Property Graph Techniques in Relational Databases

MALCOLM CROWE AND FRITZ LAUX

DBKDA 2026

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. Recently this work has been extended to cover TypedGraph Models and GQL.
- Prof. Crowe is 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).

Why Property Graph Techniques?

- In some use cases, queries traverse many data links
  - And if every link requires a join of (maybe huge) tables..
- In graph databases, links are just edges
  - Graph patterns traverse edges ("horizontal aggregation")
  - Some patterns are like big edges, and can repeat in paths
  - A query can look for matches against a graph pattern
- Entire patterns can be inserted atomically
  - Where corresponding SQL needs multiple tables and keys

- But keep good relational ideas..

Keep the best of the relational model

- But let's keep the good things in SQL
  - Transactions, constraints, isolation, determinism, security
- Even better, let's allow alternative graph models
  - At the logical level, supported by a single physical model
- Example: Edges have properties in some models
- Rows of a relational table can be nodes – or edges!
- In some models, some edges are just like properties

Insert Graph patterns and ASCII art

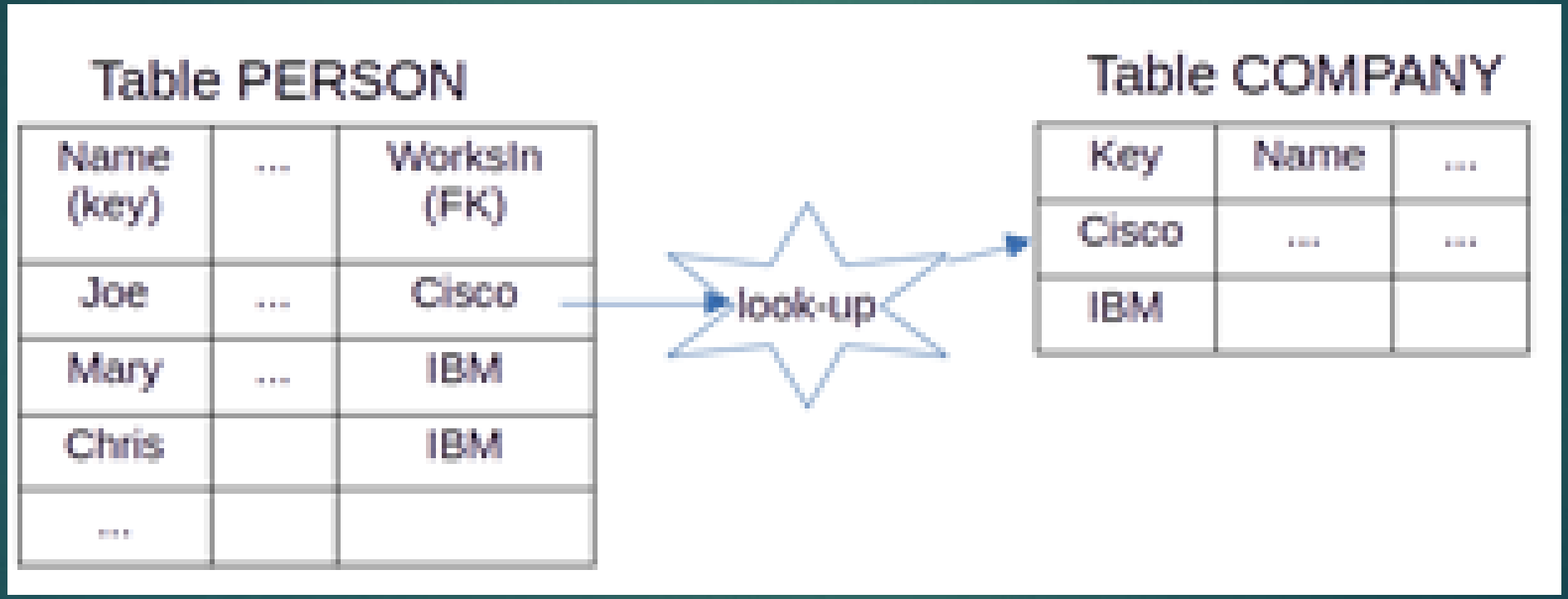
```
INSERT (:Person{Name:'Mary'})
   -[:Married {Date:'2025-07-24'}]->
       (:Person{Name:'Joe'})
   -[:WorksIn]->(:Company{Name:'Cisco'})
```

