MODUL 9
MENGENAL GRAPH DATABASE

9.1. Tujuan Praktikum

1. Mahasiswa memahami konsep database Graph
2. Mahasiswa mampu menginstall Neo4j
3. Mahasiswa mampu mengimplementasikan query dasar pada database Graph

9.2. Dasar Teori

9.2.1. Graph Database

Graf adalah representasi bergambar dari sekumpulan objek di mana beberapa pasangan objek dihubungkan dengan tautan. Graf terdiri dari dua elemen - node (simpul) dan hubungan (sisi).

Basis data graf adalah basis data yang digunakan untuk memodelkan data dalam bentuk graf. Di sini, node dari sebuah graf menggambarkan entitas sedangkan relasi menggambarkan hubungan dari node-node tersebut.

Saat ini, sebagian besar data ada dalam bentuk hubungan antara objek yang berbeda dan lebih sering, hubungan antara data lebih berharga daripada data itu sendiri. Basis data relasional menyimpan data yang sangat terstruktur yang memiliki beberapa catatan yang menyimpan jenis data yang sama sehingga dapat digunakan untuk menyimpan data terstruktur dan, mereka tidak menyimpan hubungan antara data. Tidak seperti basis data lainnya, basis data graph menyimpan hubungan dan koneksi sebagai entitas kelas satu.

Model data untuk basis data graph lebih sederhana dibandingkan dengan basis data lainnya dan, mereka dapat digunakan dengan sistem OLTP. Mereka menyediakan fitur-fitur seperti integritas transaksional dan ketersediaan operasional.

9.2.2. RDBMS Vs Graph Database

Berikut ini adalah tabel yang membandingkan database Relasional dan database Graph:

No	***RDMBS***	**Graph Database**
1	Tabel	Graph
2	Baris	Nodes(simpul)
3	Columns dan Data	Properties dan nilai

4	Contraints	Relationship
5	Joins	Traversal

9.2.3. Neo4j

Neo4j adalah sebuah sistem manajemen basis data graph yang populer. Dikembangkan oleh Neo4j, Inc., Neo4j menyediakan struktur penyimpanan dan pemrosesan data yang didasarkan pada model graph.

Dalam Neo4j, data diorganisasi dalam bentuk graph yang terdiri dari simpul (node) yang mewakili entitas dan relasi (edge) yang mewakili hubungan antara entitas tersebut. Setiap simpul dapat memiliki atribut yang menyimpan informasi tambahan tentang entitas, sedangkan setiap relasi dapat memiliki atribut yang mendefinisikan sifat dan detail dari hubungan tersebut.

Keuntungan menggunakan Neo4j sebagai basis data graph adalah kemampuannya untuk mengelola dan menganalisis data yang memiliki struktur hubungan yang kompleks. Neo4j menyediakan kemampuan pencarian yang kuat, navigasi graph yang efisien, dan algoritma graph yang dapat digunakan untuk menganalisis dan mendapatkan wawasan dari data graph.

Neo4j digunakan secara luas dalam berbagai aplikasi dan industri, termasuk media sosial, e-commerce, keamanan, analisis jaringan, ilmu pengetahuan, dan banyak lagi. Platform ini menyediakan API dan bahasa kueri bernama Cypher yang digunakan untuk mengakses dan memanipulasi data dalam basis data graph Neo4j.

9.2.4. Kelebihan dari Neo4j

- **Model data yang fleksibel**: Neo4j menyediakan model data yang fleksibel, sederhana, namun kuat, yang dapat dengan mudah diubah sesuai dengan aplikasi dan industri.
- **Real-time insight**: Neo4j memberikan hasil berdasarkan data real-time.
- **Ketersediaan tinggi**: Neo4j sangat tersedia untuk aplikasi real-time perusahaan besar dengan jaminan transaksional.
- **Data yang saling terhubung dan semi struktur**: Dengan menggunakan Neo4j, Anda dapat dengan mudah merepresentasikan data yang terhubung dan semi terstruktur.
- **Pengambilan yang mudah**: Dengan menggunakan Neo4j, Anda tidak hanya dapat merepresentasikan tetapi juga dengan mudah mengambil (melintasi / menavigasi) data yang terhubung dengan lebih cepat jika dibandingkan dengan database lain.

