The LDBC Financial Benchmark

Shipeng Qi

(with contributions from members of the FinBench Task Force)

Motivation of FinBench

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

- **FinBench,** following LDBC's chokepoint-driven benchmark design philosophy, is to design a benchmark for evaluating graph database systems in **financial scenarios with new chokepoints embedded**, based on **financial data patterns and workload patterns**.

Key Features in FinBench

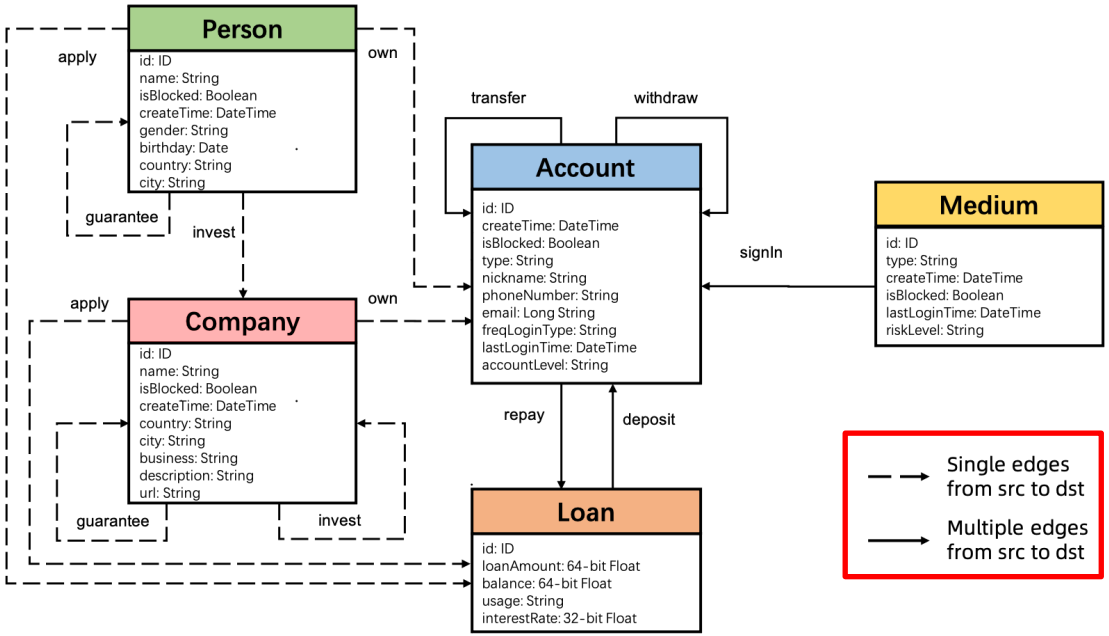
Dataset Patterns

- Power-law degree distribution
- Edge multiplicity
- Asymmetric dynamic temporal graph
- Hub vertex

Workload Patterns

- Read-Write query
- Time-window filtering
- Recursive path filtering
- Patterns in temporal graph
- Truncation
- Time-biased query mix

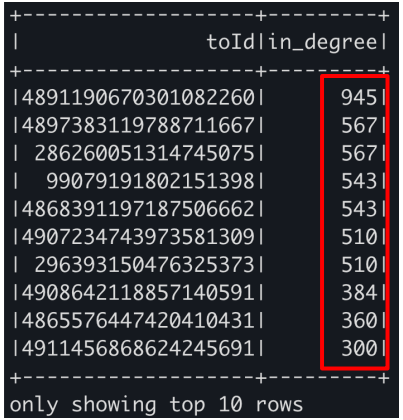
Dataset: Schema



- Vertices are entities in financial systems, while edges are activities involving them
- Asymmetric dynamic temporal graph

