There's a Property Graph in my Triplestore:
Using an RDF Triplestore to Manage, Model, and (SPARQL) Query Property Graphs

Souripriya Das
Ana Estrada
Matthew Perry
Xavier Lopez

Oracle Spatial and Graph
August 21, 2014

SEMANTIC TECHNOLOGY & BUSINESS
10th ANNUAL
August 19-21, 2014 | San Jose, California USA

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Program Agenda

1. Graph databases
2. Property Graph vs. RDF
3. Property Graph as RDF
4. Experimental Evaluation
5. Discussion

ORACLE

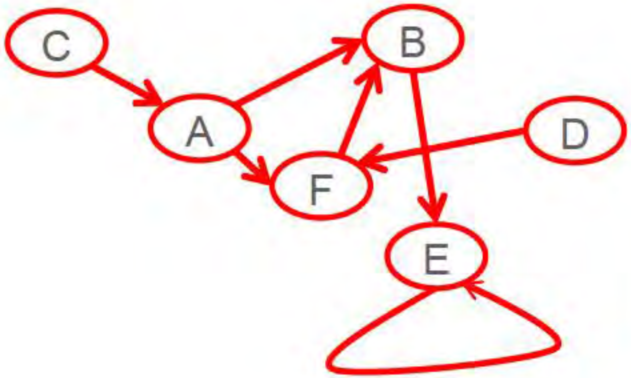
Program Agenda

1. Graph databases
2. Property Graph vs. RDF
3. Property Graph as RDF
4. Experimental Evaluation
5. Discussion

ORACLE

Graph Data Model

- What is a graph?
  - A set of vertices and edges (and optionally attributes)
  - A graph is simply **linked data**
- Why do we care?
  - Graphs are intuitive and flexible
    - Easy to navigate, easy to form a path, natural to visualize
  - Graphs are everywhere
    - Road networks, power grids, biological networks
    - Social networks/Social Web (Facebook, Linkedin, Twitter, Baidu, Google+,...)
    - Knowledge graphs (RDF, OWL)



ORACLE

Graph Database

- A **graph database** is a database that uses graph structures with nodes, edges, and properties to represent and store data.[1]

Property Graph

Neo4j

ORACLE DATABASE

*Sparksee by *Sparsity Technologies

OPENLINK SOFTWARE
Virtuoso Universal Server

AllegroGraph
Franz Inc.

RDF Graph

Ontotext
OWLIM

TITAN

Objectivity
InfiniteGraph

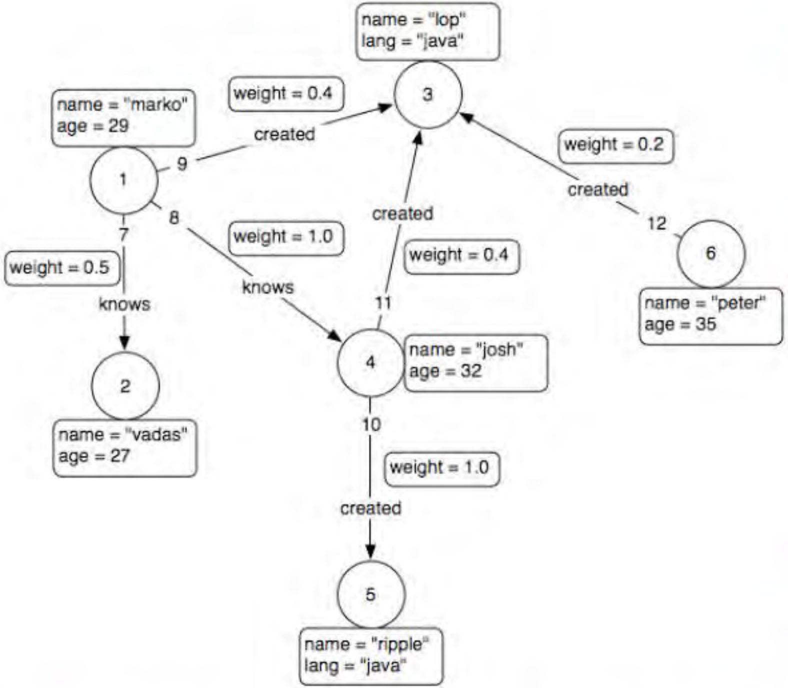
1. http://en.wikipedia.org/wiki/Graph_database

Program Agenda

1. Graph databases
2. Property Graph vs. RDF
3. Property Graph as RDF
4. Experimental Evaluation
5. Discussion

ORACLE

Property Graph Data Model

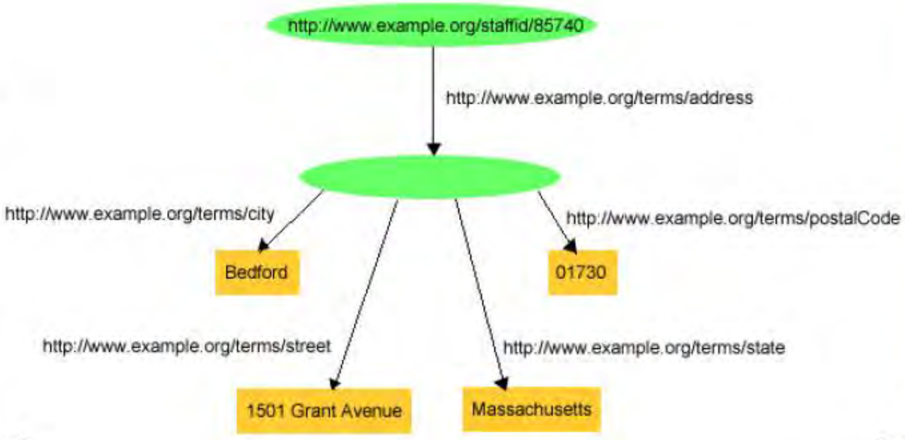


https://github.com/tinkerpop/blueprints/wiki/Property-Graph-Model

- A set of vertices (or nodes)
  - each vertex has a unique identifier.
  - each vertex has a set of in/out edges.
  - each vertex has a collection of **key-value** properties.
- A set of edges
  - each edge has a unique identifier.
  - each edge has a head/tail vertex.
  - each edge has a label denoting type of relationship between two vertices.
  - each edge has a collection of **key-value** properties.
- Blueprints Java APIs
- Implementations
  - Neo4j, InfiniteGraph, Dex, Sail, MongoDB …

RDF Graph

- Resource Description Framework
  - URIs are used to identify
    - Resources, entities, relationships, concepts
    - Data identification is a *must* for integration
- RDF Graph defines semantics
- Standards defined by W3C & OGC
  - RDF, RDFS, OWL, SKOS
  - SPARQL, RDFa, RDB2RDF, GeoSPARQL
- Implementations
  - Oracle, IBM, Cray, Bigdata ®
  - Franz, Ontotext, Openlink, Jena, Sesame, …



PG & RDF: Are They Really That Different?