- **Bahasa kueri Cypher**: Neo4j menyediakan bahasa kueri deklaratif untuk merepresentasikan graph secara visual, menggunakan sintaks ascii-art. Perintah-perintah dari bahasa ini dalam format yang dapat dibaca oleh manusia dan sangat mudah untuk dipelajari.
- **Tanpa penggabungan(joins)**: Dengan menggunakan Neo4j, TIDAK memerlukan gabungan yang rumit untuk mengambil data yang terhubung/berhubungan karena sangat mudah untuk mengambil simpul yang berdekatan atau detail hubungan tanpa gabungan atau indeks.

9.2.5. Fitur dari Neo4j

Berikut ini adalah sejumlah fitur penting dari Neo4j:

- **Data model (flexible schema)** : Neo4j mendukung aturan ACID (Atomisitas, Konsistensi, Isolasi, dan Daya Tahan) secara penuh.
- **Skalabilitas dan keandalan**: Anda dapat menskalakan basis data dengan meningkatkan jumlah pembacaan/penulisan, dan volume tanpa mempengaruhi kecepatan pemrosesan kueri dan integritas data. Neo4j juga menyediakan dukungan untuk replikasi untuk keamanan dan keandalan data.
- **Bahasa Query Cypher**: Neo4j menyediakan bahasa kueri deklaratif yang kuat yang dikenal sebagai Cypher. Bahasa ini menggunakan seni ASCII untuk menggambarkan graph. Cypher mudah dipelajari dan dapat digunakan untuk membuat dan mengambil relasi antar data tanpa menggunakan query yang rumit seperti Join.
- **Aplikasi web bawaan**: Neo4j menyediakan aplikasi web bawaan Neo4j Browser. Dengan menggunakan ini, Anda dapat membuat dan melakukan kueri terhadap data graph Anda.
- **Driver**: Neo4j dapat bekerja dengan -
    - REST API untuk bekerja dengan bahasa pemrograman seperti Java, Spring, Scala, dll.
    - Java Script untuk bekerja dengan kerangka kerja UI MVC seperti Node JS.
    - Mendukung dua jenis Java API: API Cypher dan API Java asli untuk mengembangkan
- **Aplikasi Java.**
- Selain itu, Anda juga dapat bekerja dengan database lain seperti MongoDB, Cassandra, dll.

- **Pengindeksan**: Neo4j mendukung Indeks dengan menggunakan Apache Lucence.