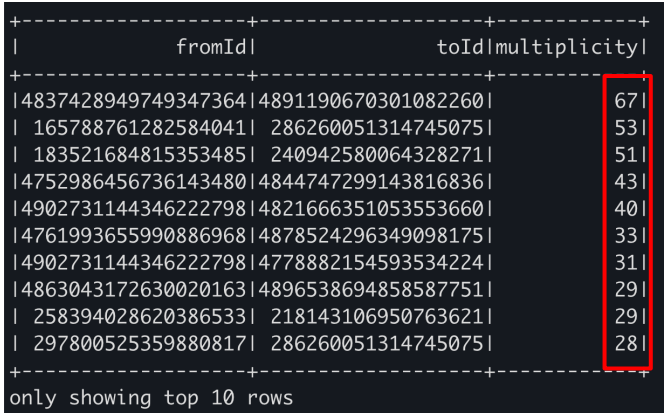
Dataset: Distribution

- Transfer edge: Power-law distribution
- Hub vertex: degree increases with scale
  - MaxDegree = 1k in SF1
  - MaxDegree = 10k in SF10
  - MaxDegree = 100k in SF100
  - …



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

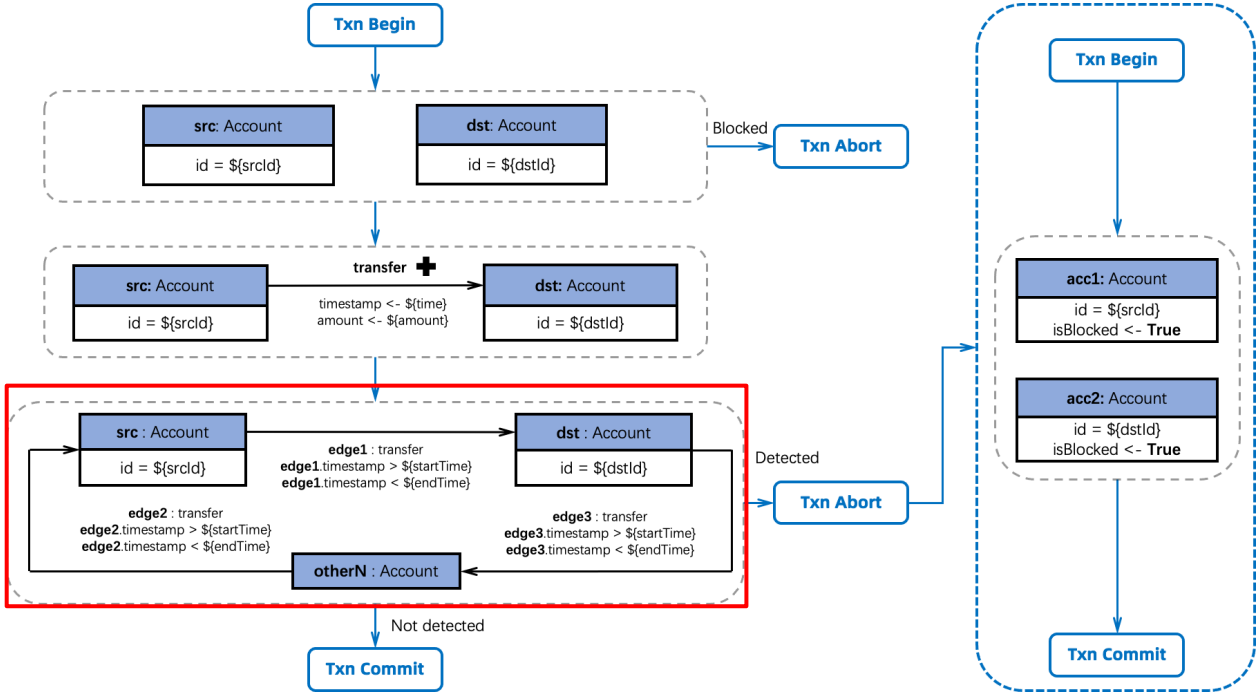
only showing top 10 rows

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

