Graph Databases

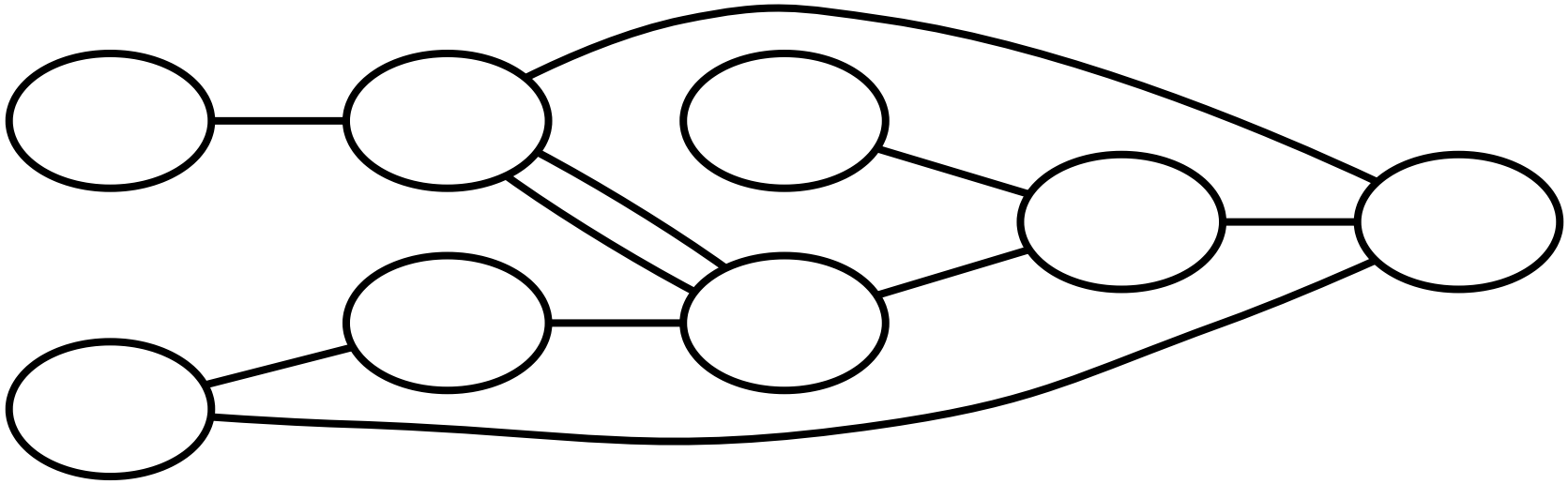
Introduction, Standardization, Opportunities

Peter Eisentraut

2ndQuadrant®
PostgreSQL

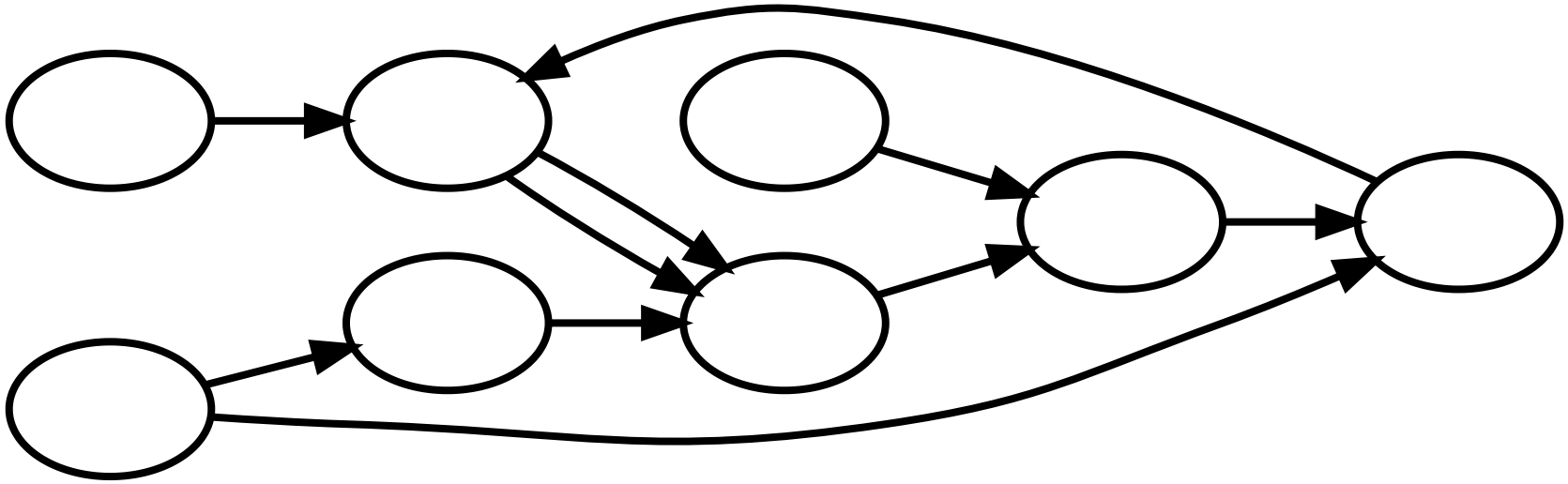
peter.eisentraut@2ndquadrant.com
@petereisentraut