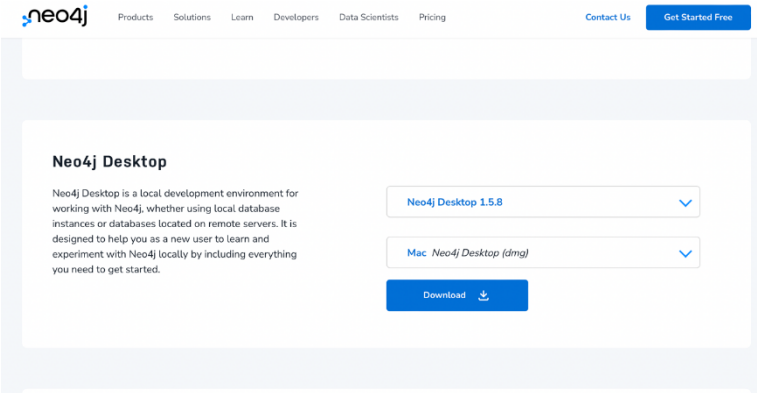
9.3. *Software*

1. Neo4j

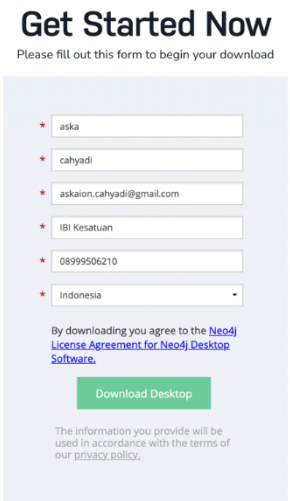
9.4. Tahapan Kerja

9.4.1. Instalasi Neo4j

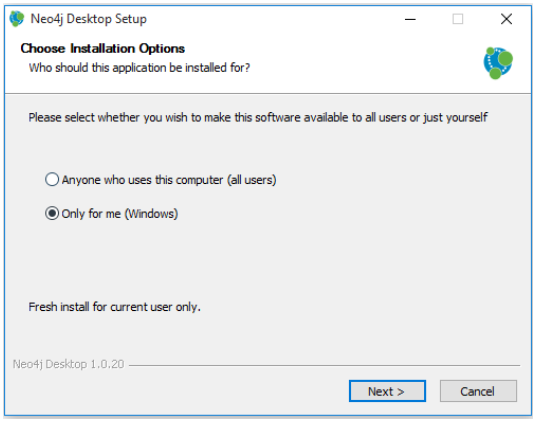
1. Kunjungi website resmi Neo4j https://neo4j.com/download/. (sesuaikan dengan jenis OS kalian)



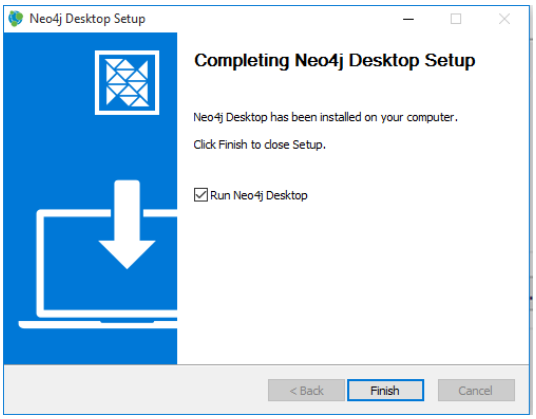
2. Registrasi untuk bisa mendownload Neo4j Desktop.



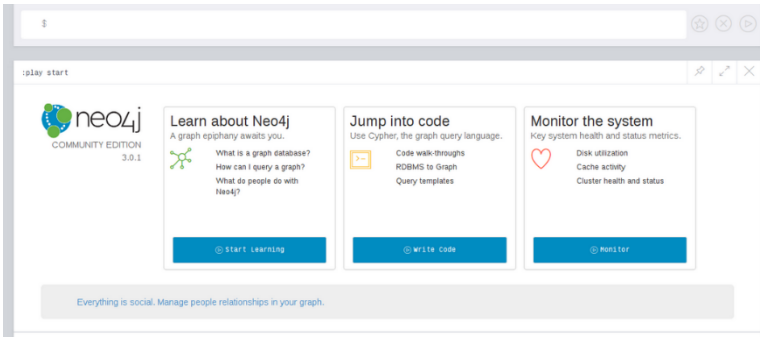
3. Double click file yang telah di download. Lalu install



4. Lalu akan muncul tampilan seperti dibawah yang menandakan telah berhasil install.



5. Setelah aplikasi sudah berhasil terinstall, lalu buka link alamat berikut http://localhost:7474/ lalu akan muncul tampilan seperti dibawah atau kalian bisa buka aplikasi Noe4j desktop yang sudah di download.



