Querying Connected Data

Graph Databases and Triple Stores

About Me

- Software Developer and Architect
- Data Architect at OUP
- Contributor / Project Lead on dotNetRDF
- Developer on BrightstarDB

DISCLAIMER: I'm an RDF guy ... kinda :-)

Agenda

- Graphs and Graph Databases
- Triple Stores
  - RDF Graphs
  - Import and Export
  - Query and Update
  - Reasoning
- Property Graph Databases
  - Property Graphs
  - Query
  - Reasoning with Property Graph Databases
- Comparison

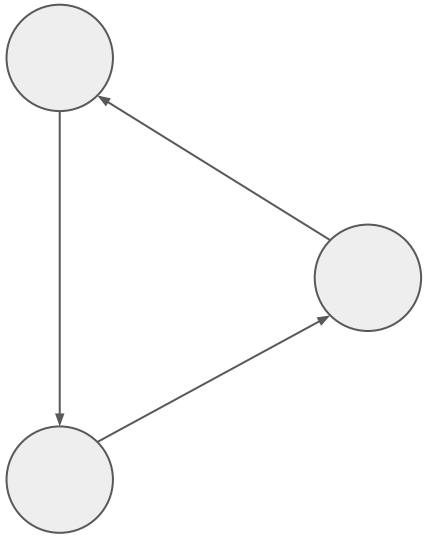
Graphs

Nodes connected by *edges*

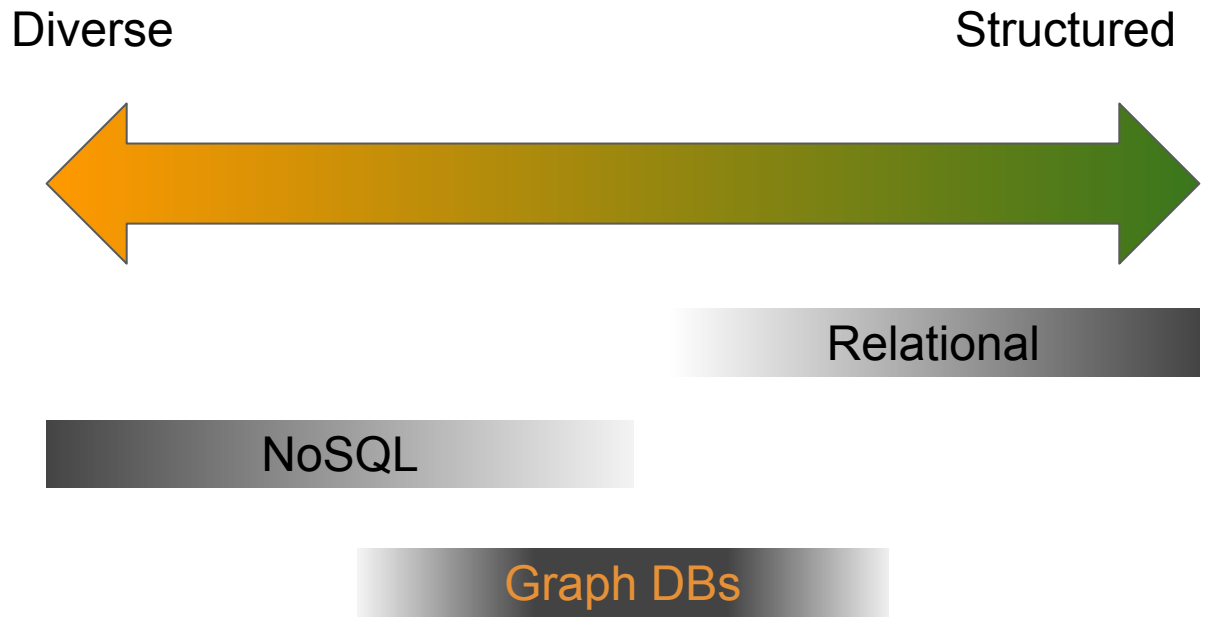
Each edge connects exactly two nodes

Edges may be run "from" one node to another => *Directed graphs*

Edges may form *cycles* that take you on a path from a given node back to that same node following two or more edges

