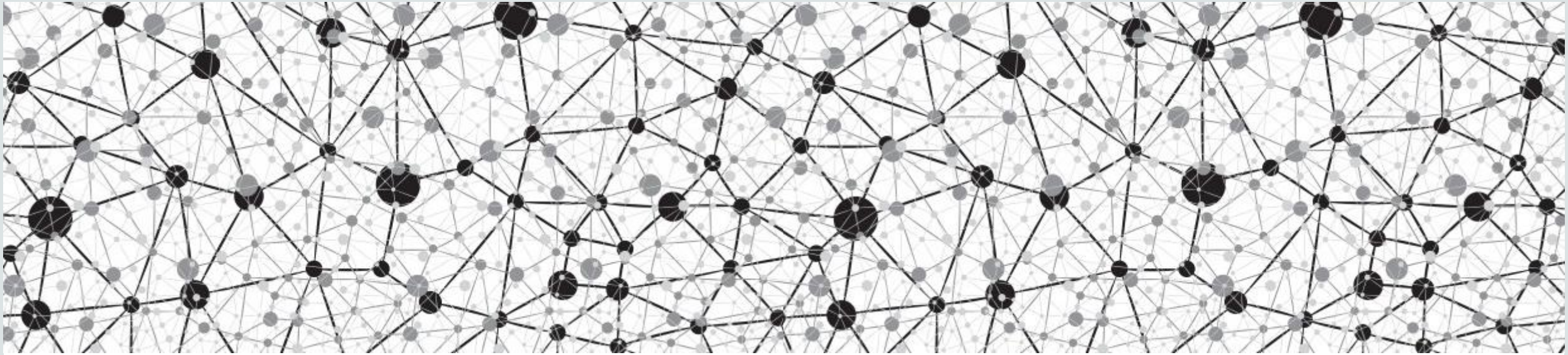


Analyzing a social network using Big Data Spatial and Graph Property Graph

Oskar van Rest
Principal Member of Technical Staff

Gabriela Montiel-Moreno
Principal Member of Technical Staff

BIWA SIG
Better Information Better Results
www.oraclebiwa.org

ORACLE

Safe Harbor Statement

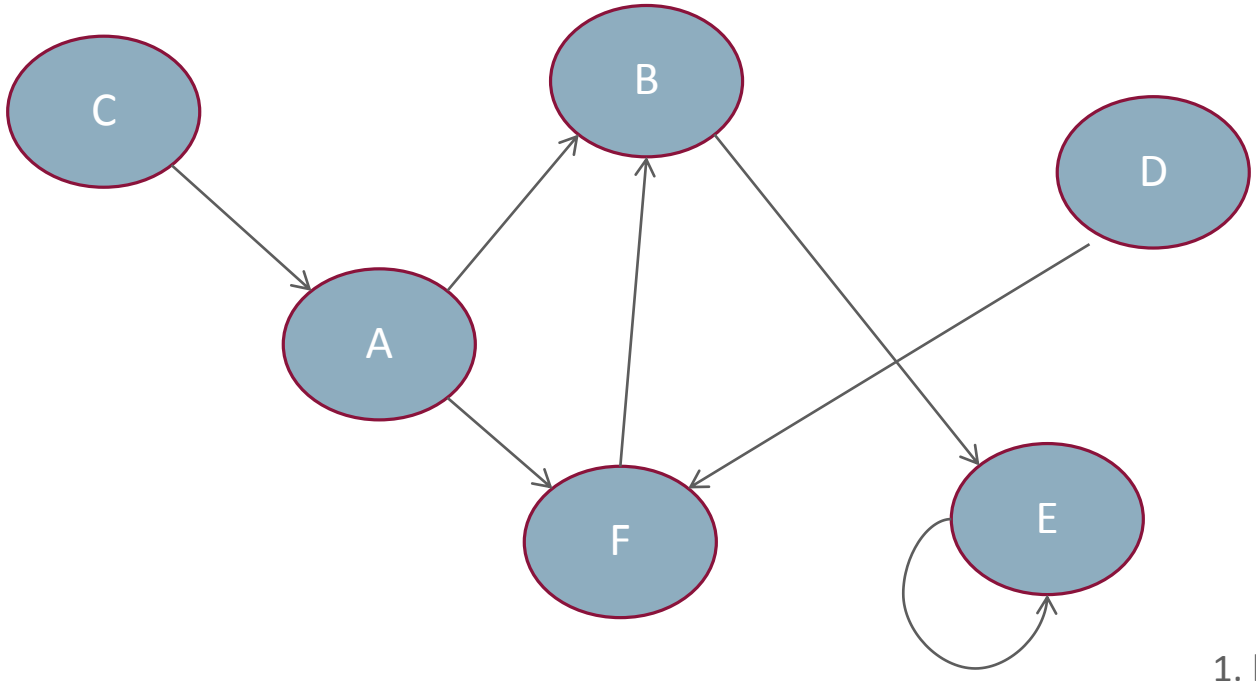
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development,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

Program Agenda

1. Introduction
2. Property Graph Data Model & BDSG Architecture
3. Oracle Big Data Spatial and Graph Core Features
4. Graph Analytics using PGX Graph Analytics Engine
5. HoL: Analyzing a social network using Property Graphs

