FinBench: The new LDBC benchmark targeting financial scenario

Shipeng Qi

(with contributions from members of the FinBench Task Force)

Benchmark Overview

FinBench Motivation

- **SNB**, Social Network Benchmark, is designed based on social network scenarios, which is limited when applied to the financial service industry.

- **FinBench** objective is to design a high-quality benchmark for evaluating the performance of graph database systems in financial scenarios, e.g. anti-fraud and risk control, based on financial data patterns and query patterns.

Key Features in FinBench

- Dataset
  - PowerLaw distribution
  - Multiplicity
  - Hub Vertex
- Transaction Workload
  - Read-write query
  - Special graph patterns
  - Time-window filtering
  - Recursive path filtering
  - Truncation

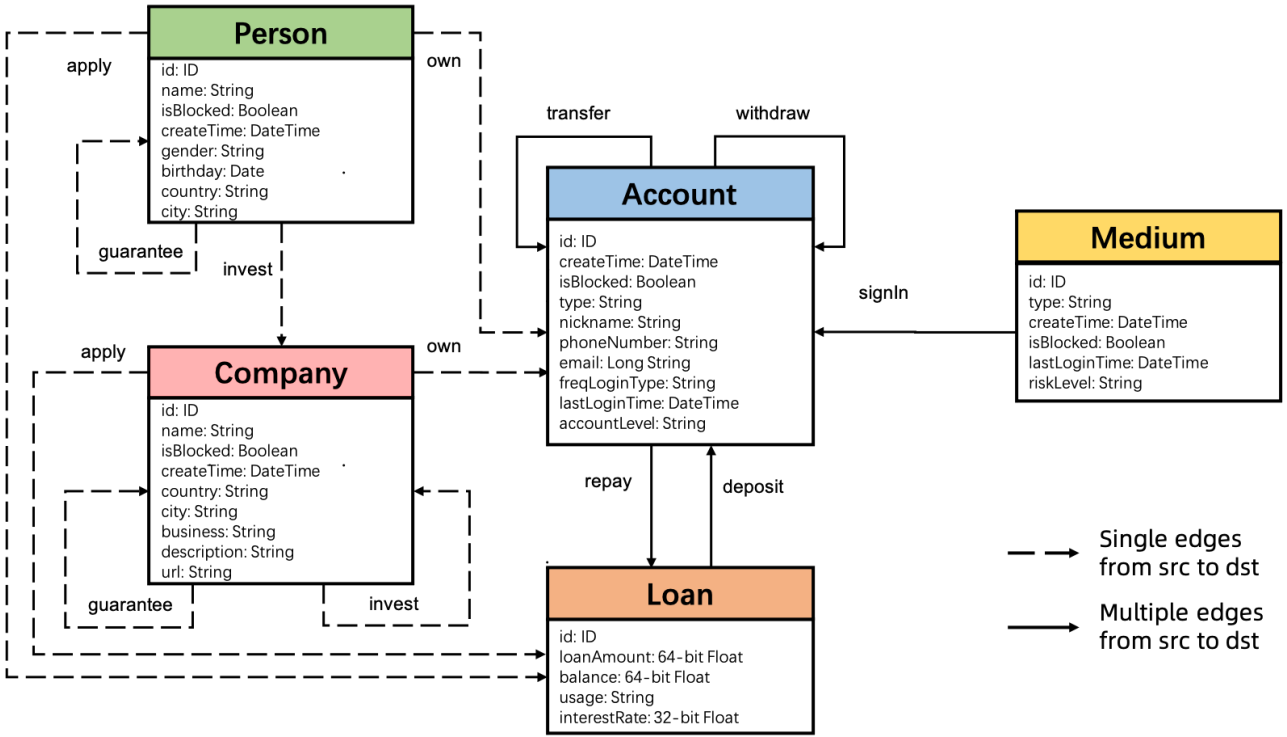
Brief of the initial version

- Standard Design: all key features in proposal implemented
- Workload: Transaction Workload, including 12 complex read queries, 6 simple read queries, 19 write queries and 3 read-write queries
- Dataset: Up to SF10 scale supported
- Implementation on 3 systems: TuGraph, Galaxybase, and UltipaGraph
- Collaboration: 9 vendors in Task Force and 6 developers