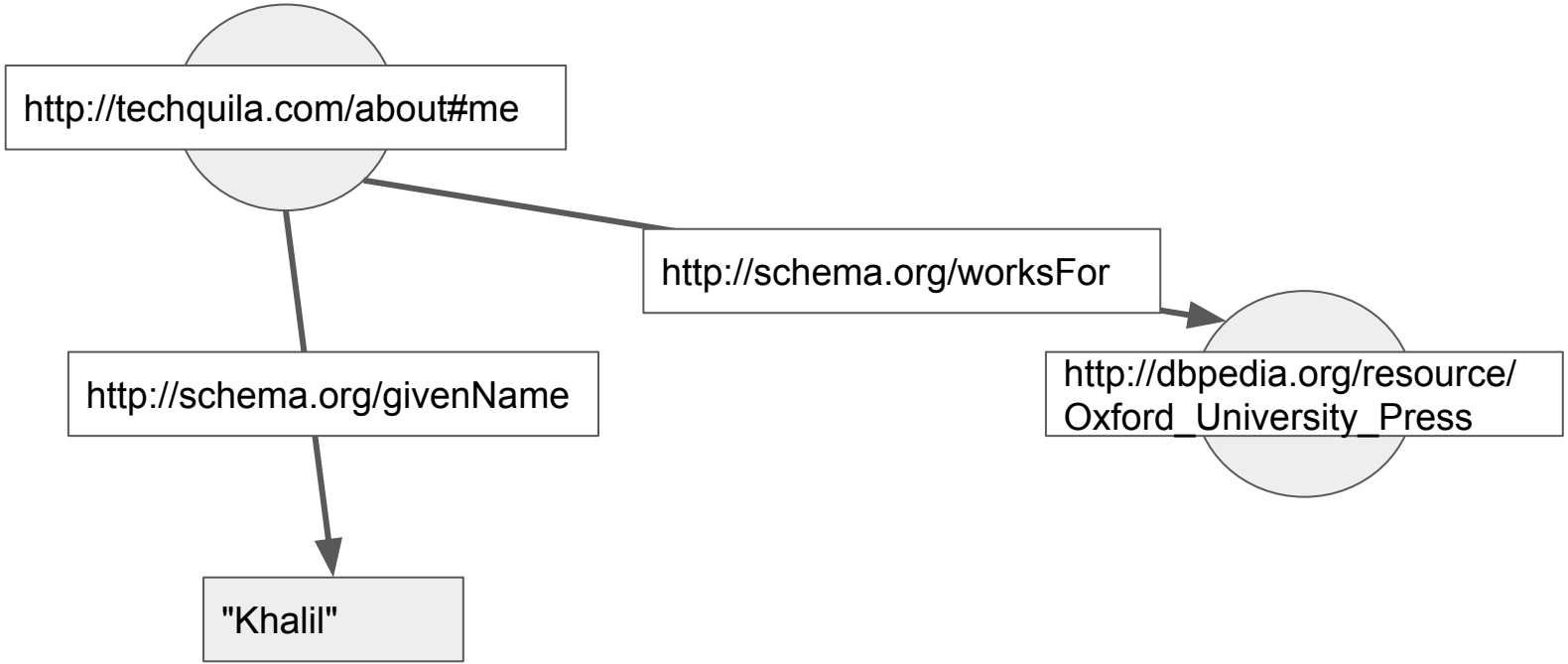


GraphDB vs NoSQL vs RDBMS



Triple Stores

RDF Graphs


http://techquila.com/about#me
http://schema.org/worksFor
http://schema.org/givenName
http://dbpedia.org/resource/
Oxford_University_Press
"Khalil"

Triple Stores - Structure

Node ID	RDF Term

S	P	O

P	O	S

O	S	P

Triple Store - Query

- SPARQL
  - W3C standard
  - SQL-like syntax
  - Matching graph patterns with variable substitution
  - Implemented by all triple stores
  - A suite of standards, including Update as well as Query
  - Provides a number of built-in functions plus hooks for vendor-specific extensions