Distinguishing features

	Property Graph	RDF Graph
KV Pairs on Edges	Easy	Non-trivial
Multi-valued Attributes	Non-trivial	Easy
Object Properties for Edges	Not possible	Non-trivial
Syntax Rules	None	Strict

ORACLE

Use Cases

Property Graph	RDF Graph
Use Cases	
▪ Graph analytics – Page rank, clustering, finding influencers, path traversal ▪ Flexible data model – Add/remove attributes over time ▪ Sparse data	▪ Data integration (URIs) – Metadata layer, linked data, data warehouse ▪ Logical Inference (Semantics) – Finding implicit relations, data/model verification ▪ Flexible data model – Add/remove attributes over time ▪ Sparse data

ORACLE

Pros and Cons

Property Graph

RDF Graph

Pros and Cons

Property Graph

- Pros
  - Simple
  - Strong analytics infrastructure
- Cons
  - No standardization (query, update)
  - Limited support for data integration/reuse
  - Lack of publicly available data

RDF Graph

- Pros
  - Rigorously defined
  - Suite of W3C standards (protocol, query, update, RDB2RDF, entailment)
  - Strong support for data integration/reuse
  - Lots of publicly available data
- Cons
  - Steep learning curve
  - Less infrastructure for analytics

ORACLE

Program Agenda

1. Graph databases
2. Property Graph vs. RDF
3. Property Graph as RDF
4. Experimental Evaluation
5. Discussion

ORACLE

Can we support Property Graph data model in RDF?

ORACLE

Why Model Property Graphs as RDF?

- Bring benefits of RDF to Property Graphs
  - Enable rules and inferencing using ontology standards like OWL or user-defined rules
  - Leverage existing standards (for interoperability)
- Utilize maturity of existing RDF implementations
- Combine / integrate property graph data with existing RDF data

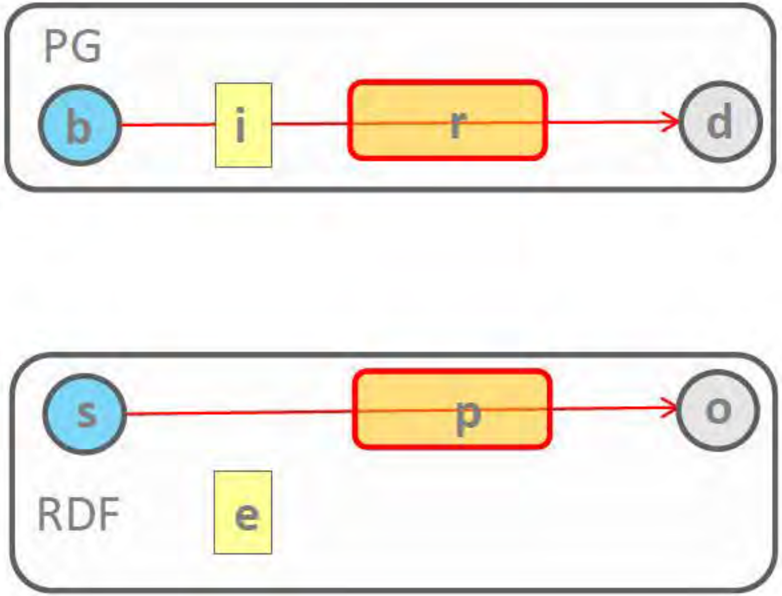
ORACLE

What's Needed to Support Property Graph Applications in a Triplestore

- KV pairs for edges
  - Can be done efficiently within the RDF standard
  - Focus of the rest of the presentation
    - Reification-based
    - Subproperty-based
    - Named graph-based
- Support for Graph Analytics
  - Many triplestores have proprietary solutions

