

openCypher developments: 2017

Petra Selmer, Stefan Plantikow
Neo4j

10th LDBC TUC, Munich, 1 September 2017

Outline

Overview of openCypher

Path pattern queries

Subqueries

Support for multiple graphs, allowing for query composition

Configurable *morphism

Goals of openCypher (oC)

Evolve Cypher through an open process

Consensus-based agreement of new features

openCypher Implementers Group ([oCIG](#))

Includes vendors, researchers and other interested parties

Open to all

Cypher Improvement Requests and Proposals ([CIRs](#) and [CIPs](#))

oC Implementers Meetings (oCIM)

February 2017: Walldorf, Germany

May 2017: London, UK

- oC artifacts (TCK, grammar)
- SAP HANA Graph, Redis Graph, Bitnine
- Formal data model and core semantics for Cypher: University of Edinburgh
- Incremental graph execution (ingraph), Gradoop
- Cypher implementation in Prolog; Cypher for Apache Spark
- Path patterns (Generalized RPQs)
- Multiple graphs; Compositional language
- Many more: **MANDATORY MATCH**, subqueries, grouping semantics

oC Implementers Group (oCIG): virtual meetings every 3 weeks



Features actively being designed

1) Path Pattern Queries

Path pattern queries: complex patterns (RPQs)

- Thoughts about RPQs in Cypher in 2014
- The Cypher Language Group considered these at Canterbury, UK, in 2015
- Latterly influenced/correlated by GXPPath and work by Libkin et al

$$((\text{M} \rightarrow [:\text{x}] \rightarrow \text{a}))^* ((\text{R} \rightarrow [:\text{r}] \rightarrow \text{O}))^*$$

M fav : Person

$\text{P} = (\text{M} \rightarrow [:\text{x}] \rightarrow \text{a}) \text{ WHERE } \text{a} \dots \text{b}$
 $\quad \quad \quad [\text{R} \rightarrow [:\text{y}] \rightarrow \text{O}] \text{ WHERE } \text{b} \dots \text{c}$

R bar : code

$\text{M} (\text{R} \rightarrow [:\text{x}] \rightarrow \text{O} \rightarrow [:\text{y}] \rightarrow \text{O})^*$
 $\quad \quad \quad [[(\text{M} \rightarrow [:\text{A}] \rightarrow \text{O})]^* \text{ OR } [\text{R} \rightarrow [:\text{B}] \rightarrow \text{O}]]^*$

$\text{P} = \langle \text{rat} \rangle^{1..5} \text{ WHERE T}^*$

$(\text{a}^* \text{b}^*)^*$

$\rho = (\text{M} \rightarrow [:\text{x}] \rightarrow \text{C}) \text{ - } [\text{R} \rightarrow [:\text{y}] \rightarrow \text{O}]$
 $\quad \quad \quad (\text{R} \rightarrow [:\text{r}] \rightarrow \text{O})$

$\text{rels}(\rho)$

Y:TYPE^*

$(\text{O} \rightarrow [:\text{ALL} \rightarrow \text{O}] \rightarrow \text{O})^*$

~~WHERE~~ $\text{r.age} < 22$

$\forall \text{r}: \text{rels}(\rho), \text{r.age} < 22$

~~WHERE~~ ALL r:

$(\text{O} \rightarrow [:\text{r}] \rightarrow \text{O})^*$
 $(\text{O} \rightarrow [:\text{A}] \rightarrow \text{O} \rightarrow [:\text{B}] \rightarrow \text{O})^*$

Path pattern queries: constructs

- Predicates on relationship type: `()-/:FOO/-()`
- Predicates on nodes: `()-/(:Alpha {beta:'gamma'})/-()`
- Alternation: `()-/:FOO | :BAR | :BAZ/-()`
- Sequence: `()-/:FOO :BAR :BAZ/-()`
- Grouping: `()-/:FOO | [:BAR :BAZ]/-()`
- Direction: `()-/<:FOO :BAR <:BAZ>/->()`
- Any relationship: `()-/-/-()`
- Repetition: `()-/:FOO? :BAR+ :BAZ* :FOO*3.. :BAR*1..5/-()`
- Predicates on relationship properties: `()-/ [- {some:'value'}] /-()`
- And more complex variants....

2) Subqueries

Subqueries

Cypher feature request: post-UNION processing #2725

! Open

aseemk opened this issue on 22 Jul 2014 · 82 comments



aseemk commented on 22 Jul 2014



Related to issue [#1879](#), but I'd like to ask for more holistic and general support for post-UNION processing, not just limiting/skipping/ordering.