Graph Database Definition

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

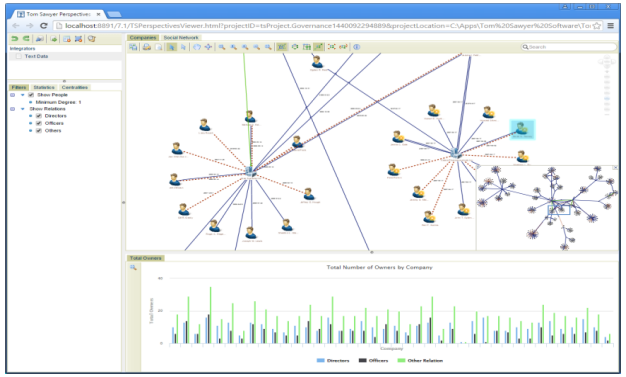


1. http://en.wikipedia.org/wiki/Graph_database

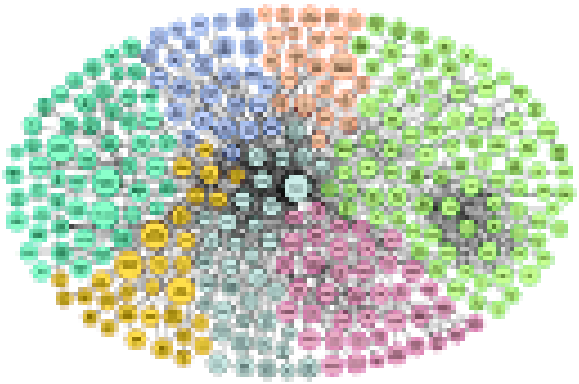
Why do we care?

Graphs are intuitive and flexible

- Easy to navigate, easy to form a path, natural to visualize

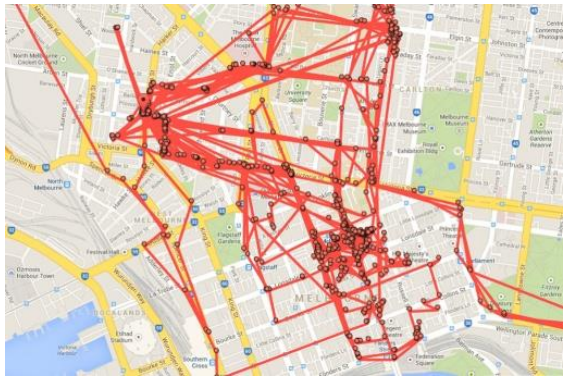


Enables views and queries that would be expensive on other databases



Graphs are everywhere

- Road networks, power grids, biological networks
- Social Networks
- Knowledge graphs (RDF, OWL)



Graph Use Case Scenarios

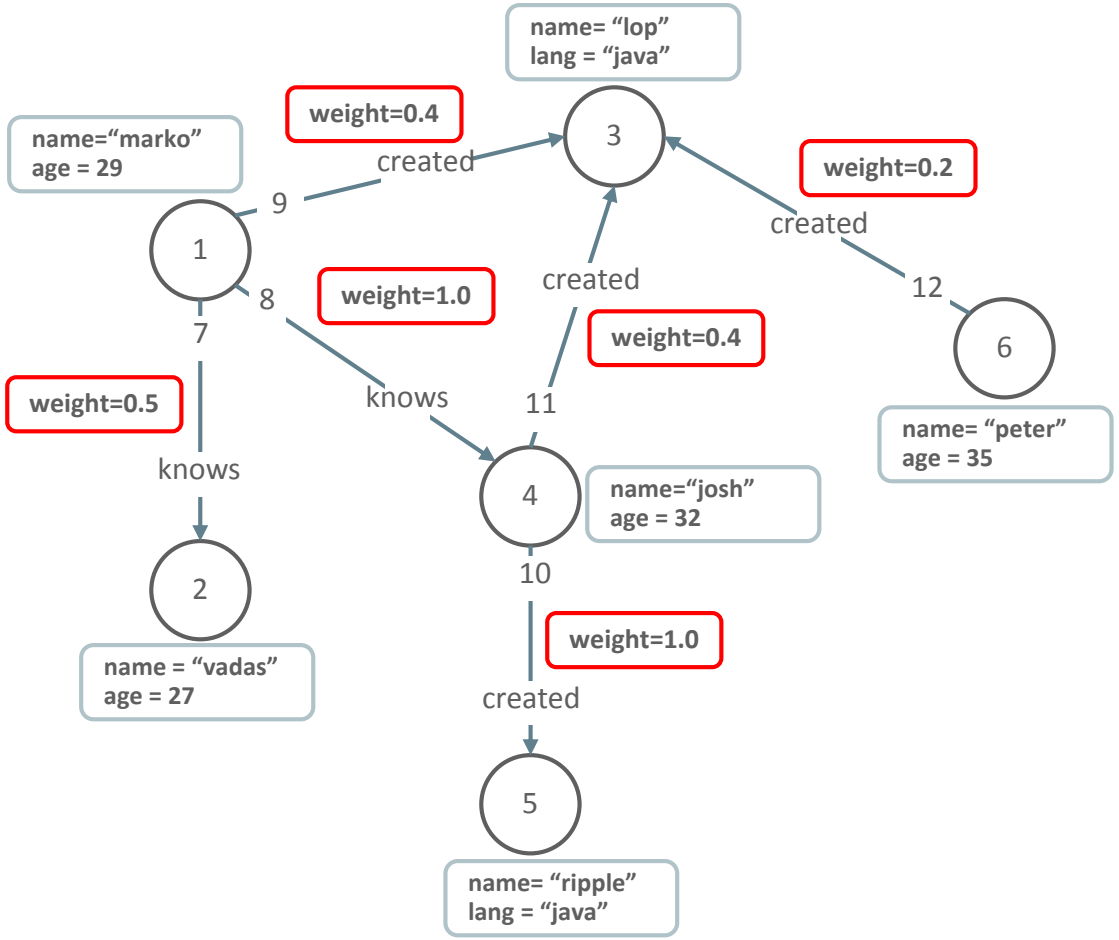
- Fraud detection
  - Find parties in insurance data who are on both sides of multiple claims, who live near each other
- Internet of Things
  - Manage graph of interconnected devices and predict the effect of an disruptions across network
- Cyber Security
  - Find entry points and affected machines
- Border Control
  - Analyze flight histories of a suspicious passenger. Indentify his co-travelers, co-traveler's co-travelers, …

ORACLE

Program Agenda

1. Introduction
2. Property Graph Data Model & BDSG Architecture
3. Oracle Big Data Spatial and Graph Core Features
4. Graph Analytics using PGX Graph Analytics Engine
5. HoL: Analyzing a social network using Property Graphs

