Project 21 - NoSQL DBMS Implementation

Sistem Informasi Akademik (SIX) ITB

Disusun oleh Kelompok 2 :

1. Fedrianz Dharma (13522090)
2. Era Desti Ramayani (23525020)
3. Kayis Shalahuddin (23525034)
4. Fadhil Rausyanfikr (23525046)
5. Wilson Tansil (23525060)

Magister Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2025

Daftar Isi

1. Studi Literatur *Database Management System* (DBMS) *NoSQL*

Studi literatur ini menyajikan tinjauan terhadap enam *platform* basis data modern terkemuka, di antaranya *MongoDB*, *Cassandra*, *Redis*, *ElasticSearch*, *Neo4j*, dan *ArangoDB*.

a. MongoDB

MongoDB merupakan contoh utama dari arsitektur NoSQL berbasis *document*. MongoDB menyimpan data dalam bentuk dokumen yang memiliki struktur fleksibel di dalam sebuah koleksi (collection), yaitu himpunan dokumen yang berfungsi seperti tabel pada sistem basis data relasional. Setiap dokumen dapat memiliki atribut yang berbeda tanpa harus mengikuti skema tetap, sehingga memberikan keleluasaan bagi developer untuk menyimpan dan mengelola data yang kompleks, bersarang, maupun berubah secara dinamis seiring perkembangan aplikasi.

MongoDB menggunakan bahasa kueri deklaratif yang mirip dengan JSON, yang disebut MongoDB *Query Language* (MQL). Bahasa ini mendukung operasi CRUD (Create, Read, Update, Delete), agregasi data, serta manipulasi dokumen yang kompleks melalui aggregation pipeline. Selain itu, MongoDB juga menyediakan fitur seperti *indexing*, *geospatial queries*, *full-text search*, serta *transactions* dengan jaminan ACID penuh yang memungkinkan beberapa operasi lintas dokumen dalam satu transaksi atomik.

Kelebihan MongoDB terletak pada kemampuannya dalam menangani data beragam dengan performa tinggi, fleksibilitas desain skema, serta kemudahan integrasi dengan teknologi modern. Namun, kelemahannya adalah kebutuhan desain indeks dan partisi yang hati-hati untuk menghindari penurunan performa pada volume data besar, serta beban pengelolaan integritas data yang sebagian bergantung pada logika aplikasi.

b. Cassandra

Apache Cassandra merupakan contoh utama dari arsitektur distributed *wide-column storage*. Berbeda dengan ArangoDB yang menjadi alternatif untuk *polyglot persistence,* Cassandra seringkali menjadi salah satu komponen kunci dalam tumpukan *polyglot persistence* itu sendiri. Cassandra memiliki tipe database *Column-Family* yang didesain secara khusus untuk menangani volume data dan *query* berskala masif dengan linear *horizontal scalability, high availability*, dan *fault tolerance* penuh. Pendekatan ini dirancang untuk aplikasi yang membutuhkan global data *replication* dengan *read and write latency* yang selalu rendah.

Cassandra memiliki struktur dasar berupa *Keyspace* yang berisi Table. Data diatur dalam *Row* yang diidentifikasi oleh *Primary Key* unik. Kunci ini terdiri dari *Partition Key* dan Clustering Columns. Cassandra menggunakan *query language* CQL (*Cassandra Query Language*), yang syntax-nya dibuat agar terlihat seperti SQL. Untuk konsistensi, Cassandra mengadopsi model *eventually consistent*. Arsitektur Cassandra memprioritaskan *Availability* dan *Partition Tolerance* di atas *full consistency*. Konsistensi bersifat "*tunable"* peroperasi, namun Cassandra tidak

mendukung *cross-partition transactions*. Fitur *lightweight transactions* yang dimilikinya terbatas hanya pada satu *single partition.*

Kelemahan utama Cassandra adalah fleksibilitas querynya yang sangat terbatas. Ketidakmampuan melakukan JOIN, *foreign key*, atau ad-hoc queries yang kompleks memaksa developer untuk melakukan data modeling dengan pendekatan *query-first design*.

c. Redis

Redis (*Remote Dictionary Server*) merupakan contoh utama dari pendekatan arsitektur *key-value* yang dirancang untuk memberikan performa sangat tinggi dengan latensi rendah. Redis menyimpan seluruh data utamanya di dalam memori utama (RAM) dan hanya melakukan penyimpanan ke disk untuk keperluan *persistence*, sehingga memungkinkan operasi baca dan tulis dieksekusi dalam waktu sub-millisecond. Arsitektur ini menjadikan Redis banyak digunakan untuk kebutuhan *caching*, *session management*, *message queuing*, serta *real-time analytics*.

Redis mendukung berbagai tipe data kompleks seperti strings, lists, sets, sorted sets, hashes, dan streams. Setiap tipe data memiliki kumpulan perintah (commands) spesifik untuk manipulasi, seperti LPUSH dan LPOP untuk list, atau HSET dan HGET untuk hash. Redis juga menyediakan fitur tambahan seperti pub/sub messaging, transactions, Lua scripting, serta data persistence melalui dua mekanisme utama, yaitu RDB (Redis Database Backup) dan AOF (Append Only File). Kedua mekanisme ini memungkinkan Redis memulihkan data ketika terjadi restart atau kegagalan sistem.

Redis ditulis dalam ANSI C dan dapat berjalan di sebagian besar sistem POSIX seperti Linux, *BSD, dan Mac OS X, tanpa ketergantungan eksternal. Linux dan OS X adalah dua sistem operasi di mana Redis paling banyak dikembangkan dan diuji, dan direkomendasikan penggunaan Linux untuk deployment.

Kelemahan utama Redis terletak pada ketergantungannya terhadap memori utama. Karena seluruh data aktif disimpan di RAM, biaya penyimpanan dapat meningkat secara signifikan seiring bertambahnya volume data, sehingga penggunaan Redis menjadi kurang efisien untuk penyimpanan data berskala besar yang jarang diakses. Meskipun Redis memiliki fitur *persistence*, sifatnya tetap lebih cocok untuk *volatile data*.

d. Elasticsearch

Elasticsearch adalah mesin pencarian dan analitik *open source* terdistribusi yang dibangun di atas Apache Lucene dirancang untuk kecepatan, skalabilitas, dan aplikasi AI dengan tipe database *document oriented*. Sebagai platform pengambilan data, Elasticsearch menyimpan data terstruktur, tidak terstruktur, dan vektor secara *real-time*, menyediakan pencarian hybrid dan

vektor yang cepat, mendukung analitik observability dan keamanan, serta memungkinkan aplikasi AI dengan performa tinggi, akurasi, dan relevansi.

Arsitektur dasar Elasticsearch :

- Cluster: Kumpulan satu atau lebih node yang bekerja bersama
- Node: *Instance* tunggal Elasticsearch yang menyimpan data
- Index: Kumpulan dokumen dengan karakteristik serupa (mirip database)
- Type: Kategori dokumen dalam index (deprecated di versi 7+)
- Document: Unit data dasar dalam format JSON
- Shard: Bagian horizontal dari index untuk distribusi data
- Replica: Salinan shard untuk redundansi dan performa

Proses Indexing : Data → *Analyzer* → *Inverted Index* → *Shard Distribution*

Data dianalisis dan dipecah menjadi *terms*. *Terms* disimpan dalam inverted index untuk pencarian cepat. Data didistribusikan ke berbagai shard untuk skalabilitas.

Proses Search : *Query* → *Query* DSL → *Distributed Search* → *Score Calculation* → *Results*

Berikut beberapa kelebihan dari Elasticsearch :

Kategori	**Fitur**	**Penjelasan**
Performance	*Near Real-time Search*	Pencarian hampir *real-time* (default refresh 1 detik)
	Horizontal Scaling	Mudah menambah node untuk meningkatkan kapasitas
	Parallel Processing	*Query* dijalankan secara paralel di multiple shard
Flexibility	*Schema-free*	Tidak memerlukan skema yang rigid (*dynamic mapping*)
	Multi-tenancy	Satu cluster bisa menangani multiple aplikasi
	RESTful API	*Interface* yang mudah digunakan via HTTP
Features	*Full-text Search*	Pencarian teks lengkap dengan scoring relevansi
	Aggregations	Analitik dan reporting yang powerful

Kategori	Fitur	Penjelasan
	Geospatial	Dukungan pencarian berdasarkan lokasi
	Auto-completion	Fitur *suggestion* dan *auto complete*
Ecosystem	ELK Stack	Integrasi dengan Logstash dan Kibana
	Rich Query DSL	Bahasa query yang ekspresif dan powerful
	Plugin Architecture	*Extensible* dengan berbagai plugin

Berikut beberapa kekurangan Elasticsearch :

Kategori	Masalah	Penjelasan
Complexity	*Learning Curve*	Butuh waktu untuk menguasai
	Configuration	Banyak parameter yang perlu dioptimasi
	Monitoring	Perlu monitoring yang intensif
Resource Intensive	*Memory Usage*	Konsumsi RAM yang tinggi
	Storage Overhead	*Inverted* index membutuhkan ruang tambahan
	CPU Usage	Indexing dan aggregation intensif CPU
Consistency	*Eventually Consistent*	Tidak ACID *compliant*
	No Transactions	Tidak mendukung *multi-document transactions*
	Update Limitations	*Update* dokumen sebenarnya adalah *delete + insert*
Operational	*Split-brain*	Risiko cluster split dalam network partition
	Version Compatibility	Upgrade antar major versionnya kompleks

Kategori	Masalah	Penjelasan
	Backup/Recovery	*Snapshot* dan *restore* lambat untuk dataset besar

Elasticsearch cocok untuk Use Case sebagai berikut :

Use Case	Contoh
Search applications	*E-commerce, content search*
Log analytics dan *monitoring*	*System logs, application logs*
Real-time analytics	*Dashboard, business intelligence*
Recommendation engines	*Product recommendations*
Geospatial applications	*Location based search*

Elasticsearch tidak cocok untuk Use Case yang :

Use Case	Alasan
Transactional systems (OLTP)	Tidak mendukung ACID *transactions*
Data yang memerlukan *strong consistency*	*Eventually consistent*
Simple CRUD *applications*	Terlalu kompleks dan overkill
Aplikasi dengan *budget* terbatas	*Resource intensive*, biaya tinggi

e. Neo4j

Neo4j merupakan *native graph database* yang dirancang khusus untuk menyimpan dan mengelola data berelasi kompleks dalam bentuk *graph* secara bawaan. Sistem ini menyimpan *node* (entitas) dan *relationship* (relasi antar entitas) langsung pada *storage engine*, bukan sekedar representasi graf yang dipetakan ke model lain seperti tabel atau dokumen. Setiap *node* dan *relationship* memiliki ID unik serta *property* (*key-value*) tersendiri, sementara relasi diperlakukan sebagai *first-class object*, sehingga *graph traversal* dapat dilakukan dengan efisien tanpa *index lookup* tambahan.

Model data Neo4j disebut *property graph*, yang memungkinkan setiap node dan relationship memiliki label serta *property* fleksibel. Struktur ini mendukung pertumbuhan graf tanpa *schema migration* yang rumit, sehingga sangat cocok diterapkan untuk sistem dengan data yang bersifat dinamis, kompleks, dan memiliki tingkat relasi yang banyak dan padat seperti *social network*, sistem rekomendasi, *knowledge graph*, maupun analisis hubungan antar entitas pada domain bisnis dan teknologi informasi.

Berbeda dengan database lain yang melakukan *query* menggunakan pendekatan berbasis tabel seperti SQL atau berbasis dokumen seperti AQL pada ArangoDB, Neo4j menggunakan bahasa kueri *Cypher* yang berorientasi pada *pattern matching* graf. Sintaksnya bergaya *ASCII-Art*, misalnya MATCH (a:Person)-[:KNOWS]->(b:Person) RETURN a, b, sehingga pola hubungan antar entitas dapat divisualisasikan dengan jelas dan mudah dipahami.

Keunggulan utama Neo4j terletak pada kemampuannya menelusuri hubungan data dengan sangat cepat melalui konsep *index-free adjacency*. Selain itu, Neo4j mendukung transaksi ACID yang menjaga konsistensi data, memiliki ekosistem pengembangan yang luas, serta menyediakan alat visualisasi interaktif seperti Neo4j Browser dan Neo4j Bloom. Fitur-fitur *enterprise* seperti *clustering*, *Fabric*, dan sistem *monitoring* terintegrasi juga menjadikan Neo4j lebih baik dan mudah di-*scaling* terutama untuk kebutuhan organisasi berskala besar.