- This constructs three nodes and two directed edges
- Maybe also specifies some new node types and edge types
  - If so, future objects will have similar properties

- SQL uses foreign keys for edges

In SQL, links use foreign key values

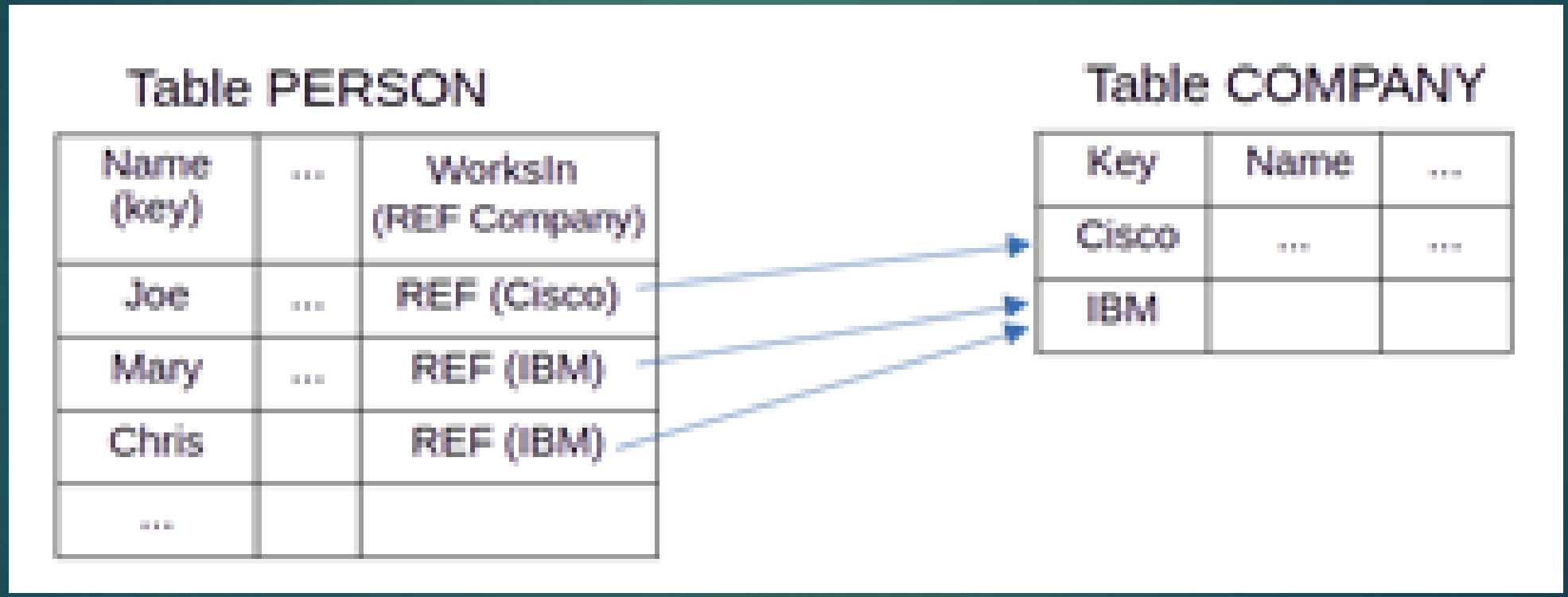
- For example, Worksin will be a column in PERSON
  - Whose values are keys in the COMPANY table



- They could use references instead

SQL relations with reference values

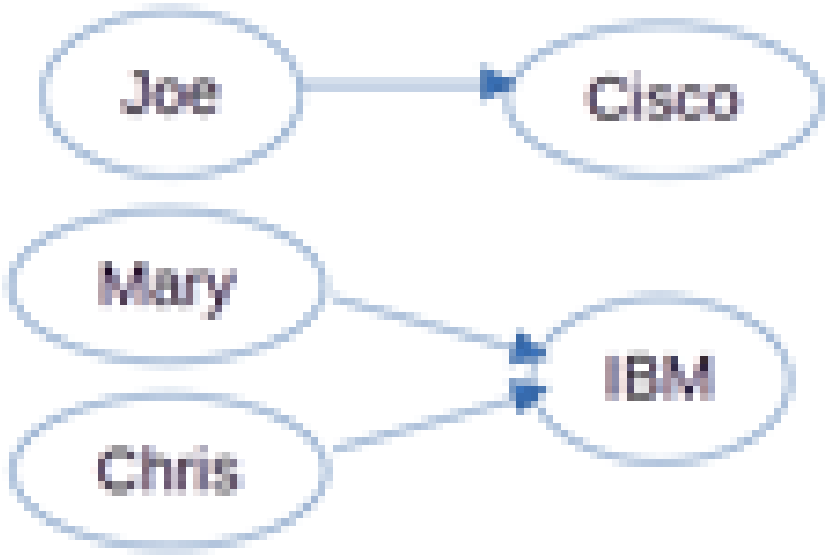
- If the DBMS has "row uids" we could use them
  - Many DBMS (Oracle…) have them for internal use