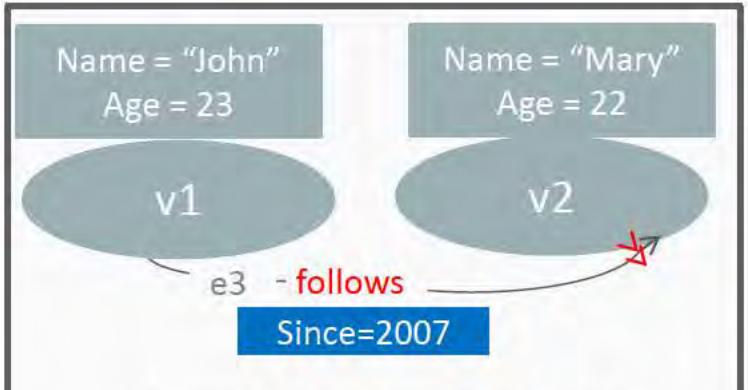
ORACLE

Mapping PG edges to RDF

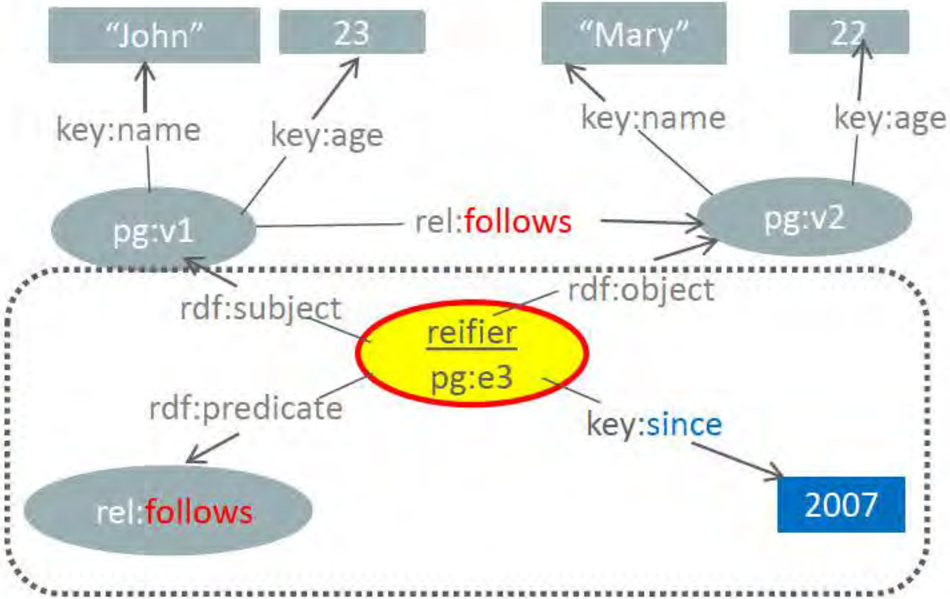


ORACLE

Reification-based Transformation



A Sample Property Graph

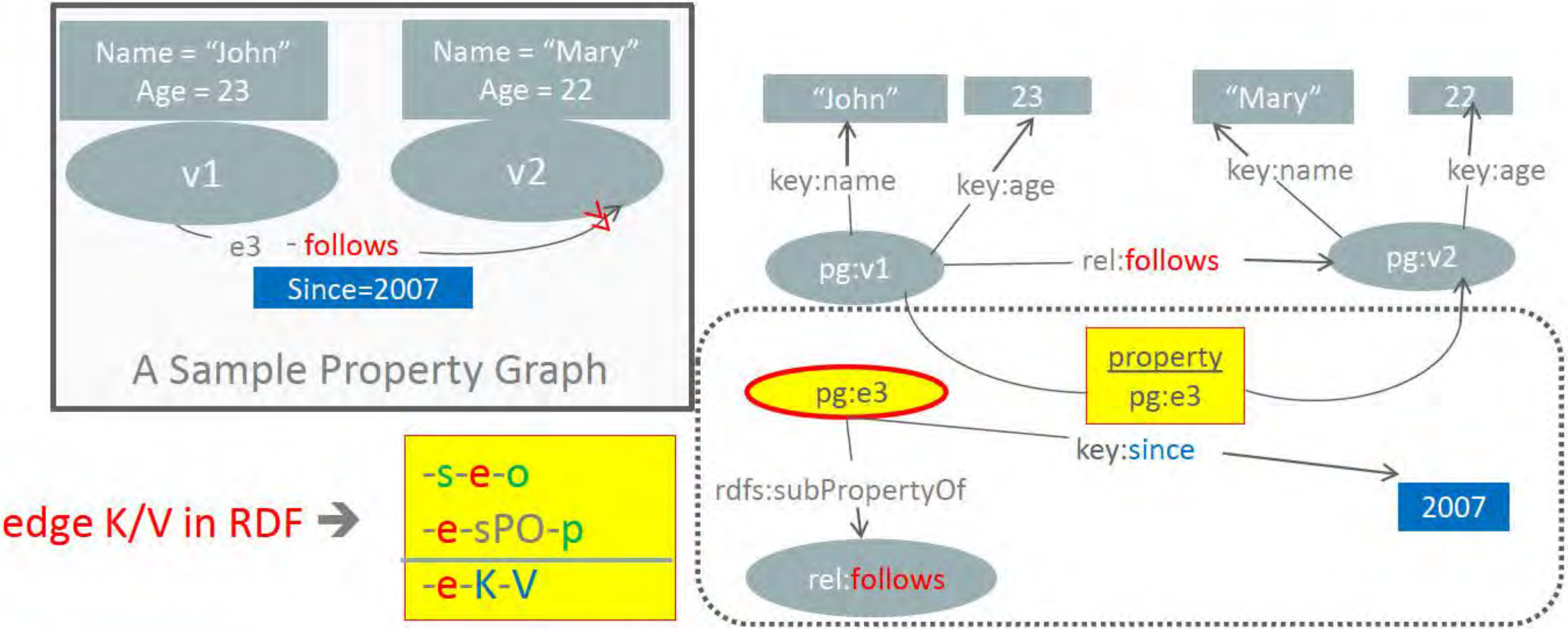


edge K/V in RDF →

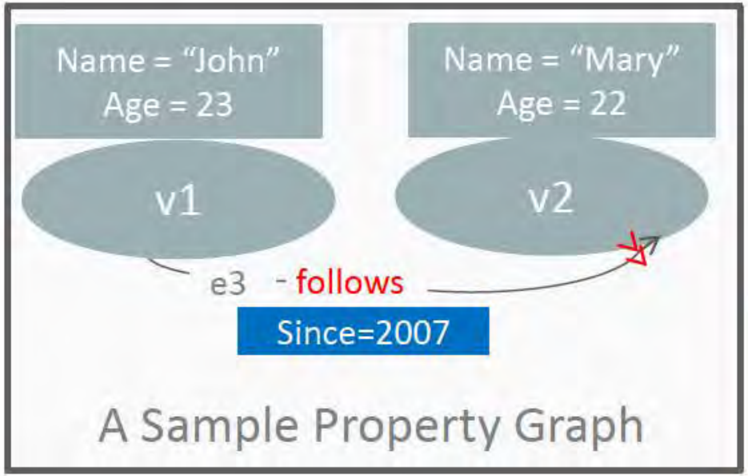
-e-sub-s
-e-pred-p
-e-obj-o

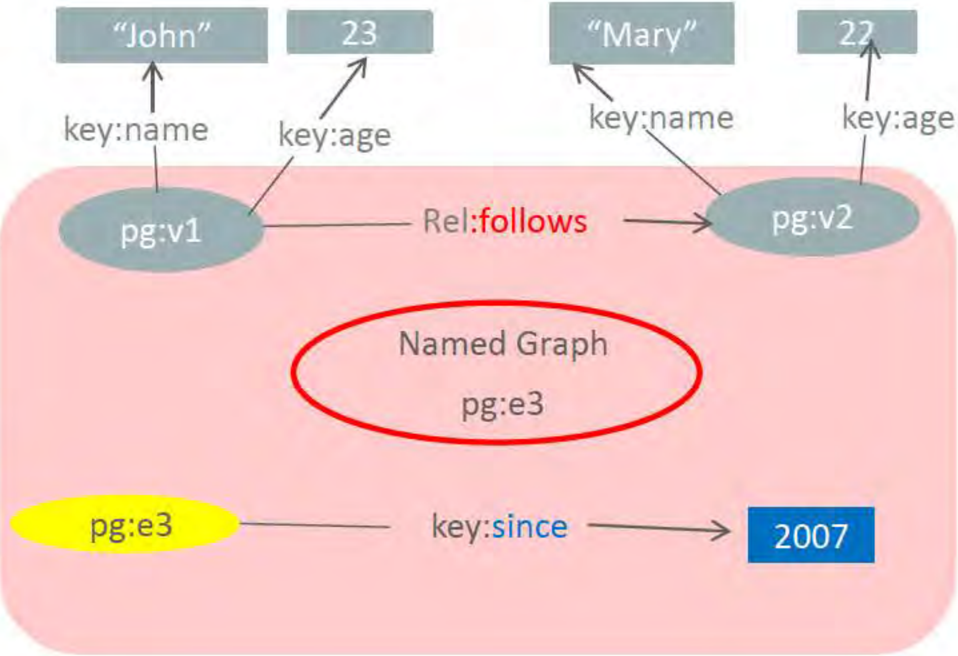
-e-K-V

RDF subProperty-based Transformation



Named Graph-based Transformation of Property Graphs to RDF





edge K/V in RDF →

e-s-p-o
e-e-K-V

ORACLE

Program Agenda

1. Graph databases
2. Property Graph vs. RDF
3. Property Graph as RDF
4. Experimental Evaluation
5. Discussion

ORACLE

Representing Edge K/V in RDF - which approach is best?

Implementation in Oracle

Supporting Property Graph as RDF

- Utilize existing Oracle RDF capabilities:
  - Triple/Quad format, bulk load, SPARQL / SEM_MATCH query, Indexing, Virtual Model
