



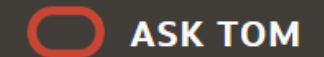
Graph Algorithms: The Core of Graph Analytics

—
Melli Annamalai and Ryota Yamanaka, Product Management, Oracle

August 27, 2020



AskTOM Office Hours: Graph Database and Analytics



- Welcome to our AskTOM Graph Office Hours series!
We're back with new product updates, use cases, demos and technical tips
<https://asktom.oracle.com/pls/apex/asktom.search?oh=3084>
- Sessions will be held about once a month
- **Subscribe** at the page above for updates on upcoming session topics & dates
And submit feedback, questions, topic requests, and view past session recordings
- **Note: Spatial** now has a new Office Hours series for location analysis & mapping features in Oracle Database:
<https://asktom.oracle.com/pls/apex/asktom.search?oh=7761>



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, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.



Agenda

1. Introduction to Graph Algorithms
2. Graph Algorithms Use Cases
3. Running Graph Algorithms
4. Scalability in Graph Analytics

Melli

Melli

Ryota

Ryota

Melli



Nashua, New Hampshire, USA
@AnnamalaiMelli

Ryota



Bangkok, Thailand
@ryotaymnk

Recap: Creating a Graph and Querying a Graph

Product Overview: Graph Database and Analytics

Graph data model: A different way to model your data

Property Graph Feature in Oracle Database:

Enterprise capabilities

Highly scalable

- In-memory query and analytics and in-database query
- 10s of billions of edges and vertices

PGQL: Powerful SQL-like graph query language

Analytics Java API: 50+ pre-built graph analysis algorithms

Visualization

- Light-weight web application, UI accessible from a browser

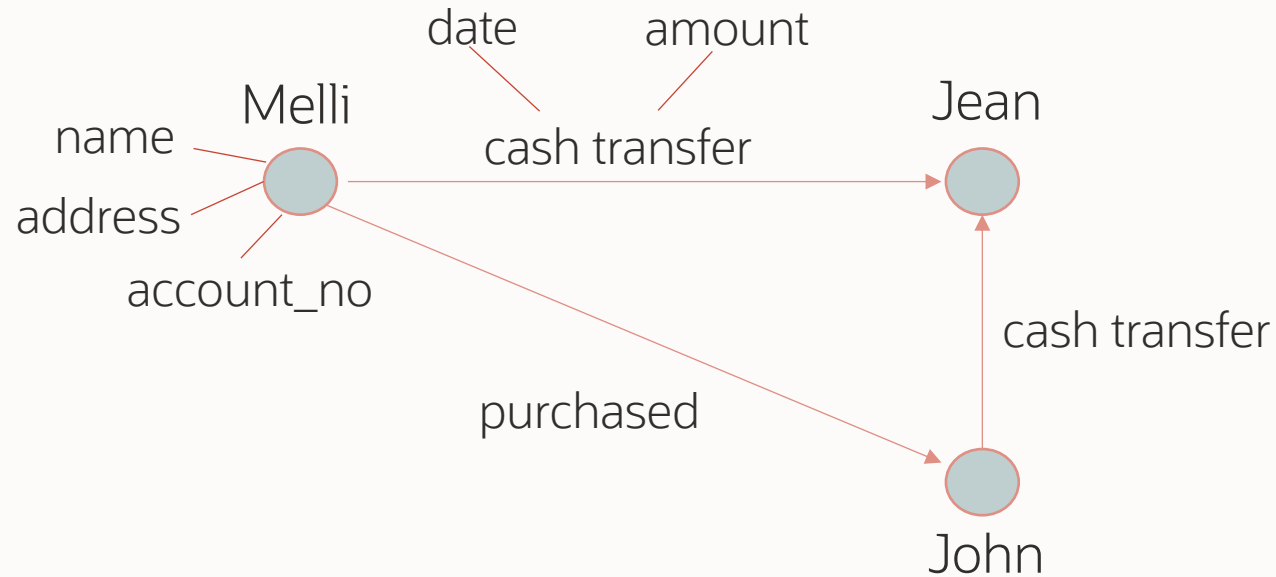
Graph Applications:

- Financial
- Law enforcement and security
- Manufacturing
- Public sector
- Pharma

and more

What is a Graph?

A collection of points (vertices/nodes) and lines between those points (edges)



Create a Graph from Database Tables

—

ACCOUNTS	ACCT_ID
	0
	1
	2
	3
	4
	5
	...

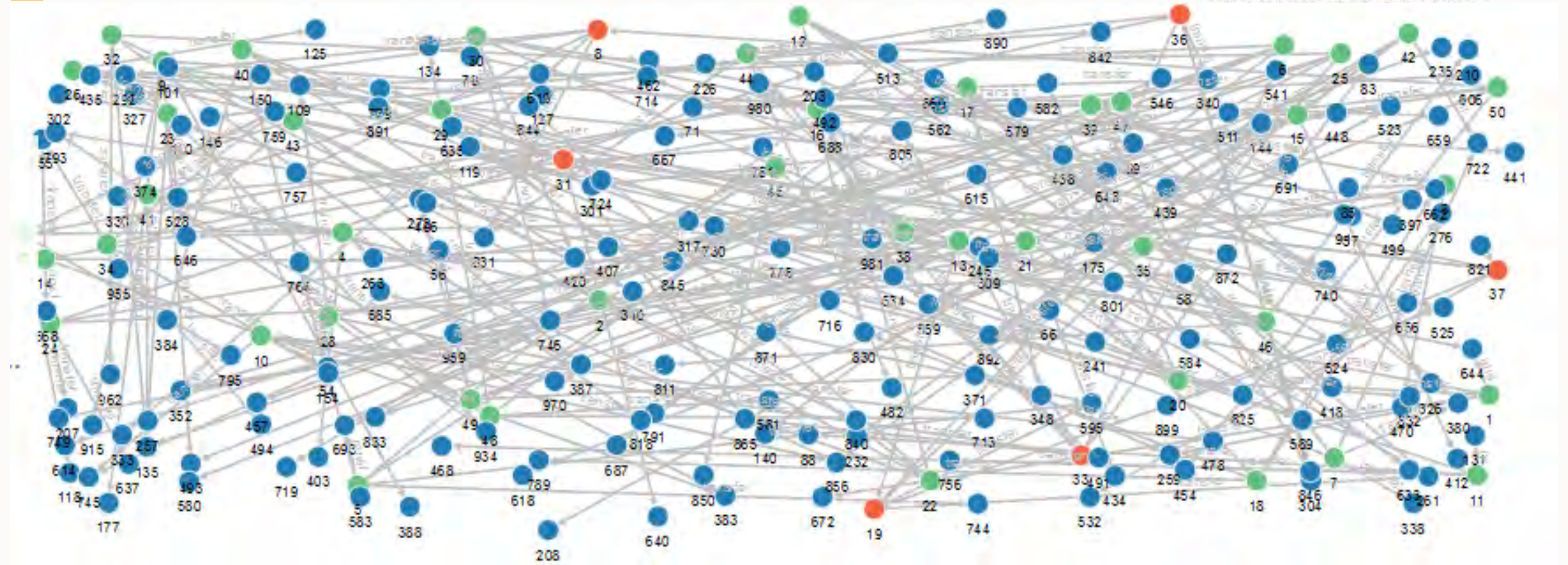
TRANSACTIONS	FROM_ACCOUNT	TO_ACCOUNT	AMOUNT
	1	672	1000
	1	584	1000
	1	259	100000
	2	833	5001
	2	840	7050
	2	493	4363

PGQL DDL SYNTAX:

```
CREATE PROPERTY GRAPH bank_graph
  VERTEX TABLES (
    ACCOUNTS LABEL Account PROPERTIES ( ACCT_ID )
  )
  EDGE TABLES (
    TRANSACTIONS
    SOURCE KEY ( FROM_ACCOUNT ) REFERENCES ACCOUNTS
    DESTINATION KEY ( TO_ACCOUNT ) REFERENCES ACCOUNTS
    LABEL transfer PROPERTIES ( AMOUNT )
```



Cash Transfer Graph

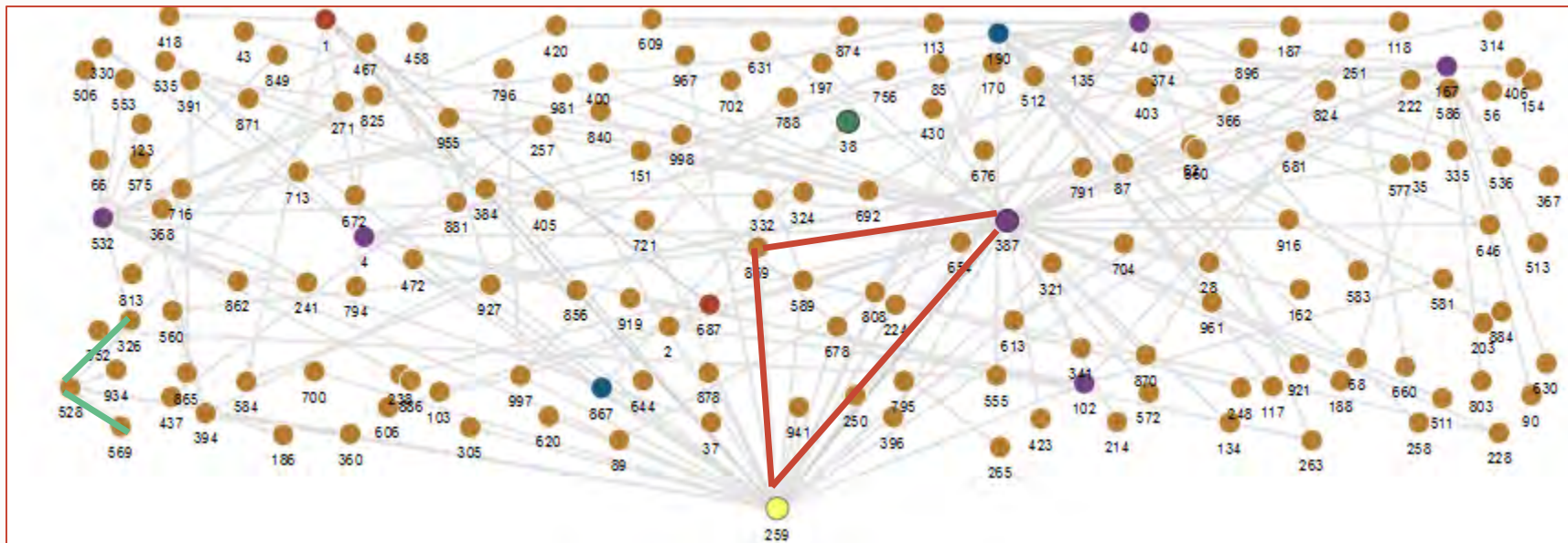


Graph Queries: Finding Patterns in a Graph

Is there a pattern that connects 528 to 326 and 569?