Meskipun demikian, dibandingkan dengan model *graph database* lain seperti ArangoDB atau TigerGraph, Neo4j memiliki beberapa kelemahan yang perlu diperhatikan. Beberapa pengguna melaporkan bahwa performa Neo4j cenderung menurun pada beban transaksi tulis (*write-heavy workload*) atau saat melakukan agregasi data berskala besar. Selain itu, penggunaan bahasa kueri **Cypher** memerlukan proses adaptasi tersendiri bagi pengguna yang terbiasa dengan SQL, serta biaya lisensi untuk versi *enterprise* juga relatif tinggi dibandingkan solusi *open-source* lainnya.

f. ArangoDB

ArangoDB merupakan contoh utama dari pendekatan arsitektur *native multi-model* yang menawarkan alternatif dalam menyelesaikan permasalahan model *polyglot persistence* (seperti *Neo4j* dan *MongoDB*). *ArangoDB* didesain untuk menyimpan, mengelola, dan meng-*query* data menggunakan berbagai tipe model, khususnya *doc-based*, *graph*, dan *key-valued*, yang digabungkan dalam satu inti basis data dan *storage engine* yang terintegrasi. Pendekatan ini secara radikal menyederhanakan tumpukan teknologi.

ArangoDB merupakan *native multi-model* yang memiliki struktur dasar berupa graf, dengan dokumen sebagai *vertex* dan relasi disimpan sebagai *edge* di *collection* dengan atribut khusus *_from* dan *_to* yang menunjuk ke *_id vertex*. *ArangoDB* menggunakan bahasa kueri AQL (*ArangoDB Query Language*) yang bersifat deklaratif seperti SQL dan dipakai untuk semua model data. AQL juga mendukung *JOIN* lintas koneksi dan bisa digabung dengan traversal graf di dalam satu *query*, sehingga hasil penjelajahan graf dapat diperkaya dengan data dokumen

dalam sekali eksekusi. Untuk konsistensi, *ArangoDB* mendukung transaksi yang ACID (*Atomicity*, *Consistency*, *Isolation*, *Durability*) secara penuh, bahkan untuk *query* yang kompleks yang melibatkan banyak koleksi atau model. Ini merupakan keunggulan signifikan dibandingkan dengan sistem *polyglot*, karena menyelesaikan permasalahan sistem *polyglot* dengan menjamin konsistensi lintas basis data.

Meskipun demikian, *ArangoDB* juga memiliki beberapa kelemahan yang dilaporkan oleh pengguna. Beberapa pengguna melaporkan performa yang lambat selama agregasi kompleks dan penelusuran *graph* yang mengindikasikan bahwa arsitektur *jack-of-all-trades* ini mungkin memiliki *trade-off* pada kinerja. Selain itu, AQL meningkatkan *learning rate* untuk pengguna baru karena sintaksnya yang unik untuk menggabungkan model, terutama bagi pengguna yang beralih dari SQL. Keluhan lainnya yang sering muncul adalah dokumentasi yang buruk yang menyulitkan pencarian solusi untuk kasus penggunaan tingkat lanjut. Dari segi biaya, *ArangoDB* bisa jadi mahal untuk aplikasi CRUD sederhana.

2. Justifikasi Pemilihan Kedua DBMS

Berikut adalah justifikasi teknis untuk perbandingan antara *MongoDB* dan *ArangoDB*. Kedua DBMS ini dipilih sebagai kandidat karena kapabilitas NoSQL mereka yang kuat, namun keduanya menawarkan pendekatan yang berbeda secara fundamental.

a. MongoDB

Pemilihan MongoDB sebagai alternatif arsitektur basis data untuk Sistem Informasi Akademik (SIX) didasari oleh kebutuhan sistem yang menuntut fleksibilitas skema dan performa tinggi untuk menangani *volume* data yang terus berkembang. Sebagai sistem NoSQL berorientasi dokumen, MongoDB memungkinkan setiap entitas seperti mahasiswa, dosen, mata kuliah, dan kelas disimpan dalam bentuk dokumen yang fleksibel di dalam koleksi. Struktur ini memudahkan pengembang untuk menyesuaikan skema data seiring dengan perubahan kebutuhan akademik tanpa harus melakukan migrasi atau modifikasi skema yang kompleks seperti pada basis data relasional. Kemampuan menyimpan data *nested* dan hierarkis juga memungkinkan representasi natural dari struktur akademik, seperti transkrip nilai yang mencakup rincian per semester, mata kuliah, hingga komponen penilaian dalam satu dokumen yang kohesif.

Selain itu, MongoDB memiliki kemampuan horizontal *scaling* melalui mekanisme *sharding* dan mendukung *replica set* untuk menjamin ketersediaan tinggi (*high availability*). Hal ini sangat relevan dengan karakteristik Sistem Informasi Akademik (SIX) yang melayani ribuan pengguna secara bersamaan, seperti mahasiswa, dosen, dan tenaga kependidikan, yang melakukan transaksi data secara *real time*.

Dari sisi pengembangan, MongoDB juga mendukung bahasa kueri deklaratif yang mudah dipahami serta *aggregation framework* yang kuat untuk melakukan analisis dan pelaporan data akademik, seperti rekap nilai, statistik kehadiran, dan data kelulusan. Integrasinya yang baik

dengan ekosistem JavaScript menjadikan MongoDB selaras dengan teknologi yang umum digunakan dalam pengembangan aplikasi web modern, sekaligus mempercepat siklus pengembangan dan deployment sistem.

b. Neo4j

Pemilihan Neo4j untuk proyek ini didasarkan pada arsitektur intinya sebagai *native graph database* yang secara khusus dioptimalkan untuk menyimpan dan mengelola data dengan relasi yang kompleks dan padat, yang merupakan kebutuhan inti dari sistem akademik SIX. Keunggulan utamanya terletak pada konsep *index-free adjacency*, yang memungkinkan *graph traversal* atau penelusuran hubungan data dilakukan dengan efisiensi dan kecepatan yang sangat tinggi karena *relationship* diperlakukan sebagai *first-class object*, bukan sekedar atribut yang dipetakan ke model lain. ArangoDB juga menawarkan arsitektur *database* yang sama, namun pada dasarnya adalah arsitektur *native multi-model*, graf direpresentasikan di atas model dokumen. Pendekatan "*jack-of-all-trades*" ArangoDB ini, seperti yang dilaporkan beberapa pengguna, dapat menimbulkan *trade-off* pada kinerja, khususnya performa yang lambat selama penelusuran *graph* yang kompleks. Mengingat proyek ini sangat bergantung pada kecepatan *query* relasional dan *pattern matching*, spesialisasi Neo4j dinilai lebih superior.

3. Implementasi *Setup*

Berikut adalah rincian implementasi teknis dari proses setup untuk *MongoDB* dan *ArangoDB*.

a. MongoDB

Implementasi *MongoDB* dilakukan menggunakan *Docker* untuk memastikan lingkungan setup yang terisolasi, konsisten, dan portabel. Proses ini didefinisikan dan dikelola menggunakan *Docker Compose*, yang memungkinkan konfigurasi layanan, jaringan, dan volume dalam satu file *YAML*.

- **Konfigurasi Layanan**

Dua file digunakan untuk konfigurasi, yakni ***docker-compose.yml*** untuk mendefinisikan layanan dan ***.env*** untuk menyimpan kredensial sensitif.

```
# docker-compose.yml
version: '3.8'

services:
  mongo:
    image: mongo:latest
    restart: always
    ports:
      - "27017:27017"
    env_file:
```

```
    - .env
   volumes:
    - mongo-data:/data/db

volumes:
  mongo-data:

# .env
MONGO_INITDB_ROOT_USERNAME=mongo
MONGO_INITDB_ROOT_PASSWORD=secret
```

- **Eksekusi dan Verifikasi**

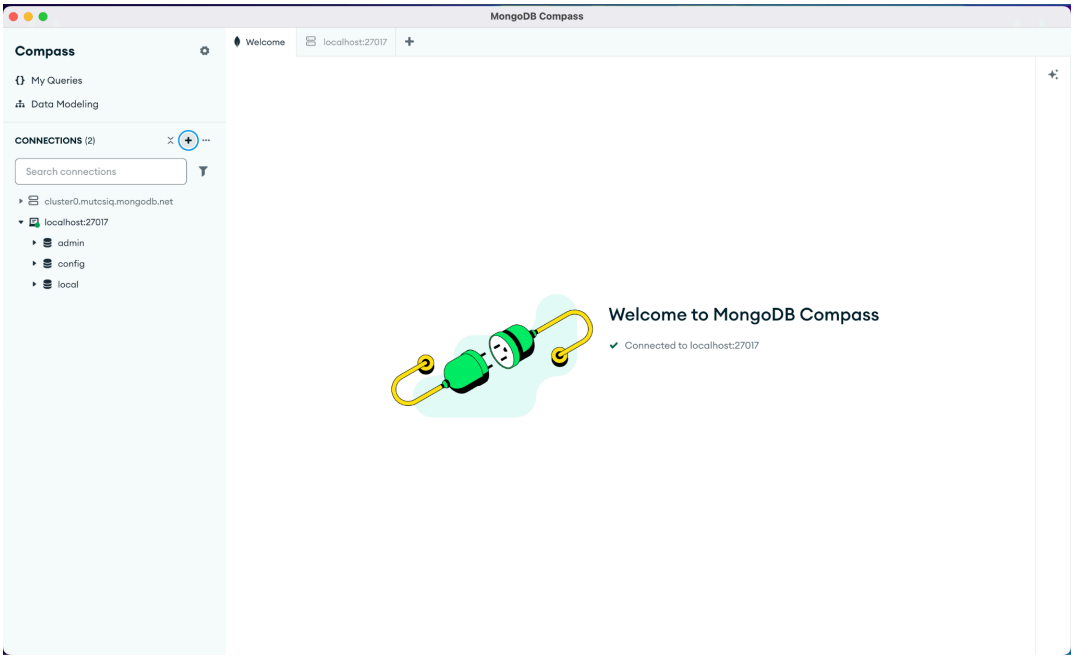
Layanan *MongoDB* dijalankan menggunakan perintah berikut:

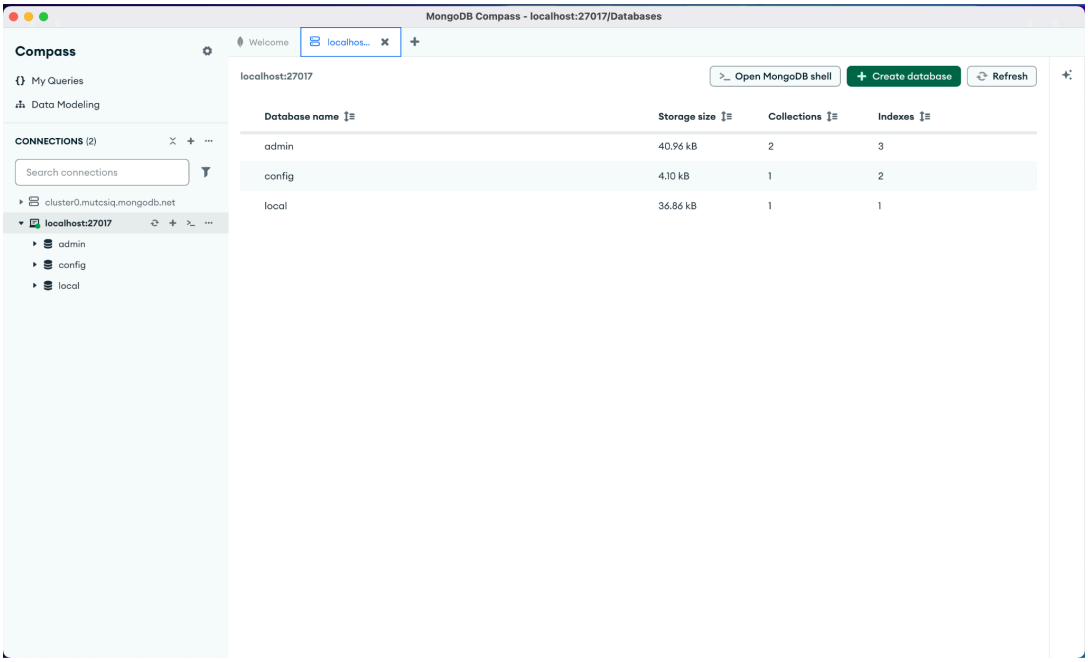
```
docker-compose up -d
```

Setelah kontainer berhasil berjalan, gunakan *MongoDB Compass* lalu buka koneksi baru menggunakan *URI* berikut:

```
mongodb://mongo:secret@localhost:27017/?authSource=admin
```

