# Implementing the draft Graph Query Language Standard: The Financial Benchmark

MALCOLM CROWE, FRITZ LAUX

DBKDA 2024



# Malcolm Crowe

University of the West of Scotland
Email: malcolm.crowe@uws.ac.uk

- Malcolm Crowe is an Emeritus Professor at the University of the West of Scotland, where he worked from 1972 (when it was Paisley College of Technology) until 2018.
- He gained a D.Phil. in Mathematics at the University of Oxford in 1979.
- He was appointed head of the Department of Computing in 1985. His funded research projects before 2001 were on Programming Languages and Cooperative Work.
- Since 2001 he has worked steadily on PyrrhoDBMS to explore optimistic technologies for relational databases and this work led to involvement in DBTech, and a series of papers and other contributions at IARIA conferences with Fritz Laux, Martti Laiho, and others.
- Prof. Crowe has recently been appointed an IARIA Fellow.

# Prof. Dr. Fritz Laux

(Retired), Reutlingen University
Email: fritz.laux@reutlingen-university.de

- Prof. Dr. Fritz Laux was professor (now emeritus) for Database and Information Systems at Reutlingen University from 1986 - 2015. He holds an MSc (Diplom) and PhD (Dr. rer. nat.) in Mathematics.
- His current research interests include
  - Information modeling and data integration
  - Transaction management and optimistic concurrency control
  - Business intelligence and knowledge discovery
- He contributed papers to DBKDA and PATTERNS conferences that received DBKDA 2009 and DBKDA 2010 Best Paper Awards. He is a panellist, keynote speaker, and member of the DBKDA advisory board.
- Prof. Laux is a founding member of DBTech.net ( http://www.dbtechnet.org/), an initiative of European universities and IT-companies to set up a transnational collaboration scheme for Database teaching. Together with colleagues from 5 European countries he has conducted projects supported by the European Union on state-of-the-art database teaching.
- He is a member of the ACM and the German Computer Society (Gesellschaft für Informatik).

# Objectives

- Graph Databases have become popular
- Particularly useful for forensic work
  - Fraud, Disinformation, Cyber attacks
- ISO Standardization effort is underway
- There is a standard benchmark test
- And it raises a research issue
  - How best to truncate searches of huge graphs
- This short paper addresses this issue
- And introduces the standardization activity





IARIA

# Database Language GQL

- Graph Databases have become popular (Neo4j was just the start)
- ISO 9075 Database Language SQL changed a lot in 2023
  - Chapter 16: Property Graph Queries (SQL/PGQ)
- New ISO Draft International Standard 2024
  - DIS 39075 – Database Languages – GQL
- Preview version available from ISO
- Public draft expected April 2024
- Many public discussion documents already

# Graph Queries are useful

- Initially popular in social media, following likes and indirection
- In areas such as fraud detection it is important to follow chains of money transfers, ownership, and responsibility
- With ordinary SQL this means lots of joins
- Graph queries use graph patterns and traversing these avoids creating explicit joins
- We could all simply use Neo4j
- But we already have a lot of products and so standardization is needed

# The LDBC Financial Benchmark

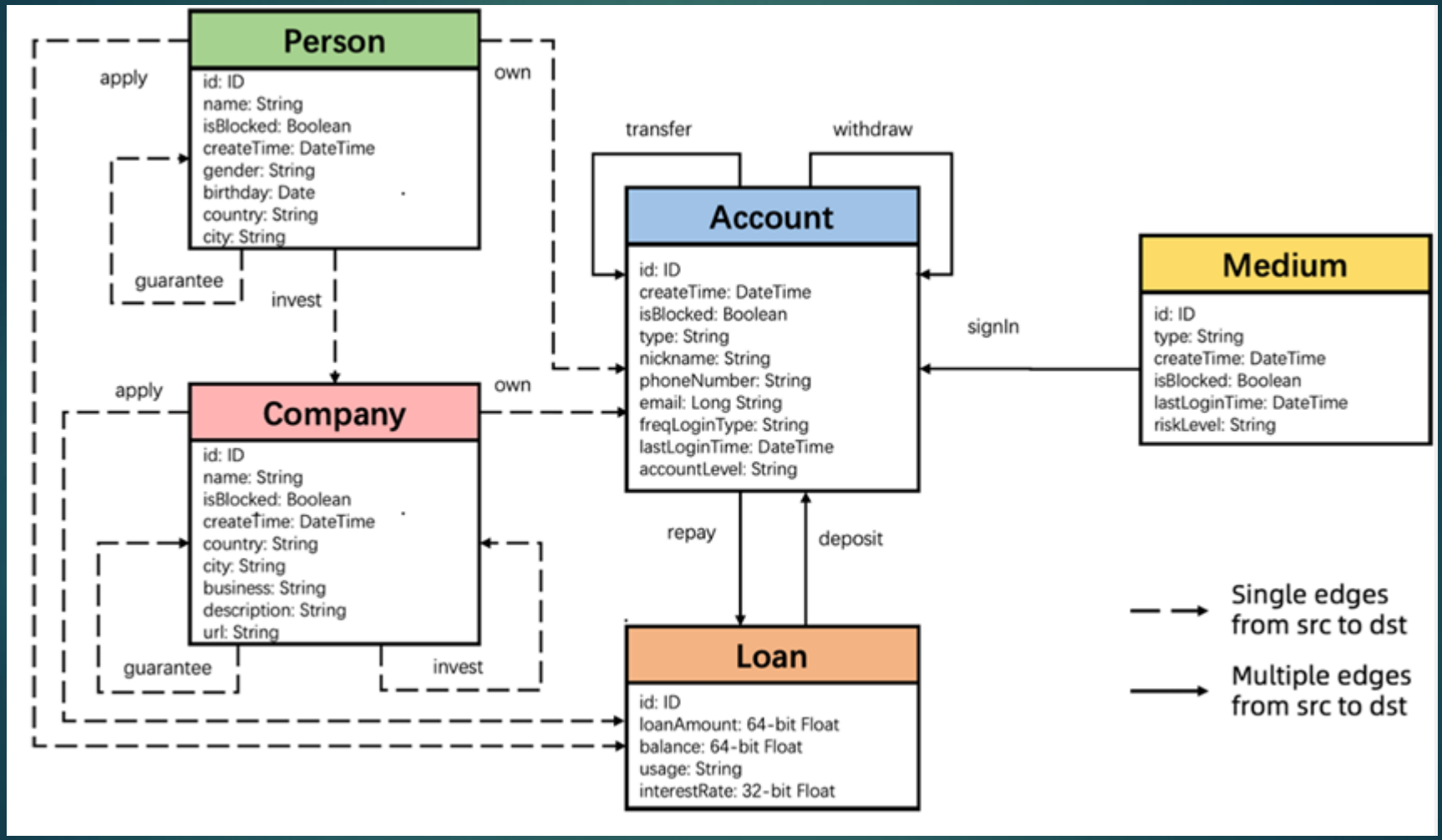
- Linked Data Benchmark Council
  - LDBCouncil.org
- Latest benchmark is for GQL
  - Financial Benchmark
  - Includes sample datasets ~5, 50, 500 MB
- Interesting constraints:
  - Transaction trails: timestamps increase
  - Truncation of graph pattern searches





