Cypher query optimisation - Utilising known properties of nodes

asked 4 months ago

viewed 41 times

active 4 months ago

0

Setup:

Neo4j and Cypher version 2.2.0. I'm querying Neo4j as an in-memory instance in Eclipse created TestGraphDatabaseFactory().newImpermanentDatabase();. I'm using this approach as it seems faster than the embedded version and I assume it has the same functionality. My graph database is randomly generated programmatically with varying numbers of nodes.

Background:

I generate cypher queries automatically. These queries are used to try and identify a single 'target' node. I can limit the possible matches of the queries by using known 'node' properties. I only use a 'name' property in this case. If there is a known name for a node, I can use it to find the node id and use this in the start clause. As well as known names, I also know (for some nodes) if there are names known not to belong to a node. I specify this in the where clause.

The sorts of queries that I am running look like this...

```
START

nvari = node(5)

MATCH

 (target:C5)-[:IN_LOCATION]->(nvara:LOCATION),
(nvara:LOCATION)-[:CONNECTED]->(nvarb:LOCATION),
(nvara:LOCATION)-[:CONNECTED]->(nvarc:LOCATION),
(nvard:LOCATION)-[:CONNECTED]->(nvarc:LOCATION),
(nvard:LOCATION)-[:CONNECTED]->(nvare:LOCATION),
(nvare:LOCATION)-[:CONNECTED]->(nvarf:LOCATION),
(nvarg:LOCATION)-[:CONNECTED]->(nvarf:LOCATION),
(nvarg:LOCATION)-[:CONNECTED]->(nvarh:LOCATION),
(nvari:C4)-[:IN_LOCATION]->(nvarg:LOCATION),
(nvarj:C2)-[:IN_LOCATION]->(nvarg:LOCATION),
(nvare:LOCATION)-[:CONNECTED]->(nvark:LOCATION),
(nvarm:C3)-[:IN_LOCATION]->(nvarg:LOCATION),

WHERE

NOT(nvarj.Name IN ['nf']) AND NOT(nvarm.Name IN ['nb','nj'])

RETURN DISTINCT target
```

Another way to think about this (if it helps), is that this is an isomorphism testing problem where we have some information about how nodes in a query and target graph correspond to each other based on restrictions on labels.

Blog

Why Stack Overflow is a Good Workplace for Women

Related

24 Cypher - Return node if relationship is not present

3 Neo4j Cypher: Find exact match to array Node property in WHERE clause

4 Limiting a Neo4j cypher query results by sum of relationship property

6 Cypher query: Finding all paths between two nodes filtered by relationship properties

7 Return unique nodes in Cypher path query

1 Create relationship between nodes having same property value in common, using one Cypher query

1 cypher query match on an optional node

0 Cypher query to reach a terminal node

2 Cypher query to return one of each sub-nodes

0 Adding new node and relationship in a single cypher query

Hot Network Questions

How to show pagination "go to page" text box?

What's the philosophy behind delaying writing data to disk

How do I know when my sauce is reduced enough?

Capture output without redirection and leave it on terminal too

Why moving fan seems transparent?

Is it possible to automate tests for in progress sprint

Where did the nickname of 'Bones' for McCoy come from?

Approximation of Borel sets by a countable collection of majorants

Question:

With regards to optimisation:

1. Would it help to include relation variables in the match clause? I took them out because the node variables are sufficient to distinguish between relationships but this might slow it down?
2. Should I restructure the match clause to have match/where couples including the where clauses from my previous example first? My expectation is that they can limit possible bindings early on. For example...

   START

   nvari = node(5)

   MATCH

   (nvarj:C2)-[:IN_LOCATION]->(nvarg:LOCATION)

   WHERE NOT(nvarj.Name IN ['nf'])

   MATCH

   (nvarm:C3)-[:IN_LOCATION]->(nvarg:LOCATION)

   WHERE NOT(nvarm.Name IN ['nb','nj'])

   MATCH

   (target:C5)-[:IN_LOCATION]->(nvara:LOCATION), (nvara:LOCATION)-[:CONNECTED]->(nvarb:LOCATION), (nvara:LOCATION)-[:CONNECTED]->(nvarc:LOCATION),

   (nvard:LOCATION)-[:CONNECTED]->(nvarc:LOCATION), (nvard:LOCATION)-[:CONNECTED]->(nvare:LOCATION), (nvare:LOCATION)-[:CONNECTED]->(nvarf:LOCATION), (nvarg:LOCATION)-[:CONNECTED]->(nvarf:LOCATION), (nvarg:LOCATION)-[:CONNECTED]->(nvarh:LOCATION), (nvare:LOCATION)-[:CONNECTED]->(nvark:LOCATION)

   RETURN DISTINCT target

On the side:

3. (Less important but still an interest) If I make each relationship in a match clause an optional match except for relationships containing the target node, would cypher essentially be finding a maximum common sub-graph between the query and the graph data base with the constraint that the MCS contains the target node?