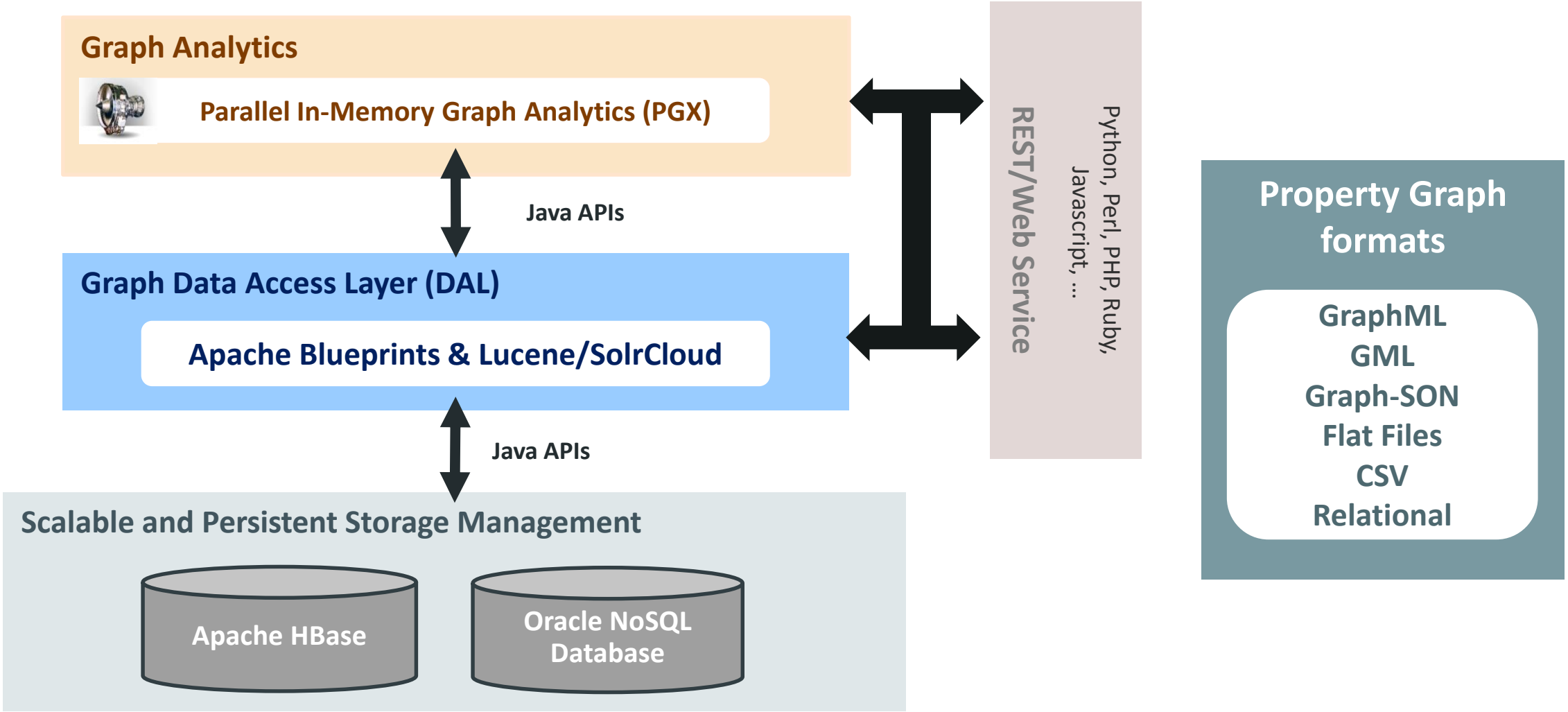
Property Graph Data Model



(example from https://github.com/tinkerpop/gremlin/wiki/Defining-a-Property-Graph)

- **A set of vertices**
  - each vertex has a unique identifier
  - each vertex has a set of outgoing/incoming 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 that denotes the type of relationship between two vertices
  - **each edge has a collection of key-value properties**
- Blueprints Java APIs
- Implementations
  - Neo4j, Titan, InfiniteGraph, Dex, Sail, MongoDB …

Oracle Big Data Spatial and Graph Architecture



Program Agenda

1. Introduction
2. Property Graph Data Model & BDSG Architecture
3. Oracle Big Data Spatial and Graph Core Features
4. Graph Analytics using PGX Graph Analytics Engine
5. HoL: Analyzing a social network using Property Graphs

