Business Intelligence Workload: Progress Report and Roadmap

Gábor Szárnyas
szarnyas@mit.bme.hu

12th TUC meeting
Amsterdam
Motivation and History
Transaction Processing Performance Council

Standardized specifications for benchmarking RDBMSs

TPC price/perf trend 1990-2005: improved 58% per year, prices have declined 37%/y

TPC-C price/perf trend 37% per year

TPC-A

TPC-C


$/tx
Mission Statement

LDBC is a non-profit organization dedicated to establishing benchmarks, benchmark practices and benchmark results for graph data management software.

LDBC’s Social Network Benchmark is an industrial and academic initiative, formed by principal actors in the field of graph-like data management.
LDBC Benchmarks: Timeline

EU FP7 project
TUC meetings
Benchmark papers

SNB Interactive
SIGMOD 2015

Graphalytics
VLDB 2016

SPB
BLINK 2016

SNB BI
GRADES 2018

New directions: G-CORE language
Workloads at a Distance
Graph Analytics

Example:
Compute the local clustering coefficient.

Also BFS, PageRank, shortest paths, etc.
Typical complexity: $O(e), O(e + n \log n), O(n^{1.5}), ...$

$LCC(v) = \frac{d(v) - 1}{z(v) - 1}$

Alexandru Iosup et al.,
LDBC Graphalytics,
VLDB 2016
Graph Queries: Local

Example:
Return “Dan” and his comments.

Also known as “point queries”.

Most queries require $O(\log n)$ steps.

Orri Erling et al.,
The LDBC Social Network Benchmark: Interactive Workload, SIGMOD 2015
Graph Queries: Global

Example:
Find people who had no interaction with “Cecil” through any comments, neither replying nor receiving a reply.

Complexity: $\mathcal{O}(n)$, $\mathcal{O}(n \log n)$, …, $\mathcal{O}(n^2)$.

Gábor Szárnyas et al., An early look at the LDBC Social Network Benchmark’s Business Intelligence Workload, GRADES-NDA 2018
Graph Processing Landscape

- **BFS**
- **PageRank**
- **Clustering coefficient**
- **Dijkstra**
- **Bellman-Ford**
- **Floyd**

**t**

- basic structure
- +weights
- +properties

**level of detail**

- Global queries
- Local queries
Graph Processing Landscape

- BFS
- PageRank
- Dijkstra
- Clustering
- Coefficient
- +weights
- +properties

Global queries
Local queries
Interactive
BI
Graphalytics
Floyd
Social Network Benchmark
The Interactive and the BI workloads
SNB Task Force

Gábor Szárnyas  MTA-BME
Arna Prat  Sparsity, DAMA-UPC
Alex Averbuch  Neo4j
József Marton  BME
J. B. Antal  BME
Jack Waudby  Newcastle University
Ben Steer  QMUL
Overview

Benchmark artifacts

Benchmark specification → DATAGEN → Driver → Reference implementation → Generate validation data → Reference validation data

Social network data sets → Substitution parameters

Developer artifacts

Tool implementation

Validation

Validation configuration → Generate validation data → Tool's validation data → Cross-validation

Benchmarking

Benchmark configuration → Run benchmark → Benchmark results → Result analysis
Datagen: Social Network Graph

- Produces graphs in different scale factors, e.g. SF1 = 1GB, SF1000 = 1TB
- Produces 3 years of activity
  - 90% for initial data
  - 10% for updates
- Uses Hadoop for scalability
- **Challenge**: Cumbersome to set up
- **Progress**: Added Docker support
  - Quick to set up
  - Single machine can easily scale up to SF300

Minh-Duc Pham, Peter Boncz, Orri Erling, *S3G2: A Scalable Structure-Correlated Social Graph Generator*, TPCTC 2012
Paramgen: Parameter Generator

- Parameter curation for selecting inputs for queries
  - Correlations, e.g. people in neighbouring countries
  - Paramgen uses Datagen’s output to create input parameters

**Challenge:** Current implementation is very slow
- Single-threaded Python code
- Multiple users reported this issue (=good!)