9.4.2. Membuat Graph

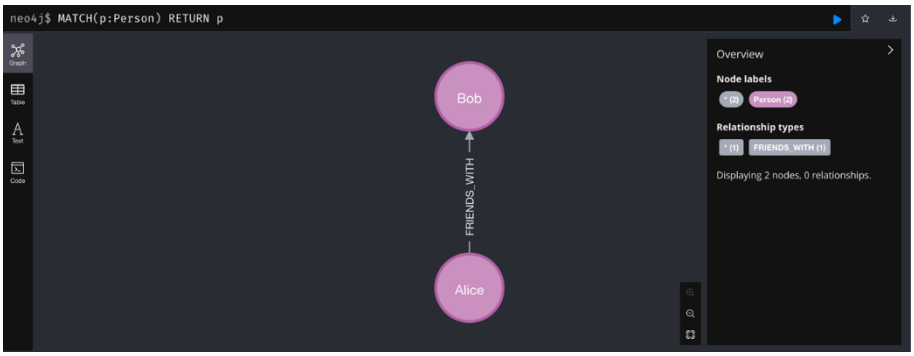
1. Buka Neo4j Browser dan jalankan perintah berikut untuk membuat simpul dan relasi baru dalam graph:

```
CREATE (p1:Person {name: 'Alice', age: 30})
CREATE (p1:Person {name: 'Bob', age: 30})
CREATE (p)-[:FRIENDS_WITDH]->(p2)
```

```
Added 2 labels, created 2 nodes, set 4 properties, created 1 relationship, completed after 32 ms.
```

2. Melakukan Pencarian. Gunakan perintah **MATCH** untuk melakukan pencarian dalam graph. Misalnya, untuk mencari semua simpul dengan label "Person":

```
MATCH (p:Person)
RETURN p
```

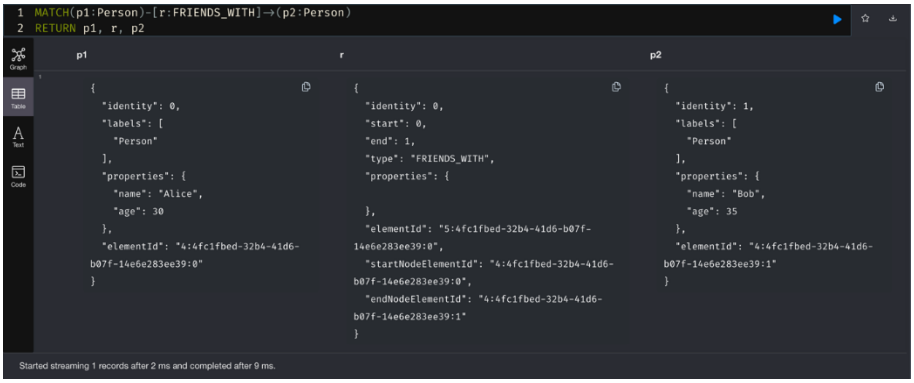


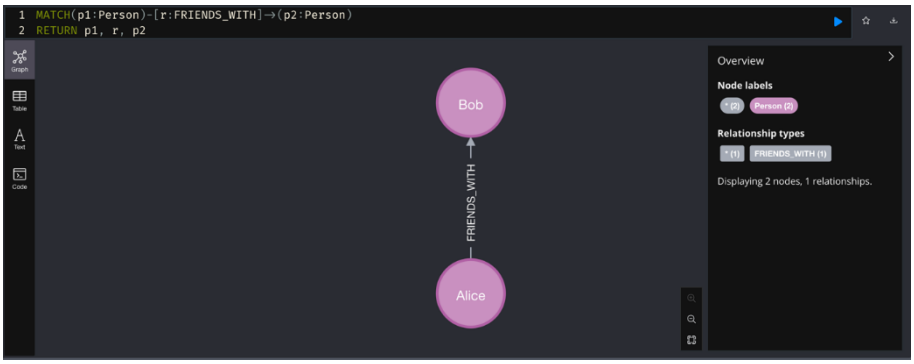
3. Melakukan Pencarian dengan Kondisi. Gunakan perintah **WHERE** untuk menambahkan kondisi pada pencarian. Misalnya, untuk mencari simpul dengan nama "Alice":

```
MATCH (p:Person)
WHERE p.name = 'Alice'
RETURN p
```

4. Mencari Relasi. Gunakan pola relasi dalam perintah **MATCH** untuk mencari relasi antara simpul. Misalnya, untuk mencari relasi "FRIENDS_WITH":