Degree and multiplicity in SF0.1 dataset of v0.1

Workload: Read-Write Query

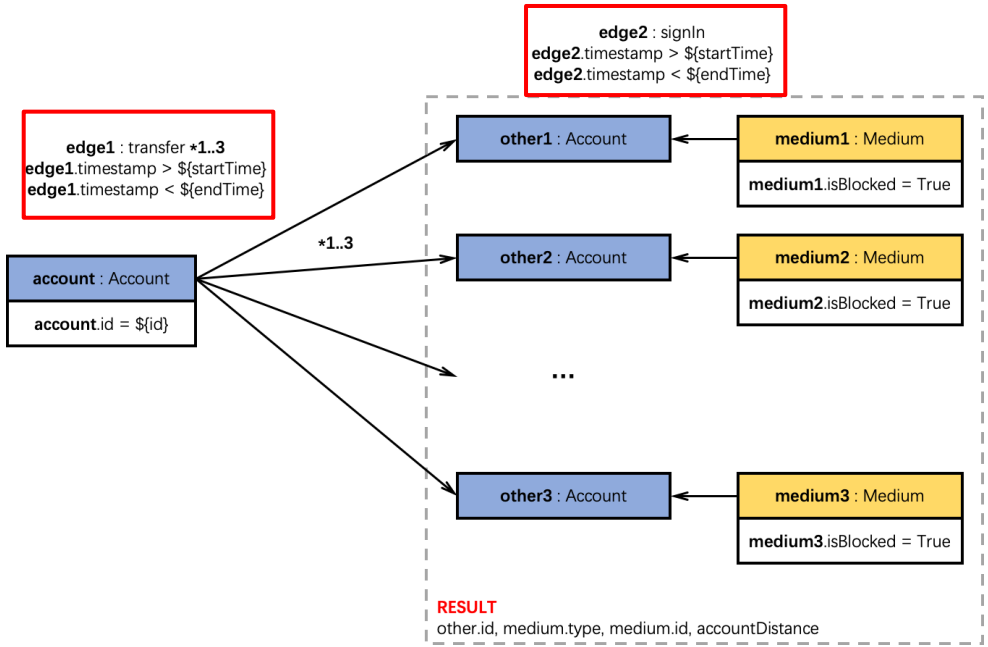
- Complex read query: it represents a risk control strategy
- RW query: Transaction-wrapped complex reads (risk control strategy)
- If the risk control strategy is not hit, transaction commits with write query. Otherwise, transaction aborts



