UNIVERSITY OF THE WEST of SCOTLAND
UWS

Typed Graph Models versus Relational Databases

MALCOLM CROWE

TUTORIAL VALENCIA 2026

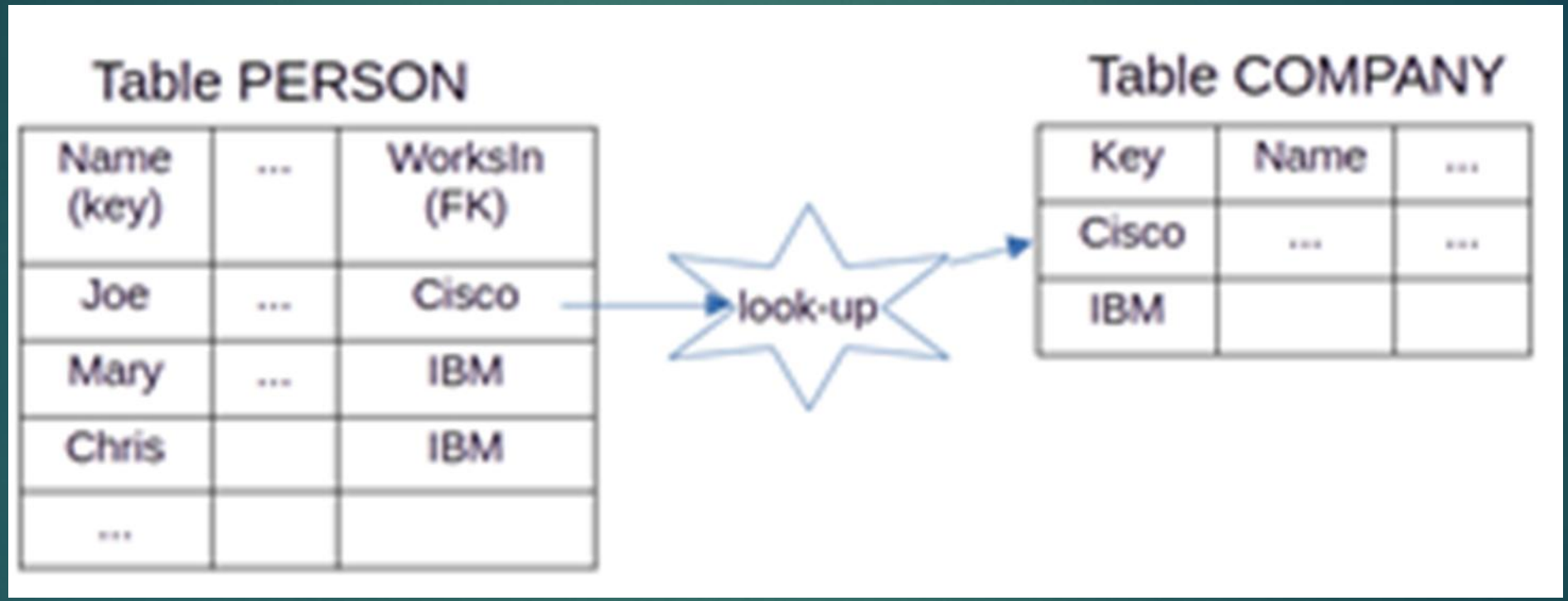
IARIA

Malcolm Crowe

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

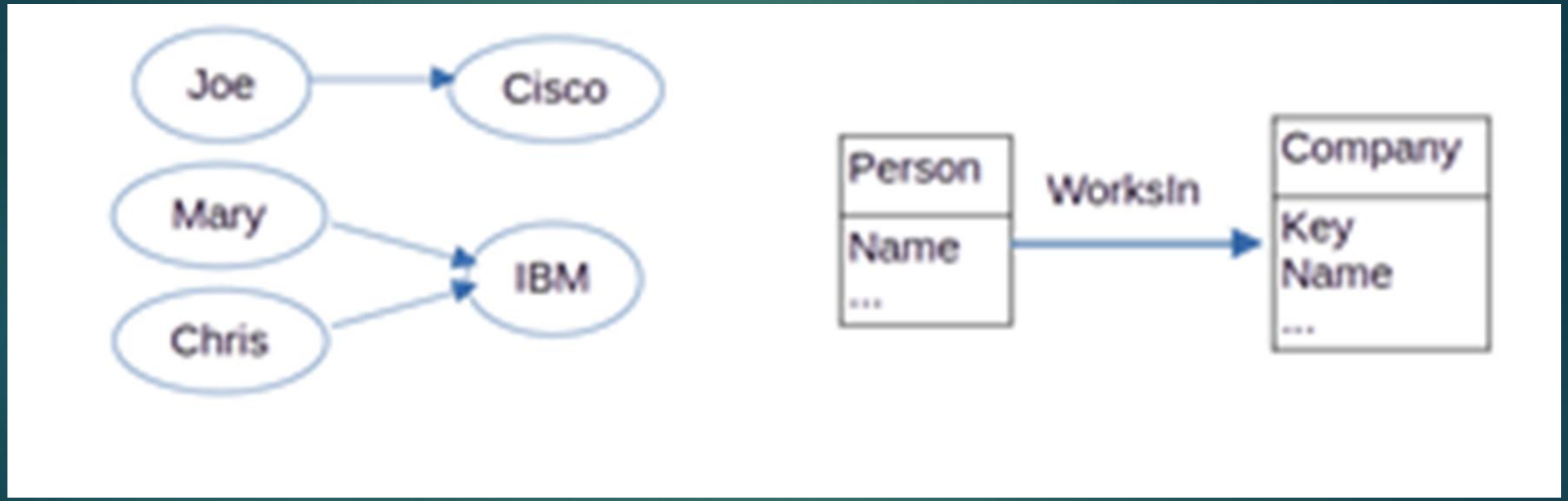
- Malcolm Crowe is an Emeritus Professor at the University of the West of Scotland, where he worked from 1972 (when it was Paisley College of Technology) until 2018.
- He gained a D.Phil. in Mathematics at the University of Oxford in 1979.
- He was appointed head of the Department of Computing in 1985. 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.