Data Design and Generated Datasets

- Data Schema
- Data Distribution
- Datasets Statistics

Data Schema


Person
id: ID
name: String
isBlocked: Boolean
createTime: DateTime
gender: String
birthday: Date
country: String
city: String
apply
own
guarantee
invest
Company
id: ID
name: String
isBlocked: Boolean
createTime: DateTime
country: String
city: String
business: String
description: String
url: String
apply
own
guarantee
invest
Account
id: ID
createTime: DateTime
isBlocked: Boolean
type: String
nickname: String
phoneNumber: String
email: Long String
freqLoginType: String
lastLoginTime: DateTime
accountLevel: String
transfer
withdraw
signIn
Medium
id: ID
type: String
createTime: DateTime
isBlocked: Boolean
lastLoginTime: DateTime
riskLevel: String
repay
deposit
Loan
id: ID
loanAmount: 64-bit Float
balance: 64-bit Float
usage: String
interestRate: 32-bit Float
Single edges from src to dst
Multiple edges from src to dst

Data Distribution: Transfer Edge

- Degree: PowerLaw Distribution
- Asymmetric directed graph
- Hub vertex: degree increases with scale
  - MaxDegree = 1000 in SF1
  - MaxDegree = 10000 in SF10
  - Larger scale to be supported

```
+------------------+---------+
|              toId|in_degree|
+------------------+---------+
|4891190670301082260|      945|
|4897383119788711667|      567|
| 286260051314745075|      567|
|  99079191802151398|      543|
|4868391197187506662|      543|
|4907234743973581309|      510|
| 296393150476325373|      510|
|4908642118857140591|      384|
|4865576447420410431|      360|
|4911456868624245691|      300|
+------------------+---------+
only showing top 10 rows
```

```
+------------------+------------------+------------+
|            fromId|              toId|multiplicity|
+------------------+------------------+------------+
|4837428949749347364|4891190670301082260|          67|
| 165788761282584041| 286260051314745075|          53|
| 183521684815353485| 240942580064328271|          51|
|4752986456736143480|4844747299143816836|          43|
|4902731144346222798|4821666351053553660|          40|
|4761993655990886968|4878524296349098175|          33|
|4902731144346222798|4778882154593534224|          31|
|4863043172630020163|4896538694858587751|          29|
| 258394028620386533| 218143106950763621|          29|
| 297800525359880817| 286260051314745075|          28|
+------------------+------------------+------------+
only showing top 10 rows
```

```
Num of accounts: 26347
Num of transfer edges: 138209
Average Degree: 5.245720575397579
Average Multiplicity: 1.616574068658986
```

Profiling of SF0.1

Transaction Workload

- Transaction Workload
- Time Window Filtering
- Recursive Path Filtering
- Read-Write Query
- Truncation
- Query Mix
- Transaction Workload Driver

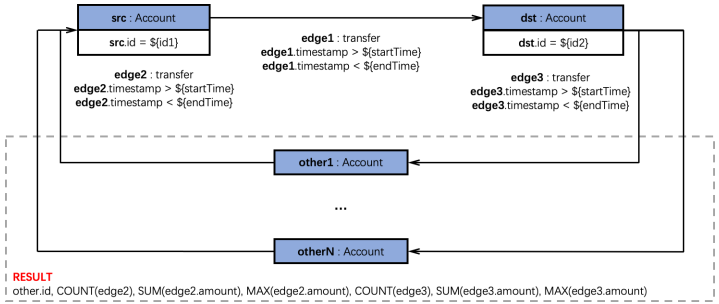
Transaction Workload

Scenario: financial activities among accounts, persons, companies, loans and media