- Transform Property Graph into RDF and use SPARQL to query property graph

ORACLE

Experimental Evaluation

Goals and Experimental Setup

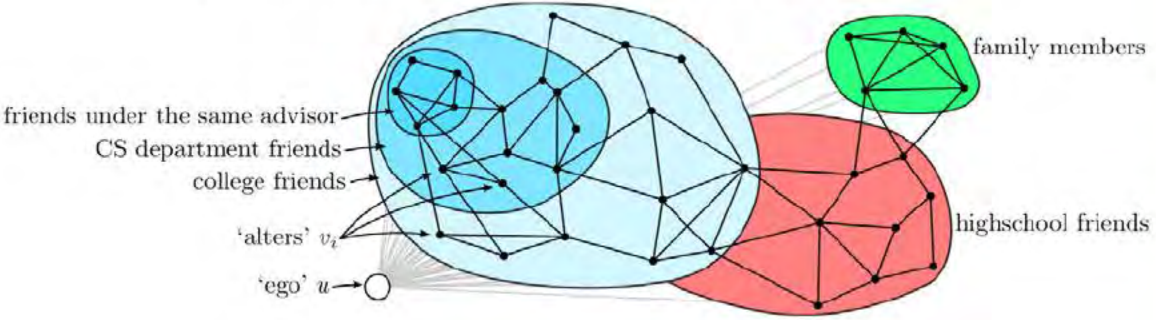
- Goals
  - Characterize query performance differences of NG and SP schemes
  - Demonstrate that query execution times are fast enough for interactive applications
- Experimental Setup
  - Lenovo ThinkPad T430 (dual-core Intel i5-3320M CPU, 8 GB RAM, 120 GB SSD)
  - Oracle Database 12c (12.1.0.1.0) running on 64-bit OEL 6
    - pga_aggregate_target=2G, sga_target=4G

ORACLE

Test Dataset:
Stanford Twitter Social Circles Dataset

- 973 ego networks
- For each edge *b follows c* in ego network *a*, generate *a knows b* and *a knows c*
- Node K/V pairs consist of *refs "@keyword"* and *hasTag "#tag"*
- Edge K/V pairs consist of the intersection of K/V pairs of start and end node.

