LDBC Social Network Benchmark

Business Intelligence Workload

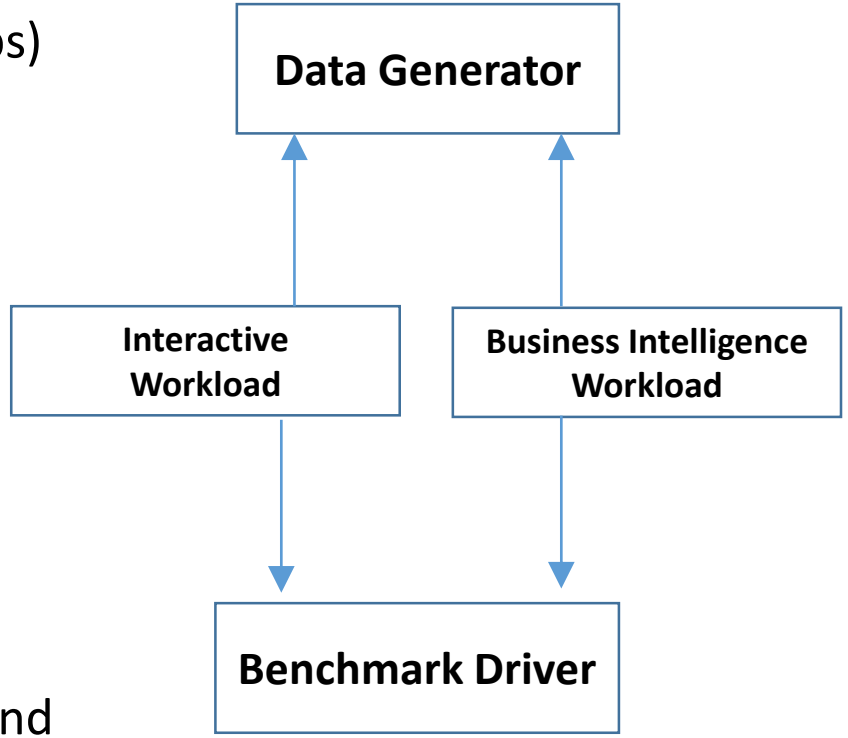
Marcus Paradies, SAP SE

Task Force Members

- Task Force members
  - Arnau Prat (DAMA-UPC)
  - Alex Averbuch (Neo Technologies)
  - Moritz Kaufmann (TU München)
  - Marcus Paradies (SAP)
  - Orri Erling (Google)
  - Peter Boncz (CWI)

Interactive and BI Workload

- Interactive workload
  - Only touch a small fraction of the data graph (< 3 hops)
  - Queries usually start at a single person vertex
  - Can be seen as OLTP-style workload
- Business intelligence workload
  - Focus on analytic queries (grouping/aggregations)
  - Queries touch the complete graph
  - Batch updates
  - Can be seen as OLAP-style workload
  - 24 queries in the BI workload
- Interactive and BI workload share the benchmark driver and generated data
- Workload-specific parameter bindings to reduce variability between queries



Examples Queries

Q1 - Posting Summary

Given a date, find all Messages created before that date.
Group them by a 3-level grouping:

1. by year of creation
2. for each year, group into message types, i.e., Posts or Comments
3. for each year-type group, split into four groups based on length of their content:
   0 <= length < 40 → *short*
   40 <= length < 80 → *one liner*
   80 <= length < 160 → *tweet*
   160 <= length → *long*

   Parameters:
   date: Date

Examples Queries

Q1 - Posting Summary

Result: (*)

For every 3-level group, return:

- year - 32-bit Integer
- message type → post/comment
- length category → (short/one-liner/tweet/long)
- message count → total number of Messages (Posts/Comments) in that group
- average message length → average length of the Message content in that group
- sum message length → sum of all message content lengths
- per messages → number of messages in a group as a percentage of all messages created before the given date

** Final sorting and top k omitted here for brevity*

Examples Queries

Q16 - Experts in a Social Circle

Given a Person, find all other Persons that live in a given country and are connected to given person by a path through the *knows* relation.
For each of these Persons, retrieve all of their Messages (Posts & Comments) that contain at least one Tag belonging to a given TagClass (direct relation not transitive). For each Message, also retrieve its Tags.

