Semantically identical cypher 2.2 queries produce different results

New issue

Closed **leocrawford** opened this issue Apr 25, 2015 · 5 comments

leocrawford commented Apr 25, 2015

The cypher query (from the console http://console.neo4j.org/r/ioasdv)

```
MATCH (ms)-[path:SATISFIES*0..]->(mp),(sc)<-[`_mandates`:MANDATED]-(`_mr`:ms)-[`_sa`:SATIS
WHERE NOT ((mp)-[:SATISFIES]->())
RETURN *
```

produces 33 entries under cypher 2.2, and 67 under cypher 2.1.

The semantically identical (I believe) query:

```
MATCH (ms)-[path:SATISFIES*0..]->(mp)
WHERE NOT ((mp)-[:SATISFIES]->())
MATCH (sc)<-[`_mandates`:MANDATED]-(`_mr`:ms)-[`_sa`:SATISFIES*0..]-(ms)
RETURN *
```

produces 67 under both cypher 2.1 and 2.2.

The test case of id(ms) = 30 can be used to show that the behaviour of 2.2 is wrong, viz:

```
MATCH (ms)-[path:SATISFIES*0..]->(mp)
WHERE NOT ((mp)-[:SATISFIES]->())
RETURN id(ms)
```

contains the entry 30, and so we write

```
MATCH (sc)<-[`_mandates`:MANDATED]-(`_mr`:ms)-[`_sa`:SATISFIES*0..]-(ms)
WHERE id(ms)=30
RETURN *
```

produces 5 results, and yet the original query doesn't return the entry 30, mostly clearly seen by using this query:

```
MATCH (ms)-[path:SATISFIES*0..]->(mp),(sc)<-[`_mandates`:MANDATED]-(`_mr`:ms)-[`_sa`:SATIS
WHERE NOT ((mp)-[:SATISFIES]->()) AND id(ms)=30
RETURN id(ms)
```

leocrawford referenced this issue Apr 25, 2015

Cypher where produces more results (not less) #4512 Closed

leocrawford commented Jul 16, 2015

Has anyone been able to replicate or investigate?

jexp commented Jul 18, 2015 Collaborator

Sorry, this fell through the cracks, I pointed the Cypher team to it.

Labels

None yet

Milestone

No milestone

Assignee

No one assigned

3 participants

leocrawford commented Jul 29, 2015

Thanks, any thoughts from the cypher team?

Mats-SX closed this Jul 30, 2015

Mats-SX reopened this Jul 30, 2015

Mats-SX commented Jul 30, 2015 Collaborator

Hello **@leocrawford**, and thank you for reporting this issue.

We have looked into it, and found that what you have seen is actually a bugfix for 2.2, which has not (and most likely will not) been back-ported to 2.1.

What has happened here is a bit subtle, but nevertheless important. There is a semantic difference between the two queries you have posted, and it comes from relationship traversal uniqueness. Each `MATCH` clause may only traverse a given relationship once.

If we rewrite your first query

```
MATCH (ms)-[path:SATISFIES*0..]->(mp),(sc)<-[`_mandates`:MANDATED]-(`_mr`:ms)-[`_sa`:SATIS
WHERE NOT ((mp)-[:SATISFIES]->())
RETURN *
```

into this (no comma)

```
MATCH (sc)<-[`_mandates`:MANDATED]-(`_mr`:ms)-[`_sa`:SATISFIES*0..]-(ms)-[path:SATISFIES*0
WHERE NOT ((mp)-[:SATISFIES]->())
RETURN *
```

we can see that what you're actually matching on is one full pattern. This is not the same as to match on one pattern and then match again on another pattern, as your second query implies (using two `MATCH` clauses).

So the overall conclusion is that you're experiencing a bug in Cypher 2.1. We were also a bit confused with your usage of the label `:ms` on the nodes matched into identifier `_mr`. Is that intended? It looks very much like a mistake, as it is easily confused with the identifier with the same name in the query. Also perhaps you did not mean to have the `(_mr:ms)-[_sa:SATISFIES*0..]-(ms)` pattern undirected?

leocrawford commented Jul 30, 2015

Thanks so much for taking a look at this, and your clear explanation. I hadn't appreciated that there were semantic differences between single and multiple match clauses - so my mistake. Sorry.

In terms of your feedback on the query - the query is as intended, but is hard to read because it is an extract of a machine generated query. I'd tried to tidy it up a bit to make it human readable, but hadn't gone far enough.

leocrawford closed this Jul 30, 2015

Sign up for free **to join this conversation on GitHub**. Already have an account? Sign in to comment