LDBC

The LDBC Financial Benchmark: Transaction Workload

VLDB 2025, London, United Kingdom

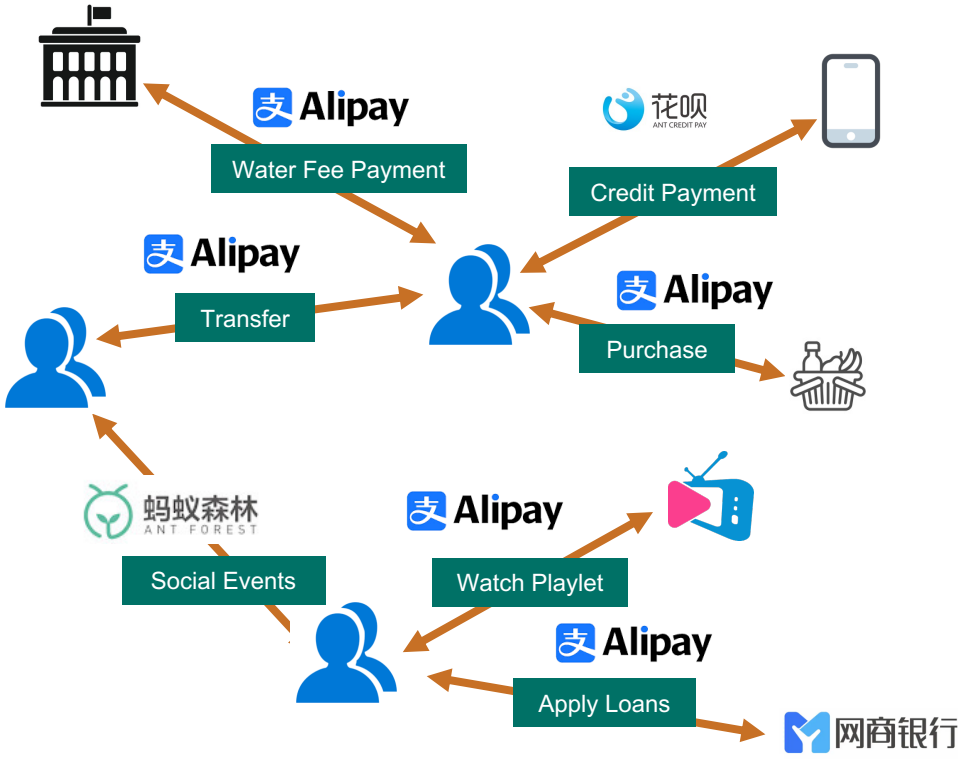
Research 60 - Database Engine Performance and Manageability II

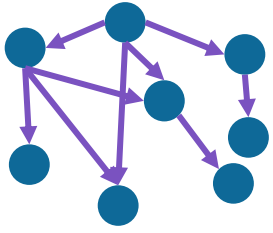
Presenter: Shipeng Qi @ AntGroup, LDBC (qishipengqsp@gmail.com)

Authors: **Shipeng Qi**, Bing Tong, Jiatao Hu, Heng Lin, Yue Pang, Wei Yuan, Songlin Lyu, Zhihui Guo, Ke Huang, Xujin Ba, Qiang Yin, Youren Shen, Yan Zhou, Tao Lv, Jia Li, Lei Zou, Yongwei Wu, Gábor Szárnyas, Xiaowei Zhu, Wenguang Chen, Chuntao Hong

(with contributions from members of the FinBench Working Group)