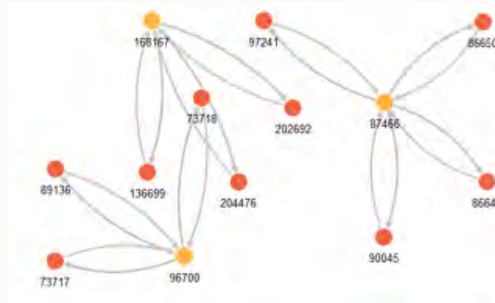
Property Graph Query Language (PGQL)

```
SELECT v1, v2, v3, e1, e2  
MATCH (v1)-[e1]->(v2),  
MATCH (v1)-[e2]->(v3)  
where v1.id=528 and v2.id=326 and v3.id=569
```

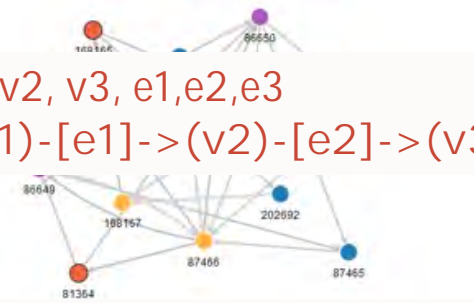


Graph Queries: Finding Patterns in a Graph

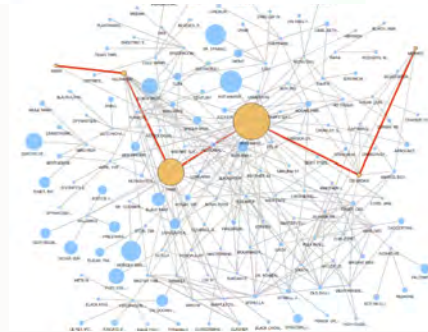
Cycles



```
SELECT v1, v2, v3, e1,e2,e3  
MATCH (v1)-[e1]->(v2)-[e2]->(v3)-[e3]->(v1)
```



Paths

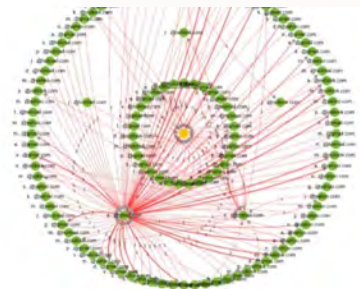
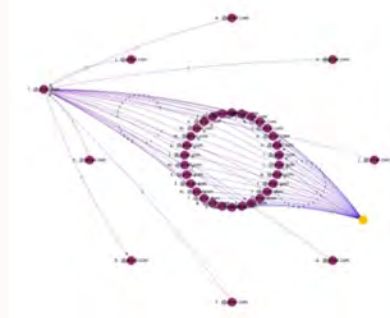


```
SELECT *  
MATCH (n)-/:transfer{1,6}/->(m)  
WHERE ID(n) = 1
```



```
SELECT n, ARRAY_AGG(ID(m)), ARRAY_AGG(ID(e))  
MATCH TOP 2 SHORTEST ((n) (-[e:transfer]->(m))* (n))  
WHERE ID(n) = 1
```

Patterns



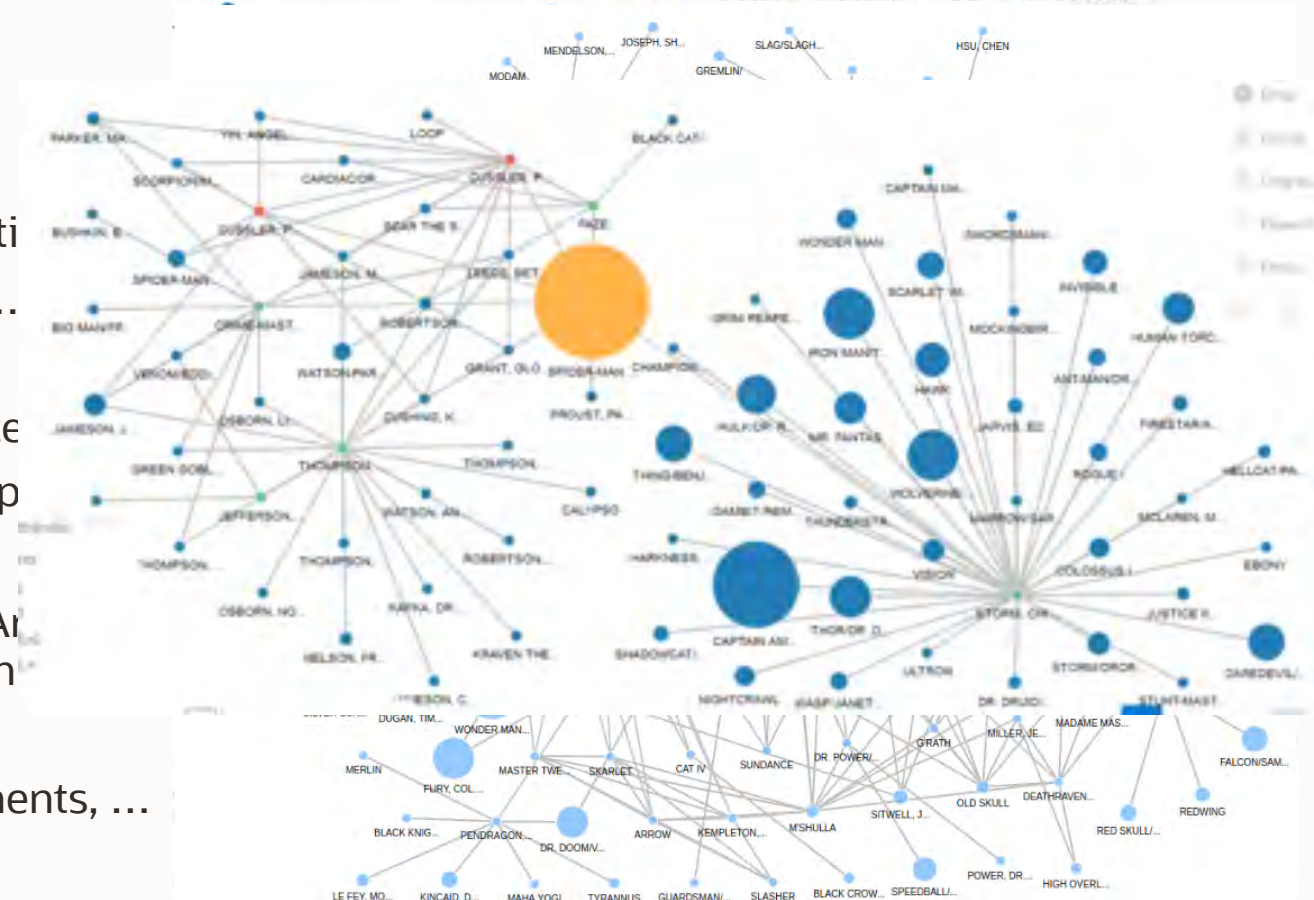
Graph Algorithms

Graph Algorithms

Analyze the full graph

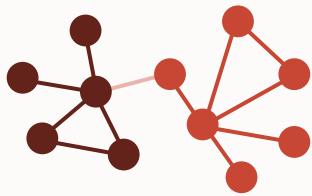
Derive information about:

- What is the value of a vertex in relation to other vertices
 - Centrality, PageRank, Betweenness Centrality, ...
- In how many ways can you reach vertex B from vertex A
 - Shortest path, Fattest path, Number of hops in path
- How tightly (or sparsely) is the graph connected? Are there tightly connected sub-graphs within the graph?
 - Are there isolated vertices in the graph?
 - Strongly connected, Weakly connected components, ...
- Evaluating structures



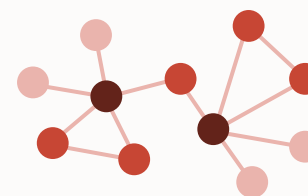
Graph Analytics - 50+ Built-in Algorithms

Detecting Components and Communities



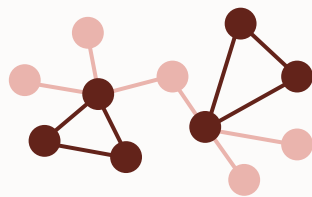
Strongly Connected Components, Weakly Connected Components, Label Propagation, Conductance Minimization, Infomap

