Nested subqueries

CIP

Petra Selmer
petra.selmer@neotechnology.com

Nested subqueries...

...are self-contained queries running within the scope of an outer query

https://github.com/opencypher/openCypher/pull/100

- Two forms
  - Read-only
  - Read/Write

Some preliminaries regarding nested subqueries

...RETURN...

- Referred to as “inner-query” from now on
- Can be any complete read-only query
- Found within **{}**

- May be correlated (*inner-query* may use vars from the outer query) or uncorrelated
- Read-only subqueries may be nested at an arbitrary depth

1. Read-only nested simple subqueries

```
{ <inner-query> }
```

- This is the simplest form
- No introductory keyword
- Only to be used as a *primary* clause, which is one of the following:
  - a top-level query
  - an inner query of another nested subquery
  - an inner query of another expression-level subquery (e.g. **EXISTS** subquery)
  - as an argument query to **UNION**
- Cannot be used as a so-called *secondary* clause succeeding a primary clause

2. Read-only nested chained subqueries

`THEN { <inner-query> }`

- Unlike the simple form, these can succeed primary clauses; i.e. may be used as a secondary clause
- Introduction of a query combinator: **THEN** (more later)

3. Optional and mandatory subqueries

`OPTIONAL { <inner-query> }`

- Read-only nested optional subqueries

`MANDATORY { inner-query }`

- Read-only nested mandatory subqueries
- More on **MANDATORY** later

Read-only nested subquery semantics

- *inner-query* is evaluated for each incoming record (from the outer query)
- 0 - n result records are produced
- All incoming vars remain in scope in *inner-query*
  - There will be no effect if any of these vars are projected by *inner-query*
  - *inner-query* cannot shadow incoming vars
- Any new var bindings introduced by the final **RETURN** will augment the var bindings of the initial record
  - If such bindings are not explicitly returned, they will be “lost” after *inner-query* completes evaluation (i.e. they will be temporary)

Read-only subquery examples

```
{
  MATCH (me:User {name: 'Alice'})-[:FOLLOWS]->(user:User),
        (user)<-[:AUTHORED]-(tweet:Tweet)
  RETURN tweet, tweet.time AS time, user.country AS country
  UNION
  MATCH (me:User {name: 'Alice'})-[:FOLLOWS]->(user:User),
     (user)<-[:HAS_FAVOURITE]-(favorite:Favorite)-[:TARGETS]->(tweet:Tw
     eet)
  RETURN tweet, favourite.time AS time, user.country AS country
}
WHERE country = 'se'
RETURN DISTINCT tweet
ORDER BY time DESC
LIMIT 10
```

Read-only subquery examples

```
MATCH (f:Farm {id: $farmId})-[:IS_IN]->(country:Country)
THEN {
  MATCH (u:User {id: $userId})-[:LIKES]->(b:Brand),
        (b)-[:PRODUCES]->(p:Lawnmower)
  RETURN b.name AS name, p.code AS code
  UNION
  MATCH (u:User {id: $userId})-[:LIKES]->(b:Brand),
        (b)-[:PRODUCES]->(v:Vehicle),
        (v)<-[:IS_A]-(:Category {name: 'Tractor'})
  WHERE v.leftHandDrive = country.leftHandDrive
  RETURN b.name AS name, p.code AS code
}
RETURN f, name, code
```

4. Read/Write nested simple updating subqueries

```
DO { <inner-update-query> }
```

- *inner-update-query*:
  - may be *any* updating query
  - does not end with **RETURN -** no data is returned
- A logical consequence is the removal of **FOREACH** from Cypher

5. Read/Write nested conditionally-updating subqueries

```
DO
    [WHEN <cond> THEN <inner-update-query>]+
    [ELSE <inner-update-query>]
END
```

- Conditions in **WHEN** evaluated in order
- *inner-update query* evaluated for the first true condition, falling through to **ELSE** if no true conditions found. Otherwise, no updates will occur.

Read/Write nested subquery semantics

- *inner-update-query* is run for each incoming record
  - Executing **DO** does not affect the cardinality
- Input records are passed on to any clauses following the subquery
  - This happens regardless if whether the record was eligible for processing by *inner-update-query*
- A query can end with a **DO** subquery in the same way as a query currently can end with any update clause
- These types of subqueries may not be contained within read-only nested subqueries

Read/Write subquery examples

```
MATCH (r:Root)
UNWIND range(1, 10) AS x
DO {
  MERGE (c:Child {id: x})
  MERGE (r)-[:PARENT]->(c)
}
```

```
MATCH (r:Root)
UNWIND range(1, 10) AS x
DO WHEN x % 2 = 1 THEN {
      MERGE (c:Odd:Child {id: x})
      MERGE (r)-[:PARENT]->(c)
  }
  ELSE {
      MERGE (c:Even:Child {id: x})
      MERGE (r)-[:PARENT]->(c)
  }
END
```

And finally...some shorthand syntax proposals!

- `WHERE <cond> <subclause>` ⇨
  `WITH * WHERE <cond> THEN { <subclause> }`
- `WITH -, RETURN -`
  - Retains input cardinality
  - Does not project any result fields
  - Allows for checking the cardinality of a **read-only nested mandatory subquery**
- `YIELD -`
  - Retains output cardinality of **CALL**
  - Does not project any result fields
  - Allows for checking the cardinality in an **EXISTS** subquery