**Progress:** Rewriting in Julia
- Faster and parallelizable
- Also runs in Docker

Andrey Gubichev, Peter Boncz, *Parameter Curation for Benchmark Queries*, TPCTC 2014
Design Methodology: Choke Points

- Challenging aspects of query processing, allows systematic design of queries

### 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
Choke Points

1. Aggregation performance
2. Join performance (+extensions?)
3. Data access locality
4. Expression calculation
5. Correlated sub-queries
6. Parallelism and concurrency (+extensions?)
7. RDF and graph-specifics (+extensions?)
8. Language features (+extensions?)
9. Data manipulation and consistency (new)
<table>
<thead>
<tr>
<th>Interactive workload</th>
<th>Complex reads</th>
<th>Short reads</th>
<th>Updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td></td>
<td>S1</td>
<td>U1-2</td>
</tr>
<tr>
<td>C2</td>
<td></td>
<td>S2</td>
<td>U3-5</td>
</tr>
<tr>
<td>C3</td>
<td></td>
<td>S3-4</td>
<td>U6</td>
</tr>
<tr>
<td>C4</td>
<td></td>
<td>S5-6</td>
<td>U7-8</td>
</tr>
<tr>
<td>C5</td>
<td></td>
<td>S7</td>
<td></td>
</tr>
<tr>
<td>C6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C14</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
BI workload

Complex reads
Interactive Q14: Trusted connection paths

Enumerate all shortest paths on knows edges from person1 to person2.

<table>
<thead>
<tr>
<th>person1: Person</th>
<th>knows*</th>
<th>person2: Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>id = $person1Id</td>
<td></td>
<td>id = $person2Id</td>
</tr>
</tbody>
</table>

There might be many such paths and communication between these people.

Calculate weight for each path:

\[ p1 \rightarrow w_1 \rightarrow p_2 \]
\[ \sum \text{weights} \]

Possible opt.: precompute weights for each knows edge.

1 point

- Person 🔄 Person
- Comment 🔄 Post

0.5 points

- Person 🔄 Person
- Comment 🔄 Comment

Order according to weight descending:

- p1 🔄 pA 🔄 pD 🔄 p2: 14.5
- p1 🔄 pB 🔄 pD 🔄 p2: 9
- p1 🔄 pB 🔄 pE 🔄 p2: 5.5
- p1 🔄 pC 🔄 pE 🔄 p2: 4
Enumerate all shortest paths on knows edges from person1 to person2

<table>
<thead>
<tr>
<th>person1: Person</th>
<th>knows*</th>
<th>person2: Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>id = $person1Id</td>
<td></td>
<td></td>
</tr>
<tr>
<td>id = $person2Id</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

There might be many such paths and communication between these people.

Calculate weight for each path

\[ \sum \text{weights} \]

1 point

- Person
- Person
- Comment
- Post
- Forum
- date

0.5 points

- Person
- Person
- Comment
- Comment
- Forum
- date
- Post

Order according to weight descending

8.5

4
Implementing an LDBC Workload

1. Generate data set
2. Implement loader
3. Implement queries

Validation
1. Generate validation data sets
2. Cross-validate for multiple SFs
3. If validation fails, fix issues and go to 2.

Validation is very time consuming, but indispensable.
• With 2 tools validated, there were bugs in both implementations
• With 3 tools validated, there were ambiguities in the spec
SNB Workloads

- **Challenge:** Helping users and pushing BI towards publication

- **Progress:**
  - Resolved many ambiguities
  - Reworked Interactive implementations [János, József, Gábor]
  - Interactive and BI can be mixed, e.g. BI reads + Interactive updates [Gábor]
  - Choke points for data manipulation [Gábor]
  - Sketched streaming features [Ben]
  - Revisited consistency criteria [Jack]
Possible Source of Ideas: Work at TTC 2018

LDBC-inspired benchmark at the 2018 Transformation Tool Contest, an annual event within the Model-Driven Engineering community (graph transformations)

Simplified schema

Interesting query: For each Comment, determine connected components in the subgraph induced by likes/knows edges