J. McAuley and J. Leskovec. Learning to Discover Social Circles in Ego Networks. NIPS, 2012

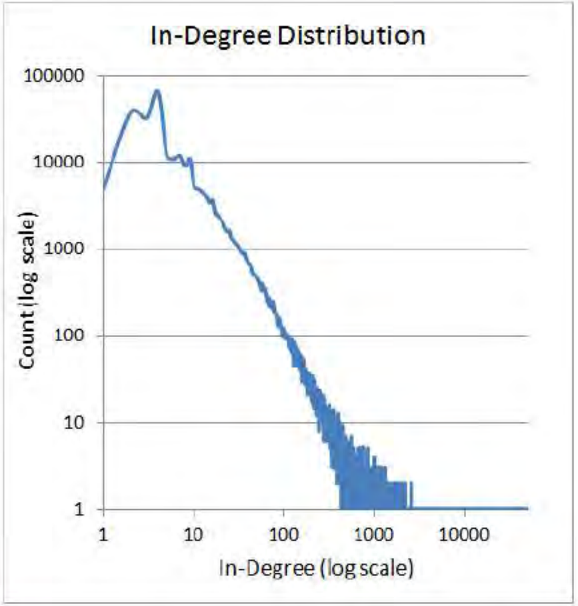


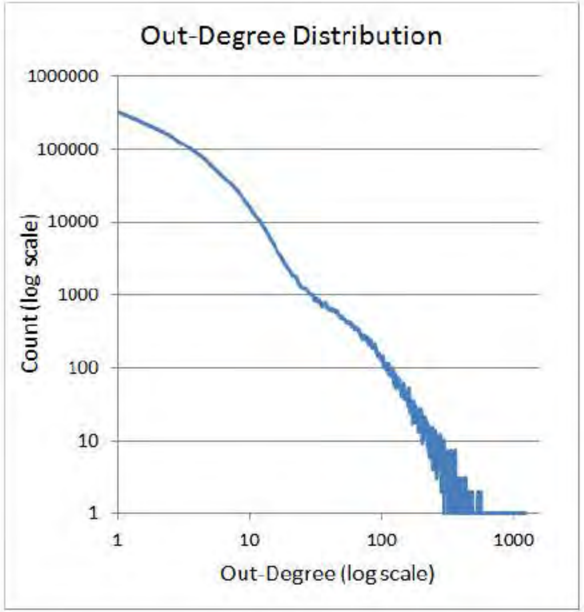
Twitter Dataset Characteristics

Nodes	Edges	Node K/Vs	Edge K/Vs
76,245	1,796,085	1,218,763	3,345,982

Test Dataset

Stanford Twitter Social Circles Dataset

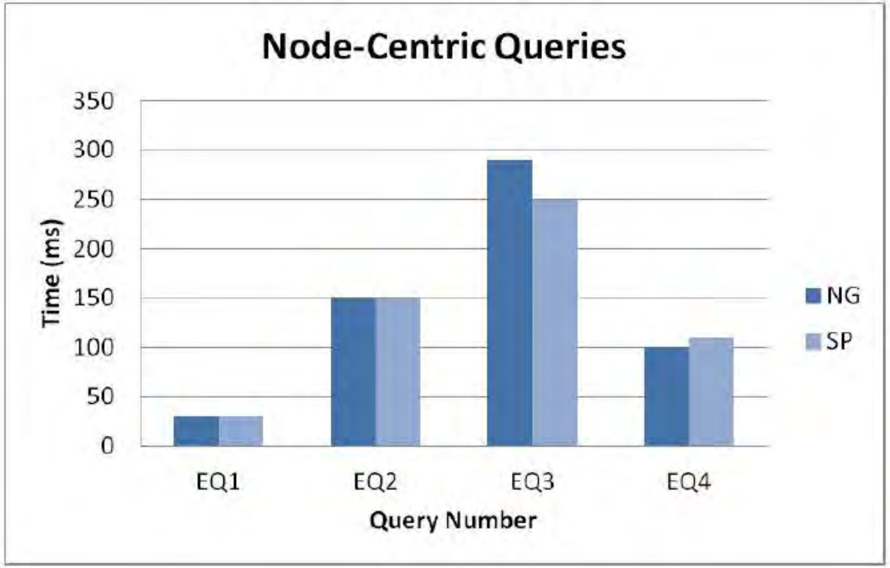




ORACLE

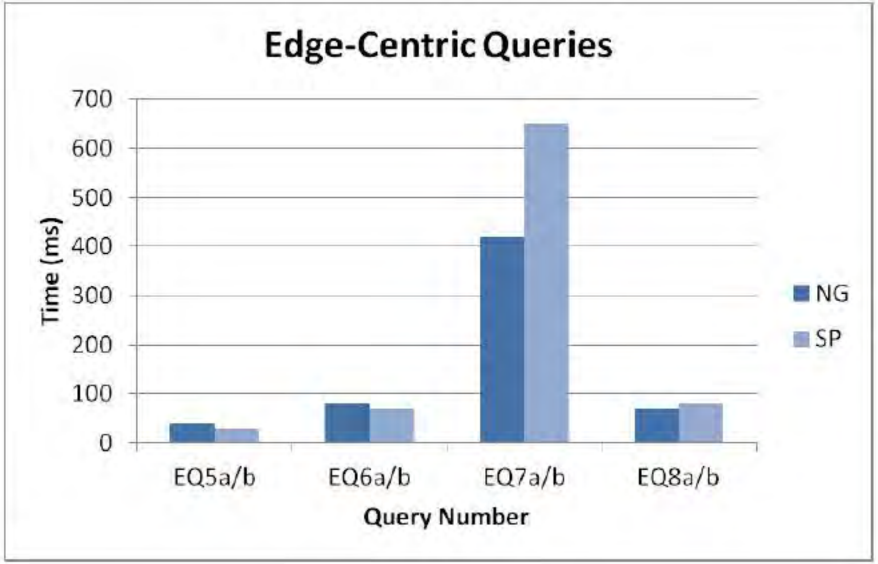
Experiment 1: Node Centric Queries

- **EQ1** – find all nodes/edges that have tag "#webseries" (251 results)
- **EQ2** – find all nodes that follow nodes with tag "#webseries" (1,249 results)
- **EQ3** – find all 3-hop paths where each node has tag "#webseries" (11,440 results)
- **EQ4** – find all key/value pairs for nodes/edges with tag "#webseries" (3,011 results)



Experiment 2: Edge Centric Queries

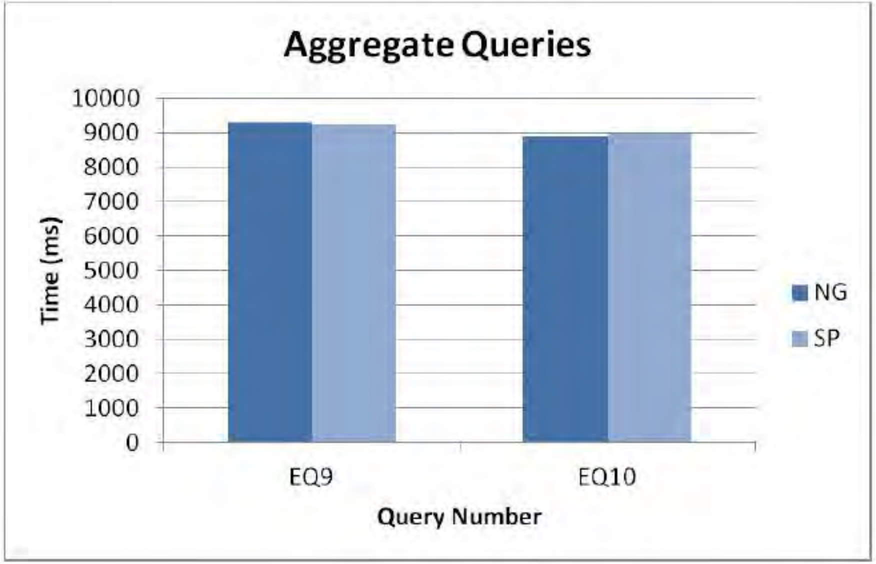
- **EQ5a/b** – find all edges with tag "#webseries" (206 results)
- **EQ6a/b** – find all nodes that are endpoints of an edge with tag "#webseries" and find all nodes that are followed by these nodes (13,012 results)
- **EQ7a/b** – find all 3-hop paths where each edge has tag "#webseries" (11,440 results)
- **EQ8a/b** – find all key/value pairs for edges with tag "#webseries" (1,269 results)