Inherently connected data in Ant Group





~1 billion
Total registered users of Alipay

~100 billion
Total transaction count of Alipay

10 thousands ~ 1 trillion
Actual graph size we face

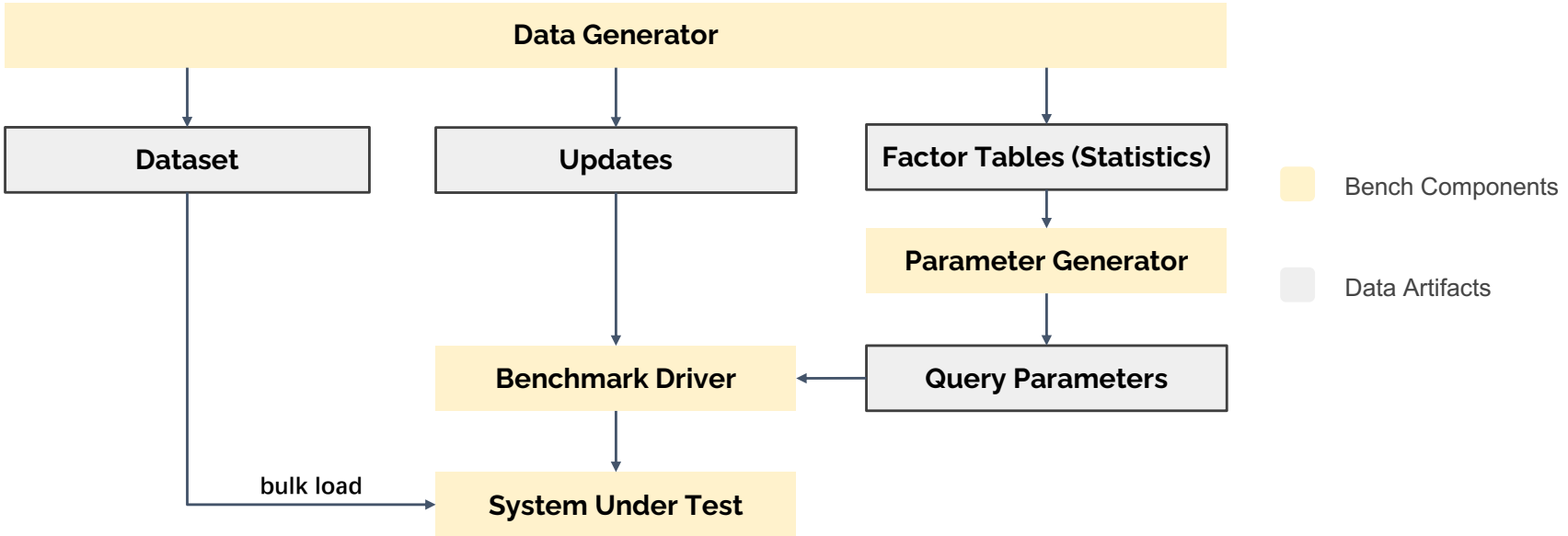
Motivation

Previous graph benchmarks fail to capture the characteristics of financial scenarios.

LDBC benchmarks	FinBench Transaction	SNB Interactive v1*	LinkBench	SNB BI	Graphalytics
labelled property graph	⊗	⊗	⊗	⊗	○
temporal property graph	⊗	⊗	○	⊗	○
edge multiplicity	⊗	○	○	○	○
insert operations	⊗	⊗	⊗	⊗	○
delete operations	⊗	○	⊗	⊗	○
query footprint	small	small	small	large	all data
inter-query parallelism	required	required	required	optional	not applicable
path finding	⊗	○	○	⊗	⊗
time-window queries	⊗	○	○	○	○
recursive path filtering	⊗	○	○	○	○
truncated traversal	⊗	○	○	○	○
read-write queries	⊗	○	○	○	○
workload type	OLTP	OLTP	OLTP	OLAP	graph algorithms
query mix	⊗	⊗	⊗	⊗	not applicable
time-biased query mix	⊗	○	○	○	not applicable

Benchmark Design Overview

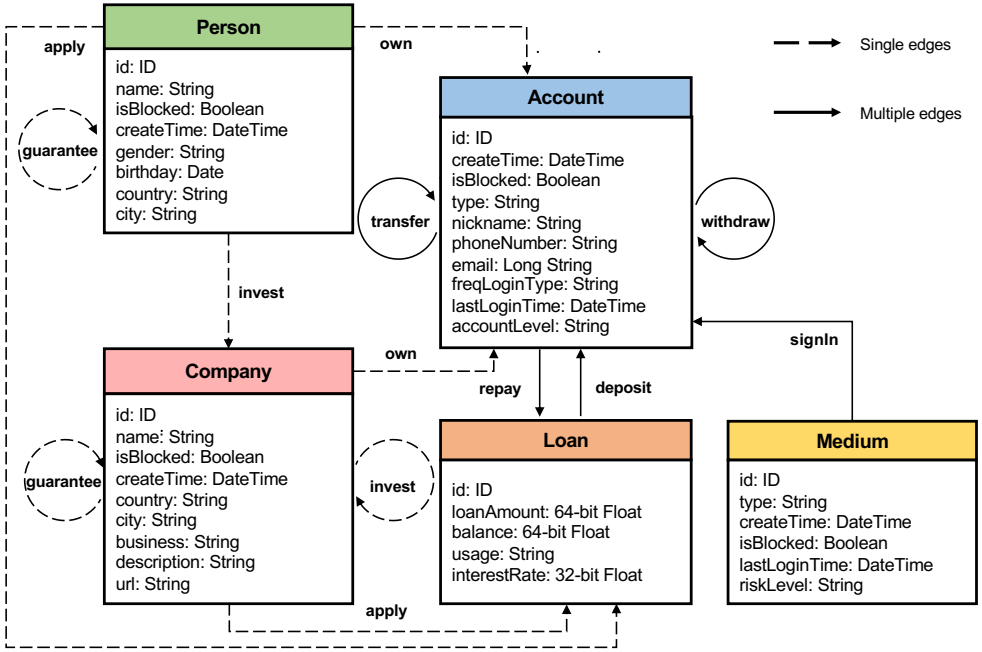
- Design philosophy: "**choke point driven methodology**"[1] to balance between the real-world business complexity and benchmark abstraction
- Key aspects: **dataset** (including schema and distribution), **query**, and **workload**



