Towards Testing ACID Compliance in the LDBC Social Network Benchmark

Jack Waudby, Benjamin A. Steer, Karim Karimov, József Marton, Peter Boncz, and Gábor Szárnyas

Linked Data Benchmark Council

LDBC is a non-profit organization established in 2012, dedicated to defining benchmarks and ***auditing results*** for graph data management software.

Similar goals to TPC's in the relational domain:

- facilitate fair comparison
- drive competition
- capture an understanding of the field

LDBC Structure

Task Forces;

- Social Network Benchmark
- Semantic Publishing Benchmark
- Graphalytics

Working Groups;

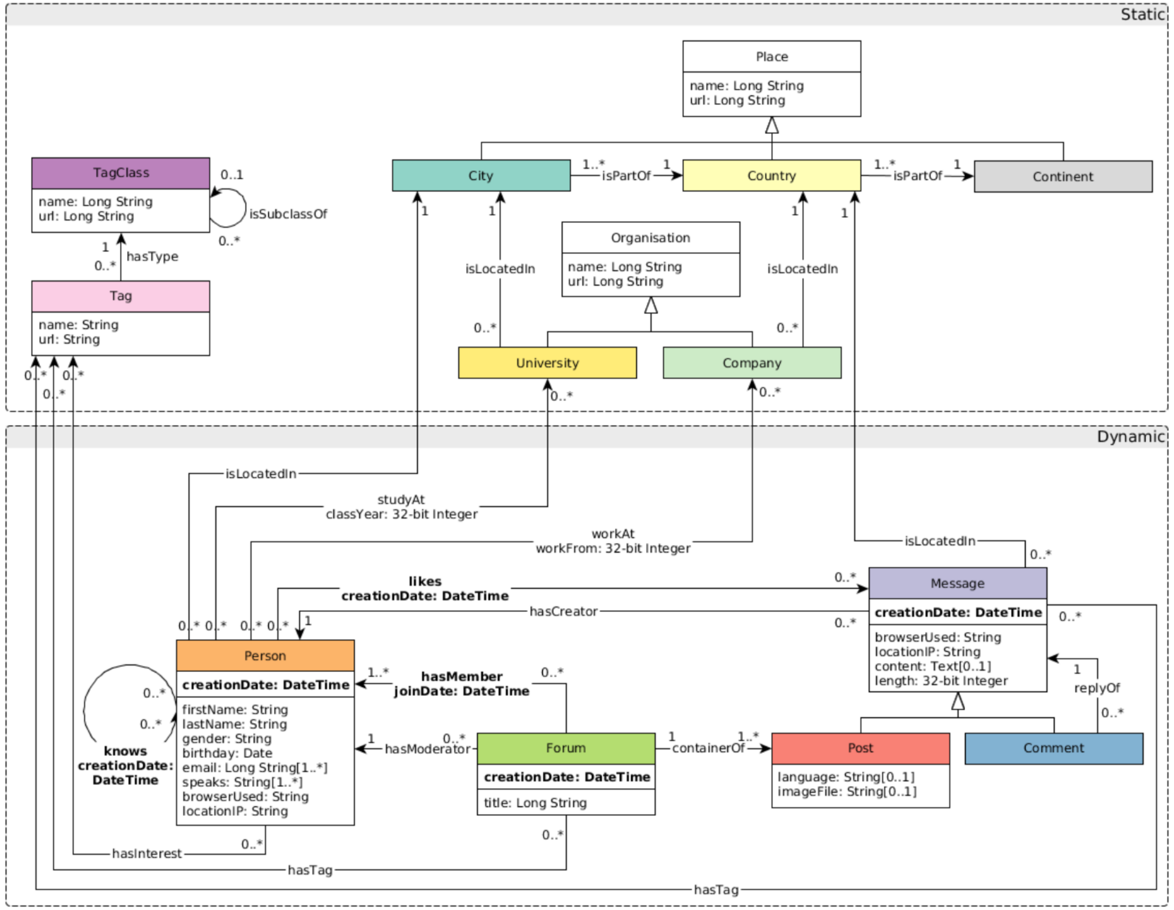
- Property Graph Schema
- Query Language (G-CORE)
- Formal Semantics (GQL)
- Existing Languages (literature review)