Georg Hinkel,
The TTC 2018 Social Media Case,
Transformation Tool Contest 2018
Summary and Roadmap
## Summary of Progress

<table>
<thead>
<tr>
<th>Trello cards</th>
<th>10(^{th}) TUC Munich</th>
<th>11(^{th}) TUC Austin</th>
<th>12(^{th}) TUC Amsterdam</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification</td>
<td>180</td>
<td>250</td>
<td>100</td>
</tr>
<tr>
<td>Datagen</td>
<td>40</td>
<td>50</td>
<td>30</td>
</tr>
<tr>
<td>Driver/impl.</td>
<td>20</td>
<td>600</td>
<td>260</td>
</tr>
<tr>
<td>Total</td>
<td>240</td>
<td>900</td>
<td>390</td>
</tr>
</tbody>
</table>

Integrated BI queries: 
Published GRADES’18 paper: 
Reworked Interactive and driver:
Roadmap

- Add more path queries
  - Algorithmic and language challenge
  - PGQL, G-CORE, GQL (talks in session #2)

- Add more cyclic queries
  - Algorithmic challenge
  - Worst-case optimal joins (talk in session #3)

- Add updates
  - Delete operations
  - Consistency

- Submit a paper for a 2020 conference
Transition from Batch to Streaming Graph Processing

Benjamin A. Steer

12th TUC meeting
Amsterdam
Current BI Ethos

Datagen Output:
Three years of activity

Batch loaded into DB of choice

Run your chosen queries

Results pop out
Month by Month Micro Batching

Full Three Years of Activity

Jan 2010  Feb 2010  Mar 2010  ...  Dec 2012

Static graph

Insert static graph into DB

Start Ingesting Updates
Month by Month Micro Batching

Jan 2010

Insert first set of updates

Run your chosen queries

Get results for current graph state
Month by Month Micro Batching

Dec 2012

Insert each set of updates

Run your chosen queries after each batch

Get results for each graph state
Why This Would Be Useful?

**CP9. Data manipulation and consistency**

- **CP9.1 Inserting data items**
- **CP9.2 Deleting data items**
  - Removal of simple edges (Friendship)
  - Simple vertex removal (Comments)
  - Complex vertex removal (Forums or People)
- **CP9.3 Refreshing data items**

- Provides a new avenue for BI queries
- Milestone towards continuous ingestion alongside queries
BI Streaming

Ingest initial graph into DB

Get outcoming stream of results

Three Years of Updates as Event Stream
Extending Existing Choke Points

CP6. Parallelism and concurrency
• CP6.1 Inter-query result reuse
• CP6.2 Asynchronous query execution
• CP6.3 Parallel query execution
Future Streaming Components

• Ingestion model
  • Varying throughput – provide different traffic patterns
  • Chaos Monkey / Simian Army: https://github.com/netflix/chaosmonkey

• Sets of queries (workflows)
  • Derived edges, vertices or properties
  • Combine with Ingestion model – establish a set workload

• If you are interested in any of these
  • https://graphtides.github.io/
Updates and Consistency for Transactional Graph Processing

Jack Waudby

12th TUC meeting
Amsterdam
Interactive Workload

• Transaction processing benchmark (OLTP)

• Focus on exercising: transaction integrity (ACID properties), etc.

• Updates are append-only, e.g. add Person, add Post, add Friendship
Key Points

1. Current specification is unclear

2. Extending update scenarios
Rules

1. All transactions have ACID guarantees

2. Stable throughput

3. Latencies of complex read-only queries are stable

4. At least 2 hours of simulation time

5. Actual start time – scheduled start time < 1 second
Interactive Paper (SIGMOD’15)

**Rules and Metrics.** Since the scope of our benchmark in terms of systems is very broad, we do not pose any restrictions on the way the queries are formulated. In fact, the preliminary results presented below were achieved by a native graph store (no declarative query language, queries formulated as programs using API) and a relational database system (queries in SQL with vendor-specific extensions for graph algorithms). Moreover, usage of materialized views (or their equivalents) is not forbidden, as long as the system can cope with updates. We require that all transactions have ACID guarantees, with serializability as a consistency requirement. Note that given the nature of the update workload, systems providing snapshot isolation behave identically to serializable.