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

Workload: Time Window Filtering

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

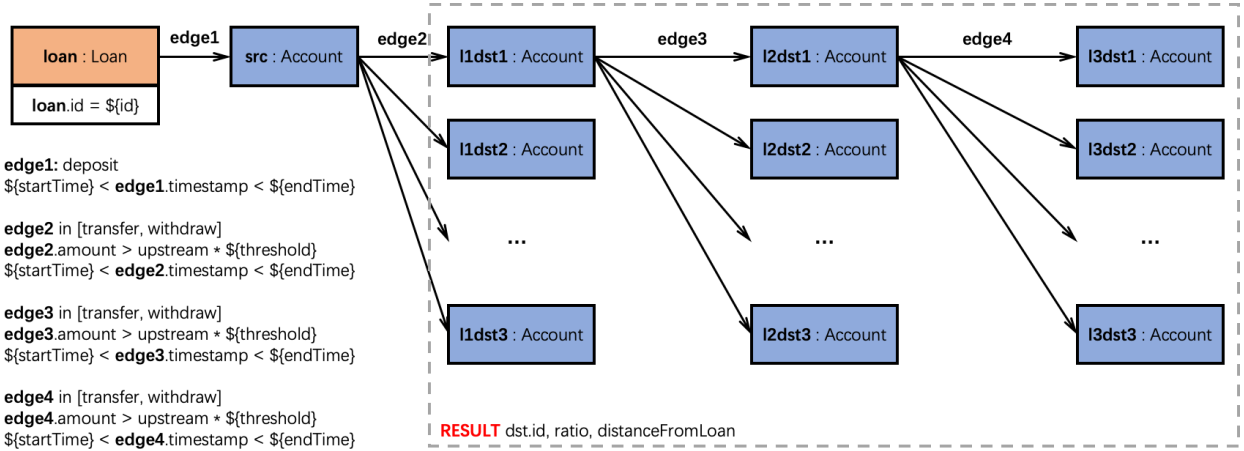


Blocked medium related accounts
[Ref: Transaction Complex Read 1]

Workload: Recursive Path Filtering

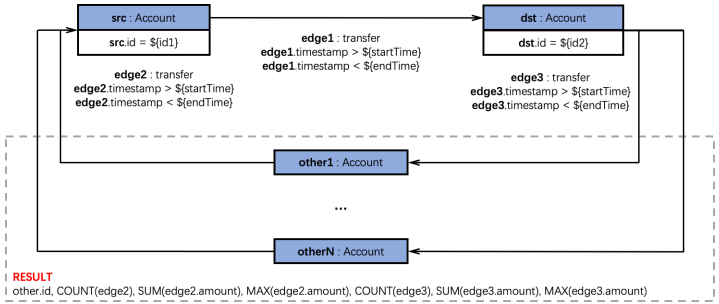
Assuming: A -[e1]-> B -[e2]-> ... -> X

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

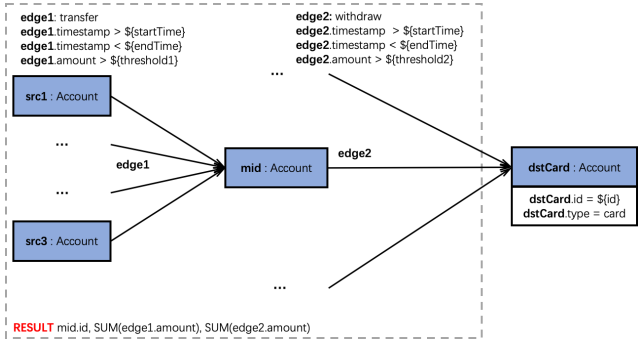


Transfer trace after loan applied
[Ref: Transaction Complex Read 8]

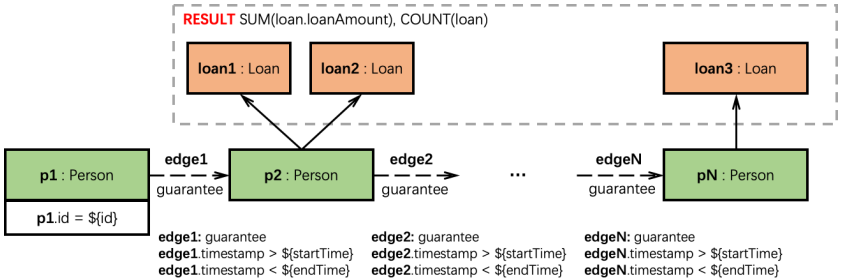
Workload: Patterns in temporal graph



Cycle
[Ref: Transaction Complex Read 4]



Tree
[Ref: Transaction Complex Read 6]