- **Tampilan *MongoDb Compass Interface***





b. Neo4j

- **Konfigurasi Layanan**

Dua file digunakan untuk konfigurasi, yakni ***docker-compose.yml*** untuk mendefinisikan layanan dan ***.env*** untuk menyimpan kredensial sensitif.

```
version: '3.8'

services:
  neo4j:
    image: neo4j:latest
    container_name: neo4j
    ports:
      - "7474:7474"
      - "7687:7687"
    environment:
      - NEO4J_AUTH=neo4j/testpassword
      - NEO4J_dbms_memory_pagecache_size=512M  # Page cache size
      - NEO4J_dbms_memory_heap_initial__size=4G  # Initial heap size
      - NEO4J_dbms_memory_heap_max__size=4G  # Max heap size
            - NEO4J_dbms_memory_transaction_total_max=4G
    volumes:
      - /$HOME/neo4j/logs:/logs
      - /$HOME/neo4j/config:/config
      - /$HOME/neo4j/data:/data
      - /$HOME/neo4j/plugins:/plugins
    restart: always
```

- **Eksekusi dan Verifikasi**

Layanan *neo4j* dijalankan menggunakan perintah berikut:

```
docker-compose up -d
```

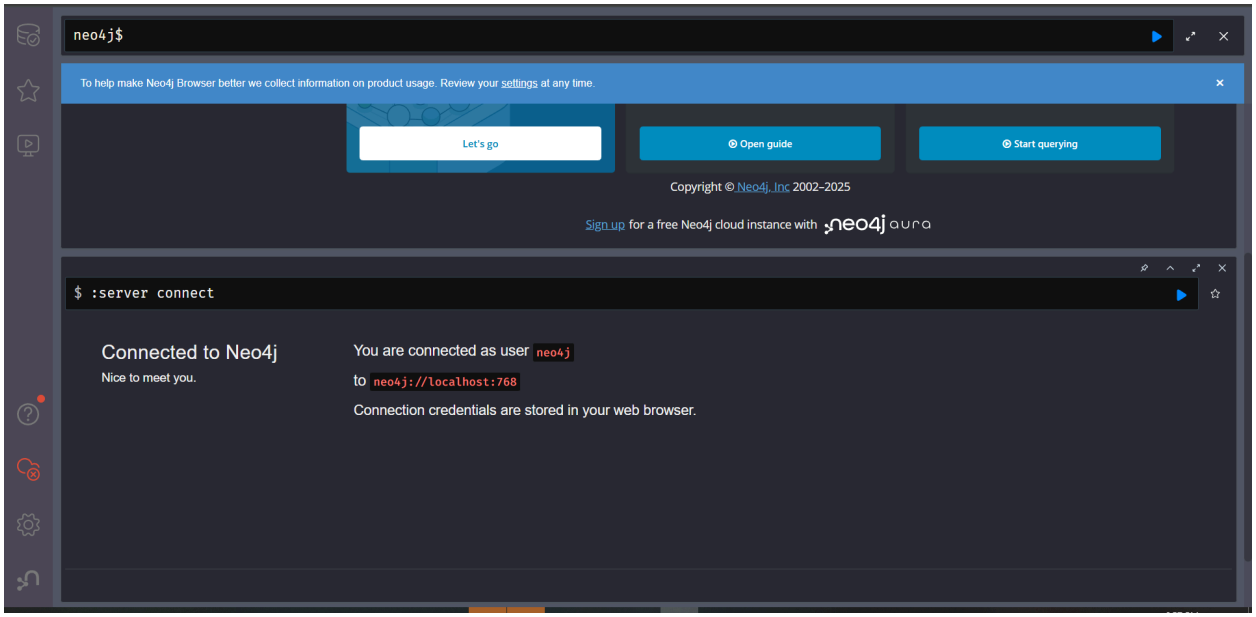
Setelah kontainer berhasil berjalan, *neo4j Web Interface* dapat diakses melalui browser dengan alamat berikut:

```
http://localhost:7687
```

Login dilakukan dengan kredensial yang telah ditetapkan:

```
Username: neo4j
Password: testpassword
Database: _system
```

- **Tampilan *Neo4j Web Interface***



Project 22 - NoSQL DBMS Implementation

Sistem Informasi Akademik (SIX) ITB

Disusun oleh Kelompok 2 :

1. Fedrianz Dharma (13522090)
2. Era Desti Ramayani (23525020)
3. Kayis Shalahuddin (23525034)
4. Fadhil Rausyanfikr (23525046)
5. Wilson Tansil (23525060)

Magister Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2025

1. **Query**

1.1 Mongo DB

Berikut implementasi query yang didefinisikan sebelumnya pada project 1 :

- **Query Index** : Query di bawah ini digunakan untuk menambahkan index

```
db.mahasiswa.createIndex({ nim: 1 })
db.dosen.createIndex({ nip: 1 })
db.program_studi.createIndex({ kodeProdi: 1 })
db.fakultas.createIndex({ kodeFakultas: 1 })
db.mata_kuliah.createIndex({ kodeMatkul: 1 })

db.rencana_studi.createIndex({ nim: 1 })
db.rencana_studi.createIndex({ "mataKuliah.kodeMatkul": 1 })

db.dosen_pengampu.createIndex({ nipDosen: 1 })
db.dosen_pengampu.createIndex({ kodeProdi: 1, kodeMatkul: 1 })

db.jadwal_kuliah.createIndex({ kodeRuangan: 1, tanggal: 1 })
db.jadwal_kuliah.createIndex({ nipDosen: 1 })
```

- **Query 1** : Query di bawah ini digunakan untuk menampilkan daftar mahasiswa + dosen wali + prodi + fakultas

```
db.mahasiswa.explain("executionStats").aggregate([
  {
    $lookup: {
      from: "dosen",
      localField: "dosenWali",
      foreignField: "nip",
      as: "dosen"
    }
  },
  { $unwind: { path: "$dosen", preserveNullAndEmptyArrays: true
} },
```

```
  {
    $lookup: {
      from: "program_studi",
      localField: "kodeProdi",
      foreignField: "kodeProdi",
      as: "prodi"
    }
  },
  { $unwind: "$prodi" },
  {
    $lookup: {
      from: "fakultas",
      localField: "prodi.kodeFakultas",
      foreignField: "kodeFakultas",
      as: "fakultas"
    }
  },
  { $unwind: "$fakultas" },
  {
    $project: {
      _id: 0,
      nim: 1,
      nama_lengkap: "$namaLengkap",
      dosen_wali: "$dosen.namaDosen",
      nama_prodi: "$prodi.namaProdi",
      nama_fakultas: "$fakultas.namaFakultas"
    }
  },
  {
    $sort: {
      nama_fakultas: 1,
      nama_prodi: 1,
      nama_lengkap: 1
    }
  }
])
```

Item	Value
Row Output	33296
Execution Time	3155 ms

Berdasarkan tabel di atas dapat dilihat bahwa dengan waktu eksekusi 3155 ms dengan row output 33296 tergolong cukup lambat. Solusinya dapat melakukan denormalisasi dan menyimpan nama fakultas, nama prodi, dan nama dosen wali pada kolesi mahasiswa.

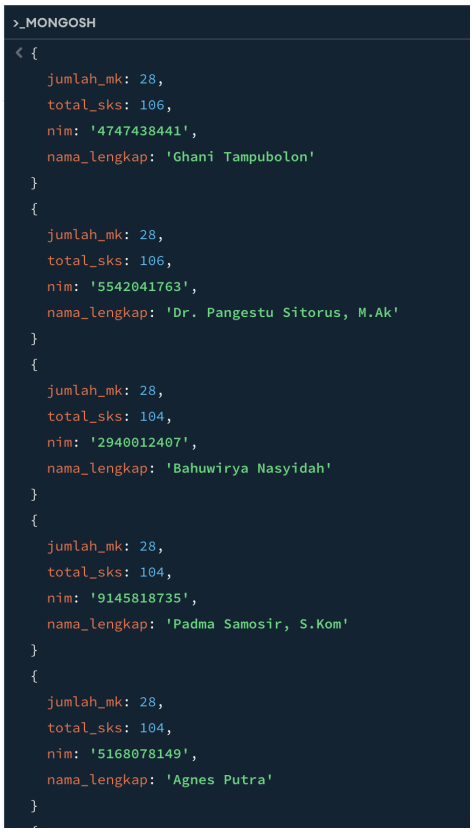
- **Query 2** : Query di bawah ini digunakan untuk menampilkan jumlah Mata Kuliah dan total SKS per mahasiswa

```
db.rencana_studi.explain("executionStats").aggregate([
  { $unwind: "$mataKuliah" },
  {
    $group: {
      _id: "$nim",
      jumlah_mk: { $sum: 1 },
      total_sks: { $sum: "$mataKuliah.sks" }
    }
```

```
  },
  {
    $lookup: {
      from: "mahasiswa",
      localField: "_id",
      foreignField: "nim",
      as: "mhs"
    }
  },
  { $unwind: "$mhs" },
  {
    $project: {
      _id: 0,
      nim: "$_id",
      nama_lengkap: "$mhs.namaLengkap",
      jumlah_mk: 1,
      total_sks: 1
    }
  },
  { $sort: { total_sks: -1 } }
])
```

Item	Value
Row Output	33296
Execution Time	1827 ms

Berdasarkan tabel di atas dapat dilihat bahwa dengan waktu eksekusi 1827 ms dengan row output 33296 tergolong cukup lambat. Solusinya dapat melakukan denormalisasi dan menyimpan nama lengkap mahasiswa pada embedding rencana studi sehingga tidak perlu dilakukan $lookup pada koleksi mahasiswa.



- **Query 3 :** Query di bawah ini digunakan untuk menampilkan daftar Mahasiswa yang rencana studinya belum dikirim

```
db.rencana_studi.explain("executionStats").aggregate([
  { $match: { pengirimanRencanaStudi: null } },
  {
    $lookup: {
      from: "mahasiswa",
      localField: "nim",
      foreignField: "nim",
      as: "mhs"
    }
  },
  { $unwind: "$mhs" },
  {
    $project: {
      _id: 0,
      nim: "$mhs.nim",
```

```
      nama_lengkap: "$mhs.namaLengkap",
      tahun_ajaran: "$tahunAjaran",
      semester: "$semester"
    }
  },
  { $sort: { tahun_ajaran: -1 } }
])
```

Item	Value
Row Output	53557
Execution Time	1749 ms

Berdasarkan tabel di atas dapat dilihat bahwa dengan waktu eksekusi 1749 ms dengan row output 53557 tergolong cukup lambat. Solusinya dapat melakukan denormalisasi dan menyimpan nama lengkap mahasiswa pada embedding rencana studi sehingga tidak perlu dilakukan $lookup pada koleksi mahasiswa.

- **Query 4 :** Query di bawah ini digunakan untuk menampilkan jumlah MK & kelas (ruangan) per dosen di 2024

```
db.dosen.explain("executionStats").aggregate([
  {
    $lookup: {
      from: "dosen_pengampu",
      localField: "nip",
      foreignField: "nipDosen",
      as: "pengampu"
    }
  },
  {
    $lookup: {
      from: "jadwal_kuliah",
      let: { nip: "$nip" },
      pipeline: [
        { $match: { tahun: 2024 } },
        {
          $match: {
            $expr: { $eq: ["$nipDosen", "$$nip"] }
          }
        }
      ],
      as: "jadwal"
    }
  },
  {
    $project: {
      nip: 1,
      nama_dosen: "$namaDosen",
      mkCodes: { $setUnion: ["$pengampu.kodeMatkul", []] },
      rooms: { $setUnion: ["$jadwal.kodeRuangan", []] }
    }
  },
  {
    $project: {
      _id: 0,
      nip: 1,
```

```
      nama_dosen: 1,
      jumlah_mk: { $size: "$mkCodes" },
      jumlah_kelas: { $size: "$rooms" }
    }
  }
])
```

Item	Value
Row Output	3540
Execution Time	332 ms

Berdasarkan tabel di atas, execution time 332 ms dengan row output 3540 tergolong baik.