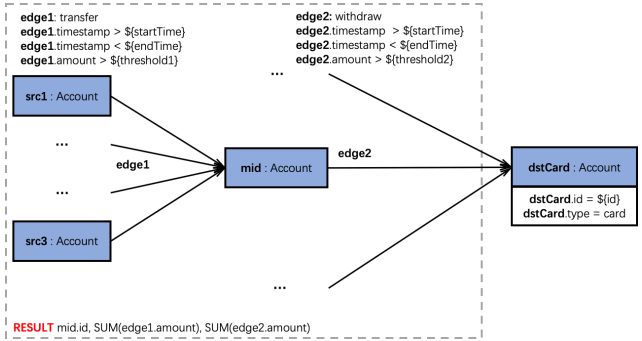
Queries:

- 12 complex reads: match exact patterns including cycles and trees(see next slide) starting from one or two vertices
- 6 simple reads: discover the neighbourhood of an Account node
- 19 write queries: inserts, updates, deletes(cascade deletion)
- 3 read-write queries: transaction-wrapped complex reads

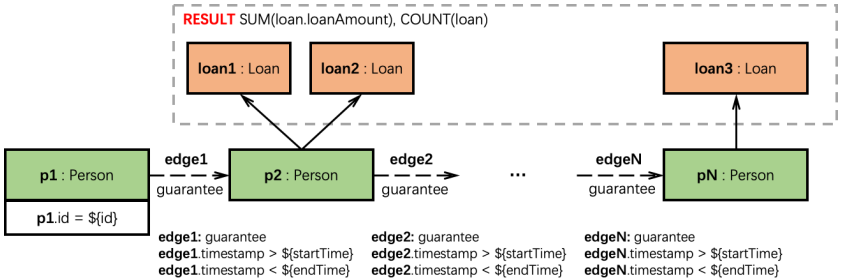
Transaction Workload: Example Patterns



Cycle
[Ref: Transaction Complex Read 4]



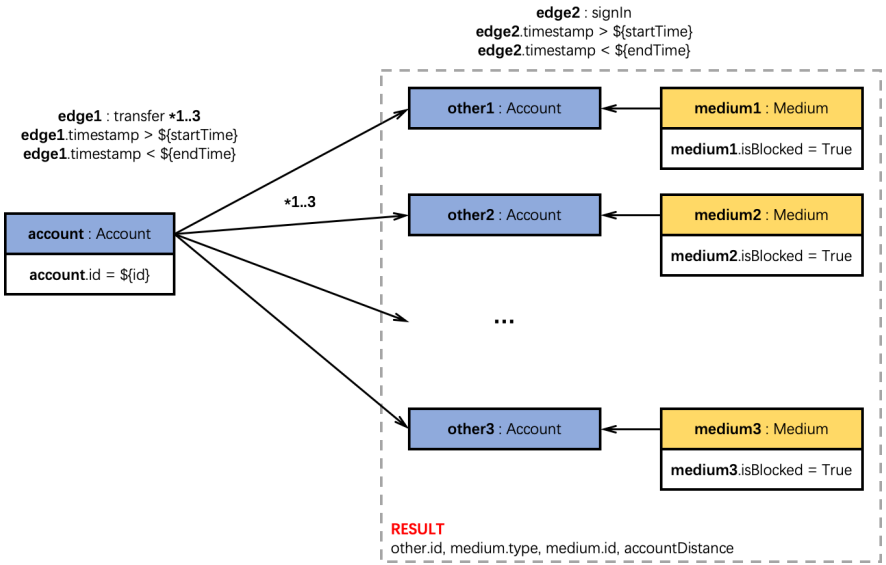
Tree
[Ref: Transaction Complex Read 6]



chain
[Ref: Transaction Complex Read 11]

Time Window Filtering

- Fact: queries only look back in a limited time window
- Filtering: filter edges between ***startTime*** and ***endTime*** in traversal