Query Examples - SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?p ?f WHERE {
  ?p foaf:knows ?f
}
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1>
SELECT ?p ?fof WHERE {
  ?p foaf:knows ?f .
  ?f foaf:knows ?fof .
  FILTER(!sameTerm(?p, ?fof))
}
```

Update

- Graph Store Protocol
  - DELETE - remove a graph
  - POST - insert into a graph
  - PUT - replace a graph
- SPARQL Update
  - Builds on Query syntax
  - Assert triples to add / delete rather than generate a result set

SPARQL Update

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>

DELETE { ?person foaf:givenName 'Bill' }
INSERT { ?person foaf:givenName 'William' }
WHERE  {
  ?person foaf:givenName 'Bill'
}
```

Reasoning

- RDFS/OWL
  - Declarative rules for reasoning about RDF data
  - Rules allow the inference of the existence of implicit triples (and query across them)
  - OWL reasoning is implemented by many of the commercial triple stores
  - RDFS reasoning is a (much) reduced subset - more widely implemented
- Useful when wrangling diverse datasets
  - Declare superclass-subclass relationships (RDFS)
  - Declare domain and range constraints for properties (RDFS)
  - Declare property and class equivalences
  - Cause things to merge based on unique properties
  - Detect contradictory statements
- Rules are just data
  - Can be expressed in RDF and managed in a triple store

Reasoning Example

```
:bart :hasParent :homer .
:homer :hasParent :grampa
```

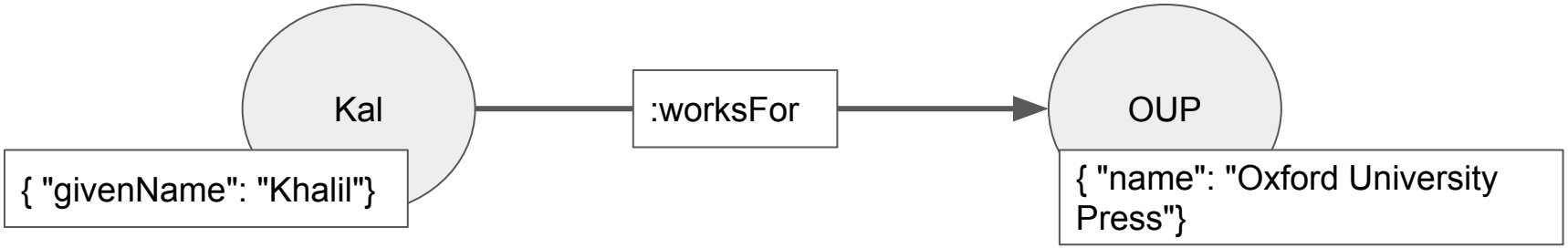
```
:hasGrandparent owl:propertyChainAxiom (:hasParent :hasParent)
```

```
SELECT ?gc WHERE {
  ?gc :hasGrandparent ?gp
}
```