[1] Peter Boncz, Thomas Neumann, and Orri Erling. 2013. TPC-H Analyzed: Hidden Messages and Lessons Learned from an Influential Benchmark. In Revised Selected Papers of the 5th TPC Technology Conference on Performance Characterization and Benchmarking - Volume 8391. Springer-Verlag, Berlin, Heidelberg, 61–76. https://doi.org/10.1007/978-3-319-04936-6_5

Dataset Schema

Surveyed on over 20 business clusters and take typical transaction scenario as design



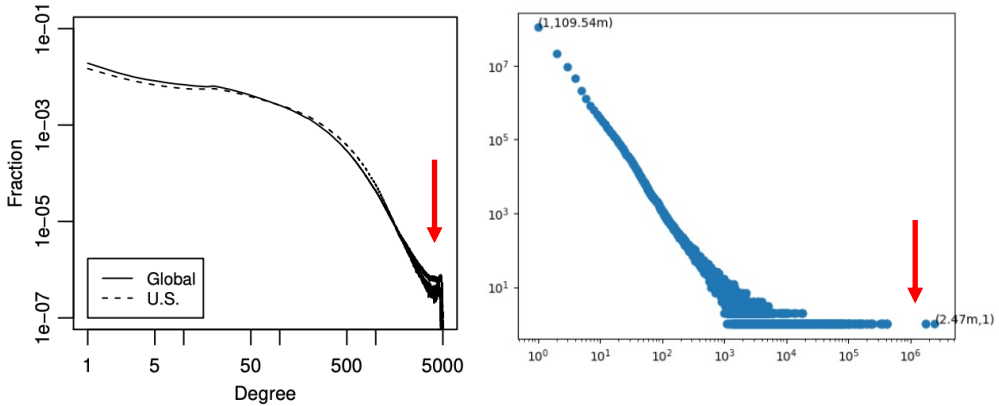
5 vertex types and 9 edge types

- Entities as vertices while activities as edges
- Timestamp frequently used
- Various subtypes of same entities
- Rich properties
- Edge multiplicity

Note: edge properties are omitted here.

Dataset Distribution

We randomly sampled three subgraphs from the online production environment by uniformly selecting vertices to ensure representative profiling



social network[2] financial graph

Power law distribution of the vertex degree

What is the difference?

Max 5000 friend limited by Facebook

↓

Social Network

↓

Bounded skewness with a shallower tail

V.S.