graph

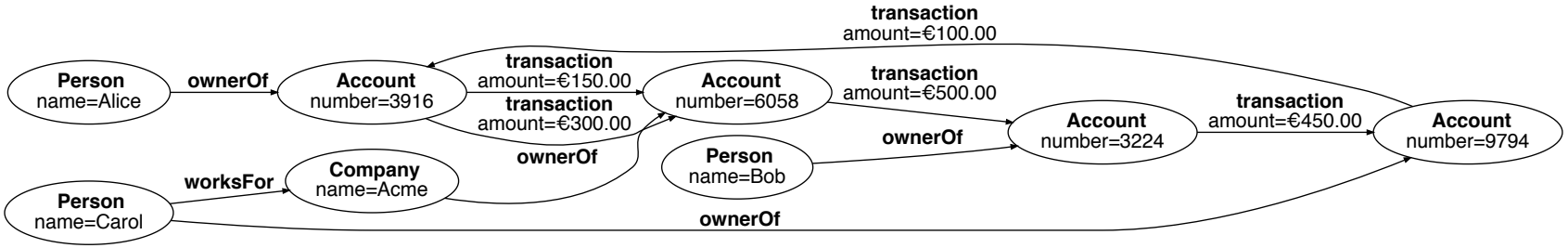


terms: vertex, node; edge, relationship, arc

directed graph

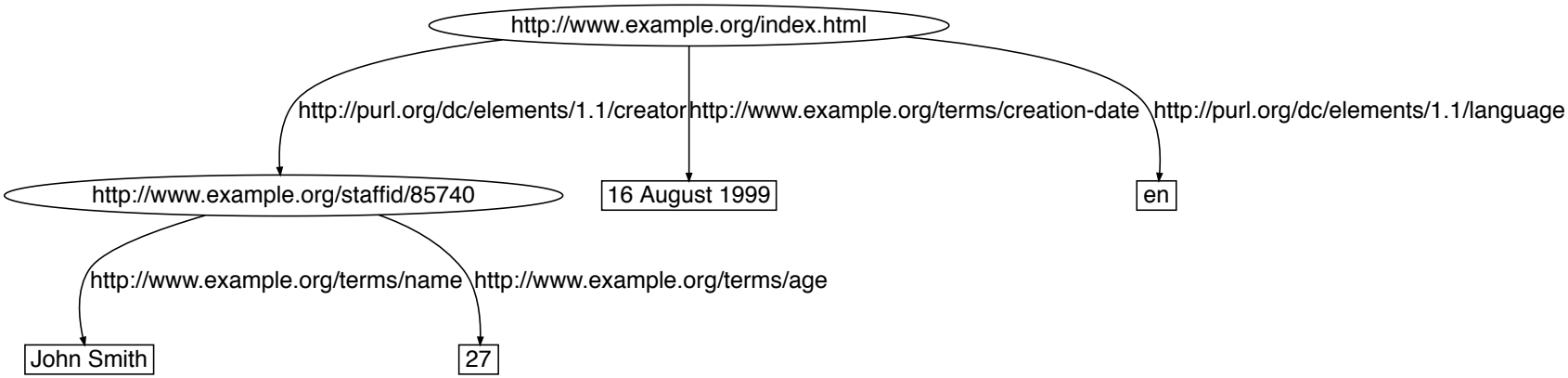


property graph



terms: property, label

RDF



terms: triple, subject, predicate, object

property graph vs. RDF

	PG	**RDF**
standardization	ISO	W3C
languages	Cypher, PGQL, G-CORE, GSQL, GQL	SPARQL, OWL
serialization	(CSV)	XML, JSON
vendors	Neo4j, Oracle, TigerGraph, AWS	Virtuoso, Apache, AWS, many
logic	closed-world	open-world(?)