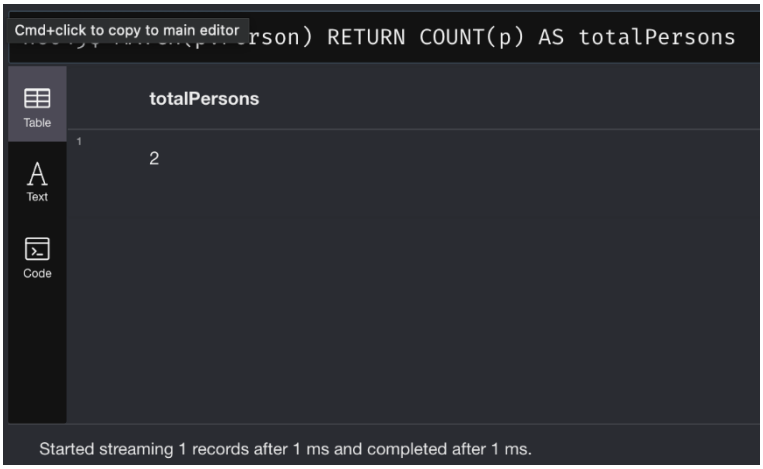
```
MATCH (p:Person)-[r:FRIENDS_WITH]->(p2:Person)
RETURN p1, r, p2
```





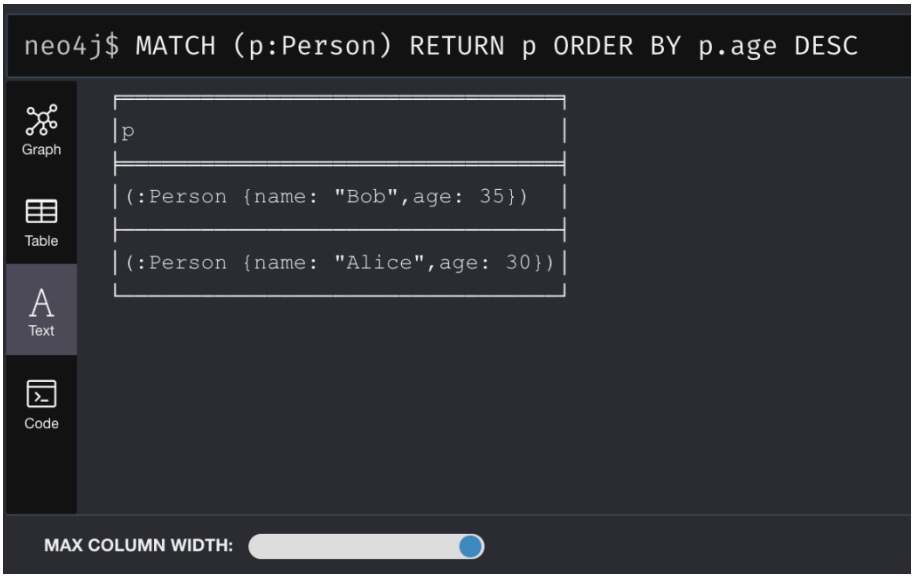
5. Menggunakan Fungsi Agregasi. Gunakan fungsi agregasi seperti **COUNT**, **SUM**, **AVG**, dll., untuk melakukan perhitungan pada hasil kueri. Misalnya, untuk menghitung jumlah simpul dengan label "Person":

```
MATCH (p:Person)
RETURN COUNT(p) as totalPersons
```



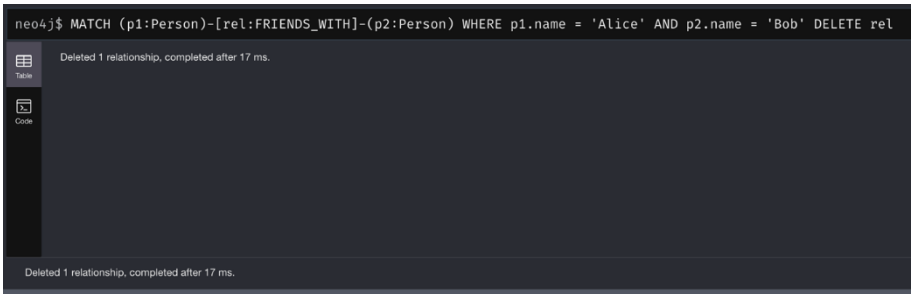
6. Mengurutkan dan Membatasi Hasil. Gunakan perintah **ORDER BY** untuk mengurutkan hasil kueri berdasarkan properti tertentu. Misalnya, untuk mengurutkan simpul berdasarkan umur secara menurun:

```
MATCH (p:Person)
RETURN p
ORDER BY p.age DESC
```