Ranking and Walking



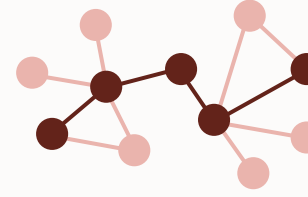
PageRank, Personalized PageRank, Degree Centrality, Closeness Centrality, Vertex Betweenness Centrality, Eigenvector Centrality, HITS, SALSA, Random Walk with Restart

Evaluating Structures



Adamic-Adar Index, Conductance, Cycle Detection, Degree Distribution, Eccentricity, K-Core, LCC, Modularity, Reachability Topological Ordering, Triangle Counting

Path-Finding



Shortest Path (Bellman-Ford, Dijkstra, Bidirectional Dijkstra), Fattest Path, Compute Distance Index, Enumerate Simple Paths, Fast Path Finding, Hop Distance

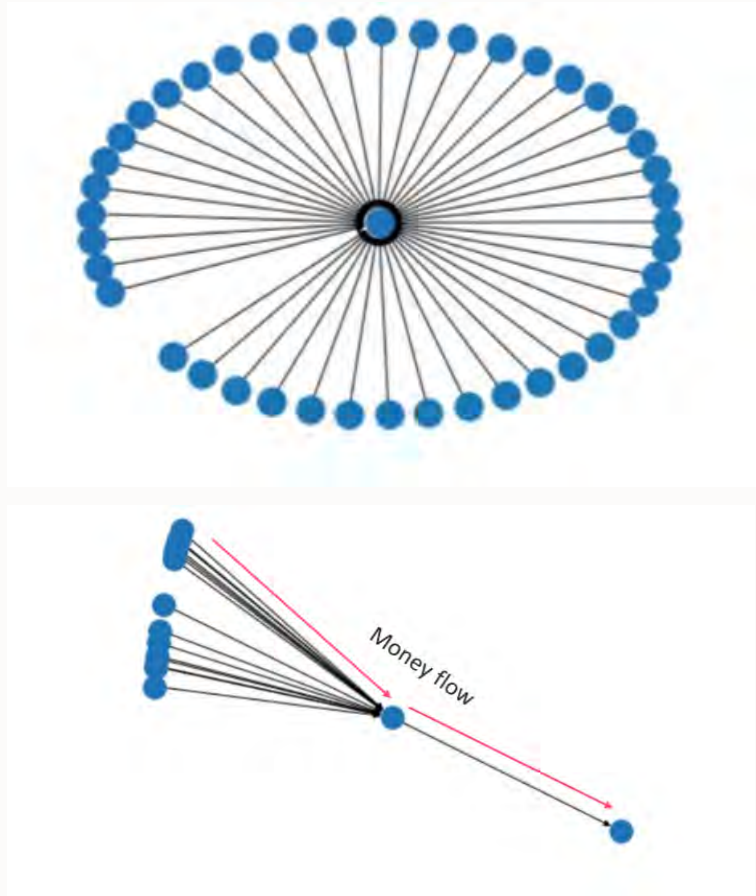
Link Prediction

WTF (Who to follow)

Others

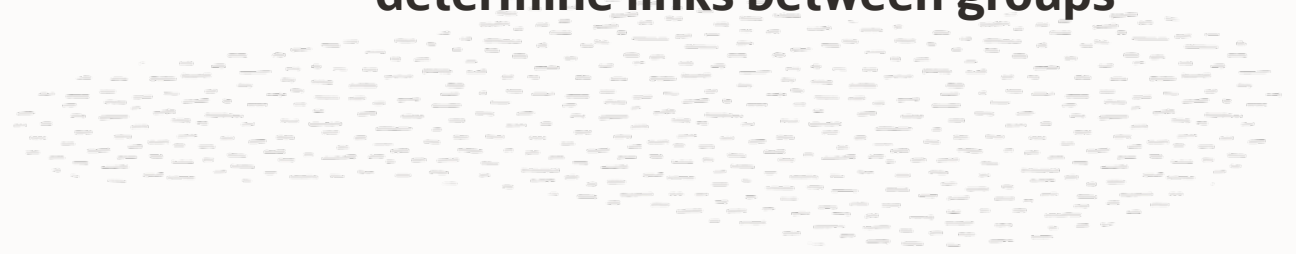
Minimum Spanning-Tree, Matrix Factorization

Ranking: Importance of Vertices

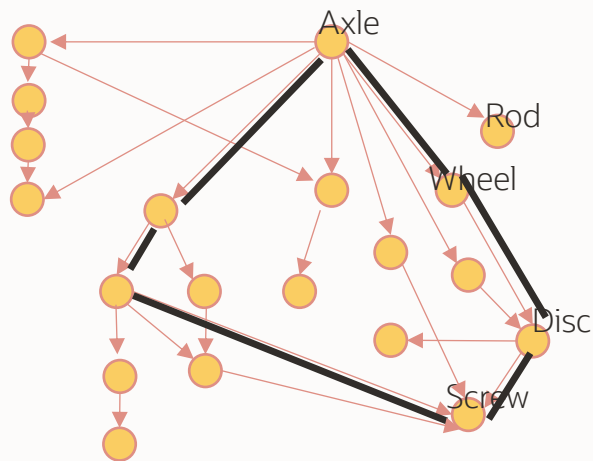
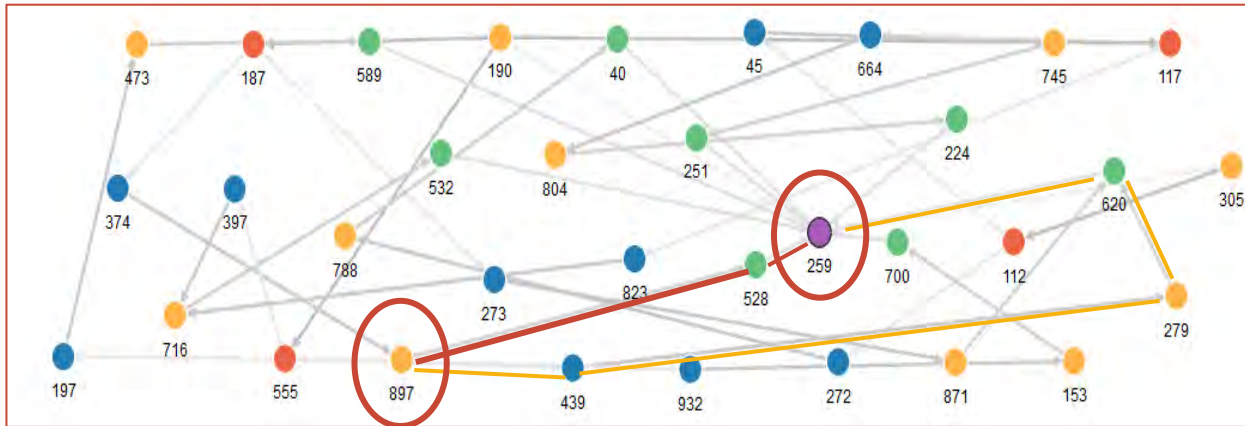


Use cases

- **Financial:** Find accounts through which most of the money flows
- **Retail:** Influencers in a social network, for product recommendation
- **Law enforcement:** Vertices with high betweenness centrality values determine links between groups



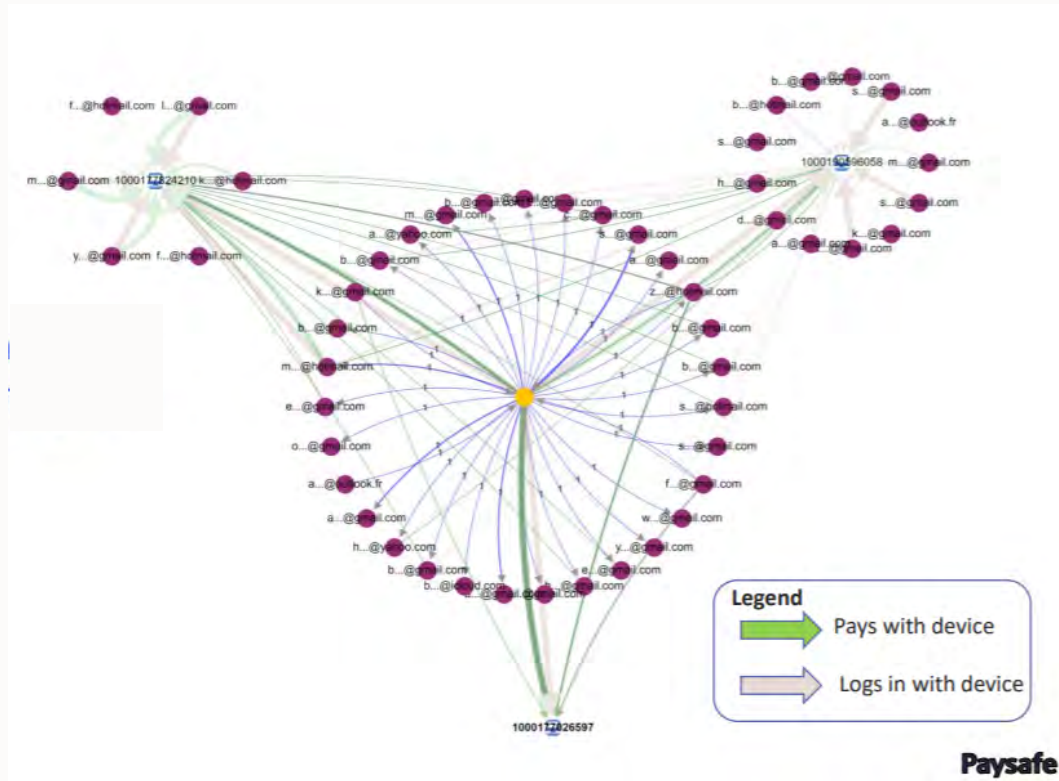
Paths between Vertices



Use cases

- **Telecommunications:** Network management: What-If analysis.
- **Financial:** Was there a cash transfer between these two accounts?
- **Manufacturing:** Dependency analysis in a Bill of Materials graph

Detecting Communities



Use cases

- **Financial:** Does this suspicious account belong to a community of fraudsters?
- **Financial:** Community of users using the same device
- **Retail:** Can behavior of members of the community predict churn?