In SQL links use keys and indexes



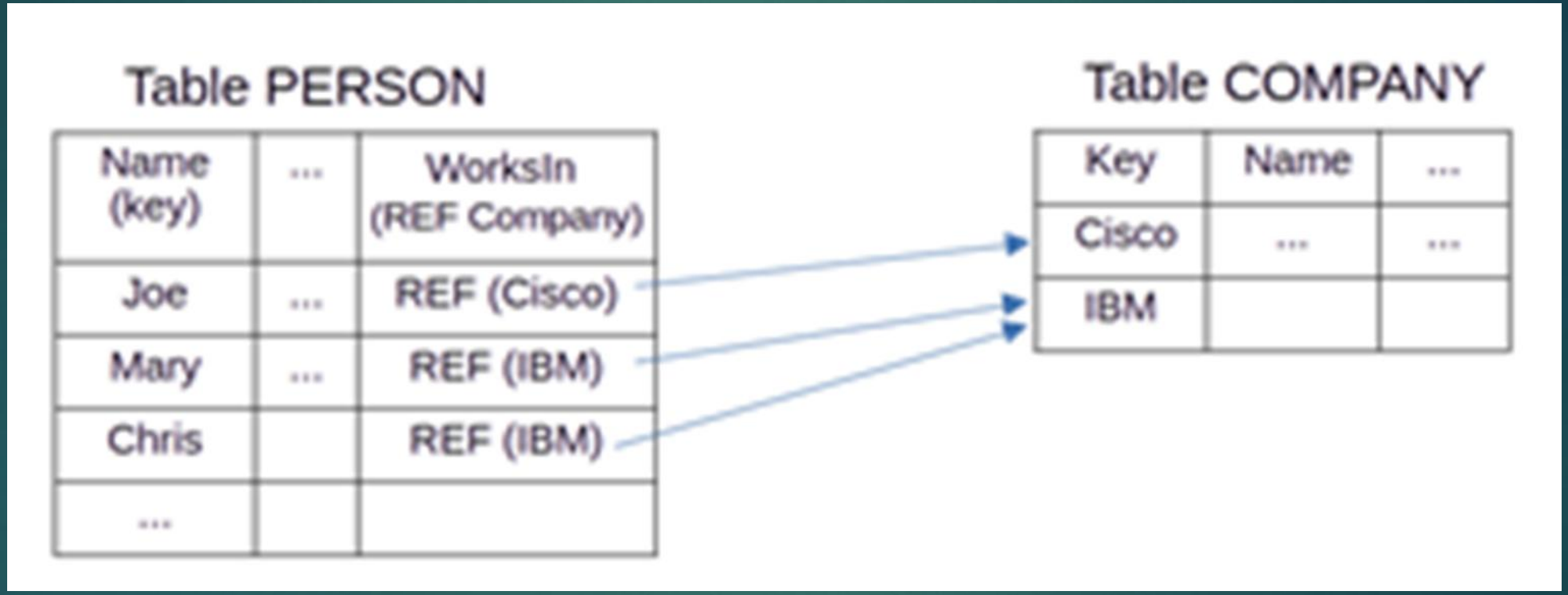
- Graphical approach

Graphical databases use edge types



- Use reference values

Simnpler to use reference values!



- SQL and GQL standards

International Standards: SQL and GQL

- ISO/IEC 9075: SQL used for decades: 12 sections
  - Updated frequently: latest version 2023 (some sections)
  - ISO/IEC 9075-16:2023 Property Graph Queries (SQL/PGQ)
  - Most commercial RDBMS sort of follow some version of this
- ISO/IEC 39075: GQL v.1 2024
  - No full implementations yet
  - Compatible with SQL

  - Database of records

Databases of Records

- Let's think of our database as containing records
- Each record has a set of properties (name,value)
- Groups of properties can be labelled (a named type)
  - Record added by a known user with a given role
  - In a transaction which committed at a particular time
- The database is the transaction log (append only)
  - Records can be updated or deleted from the database
    - But not from the log!
    - Records retain their identity for ever (location in the log is a uid)

- More details

Some more details about records

- Any record can have more than one label
  - Sets of properties can overlap
  - Property values are strongly typed
    - We have subtypes, collection types (sets, lists, multisets..)
    - And reference values (record uids) are a special type called REF
  - Properties can be optional (or not)
- We can define indexes for efficiency
- All of this can be handled with relational technology
  - But maybe not with your favourite DBMS
    - SQL style statements

SQL-style relational statements

- Create table, type (define label, property list, etc)
- Insert, Delete, Update records
- Queries: Select from tables and joins, aggregation
- Procedures, triggers
- Grant privileges to users, roles
- The only way to follow links in SQL is to make joins

- SQL example code

Example SQL style

```
create table Person (name string, worksin
string)
create table Company (name string)
```

Making a link

- To link the tables, we need keys

```
create table Company (name string primary key)
create table Person (name string, worksin string
references Company)
```

- This really only works for one-way references
- This creates two indexes for the keys
- And then we need insert statements to add the rows

- The GQL way

Nicer to be able to write

```
INSERT (:Person{name:'Fred'})-worksin
->(:Company{name:'AWS'})
```

- We'd like this to do everything
  - create both tables, insert records, and form the link

- Does GQL do this?

Does GQL do this?

- GQL has this sort of syntax but

```
INSERT (:Person{name:'Fred'})-[:worksin]
->(:Company{name:'AWS'})
```

- makes another element type Worksin and two links
- Similar to what we need in SQL for many-many links

Querying with links

- In SQL this always uses joins
- SELECT from Person p, Company c where p.worksin=c.name
- In GQL we have a powerful statement called MATCH
- MATCH (p)-[]->(c) return p.name,c.name
- Match's graph pattern can have many links, repeats
  - Details in the brackets select nodes, labels
  - Can be combined with , and |

