SQLite mendapatkan 1.579,93 ms dan DuckDB mendapatkan 33,448 ms.

Dari hasil pengujian pada fungsi agregat DuckDB mendapatkan *response time* ratusan kali lebih cepat dibandingkan SQLite. Hal ini dapat terjadi karena DuckDB menggunakan *vectorized execution* yaitu proses yang

memungkinkan pengolahan data dalam jumlah besar dalam satu operasi. Selain itu, DuckDB menyimpan data dalam format *columnar* tidak seperti SQLite yang menggunakan *row oriented*. Format penyimpanan *columnar* membuat DuckDB lebih efisien dalam menjalankan fungsi agregat karena hanya memproses kolom yang dibutuhkan saja. Dengan fitur-fitur optimasi tersebut DuckDB dapat mengeksekusi fungsi agregat dengan sangat cepat dan efisien.

Penelitian mengenai performa *database* SQLite juga pernah dilakukan oleh Nikola Obradovic dan kawan-kawan. Pada penelitian tersebut dilakuan pengujian SQLite pada android dalam mengeksekusi data yang terenkripsi. Hasil yang didapatkan memiliki beberapa kesamaan dengan penelitian yang dilakukan oleh penulis di mana proses tercepat yang dilakukan oleh SQLite adalah *query select* [17]. Namun untuk proses *query* lainnya mendapatkan *response time* yang lebih lama karena perlu dilakuan proses dekripsi data terlebih dahulu. Pengujian *database* DuckDB yang penulis lakukan menunjukkan bahwa DuckDB dapat mengeksekusi *query* dengan fungsi agregat dengan cepat. Tidak hanya itu saja, pada penelitian yang dilakukan oleh Lurens Kuiper dan kawan-kawan menunjukkan bahwa DuckDB juga cepat dalam melakukan *query select* yang menggunakan *order by* yaitu *query* untuk mengurutkan data. Proses tersebut lebih cepat dieksekusi oleh DuckDB dibandingkan *database* lain seperti SQLite, HyPer, dan ClickHouse [18].

IV. KESIMPULAN DAN SARAN

Berdasarkan penelitian yang telah dilakukan *database* SQLite dan DuckDB memiliki kelebihan dan kekurangan tersendiri berdasarkan *query data* yang dieksekusi. SQLite unggul dalam melakukan *insert, update* semua kolom, *delete*, dan *select* seluruh data. Sedangkan DuckDB unggul dalam melakukan *update* dua buah kolom dan unggul pada semua *query* yang menggunakan fungsi agregat. Dengan hasil tersebut dapat disimpulkan bahwa SQLite lebih cocok digunakan untuk menangani proses transaksi karena SQLite dapat mengeksekusi *query insert, update, delete*, dan *select data* dengan cepat. Sementara itu, DuckDB lebih cocok digunakan untuk melakukan proses analisis *big data* karena DuckDB dapat mengeksekusi fungsi agregat dengan waktu yang jauh lebih cepat dibandingkan dengan SQLite. Oleh karena itu, langkah yang tepat dalam memilih *database* adalah dengan menyesuaikan kebutuhan masing-masing pengguna. Dengan memilih *database* yang sesuai kebutuhan, proses akan dieksekusi lebih efektif dan efisien sehingga performa yang didapatkan lebih baik.

Pada penelitian ini masih terdapat beberapa kekurangan di antaranya yaitu tabel yang digunakan adalah tabel tanpa relasi dan tipe data yang kurang bervariasi. Untuk penelitian selanjutnya disarankan untuk menggunakan tabel dan data yang lebih kompleks. Selain itu, skema pengujian dapat dibuat lebih kompleks seperti menggunakan *join table* dan *where clause* yang rumit.