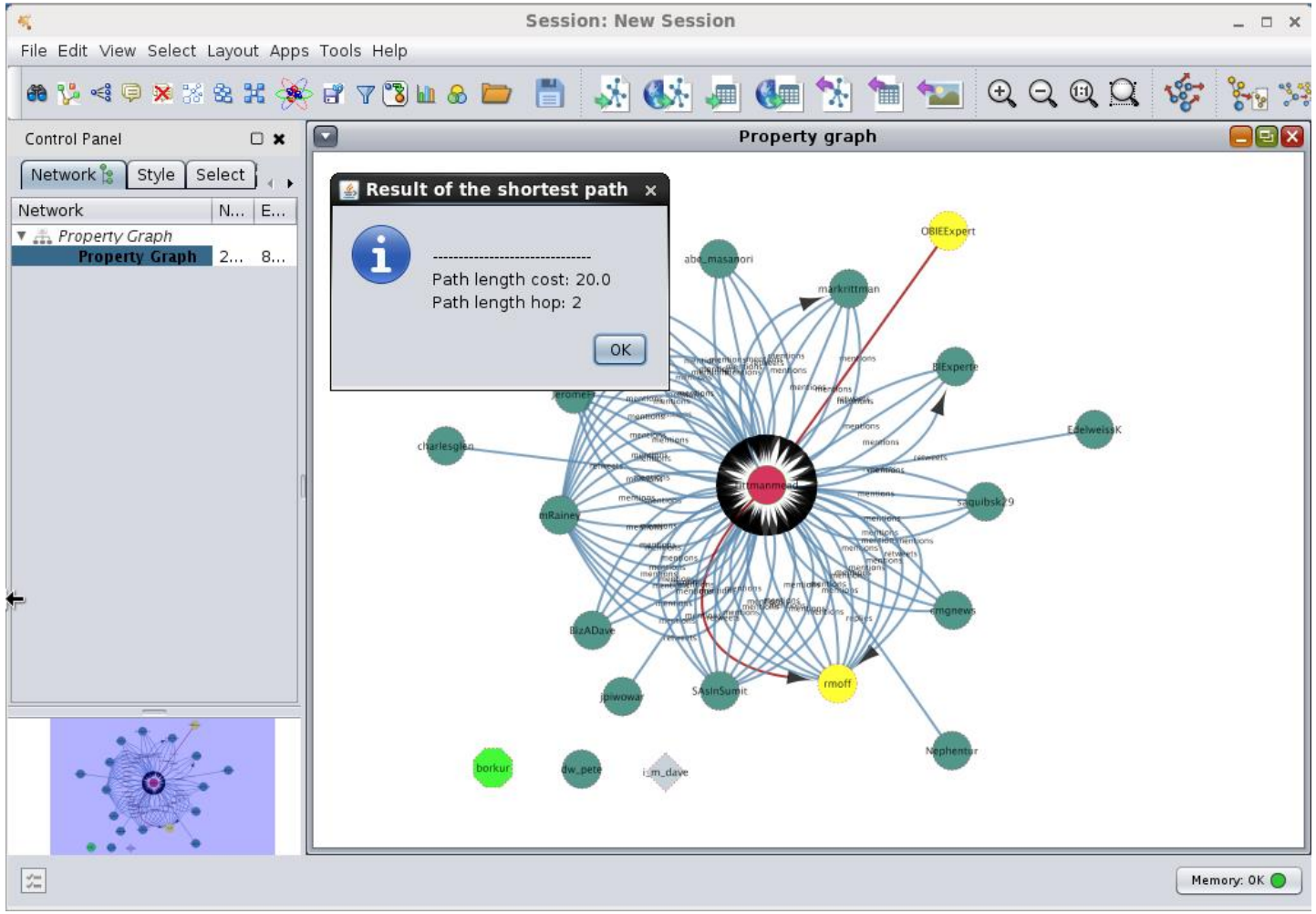
Data Access (APIs)

- Blueprints 2.3.0, Gremlin 2.3.0, Rexster 2.3.0
- Groovy shell for accessing property graph data
- REST APIs (through Rexster integration)
- PGQL (Property Graph Query Languge)

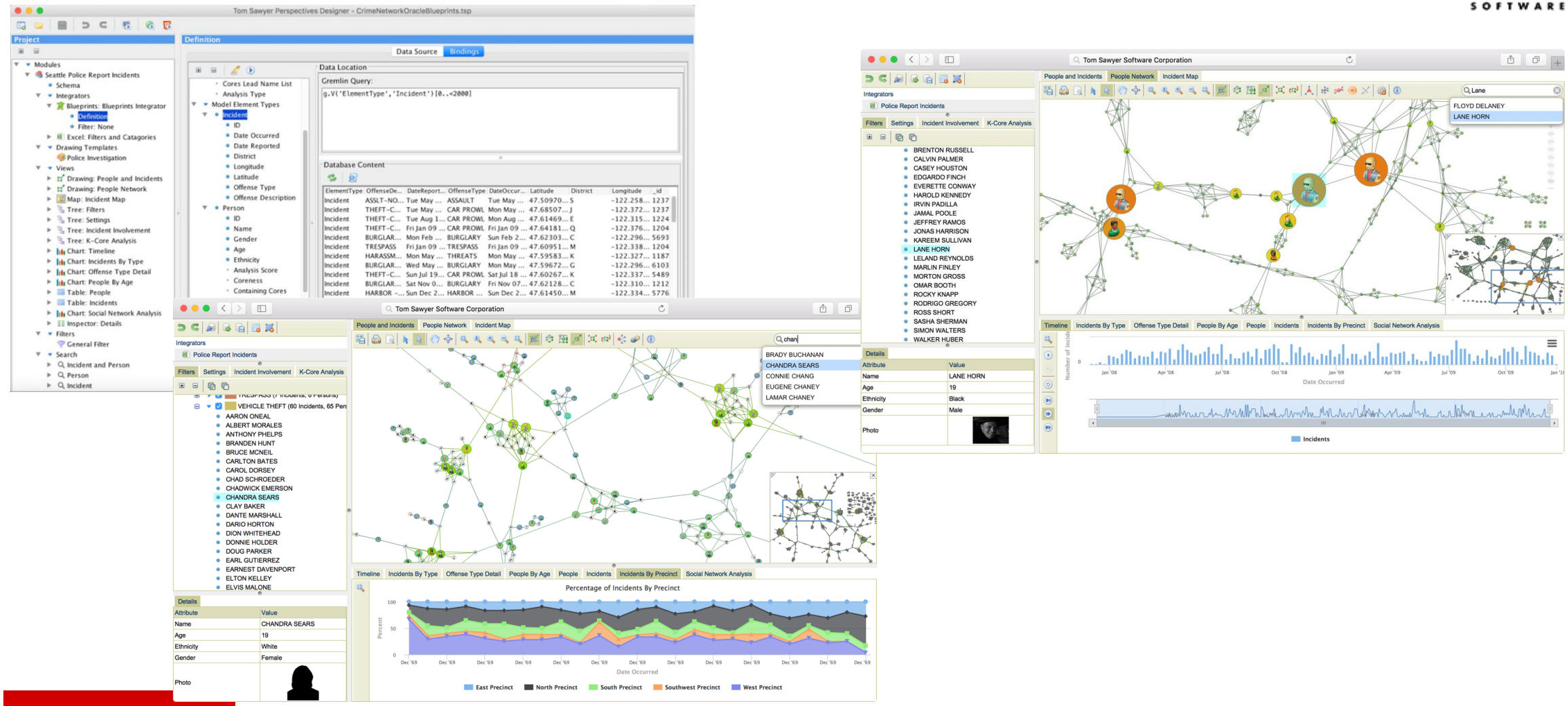
Text Search through Apache Lucene/Solr

- Use text indexing to access vertices or edges
  - Eg. find person with given name as starting point for reachability analysis
  - oraclePropertyGraph.createKeyIndex(“name”, Vertex.class);
  - oraclePropertyGraph.getVertices(“name”, “*Obama*”, true);
- Based on Apache Solr/Solr Cloud
  - Highly scaleable through sharding and replication
- Uses Apache Lucene under the covers
  - open source text search engine library
  - inverted index, ranked searching, fuzzy matching …
- Supports manual and auto indexing of Graph elements