- **Query 5 :** Query di bawah ini digunakan untuk menampilkan total SKS lulus per mahasiswa (nilai A–C, top 5)

```
db.rencana_studi.explain("executionStats").aggregate([
  { $unwind: "$mataKuliah" },

  {
    $match: {
      "mataKuliah.nilai": { $in: ["A", "AB", "B", "BC", "C"] }
    }
  },

  {
    $group: {
      _id: "$nim",
      total_sks_lulus: { $sum: "$mataKuliah.sks" }
    }
  },

  {
    $lookup: {
      from: "mahasiswa",
      localField: "_id",
      foreignField: "nim",
      as: "mhs"
    }
  },
  { $unwind: "$mhs" },

  {
    $project: {
      _id: 0,
      nim: "$_id",
      nama_lengkap: "$mhs.namaLengkap",
      total_sks_lulus: 1
    }
  },
```

```
  { $sort: { total_sks_lulus: -1 } },
  { $limit: 5 }
])
```

Item	Value
Row Output	5
Execution Time	1989 ms

Berdasarkan tabel di atas dapat dilihat bahwa dengan waktu eksekusi 1989 ms dengan row output 5 tergolong cukup lambat. Solusinya dapat melakukan denormalisasi dan menyimpan nama lengkap mahasiswa pada embedding rencana studi sehingga tidak perlu dilakukan $lookup pada koleksi mahasiswa.


```
{
  total_sks_lulus: 64,
  nim: '0275407507',
  nama_lengkap: 'Padmi Pratama'
}
{
  total_sks_lulus: 61,
  nim: '4822214531',
  nama_lengkap: 'Teguh Salahudin'
}
{
  total_sks_lulus: 60,
  nim: '6014510438',
  nama_lengkap: 'Wage Waskita'
}
{
  total_sks_lulus: 60,
  nim: '4809153042',
  nama_lengkap: 'drg. Juli Pradipta, S.Sos'
}
{
  total_sks_lulus: 60,
  nim: '9045247238',
  nama_lengkap: 'Unjani Pradipta'
}
```

- **Query 6 :** Query di bawah ini digunakan untuk menampilkan Mata kuliah di kurikulum tetapi yang belum memiliki dosen pengampu

```
db.kurikulum.explain("executionStats").aggregate([
  { $unwind: "$mataKuliah" },
  {
    $lookup: {
      from: "mata_kuliah",
      localField: "mataKuliah.kodeMatkul",
      foreignField: "kodeMatkul",
      as: "mk"
    }
  },
  { $unwind: "$mk" },
  {
    $lookup: {
      from: "program_studi",
      localField: "kodeProdi",
      foreignField: "kodeProdi",
      as: "prodi"
    }
  },
  { $unwind: "$prodi" },
  {
    $lookup: {
      from: "dosen_pengampu",
      let: { kodeProdi: "$kodeProdi", kodeMatkul:
"$mataKuliah.kodeMatkul" },
      pipeline: [
        {
          $match: {
            $expr: {
              $and: [
                { $eq: ["$kodeProdi", "$$kodeProdi"] },
                { $eq: ["$kodeMatkul", "$$kodeMatkul"] }
              ]
            }
          }
        }
```

```
      ],
      as: "pengampu"
    }
  },
  { $match: { pengampu: { $eq: [] } } },
  {
    $project: {
      _id: 0,
      kode_matkul: "$mk.kodeMatkul",
      nama_mk: "$mk.namaMk",
      nama_prodi: "$prodi.namaProdi"
    }
  }
])
```

Item	Value
Row Output	280
Execution Time	400 ms

Berdasarkan tabel di atas dengan execution time 400 ms tergolong baik.

- **Query 7 :** Query di bawah ini digunakan untuk mencari rata-rata nilai per mata kuliah

```
db.rencana_studi.explain("executionStats").aggregate([
  { $unwind: "$mataKuliah" },
  {
    $addFields: {
      bobot: {
        $switch: {
          branches: [
            { case: { $eq: [ "$mataKuliah.nilai", "A" ] },
then: 4.0 },
            { case: { $eq: [ "$mataKuliah.nilai", "AB" ] },
then: 3.5 },
            { case: { $eq: [ "$mataKuliah.nilai", "B" ] },
then: 3.0 },
            { case: { $eq: [ "$mataKuliah.nilai", "BC" ] },
then: 2.5 },
            { case: { $eq: [ "$mataKuliah.nilai", "C" ] },
then: 2.0 },
            { case: { $eq: [ "$mataKuliah.nilai", "D" ] },
then: 1.0 }
          ],
          default: 0.0
        }
      }
    }
  },
  {
    $group: {
      _id: "$mataKuliah.kodeMatkul",
      rata_nilai: { $avg: "$bobot" }
    }
  },
  {
    $lookup: {
      from: "mata_kuliah",
      localField: "_id",
      foreignField: "kodeMatkul",
      as: "mk"
```

```
    }
  },
  { $unwind: "$mk" },
  {
    $project: {
      _id: 0,
      kode_matkul: "$_id",
      nama_mk: "$mk.namaMk",
      rata_nilai: { $round: ["$rata_nilai", 2] }
    }
  },
  { $sort: { rata_nilai: -1 } }
])
```

Item	Value
Row Output	3080
Execution Time	1090 ms

Berdasarkan tabel di atas dengan execution time 1090 ms tergolong lambat. Solusinya dapat melakukan denormalisasi dan menyimpan nama matkul pada embedding rencana studi sehingga tidak perlu dilakukan $lookup pada koleksi mata_kuliah.

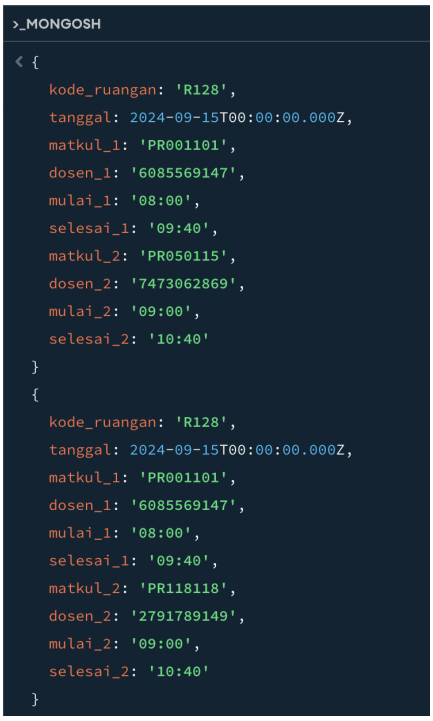
- **Query 8 :** Query di bawah ini digunakan untuk mendeteksi jadwal bentrok di ruangan yang sama pada hari yang sama

```
db.jadwal_kuliah.explain("executionStats").aggregate([
  {
    $lookup: {
      from: "jadwal_kuliah",
      let: {
        ruang: "$kodeRuangan",
        tgl: "$tanggal",
        mulai: "$jamMulai",
        selesai: "$jamSelesai"
      },
      pipeline: [
        {
          $match: {
            $expr: {
              $and: [
                { $eq: ["$kodeRuangan", "$$ruang"] },
                { $eq: ["$tanggal", "$$tgl"] },
                { $gt: ["$jamMulai", "$$mulai"] },
                { $lt: ["$jamMulai", "$$selesai"] }
              ]
            }
          }
        }
      ],
      as: "bentrok"
    }
  },
  { $unwind: "$bentrok" },
  {
    $project: {
      _id: 0,
      kode_ruangan: "$kodeRuangan",
      tanggal: "$tanggal",

      matkul_1: "$kodeMatkul",
```

```
      dosen_1: "$nipDosen",
      mulai_1: "$jamMulai",
      selesai_1: "$jamSelesai",

      matkul_2: "$bentrok.kodeMatkul",
      dosen_2: "$bentrok.nipDosen",
      mulai_2: "$bentrok.jamMulai",
      selesai_2: "$bentrok.jamSelesai"
    }
  }
])
```

Item	Value
Row Output	78178
Execution Time	4463 ms

Berdasarkan tabel di atas dengan execution time 4463 ms tergolong lambat. Solusi nya adalah hindari self $lookup dan pakai $setWindowFields untuk cek bentrok dengan jadwal sebelumnya di ruangan yang sama.



```
db.jadwal_kuliah.aggregate([
  {
    $sort: {
      kodeRuangan: 1,
      tanggal: 1,
      jamMulai: 1
    }
  },
  {
    $setWindowFields: {
      partitionBy: { kodeRuangan: "$kodeRuangan", tanggal:
"$tanggal" },
      sortBy: { jamMulai: 1 },
      output: {
        prevKodeMatkul: { $shift: { output: "$kodeMatkul", by:
-1 } },
        prevNipDosen:   { $shift: { output: "$nipDosen", by: -1
} },
        prevJamMulai:   { $shift: { output: "$jamMulai", by: -1
} },
        prevJamSelesai: { $shift: { output: "$jamSelesai", by:
-1 } }
      }
    }
  },

  // 3) deteksi bentrok:
  {
    $match: {
      prevJamMulai: { $ne: null },             // ada jadwal
sebelumnya
      $expr: { $gt: ["$prevJamSelesai", "$jamMulai"] }
    }
  },
  {
    $project: {
      _id: 0,
      kode_ruangan: "$kodeRuangan",
      tanggal: "$tanggal",

      matkul_1: "$prevKodeMatkul",
      dosen_1: "$prevNipDosen",
      mulai_1: "$prevJamMulai",
      selesai_1: "$prevJamSelesai",
```

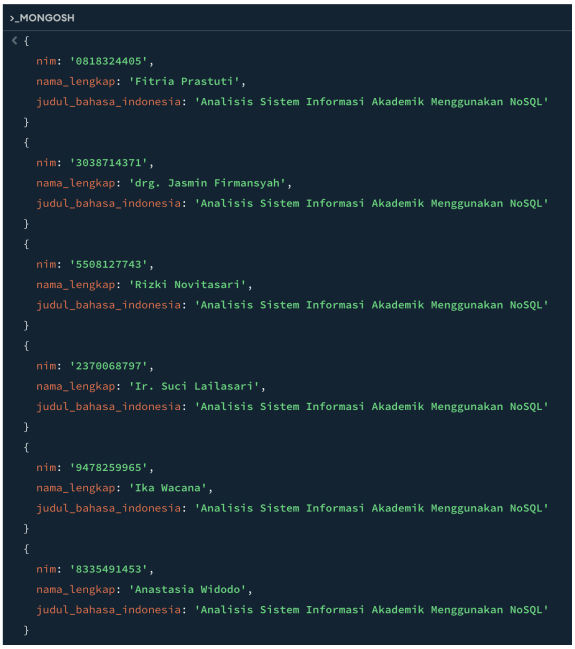
```
      matkul_2: "$kodeMatkul",
      dosen_2: "$nipDosen",
      mulai_2: "$jamMulai",
      selesai_2: "$jamSelesai"
    }
  }
])
```

- **Query 9 :** Query di bawah ini digunakan untuk menampilkan daftar Mahasiswa punya TA tapi belum wisuda

```
db.mahasiswa.explain("executionStats").aggregate([
  {
    $match: {
      tugasAkhir: { $ne: null },
      alumni: { $eq: null }
    }
  },
  {
    $project: {
      _id: 0,
      nim: 1,
      nama_lengkap: "$namaLengkap",
      judul_bahasa_indonesia:
"$tugasAkhir.judulBahasaIndonesia"
    }
  }
])
```

Item	Value
Row Output	2508
Execution Time	59 ms

Berdasarkan tabel di atas dengan execution time 59 ms tergolong sangat baik.