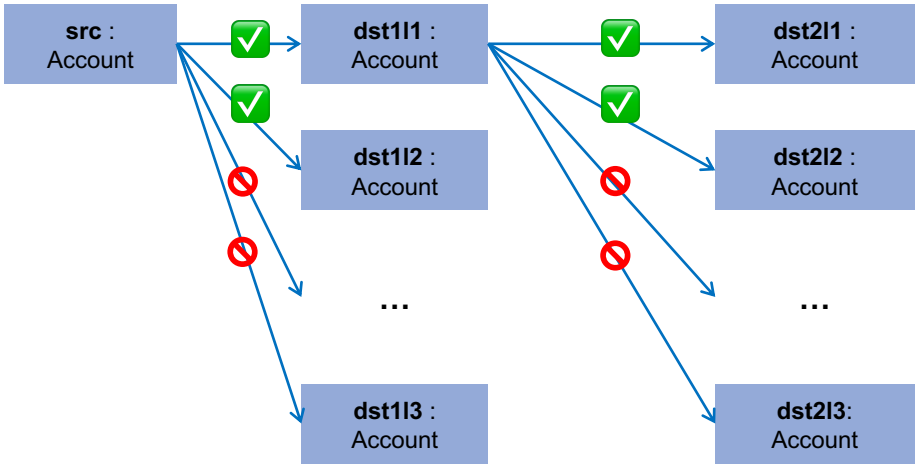


chain
[Ref: Transaction Complex Read 11]

Workload: Truncation

- In practice, system optimization cannot keep up with the increase of the workload complexity
- Truncate less-important edges to avoid complexity explosion, which 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



Workload: Time-biased query mix

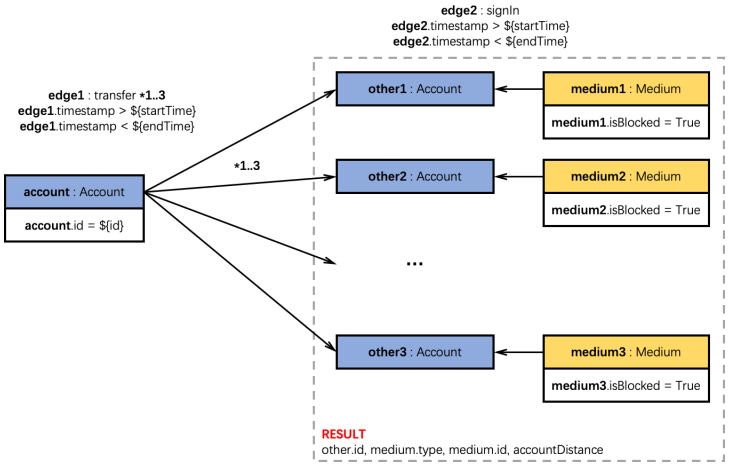
Inherited from SNB' query mix:

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

Time-biased query mix:

- **Fact**: complex read – risk control strategy, simple read – simple checks
- A time-biased function is designed that a complex read of longer time window is followed by more simple reads

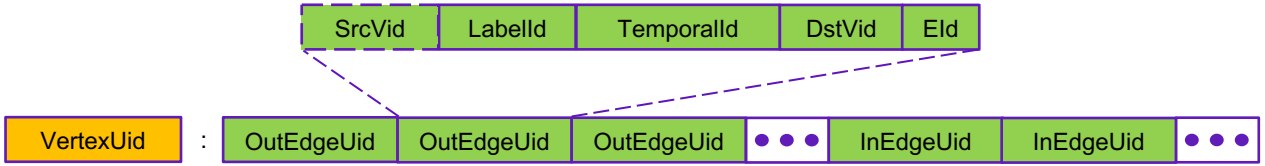
New Chokepoint #1 in Storage



[STORAGE] Temporal access locality and performance

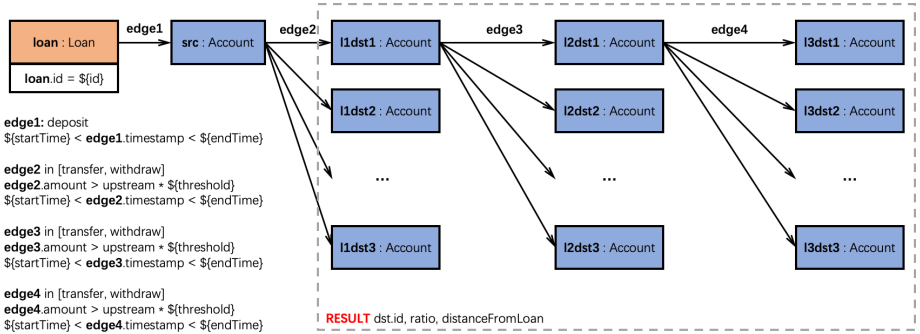
Queries access the edges in specific time windows