Support for Cytoscape Open Source Visualization



Integration with Tom Sawyer Perspectives
via property graph REST APIs



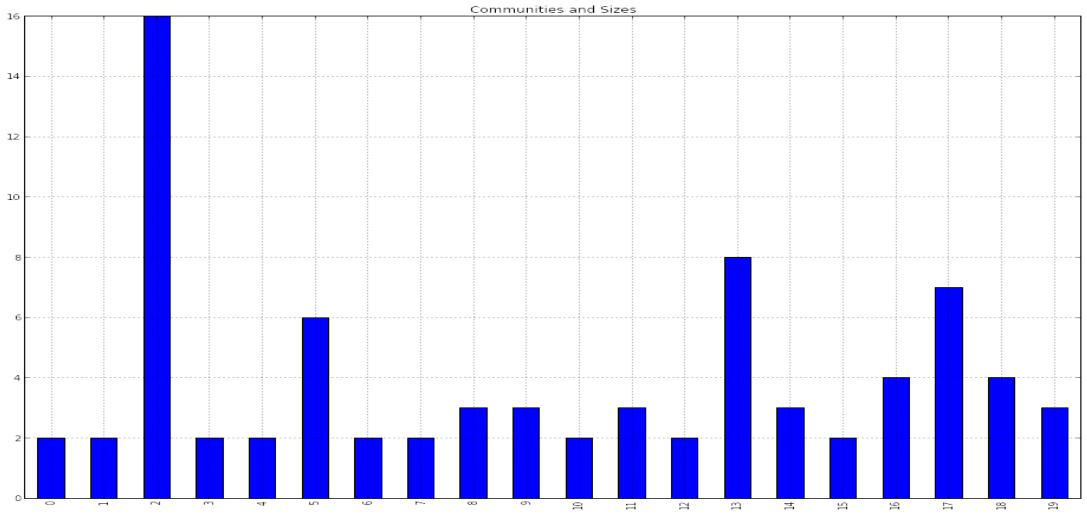
Python Interface

- Installation
  - property_graph/pyopg/README
- Usage
  - cd ${ORACLE_HOME}/md/property_graph/pyopg

./pyopg.sh

ipython notebook

```
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(16,12));
community_frame["size"].plot(kind="bar", title="Communities and Sizes")
ax.set_xticklabels(community_frame.index);
```



Program Agenda

1. Introduction
2. Property Graph Data Model & BDSG Architecture
3. Oracle Big Data Spatial and Graph Core Features
4. Graph Analytics using PGX Graph Analytics Engine
5. HoL: Analyzing a social network using Property Graphs

Graph Analytics workloads

Computational Graph Analytics

Connected Components

Modularity

Conductance

Shortest Path

Pagerank

Spanning Tree

Clustering Coefficient

Centrality

Coloring

Compute certain values on nodes and edges

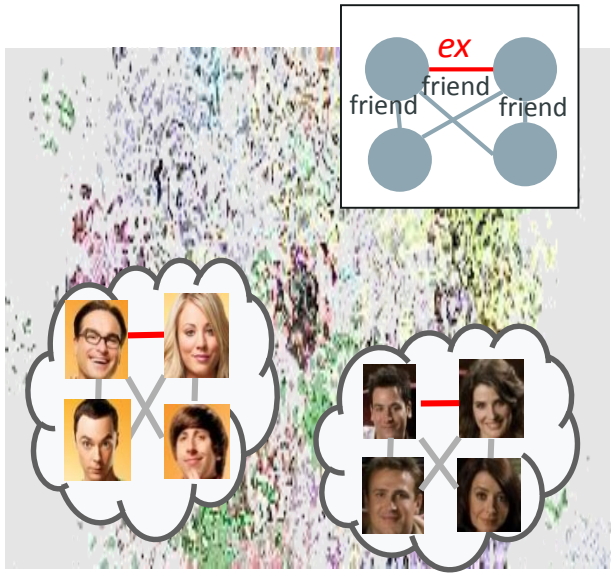
While (repeatedly) traversing or iterating on the graph