Membership;

- Non-profit orgs (ENS-Paris, University of Edinburgh)
- Commercial orgs (TigerGraph, Neo4j, Oracle, OpenLink Software, Ontotext, JCC Consulting, Intel)
- Individual members
- Associate members

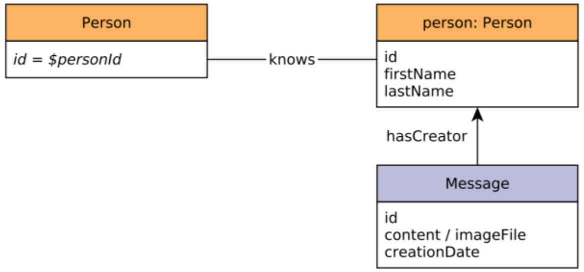
LDBC Social Network Benchmark (SNB) Suite

- Models a social network, e.g., Facebook
- People connect with each other and post messages in groups
- Correlated synthetic data produced by LDBC's Datagen
- 2 workloads; Interactive and Business Intelligence (BI)



LDBC Social Network Benchmark (SNB) Suite

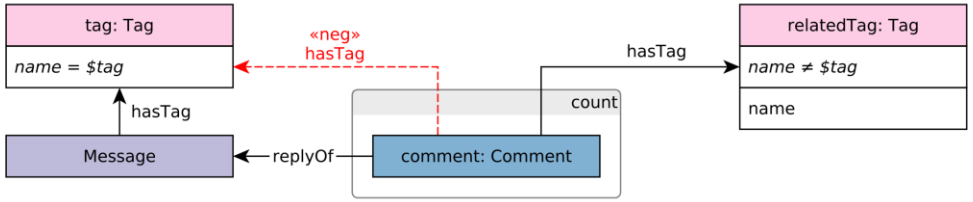
- **Interactive (OLTP)**
  - complex/short read queries, updates
  - explore neighborhood of a node/query
  - established, multiple implementations




The LDBC Social Network Benchmark: Interactive Workload

Erling et al.
The LDBC Social Network Benchmark: Interactive Workload,
SIGMOD 2015

- **BI (OLAP)**
  - currently read-only queries
  - large portion of the graph/query
  - ongoing adoption




An early look at the LDBC Social Network Benchmark's Business Intelligence workload

Szárnyas et al.
An early look at the LDBC Social Network Benchmark's Business Intelligence Workload,
GRADES-NDA 2018

Motivation

- Starting receiving requests for audits;
  - first audit completed July 2020, FMA's TuGraph
- ACID compliance important for fair comparison between systems
- No mechanism for validating ACID compliance*

→ **Design an ACID compliance test suite**

→ Focus on Atomicity and Isolation

**Durability test already part of benchmark specification*

Related Work

- **TPC-* Tests:**
  assume lock-based concurrency control, tests for 3 isolation anomalies
  → not generalizable, limited anomaly test coverage

- **Hermitage (Martin Kleppmann)**
  tests performed by hand
  → hard to induce anomalies that relied on fast timings

- **Jepsen (Kyle Kingsbury)**
  focuses on distributed systems under various failure modes
  → too heavyweight