- GQL style statements

GQL-style statements

- (In addition to SQL-style)
- Insert graph: draw a graph pattern using "ASCII-art"
- (..) for nodes, [..] for edges, {..} for properties
- And arrows (..)-[..]->(..) (..)<-[..]-(..) (..)~[..]~(..)
- Creates nodes, edges with given properties
- Match graph: similar-looking graph pattern
- Unbound identifiers get rows of values from data
  - Also allows repeating path patterns

- The data model so far

The data model so far

- We have labeled groups of typed properties
- Some constraints (e.g. uniqueness, multiplicity,..)
- Referential constraints (no orphaned references)
- Subtypes have more properties, supertypes fewer
- Subtypes must maintain the supertype's constraints
- The model can be changed
  - by someone with the right privileges
- The current state of the database is in memory
  - And is maintained consistent with the data model

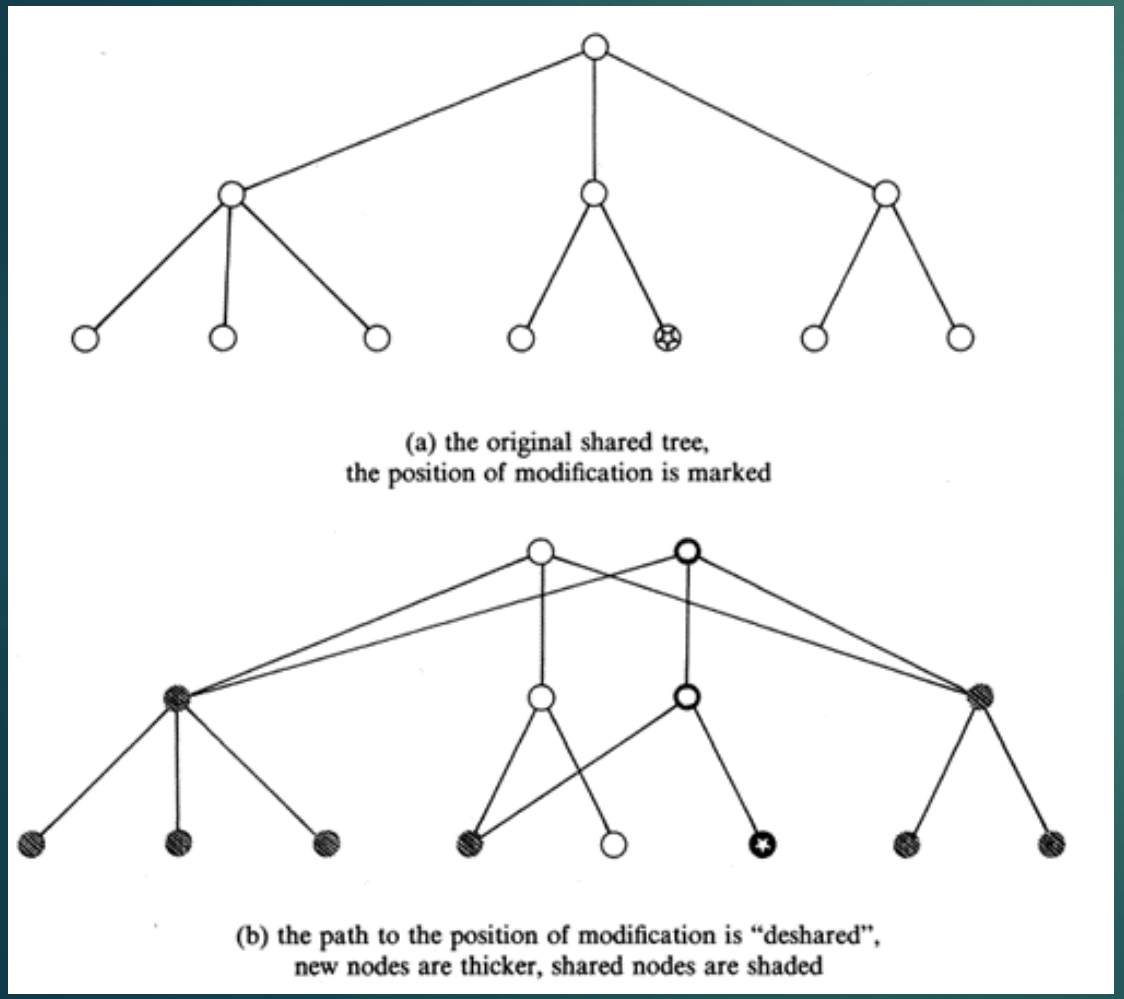
Something about PyrrhoDBMS

- Research project (2001-): a DBMS implementation
- Open source, free to use (on github)
- Record based, full transaction log, append storage
- Insists on ACID: serialising validation, no locking
- All main features of SQL (udt, triggers, ..)
- Added subtypes and new strong types
- Has explored partitioned, RDF/SPARQL etc
- Based on Shareable data types since 2018
- Is implementing GQL on top of SQL