DAFTAR PUSTAKA

[1] Medinah A. (Juni 2020). Pengaruh Kemajuan Teknologi Terhadap Pola Komunikasi Mahasiswa Universitas Muhammadiyah Malang (UMM). *Jurnal Sosiologi Nusantara.* [Online]. 6(1), hal. 45-54. Tersedia: https://doi.org/10.33369/jsn.6.1.45-54
[2] Agung P. dkk. (September 2018). Pemanfaatan Big Data dan Perlindungan Privasi Konsumen di Era Ekonomi Digital. *Majalah Ilmiah BIJAK.* [Online]. 15(2), hal. 127-137. Tersedia: https://doi.org/10.31334/bijak.v15i2.201
[3] Mohammad G. E. (Juli 2020). Pemanfaatan Big Data Dalam Penelitian Teknologi Pendidikan. *Jurnal Teknologi Pendidikan.* [Online]. 5(2), hal. 107-120. Tersedia: https://doi.org/10.32832/educate.v5i2.3381
[4] Ari F. dkk. (November 2020). Analisis Perbandingan Unjuk Kerja Database SQL dan Database NoSQL Untuk Mendukung Era Big Data. *Jurnal Nasional Teknik Elektro.* [Online]. 9(3), hal. 154-158. Tersedia: https://doi.org/10.25077/jnte.v9n3.774.2020
[5] Riyan S. P. dan Ubaidillah Z. (Febriari 2021). Time Management Skills For Entrepreneur Success. *Jurnal ABDIMAS.* [Online]. 1(1), hal. 38:42. Tersedia: https://doi.org/10.47927/jasd.v1i1.88
[6] Alvin D. H. dan Catur N. P. D. (Agustus 2020). Perancangan Basis Data Sistem Informasi Perwira Tugas Belajar (Sipatubel) Pada Kementerian Pertahanan. *Seminar Nasional Mahasiswa Ilmu Komputer dan Aplikasinya (SENAMIKA).* [Online]. 1(2), hal. 222-233. Tersedia: https://conference.upnvj.ac.id/index.php/senamika/article/view/529
[7] Alek S. (Januari 2021). Pemanfaatan Basis Data, Perangkat Lunak, dan Mesin Industri Dalam meningkatkan Produksi Perusahaan. J*urnal Manajemen Pendidikan dan Ilmu Sosial.* [Online]. 3(1), hal. 1-14. Tersedia: https://doi.org/10.38035/jmpis.v3i1
[8] Yudha Y. P. dkk. (Februari 2022). Perbandingan Performa Respon Waktu Kueri MySQL, PostgreSQL, dan MongoDB. *Jurnal Sistem Informasi dan Bisnis Cerdas (SIBC).* [Online]. 15(1), hal. 39-48. Tersedia: https://doi.org/10.33005/sibc.v15i1.7
[9] Geraldo A. L. dkk. (Juli 2019). Rancang Bangun Aplikasi Ensiklopedia Hukum Indonesia Berbasis Android. *Jurnal Teknik Informatika.* [Online]. 14(3), hal. 341-348. Tersedia: https://doi.org/10.35793/jti.14.3.2019.27125
[10] Dibyo S. dkk. (Mei 2021). Perbandingan Performansi Antara MongoDB dan MariaDB. *Jurnal FAHMA.* 19(2), hal. 1-11. Tersedia: https://stmikelrahma.e-journal.id/FAHMA/article/view/72
[11] Aaron D. C. O. dkk. (Februari 2019). Implementasi ETL dan Perbadingan Performa Column-Oriented Database dan Relational Database sebagai Data Warehouse. *Jurnal Infra.* [Online]. 7(1), hal. 27-32. Tersedia: https://publication.petra.ac.id/index.php/teknik-informatika/article/view/8040
[12] Ardian D. P. dan Maryanah S. (Desember 2020). Studi Perbandingan Performansi Antara MySQL dan PostgreSQL. *Jurnal Khatulistiwa Informatika.* [Online]. 3(2), hal. 88-93. Tersedia: https://doi.org/10.31294/jki.v8i2.8851
[13] Nelly M. dkk. (Agustus 2018). Aplikasi Data Mahasiswa Berbasis Android: Studi Pada Sekolah Tinggi Ilmu Ekonomi Labuhanbatu. *IT Journal Research and Development.* [Online]. 3(1), hal. 43-53. Tersedia: https://doi.org/10.25299/itjrd.2018.vol3(1).1849
[14] Muhammad F.. dkk. (Desember 2017). Pembangkitan Formulir Web Berdasarkan Metadata SQL dan Spesifikasi W3C. *Jurnal CoreIT.* [Online]. 3(2), hal. 63-68. Tersedia: http://dx.doi.org/10.24014/coreit.v3i2.4417
[15] Didik S. dan Herlawati. (Juni 2019). Structured Query Language (SQL) untuk Purchase Order (PO) Menggunakan SQL Server 2008. *Bina Insani ICT Journal.* [Online]. 6(1), hal. 75-88. Tersedia: https://ejournal-binainsani.ac.id/index.php/BIICT/article/view/1102

[16] Y. V. Sai B. dkk. (Februari 2019). SQLite Database and its Application on Embedded Platform. *International Journal of Computer Trends and Technology*. [Online]. 67(2), hal. 1-6. Tersedia: https://doi.org/10.14445/22312803/IJCTT-V67I2P101
[17] Nikola O. dkk. (Maret 2019). Performance analysis on Android SQLite database. *18th International Symposium INFOTEH-JAHORINA*. [Online]. Tersedia: https://doi.org/10.1109/INFOTEH.2019.8717652
[18] Laurens K. dkk. (Maret 2021). Efficient External Sorting in DuckDB. *The British International Conference on Databases 2021*. [Online]. Hal. 40-45. Tersedia: https://ceur-ws.org/Vol-3163/BICOD21_paper_9.pdf