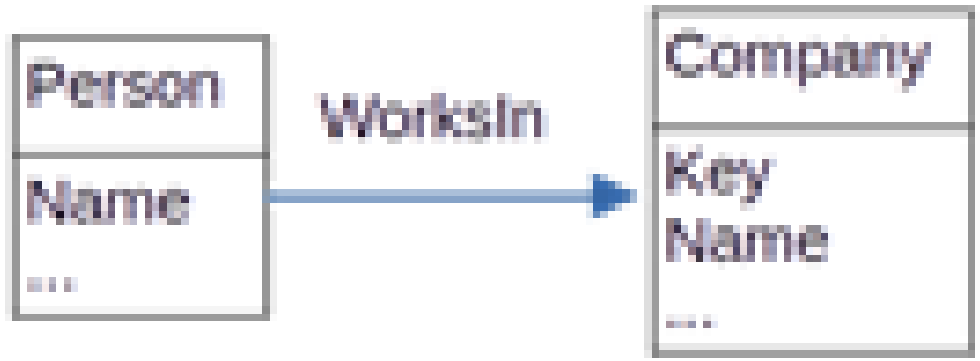


- Graphs vs UML data models

Graphs and data models

- Graph DBMS can draw pictures,
- Relational DBMS can have a UML model





- What is an edge?

Edges like Married have 2 foreign keys

- Or two reference columns..
- WorksIn just has one reference column
- Does it make sense to have more than two?
- Many SQL tables have more than 2 foreign keys!
- In such cases different models can choose which columns create edges, and choose direction
- Maybe hide or rename some properties
- But be based on the same SQL structure!

- Uniform implementation

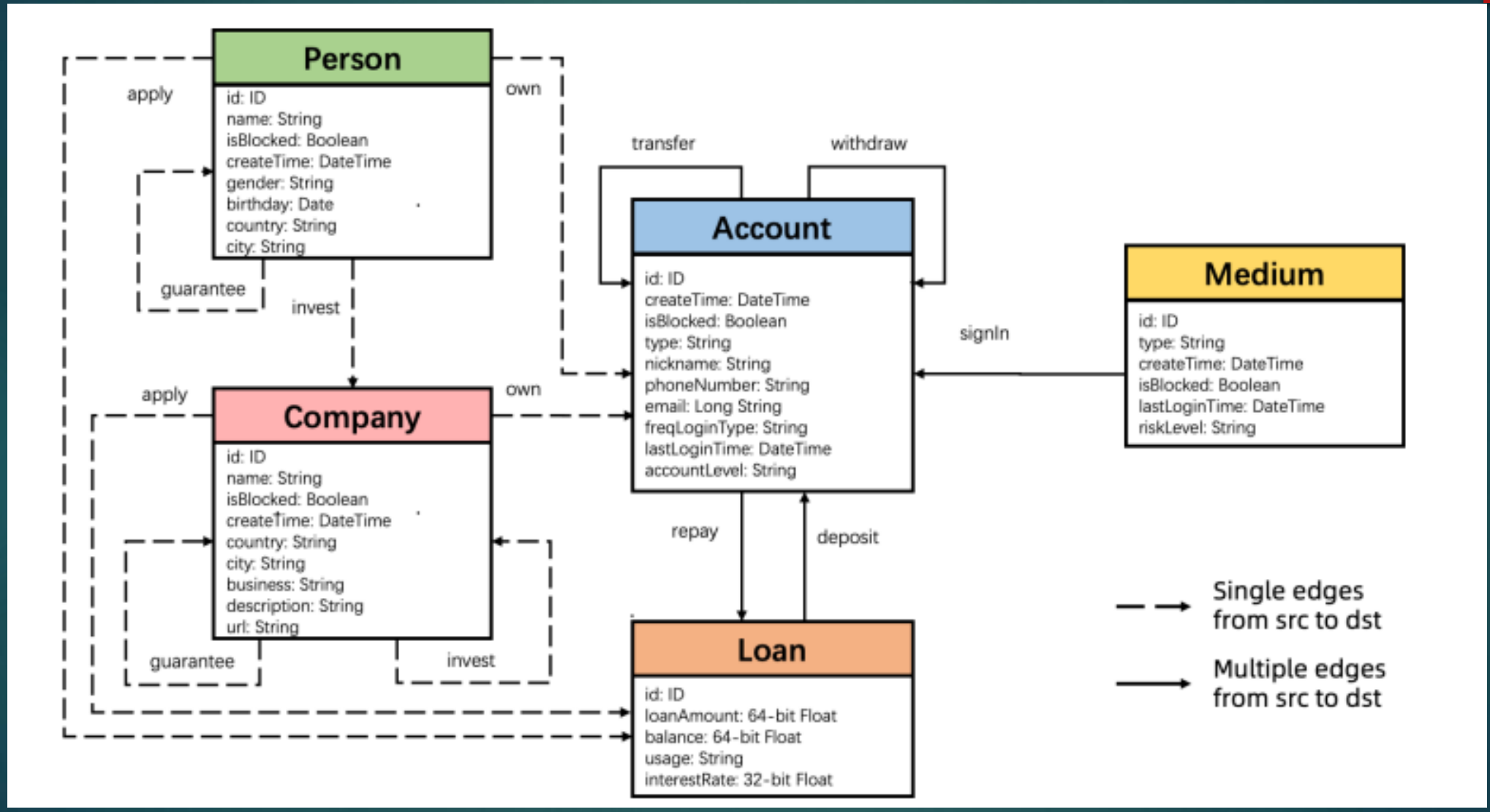
Foreign keys or references?