- **Query 10 :** Query di bawah ini digunakan untuk mengetahui dosen yang mengampu MK di atas rata-rata dosen lain

```
db.dosen_pengampu.explain("executionStats").aggregate([
  {
    $group: {
      _id: "$nipDosen",
      mk_set: { $addToSet: "$kodeMatkul" }
    }
  },
  {
    $project: {
      _id: 0,
      nip: "$_id",
      jumlah_mk: { $size: "$mk_set" }
    }
  },
  {
    $facet: {
      perDosen: [
```

```
        { $project: { nip: 1, jumlah_mk: 1 } }
      ],
      avgStat: [
        {
          $group: {
            _id: null,
            avgJumlah: { $avg: "$jumlah_mk" }
          }
        }
      ]
    }
  },
  { $unwind: "$avgStat" },
  {
    $project: {
      perDosen: 1,
      avgJumlah: "$avgStat.avgJumlah"
    }
  },
  { $unwind: "$perDosen" },
  {
    $replaceRoot: {
      newRoot: {
        nip: "$perDosen.nip",
        jumlah_mk: "$perDosen.jumlah_mk",
        avgJumlah: "$avgJumlah"
      }
    }
  },
  {
    $match: {
      $expr: { $gt: ["$jumlah_mk", "$avgJumlah"] }
    }
  },
  {
    $lookup: {
      from: "dosen",
      localField: "nip",
      foreignField: "nip",
      as: "dosen"
```

```
    }
  },
  { $unwind: "$dosen" },
  {
    $project: {
      _id: 0,
      nip: 1,
      nama_dosen: "$dosen.namaDosen",
      jumlah_mk: 1
    }
  },
  { $sort: { jumlah_mk: -1 } }
])
```

Item	Value
Row Output	651
Execution Time	54 ms

Berdasarkan tabel di atas dengan execution time 54 ms tergolong sangat baik.

- **Query 11:** Insert (menambahkan mahasiswa baru)

```
db.calon_mahasiswa_baru.insertOne({
noRegistrasi: "REG2025001",
preferensiBank: "BNI",
sertifikatBahasaInggris: "IELTS",
tesPotensiAkademik: "TPA OTO Bappenas"
});
db.akun.insertOne({
username: "mhs_1352500001",
password: "hashed_password_dummy",
tipe: "mahasiswa"
});
db.mahasiswa.insertOne({
nim: "1352500001",
namaLengkap: "Fedrianz Dharma",
kelas: "A",
tahunMasuk: 2025,
kodeFakultas: "FK01",
kodeProdi: "PR001",
noRegistrasi: "REG2025001",
username: "mhs_1352500001",
dosenWali: null,
```

```
biodata: {
pasFoto: "fedrianz.jpg",
namaDiDokumenKuliah: "Fedrianz Dharma",
jenisKelamin: "L",
tanggalLahir: new Date("2005-01-15T00:00:00Z"),
negaraKelahiran: "Indonesia",
statusPernikahan: "Belum Menikah",
wargaNegara: "Indonesia",
nik: "3273011501050001",
agama: "Islam"
},
alamat: [
{
tipeAlamat: "selama kuliah",
alamat: "Jalan Cisitu Lama No. 10",
kodePos: "40135",
kodeKota: "BDG",
negara: "Indonesia",
noKontak: "022-1234567",
namaKota: "Bandung",
email: "fedrianz.dharma@student.example.com",
noHandphone: "081234567890"
},
```

```
{
tipeAlamat: "tetap",
alamat: "Jalan Melati No. 5",
kodePos: "12345",
kodeKota: "JKT",
negara: "Indonesia",
noKontak: "021-7654321",
namaKota: "Jakarta",
email: "fedrianz.dharma@example.com",
noHandphone: "081298765432"
}
],
orangTua: [
{
nama: "Budi Dharma",
hubungan: "Ayah",
penghasilanKotor: 15000000,
pekerjaan: "Karyawan Swasta",
instansiBekerja: "PT Maju Jaya",
pendidikan: "S1"
},
{
nama: "Sari Lestari",
```

```
hubungan: "Ibu",

penghasilanKotor: 8000000,

pekerjaan: "Ibu Rumah Tangga",

instansiBekerja: null,

pendidikan: "SMA"

}

],

statusKeuangan: [],

tugasAkhir: null,

alumni: null,

calonPesertaWisuda: null,

pengambilanIjazah: []

});
```

Item	Value
Row Output	3
Execution Time	10 ms

Berdasarkan tabel di atas dengan execution time 10 ms tergolong sangat baik.

- **Query 12 :** Insert (menambahkan jadwal mengajar baru)

```
db.jadwal_kuliah.insertOne({
  kodeProdi: "107",
```

```
  tahun: 2024,
  kodeMatkul: "MK1615",
  nipDosen: "197707131299",
  kodeRuangan: "R.GKU.06",
  tanggal: new Date("2025-09-01"),
  jamMulai: "09:00",
  jamSelesai: "10:40",
  kehadiran: []
})
```

Item	Value
Row Output	1
Execution Time	1 ms

Berdasarkan tabel di atas dengan execution time 1 ms tergolong sangat baik.

- **Query 13 :** Update (memperbaharui nilai mahasiswa pada semester dan mata kuliah tertentu)

```
db.rencana_studi.updateOne(
  {
    nim: "18121001",
    tahunAjaran: "2021/2022",
    semester: 2
  },
  {
    $set: {
      "mataKuliah.$[mk].nilai": "A"
    }
  },
  {
    arrayFilters: [
      {
        "mk.kodeProdi": "181",
        "mk.tahun": 2023,
        "mk.kodeMatkul": "MK1927"
      }
```

```
    ]
  }
)
```

Item	Value
Row Output	1
Execution Time	19 ms

Berdasarkan tabel di atas dengan execution time 19 ms tergolong sangat baik.

- **Query 14 :** Update (menghapus mata kuliah dari rencana studi)

```
db.rencana_studi.updateOne(
  {
    nim: "18121001",
    tahunAjaran: "2021/2022",
    semester: 2
  },
  {
    $pull: {
      mataKuliah: {
        kodeProdi: "181",
        tahun: 2023,
        kodeMatkul: "MK1806"
      }
    }
  }
)
```

Item	Value
Row Output	1
Execution Time	17 ms

Berdasarkan tabel di atas dengan execution time 17 ms tergolong sangat baik.

- **Query 15 :** Delete (menghapus ruangan)

```
db.jadwal_kuliah.explain("executionStats").deleteMany({
kodeRuangan: "R005" });

db.ruangan.deleteOne({ kodeRuangan: "R005" });
```

Item	Value
Row Output	241
Execution Time	17 ms

Berdasarkan tabel di atas dengan execution time 17 ms tergolong sangat baik.

1.2 Neo4j

Berikut implementasi query yang didefinisikan sebelumnya pada project 1 :

- **Query 1** : Query di bawah ini digunakan untuk menampilkan daftar mahasiswa + dosen wali + prodi + fakultas yang kemudian diurutkan berdasarkan nama fakultas, program studi dan nama lengkap.

```
MATCH (m:Mahasiswa)-[:DIWALIKAN_OLEH]->(d:Dosen),

(m)-[:DARI_PRODI]->(p:ProgramStudi)-[:BAGIAN_DARI_FAKULTAS]->(
f:Fakultas)
    RETURN
        m.nim,
        m.nama_lengkap,
        d.nama_dosen AS dosen_wali,
        p.nama_prodi,
        f.nama_fakultas
    ORDER BY f.nama_fakultas, p.nama_prodi, m.nama_lengkap
```

```
Total baris: 33296. Waktu eksekusi: 625.88 ms
Menampilkan 5 baris pertama...
1. Ade Dongoran (ME2318696)
    Prodi: Meteorologi, Fakultas Ilmu dan Teknologi Kebumian
    Wali : Zahra Handayani, S.Kom
2. Adhiarja Wastuti (ME2426778)
    Prodi: Meteorologi, Fakultas Ilmu dan Teknologi Kebumian
    Wali : H. Jumadi Kusumo, M.Pd
3. Adiarja Prastuti (ME2323737)
    Prodi: Meteorologi, Fakultas Ilmu dan Teknologi Kebumian
    Wali : Ikhsan Padmasari
4. Adiarja Saptono (ME2300831)
    Prodi: Meteorologi, Fakultas Ilmu dan Teknologi Kebumian
    Wali : Vanesa Yuliarti
5. Adinata Saefullah, M.Pd (ME2115743)
    Prodi: Meteorologi, Fakultas Ilmu dan Teknologi Kebumian
    Wali : Ibun Hartati
```

Berdasarkan gambar di atas, dengan waktu eksekusi 625.88 ms untuk mengambil sebanyak 33296 baris sudah tergolong baik.

- **Query 2** : Query di bawah ini digunakan untuk menampilkan jumlah Mata Kuliah dan total SKS per mahasiswa

```
MATCH (m:Mahasiswa)-[:MEMILIKI_RENCANA_STUDI]->(:RencanaStudi)
          -[menc:MENCANTUMKAN]->(:Kelas)
          -[:ADALAH_INSTANSI_DARI]->(mk:MataKuliah)
    RETURN
        m.nim,
        m.nama_lengkap,
        count(DISTINCT mk) AS jumlah_mk,
        sum(mk.sks) AS total_sks
    ORDER BY total_sks DESC
```

```
Total baris: 33296. Waktu eksekusi: 5225.51 ms
Menampilkan 10 baris pertama...
Hardana Uwais: 50 MK, 177 SKS
Titi Haryanto: 51 MK, 176 SKS
Siska Usamah: 47 MK, 168 SKS
Puji Namaga: 45 MK, 164 SKS
Dr. Nyana Tarihoran: 46 MK, 164 SKS
Rusman Najmudin, S.E.I: 48 MK, 164 SKS
Tari Farida: 46 MK, 164 SKS
Mujur Wastuti: 44 MK, 163 SKS
Unjani Saefullah: 42 MK, 163 SKS
Salwa Handayani: 49 MK, 163 SKS
```

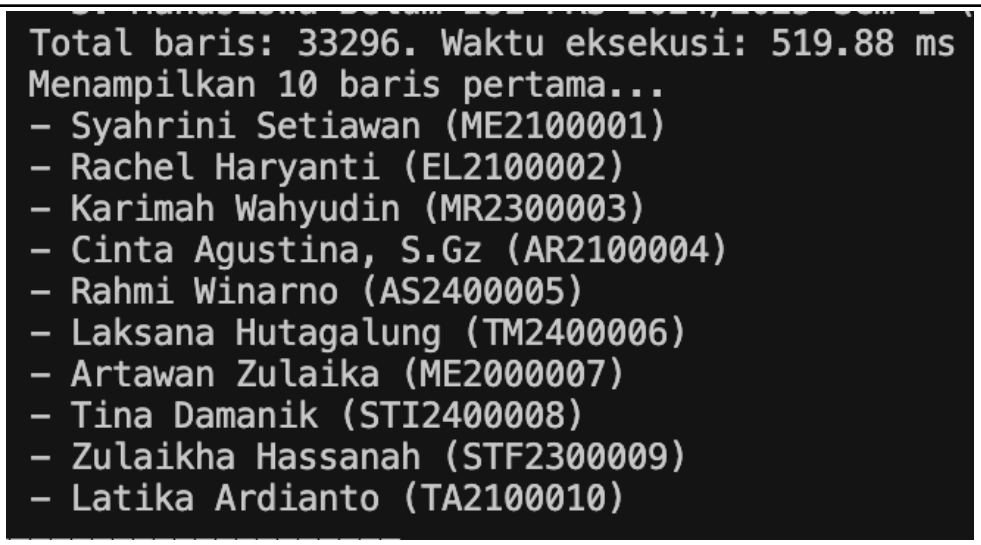
Berdasarkan gambar di atas, dengan waktu eksekusi 5225.51 ms untuk mengambil sebanyak 33296 baris tergolong lambat. Solusinya yakni :

1. Menggunakan alias supaya Neo4j tidak harus scan semua node dengan label tersebut, yang sebelumnya (:RencanaStudi) menjadi >(rs:RencanaStudi)dan (:Kelas) menjadi (k:Kelas)
2. Membuat index untuk mempercepat *start node & grouping*

```
CREATE INDEX mahasiswa_nim_index IF NOT EXISTS
FOR (m:Mahasiswa) ON (m.nim);
```

- **Query 3** : Query di bawah ini digunakan untuk menampilkan daftar mahasiswa yang rencana studinya belum dikirim

```
MATCH (m:Mahasiswa)
    WHERE NOT (m)-[:MEMILIKI_RENCANA_STUDI {tahun_ajaran:
2024/2025, semester: 1}]->()
    RETURN m.nim, m.nama_lengkap
```



Berdasarkan gambar di atas, dengan waktu eksekusi 519.88 ms untuk 33296 ribu row itu termasuk normal dan bagus.