One way to look at it is very much like a `WITH` clause. Perhaps `UNION WITH` would thus be a good name for it, or maybe `UNION RESULT` to convey that you're now acting on the entire union'ed result.

Our major use case is aggregating a stream of content, where that content is queried in different

Subqueries

Added the nested subqueries CIP #100

 Open petraselmer wants to merge 15 commits into `opencypher:master` from `petraselmer:CIP-nested-subqueries`

Conversation 22

Commits 15

Files changed 1



petraselmer commented on 22 Jun 2016

Owner



No description provided.



6



2

 Added the nested subqueries CIP

✓ 10c87a0



 petraselmer added `language feature` `CIP` labels on 22 Jun 2016



 petraselmer referenced this pull request in `neo4j/neo4j` on 23 Jun 2016

Cypher feature request: post-UNION processing #2725

 Open

Subqueries

Nested:

- Run any complete read-only Cypher query
- Incoming variables remain in scope: correlated subquery
- Arbitrary depth

Will be discussed at the oCIG meeting: **28 Sep 2017**

Existential: returns true if at least one match found; false otherwise

Scalar: result is a single value in a single row

List: result is the list formed by collecting all the values of all rows (single value per row)

Updating: simple and conditional updates, executed once per incoming row

Examples of subqueries

```

MATCH (f:Farm)-[:IS_IN]->(country:Country)
WHERE country.name IN $countryNames
THEN {
    MATCH (u:User {id: $userId})-[:LIKES]->(b:Brand),
          (b)-[:PRODUCES]->(p:Lawnmower)
    RETURN b AS brand, b.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 AS brand, b.code AS code
}
WHERE f.type = 'organic'
    AND b.certified
RETURN f, brand.name AS name, code

```

```

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

```

3) Multiple graphs

Multiple graphs: History

We (in Neo) have been actively working on graph query composition and support for multiple graphs since 2016; in openCypher this was discussed extensively since Feb 2017

- oCIM 1 + LDBC TUC Walldorf (February 2017)
- oCIM 2 (CIR May 2017)
- openCypher proposal (CIP) (June 2017)
- oCIG 4 (August 2017)
- Created new task force for working on details
- Implementing in Cypher for Apache Spark (CAPS)
- Upcoming oCIM 3 in New York, 23. October 2017 (Day before GraphConnect)