~~GraphQL~~

graph database uses

- social network
- recommendations
- knowledge representation
- bioinformatics
- logistics
- public infrastructure
- finance analytics
- access control

SPARQL

W3C RDF query language

```
PREFIX ex: <http://example.com/exampleOntology#>
SELECT ?capital
       ?country
WHERE
  {
    ?x  ex:cityname        ?capital   ;
        ex:isCapitalOf     ?y         .
    ?y  ex:countryname     ?country   ;
        ex:isInContinent   ex:Africa  .
  }
```

Cypher

graph query language by Neo4j

```
MATCH (nicole:Actor {name: 'Nicole Kidman'})-[:ACTED_IN]->(movie:Movie)
WHERE movie.year < $yearParameter
RETURN movie
```

PGQL

graph query language by Oracle

```
SELECT owner.name AS account_holder,
       SUM(t.amount) AS total_transacted
FROM financial_transactions
MATCH (p:Person) -[:ownerOf]-> (:Account)
      -[t:transaction]- (:Account) <-[:ownerOf]- (owner:Person|Company)
WHERE p.name = 'Alice'
GROUP BY owner
```

G-CORE

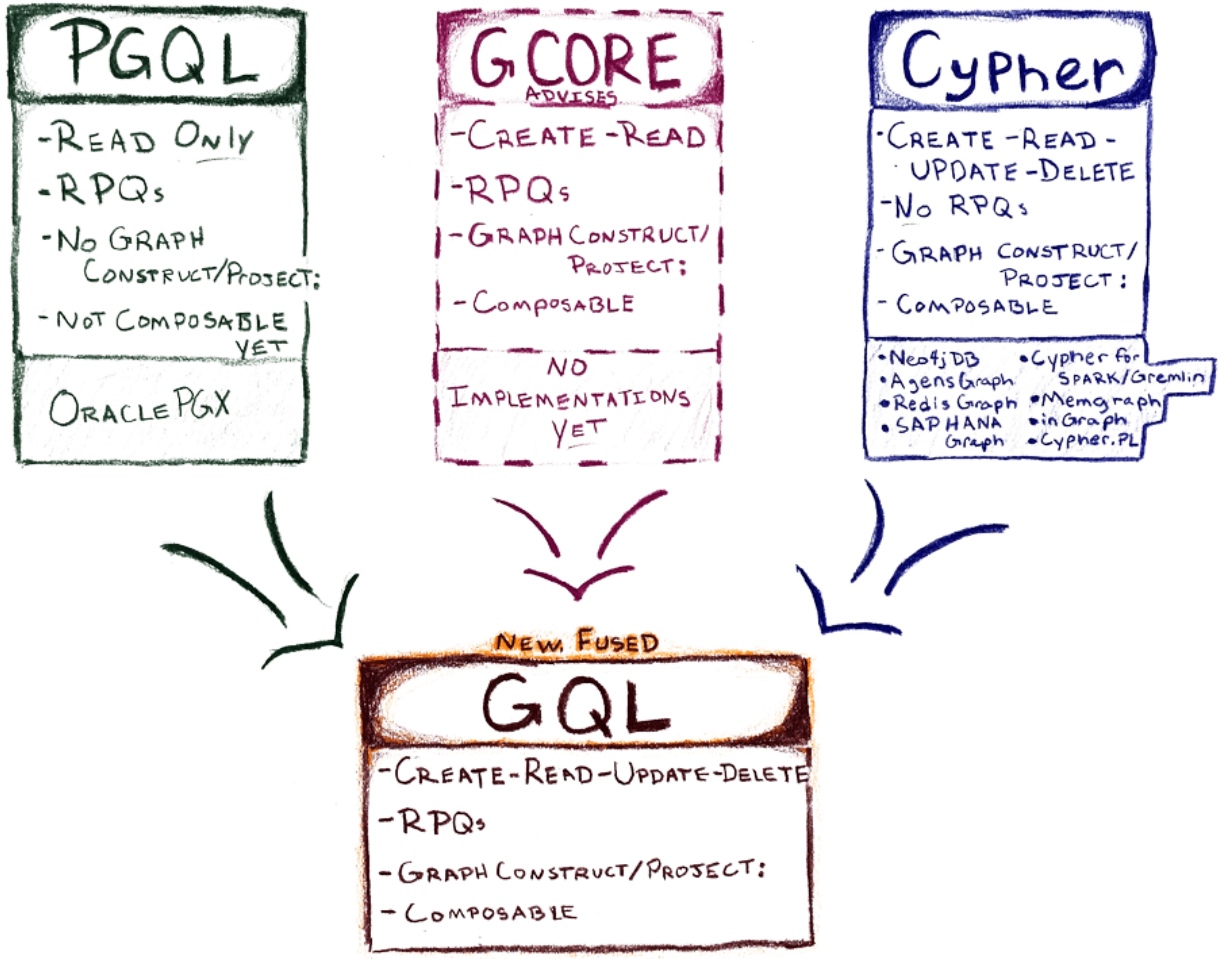
graph query research language by LDBC

```
CONSTRUCT (c)<-[:worksAt]-(n)
MATCH (c: Company) ON company_graph,
      (n: Person) ON social_graph
WHERE c.name = n.employer
```