- For record-based databases, we can simplify things
- By always implementing links by reference values
- Keep foreign keys if these have been defined
  - Primary keys can be useful for looking things up
- Some links can be specified as the record is created

Experiments

- There is a test suite PyrrhoTest on github
- The GDC Financial Benchmark with changes:
  - Subtypes for nodes and edges, ternary Transfer edges
- Match used for connecting the ternary edges
- Showed excellent scalability
- Sf001, Sf01, Sf1 have sizes 5MB, 52 MB and 531 MB
  - Built in 62.8 sec, 625 sec, and 6720 sec
  - Using a desktop PC with Intel Ultra 5

The GDC Financial Benchmark



Loading the benchmark files

insert into medium(id,type,isblocked,createtime,lastlogin,risklevel) values ~c:\LDBC\sf001\Medium.csv

set referencing // convert ids to pos

insert into personapplyloan(from1,to1,createTime,organization) values ~c:\LDBC\sf001\PersonApplyLoan.csv

Subtypes and primary keys

```
create type legalentity as(name string,isBlocked boolean,createtime timestamp,country string,city string) nodetype

create type person under legalentity as (id int,gender string,birthday timestamp) primary key(id)

create type company under legalentity as(cid int,business string,description string,url string) primary key(cid)
```

Ternary edge for transfer and its subtypes

```
create type activatedfor as (createtime
timestamp,location string) edgetype (med=medium,account)

create type transfer as(amount float64,createTime
timestamp,orderNumber int optional,comment string
optional,payType string optional,goodsType string
optional) edgetype(from actbase with activatedfor
optional to actbase)

alter table transfer alter with1 add default
(MATCH(FROM1)<-[a:activatedfor where
a.createtime>=from1.createtime]-() limit 1 yield a)

create type accountrepayloan under transfer edgetype(from
account to loan)
```


References

- S. Plantikow, "Towards an International Standard for the GQL Graph Query Language." *W3C workshop in Berlin on graph data management standards*. Vol. 84. 2019.
- https://www.w3.org/Data/events/data-ws-2019/assets/position/Stefan%20Plantikow.pdf
- ISO/IEC 39075:2024, Information technology — Database languages — GQL, Published (Edition 1, 2024).
- Graph Data Council (formerly Linked Data Benchmark Council), Financial Benchmark https://ldbcouncil.org/benchmarks/finbench/
- See https://pyrrhodb.blogspot.com for an introduction.