IARIA

# The data model






From: LDBCouncil.org [3]

- A simple query

IARIA

# For example

- All transfers from accounts owned by Hatfield showing the amount, the date, and the payee

```
MATCH (:Person{name:'Hatfield'})-[:own]->()
-[:transfer{amount:m,"timestamp":d}]->()
<-[:own]-(:person{name:r})
```

- This joins 7 tables, each of which can be large
  - PERSON, OWN, ACCOUNT, TRANSFER, ACCOUNT, OWN, PERSON
- Traversal by row can be very efficient

```
SQL> MATCH (:Person{name:'Hatfield'})-[:own]->()-[:transfer{amount:m,"timestamp":d}]->()
<-[:own]-(:person{name:r})
|----------|-------------------|---------------|
|M         |D                  |R              |
|----------|-------------------|---------------|
|2977613.82|07/10/2022 04:35:24|Skundric       |
|6888877.75|16/10/2022 03:43:21|Hamahang       |
|989617.6  |26/10/2022 18:14:50|Buchkin        |
|4024112.15|27/10/2022 10:04:02|Alfaro Siqueiros|
|----------|-------------------|---------------|
SQL>
```

  - A complex query

# Patterns can repeat

- From the first query in the LDBC workload
  - Id1 is an input parameter
  - Other input parameters limit the timestamp

```
MATCH
 truncating ..
 trail p=(m:Medium{isBlocked:true})
  -[:signIn where ..]->
    (:Account{id:otherId})
   [()-[x:transfer where ..
     and later(p.x,"timestamp")]->()]{1,3}
    (:Account{id:id1}) return ..
```

- The pattern [..]{1,3}] can repeat up to 3 times
- Giving an array of transfers on each row



- The sequence constraint

IARIA

# The constraint on sequence

```
create function later (a Transfer array, t timestamp)
returns boolean
  begin
    declare c int=cardinality(a);
    if (c=0) then
       return true
    else
       return a[c-1]."timestamp"<t
   end if
  end
```

- In the query, this is called each time a transfer is added to the trail
  - a is the array of transfers so far
  - t is the timestamp of the transfer being added



- Truncation

IARIA

# Truncation

- Aims to limit the number of alternatives
  - At particular points in the search
- To avoid arbitrary loss of data
  - The limit is applied with a given ordering
  - To maximise relevance of the data returned
- We propose a syntax for specifying truncation

```
Truncation = TRUNCATING TruncationSpec
  {',' TruncationSpec} .
TruncationSpec = [EdgeType_id]
   ['(' OrderSpec {',' OrderSpec} ')'] '=' int .
```



- How it is applied

IARIA

# In our example

- TruncationOrder and TruncationLimit are parameters to the query

```
truncating Transfer
   ("timestamp" truncationOrder) =
  truncationLimit
```

- Useful values are DESC and 10
- Limits the number of transfers added to the search to 10 at each pattern node



IARIA

# Conclusions

- This proposed syntax has been implemented in our database PyrrhoDBMS, available on github
- The truncation mechanism (and PyrrhoDBMS itself) proved successful in this benchmark in giving efficient and realistic searches
- The limit can obviously be tuned



IARIA

# References


1. M. Crowe, and F. Laux, "Database Technology Evolution II: Graph Database Language", IARIA International Journal on Advances in Software, vol 16 (3-4) 2023, pp. 192-203, ISSN: 1942-2628
2. N. Francis, A. Gheerbrant, P. Guagliardo, L. Leonid, V. Marsault, et al.. A Researcher's Digest of GQL. 26th International Conference on Database Theory (ICDT 2023), Mar 2023, Ioannina, Greece. doi:10.4230/LIPIcs.ICDT.2023.1. https://hal.science/hal-04094449 [retrieved: Aug 2023]
3. Linked Data Benchmark Council: The LDBC Financial Benchmark (version 0.1.0), https://arxiv.org/pdf/2306.15975.pdf [retrieved Jan 2024]




IARIA