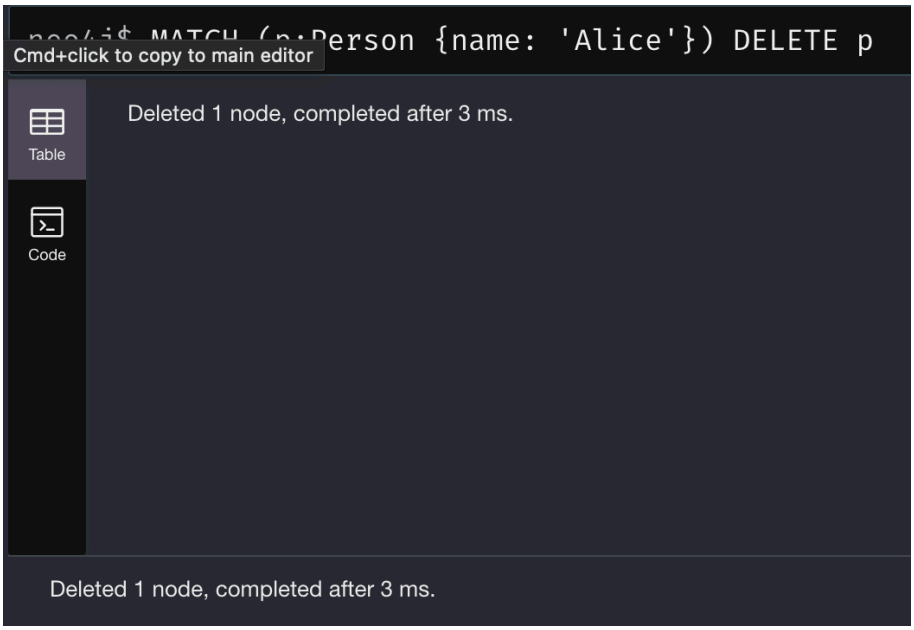
7. Menghapus relasi. Untuk menghapus relasi "FRIENDS_WITH" antara dua orang, gunakan query berikut:

```
MATCH (person1:Person)-[rel:FRIENDS_WITH]-(person2:Person)
WHERE person1.name = 'Alice' AND person2.name = 'Bob'
DELETE rel
```



8. Menghapus Data. Gunakan perintah **DELETE** untuk menghapus simpul atau relasi dari graph. Misalnya, untuk menghapus simpul dengan nama "Alice":

```
MATCH (p:Person {name:'Alice'})
DELETE p
```



9. Melakukan Pembaruan Data. Gunakan perintah **SET** untuk memperbarui properti pada simpul atau relasi. Misalnya, untuk mengubah umur seseorang menjadi 40:

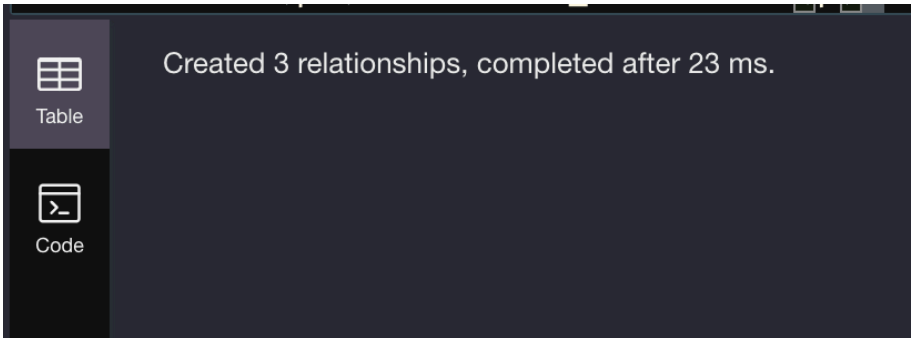
```
MATCH (p:Person {name:'Alice'})
SET p.age = 40
RETURN p
```