Please see openCypher.org for slides + github.com/openCypher/openCypher for CIPs

Multiple graphs

Accepting multiple graphs as input

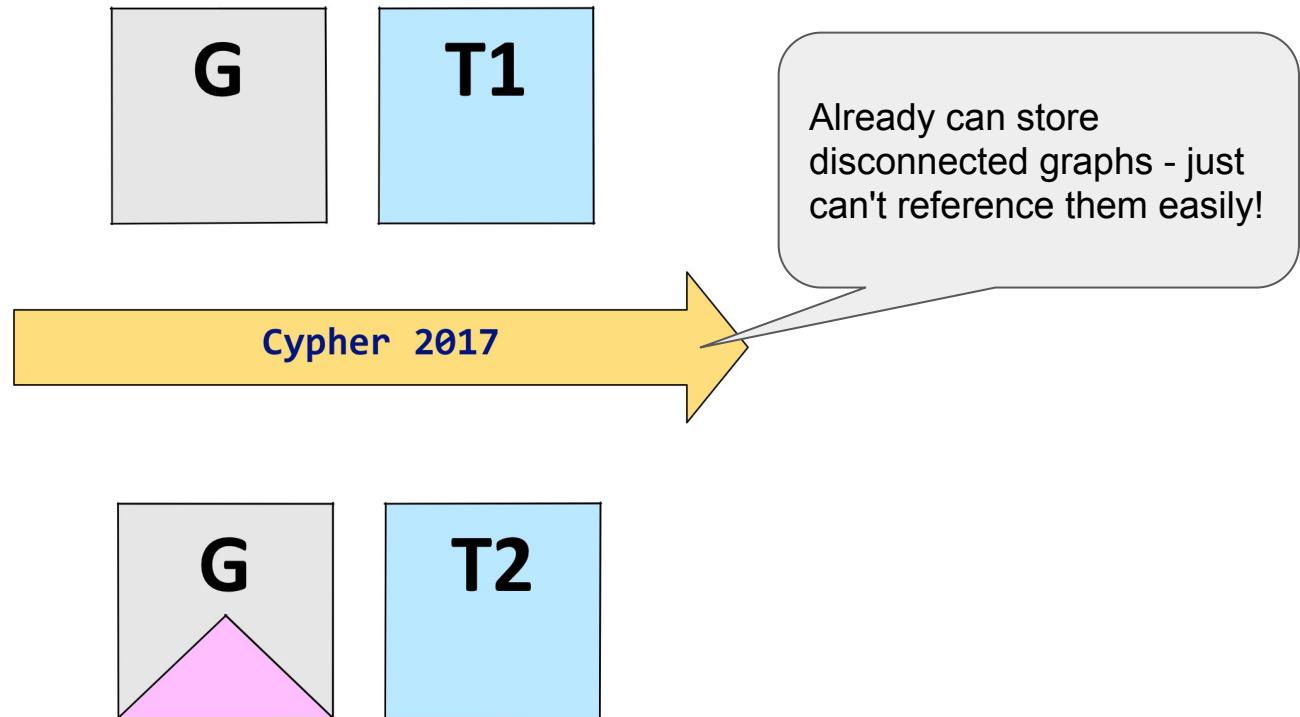
Graph addressing

Returning graphs and a table as output

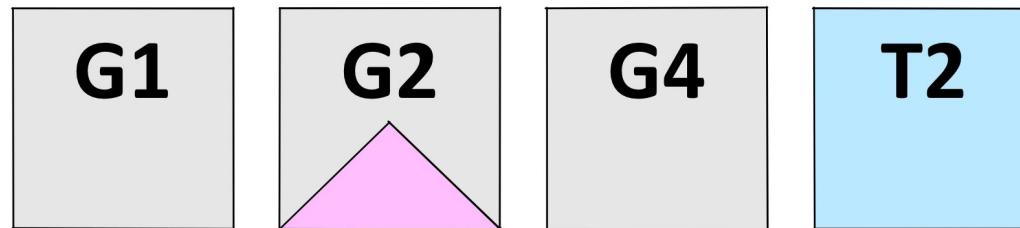
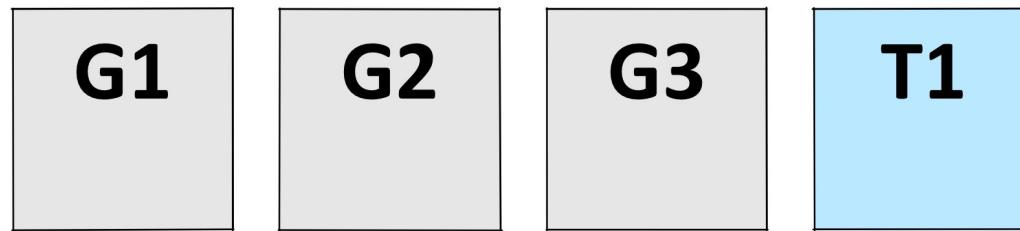
Graph query composition (and by extension, graph transformation and views)

Updating and materialising graphs

Language Model: CYPHER 2017



Language Model: CYpher 2018



Language changes

- DQL: Referencing graphs and selecting which graph to match from
- DQL: Returning graphs and graph transformation
- DQL: Graph set operations (union etc.)
- DML: Selecting which graph to write to
- DDL: Creating and handling persisting graphs , creating constraints etc.
- DCL: To be done

Selecting graph to query from

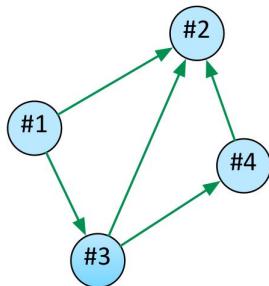
Which graph is queried by MATCH?

```
FROM GRAPH cities AT "hdfs://.../cities"
MATCH (city:City)-[:IN]->(:Country {name: "Germany"})
FROM GRAPH people AT "hdfs://.../germany/people"
MATCH (person)-[:LIVES_IN]->(city)
RETURN person ORDER BY person.age LIMIT 1
```

Tables from graphs

It's easy to construct tables from a graph... but what's the inverse?

MATCH (a)-->(b) WITH a, b ...



GRAPH

(#1)->(#2)
(#1)->(#3)
(#3)->(#2)
(#3)->(#4)
(#4)->(#2)

MATCHES

a: #1, b: #2
a: #1, b: #3
a: #3, b: #2
a: #3, b: #4
a: #4, b: #2

RECORDS

a: #1, b: #2
a: #1, b: #3
a: #3, b: #2
a: #3, b: #4
a: #4, b: #2

TABLE

Graphs from tables

...a graph is a set of pattern matches!