Parameters:

Person.id - 64-bit Integer
Country.name - String
TagClass.name - String

Examples Queries

Q16 - Experts in a Social Circle

Grouping of results:

First, by Tag.name

Second, by Person.id

Result: (*)

For each group, return:

- Person.id
- Tag.name
- post_count → number of Messages created by that Person containing that Tag

** Final sorting and top k omitted here for brevity*

SNB BI Workload in a Nutshell

Q1 - Posting summary
Q2 - Top tags for country, age, gender, time
Q3 - Tag evolution
Q4 - Popular topics in a country
Q5 - Top posters in a country
Q6 - Most active Posters of a given Topic
Q7 - Most authoritative users on a given topic
Q8 - Related Topics
Q9 - Forum with related Tags
Q10 - Central Person for a Tag
Q11 - Unrelated Replies
Q12 - Trending Posts
Q13 - Popular Tags per month in a country
Q14 - Top thread initiators
Q15 - Social Normals
Q16 - Experts in Social Circle
Q17 - Friend Triangles
Q18 - How many persons have a given number of posts
Q19 - Stranger's Interaction
Q20 - High level topics
Q21 - Zombies in a country
Q22 - International Dialog
Q23 - Holiday Destinations
Q24 - Messages By Topic And Continent

Choke Point Identification

- Choke point = implementation challenge
- Interactive workload already contains a number of choke points
- We extend them by more choke points from TPC-H that are relevant for BI queries

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22
CP1 Aggregation Performance. Performance of aggregate calculations.																					
CP1.1 QEXE: Ordered Aggregation.																					
CP1.2 QOPT: Interesting Orders.																					
CP1.3 QOPT: Small Group-by Keys (array lookup).																					
CP1.4 QEXE: Dependent Group-By Keys (removal of).																					
CP2 Join Performance. Voluminous joins, with or without selections.																					
CP2.1 QEXE: Large Joins (out-of-core).																					
CP2.2 QEXE: Sparse Foreign Key Joins (bloom filters).																					
CP2.3 QOPT: Rich Join Order Optimization.																					
CP2.4 QOPT: Late Projection (column stores).																					
CP3 Data Access Locality. Non-full-scan access to (correlated) table data.																					
CP3.1 STORAGE: Columnar Locality (favors column storage).																					
CP3.2 STORAGE: Physical Locality by Key (clustered index, partitioning).																					

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

Peter Boncz[1], Thomas Neumann[2], and Orri Erling[3]

[1] CWI, Amsterdam, The Netherlands
boncz@cwi.nl
[2] Technical University Munich, Germany
neumann@in.tum.de
[3] Openlink Software, United Kingdom
oerling@openlinksw.com

Abstract. The TPC-D benchmark was developed almost 20 years ago, and even though its current existence as TPC-H could be considered superseded by TPC-DS, one can still learn from it. We focus on the technical level, summarizing the challenges posed by the TPC-H workload as we now understand them, which we call "choke points". We identify

Current Status of the BI Workload

- Query Definition
  - All 24 queries are specified in plain english text
- Benchmark Driver
  - Reads generated parameter bindings and issues queries
  - Query mix has to be defined
- 22/24 queries are validated against Sparsity,Neo4j, and Postgres
- Choke points identified for 8/24 queries

Link to Postgres draft reference implementation:

https://github.com/ldbc/ldbc_snb_implementations/tree/new_bi/bi/postgres/src/main/sql/postgres/queries/bi

Outlook and Next Steps

- Finishing of chokepoint identification and analysis of whether all defined chokepoints are already triggered
- Finishing validation of remaining two queries
- Addition of refresh data sets (update batches)
  - Definition
  - Implementation in the benchmark driver
- Final polishing of query specifications