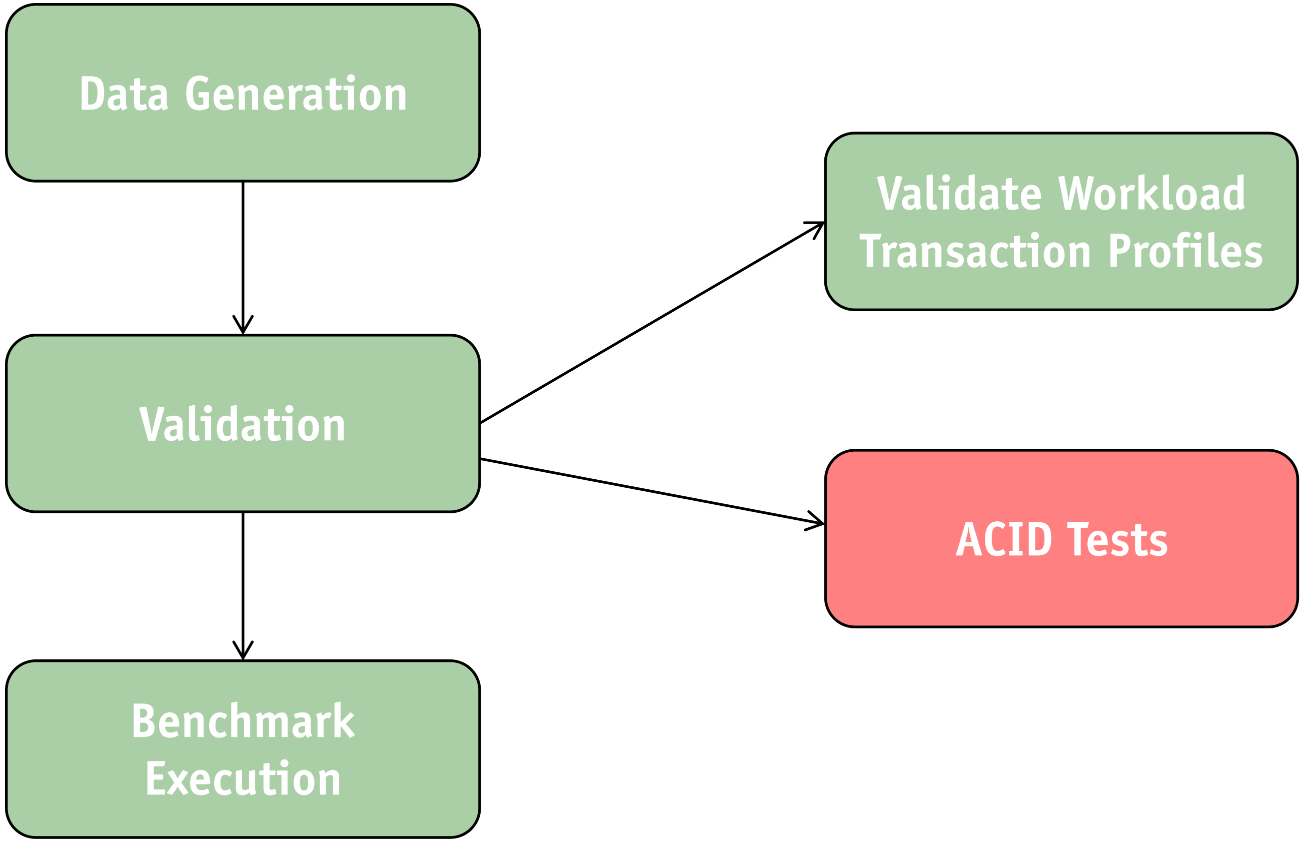
Design Considerations

Disclaimer: verifying ACID compliance with a finite set of tests is not possible

1. **Generalizable** – agnostic of system-level implementation details and query API
2. **Lightweight** – not add significant overhead to benchmarking process
3. **Improved Coverage** – test for more isolation anomalies, e.g., lost updates, write skew