10. Query kombinasi CREATE dan SET. Membuat simpul baru dengan label "Person" dan mengatur beberapa properti:

```
CREATE (p:Person)
SET p.name = 'John', p.age = 25, p.city = 'Jakarta'
RETURN p
```

11. Membuat hubungan MUTUAL_FRIEND untuk Person John

```
MATCH (p:Person {name:'John'})
MATCH (p2:Person {name: 'Bob'})
MATCH (p3:Person {name:'Alice'})
CREATE (p)-[:FRIENDS_WITDH]->(p2)
CREATE (p)-[:MUTUAL_FRIEND]->(p3)
CREATE (p3)-[:MUTUAL_FRIEND]->(p)
```



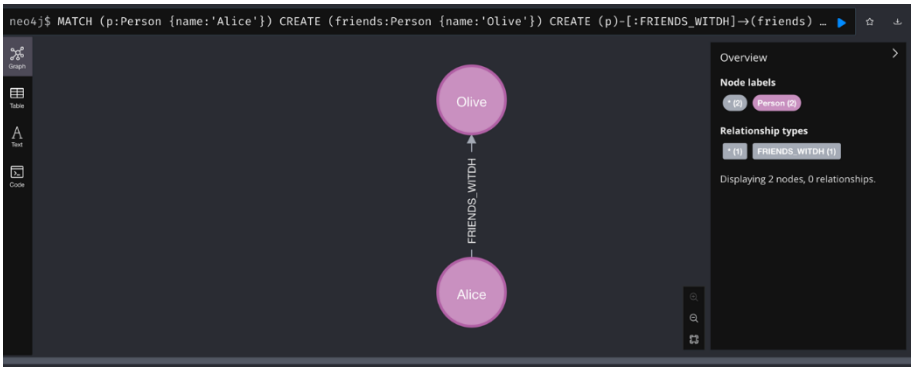
12. Query kombinasi MATCH, SET, dan DELETE. Mencari simpul dengan label "Person" yang memiliki nama "John", mengubah umurnya menjadi 30, dan menghapus relasi "FRIENDS_WITH" yang terhubung dengan simpul tersebut:

```
MATCH (p:Person {name: 'John'})
OPTIONAL MATCH (p)-[r:MUTUAL_FRIEND]-()
SET p.age=13
DELETE r
RETURN p
```

13. Query kombinasi MATCH, CREATE, dan SET. Mencari simpul dengan label "Person" yang memiliki nama "Alice", membuat simpul teman baru, dan mengatur relasi

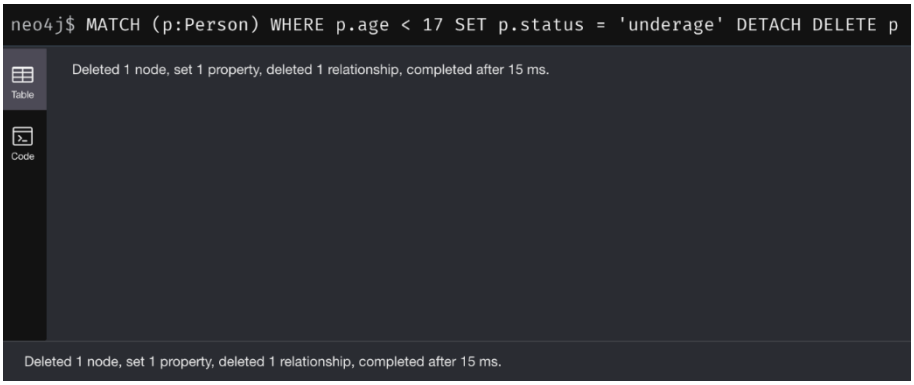
"FRIENDS_WITH" antara mereka:

```
MATCH (p:Person {name: 'Alice'})
CREATE(friends:Person {name: 'Olive'})
CREATE (p)-[:FRIENDS_WITH]->(friends)
RETURN p, friends
```