Query 4 : Query di bawah ini digunakan untuk menampilkan jumlah Mata Kuliah dan kelas per dosen (beban mengajar) di tahun 2024

```
MATCH (d:Dosen)-[:MENGAJAR]->(k:Kelas {tahun_ajaran: 2024/2025,
semester: 1})
    MATCH (k)-[:ADALAH_INSTANSI_DARI]->(mk:MataKuliah)
    RETURN
```

```
        d.nip,
        d.nama_dosen,
        count(DISTINCT mk) AS jumlah_mk,
        count(DISTINCT k) AS jumlah_kelas
    ORDER BY jumlah_kelas DESC, jumlah_mk DESC
```

```
--- 4. Beban Mengajar Dosen 2024/2025 Sem 1 (Cetak 10 baris pertama) ---
  Total baris: 3257. Waktu eksekusi: 117.22 ms
  Menampilkan 10 baris pertama...
  H. Martaka Winarno, S.Pt: 11 kelas, 11 MK unik
  Jati Mansur: 9 kelas, 9 MK unik
  Ade Siregar: 9 kelas, 9 MK unik
  Amelia Rajata: 9 kelas, 9 MK unik
  Warta Budiyanto: 9 kelas, 9 MK unik
  Hasta Sihotang: 9 kelas, 9 MK unik
  Kasusra Najmudin: 9 kelas, 9 MK unik
  Bakiman Wasita: 8 kelas, 8 MK unik
  dr. Cager Padmasari, S.T.: 8 kelas, 8 MK unik
  Usyi Widodo: 8 kelas, 8 MK unik
```

Berdasarkan gambar di atas, waktu eksekusi 117.22 ms dengan total baris 3257 tergolong sudah sangat baik.

- **Query 5** : Query di bawah ini digunakan untuk menampilkan 5 mahasiswa teratas dengan SKS lulus terbanyak.

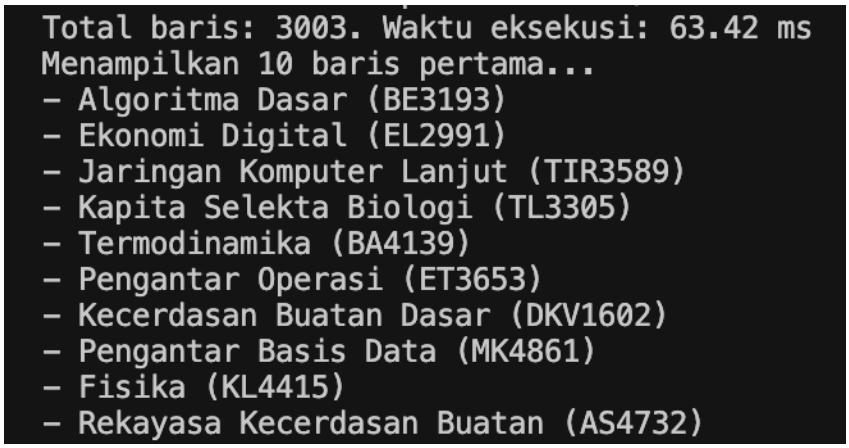
```
MATCH (m:Mahasiswa)
    RETURN
        m.nim,
        m.nama_lengkap,
        m.sks_lulus_materialized AS total_sks_lulus
    ORDER BY total_sks_lulus DESC
    LIMIT 5
```

```
Total baris: 5. Waktu eksekusi: 39.00 ms
Sakura Rahmawati: 144 SKS Lulus
Jaga Kuswoyo: 144 SKS Lulus
Elvina Mustofa, S.Pt: 144 SKS Lulus
Lulut Lestari, S.E.I: 144 SKS Lulus
Dadap Kurniawan: 144 SKS Lulus
```

Gambar di atas menunjukkan dengan waktu eksekusi 39ms dan total baris 5 sangat baik dan sudah optimal.

- **Query 6** : Query di bawah ini digunakan untuk menampilkan mata kuliah di kurikulum tetapi belum memiliki dosen pada tahun ajaran 2025/2026

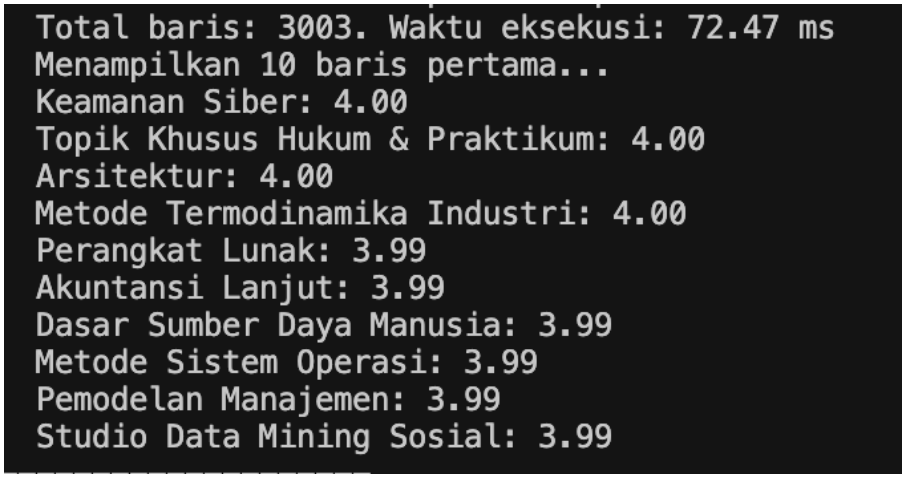
```
WITH "2024/2025" AS ta, 1 AS sem

MATCH (mk:MataKuliah)
WHERE NOT (mk)<-[:ADALAH_INSTANSI_DARI]-(:Kelas {tahun_ajaran:
ta, semester: sem})

WITH mk, ta, sem LIMIT 5

CREATE (k:Kelas {tahun_ajaran: ta, semester: sem})
CREATE (k)-[:ADALAH_INSTANSI_DARI]->(mk)

RETURN mk.kode_matkul, mk.nama_mk AS Created_Class_For
```



Gambar di atas menunjukkan bahwa waktu eksekusi 63.42 ms dengan total baris 3003 tergolong sangat baik.

- **Query 7** : Query di bawah ini digunakan untuk menampilkan rata-rata nilai per mata kuliah

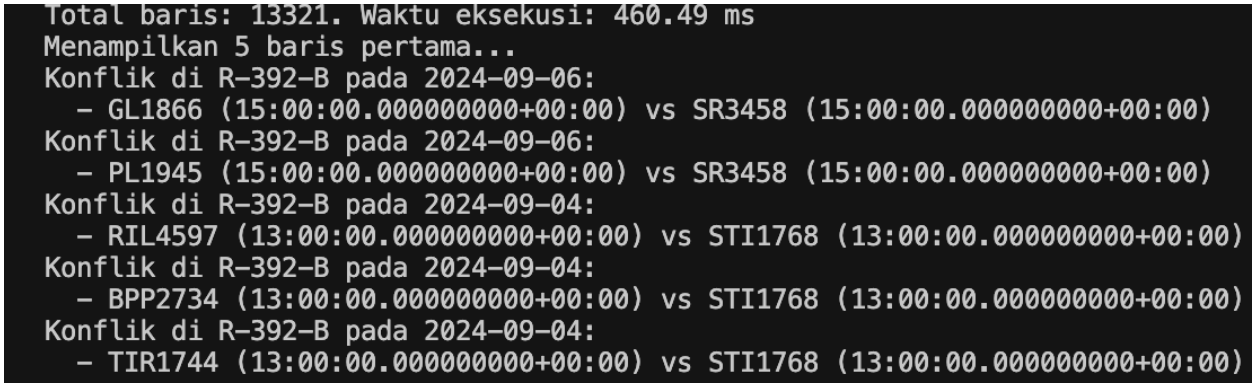
```
MATCH (mk:MataKuliah)
    WHERE mk.rata_nilai_materialized IS NOT NULL
    RETURN
        mk.kode_matkul,
        mk.nama_mk,
        mk.rata_nilai_materialized AS rata_nilai
    ORDER BY rata_nilai DESC
```



Gambar di atas menunjukkan bahwa waktu eksekusi 72.7 ms dengan total baris 3003 tergolong baik/normal.

- **Query 8** : Query di bawah ini digunakan untuk mendeteksi jadwal bentrok di ruangan yang sama pada hari yang sama

```
MATCH
(:Kelas)-[a:DIJADWALKAN_DI]->(r:Ruangan)<-[b:DIJADWALKAN_DI]-(:
Kelas)
    WHERE elementId(a) < elementId(b)
      AND a.tanggal = b.tanggal
      AND a.jam_mulai < b.jam_selesai
      AND a.jam_selesai > b.jam_mulai
    RETURN
      r.kode_ruangan AS Ruangan,
      a.tanggal AS Tanggal,
      a.kode_matkul AS matkul_1,
      b.kode_matkul AS matkul_2,
      a.jam_mulai AS mulai_1,
      b.jam_mulai AS mulai_2
```



Gambar di atas menunjukkan bahwa waktu eksekusi 460.49 ms dengan total baris 13321 tergolong kurang baik, karena :

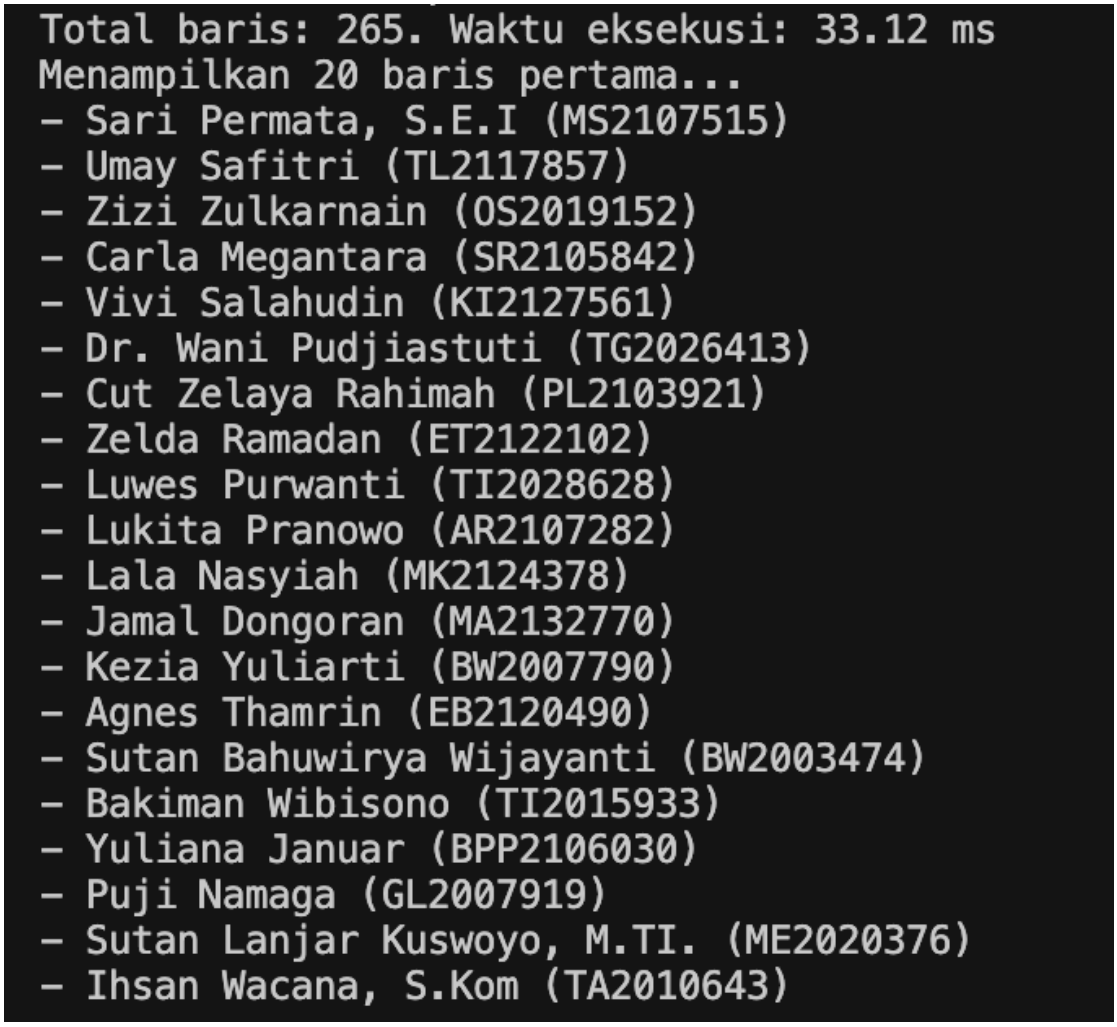
1. No Index Usage : properti tanggal, jam_mulai, jam_selesai tidak terindeks
2. Full Relationship Scan : semua relasi DIJADWALKAN_DI di-scan tanpa filter awal

Solusi : filter by tanggal terlebih dahulu

```
MATCH (r:Ruangan)<-[a:DIJADWALKAN_DI]-(ka:Kelas)
WITH r, a.tanggal AS tanggal, collect({rel: a, kelas: ka}) AS
jadwals
WHERE size(jadwals) > 1
UNWIND jadwals AS j1
UNWIND jadwals AS j2
WITH r, tanggal, j1, j2
WHERE elementId(j1.rel) < elementId(j2.rel)
  AND j1.rel.jam_mulai < j2.rel.jam_selesai
  AND j1.rel.jam_selesai > j2.rel.jam_mulai
RETURN
  r.kode_ruangan AS Ruangan,
  tanggal AS Tanggal,
  j1.rel.kode_matkul AS matkul_1,
  j2.rel.kode_matkul AS matkul_2,
  j1.rel.jam_mulai AS mulai_1,
  j2.rel.jam_mulai AS mulai_2
```

- **Query 9** : Query di bawah ini digunakan untuk menampilkan data mahasiswa yang telah yudisium namun belum wisuda

```
MATCH (m:Mahasiswa)-[:MENGERJAKAN_TA]->(:TugasAkhir)
    WHERE NOT (m)-[:MENJADI_ALUMNI]->(:DataAlumni)
    RETURN m.nim, m.nama_lengkap
```



Gambar di atas menunjukkan bahwa waktu eksekusi 33.12 ms dengan total baris 265 tergolong baik.