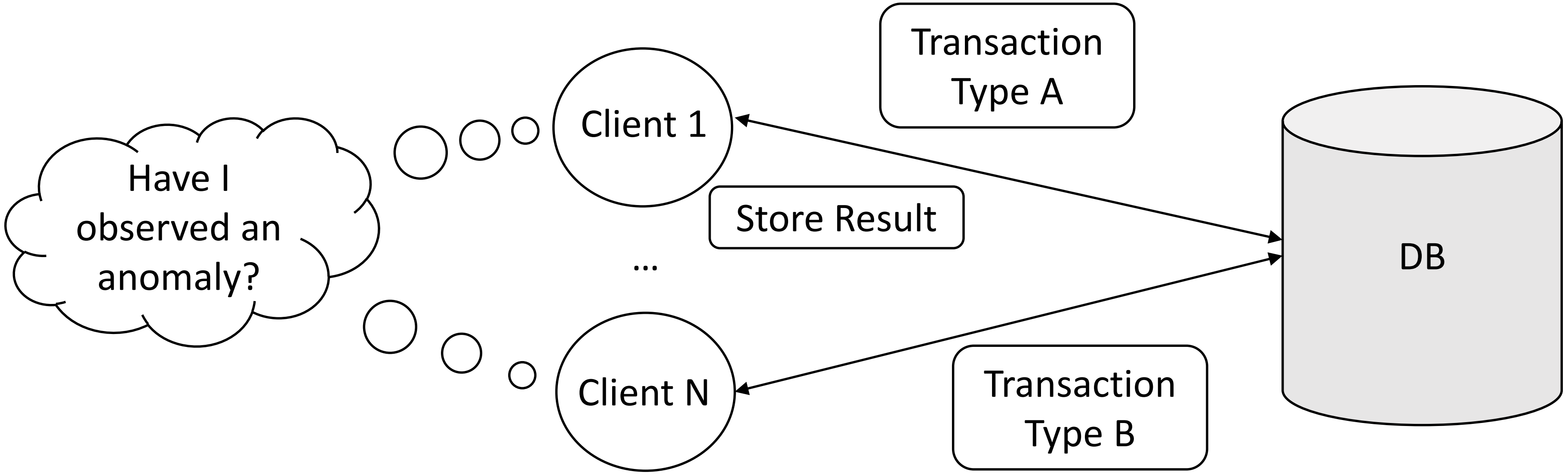
LDBC ACID Test Suite

Included in benchmark workflow as an additional step in the **Validation Phase**



LDBC ACID Test Suite Design

- Based solely on ***client observations*** to detect anomalies (generalizable)
- Each test consists of a handcraft set of transactions, which when interleaved create conditions in an anomaly could occur
- After execution, transaction results are gathered, and an ***anomaly check*** performed



LDBC ACID Test Suite Execution Flow

For each test;

1. Load required test graph
2. Initiate *N* clients
3. Execute test's transaction set for duration *T*
4. Gather transaction results from clients
5. Perform anomaly check

Test Coverage

- Atomicity;
  - Commit
  - Rollback
- Isolation;
  - Dirty Writes[1], Aborted Reads[1], Intermediate Reads[1], Circular Information Flow[1], Write Skew[1], Lost Updates[1], Item-Many Preceders[2], Predicate-Many Preceders[2], Observed Transaction Vanishes[2], and Fractured Reads[3].