Running Graph Algorithms

Setup Your Graph Server



Installation: <https://github.com/ryotayamanaka/oracle-pg/tree/20.3>

Clone repository (Note, the branch is 20.3)

```
$ git clone https://github.com/ryotayamanaka/oracle-pg.git -b 20.3
```

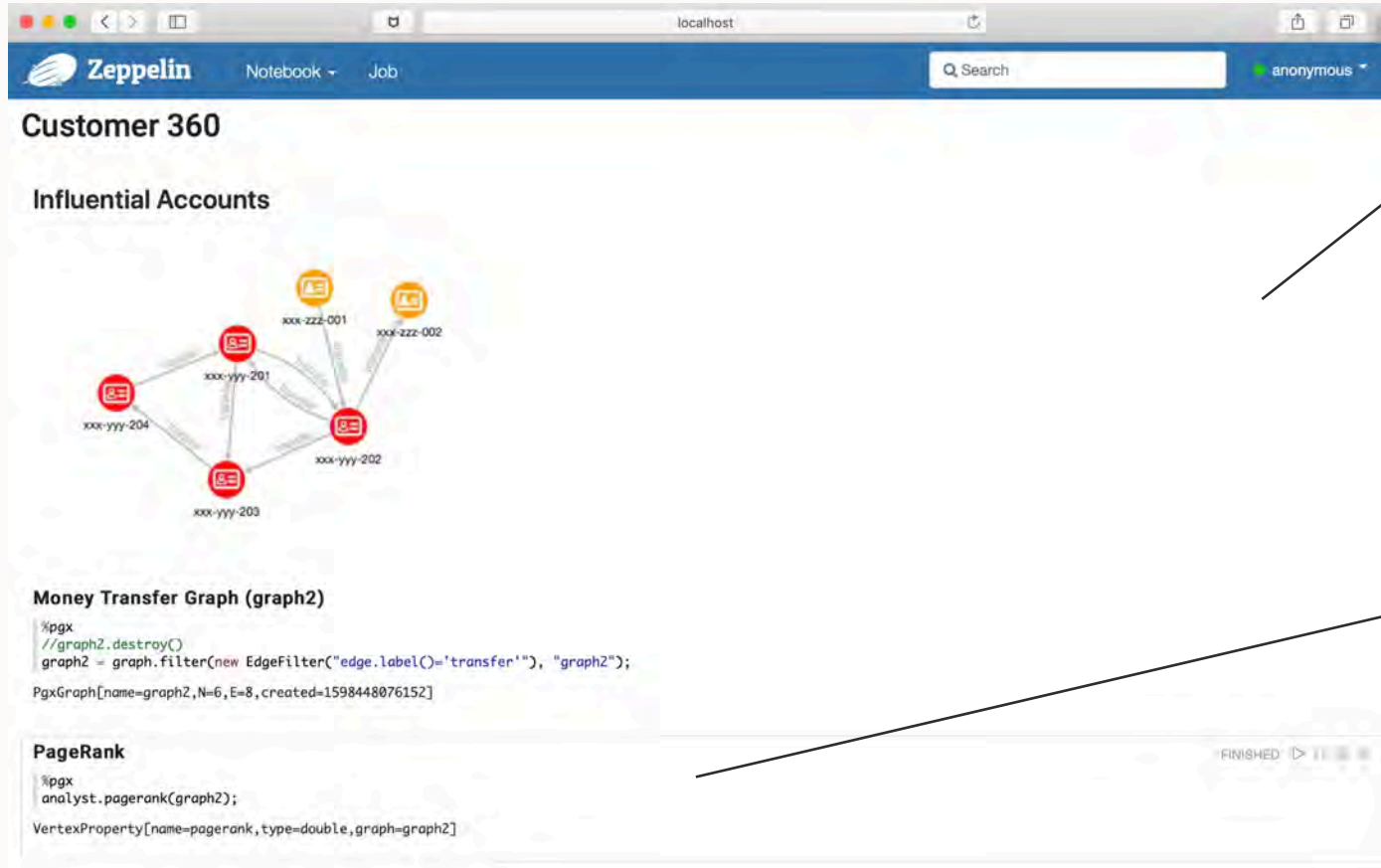
Download and extract packages (Note, the packages are version 20.3)

```
$ sh extract.sh
```

Build and start the docker containers

```
$ docker-compose up -d
```

Run Algorithms from JShell or Zeppelin



The interpreter for Apache Zeppelin Notebook is provided and Groovy syntax is supported.

(Python API is also planned)

We can run algorithms:
`analyst.pagerank(graph2)`

Write Queries to Get the Results

After running algorithms, the results are stored into the graph, e.g. each node gets a new property "pagerank".

Users can access the results using PGQL queries:

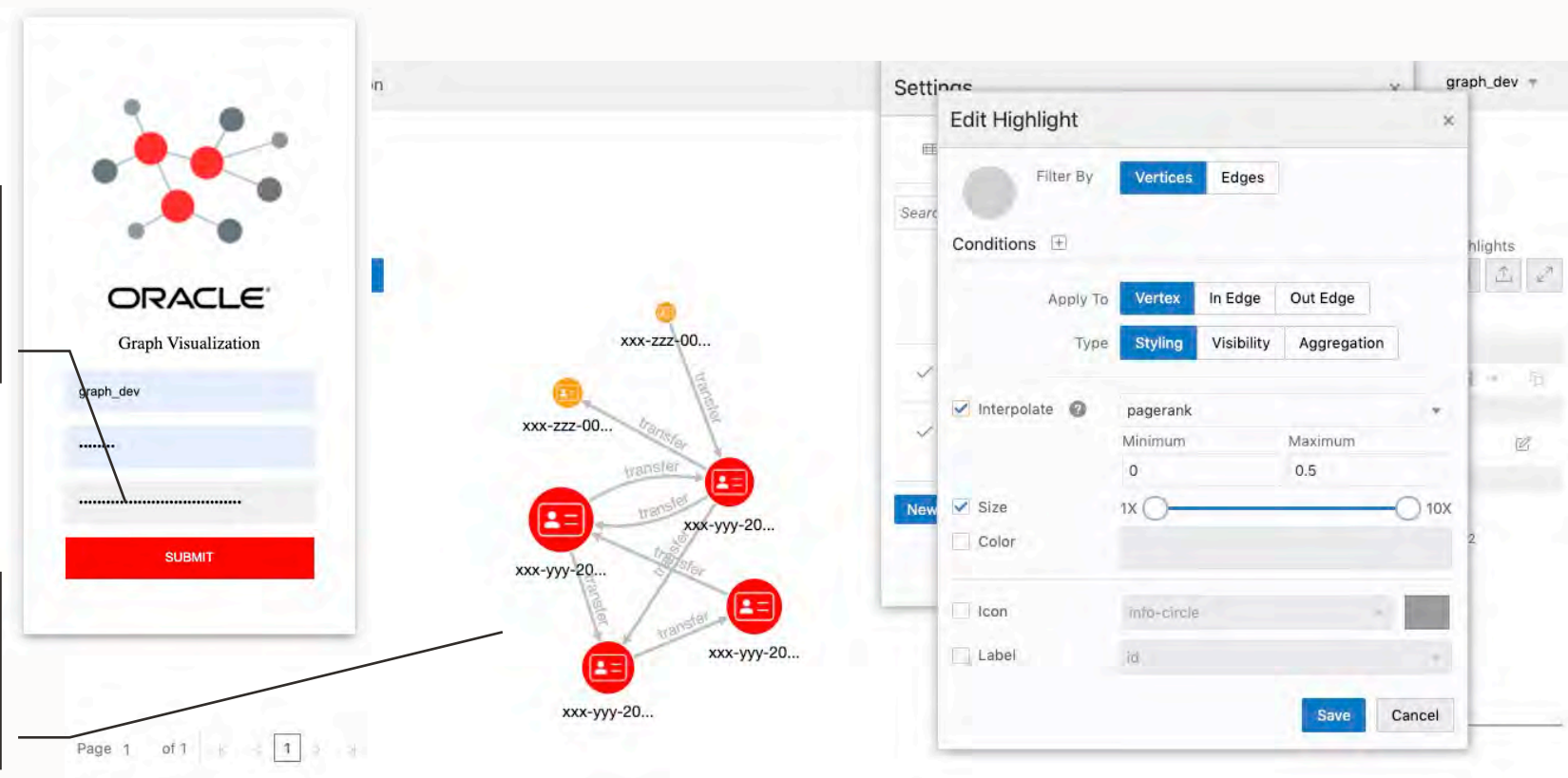
```
SELECT a.account_no, a.pagerank
FROM MATCH (a)
ORDER BY a.pagerank DESC
```



Visualize the Results

Login with the same **session ID** as the one which ran the algorithms.