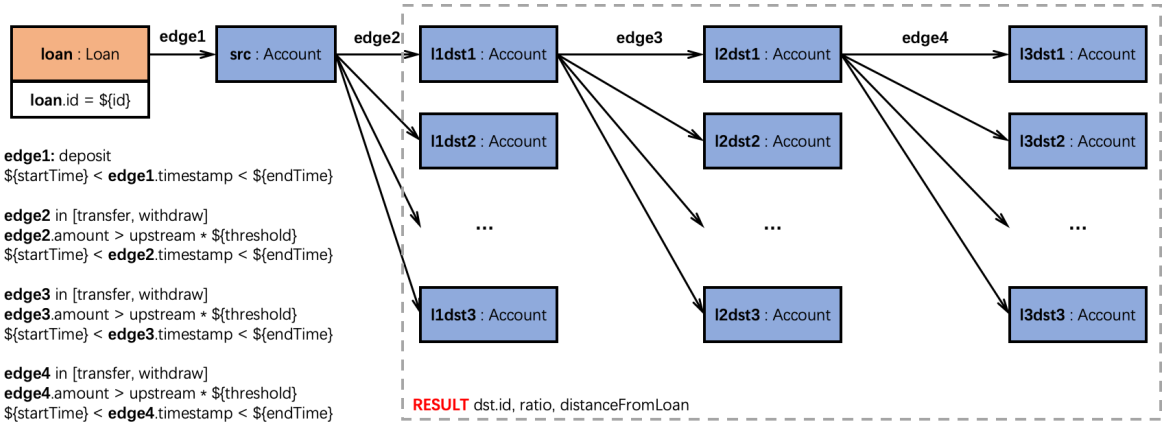


Blocked medium related accounts
[Ref: Transaction Complex Read 1]

Recursive Path Filtering

Assuming: A -[e1]-> B -[e2]-> ... -> X

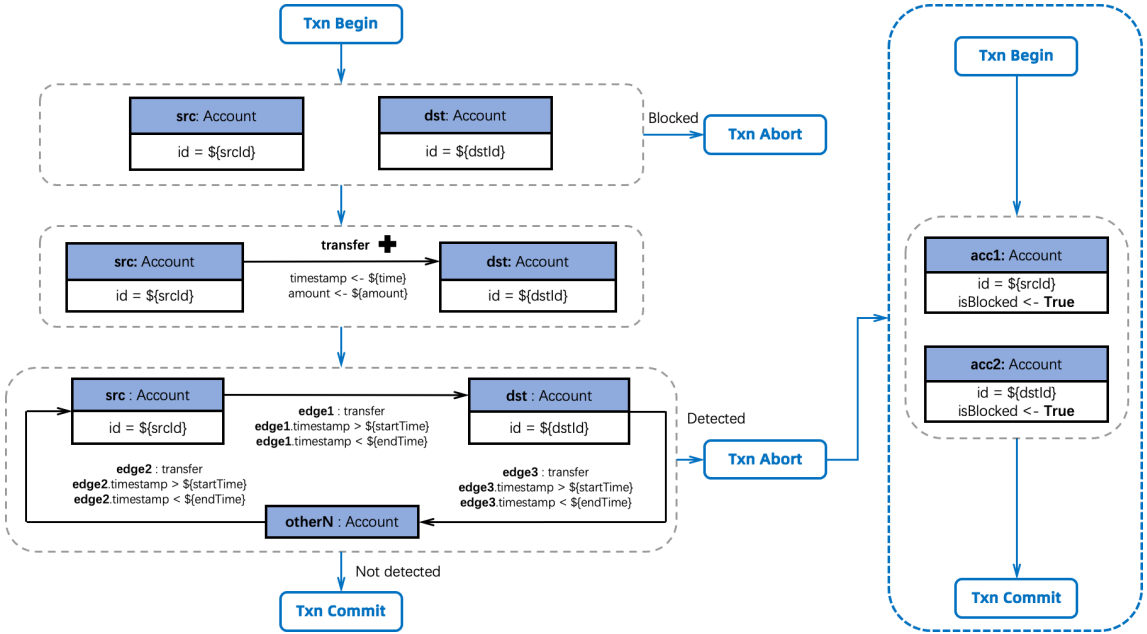
- Timestamp order: e1 < … < ei
- Amount order: e1 > … > ei



