Social Network Benchmark: Business Intelligence workload

Gábor Szárnyas

szarnyas@db.bme.hu

10th LDBC TUC meeting

1 September, Munich
The SNB task force

Arnau Prat
Sparsity / DAMA-UPC
(Task Force Leader)

Marcus Paradies
SAP

Moritz Kaufmann
TUM / Tableau

Alex Averbuch
Neo4j

Gábor Szárnyas
BME / McGill
Social Network Benchmark

- Social network graph
- Scalable generator (DATAGEN)
- Analytical workload
  - Graphalytics (VLDB 2016)
- Query evaluation workloads
  - Interactive (SIGMOD 2015)
  - Business Intelligence
- Choke-point driven design
Challenges
Queries difficult to comprehend

Complex Reads Query Descriptions

Notes:

• Some queries require returning the content of a post. As stated in the schema, posts have content or imageFile, but not both. An empty string in content represents the post not having content, therefore, it must have a non empty string in imageFile and the other way around.

1. Friends with certain name

• Description: Given a start Person, find Persons with a given first name that the start Person is connected to (excluding start Person) by at most 3 steps via Knows relationships. Return Persons, including summaries of the Persons workplaces and places of study.

• Parameters:
  Person.id    ID
  Person.firstName  String

• Results:
  Person.id    ID
  Person.lastName  String
Outdated docs / inconsistencies

```sql
ADD
POST

3. Post.id
4. Post.imageFile
5. Post.creationDate
6. Post.locationIp
7. Post.browserUsed
8. Post.language
9. Post.content
10. Post.length
11. Post.hasCreator->Person.id
12. Forum.containedBy->Post.id
13. Post.isLocatedIn->Country.id
14. {Post-hasTag->Tag.id}
```

ID
String
DateTime
String
String
String
Text
32-bit Integer
ID
ID
ID
{ID}
Graphical notation
Query specification

• YAML files as a single source of truth
  • Generating query cards (TeX)
  • Generating wiki entries (Markdown) is also possible
• Python/Jinja2 for defining templates
workload: BI
operation: read
number: 17
title: Friend triangles
description: |

For a given country, count all the distinct triples of persons such that `a` is friend of `b`, `b` is friend of `c`, and `c` is friend of `a`.

Distinct means that given a triple $t_1$ in the result set $R$ of all qualified triples, there is not a triple $t_2$ in $R$ such that $t_1$ and $t_2$ have the same set of elements.

parameters:
- name: country
type: String

result:
- name: count
type: 32-bit Integer
category: aggregated

choke_points: [1.1, 2.3]
**Note:** In case of ambiguity, the textual description takes precedence.
Query evaluation workloads
Query evaluation workloads

- Business Intelligence
- Interactive

amount of data accessed vs. expected execution time
Interactive workload

- 14 complex read queries
- 8 short read queries
- 7 update queries

- Queries explore the graph around a given node

O. Erling (Openlink), A. Averbuch (Neo), J.L. LarribaPey (UPC), Hassan Chafi (Oracle Labs), Andrey Gubichev (TU Munich), Arnau Prat (DAMA-UPC), Minh-Duc Pham (VU Amsterdam), Peter Boncz (CWI).

The LDBC Social Network Benchmark: Interactive Workload. Proceedings of SIGMOD 2015, Melbourne
<table>
<thead>
<tr>
<th>query</th>
<th>Interactive / complex / 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>Recent posts and comments by your friends</td>
</tr>
</tbody>
</table>

### Pattern

![Graph pattern diagram]

| desc. | Given a start Person, find (most recent) Messages from all of that Person’s friends, that were created before (and including) a given date. |

### Params

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Person.id</td>
</tr>
<tr>
<td>2</td>
<td>date</td>
</tr>
</tbody>
</table>

### Result

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Message.hasCreator \rightarrow Person.id</td>
<td>ID</td>
</tr>
<tr>
<td>2</td>
<td>Message.hasCreator \rightarrow Person.firstName</td>
<td>String</td>
</tr>
<tr>
<td>3</td>
<td>Message.hasCreator \rightarrow Person.lastName</td>
<td>String</td>
</tr>
<tr>
<td>4</td>
<td>Message.id</td>
<td>ID</td>
</tr>
<tr>
<td>5</td>
<td>Message.content or Post.imageFile</td>
<td>String</td>
</tr>
<tr>
<td>6</td>
<td>Message.creationDate</td>
<td>DateTime</td>
</tr>
</tbody>
</table>

### Sort

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Message.creationDate ↓</td>
</tr>
<tr>
<td>2</td>
<td>Message.id ↑</td>
</tr>
</tbody>
</table>

