Cypher where produces more results (not less)

New issue

Closed **leocrawford** opened this issue Apr 23, 2015 · 4 comments

leocrawford commented Apr 23, 2015

In the following neo console the use of the where clause creates 22 results, removing it creates 20. I can't see any explanation for why adding a constraint would add more rows.

http://console.neo4j.org/r/nn8bnl

```
Query: MATCH (c)<-[cc*0..0]-(sc:sc { name:'enterprise' })<-[]-(`_mr`:ms)-[sa:SATISFIES*0..
count(*)
22
```

and..

```
Query took 27 ms and returned 1 rows. Query: MATCH (c)<-[cc*0..0]-(sc:sc { name:'enterpris
count(*)
20
```

These results are from v2.1, I get different answers again with later versions.

thobe commented Apr 23, 2015 Owner

Thank you for reporting this. I've been able to reproduce what you are experiencing. We will investigate further.

I can conclude (from profiling) that the two queries yield quite different plans.

The query:

```
MATCH (c)<-[cc*0..0]-(sc:sc { name:'enterprise' })<-[]-(`_mr`:ms)-[sa:SATISFIES*0..]-(ms)-
RETURN count(*)
```

Yields the plan:

```
Compiler CYPHER 2.1; Planner RULE
+-------------------+------+--------+------------------------------------------------------------
|     Operator tree | Rows | DbHits |
+-------------------+------+--------+------------------------------------------------------------
| ColumnFilter      |    1 |      0 |
| |                 +------+--------+------------------------------------------------------------
| +EagerAggregation |    1 |      0 |
| |                 +------+--------+------------------------------------------------------------
| +PatternMatcher   |   20 |      0 | sc, mp,   UNNAMED65, _mr, c,   UNNAMED89, ms,   UNNA
| |                 +------+--------+------------------------------------------------------------
| +TraversalMatcher |    2 |    545 |
+-------------------+------+--------+------------------------------------------------------------
```

The query:

Labels
None yet

Milestone
No milestone

Assignee
No one assigned

3 participants

```
MATCH (c)<-[cc*0..0]-(sc:sc { name:'enterprise' })<-[]-(`_mr`:ms)-[sa:SATISFIES*0..]-(ms)-
WHERE id(sc)=6
RETURN count(*)
```

Yields the plan:

```
Compiler CYPHER 2.1; Planner RULE
+--------------------+------+--------+------------------------------------------------------------
|     Operator tree | Rows | DbHits |
+--------------------+------+--------+------------------------------------------------------------
| ColumnFilter      |    1 |      0 |
| |                 +------+--------+------------------------------------------------------------
| +EagerAggregation |    1 |      0 |
| |                 +------+--------+------------------------------------------------------------
| +PatternMatcher   |   22 |      0 | sc, mp,   UNNAMED65, _mr, c,   UNNAMED89, ms,   UNNA
| |                 +------+--------+------------------------------------------------------------
| +PatternMatcher   |    9 |      0 | sc, mp,   UNNAMED65, _mr, c,   UNNAMED89, ms,   UNNA
| |                 +------+--------+------------------------------------------------------------
| +Filter           |    9 |     27 |
| |                 +------+--------+------------------------------------------------------------
| +TraversalMatcher |    9 |     37 |
+--------------------+------+--------+------------------------------------------------------------
```

systay commented Apr 24, 2015 Owner

Hiya!

Writing `(c)<-[cc*0..0]-(sc:sc { name:'enterprise' })` is a little weird. Why are you using *0..0? Although it's legal Cypher, I don't follow why you want that.

As for the confusing results - Neo4j 2.1 has some bugs around relationship uniqueness that has been fixed in Neo4j 2.2. You should see consistent numbers on that version. Have you had the chance to try Neo4j 2.2?

Best regards,

Andres

leocrawford commented Apr 25, 2015

Hi,

Thanks for taking a look at this so quickly. In answer to your question, the `*0..0` in this case was a debugging simplification from `*0..` which allowed me to see more clearly what was going wrong. However I do frequently use `(a)-[_cast*0..0]-(b:sc)` after an OPTIONAL MATCH as a sort of down cast from a to b. if you have views on better ways of approaching this I'd be hugely grateful.

In response to the main question, I had tried cypher 2.2, but got very different results from 2.1 and hadn't had a chance to figure out why. I have now raised a defect (#4527) for this as I'm pretty sure that's a separate bug in 2.2

I'm now in the position where both 2.1 and 2.2 seem to be broken in different ways. My cypher is being generated so it's a bit tricky to just work-around. All ideas much appreciated.

leocrawford commented Apr 25, 2015

I can confirm this bug is fixed in 2.2 once you work around #4527

leocrawford closed this Apr 25, 2015