Transfer trace after loan applied
[Ref: Transaction Complex Read 8]

Read-Write Query

- Transaction-wrapped complex reads (risk control stategy)
- If the complex read matches, commit the transaction with write query. Otherwise, transaction abort

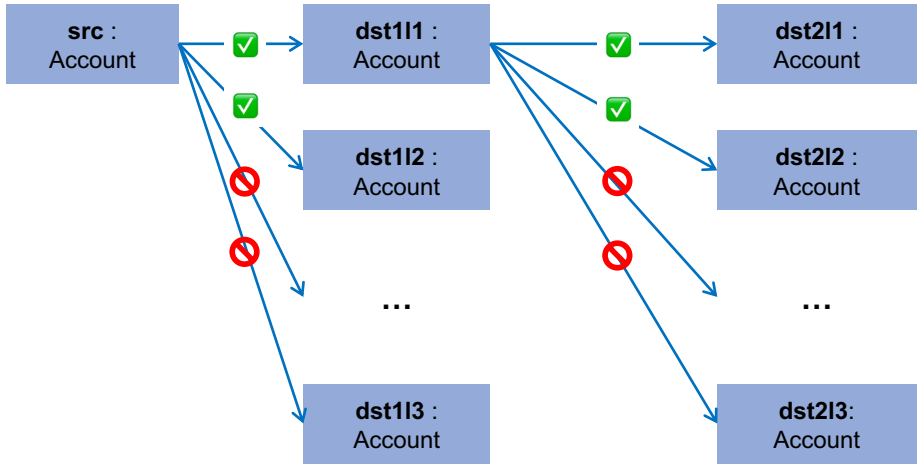


Transfer under transfer cycle detection strategy
[Ref: Transaction Read Write 3]

Truncation

- Truncate less-important edges to avoid complexity explosion when traversing
- Truncating is actually sampling
- TruncationLimit and truncationOrder is defined to ensure consistency of results.

For example, keep only the top 100 edges in order of timestamp descending



Benchmark Suite

Datasets Statistics

Supported Scale Factor	|V|	|E|
0.01	8663	61674
0.1	64485	610658
0.3	192971	1830891
1	643241	6091820
3	1928439	18243343
10	6069955	51889416

FinBench datasets of SF0.01 to SF10 are published at the Google Drive. These datasets were all generated using csv serializers in the initial version.

*Note: please see the tables in **Appendix A** for detailed statistics*

Transaction Workload Driver

Inherited from SNB Interactive driver, the driver has 3 modes of operation, all starting with a database containing the initial data set.

1. Generate validation data set

- single-threaded, sequential execution
- output: validation results

2. Validate implementation

- single-threaded, sequential execution
- input: validation results
- output:
  - passed/failed validation
  - if failed: expected vs. actual results

3. Execute benchmark

- multi-threaded, concurrent execution
- Use TCR to control the load scale
- output:
  - passed/failed schedule audit
  - throughput (operations per second)
  - per-query performance results

Query Mix

Inherited from SNB design:

- **Write queries and read-write queries:** operations issue times generated by the data generator
- **Complex read queries:** complex reads times are expressed in terms of update operations (update frequencies)
- **Simple read queries:** a sequence of short reads follows each complex read instance

Implementations and Standard-establishing Audits

Implementations and Standard-establishing Audits

system	data model	language
TuGraph™	graph	Cypher
GALAXYBASE	graph	Cypher
ULTIPA	graph	UQL

- Packages and Reports available at https://drive.google.com/drive/folders/1OQXrz2CkQke7SE9KWBiMeEn0KYx-QCOl
- All systems passed cross-validation