### Limit

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20</td>
</tr>
</tbody>
</table>

### CPs

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.1, 2.2, 2.3, 3.2</td>
</tr>
</tbody>
</table>
Interactive workload
Business Intelligence workload

- 25 read queries
- Batch updates (not yet defined)

- Queries explore vast portions of the graph
query B1 / read / 7

<table>
<thead>
<tr>
<th>query</th>
<th>B1 / read / 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>Most authoritative users on a given topic</td>
</tr>
<tr>
<td>pattern</td>
<td>Given a Tag, find all Persons that ever created a Message with the given Tag. For each of these Persons compute their “authority score” as follows:</td>
</tr>
</tbody>
</table>
| desc.            | - The “authority score” is the sum of “popularity scores” of the Persons that liked any of that Person’s Messages with the given Tag.  
- A Person’s “popularity score” is defined as the total number of likes on all of their Messages. |
| params           | 1 tag 32-bit Integer |
| result           | 1 person1.id 64-bit Integer R  
2 authorityScore 32-bit Integer R |
| sort             | 1 authorityScore ↓  
2 person1.id ↑ |
<p>| limit            | 100 |
| CPs              | 1.2, 2.3, 3.2, 3.3, 6.1 |</p>
<table>
<thead>
<tr>
<th>query</th>
<th>BI / read / 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>Related topics</td>
</tr>
<tr>
<td>pattern</td>
<td><img src="#" alt="Diagram" /></td>
</tr>
<tr>
<td>desc.</td>
<td>Find all Messages that have a given Tag. Find the related Tags attached to replies of these Messages (direct relation not transitive), but only of those replies that do not have the given Tag. Group the Tags by name, and get the count of replies in each group.</td>
</tr>
<tr>
<td>params</td>
<td>1. <code>tag</code> 32-bit Integer</td>
</tr>
<tr>
<td>result</td>
<td>1. <code>relatedTag.name</code> String R</td>
</tr>
<tr>
<td></td>
<td>2. <code>count</code> 32-bit Integer R</td>
</tr>
<tr>
<td>sort</td>
<td>1. <code>count</code> ↓</td>
</tr>
<tr>
<td></td>
<td>2. <code>relatedTag.name</code> ↑</td>
</tr>
<tr>
<td>limit</td>
<td>100</td>
</tr>
<tr>
<td>CPUs</td>
<td>1.6, 3.3, 5.2</td>
</tr>
</tbody>
</table>
BI workload
Choke points
A.2 Join Performance

CP-2.1: [QOPT] Rich join order optimization

This choke-point tests the ability of the query optimizer to find optimal join orders. A graph can be traversed in different ways. In the relational model, this is equivalent as different join orders. The execution time of these orders may differ by orders of magnitude. Therefore, finding an efficient join (traversal) order is important, which in general, requires enumeration of all the possibilities. The enumeration is complicated by operators that are not freely re-orderable like semi-, anti-, and outer-joins. Because of this difficulty most join enumeration algorithms do not enumerate all possible plans, and therefore can miss the optimal join order. Therefore, these chokepoint tests the ability of the query optimizer to find optimal join (traversal) orders.

**Queries.**

<table>
<thead>
<tr>
<th>BI 2</th>
<th>BI 4</th>
<th>BI 5</th>
<th>BI 9</th>
<th>BI 10</th>
<th>BI 11</th>
<th>BI 19</th>
<th>BI 20</th>
<th>BI 21</th>
<th>BI 22</th>
<th>BI 24</th>
<th>BI 25</th>
</tr>
</thead>
</table>

*Interactive 1*  *Interactive 3*

Peter Boncz, Thomas Neumann, Orri Erling.

**TPC-H Analyzed: Hidden Messages and Lessons Learned from an Influential Benchmark.**

TPCTC 2013
Continuous integration

• Use Travis CI to
  • build DATAGEN
  • generate the technical report

• https://github.com/ldbc/ldbc_snb_docs
• https://github.com/ldbc/ldbc_snb_datagen
**LDPC Slack team**

[https://ldbcouncil.slack.com/](https://ldbcouncil.slack.com/)
Progress

- 54 Trello cards
- Documentation
  - 180+ commits
  - 14 issues
  - +12 LaTeX packages
- DATAGEN
  - 40+ commits
  - Talk by Arnau at 13:30
Roadmap

- Implement & validate for Neo4j, PostgreSQL and Sparksee
- Publish a subset of the benchmark in a workshop
  - GraphQ @ EDBT (late Nov)
  - GRADES @ SIGMOD (late March)
- Gather feedback & refine
- Define update operations

- We are recruiting!