In certain procedural ways

Graph Pattern Matching

Given a description of a pattern

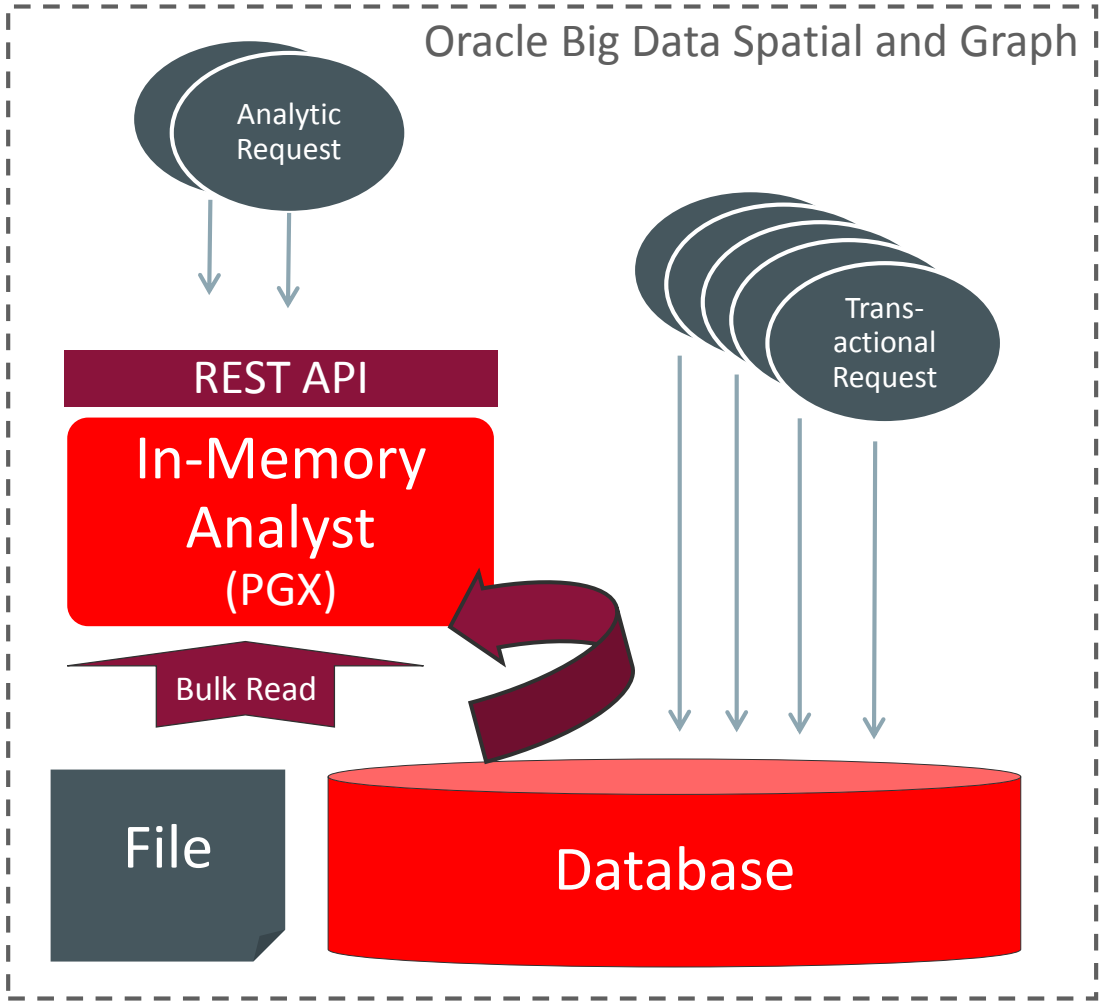
Find every sub-graph that matches it



Typical graph analysis systems do not support both

In-Memory Analyst (PGX)

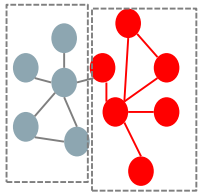
- PGX is the **in-memory**, parallel graph analytics engine of **Oracle Big Data Spatial and Graph**
- Approaches
  - Reads snapshot of graph data from database (or file)
  - Support delta-update from transactional changes in database
  - Processes analytic requests efficiently in-memory
    - Supports remote clients via REST



Computational Analytics: Built-in Package

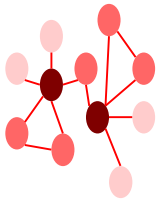
Rich set of built-in parallel graph algorithms

Detecting Components and Communities



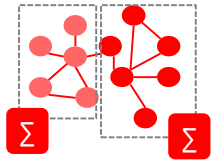
Tarjan's, Kosaraju's, Weakly Connected Components, Label Propagation (w/ variants), Soman and Narang's Spacification

Ranking and Walking



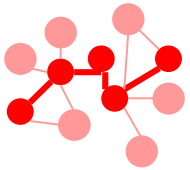
Pagerank, Personalized Pagerank, Betwenness Centrality (w/ variants), Closeness Centrality, Degree Centrality,
Eigenvector Centrality, HITS,
Random walking and sampling (w/ variants)