Roadmap and acknowledgement

Roadmap

Version	Estimated Time	Features
✅ 0.1.0	Mid of 2023	• Runnable and auditable
0.2.0	End of 2023	• Larger scale data generation • Optimize parameter curation • Query mix profiling and design
0.3.0	2024	• New workload: Analytics workload

Acknowledgement

Task Force Members

Developers

Name	Affiliation
Shipeng Qi	Ant Group
Bing Tong	CreateLink
Changyuan Wang	Vesoft
Yang Bin	Ultipa
Shenghao Zhang	StarGraph


LDBC
The graph & RDF
benchmark reference

Appendix

Work Chart: Goals of FinBench

Intended output

- The intended output is LDBC FinBench, a precise specification for evaluating graph database query and computation performance based on financial scenarios. It is capable of independent implementations using various graph database products, intended for approval as one of LDBC Standards. https://github.com/ldbc/ldbc_finbench_docs

Work product

- Software for data generation：https://github.com/ldbc/ldbc_finbench_datagen
- Software for query driver：https://github.com/ldbc/ldbc_finbench_driver
- Reference implementation：https://github.com/ldbc/ldbc_finbench_transaction_impls

Resources

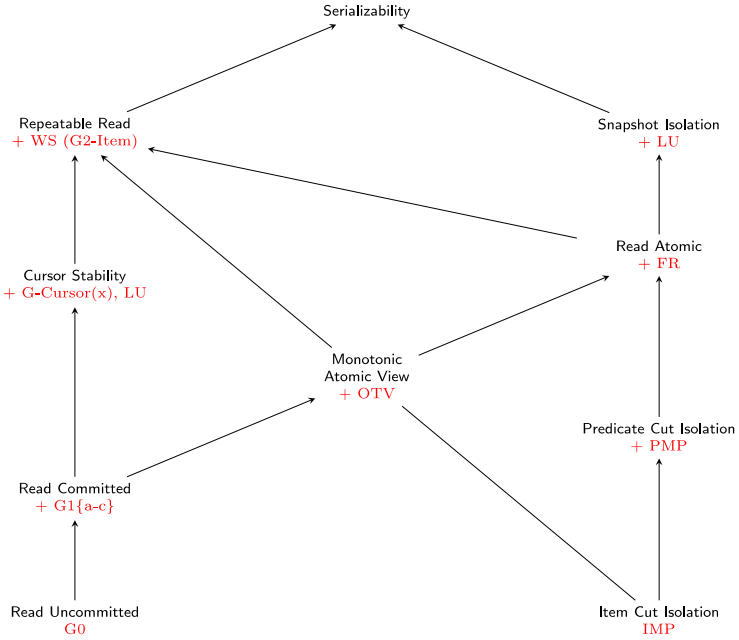
- Specification: https://github.com/ldbc/ldbc_finbench_docs
- Benchmark Suite
  - https://github.com/ldbc/ldbc_finbench_driver
  - https://github.com/ldbc/ldbc_finbench_datagen
  - https://github.com/ldbc/ldbc_finbench_transaction_impls
  - https://github.com/ldbc/ldbc_finbench_acid
- Datasets: https://drive.google.com/drive/folders/1tURBIJE56ZNC9YvMtug31peYD5csizCa?usp=sharing
- Certification audit packages: https://drive.google.com/drive/folders/1OQXrz2CkQke7SE9KWBiMeEn0KYx-QCOl?usp=sharing

