Graph Databases

Introduction, Standardization, Opportunities

Peter Eisentraut

2ndQuadrant

PostgreSQL

peter.eisentraut@2ndquadrant.com
@petereisentraut
graph

terms: vertex, node; edge, relationship, arc
directed graph
terms: property, label
terms: triple, subject, predicate, object
## Property Graph vs. RDF

<table>
<thead>
<tr>
<th></th>
<th>PG</th>
<th>RDF</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>standardization</strong></td>
<td>ISO</td>
<td>W3C</td>
</tr>
<tr>
<td><strong>languages</strong></td>
<td>Cypher, PGQL, G-CORE, GSQL, GQL</td>
<td>SPARQL, OWL</td>
</tr>
<tr>
<td><strong>serialization</strong></td>
<td>(CSV)</td>
<td>XML, JSON</td>
</tr>
<tr>
<td><strong>vendors</strong></td>
<td>Neo4j, Oracle, TigerGraph, AWS</td>
<td>Virtuoso, Apache, AWS, many</td>
</tr>
<tr>
<td><strong>logic</strong></td>
<td>closed-world</td>
<td>open-world (?)</td>
</tr>
</tbody>
</table>
graph database uses

- social network
- recommendations
- knowledge representation
- bioinformatics
- logistics
- public infrastructure
- finance analytics
- access control
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
SELECT owner.name AS account_holder,
       SUM(t.amount) AS total_transacted
FROM financial_transactions
MATCH (p:Person) -[:ownerOf]-> (:Account)
       -[: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

**GCore**
- Create - Read
- RPQs
- Graph Construct/Project
- Composable
- No Implementations Yet

**Cypher**
- Create - Read - Update - Delete
- No RPQs
- Graph Construct/Project
- Composable

**New Fused GQL**
- Create - Read - Update - Delete
- RPQs
- Graph Construct/Project
- Composable

*Neo4j, DB
*AgentsGraph, SPARQL/Gremlin
*Redis Graph, Memgraph
*SAP HANA, pGraph
*Cypher, PL
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 ( ... );
CREATE PROPERTY GRAPH my_graph
   VERTEX TABLES (person, message)
   EDGE TABLES (  
      created SOURCE person DESTINATION message,
      commented SOURCE person DESTINATION message
   );
SELECT gt.creation_date, gt.content
FROM my_graph GRAPH_TABLE (QUALIFY
    (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;
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)  
    -[: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://www.w3.org/TR/sparql11-overview/
- Cypher
  - https://neo4j.com/docs/cypher-manual/current/
  - https://www.opencypher.org/
- PGQL
  - http://pgql-lang.org/
- G-CORE
- GQL
  - https://www.gqlstandards.org/
- SQL/PGQ
  - https://www.w3.org/Data/events/data-ws-2019/