Evaluating Community Structures



Conductance,
Modularity
Clustering Coefficient
(Triangle Counting)
Adamic-Adar

Path-Finding

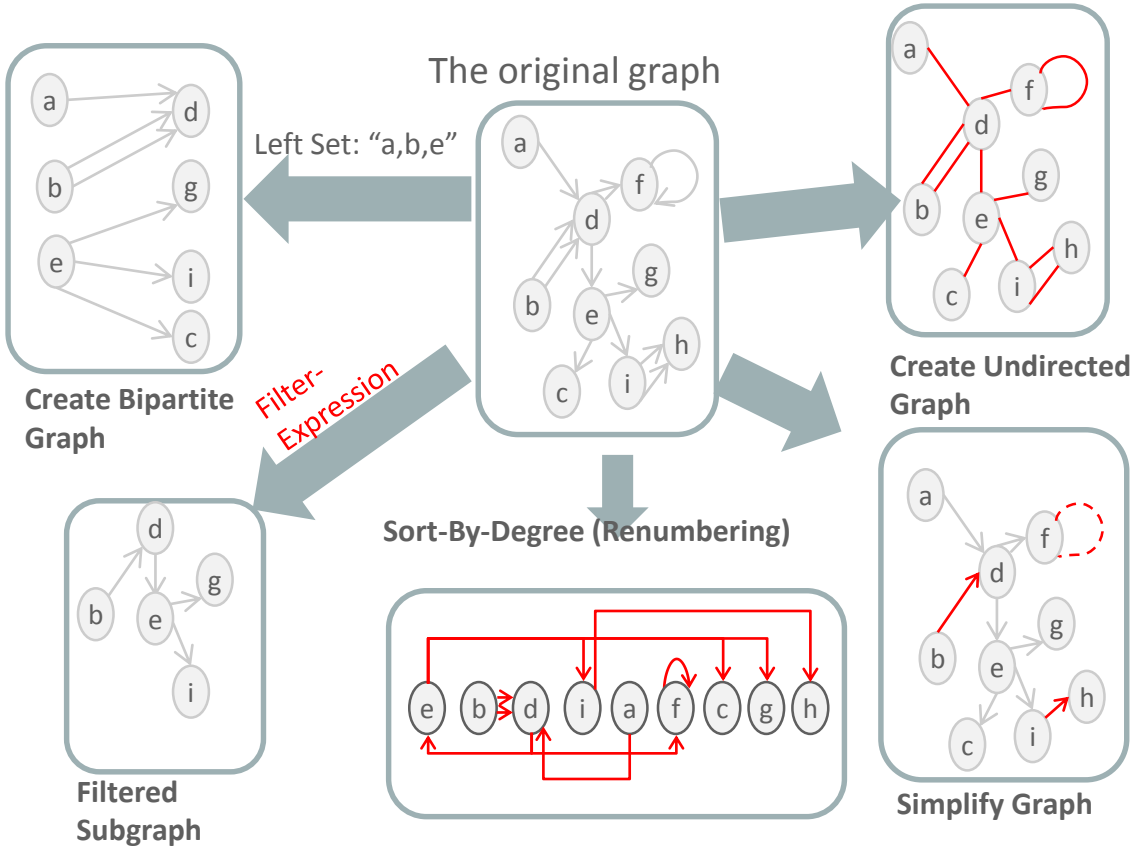


Hop-Distance (BFS)
Dijkstra's,
Bi-directional Dijkstra's
Bellman-Ford's

Link Prediction SALSA
(Twitter's Who-to-follow)

Other Classics Vertex Cover
Minimum Spanning-Tree
(Prim's)

… and parallel graph mutation operations

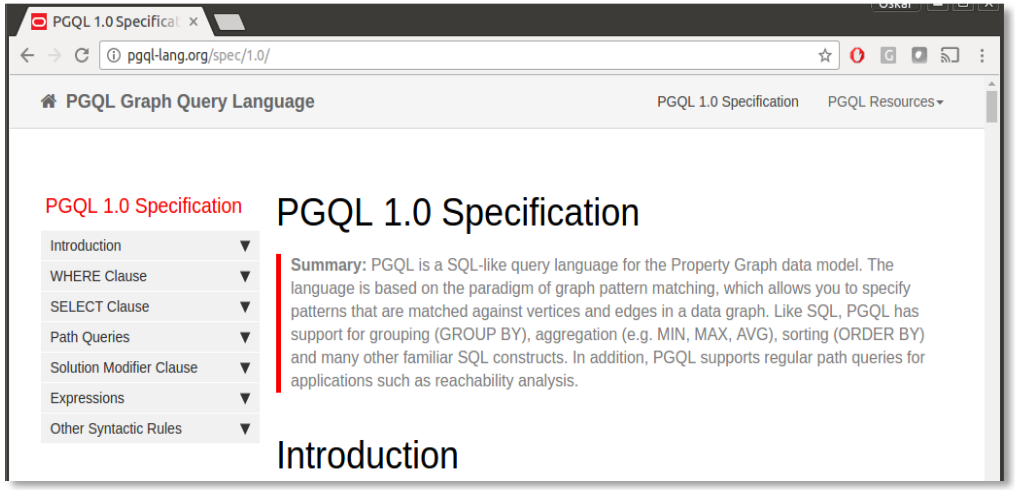


Pattern matching using PGQL

- SQL-like syntax but with graph pattern description and property access
  - Interactive (real-time) analysis
  - Supporting aggregates, comparison, such as max, min, order by, group by
- Finding a given pattern in graph
  - Fraud detection
  - Anomaly detection
  - Subgraph extraction
  - …
- Recursive path querying

- Proposed for standardization by Oracle
  - Specification available on-line
  - Open-sourced front-end (i.e. parser)

https://github.com/oracle/pgql-lang

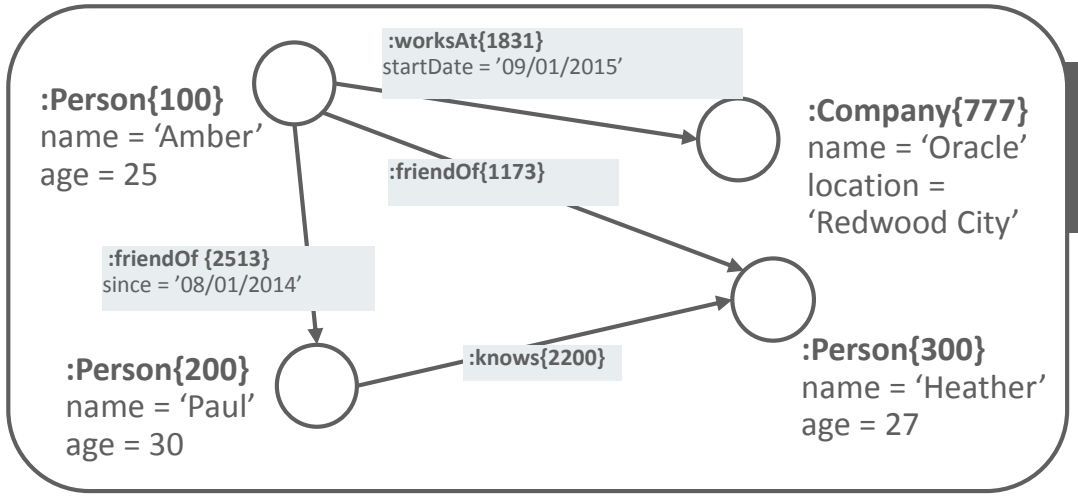


PGQL Example query

- Find all instances of a given pattern/template in data graph
- Fast, scaleable query mechanism

```
SELECT v3.name, v3.age
FROM 'myGraph'
WHERE
  (v1:Person WITH name = 'Amber') -[:friendOf]-> (v2:Person) -[:knows]-> (v3:Person)
```

query



data graph 'myGraph'

Query: Find all people who are known to friends of 'Amber'.