ORACLE

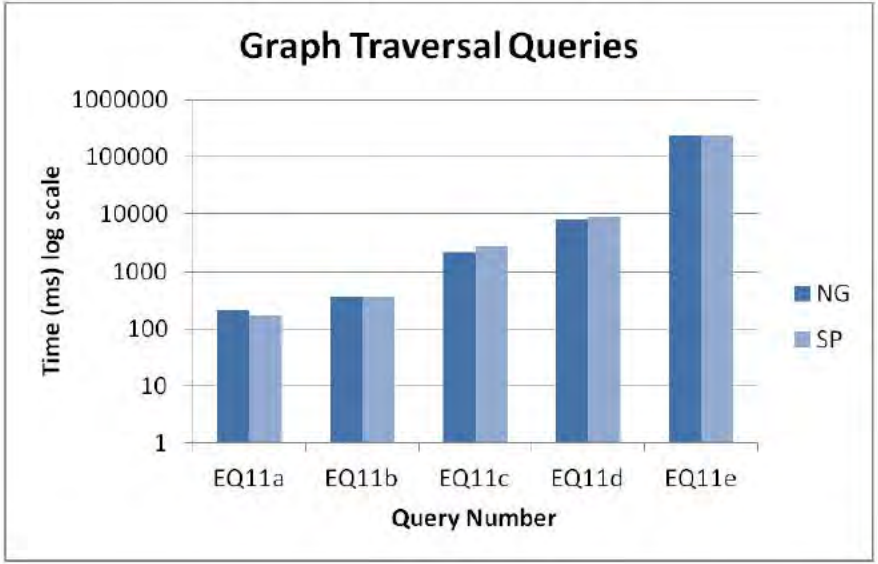
Experiment 3: Aggregate Queries

- **EQ9** – find the distribution of node in-degree
- **EQ10** – find the distribution of node out-degree



Experiment 4: Path Traversal Queries

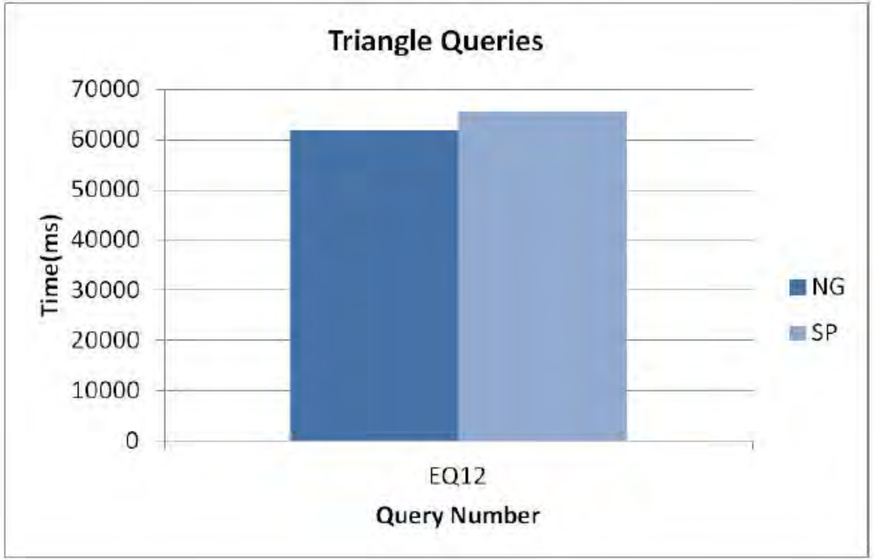
- **EQ11a** – find all 1-hop follows paths from a start node (21 results)
- **EQ11b** – find all 2-hop follows paths from a start node (900 results)
- **EQ11c** – find all 3-hop follows paths from a start node (52,540 results)
- **EQ11d** – find all 4-hop follows paths from a start node (3,573,916 results)
- **EQ11e** – find all 5-hop follows paths from a start node (257,861,728 results)



ORACLE

Experiment 5: Triangle Counting

- **EQ12** – count all follows triangles (20,211,887 results)



Summary of Results

- Interactive query times are possible for test hardware and dataset
- SP and NG provide similar performance on many queries
- Only significant difference occurs when accessing edge K/V pairs
  - NG performs better due to fewer joins
- NG approach performs slightly better for path and triangle queries due to smaller triples table size

ORACLE

Program Agenda

ORACLE

Discussion

Path Queries in SPARQL 1.1

- SPARQL is mainly intended for subgraph matching not path traversal
- SPARQL 1.1 Property Paths test connectivity of arbitrary-length paths
  - There is a need for improved path search capabilities to fully compete with NoSQL graph databases
- Several Extensions
  - SPARQLeR, SPARQ2L, G-SPARQL

Conclusions

- Property Graph data (including edge K/V pairs) can be modeled using standard RDF
- Many benefits:
  - Maturity of software infrastructure
  - Standard query and update language
  - Enhanced data integration possibilities
  - Potential for OWL-based logical inference
- Demonstrated feasibility with an implementation using Oracle 12c Spatial and Graph
- More Information:

  Souripriya Das, Jagannathan Srinivasan, Matthew Perry, Eugene Inseok Chong, Jayanta Banerjee: ***A Tale of Two Graphs: Property Graphs as RDF in Oracle***. EDBT 2014: 762-773

ORACLE

Hardware and Software

Engineered to Work Together

Appendix

Cardinalities of RDF triples or quads and terms

Table 2. Property graph vs. RDF cardinalities

Property Graph cardinalities			
E edges (*E*^{1} of them with >=1 edge-KVs), *V* vertices, *eKV* edge-KVs, *nKV* node-KVs, *eL* distinct edge-labels (rel. types), *eK* distinct keys for edge-KVs, and *nK* distinct keys for node-KVs			
RDF cardinalities for models	RF	NG	SP
Named Graphs	*0*	*E*	*0*
Obj-prop triples/quads	*4 * E*	*E (quads)*	*3 * E*
Data-prop triples (quads in NG)	*eKV+nKV*		
Distinct sub/obj count	*V+E*	*V+E*^{1}	*V+E*
Distinct obj-properties	*eL+3*	*eL*	*eL+E+1*
Distinct data-properties	*Distinct (eK UNION nK)*		

ORACLE

Partitioned Storage

Create separate semantic model

- Queries may access specific forms of RDF triples / quads
  - Partitioned into edge quads/triples partition, node K/V triples partition, edge K/V triples partition
- Each partition can be created as separate semantic model with its local indexes.
- Virtual Model is used to access more than one partition