- **Query 10** : Query di bawah ini digunakan untuk menampilkan data dosen yang mengampu mata kuliah di atas rata-rata dosen lain

```
MATCH
(d:Dosen)-[:MENGAJAR]->(:Kelas)-[:ADALAH_INSTANSI_DARI]->(mk:Ma
taKuliah)
    WITH d, count(DISTINCT mk) AS jumlah_mk

    WITH avg(jumlah_mk) AS rataan_mk, collect({dosen: d, count:
jumlah_mk}) AS dosen_data

    UNWIND dosen_data AS data
    WITH data.dosen AS dosen, data.count AS jumlah_mk_dosen,
rataan_mk
```

```
    WHERE jumlah_mk_dosen > rataan_mk

    RETURN dosen.nama_dosen, jumlah_mk_dosen, rataan_mk
    ORDER BY jumlah_mk_dosen DESC
```

Gambar di atas menunjukkan bahwa waktu eksekusi 77.62 ms dengan total baris 1662 tergolong baik.

- **Query 11** : Query di bawah ini digunakan untuk menambahkan mahasiswa baru

```
MATCH (p:ProgramStudi {kode_prodi: "TL"})
MATCH (d:Dosen {nip: "1986235502"})

CREATE (m:Mahasiswa {
    nim: "IF13521096",
    nama_lengkap: "Fedrianz Dharma",
    kelas: 'Ganesha',
    tahun_masuk: 2025,
    ipk_materialized: 0.0,
    sks_lulus_materialized: 0,
```

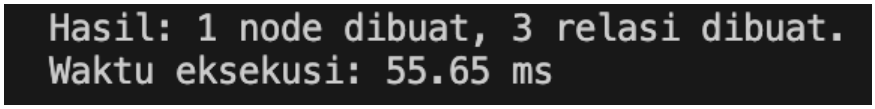
```
    alamat_list: '{"jenis": "Asal", "alamat": "Bandung,
Simpang Dago"}',
    orang_tua_list: '{"nama": "Mama Fedrianz", "peran":
"Ibu"}'
})

CREATE (a:Akun {username: "Akun_Fedrianz", password_hash:
'new_hash', tipe: 'mahasiswa'})

MERGE (m)-[:DARI_PRODI]->(p)
MERGE (m)-[:DIWALIKAN_OLEH]->(d)
MERGE (m)-[:MEMILIKI_AKUN]->(a)
RETURN m
```

```
Hasil: 2 node dibuat, 3 relasi dibuat.
Waktu eksekusi: 59.14 ms
```

Gambar di atas menunjukkan bahwa waktu eksekusi 59.14 ms dengan 2 node dan 3 relasi tergolong cukup baik.

- **Query 12** : Query di bawah ini digunakan untuk melakukan Insert sesi kelas baru

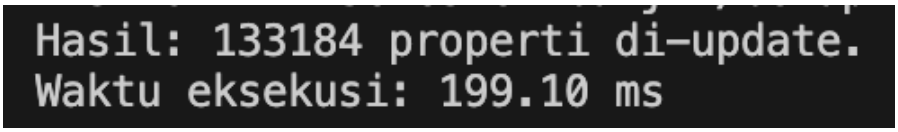
```
MATCH (mk:MataKuliah {kode_matkul: "EL2991"})
MATCH (d:Dosen {nip: "1986235502"})
MATCH (r:Ruangan {kode_ruangan: "R-885-C"})

CREATE (k:Kelas {
    id_kelas_semester: "TEST-CLASS-02",
    tahun_ajaran: "2025/2026",
    semester: 1,
    nama_kelas: 'K-99'
})

MERGE (k)-[:ADALAH_INSTANSI_DARI]->(mk)
MERGE (d)-[:MENGAJAR]->(k)
MERGE (k)-[j:DIJADWALKAN_DI]->(r)

SET j.tanggal = date("2025-09-01"),
    j.jam_mulai = time("09:00:00"),
    j.jam_selesai = time("11:00:00"),
    j.kode_matkul = "EL2991",
    j.nip_dosen = "1986235502"
```



Gambar di atas menunjukkan bahwa waktu eksekusi 55.65 ms dengan 1 node dan 3 relasi tergolong cukup baik, akan lebih optimal jika menambahkan indexs.

- **Query 13** : Query di bawah ini digunakan untuk UPDATE Semester Ganjil/Genap (Memproses 133k+ Node)

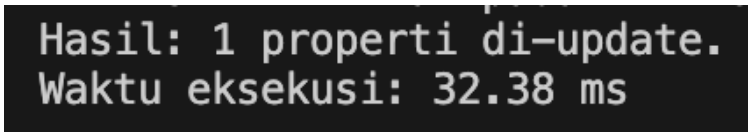
```
MATCH (rs:RencanaStudi)
    SET rs.semester = CASE
        WHEN rs.semester % 2 = 0 THEN 2
        ELSE 1
    END
```



Gambar di atas menunjukkan bahwa waktu eksekusi 55.65 ms dengan 1 node dan 3 relasi tergolong cukup baik, akan lebih optimal jika menambahkan indexs.

- **Query 15 :** Query di bawah ini digunakan untuk melakukan Update nilai pada 1 enrollment

```
MATCH (m:Mahasiswa {nim: "TF2300001"})
          -[:MEMILIKI_RENCANA_STUDI]->(:RencanaStudi)
          -[r:MENCANTUMKAN]->(:Kelas)
WITH r LIMIT 1
SET r.nilai = 'A+'
```



Gambar di atas menunjukkan bahwa waktu eksekusi 32.38 ms tergolong baik.

- **Query 16 :** Query di bawah ini digunakan untuk DELETE 10 Enrollment (Mahasiswa: TF2300001, Sem: 2023/2024-1)

```
MATCH (m:Mahasiswa {nim:
```

```
"TF2300001"})-[:MEMILIKI_RENCANA_STUDI]->(rs:RencanaStudi
{tahun_ajaran: "2023/2024", semester: 1})

MATCH (rs)-[r:MENCANTUMKAN]->(:Kelas)

WITH r LIMIT 10
DELETE r
```

```
Hasil: 7 relasi dihapus.
Waktu eksekusi: 53.35 ms
```

Gambar di atas menunjukkan bahwa waktu eksekusi 53.35 ms dengan hasil 7 relasi dihapus tergolong cukup baik.

2. Sample Data

a. Mongo DB

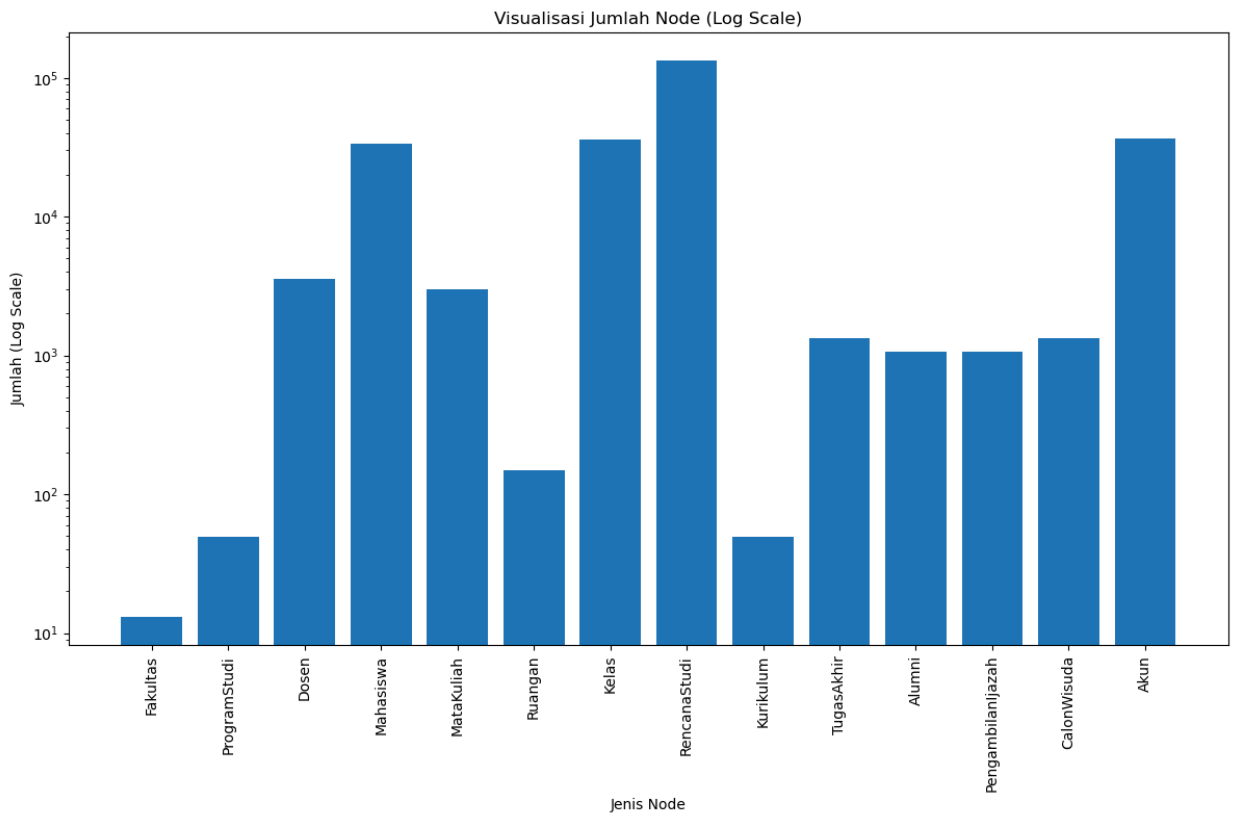
Sample data yang digunakan sama seperti pada project 1 yakni :

Collection name	Properties	Storage size	Documents	Avg. document size	Indexes	Total index size
akun	-	872.45 kB	37K	106.00 B	1	389.12 kB
calon_mahasiswa_baru	-	36.86 kB	100	146.00 B	1	36.86 kB
dosen	-	704.51 kB	3.5K	246.00 B	2	241.66 kB
dosen_pengampu	-	192.51 kB	2.8K	117.00 B	3	282.62 kB
fakultas	-	32.77 kB	13	74.00 B	2	69.63 kB
jadwal_kuliah	-	42.01 MB	34K	1.55 kB	3	1.85 MB
kurikulum	-	73.73 kB	140	1.08 kB	1	36.86 kB
mahasiswa	-	51.77 MB	33K	2.15 kB	2	1.75 MB
mata_kuliah	-	405.50 kB	3.1K	191.00 B	2	200.70 kB
program_studi	-	36.86 kB	140	153.00 B	2	73.73 kB
rencana_studi	-	16.98 MB	133K	906.00 B	3	10.80 MB
ruangan	-	45.06 kB	150	133.00 B	1	36.86 kB

b. Neo4j

Database di seeding dengan jumlah sample data yang setara dengan jumlah sample data pada Postgres yang dilakukan pada project 1.