The size of nodes can be linked to the pagerank scores.



Example - Bank Customer Analysis



Oracle as a Property Graph

Use Case: Customer 360 analysis

Overview

This example shows how integrating multiple datasets, using a graph, facilitates additional analytics can lead to new insights. We will use three small datasets for illustrative purposes. The first contains accounts and account owners. The second is purchases by the people who own those accounts. The third is transactions between these accounts.

The combined dataset is then used to perform the following common graph query and analyses: pattern matching, detection of cycles, finding important nodes, community detection, and recommendation.

Note: This lab assumes you have successfully completed Lab 1 Setup with Docker and have an environment up and running with Zeppelin at <http://localhost:8080> and the Graph Visualization component at <http://localhost:7007/ui/>.

[Expand All Steps](#)

- Overview
- Graph Query and Analysis in Apache Zeppelin
- Graph Visualization
- Graph Query and Analysis in JShell
- Acknowledgements

<https://oracle.github.io/learning-library/data-management-library/database/graph/livelabs/?lab=lab-2-customer-360-analysis>

Influential Accounts

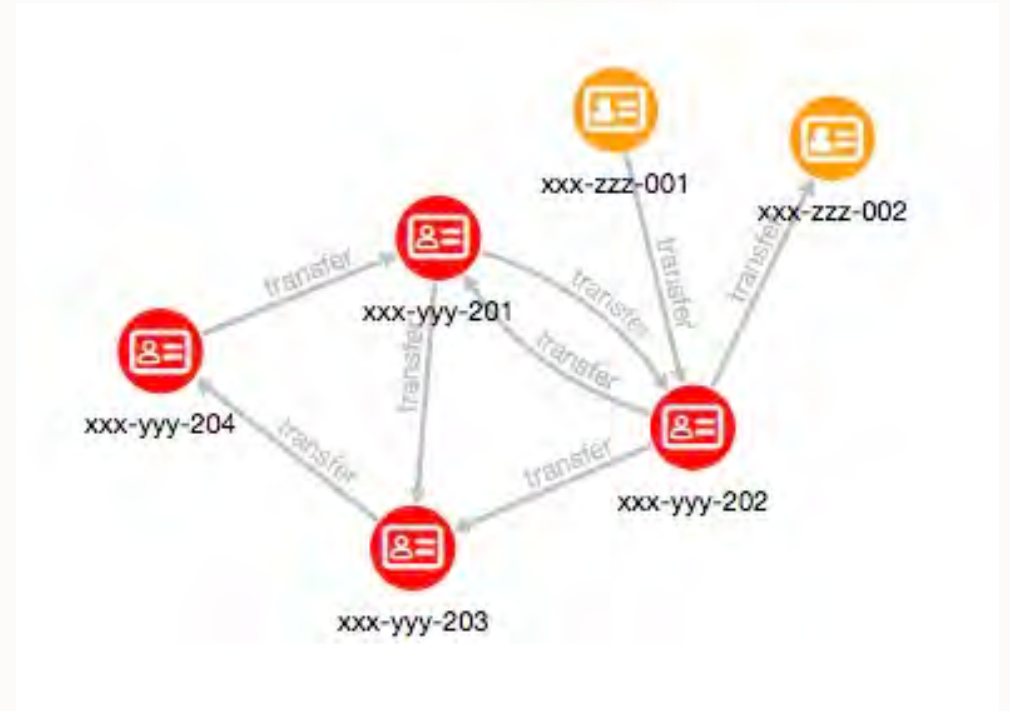
Which is the account with the highest pagerank?

Filter by "transfer" relationship, run **pagerank** algorithm. The result is stored as "pagerank" new node property, and it can be queried.

```
graph2 = graph.filter(new EdgeFilter(  
    "edge.label()='transfer'"));
```

```
analyst.pagerank(graph2);
```

```
SELECT a.account_no, a.pagerank  
FROM MATCH (a)  
ORDER BY a.pagerank DESC
```



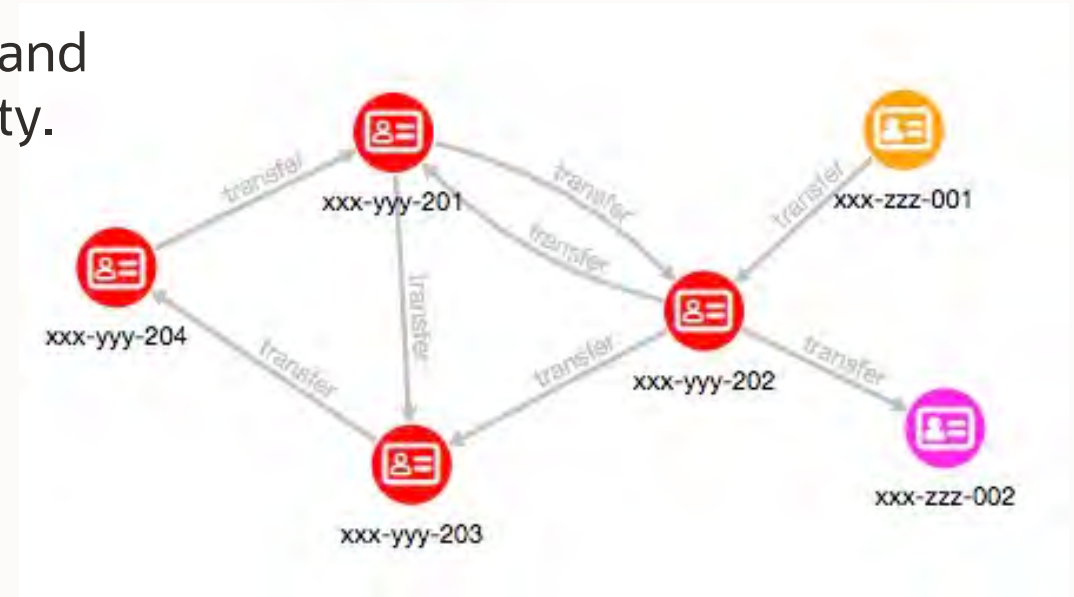
Community Detection

Find the communities where every account is reachable from every other account

Run **strongly connected component** algorithm and reflect the results into "component" node property.

```
result = analyst.sccKosaraju(sg)
```

```
SELECT  
  a.scc_kosaraju, COUNT(a.account_no)  
FROM MATCH (a)  
GROUP BY a.scc_kosaraju
```

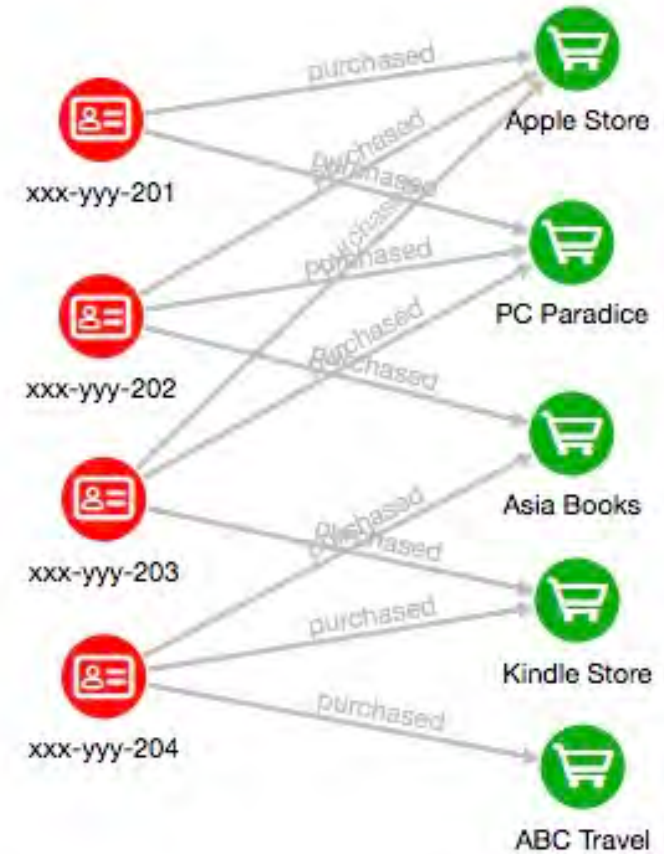


Recommendation

Which merchants should be recommended to the account "-201" based on the purchase history?

Filter by "purchased" relationship, run **personalized pagerank** from the node 201, and then query :

```
SELECT ID(x), x.name, x.pagerank
FROM MATCH (x)
WHERE x.type = 'merchant'
AND NOT EXISTS (
  SELECT *
  FROM MATCH (x)-[:purchased_by]->(a)
  WHERE ID(a) = 201
)
ORDER BY x.pagerank DESC
```



More Algorithms

PGX 20.1.1 Documentation

Search the PGX documentation

Home

Getting Started

- System Configuration
- Quickstart
- Samples and Use Cases

Reference

- Overview
- Analytics
 - Graph Queries (PGQL)
 - Graph Algorithms (PGX Algorithm)
 - Graph Algorithms (Green-Marl)
 - Built-In Graph Algorithms**
 - Adamic-Adar index
 - All Vertices and Edges on Filtered Path
 - Bellman-Ford Algorithms
 - Bidirectional Dijkstra Algorithms
 - Bipartite Check

Built-In Algorithms

PGX includes a wide selection of optimized graph algorithms that can be invoked through the [Analyst](#). The following table provides an overview of the available algorithms, grouped by category.