Unbounded skewness in FinBench

↓

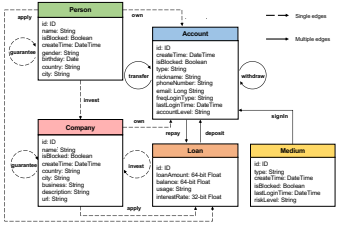
Financial Graphs

↓

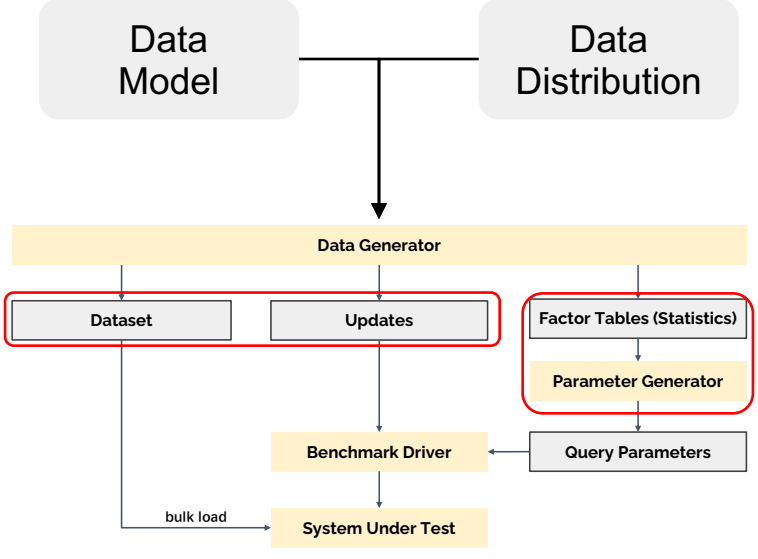
Hub vertices of **higher** degree

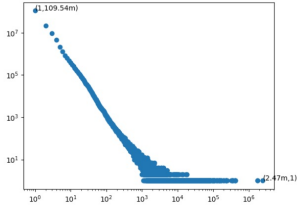
[2] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. 2011. The Anatomy of the Facebook Social Graph. arXiv:1111.4503 [cs.SI] Retrieved June 17, 2025 from https://arxiv.org/abs/1111.4503

Dataset Generation



The scalable and configurable data generator simulates the financial activities and then segments the overall data to **snapshot data for bulk loading** and **incremental data for write queries.**





Fact: The cost of graph queries varies depending on the starting vertex. The factor tables record **statistics of query footprints**, and are used to guide the read query parameter selection to ensure the benchmark reliability.

Case Study: Temporal Queries

- **Feature**: Query usually looks back in a limited time window. They filter edges between startTime and endTime in traversal.
- **Practices**: Hot/cold storage tiering, data compaction
- **Choke point**: Query benefits from temporal access locality (for edges).

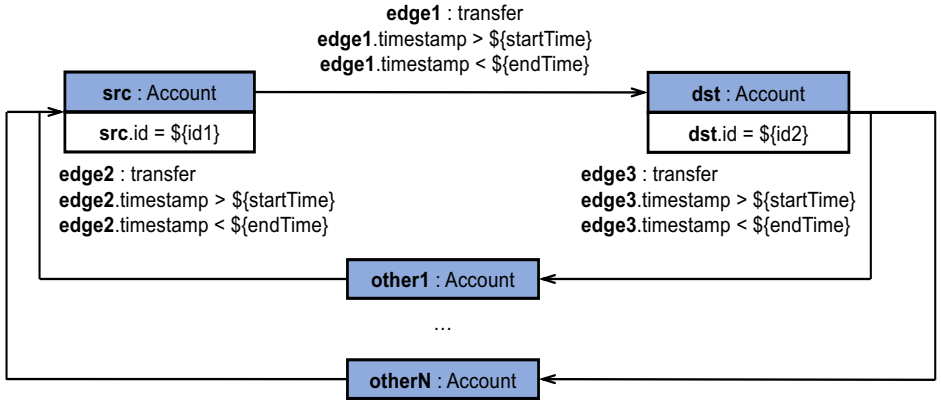


Figure 6: The pattern of complex read query 4 (TCR 4) in transaction workload.

Query Explanation:

TCR 4 matches a transfer cycle within a specified time window between *startTime* and *endTime.*

Case Study: Recursive Path Filtering

Given a path: A -[e_1]-> B -[e_2]-> ... -> X

- Timestamp order: $e_1 < \ldots < e_i$
- Amount order: $e_1 > \ldots > e_i$
- Optional: $e_i \in (e_{i-1}, e_{i-1} + k)$, k is const.

- **Feature**: Query matches paths filtered by monotonicity predicates.
- **Choke point**: Query benefits from path-level predicates push-down, pruning invalid intermediate results.