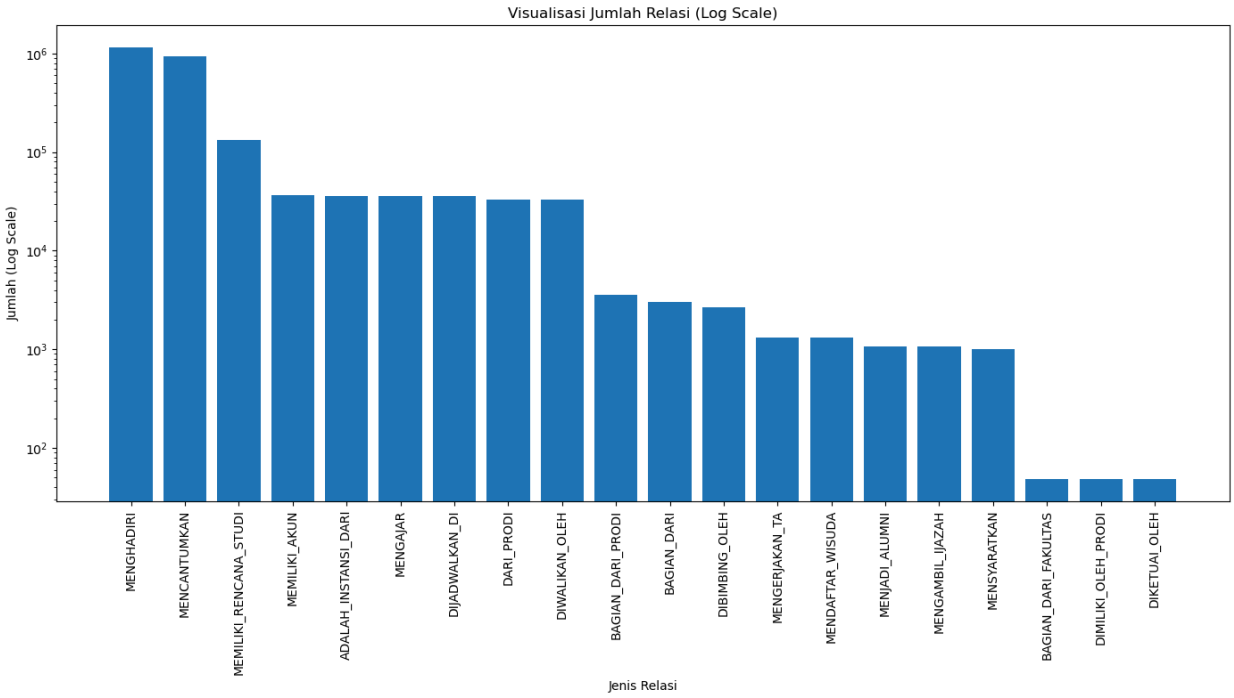
Node	Jumlah
RencanaStudi	133.184
Akun	36.836
Kelas	36.036
Mahasiswa	33.296
Dosen	3.540
MataKuliah	3.003
TugasAkhir	1.332
CalonPesertaWisuda	1.332
DataAlumni	1.065
PengambilanIjazah	1.065
Ruangan	150
Kurikulum	48
ProgramStudi	48
Fakultas	13
RencanaStudi	133.184
Akun	36.836
Kelas	36.036
Mahasiswa	33.296



Grafik di atas merupakan visualisasi jumlah node (*Log Scale*). RencanaStudi memiliki jumlah node terbanyak mencapai 10^5 yaitu lebih dari 100.000. Sedangkan fakultas memiliki jumlah node paling sedikit.

Relasi	Jumlah
MENGHADIRI	1.162.476
MENCANTUMKAN	932.439
MEMILIKI_RENCANA_STUDI	133.184
MEMILIKI_AKUN	36.836
ADALAH_INSTANSI_DARI	36.036
MENGAJAR	36.036
DIJADWALKAN_DI	36.036
DARI_PRODI	33.296

Relasi	Jumlah
DIWALIKAN_OLEH	33.296
BAGIAN_DARI_PRODI	3.540
BAGIAN_DARI	3.003
DIBIMBING_OLEH	2.664
MENGERJAKAN_TA	1.332
MENDAFTAR_WISUDA	1.332
MENJADI_ALUMNI	1.065
MENGAMBIL_IJAZAH	1.065
MENSYARATKAN	1.000
MENGHADIRI	1.162.476
BAGIAN_DARI_FAKULTAS	48
DIMILIKI_OLEH_PRODI	48
DIKETUAI_OLEH	48



Grafik di atas merupakan visualisasi jumlah relasi (*Log Scale*). Menghadiri menjadi jumlah relasi tertinggi.

3. Pemilihan Desain

3.1 Mongo DB

Koleksi	Embedded	Referensi ke Koleksi Lain
fakultas	–	–
program_studi	–	kodeFakultas → fakultas
dosen	usernames[]	kodeFakultas → fakultas
akun	–	-

Koleksi	Embedded	Referensi ke Koleksi Lain
mahasiswa	biodata, alamat[], orangTua[], statusKeuangan[], tugasAkhir, alumni, calonPesertaWisuda, pengambilanIjazah[]	kodeFakultas → fakultas; kodeProdi → program_studi; dosenWali → dosen.nip; username → akun.username; noRegistrasi → calon_mahasiswa_baru
calon_mahasiswa_baru	–	–
mata_kuliah	–	kodeProdi → program_studi
kurikulum	mataKuliah[]	kodeProdi → program_studi; mataKuliah.kodeMatkul → mata_kuliah
rencana_studi	mataKuliah[]	nim → mahasiswa; mataKuliah.kodeMatkul → mata_kuliah; mataKuliah.kodeProdi → program_studi
dosen_pengampu	–	kodeProdi → program_studi; kodeMatkul → mata_kuliah; nipDosen → dosen.nip
ruangan	–	–
jadwal_kuliah	kehadiran[]	kodeRuangan → ruangan; kodeProdi → program_studi; kodeMatkul → mata_kuliah; nipDosen → dosen.nip; kehadiran.nim → mahasiswa

Alasan pemilihan desain antara lain :

1. Embed untuk relasi 1 - N yang *strongly-owned* dan query-nya selalu dari parent, contoh
    - mahasiswa → biodata, alamat, orangTua, statusKeuangan, tugasAkhir, alumni, calonPesertaWisuda, pengambilanIjazah
    - rencana_studi → mataKuliah
    - jadwal_kuliah → kehadiran

Alasan:

- Mengurangi jumlah $lookup / join di Mongo.
- Membuat dokumen self-contained, cocok untuk use-case “lihat profil mahasiswa / jadwal kuliah”.

2. Reference untuk entitas utama yang dipakai lintas konteks, contoh:
   - kodeProdi, kodeFakultas, kodeMatkul, nipDosen, nim

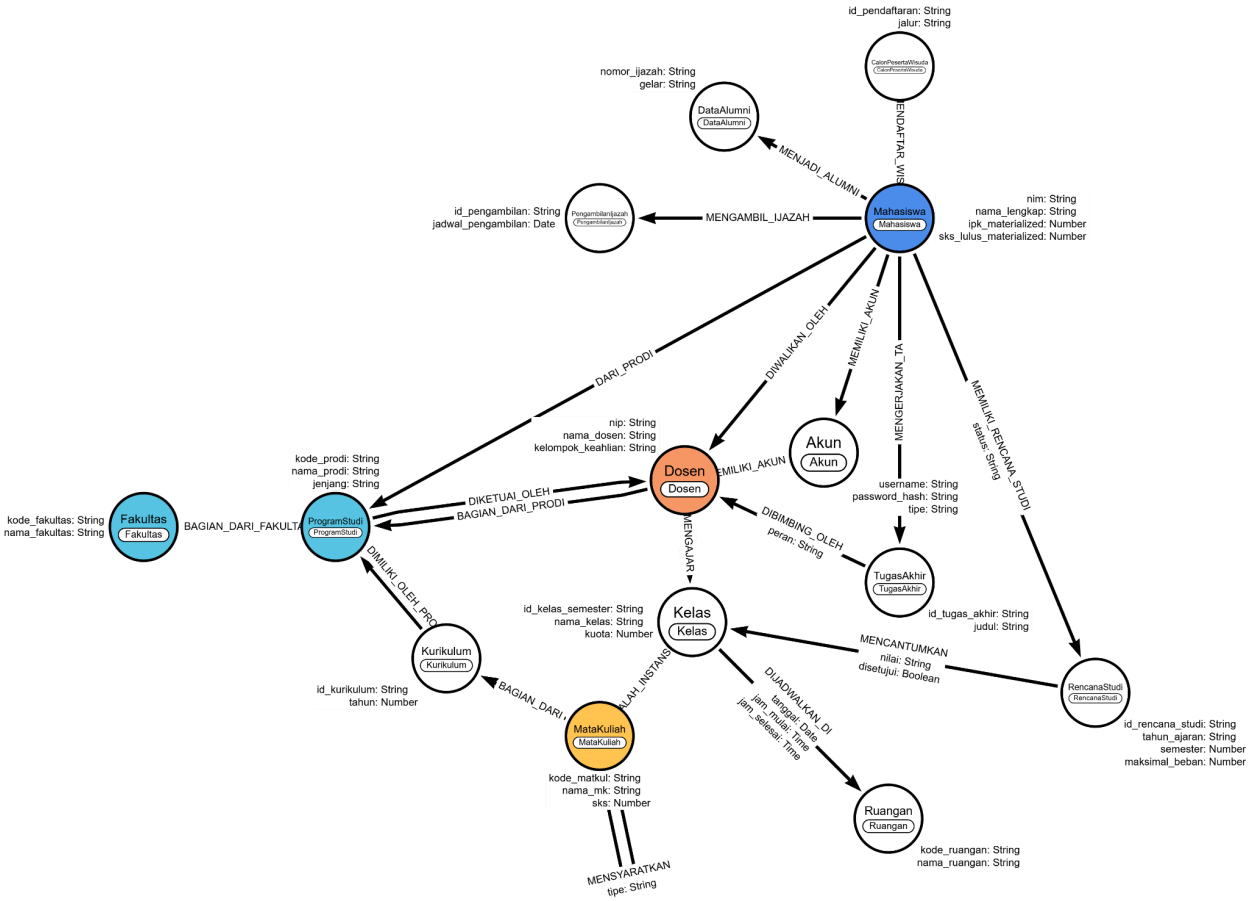
   Alasan:

   - Menghindari duplikasi besar (data prodi/fakultas/matkul/dosen).
   - Memudahkan perubahan global (ganti nama prodi, matkul, dll. cukup di satu koleksi).

3. Denormalisasi terkontrol, contoh: SKS matkul disimpan di:
   - Mata_kuliah.sks
   - disalin ke kurikulum.mataKuliah[].sks dan rencana_studi.mataKuliah[].sks

   Alasan: mempercepat query seperti “total SKS lulus” tanpa join berat.

3.2 Neo4j



4. Reference

- https://neo4j.com/docs/
- https://www.mongodb.com/docs/
- https://www.elastic.co/virtual-events/getting-started-elasticsearch
- https://docs.arango.ai/arangodb/stable/get-started/
- https://redis.io/docs/latest/
- https://cassandra.apache.org/doc/latest/
- https://docs.docker.com/get-started/

5. **Pembagian Tugas**

NIM	Nama	Pembagian
13522090	Fedrianz Dharma	Studi literatur Cassandra, Memperbaiki desain Graph, Implementasi Neo4j, Mengenerate sample data
23525020	Era Desti Ramayani	Studi literatur Elasticsearch, justifikasi tambahan mongoDB dan ArangoDB terkait AQL, Implementasi MongoDB
23525034	Kayis Shalahuddin	Studi literatur Neo4j, Membuat desain Graph, Implementasi Neo4j
23525046	Fadhil Rausyanfikr	Studi literatur mongoDB, Justifikasi MongoDB, Implementasi MongoDB, generate sample data
23525060	Wilson Tansil	Studi literatur, justifikasi, dan implementasi *setup ArangoDB,* Implementasi Neo4j

6. **Link Video**

   https://drive.google.com/file/d/176sj30OcS3bsvOqGhnR-8XKFJftjcU-U/view?usp=sharing

7. **Link PPT**
   https://docs.google.com/presentation/d/1UjhcetWiIdTWLvkhFriGM3z3J-2ipfsL/edit?usp=sharing&ouid=112969377815324423828&rtpof=true&sd=true