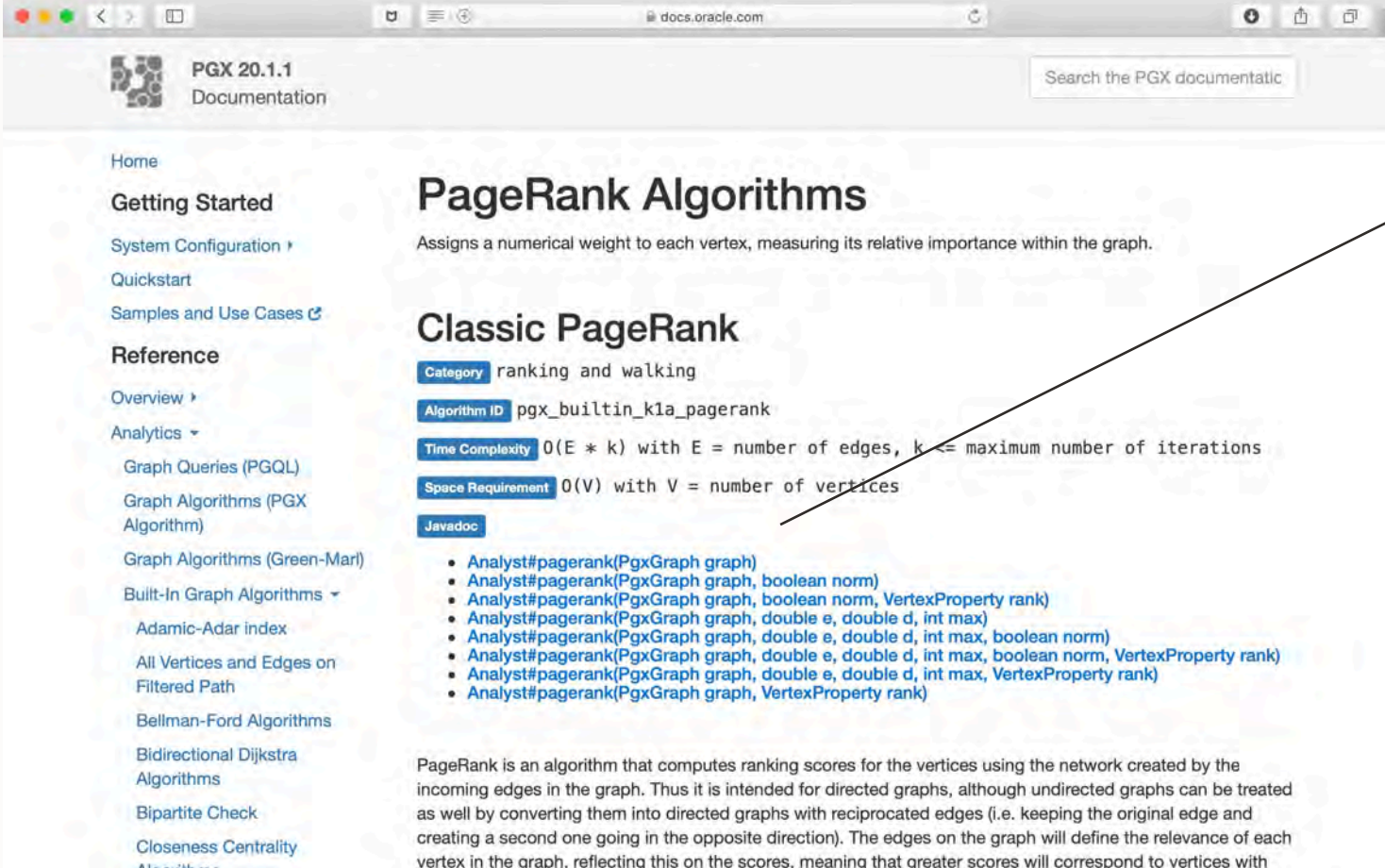
Category	Algorithms
Classic graph algorithms	Prim's Algorithm
Community detection	Conductance Minimization (Soman and Narang Algorithm) , Infomap , Label Propagation , Louvain
Connected components	Strongly Connected Components , Weakly Connected Components (WCC)
Link prediction	WTF (Whom To Follow) Algorithm
Matrix factorization	Matrix Factorization
Other	Graph Traversal Algorithms
Path finding	All Vertices and Edges on Filtered Path , Bellman-Ford Algorithms , Bidirectional Dijkstra Algorithms , Compute Distance Index , Compute High-Degree Vertices , Dijkstra Algorithms , Enumerate Simple Paths , Fast Path Finding , Fattest Path , Filtered Fast Path Finding , Hop Distance Algorithms

All built-in algorithms are listed here.

https://docs.oracle.com/cd/E56133_01/latest/reference/analytics/builtins.html



More Algorithms



The screenshot shows the PGX 20.1.1 Documentation page. The left sidebar contains a navigation menu with links to Home, Getting Started, System Configuration, Quickstart, Samples and Use Cases, Reference, Overview, Analytics, Graph Queries (PGQL), Graph Algorithms (PGX Algorithm), Graph Algorithms (Green-Marl), Built-In Graph Algorithms, Adamic-Adar index, All Vertices and Edges on Filtered Path, Bellman-Ford Algorithms, Bidirectional Dijkstra Algorithms, Bipartite Check, and Closeness Centrality Algorithms. The main content area is titled "PageRank Algorithms" and describes the algorithm as assigning a numerical weight to each vertex. Below this, the "Classic PageRank" section lists the category (ranking and walking), algorithm ID (pgx_builtin_k1a_pagerank), time complexity ($O(E * k)$), space requirement ($O(V)$), and a list of Javadoc links for various `Analyst#pagerank` methods. A line from a text box on the right points to the Javadoc link for `Analyst#pagerank(PgxGraph graph, VertexProperty rank)`.

PGX 20.1.1 Documentation

Search the PGX documentation

PageRank Algorithms

Assigns a numerical weight to each vertex, measuring its relative importance within the graph.

Classic PageRank

Category ranking and walking

Algorithm ID pgx_builtin_k1a_pagerank

Time Complexity $O(E * k)$ with E = number of edges, k = maximum number of iterations

Space Requirement $O(V)$ with V = number of vertices

Javadoc

- [Analyst#pagerank\(PgxGraph graph\)](#)
- [Analyst#pagerank\(PgxGraph graph, boolean norm\)](#)
- [Analyst#pagerank\(PgxGraph graph, boolean norm, VertexProperty rank\)](#)
- [Analyst#pagerank\(PgxGraph graph, double e, double d, int max\)](#)
- [Analyst#pagerank\(PgxGraph graph, double e, double d, int max, boolean norm\)](#)
- [Analyst#pagerank\(PgxGraph graph, double e, double d, int max, boolean norm, VertexProperty rank\)](#)
- [Analyst#pagerank\(PgxGraph graph, double e, double d, int max, VertexProperty rank\)](#)
- [Analyst#pagerank\(PgxGraph graph, VertexProperty rank\)](#)

PageRank is an algorithm that computes ranking scores for the vertices using the network created by the incoming edges in the graph. Thus it is intended for directed graphs, although undirected graphs can be treated as well by converting them into directed graphs with reciprocated edges (i.e. keeping the original edge and creating a second one going in the opposite direction). The edges on the graph will define the relevance of each vertex in the graph, reflecting this on the scores, meaning that greater scores will correspond to vertices with

Each algorithm page provides the description, implementation, and the link to **Javadoc** page.

More Algorithms



The screenshot shows the Oracle Java API documentation for the `pagerank` method. The browser address bar shows `docs.oracle.com`. The page title is `pagerank` and the class is `class in oracle.pgx.api`. The method signature is:

```
public <ID> VertexProperty<ID, java.lang.Double> pagerank(PgxGraph graph)
    throws java.util.concurrent.ExecutionException,
           java.lang.InterruptedException
```

Below the signature, a paragraph states: "PageRank computes ranking scores based on the edges in a graph. It compares and spots out important vertices in a graph".

Definition

PageRank is an algorithm that computes ranking scores for the vertices using the network created by the incoming edges in the graph. Thus it is intended for directed graphs, although undirected graphs can be treated as well by converting them into directed graphs with reciprocated edges (i.e. keeping the original edge and creating a second one going in the opposite direction). The edges on the graph will define the relevance of each vertex in the graph, reflecting this on the scores, meaning that greater scores will correspond to vertices with greater relevance.

Implementation Details

The implementation of this algorithm uses an iterative method. The PageRank values of all the vertices in the graph are computed, hence updated, at each iteration step.

.....

Examples

```
PgxGraph graph = ...;
VertexProperty<Integer, Double> pagerank = analyst.pagerank(graph);
PgqlResultSet rs = graph.queryPgql(
    "SELECT x, x." + pagerank.getName() + " MATCH (x) ORDER BY x." + pagerank.getName() + " DESC");
rs.print();
```

At the bottom of the page, there are links for "Cookie Preferences" and "Ad Choices".

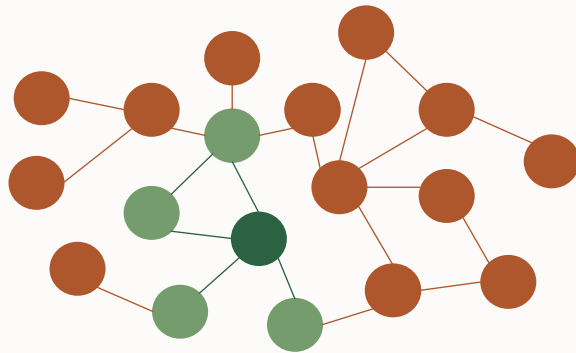
The method section for each algorithm has **examples** to show how it can be used in Java.