The GQL Manifesto

https://gql.today/

Cypher + PGQL + G-CORE = GQL?


PGQL
-READ ONLY
-RPQs
-NO GRAPH CONSTRUCT/PROJECT:
-NOT COMPOSABLE YET
ORACLE PGX
G CORE
ADVISES
-CREATE -READ
-RPQs
-GRAPH CONSTRUCT/PROJECT:
-COMPOSABLE
NO IMPLEMENTATIONS YET
Cypher
-CREATE -READ - UPDATE -DELETE
-NO RPQs
-GRAPH CONSTRUCT/PROJECT:
-COMPOSABLE
•Neo4j DB
•Agens Graph
•Redis Graph
•SAP HANA Graph
•Cypher for SPARK/Gremlin
•Memgraph
•inGraph
•Cypher.PL
NEW, FUSED
GQL
-CREATE-READ-UPDATE-DELETE
-RPQs
-GRAPH CONSTRUCT/PROJECT:
-COMPOSABLE

GQL

- new standardization project of ISO/IEC JTC1 SC32 WG3 (ISO 39075?)
- could be ready in 3–4 years
- not compatible with SQL

SQL/PGQ

- will be new SQL:202x part 16
- read-only graph queries on top of tables

SQL/PGQ: create tables

```
CREATE TABLE person ( ... );
CREATE TABLE message ( ... );

CREATE TABLE created ( ... );
CREATE TABLE commented ( ... );
```

SQL/PGQ: create graph

```
CREATE PROPERTY GRAPH my_graph
    VERTEX TABLES (person, message)
    EDGE TABLES (
        created SOURCE person DESTINATION message,
        commented SOURCE person DESTINATION message
    );
```

SQL/PGQ: query graph

```
SELECT gt.creation_date, gt.content
FROM my_graph GRAPH_TABLE (
  MATCH
    (creator IS person WHERE creator.email = 'foo@example.com')
      -[ IS created ]->
    (m IS message)
      <-[ IS commented ]-
    (commenter IS person)
    WHERE creator.email <> commenter.email
    COLUMNS (m.creation_date, m.content)
) AS gt;
```

SQL/PGQ: another query

```
SELECT id, name
FROM movies_graph GRAPH_TABLE (
  MATCH
    (nicole:Actor WHERE name = 'Nicole Kidman')
    -[:ACTED_IN]->
    (movie:Movie)
    WHERE movie.year < $1
    COLUMNS (movie.id, movie.name)
) AS gt;
```

SQL/PGQ: and another one

```
SELECT owner_name AS account_holder,
       SUM(t_amount) AS total_transacted
FROM financial_transactions GRAPH_TABLE (
  MATCH (p:Person) -[:ownerOf]-> (:Account)
        -[t:transaction]- (:Account) <-[:ownerOf]- (owner:Person|Company)
  WHERE p.name = 'Alice'
  COLUMNS (owner.name AS owner_name, t.amount AS t_amount)
) AS ft
GROUP BY owner_name;
```

summary

- property graphs
- GQL
- SQL/PGQ

links and credits

- RDF
  - https://www.w3.org/TR/rdf-primer/
- SPARQL
  - https://en.wikipedia.org/wiki/SPARQL
  - https://www.w3.org/TR/sparql11-overview/
- Cypher
  - https://en.wikipedia.org/wiki/Cypher_Query_Language
  - https://neo4j.com/docs/cypher-manual/current/
  - https://www.opencypher.org/
- PGQL
  - http://pgql-lang.org/
- G-CORE
  - http://ldbcouncil.org/sites/default/files/main-cr.pdf
- GQL
  - https://www.gqlstandards.org/
- SQL/PGQ
  - https://www.w3.org/Data/events/data-ws-2019/assets/lightning/OskarVanRest.pdf
- https://www.w3.org/Data/events/data-ws-2019/