https://github.com/oracle/pgql-lang/

http://pgql-lang.org/

In-Memory Analyst (PGX) in Apache Zeppelin

Create notebooks with paragraphs that run graph queries or graph algorithms

Zeppelin Notebook Search your Notebooks anonymous

Pharma Use Case

Plot: Degree Distribution FINISHED

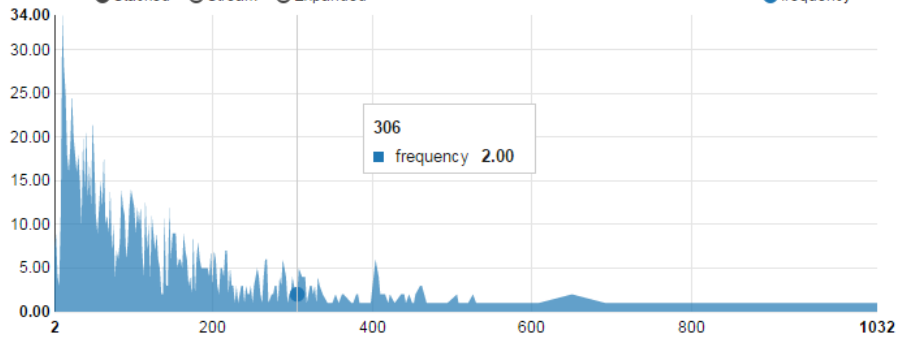
```
%pgx
// PGQL Query that groups and aggregates vertices filtered by 'node_type' based on a centrality measure of
interest that was computed in the panel above.
result_one = g.queryPgql("SELECT m.${centrality_measure} as ${centrality_measure}, COUNT(*) as frequency
WHERE (m), m.type_ = '${node_type}', m.${centrality_measure} !=0  GROUP BY m.${centrality_measure} ORDER
BY frequency DESC")
```

node_type: drug

centrality_measure: degree

settings

Stacked Stream Expanded frequency



Zeppelin Notebook Interpreter Search in your notebooks Connected

Reachability

Our Green-Marl program will populate it. Then we will run some code to query this property and return the graph nodes with a high value for it.

```
// create a new property named 'count'
count = graph.createVertexProperty(PropertyType.INTEGER, "count");
```

```
==> Vertex Property named 'count' of type integer belonging to graph flight
```

Now we are ready to run our Green-Marl program against the graph:

Run this paragraph (Shift+Enter)

```
findCoTravellers.run(graph, graph.getVertexProperty("vtype"), knownMembers, 2, count);
```

FINISHED

```
==> {
  "success" : true,
  "canceled" : false,
  "exception" : null,
  "returnValue" : null,
  "executionTimeMs" : 26
}
```

Took 0 seconds

Now we use *PGQL - Parallel Graph Query Language*, the graph *pattern matching* language PGX provides - to get our results:

```
rs = graph.queryPgql("SELECT p.id(), p.first_name, p.last_name, p.count WHERE (p: passenger WITH count > 0) ORDER BY p
.count DESC");
```

p.id()	p.first_name	p.last_name	p.count
23	LUCRECIA	DEBBRA	24
6	CAMMIE	KITTY	23
11	CHANTAY	ALEASE	22
13	HERIBERTO	SIGRID	22

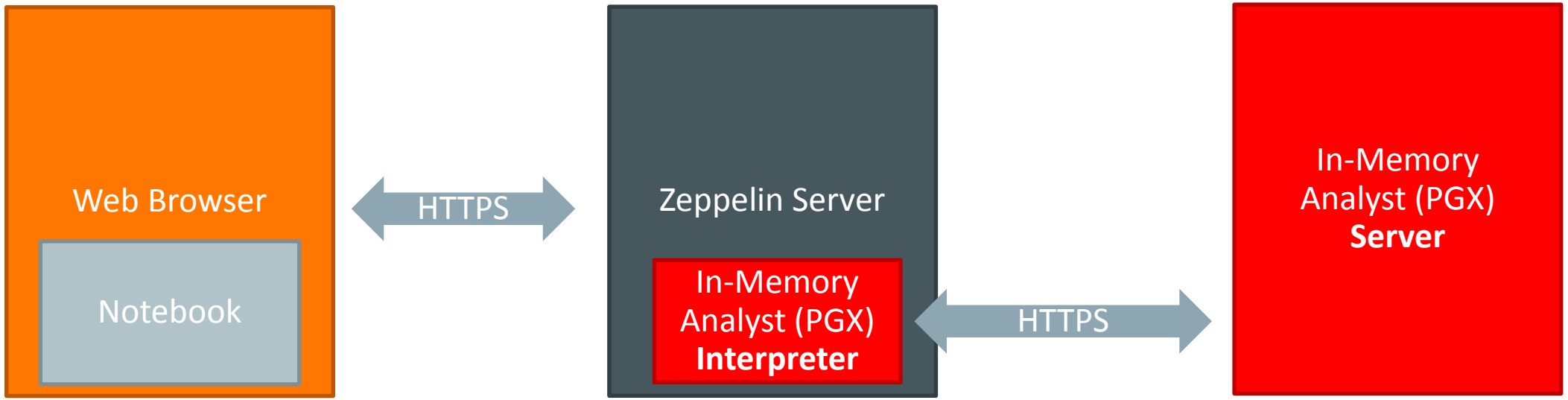
ORACLE

Apache Zeppelin Integration

- Apache Zeppelin is a **multi-purpose notebook** for data analysis and visualization similar to iPython/Jupyter
- Lots of language bindings and interpreters **built in ->**
- JVM based
- Very active development community
- Easy extensible

Oracle Labs
PGX

In-Memory Analyst (PGX) in Apache Zeppelin



Program Agenda

1. Introduction
2. Property Graph Data Model & BDSG Architecture
3. Oracle Big Data Spatial and Graph Core Features
4. Graph Analytics using PGX Graph Analytics Engine
5. HoL: Analyzing a social network using Property Graphs