TuGraph™



Boost the time-window filtering due to edges well-sorted in storage

New Chokepoint #2 in Language



Possible solution (non-official):

- Outside the brackets: vertex to start, edge types, hopping from 1 to $maxhops
- Clause inside brackets constrains the path
- SLIDING subclause means a window sliding on the path of LENGTH and STEP
- WHERE subclause defines the filter
- UNTIL subclause defines the termination condition

Acknowledgement: Prof. Yin and his team at SJTU

[LANG] Language Features: Recursive path filtering pattern

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

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

```
MATCH
(src:card_label WHERE src.id=$cardId)
-[e:transfer|withdraw where e.time < $time+$range]->{1,$maxhops}
(dst)
[
    SLIDING
        LENGTH 2
        STEP 1
    AS p
    WHERE p[1].e.gmt_occur - p[0].e.gmt_occur < $delta
    UNTIL dst.type = people
]
return path
```

Another example in language chokepoint

```
MatchStatement = MATCH [Truncation] Match {',' Match} [WhereClause] [Statement] .
Truncation = TRUNCATING TruncationSpec {',' TruncationSpec} .
TruncationSpec = [EdgeType_id] ['(' OrderSpec {',' OrderSpec} ')'] '=' int .
Match = (MatchMode [id '='] MatchNode) {'|' Match} .
MatchNode = '(' MatchItem ')' {(MatchEdge|MatchPath) MatchNode}.
MatchEdge = '-[' MatchItem '->' | '<-' MatchItem ']-' .
MatchItem = [id | Node_Value] [GraphLabel] [ Document | Where ] .
MatchPath = '[' Match ']' MatchQuantifier .
MatchQuantifier = '?' | '*' | '+' | '{' int , [int] '}' .
MatchMode = [TRAIL|ACYCLIC| SIMPLE] [SHORTEST |ALL|ANY] .
```

Implementing the draft Graph Query Language Standard

The Financial Benchmark

Malcolm Crowe
Emeritus Professor
University of the West of Scotland
United Kingdom
e-mail: malcolm.crowe@uws.ac.uk

Fritz Laux
Emeritus Professor
Reutlingen University
Germany
e-mail: fritz.laux@reutlingen-university.de

Arxiv doi: 2407.09566

Acknowledgement: Prof. Crowe

Malcom Crowe designed a syntax to expand the definition and usage of **Truncation,** which is also implemented in his PyrrhoDBMS, according to his short paper at DBKDA 2024.

We are working on the cross-validation to accept it as an official implementation.

Version plan

Version 0.1.0

- All key features in proposal implemented
- Dataset: Up to SF10 scale supported
- Workload: Transaction Workload, including 12 complex read queries, 6 simple read queries, 19 write queries and 3 read-write queries

Version 0.2.0 (TBA in short)

- Benchmark suite: parameter curation optimization
- Dataset: support SF30 and SF100, SF 300 WIP
- Paper: WIP

More future work： automated benchmarking, analytic workload, etc…

Thanks!

Acknowledgement

Developers: Shipeng Qi, Bing Tong, Jiatao Hu, Bin Yang, Changyuan Wang

Collaboration: Tao Lv@CSTC, Prof. Lei Zou@PKU， Prof. Malcome Crowe, Prof. Qiang Yin@SJTU

Welcome collaboration on benchmark and research on chokepoints

Contact me at **shipeng.qi AT ldbcouncil.org**

WeChat QR Code


LDBC
The graph & RDF
benchmark reference