Dataset statistics

V/E	Entity	SF0.01	SF0.1	SF0.3	SF1	SF3	SF10
V	account	2633	26347	79199	264075	791769	1980883
V	company	2633	4000	12000	40000	120000	300000
E	companyApplyLoan	524	5332	15761	52820	158678	397060
E	companyGuarantee	248	2315	7123	23870	71716	179526
E	companyInvest	860	8639	25853	86092	259884	650190
E	companyOwnAccount	864	8805	26356	88119	264352	660625
E	deposit	5199	51686	153521	512680	1534595	3829905
V	loan	1597	16138	47772	159166	476670	1189072
E	loanTransfer	4886	49180	145679	484657	1453874	3625556
V	medium	1000	10000	30000	100000	300000	2000000
V	person	800	8000	24000	80000	240000	600000
E	personApplyLoan	1073	10806	32011	106346	317992	792012
E	personGuarantee	469	4694	14221	47935	144064	359283
E	personInvest	1650	17296	52002	174064	520584	1300980
E	personOwnAccount	1769	17542	52843	175956	527417	1320258
E	repay	5046	50495	149559	497033	1488916	3715487
E	signIn	4384	44540	134532	451362	1350759	8996781
E	transfer	14145	138209	411882	1379527	4136803	11005032
E	withdraw	20557	201119	609548	2011359	6013709	15056721

ACID Test Suite

- Based on the “ACID Test” work in LDBC SNB
- Atomicity and Isolation Test: Based on failing cases
- Consistency and Durability Test
  - Execute the benchmark workload for duration T
  - Inject failure(e.g. a power failure, software crash, reboot, etc) into tested system
  - After the restart of system, check if all the last committed data survive
  - Check if all the constraints (uniquness, precomputed properties, indices) are not violated



Atomicity and Isolation Test

Auditing rules

Audit workflow:

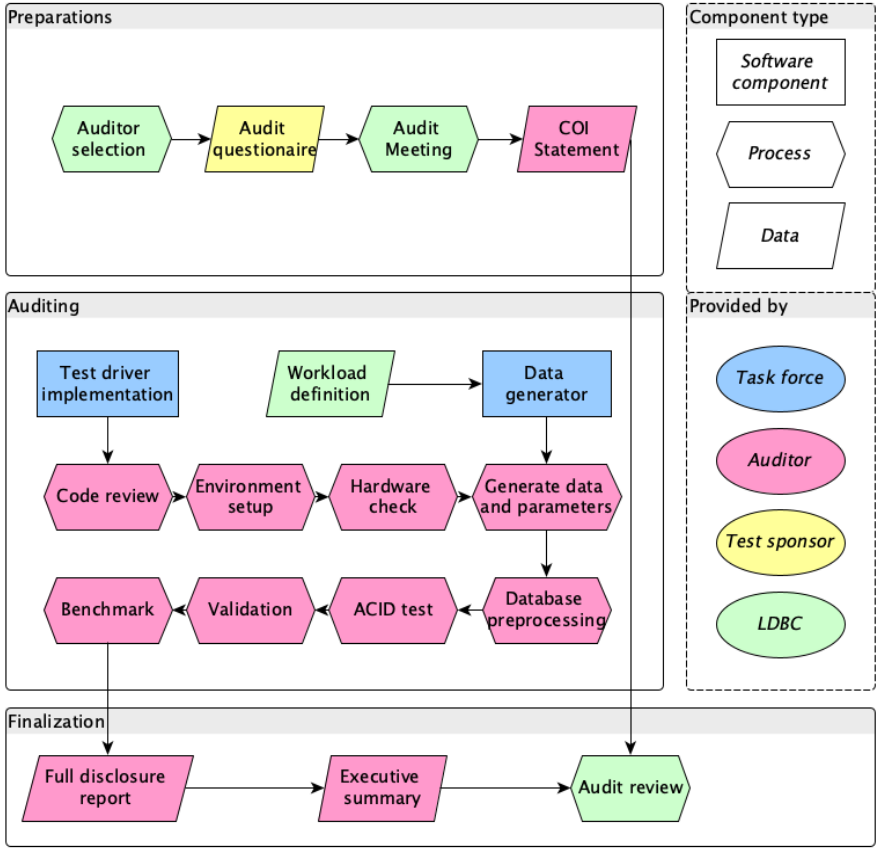
- Start from ACID to find problems earlier
- Contract -> Audit -> Review -> Publish

Audited benchmark results:

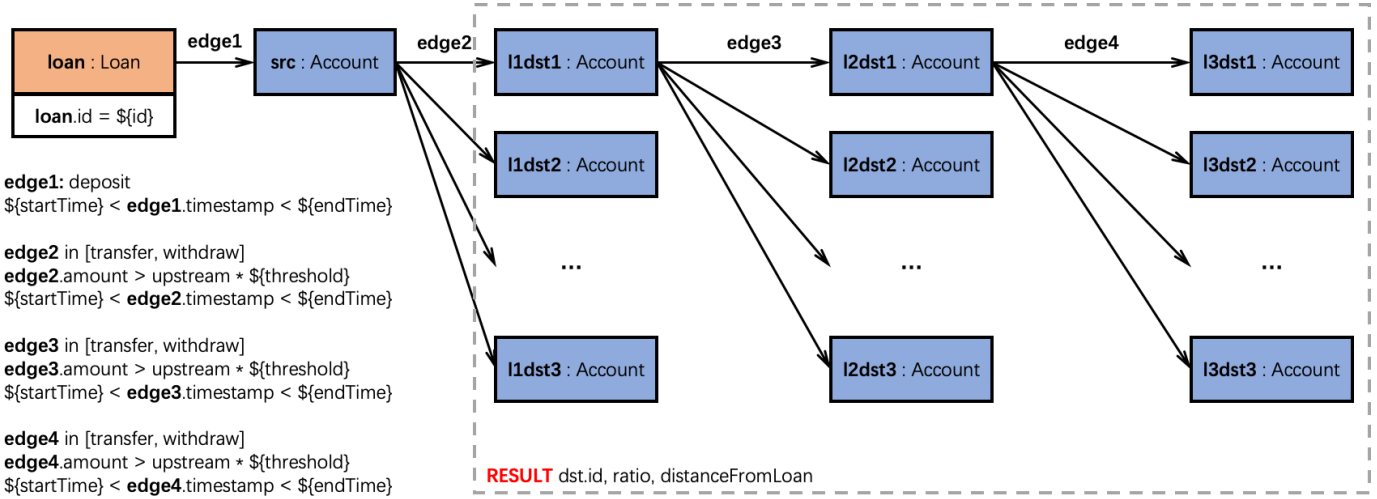
- Produced by an independent auditor
- Reviewed by Task Force Lead and LDBC
- Published as “LDBC benchmark results”

Auditor selection:

- Independent with no conflict of interest
- Provide COI if needed considering auditors are from vendors



New Chokepoint Example #1



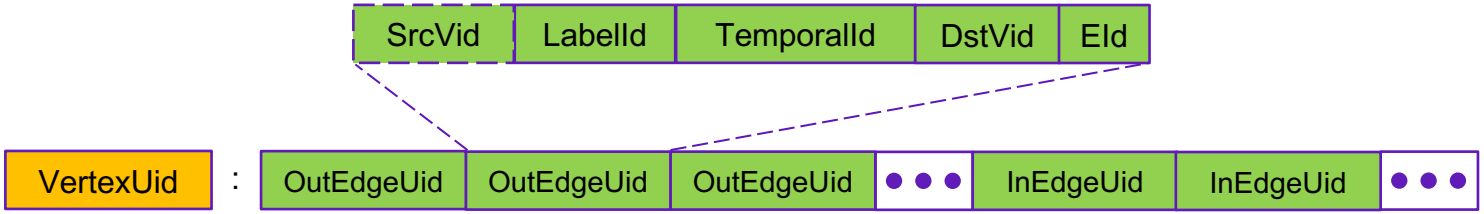
Assuming: A -[e1]-> B -[e2]-> ... -> X

- Timestamp order: e1 < e2 < … < ei
- Amount order: e1 > e2 > … > ei
- Time window: ei-1 < ei < ei-1 + △

[LANG] Language Features: Recursive path filtering pattern

More flexible expression is wished to support this filtering pattern.

New Chokepoint Example #2



[STORAGE] Data Access Locality: Temporal access locality and performance

Boost the time-window filtering with well-sorted data in storage layer