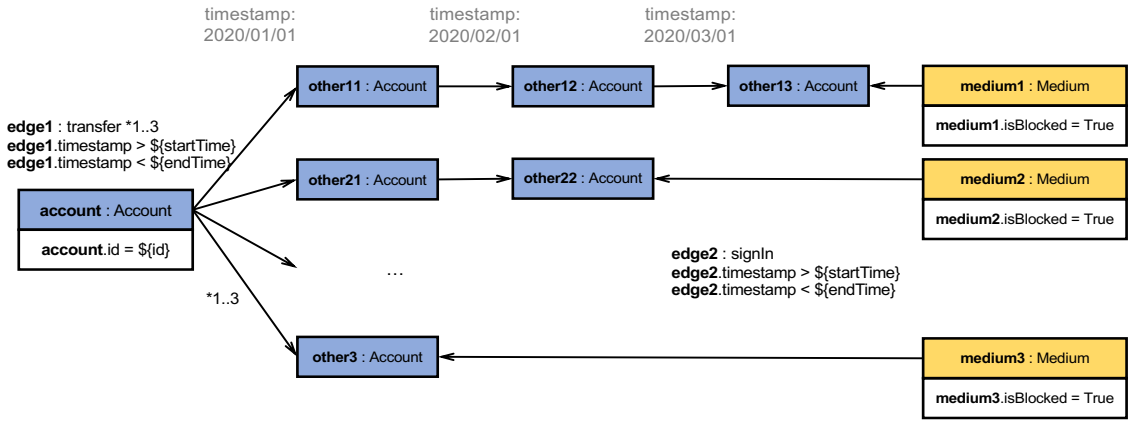


Figure 7: The pattern of complex read query 1 (TCR 1) in transaction workload.

Query Explanation:

TCR 1 matches transfer paths in ascending temporal order, tracing the divergence/downstream of a fund flow.

Case Study: Read-Write Query

- **Query Explanation:** Query usually checks graph pattern indicating illegal activities before write operation execution.
- **Feature**: Query is a complex read (the risk strategies) wrapped in transactions
- **Choke point**: Query benefits from optimization on write operation contention and conflicts.