ORACLE

Partitioned Storage

Storage overhead

- No storage overhead for Node Attributes for both NG and SP models
- Storage overhead for Edge Attributes for SP model
  - SP model needs anchor triple –e-subPropertyOf-p to associate edge with its attributes
  - Proportional to number of edges
  - Mitigated by compression and partitioning
    - topology info: compressed on (:e :sPO)
    - edge attributes: compressed on (:e :K)
    - node attributes: compressed on (:n :K)

ORACLE

Query Types: NG vs. SP

Query Type	Forms of quads/triples to be queried	
	NG	**SP**
edge traversal, no edge-KV	e-s-p-o	-s-p-o
edge + edge-KV	e-s-p-o e-e-K-V	-s-e-o -e-sPO-p -e-K-V
Node-KV	-n-K-V	-n-K-V

- Edge traversal only can be queried in single partition
- Edge and edge attributes can be queried in virtual model for NG; in single partition for SP

ORACLE

Indexing

Speed up querying

- **Indexes are ID-based**
- **Allow any permutation of S, P, C, G, M** (C: Canonical Object)
- **Pre-built by default:** PCSGM and PSCGM

ORACLE

Indexing

Index usage for Property Graph queries

- Edge traversal without filter on edge attributes:
  - Indexes PCSGM and PSCGM may be used
  - only topology partition will be accessed (e.g. Get triangles (three edge cycles) of "follows" edges).
- Edge traversal with filter on edge attributes:
  - Anchor triple <:e :sPO :p> needs to be used to get edge attributes
  - Indexes PCSGM and GSPCM or PCSGM and SCPGM may be used
  - only edge KV partition is accessed (e.g. Get vertex pairs and *all KVs of edges* with "follows" label).
- Vertex with attributes filter:
  - No changes are required for filtering vertex attributes
  - Indexes PCSGM and SCPGM may be used
  - topology and vertex property partitions are accessed (e.g. Get *all KVs of vertices* matching a given KV: name = "Amy")

ORACLE

Query Details: EQ1 – EQ6

EQ1	`SELECT ?n` `WHERE { ?n k:hasTag "#webseries" }`	251
EQ2	`SELECT ?nf` `WHERE { ?n k:hasTag "#webseries" . ?nf r:follows ?n }`	1,249

EQ3	`SELECT ?n4`
	`WHERE { ?n k:hasTag ?t . ?n r:follows ?n2 . ?n2 k:hasTag ?t .`
	`?n2 r:follows ?n3 . ?n3 k:hasTag ?t . ?n3 r:follows ?n4 .`
	`?n4 k:hasTag ?t FILTER (?t = "#webseries") }`
		11,440

EQ4	`SELECT ?n ?k ?v` `WHERE { ?n k:hasTag "#webseries" . ?n ?k ?v FILTER (isLiteral(?v)) }`	3,011
EQ5a	`SELECT ?n2` `WHERE { GRAPH ?g1 { ?n r:follows ?n2 . ?g1 k:hasTag "#webseries" } }`	206

EQ5b	`SELECT ?n2`
	`WHERE { ?s ?p ?n2 . ?p rdfs:subPropertyOf r:follows . ?p k:hasTag`
	`"#webseries" }`
		206

EQ6a	`SELECT ?n3`
	`WHERE { GRAPH ?g1 { ?n r:follows ?n2 . ?g1 k:hasTag "#webseries" }`
	`?n2 r:follows ?n3 }`
		13,012

EQ6b	`SELECT ?n3`
	`WHERE { ?s ?p ?n2 . ?p rdfs:subPropertyOf r:follows .`
	`?p k:hasTag "#webseries" . ?n2 r:follows ?n3 }`
		13,012

ORACLE

Query Details: EQ7– EQ8

EQ7a	`SELECT ?n4`
	`WHERE { GRAPH ?g1 { ?n  r:follows ?n2 . ?g1 k:hasTag "#webseries" }`
	`GRAPH ?g2 { ?n2 r:follows ?n3 . ?g2 k:hasTag "#webseries"  }`
	`GRAPH ?g3 { ?n3 r:follows ?n4 . ?g3 k:hasTag "#webseries"  } }`
		11,440

EQ7b	`SELECT ?n4`
	`WHERE { ?s ?p ?n2 . ?p rdfs:subPropertyOf r:follows . ?p k:hasTag "#webseries" .`
	`?n2 ?p2 ?n3 . ?p2 rdfs:subPropertyOf r:follows . ?p2 k:hasTag "#webseries" .`
	`?n3 ?p3 ?n4 . ?p3 rdfs:subPropertyOf r:follows . ?p3 k:hasTag "#webseries" }`
		11,440

EQ8a	`SELECT ?n2 ?k ?v`
	`WHERE { GRAPH ?g1 { ?n  r:follows ?n2 . ?g1 k:hasTag "#webseries" .`
	`?g1 ?k ?v  FILTER (isLiteral(?v)) } }`
		1,269

EQ8b	`SELECT ?n2 ?k ?v`
	`WHERE { ?s ?p ?n2 . ?p rdfs:subPropertyOf r:follows .`
	`?p k:hasTag "#webseries" . ?p ?k ?v FILTER (isLiteral(?v)) }`
		1,269

Query Details EQ9 – EQ12

EQ9	`SELECT ?inDeg (COUNT(*) as ?cnt) WHERE { SELECT ?n2 (COUNT(*) as ?inDeg) WHERE { ?n1 (r:knows|r:follows) ?n2 } GROUP BY ?n2 } GROUP BY ?inDeg ORDER BY DESC(?inDeg)`	580
EQ10	`SELECT ?outDeg (COUNT(*) as ?cnt) WHERE { SELECT ?n1 (COUNT(*) as ?outDeg) WHERE { ?n1 (r:knows|r:follows) ?n2 } GROUP BY ?n1 } GROUP BY ?outDeg ORDER BY DESC(?outDeg)`	412
EQ11a	`SELECT (COUNT(?y) as ?cnt) WHERE {<http://pg/n6160742> r:follows ?y }`	21
EQ11b	`SELECT (COUNT(?y) as ?cnt) WHERE {<http://pg/n6160742> r:follows/r:follows ?y }`	900
EQ11c	`SELECT (COUNT(?y) as ?cnt) WHERE {<http://pg/n6160742> r:follows/r:follows/r:follows ?y }`	52,540
EQ11d	`SELECT (COUNT(?y) as ?cnt) WHERE {<http://pg/n6160742> r:follows/r:follows/r:follows/r:follows ?y }`	3,573,916
EQ11e	`SELECT (COUNT(?y) as ?cnt) WHERE {http://pg/n6160742 r:follows/r:follows/r:follows/r:follows/r:follows ?y}`	257,861,728
EQ12	`SELECT (COUNT(*) AS ?cnt) WHERE { ?x r:follows ?y . ?y r:follows ?z . ?z r:follows ?x }`	20,211,887

Mapping PG vertices/labels/keys/values to RDF

Edges

StartVertex	Edge	Label	EndVertex
1	3	follows	2
1	4	knows	2

ObjKVs

ObjId	Key	Type	Value
1	name	VARCHAR	Amy
1	age	NUMBER	23
...	...	...	...
3	since	NUMBER	2007

Figure 3. A sample property graph in relational format.