Generalized Isolation Level Definitions

Adya et al. [1]
Generalized Isolation Level Definitions,
ICDE 2000


Highly Available Transactions: Virtues and Limitations

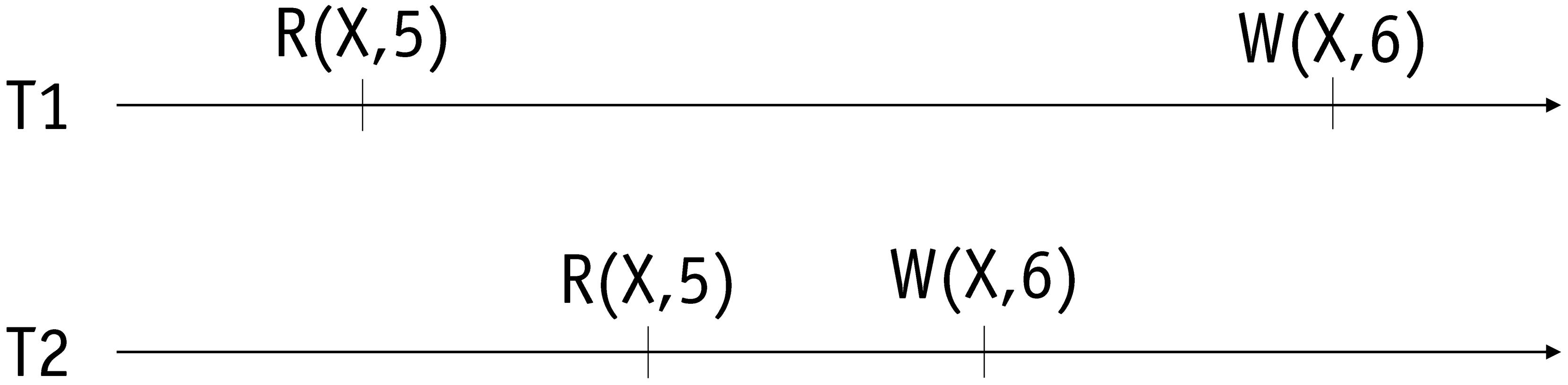
Bailis et al. [2]
Highly Available Transactions:
Virtues & Limitations,
VLDB 2014


Scalable Atomic Visibility with RAMP Transactions

Bailis et al. [3]
Scalable Atomic Visibility with
RAMP Transactions,
SIGMOD 2014

Example: Lost Updates

A **lost update** occurs when 2 transactions concurrently attempt to make conditional modifications to the same data item(s)



If T1 and T2 are executed sequential, X=7, T2’s update is ”lost”

Example: Lost Updates

Person
$id
$numFriends

1. Load a test graph containing *Person* nodes, with unique *id* and *numFriends* properties
2. Clients choose a random *Person* and increment its *numFriends* property

```
MATCH (p:Person {id: $personId})
SET p.numFriends = p.numFriends + 1
```

3. Clients store local counters (*expNumFriends*) for each *Person*, which is incremented each time a *Person* is selected and the transaction successfully commits
4. After execution, gather counters, then,
5. Perform anomaly check: for each *Person*,

   (global) *expNumFriends* = (observed) *numFriends*

Experimental Setup

- Implemented as extensible framework in a Java application
- Ubuntu 18.04 running AdoptOpenJDK 11.0.4.hs
- 4 graph database systems and 1 relational database;
  - Neo4j 3.5.20 and 4.1.1
  - Memgraph 1.0
  - Dgraph 20.07.0
  - JanusGraph 0.5.2 (BerkeleyDB 7.5.11 and Cassandra 3.11.0 backends)
  - PostgreSQL 9.6
- For all systems, we used their declarative query languages (Cypher, GraphQL, Gremlin, and SQL) and the officially recommended Java drivers

Results

- **Overall, most systems met their claims**
- Selected results;
  - Memgraph’s default isolation level is Snapshot Isolation; only Write Skew occurred
  - Neo4j’s default isolation level is Read Committed with *some* built-in protection again Lost Updates; v3.5.20 and v4.1.1 met requirements for Read Committed, but v4.1.1 displayed Lost Updates
  - JanusGraph had the worst user experience, passing some tests due to serving *extremely* stale reads; also, execution of some tests timed out
  - Dgraph’s default isolation level is Snapshot Isolation; passed our Write Skew test

Future Work

- Included test suite in LDBC Auditing Policies;
  - Used test suite in recent audit of FMA's TuGraph
- Extend test suite to incorporate complex consistency constraints;
  - Graph databases generally do not support constraints; sometimes domain and primary key constraints
  - Graph-specific constraints are expected to be introduced;
    - (Partial) compliance to a schema on top of property graphs,
    - Structural constraints, e.g., connectedness of the graph, absence of cycles, or arbitrary well-formedness constraints
  - LDBC PGSWG actively working in this area
- Add tests for distributed graph processing systems