WITH a, r, b RETURN GRAPH OF (a)-[r]->(b) AS foo

a:#1, r:#5, b:#2
a:#1, r:#6, b:#3
a:#3, r:#7, b:#2
a:#3, r:#8, b:#4
a:#4, r:#9, b:#2

a:#1, r:#5, b:#2
a:#1, r:#6, b:#3
a:#3, r:#7, b:#2
a:#3, r:#8, b:#4
a:#4, r:#9, b:#2

(#1)-[#5]->(#2)
(#1)-[#6]->(#3)
(#3)-[#7]->(#2)
(#3)-[#8]->(#4)
(#4)-[#9]->(#2)

TABLE

RECORDS

MATCHES

GRAPH

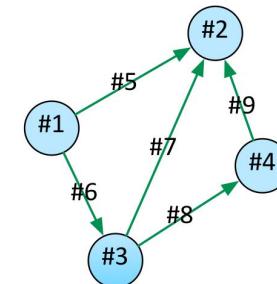


Table => Graph => Table => Graph => ...

Tables from Graphs + Graphs from tables => Query composition

```
WITH GRAPHS people, teams
FROM GRAPH people
MATCH (a:Person)-[:WORKS_AT]->(:Office)<-[:WORKS-AT]-(b:Person)
FROM GRAPH teams
MATCH (t:Team) WHERE EXISTS (a)-[:MEMBER_OF]->(t)<-[:MEMBER_OF]-(b)
// 
// More relational and graph processing ...
// (no need to hide the "invisible binding table" ;)
//
RETURN GRAPH OF (a)-[:COLOCATED_TEAM_MEMBER {name: t.name}]->(b)
```

Table => Graph => Table => Graph => ...

Tables from Graphs + Graphs from tables => Query composition

```
WITH GRAPH OF (a)-[:COLOCATED_TEAM_MEMBER {name: t.name}]- (b)
// 
// Keep querying: Co-located pairs that are in multiple locations
//
MATCH (a)-[r:COLOCATED_TEAM_MEMBER-> (b)
WITH a, b, count(r.name) AS count WHERE count >= 2
RETURN a, b
```

Creating & updating graphs

Graphs may be created, persisted, relocated, and deleted in the catalog

- **CREATE GRAPH graph AT "graph-uri" // e.g. catalog name**
- **SNAPSHOT GRAPH graph AT "graph-uri"**
- **DELETE GRAPH graph**

Updating graphs using Cypher's existing and proven DML

- **INTO GRAPH talks**

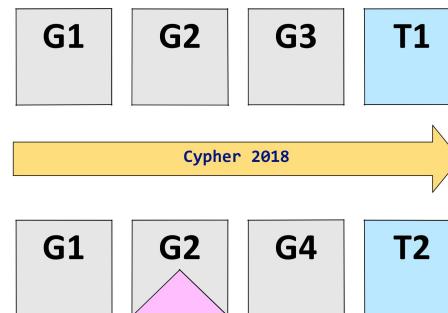
```
MERGE (a:Person {name: "Stefan"}), (b:Person {name: "Petra"})  
CREATE (a)-[:SPEAKER]->(:Talk {title: ...})<-[:SPEAKER]-(b)
```

...

Multiple Graphs Cypher Summary

- Cypher supports named graphs as input and output to a query
- Refer to graphs using Graph URIs
- Clauses work using specified source and target graphs
- Allow use of both DML and graph transformation for creating new graphs
- Cypher becomes graph compositional but supports tables in and out...

Naturally integrates with SQL Graph Querying Extensions



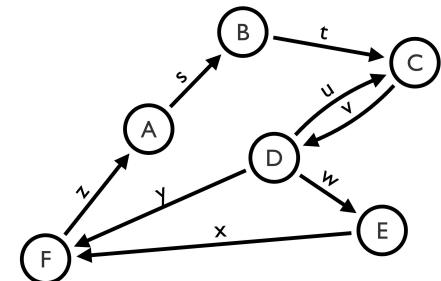
4) Configurable *morphism

Configurable *morphism and path matching semantics

- Configurable homomorphism, node/edge isomorphism
MATCH EVERY | UNIQUE NODES | UNIQUE RELS ...

- Configurable path kinds
MATCH WALKS | TRAILS | SIMPLE PATHS p=() -/.../-

- Configurable path matching semantics
MATCH ALL | ALL SHORTEST | SHORTEST | SHORTEST DISJUNCT p=
- Surprisingly, **MATCH ALL** turns out to be relevant in many real-world use-cases
(e.g. for exploring very sparse subgraphs)



Comparison (tbd.)

Feature	GCore	openCypher
...	???	???