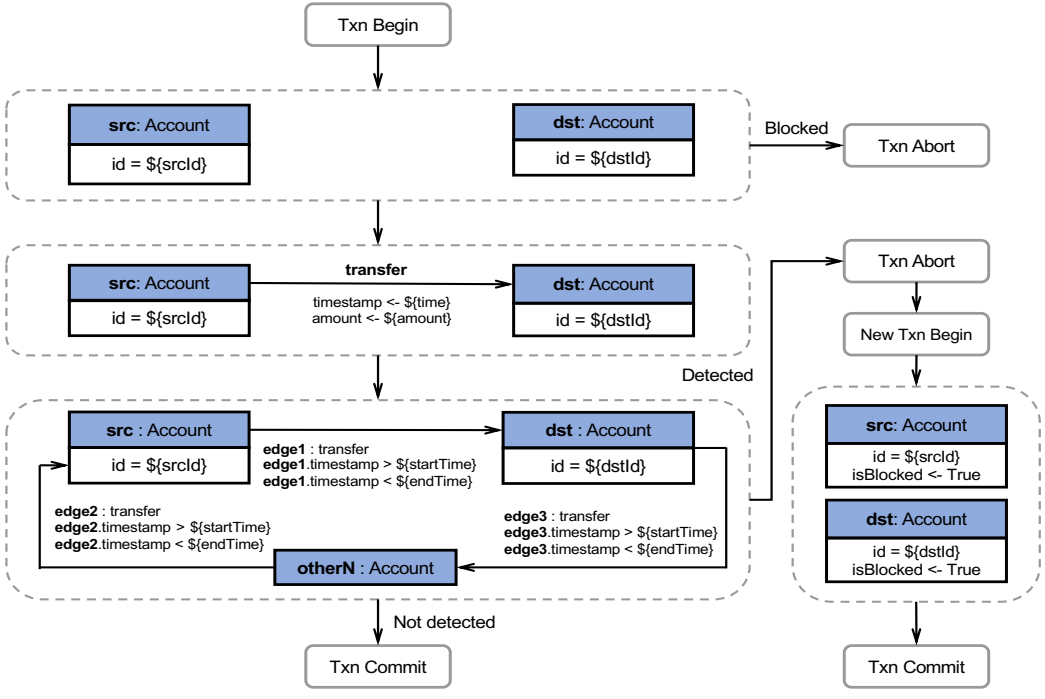


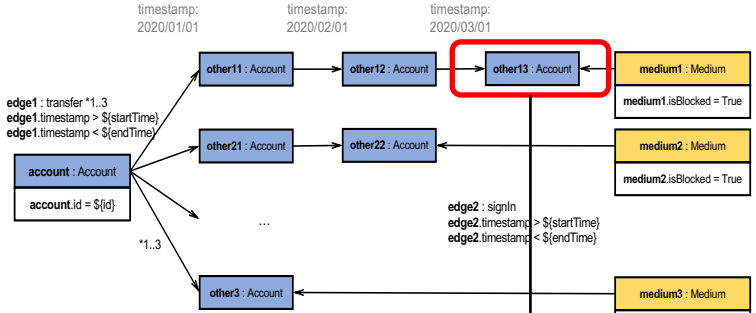
Figure 8: The pattern of read-write query 1 (TRW 1) in transaction workload.

Workload Orchestration

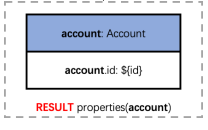
Transaction Workload includes 12 Complex Reads, 6 Simple Reads, 17 Writes, 3 Read-Writes

- Write queries and read-write queries are incremental data.
- Read queries are mixed as random walks on the graph.

Complex Read 1



Simple Read 1
find the properties of exact account



Time-biased random walk: $Prob(x) = 1 - U^{k \cdot t}$

- U is a random variable uniformly distributed in the interval [0, 1).
- t is the time window length for the complex read query.
- k is a coefficient that adjusts the time window

Longer temporal window in Complex Read Queries indicates more Simple Read Queries following it.

Chokepoint Analysis Experiments

Experiments are conducted on TuGraph-DB (open-source) to demonstrate choke point analysis

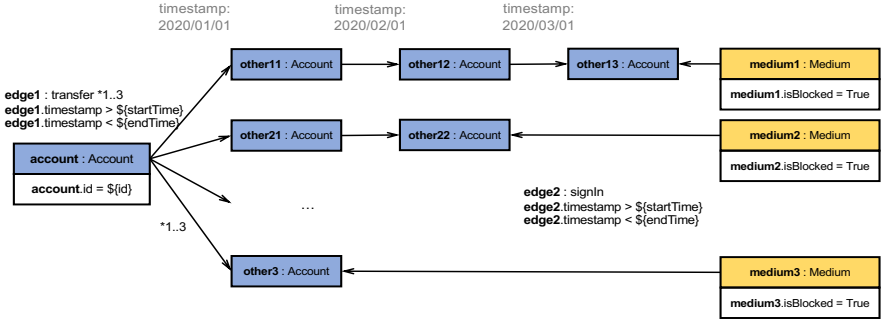
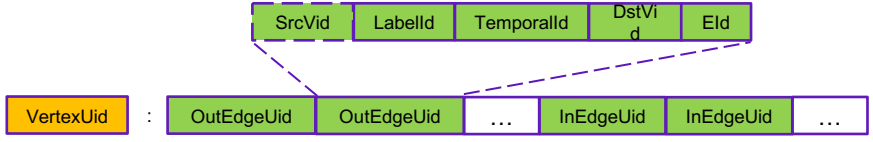


Figure 7: The pattern of complex read query 1 (TCR 1)

```
Produce Results
  Distinct
    Expand(All) [other <-- medium EdgeFilter e2]
      VarLen Expand(All) [acc -->*1..3 other EdgeFilter isAsc]
        Node Index Seek [acc]  IN []
```

Simplified logical plan of TCR 1



Experiment 1: Achieved ~41% query execution speedup on TCR 1 with temporal edge storage

Complex Read Query	Push-down Speedup
TCR 1	70.3%
TCR 2	87.4%
TCR 5	83.3%

Experiment 2: Achieved at least 70% query execution speedups with path filters pushed down

Note: Please refer to the paper for the end2end experiments

Takeaways & Future Work

- Profiled financial graphs and found features including timestamp properties, edge multiplicity and unbounded skewness.
- Provided a scalable (currently SF100) and configurable data generator conforming to the data features to simulate financial graphs.
- Identified several choke points embedded in queries and designed a Transaction Workload with 4 query categories.
- Provided a benchmark driver and reference implementations for demonstration and validation.
- Conducted choke point analysis experiments for further system optimization.

Future plan: Larger dataset scales and a new workload for OLAP (like SNB BI) in financial scenarios

Thanks!

- ldbc/ldbc_finbench_docs
- ldbc/ldbc_finbench_datagen
- ldbc/ldbc_finbench_driver
- ldbc/ldbc_finbench_transaction_impls