REFERENSI CEPAT NEO4J / CYPHER

Query graph database, node, relasi, dan pola

Dasar Cypher

Struktur Query

`MATCH`	Temukan pola dalam graph
`WHERE`	Filter hasil
`RETURN`	Tentukan kolom output
`CREATE`	Buat node dan relasi
`SET / REMOVE`	Perbarui properti dan label
`DELETE / DETACH DELETE`	Hapus node dan relasi

Menjalankan Query

```
// Neo4j Browser: tempel dan jalankan dengan Ctrl+Enter
// cypher-shell:
cypher-shell -u neo4j -p secret "MATCH (n) RETURN n LIMIT 5"
```

Node & Label

Sintaks Node

```
(n)                    // node anonim
(p:Person)             // node dengan label
(p:Person:Employee)    // beberapa label
(p:Person {name: "Alice", age: 30})
```

Operasi Label

```
SET n:Active           // tambah label
REMOVE n:Active        // hapus label
MATCH (n) RETURN labels(n)  // daftar label
```

Constraint & Index

```
CREATE CONSTRAINT FOR (p:Person)
  REQUIRE p.email IS UNIQUE
CREATE INDEX FOR (p:Person) ON (p.name)
SHOW INDEXES
```

Relasi

Sintaks Relasi

```
-[r]->                 // berarah (keluar)
<-[r]-                 // berarah (masuk)
-[r]-                  // tidak berarah
-[:KNOWS]->            // relasi bertipe
-[r:KNOWS {since: 2020}]->  // dengan properti
```

Path dengan Panjang Variabel

```
-[:KNOWS*2]->          // tepat 2 hop
-[:KNOWS*1..3]->       // 1 sampai 3 hop
-[:KNOWS*]->           // jumlah hop bebas
shortestPath((a)-[*]-(b))  // path terpendek
```

CREATE

Membuat Node

```
CREATE (p:Person {name: "Alice", age: 30})
CREATE (p:Person {name: "Bob"})
RETURN p
```

Membuat Relasi

```
MATCH (a:Person {name: "Alice"})
MATCH (b:Person {name: "Bob"})
CREATE (a)-[:KNOWS {since: 2020}]->(b)
```

MERGE (Upsert)

```
MERGE (p:Person {email: "alice@example.com"})
ON CREATE SET p.name = "Alice", p.created = date()
ON MATCH SET p.lastSeen = date()
```

MATCH

Pola Dasar

```
MATCH (p:Person) RETURN p
MATCH (p:Person)-[:KNOWS]->(f) RETURN p, f
MATCH (a)-[r]->(b) RETURN type(r), a, b
```

OPTIONAL MATCH

```
// Mengembalikan null untuk kecocokan yang tidak ada (seperti LEFT
JOIN)
MATCH (p:Person)
OPTIONAL MATCH (p)-[:OWNS]->(c:Car)
RETURN p.name, c.model
```

Pattern Comprehension

```
MATCH (p:Person)
RETURN p.name,
  [(p)-[:KNOWS]->(f) | f.name] AS friends
```

WHERE

Perbandingan & Logika

```
WHERE p.age > 25
WHERE p.age >= 18 AND p.active = true
WHERE p.name <> "Bob" OR p.role = "admin"
WHERE NOT (p)-[:BLOCKED]->()
```

Predikat String & List

```
WHERE p.name STARTS WITH "Al"
WHERE p.name CONTAINS "ice"
WHERE p.name =~ "(?i)alice.*"   // regex
WHERE p.age IN [25, 30, 35]
```

Pemeriksaan Null & Eksistensi

```
WHERE p.email IS NOT NULL
WHERE p.phone IS NULL
WHERE EXISTS { (p)-[:KNOWS]->(:Person) }
```

RETURN

Opsi Output

```
RETURN p.name AS name, p.age AS age
RETURN DISTINCT p.city
RETURN p, collect(f) AS friends
RETURN count(*) AS total
```

Pengurutan & Paginasi

```
RETURN p.name ORDER BY p.age DESC
RETURN p SKIP 10 LIMIT 5
```

UNWIND

```
// Perluas list menjadi baris
UNWIND [1, 2, 3] AS x RETURN x
UNWIND $names AS name
MERGE (p:Person {name: name})
```

UPDATE & DELETE

SET Properti

```
MATCH (p:Person {name: "Alice"})
SET p.age = 31, p.updated = date()
SET p += {city: "NYC", active: true}
```

REMOVE

```
MATCH (p:Person {name: "Alice"})
REMOVE p.temp_field     // hapus properti
REMOVE p:Inactive       // hapus label
```

DELETE

```
MATCH (p:Person {name: "Bob"})
DETACH DELETE p          // hapus node + semua relasi
// DELETE p              // gagal jika node punya relasi
MATCH ()-[r:OLD_REL]->() DELETE r  // hapus relasi
```

Agregasi

Fungsi Agregat

`count(x)`	Jumlah nilai non-null
`sum(x)`	Jumlah total nilai numerik
`avg(x)`	Rata-rata nilai numerik
`min(x) / max(x)`	Nilai minimum / maksimum
`collect(x)`	Agregasi nilai ke dalam list
`percentileCont(x, 0.5)`	Persentil kontinu

GROUP BY (Implisit)

```
// Kolom non-agregat menjadi kunci pengelompokan
MATCH (p:Person)-[:LIVES_IN]->(c:City)
RETURN c.name, count(p) AS population
ORDER BY population DESC
```

WITH (Agregasi Berantai)

```
MATCH (p:Person)-[:KNOWS]->(f)
WITH p, count(f) AS friendCount
WHERE friendCount > 5
RETURN p.name, friendCount
```

Pola Umum

Temukan Teman Bersama

```
MATCH (a:Person {name:"Alice"})-[:KNOWS]->(m)<-[:KNOWS]-(b:Person
{name:"Bob"})
RETURN m.name AS mutualFriend
```

Rekomendasi (Teman dari Teman)

```
MATCH (p:Person {name:"Alice"})-[:KNOWS*2]-(fof)
WHERE NOT (p)-[:KNOWS]-(fof) AND p <> fof
RETURN DISTINCT fof.name
```

Import Data CSV

```
LOAD CSV WITH HEADERS FROM 'file:///people.csv' AS row
MERGE (p:Person {id: row.id})
SET p.name = row.name, p.age = toInteger(row.age)
```

Info Database

```
CALL db.labels()        // daftar semua label
CALL db.relationshipTypes()  // daftar tipe relasi
CALL db.schema.visualization()
```