What makes the graph slow when getting data?

Contoh query lambat:

```
MATCH (t:Track)-[*1..4]->(n)
RETURN n
```

Query ini menelusuri terlalu dalam tanpa batas yang jelas (berbahaya apalagi jika data besar). Jika punya ribuan track dan relasi invoice, invoiceLine, dll, traversal ini bisa memakan waktu lama.

Solusi:

1. Gunakan label dan relasi eksplisit:

   ```
   MATCH (t:Track)-[:BELONGS_TO]->(a:Album)
   RETURN t.name, a.title
   ```

2. Gunakan LIMIT, dan filter properti spesifik:

   ```
   MATCH (c:Customer)-[:MADE]->(i:Invoice)
   WHERE c.Country = "USA"
   RETURN c.FirstName, i.Total
   LIMIT 50
   ```

3. Gunakan PROFILE untuk cek performa:

   ```
   PROFILE MATCH (c:Customer)-[:MADE]->(i:Invoice) RETURN i
   ```

How to handle data changes in the middle of a node relationship?

Contoh: Ganti Album lama dengan yang baru

1. Temukan Album lama yang ingin diganti

   ```
   MATCH (ar:Artist {Name: "AC/DC"})-[:PRODUCED]->(a:Album {Title: "For Those
   About To Rock We Salute You"})
   ```

2. Buat Album baru

   ```
   CREATE (a2:Album {Title: "New Album Title", AlbumId: 999})
   ```

3. Hubungkan ulang relasi

   ```
   CREATE (ar)-[:PRODUCED]->(a2)
   ```

4. Copy semua track dari album lama

```
MATCH (a)-[:CONTAINS]->(t:Track)
CREATE (a2)-[:CONTAINS]->(t)
```

5. Hapus album lama jika perlu

```
DETACH DELETE a
```

How do you detect dangling nodes in Chinook?

Deteksi node yang tidak punya relasi apapun:

```
MATCH (n)
WHERE NOT (n)--()
RETURN n
```

Contoh :

1. Temukan Track tanpa album

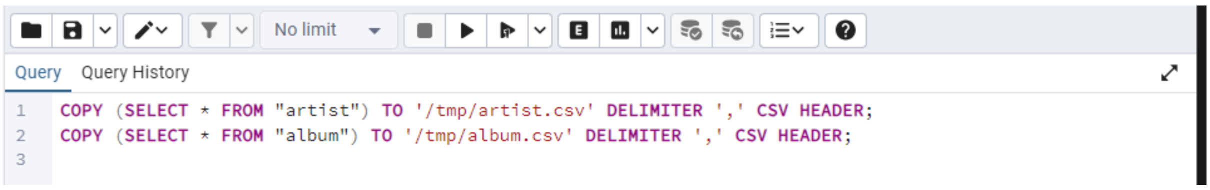
```
MATCH (t:Track)
WHERE NOT (t)-[:BELONGS_TO]->(:Album)
RETURN t
```

2. Temukan Customer yang tidak pernah membeli apapun

```
MATCH (c:Customer)
WHERE NOT (c)-[:MADE]->(:Invoice)
RETURN c
```

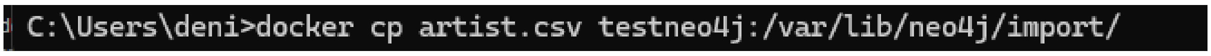
How to load data from RDBMS (PostgreSQL) to Neo4j using Chinook?

Export dari PostgreSQL:

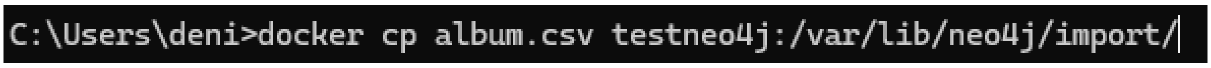


```
COPY (SELECT * FROM "artist") TO '/tmp/artist.csv' DELIMITER ',' CSV HEADER;
COPY (SELECT * FROM "album") TO '/tmp/album.csv' DELIMITER ',' CSV HEADER;
```

Copy file ke container:

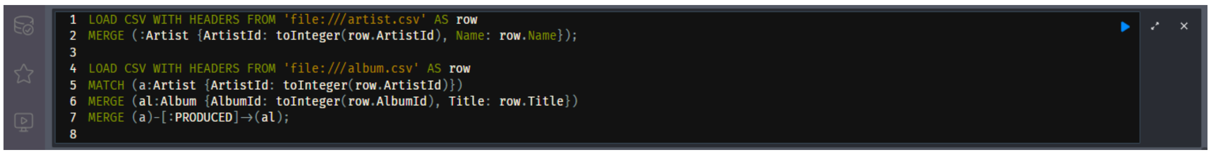


```
C:\Users\deni>docker cp artist.csv testneo4j:/var/lib/neo4j/import/
```



```
C:\Users\deni>docker cp album.csv testneo4j:/var/lib/neo4j/import/
```

Load dari Neo4j:



```
LOAD CSV WITH HEADERS FROM 'file:///artist.csv' AS row
MERGE (:Artist {ArtistId: toInteger(row.ArtistId), Name: row.Name});

LOAD CSV WITH HEADERS FROM 'file:///album.csv' AS row
MATCH (a:Artist {ArtistId: toInteger(row.ArtistId)})
MERGE (al:Album {AlbumId: toInteger(row.AlbumId), Title: row.Title})
MERGE (a)-[:PRODUCED]->(al);
```