- How shareable data works

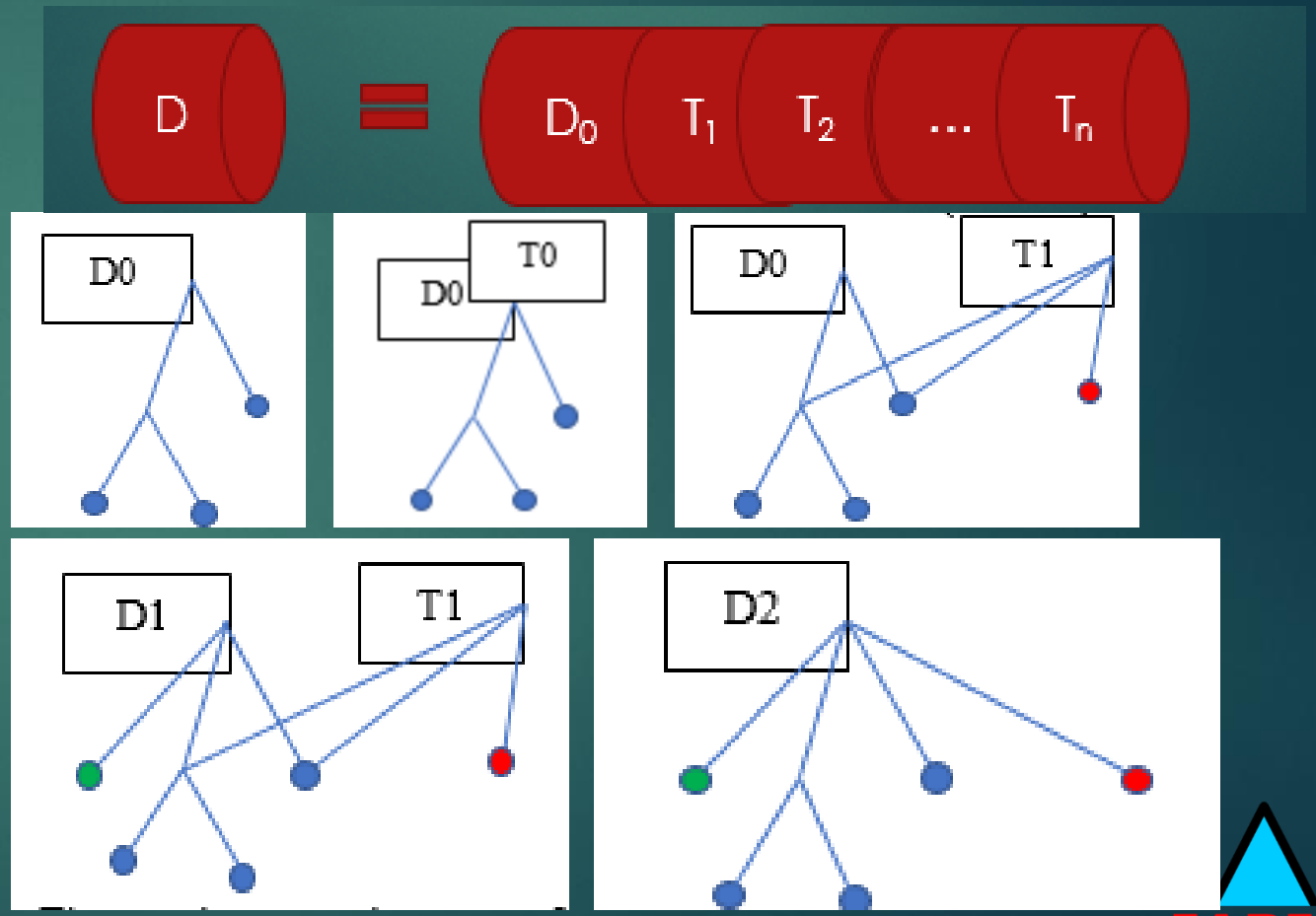
Shareable data types, transactions

- Like strings: change gives new string, new database



(a) the original shared tree,
the position of modification is marked

(b) the path to the position of modification is "deshared",
new nodes are thicker, shared nodes are shaded



- Transactions review

The transaction log and its format

```
QL> table "Log$"
|---|-----------------------------------------------------
|Pos|Desc
|---|-----------------------------------------------------
|5  |PTransaction for 3 Role=-502 User=-501 Time=26/09
|23 |PTable A
|29 |PColumn3 B for 23(0)[ INTEGER]
|50 |PColumn3 C for 23(1)[ CHAR]
|71 |PTransaction for 1 Role=-502 User=-501 Time=26/09
|89 |Record 89[23]: 50=Three[CHAR]
|---|-----------------------------------------------------
QL>
```

- See Pyrrho doc folder
  - Pyrrho.pdf
  - SocurceIntro.pdf

9.2 Record formats

The record formats are as follows (note that
indicated below):

Code	Record type	
	Physical	1 integer (
0	*EndOfFile*	*4 bytes (v*
1	Table	1 string (n
2	Role	2 strings (
3	Column	1 integer (

Transactions

- To add something to the database transaction log
  - And/or change the model
  - Changes must be committed (autocommit is an option)
  - Until then, nobody else can see them (isolation)
- The DBMS performs a validation step on commit
  - Records must be consistent with the data model
  - Updates may conflict with changes made by other users
- Transactions are serialised on commit (for the log)
  - Multiple transactions may be concurrently in preparation
  - If not committed, they can simply be forgotten (no change)
    - Links between records

References and links

- Reference values represent links between records
- The type of a reference is the type of the linked record
  - Or maybe a supertype
- Relational data model has foreign keys to create links
- The DBMS can use reference values behind the scenes
  - Automatically translating keys to references and vice versa
- To maintain consistency the DBMS keeps track of references
  - Like reverse indexes, this information helps follow links, explore data

It makes sense to allow GQL in databases

- Using graph insert we can add related information
  - In a single step, capturing data references
- Using match, we can create references, add links
- Follow chains: cash transfers, suppliers, dependency