Scalability in Graph Analytics

Two Types of Processing in Graph Analytics

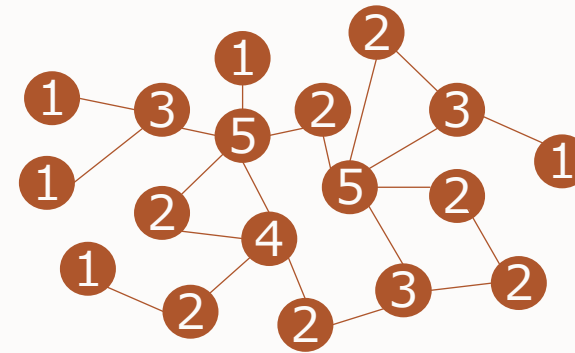
Query

- Search for surrounding nodes
- Traverse property paths
- Pattern matching
- Extract sub-graphs



Algorithm

- Rank importance of nodes
- Detect components and clusters
- Evaluate structure of communities
- Shortest paths



Two Types of Processing in Graph Analytics

Oracle Graph Server can execute both **queries**, **algorithms**, mutation (simplify, filter, ...) and their combination.

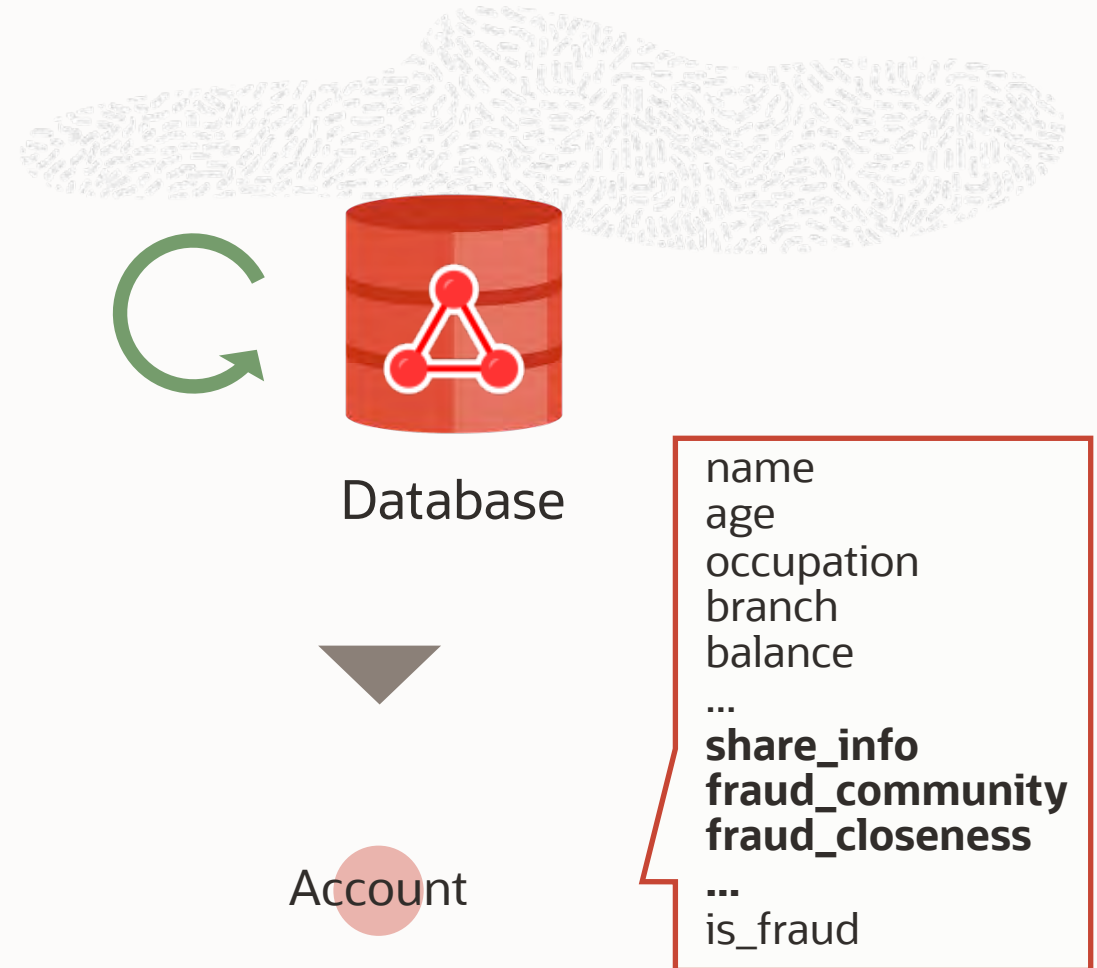
Which is the node in **3 steps**
from **my node**, and has the **highest**
pagerank among them?



Graph Algorithms and Analysis Flows

Pattern 1 - Finding significant nodes

1. Run graph algorithms for scoring all nodes
 - ranking (centrality, pagerank, ...)
 - community detection
 2. The scores (=new features) are used for:
 - finding high score nodes
 - as conditions in graph query
 - as machine learning input
- Use Cases
 - Financial fraud detection, Tax fraud detection, Influencer detection, Anormal behavior detection in cyber security, Scoring importance of devices in networks (utility and communication)

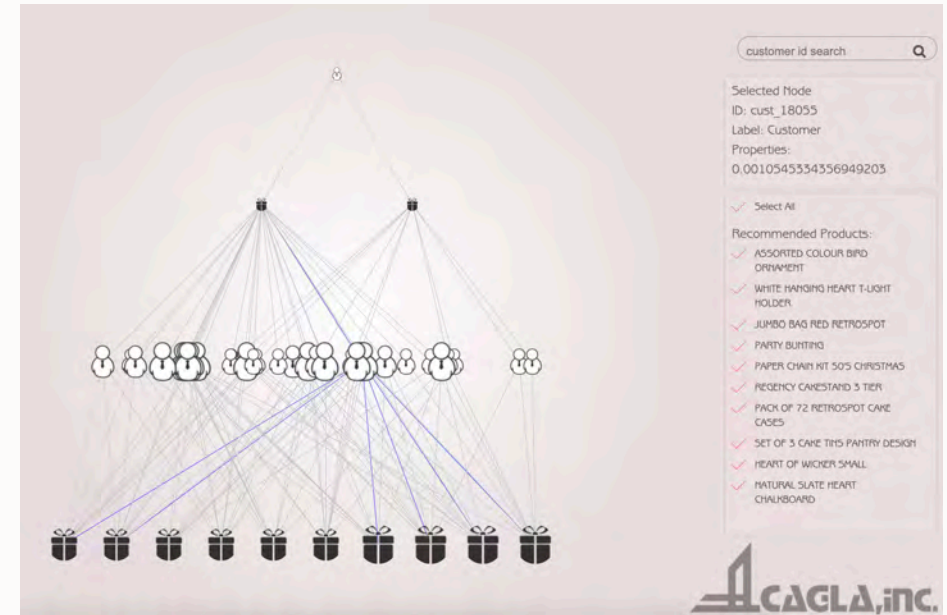


Mule account detection
(AskTOM - May 28, 2020)

Graph Algorithms and Analysis Flows

Pattern 2 - Enabling advanced queries

1. Select specific node(s)
 2. Run graph algorithms against the node(s)
 - personalized ranking (PPR, ...)
 - community detection
 - reachability
 3. Query to return the results to applications:
 - often real-time
- Use Cases
 - Reachability between devices (utility and communication), Recommendation for a customer (retail), Find other members in the same community (fraud and crime analysis)