Property graph	RDF IRIs
vertex id v	pg:v
edge id e	pg:e
Edge label l	rel:l
key k	key:k

Property graph	RDF literals
Amy (varchar)	"Amy"
23 (integer)	"23"^^xsd:integer
2007 (integer)	"2007"^^xsd:integer

SPARQL Query over PG-as-RDF data

Edge, but NOT edge K/V

Q1. Get triangles (three edge cycles) of "follows" edges	
All	{?x rel:follows ?y . ?y rel:follows ?z . ?z rel:follows ?x}

Edge K/V

Q2. Get vertex pairs and *all KVs of edges* with "follows" label	
RF	{?e rdf:subject ?x; rdf:predicate rel:follows; rdf:object ?y . ?e ?k ?V}
NG	{GRAPH ?e {?x rel:follows ?y . ?e ?k ?V}}
SP	{?x ?e ?y . ?e rdfs:subPropertyOf rel:follows . ?e ?k ?V}

Node K/V

Q3. Get *all KVs of vertices* matching a given KV: name = "Amy"	
All	{?x key:name "Amy" . ?x ?k ?V FILTER isLiteral(?V)}

ORACLE

Characteristics of Converted RDF Datasets

NG vs. SP

Transformed RDF data characteristics: number of triples

	Edges		K/Vs	
	follows	knows	refs	hasTag
Triples (core)	1,667,885	128,200	3,771,755	792,990
Total Quads (NG)	6,360,830 (core = e-s-p-o)			
Total Triples (SP)	9,953,000 (core + e-sPO-p + s-e-o)			

Physical storage characteristics

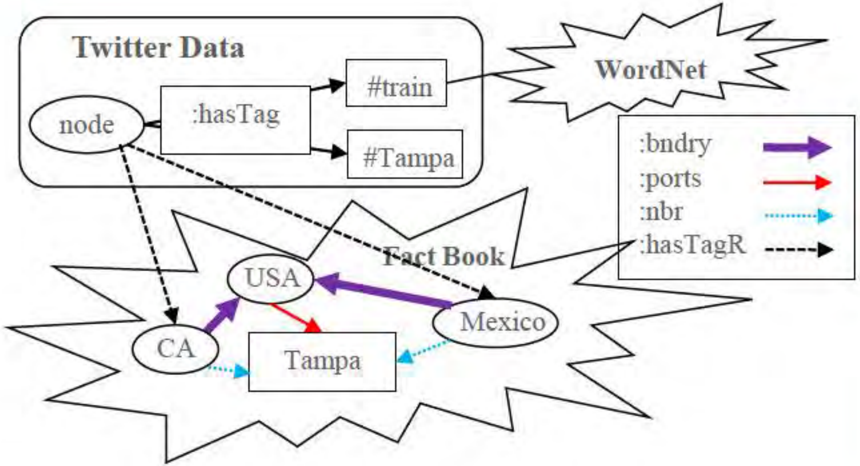
DB Object	Size (MB)	
	NG	SP
Triples Table	248	329
Values Table	56	57
PCSGM Idx	259	398
PSCGM Idx	338	504
GPSCM Idx	366	NA
SPCGM Idx	358	506
Total	1,625	1,794

ORACLE

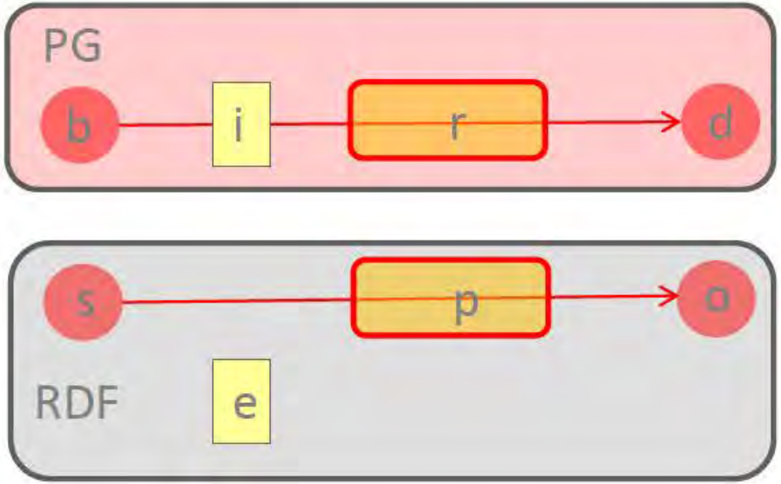
Discussion

Benefits of Modeling Property Graphs using RDF

- Easier to link to exiting domain ontologies and utilize inference capabilities
- Enable semantically rich applications
  - e.g. Linking Twitter Data with WordNet – enable query term expansion
  - e.g. Linking Twitter Data with World Fact Book - use of Inference



Mapping PG edges to RDF



edge-IRI **e** is associated using:

- Reification → -e-sub-s, -e-pred-p,-e-obj-o
- Sub-property → -s-e-o, -e-sPO-p
- Named Graph → -e-s-p-o

Table 1. RDF representation for three models

PG-as-RDF model	RDF quads/triples for PG element type		
	Topology edge	EdgeKV	NodeKV
RF	-e-rdf:subject-s -e-rdf:predicate-p -e-rdf:object-o -s-p-o	-e-K-V	-n-K-V
NG	e-s-p-o	e-e-K-V	-n-K-V
SP	-s-e-o -e-rdfs:subPropertyOf-p -s-p-o	-e-K-V	-n-K-V

ORACLE