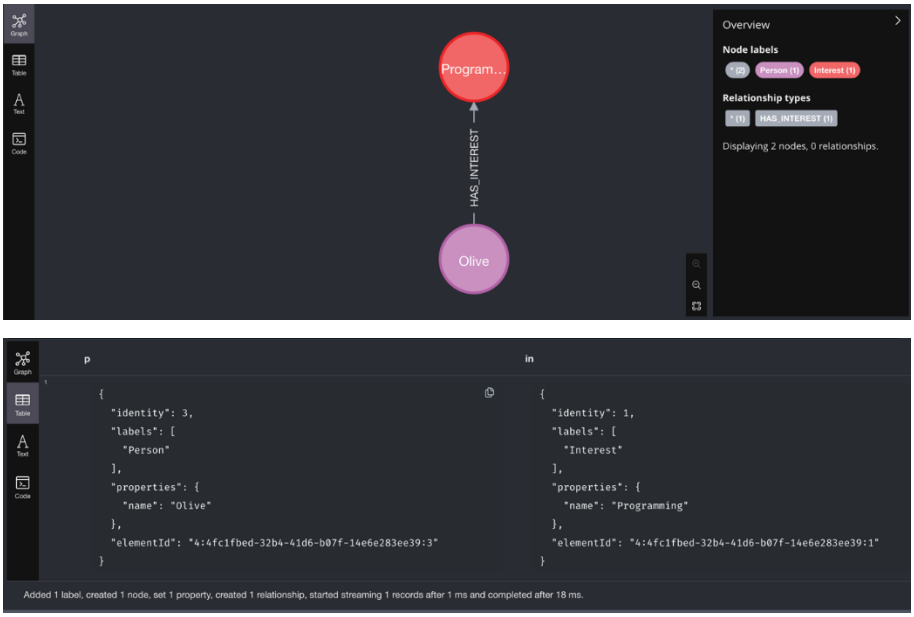
14. Query kombinasi MATCH, SET, dan DELETE. Mencari simpul dengan label "Person" yang memiliki umur di bawah 20 tahun, mengubah properti "status" mereka menjadi "Underage", dan menghapus simpul tersebut beserta relasinya:

```
MATCH (p:Person)
WHERE p.age < 20
SET p.status = 'Underage'
DETACH DELETE p
```



15. Query kombinasi MATCH, CREATE, dan SET. Mencari simpul dengan label "Person" yang memiliki nama "Olive", membuat simpul baru dengan label "Interest", dan mengatur relasi "HAS_INTEREST" antara simpul tersebut:

```
MATCH (p:Person {name: 'Jane'})
CREATE(in:Interest {name: 'Programming'})
CREATE (p)-[:HAS_INTEREST]->(in)
RETURN p, in
```



9.5. Tugas dan Latihan

Buatlah Query berikut:

1. Temukan semua orang yang tinggal di kota 'Jakarta'.
2. Berapa total jumlah teman dari setiap individu?
3. Update umur semua orang menjadi 40 tahun.
4. Hapus semua relasi 'FRIENDS_WITH' yang terhubung dengan individu yang berumur di atas 50 tahun.
5. Buatlah simpul baru dengan label "Hobby" dan properti "name" adalah 'Reading', kemudian hubungkan dengan individu 'Alice' dengan relasi 'HAS_HOBBY'.
6. Hapus semua simpul dengan label "Hobby" yang tidak memiliki relasi 'HAS_HOBBY'.
7. Tambahkan properti "gender" dengan nilai 'Male' pada semua orang yang berumur di atas 30 tahun.
8. Temukan teman-teman dari teman individu 'Alice' yang tinggal di kota yang sama.
9. Berapa jumlah individu yang memiliki properti "gender"?
10. Hapus semua simpul dan relasi dalam graph.