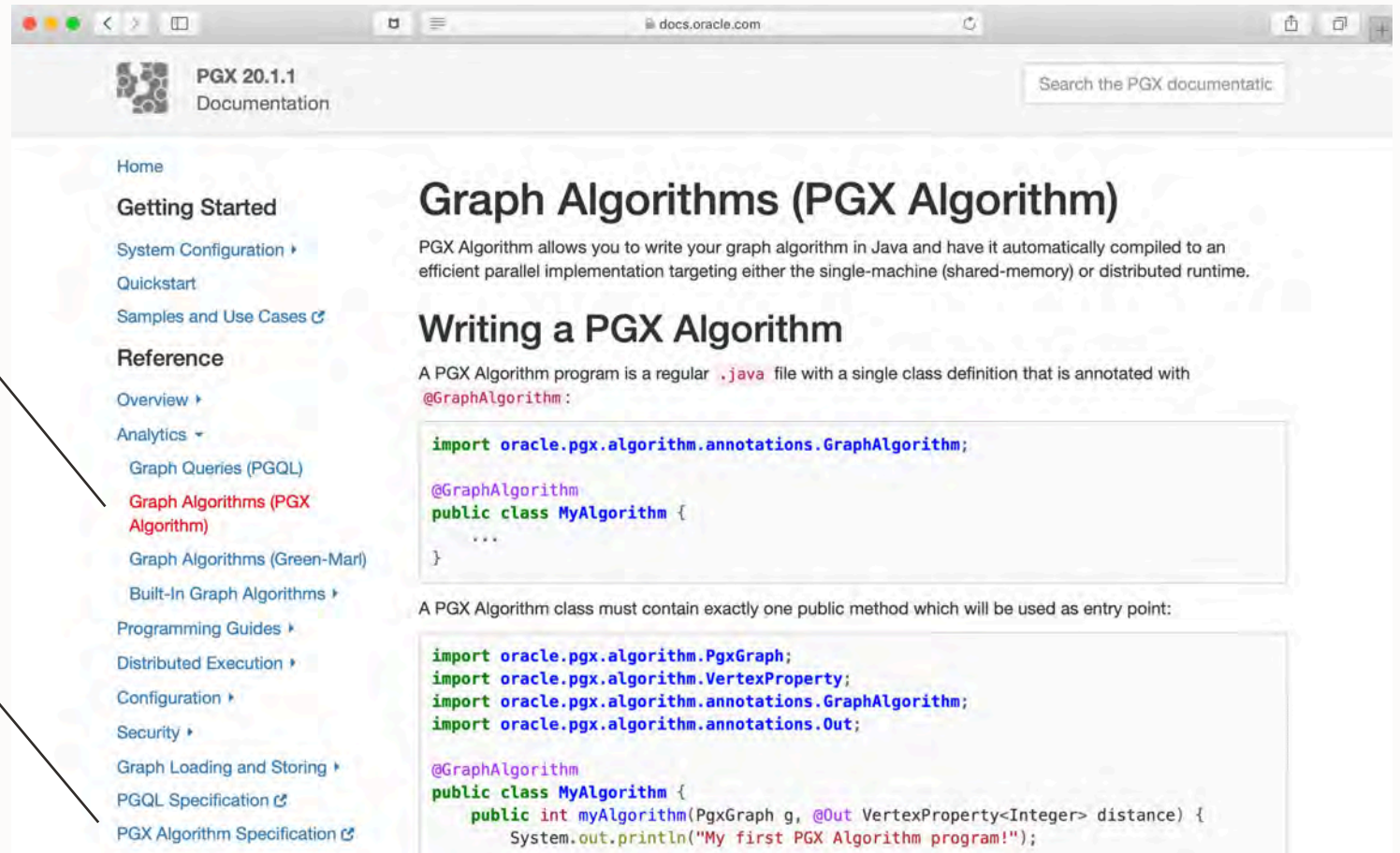


Real-time recommendation app
(AskTOM - July 2, 2020)

Custom Algorithms

PGX Algorithm is a Java-syntax language to write custom graph algorithms (Using Green-Marl is not supported)

PGX Algorithm **specification** is here. Also, each built-in algorithm page shows their implementations in PGX Algorithm.



PGX 20.1.1 Documentation

Search the PGX documentation

Home

Getting Started

- System Configuration ▶
- Quickstart
- Samples and Use Cases ↗

Reference

- Overview ▶
- Analytics ▶
 - Graph Queries (PGQL)
 - Graph Algorithms (PGX Algorithm)**
 - Graph Algorithms (Green-Marl)
 - Built-In Graph Algorithms ▶
- Programming Guides ▶
- Distributed Execution ▶
- Configuration ▶
- Security ▶
- Graph Loading and Storing ▶
- PGQL Specification ↗
- PGX Algorithm Specification ↗

Graph Algorithms (PGX Algorithm)

PGX Algorithm allows you to write your graph algorithm in Java and have it automatically compiled to an efficient parallel implementation targeting either the single-machine (shared-memory) or distributed runtime.

Writing a PGX Algorithm

A PGX Algorithm program is a regular `.java` file with a single class definition that is annotated with `@GraphAlgorithm`:

```
import oracle.pgx.algorithm.annotations.GraphAlgorithm;

@GraphAlgorithm
public class MyAlgorithm {
    ...
}
```

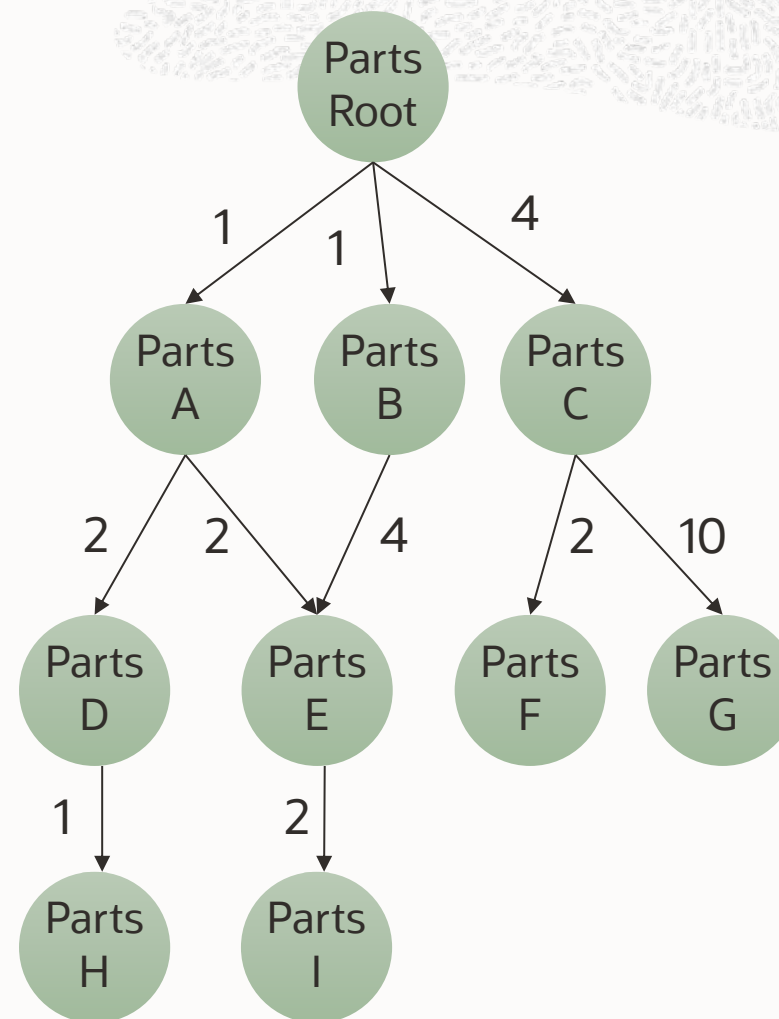
A PGX Algorithm class must contain exactly one public method which will be used as entry point:

```
import oracle.pgx.algorithm.PgxGraph;
import oracle.pgx.algorithm.VertexProperty;
import oracle.pgx.algorithm.annotations.GraphAlgorithm;
import oracle.pgx.algorithm.annotations.Out;

@GraphAlgorithm
public class MyAlgorithm {
    public int myAlgorithm(PgxGraph g, @Out VertexProperty<Integer> distance) {
        System.out.println("My first PGX Algorithm program!");
    }
}
```

Custom Algorithms

- Use case of custom algorithm in **Manufacturing BoM** (bill of materials):
- Some of the calculation processes can be optimized, implementing them as **stored programs** on Graph Server.
 - E.g. sum up the numbers of parts in a BoM tree.
 - One-time traversal from the root node (BFS: breath first search) calculates the numbers of all parts and the graph can keep the results as node properties.



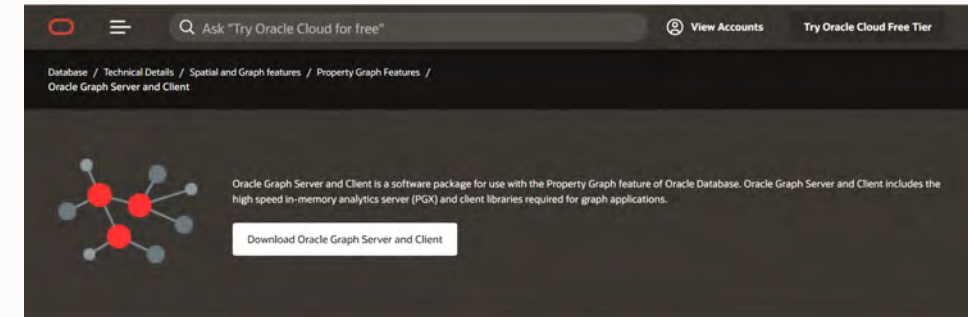
ID	num
A	1
B	1
C	4
D	2
E	6
F	8
G	40
H	2
I	12



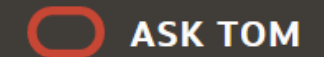
Helpful Links

- Graphs at Oracle
<https://www.oracle.com/goto/graph>
- Oracle Property Graph
<http://www.oracle.com/goto/propertygraph>
- Blog: Examples, Tips and Tricks
<http://bit.ly/OracleGraphBlog>
- AskTOM Series: <https://asktom.oracle.com/pls/apex/asktom.search?office=3084>
- Social Media
 - Twitter: @OracleBigData, @SpatialHannes, @Jeanlhm, @ryotaymnk
 - LinkedIn: Oracle Spatial and Graph Group
 - YouTube: youtube.com/c/OracleSpatialandGraph

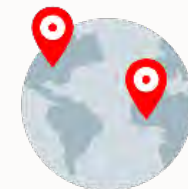
Search for "Oracle Graph Server and Client" to [download](#) from oracle.com



AskTOM Office Hours: Graph Database and Analytics



- Welcome to our AskTOM Graph Office Hours series!
We're back with new product updates, use cases, demos and technical tips
<https://asktom.oracle.com/pls/apex/asktom.search?oh=3084>
- Sessions will be held about once a month
- **Subscribe** at the page above for updates on upcoming session topics & dates
And submit feedback, questions, topic requests, and view past session recordings
- **Note: Spatial** now has a new Office Hours series for location analysis & mapping features in Oracle Database:
<https://asktom.oracle.com/pls/apex/asktom.search?oh=7761>



Our mission is to help people
see data in new ways, discover insights,
unlock endless possibilities.





ORACLE