Serializability
SNB-Interactive. This workload consists of a set of relatively complex read-only queries, that touch a significant amount of data, often the two-step friendship neighborhood and associated messages. Still these queries typically start at a single point and the query complexity is sublinear to the dataset size. Associated with the complex read-only queries are simple read-only queries, which typically only lookup one entity (e.g. a person). Concurrent with these read-only queries is an insert workload, under at least read committed transaction semantics. All data generated by the SNB data generator is timestamped, and a standard scale factor covers three years. Of this 32 months are bulkloaded at benchmark start, whereas the data from the last 4 months is added using individual DML statements.
Rules

• Let's be clearer!

• Executing the current workload under **Read Committed** isolation is equivalent to **Serializable** isolation
Extending Update Scenarios

Test:

• Simultaneous execution of multiple transaction types that span a breadth of complexity

• Contention on data access and update
Extending Update Scenarios

Simple deletions:
• Delete “knows” edges
• Delete “Comment” nodes

Complex deletions:
• Delete “Person” nodes
• Delete “Forum” nodes

Complex Read-Write Transactions:
• Identify and delete subgraph of trolls

Serializable isolation is a requirement in the presence of deletions
Example

<table>
<thead>
<tr>
<th>query</th>
<th>Interactive / complex / 13</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>Single shortest path</td>
</tr>
<tr>
<td>pattern</td>
<td></td>
</tr>
</tbody>
</table>

![Graph Diagram]

- Tx complex read 13
- Ty delete Edge(n3,n5)
- Tz add Edge(n5,n7)

Tx returns n7 reachable from n1

Ayush Dubey et al.,
*Weaver: A High-Performance, Transactional Graph Database Based on Refinable Timestamps,*
VLDB 2016
Example cont.

\[ T_x: \quad R(n_1, n_3) \quad R(n_3, n_5) \quad R(n_5, n_7) \]

\[ T_y: \quad D(n_3, n_5) \]

\[ T_z: \quad W(n_5, n_7) \]
# What TPC Benchmarks Do

Although arbitrary, the transaction $T_n$ may not do dirty writes.

The following table defines the isolation requirements which must be met by the TPC-C transactions:

<table>
<thead>
<tr>
<th>Req. #</th>
<th>For transactions in this set:</th>
<th>these phenomena:</th>
<th>must NOT be seen by this transaction:</th>
<th>Textual Description:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>${T_i, T_j}$</td>
<td>P0, P1, P2, P3</td>
<td>$T_i$</td>
<td>Level 3 isolation between New-Order, Payment, Delivery, and Order-Status transactions.</td>
</tr>
<tr>
<td>2.</td>
<td>${T_i, T_n}$</td>
<td>P0, P1, P2</td>
<td>$T_i$</td>
<td>Level 2 isolation for New-Order, Payment, Delivery, and Order-Status transactions relative to any arbitrary transaction.</td>
</tr>
<tr>
<td>3.</td>
<td>${T_i, T_5}$</td>
<td>P0, P1</td>
<td>$T_5$</td>
<td>Level 1 isolation for Stock-Level transaction relative to TPC-C transactions and any arbitrary transaction.</td>
</tr>
</tbody>
</table>

Sufficient conditions must be enabled at either the system or application level to ensure the required isolation defined above is obtained.

The following terms are defined:

- $T_1 =$ New-Order transaction
- $T_2 =$ Payment transaction
- $T_3 =$ Delivery transaction
- $T_4 =$ Order-Status transaction
- $T_5 =$ Stock-Level transaction
- $T_n =$ Any arbitrary transaction
What TPC Benchmarks Do

• Perform series of tests to ensure ACID properties

• Tests provided in spec

Comment: These tests are intended to demonstrate that the ACID principles are supported by the SUT and enabled during the performance measurement interval. They are not intended to be an exhaustive quality assurance test.
Summary

1. Clearer on rules

2. Incorporate ACID properties testing into validation step (provide test scripts)

3. Develop update scenarios – realistic read-write transactions?