Thanks a lot in advance! I hope I have made my requirements clear but I appreciate that this is not a typical use-case for Neo4j.

neo4j cypher

share improve this question

asked Apr 11 at 15:43
Michael Anslow
69 ● 9

add a comment

1 Answer

active oldest votes

▲
1
▼
✓

1. I think querying with node properties is almost always preferable to using relationship properties (if you had a choice), as that opens up the possibility that indexing can help speed up the query.

   As an aside, I would avoid using the `IN` operator if the collection of possible values only has a single element. For example, this snippet: `NOT(nvarj.Name IN ['nf'])`, should be `(nvarj.Name <> 'nf')`. The current versions of Cypher might not use an index for the `IN` operator.

2. Restructuring a query to eliminate undesirable bindings earlier is exactly what you*should* be doing.

3. First of all, you would need to keep using `MATCH` for at least the first relationship in your query (which binds `target`), or else your result would contain a lot of `null` rows -- not very useful.

   But, thinking clearly about this, if all the other relationships were placed in separate `OPTIONAl MATCH` clauses, you'd be essentially saying that you want a match even if*none* of the optional matches succeeded. Therefore, the logical equivalent would be:

   ```
   MATCH (target:C5)-[:IN_LOCATION]->(nvara:LOCATION)
   RETURN DISTINCT target
   ```

   I don't think this is a useful result.

share improve this answer

Does Lebanon ban the diary of Anne Frank?
My 4.5 yrs old son has no dominant hand
Scraping High-Res image tiles from the MoMA website
Can a Glaive be stowed in a Ruby Scabbard?
Can you open source firmware to closed hardware?
Need to convert date format in apex
How do I calculate approximate equity liquidity?
If a stock doesn't pay dividends, then why is the stock worth anything?
Noun form of "extant"?
Does touching the tent outer wall from the inside make it leak?
Using Emergency Fund to Sell Upside-down Car
Why are carpenter's pencils flat?
Do companies only pay dividends if they are in profit?
Shut down list of servers given in a file
Straight line in TikZ with 'continued' marking in the middle
The Programming Language Quiz

share improve this answer

answered Apr 12 at 3:46
cybersam
9,941 ● 2 ● 7 ● 21

1. In that case, would chaining multiple inequalities with <> bet better than using NOT with IN? E.g. (nvarj.Name <> 'nf') AND (nvarj.Name <> 'ng') 2. Great, good to know it matches my intuition. I assume this is done in order from the first match declared to the last? 3. Good point. I suppose what I really want is possible valuations for target ordered by how many of the optional match clauses were bound. I suppose I would have to use variables for relationships, return these relationships and create this order manually? – Michael Anslow Apr 22 at 12:09

add a comment

Your Answer

Sign up or log in

Sign up using Google

Sign up using Facebook

Sign up using Stack Exchange

Post as a guest

Name

Email

required, but never shown

Post Your Answer

By posting your answer, you agree to the privacy policy and terms of service.

Not the answer you're looking for? Browse other questions tagged neo4j cypher or ask your own question.

question feed

tour help blog chat data legal privacy policy work here advertising info mobile **contact us** **feedback**

TECHNOLOGY

Stack Overflow
Server Fault
Super User
Web Applications
Ask Ubuntu
Webmasters
Game Development
TeX - LaTeX

Programmers
Unix & Linux
Ask Different (Apple)
WordPress Development
Geographic Information Systems
Electrical Engineering
Android Enthusiasts
Information Security

Database Administrators
Drupal Answers
SharePoint
User Experience
Mathematica
Salesforce
ExpressionEngine® Answers
more (13)

LIFE / ARTS

Photography
Science Fiction & Fantasy
Graphic Design
Movies & TV
Seasoned Advice (cooking)
Home Improvement
Personal Finance & Money
Academia
more (9)

CULTURE / RECREATION

English Language & Usage
Skeptics
Mi Yodeya (Judaism)
Travel
Christianity
Arqade (gaming)
Bicycles
Role-playing Games
more (21)

SCIENCE

Mathematics
Cross Validated (stats)
Theoretical Computer Science
Physics
MathOverflow
Chemistry
Biology
more (5)

OTHER

Stack Apps
Meta Stack Exchange
Area 51
Stack Overflow Careers