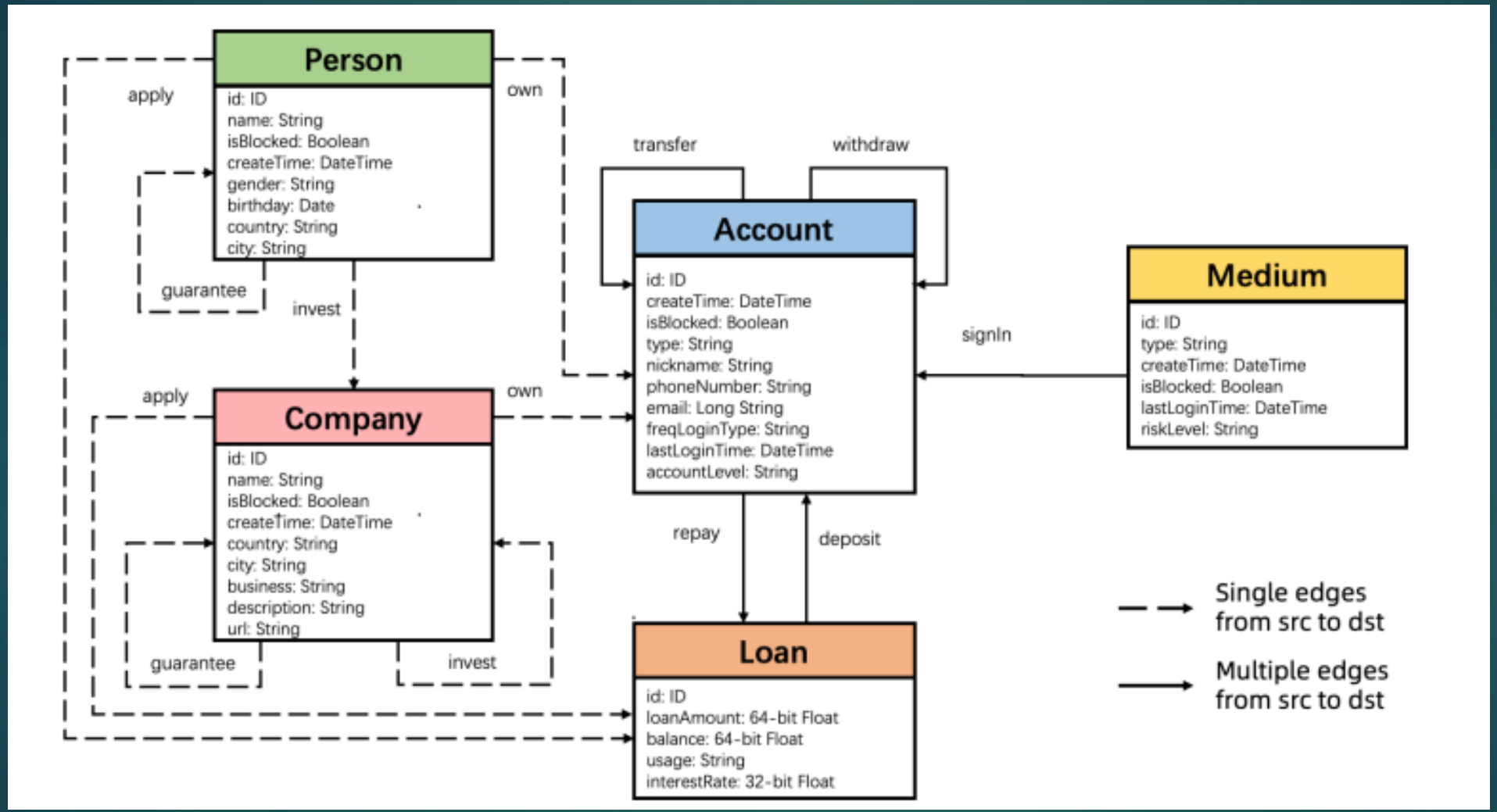
Is there a graph model and what is it?

- What are the nodes and what are the edges?
- We can decide which reference values make edges
- We may implement properties as simple edges
  - E.g. bank branch, sort code
- Edges in most graph models have two ends
  - and maybe a direction (and set-valued ends?)
  - But many SQL tables have more than 2 foreign keys
  - Lots of different graph models for the same database!

Defining the graph model

- GQL offers a closed graph model
- Full specification of node types and edge types
- Specifying connections for edges
- Each graph model defined by a URL
- An alternative is to add metadata to the data model
- Create the graph model as a role in the database

Experiments: The GDC Financial Model



- Latest results

Latest results

- Work in progress (see references)
- Github has a version of PyrrhoV8 including a test suite for its versions of SQL and GQL and documentation
- DBKDA2026 reports on initial tests with the Financial Benchmark: two versions of its data model spec
- An unpublished PyrrhoV8 version has built the Financial Benchmark with ternary edges on a desktop PC Ultra 5
  - sf001 5MB in 62.8 sec
  - sf0152MB in 625 sec
  - sf1 531MB in 6720 sec

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.
- 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
- https//github.com/MalcolmCrowe/ShareableDataStructures PyrrhoV8alpha subfolder
- dbkda_2026_1_50_50018.pdf for DBKDA 2026 paper

IARIA