Query Combinators

Unleash the power of query combination

Stefan Plantikow
stefan.plantikow@neotechnology.com

Lead openCypher Language Group (CLG)
Neo Technology

opencypher.org | opencypher@googlegroups.com
Query Combinators

Nested subqueries allow us to group queries together enabling:

- Union post-processing
- Grouping updates with DO
- Generic optional queries
- Generic mandatory queries

Are there other ways to compose compound queries?
Set Operations

- UNION
- INTERSECT
Set Operations

- EXCLUSIVE UNION

- EXCLUDE
Simple example

MATCH (a:Person)-[:LIVES_IN]->(:City {name: "Berlin"})
RETURN *
INTERSECT
MATCH (a:Person)-[:BORN_IN]->(:City {name: "London"})
RETURN *
Example using nested subqueries

MATCH (a:Person)-[:LIVES_IN]->(:City {name: "Berlin"})
RETURN *
INTERSECT {
  MATCH (a:Person)-[:BORN_IN]->(:City {name: "London"})
  RETURN *
  UNION
  MATCH (a:Person)-[:BORN_IN]->(:City {name: "New York City"})
  RETURN *
}

Query Combinators are left associative!

```
MATCH (a:Person)-[:BORN_IN]->(:City {name: "London"})
RETURN *

UNION

MATCH (a:Person)-[:BORN_IN]->(:City {name: "New York City"})
RETURN *

INTERSECT

MATCH (a:Person)-[:LIVES_IN]->(:City {name: "Berlin"})
RETURN *
```
Multiset Operators

- **UNION ALL**  \( n + k \)
- **INTERSECT ALL**  \( \min(n, k) \)
- **EXCLUDE ALL**  \( \max(0, n-k) \)
- **UNION MAX**  \( \max(n, k) \)
- **EXCLUSIVE UNION MAX**  \( \max(n, k) - \min(n,k) \)
Ground Rules (I)

- Query combinators are left Associative

  \[
  \langle q_1 \rangle \ \text{EXCEPT} \ \langle q_2 \rangle \ \text{EXCEPT} \ \langle q_3 \rangle \ \ldots \\
  \Rightarrow \ \langle q_1 \rangle \ \text{EXCEPT} \ (\langle q_2 \rangle \ \text{EXCEPT} \ (\langle q_3 \rangle \ \text{EXCEPT} \ (\ldots \ )))
  \]

- Records compared by equivalence (i.e. NULLs are treated as identical)

  \[
  \text{RETURN NULL EXCEPT RETURN NULL} \\
  \Rightarrow \ \text{Empty Result}
  \]
Ground Rules (II)

- All query combinators can be used with nested subqueries

\[
\{ \ldots \} \text{ UNION } \{ \ldots \}
\]

At arbitrary (finite) depth

\[
\{ \{ \ldots \} \text{ UNION } \{ \ldots \} \text{ EXCEPT } \ldots \}
\]
Can we do more?

This adds basic set and multiset operations to Cypher, what else could we provide?

- THEN
- CROSS
- OTHERWISE
And THEN

- Cypher already supports pipelining of query parts using **WITH**
- **THEN** is for pipelining at the query level
- Why?
  - Avoids parsing ambiguity
  - Allows pipelining into nested subqueries

```
MATCH ... RETURN *
THEN
MATCH ... RETURN *
```
MATCH (city:City)
RETURN city
CROSS
MATCH (continent:Continent)
RETURN continent
THEN
RETURN city.name, continent.name,
EXISTS { (city)-[:PARTNER]-(other:City)-[:IN]->(continent) }
Or OTHERWISE..

Sometimes it is useful to try different matches in some order

MATCH (p:Person)-[:LIVES_IN]->(c:City)
RETURN * LIMIT 1
OTHERWISE
MATCH (a:Person)-[:WORKS_IN]->(c:City)
RETURN * LIMIT 1
THEN
MATCH (c)-[:RECOMMEND]->(attraction:Attraction)
RETURN p.name, attraction.name, attraction.price
Summary

- Query Combinators provide new ways to compose Cypher queries
- They compose nicely with nested subqueries
- We plan to re-use this language infrastructure in the context of Cypher for multiple graphs