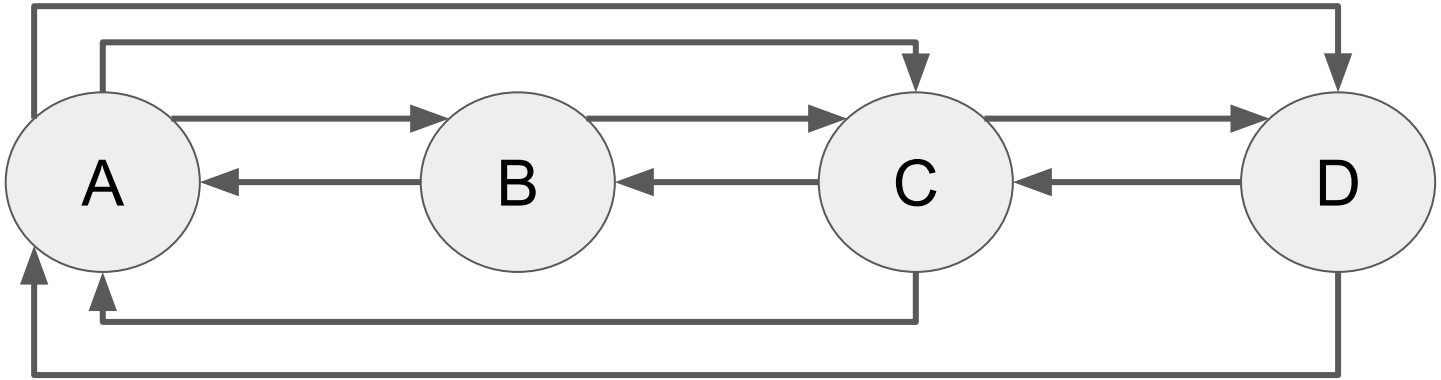
?gc	**?gp**
:bart	:grampa

Property Graph Databases

Property Graphs



Property Graph Databases - Structure



Property Graph DB vs Triple Store Indexing

Property Graph

- All the information required to traverse edges into/out of a node is stored on the node itself.
- Typically not everything is indexed
- Faster graph traversal

Triple Store

- Typically index each triple by subject, predicate and object
- Traversal is essentially a series of JOIN operations
- Typically everything is indexed

Property Graph Databases - Query

- Cypher
  - Declarative graph query language
  - Vague resemblance to SQL
- GraphQL
  - Declarative query language based
  - Pattern matching
- Gremlin
  - Procedural graph-traversal language
- Also rans
  - PGQL
  - GCore

Finding Friends - Cypher

```
MATCH (n:Person)-[:FRIEND]-(f)
RETURN n, f
```

```
MATCH (n:Person)-[:FRIEND]-()-[:FRIEND]-(fof)
RETURN n, fof
```

Finding Friends - GraphQL

```
{
  person {
    name
    friends {
      name
    }
  }
}
```

```
{
  person {
    name
    friends {
      name
      friends {
        name
      }
    }
  }
}
```

Finding Friends - Gremlin

```
g.V().hasLabel("person")
  .emit()
  .out("friend")
  .values("name")
  .fold()
```

```
g.V().hasLabel("person")
  .emit()
  repeat(out("knows")).times(2)
  .values("name")
  .dedup()
  .fold()
```

Inferencing

- No PG equivalent to OWL
- Write code
  - Server-side extensions
  - Client-side application code

Comparison

Speed / Scale

- Distributed implementations available of both
- PG is faster for graph traversal - O(1) vs O(log N)
- PG tend to require explicit indexing for fast property filtering
- Speed of update/retraction can be an issue for triple stores with OWL reasoning
- Either technology is capable of scaling up to billions of graph nodes and more

Maturity

- Technologies are roughly equivalent ages
- There are a small number of key commercial implementations of both
- There are a number of open-source implementations of both
- W3C standards stack changes more slowly
  - -ve: Innovation gets shut out for too long
  - +ve: Stability of query language, focus on backwards compatibility

Environment

- Both support multiple serialization formats
- Query:
  - PG: Fragmented with many vendor-specific languages
  - Triple Stores: SPARQL
- Triple Stores and RDF fit into a wider W3C Semantic Web Stack
  - Update (Graph Store Protocol, SPARQL Update)
  - Reasoning (OWL)
  - Data access (Linked Data Platform)
  - Wide variety of RDF vocabularies
- Stronger standards support give Triple Stores the edge for portability

In Practice...

- PG tends to be used where:
  - Data is faster moving
  - More regularity to the data (particularly node/edge properties)
- Triple stores are typically used where:
  - Greater variety in the data - e.g. integrating data from multiple sources;
  - Where RDF and other Linked Data technologies are important
  - Where OWL reasoning provides declarative data integration and query benefits

Best of Both Worlds?

- Model RDF in a Property Graph
  - Possibility to compress certain triples into PV pairs on nodes
  - Use edge labels to index predicate IRI
  - Use property labels to index node IRI
- Amazon Neptune
  - PG with SPARQL support
  - Only Turtle supported as an import format
  - Limited support for SPARQL Update
  - No support for Graph Store Protocol
- Gremlinator
  - Open-source plugin for Tinkerpop
  - Converts a subset of SPARQL queries into Gremlin
  - Just query (no support for import/export of RDF for example)