ORACLE

Graph Analytics Using the Python API

Melli Annamalai and Ryota Yamanaka, Product Management, Oracle

October 29, 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

Agenda

1.	Recap – Graph Analytics and APIs	Melli
2.	Basic Operations of Python API	Melli
3.	Demo: Convert and Load Data from Database	Ryota
4.	Demo: Query, Run Algorithms, and Visualize	Ryota
5.	Demo: Combine with Machine Learning	Ryota

Melli

Nashua, New Hampshire, USA
@AnnamalaiMelli

Ryota

Bangkok, Thailand
@ryotaymnk

Recap – Graph Analytics and APIs

Graph Database and Analytics

Store, manage, query, and analyze graphs

- Enterprise capabilities: Built on Oracle infrastructure
- Manageability, fine-grained security, high availability, integration

Highly scalable

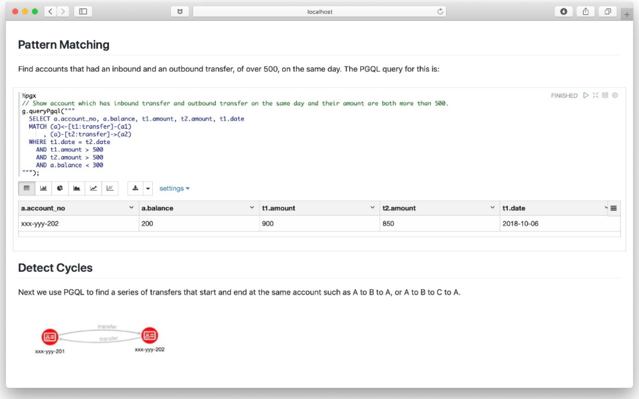
- In-memory query and analytics and in-database query
- 10s of billions of edges and vertices

PGQL: Powerful SQL-like graph query language

Analytics: 50+ pre-built graph analysis algorithms
Java and Python APIs

Visualization

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



Pattern Matching

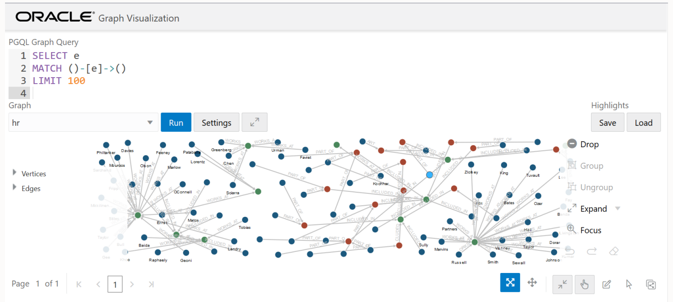
Find accounts that had an inbound and an outbound transfer, of over 500, on the same day. The PGQL query for this is:

```
%pgx
// Show account which has inbound transfer and outbound transfer on the same day and their amount are both more than 500.
g.queryPgql("""
  SELECT a.account_no, a.balance, t1.amount, t2.amount, t1.date
  MATCH (a)<-[t1:transfer]-(a1)
      , (a)-[t2:transfer]->(a2)
  WHERE t1.date = t2.date
    AND t1.amount > 500
    AND t2.amount > 500
    AND a.balance < 300
""");
```

a.account_no	a.balance	t1.amount	t2.amount	t1.date
xxx-yyy-202	200	900	850	2018-10-06

Detect Cycles

Next we use PGQL to find a series of transfers that start and end at the same account such as A to B to A, or A to B to C to A.



```
SELECT e
MATCH ()-[e]->()
LIMIT 100
```

Graph APIs and Clients

- **Java API** for PGQL queries and graph analytics

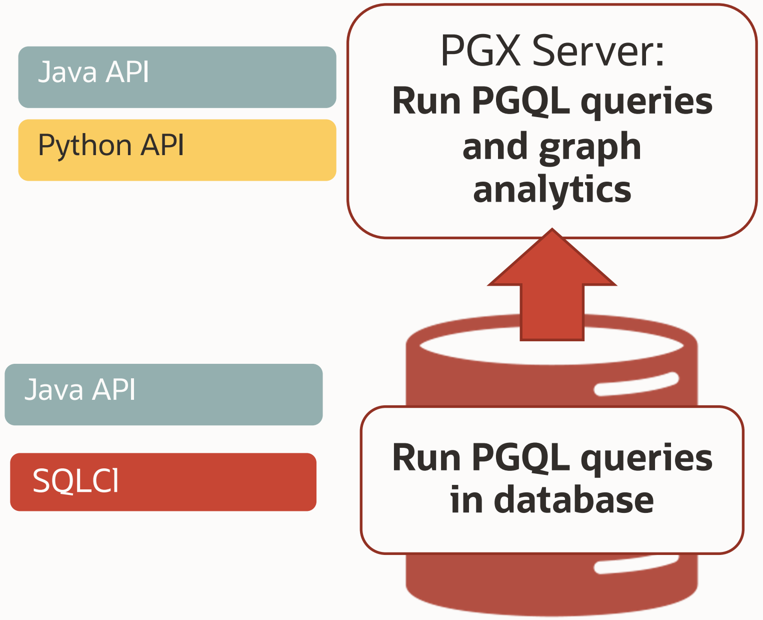
```
opg-jshell> session.queryPgql(“SELECT e from MATCH ()-[e]->()”)
opg-jshell> analyst.pagerank(my_graph)
```

  - Zeppelin notebook (PGX interpreters), Java application

- New in Oracle Graph Server and Client 20.4
  - **Python API** for PGQL queries and graph analytics in the in-memory graph server (PGX)

- **SQLcl** for PGQL queries (20.3 onward)

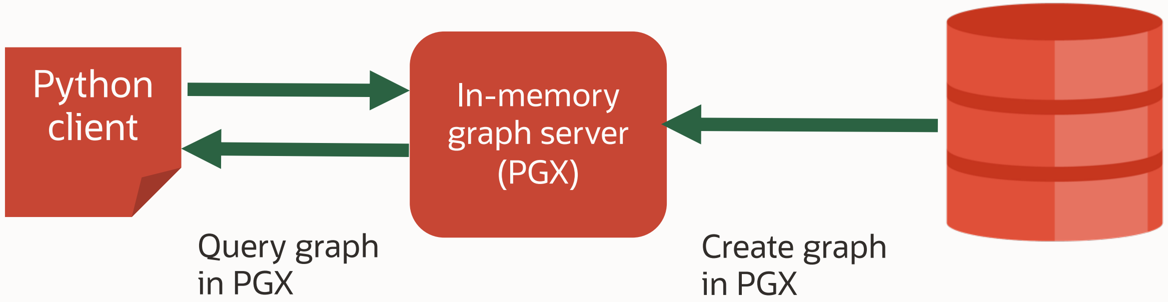
```
SQL> pgql auto on
PGQL> SELECT e from MATCH ()-[e]->();
```


Java API
Python API
PGX Server:
Run PGQL queries and graph analytics
Java API
SQLCl
Run PGQL queries in database

Why Python?

Enable data scientists to easily work with graphs

Integration with data science tools and environments



Python Client

- Python module is called **pypgx**
- Can be used
  - Interactively (Python shell), or
  - As module imported into a Python application
- Works with graphs in PGX
- If using PGQL-in-database
  - Use PGQL in SQLcl to create graph and run queries
  - Use Python API to load into PGX for analytics

Interactive Shells

- Graph Client includes shells for Java and Python

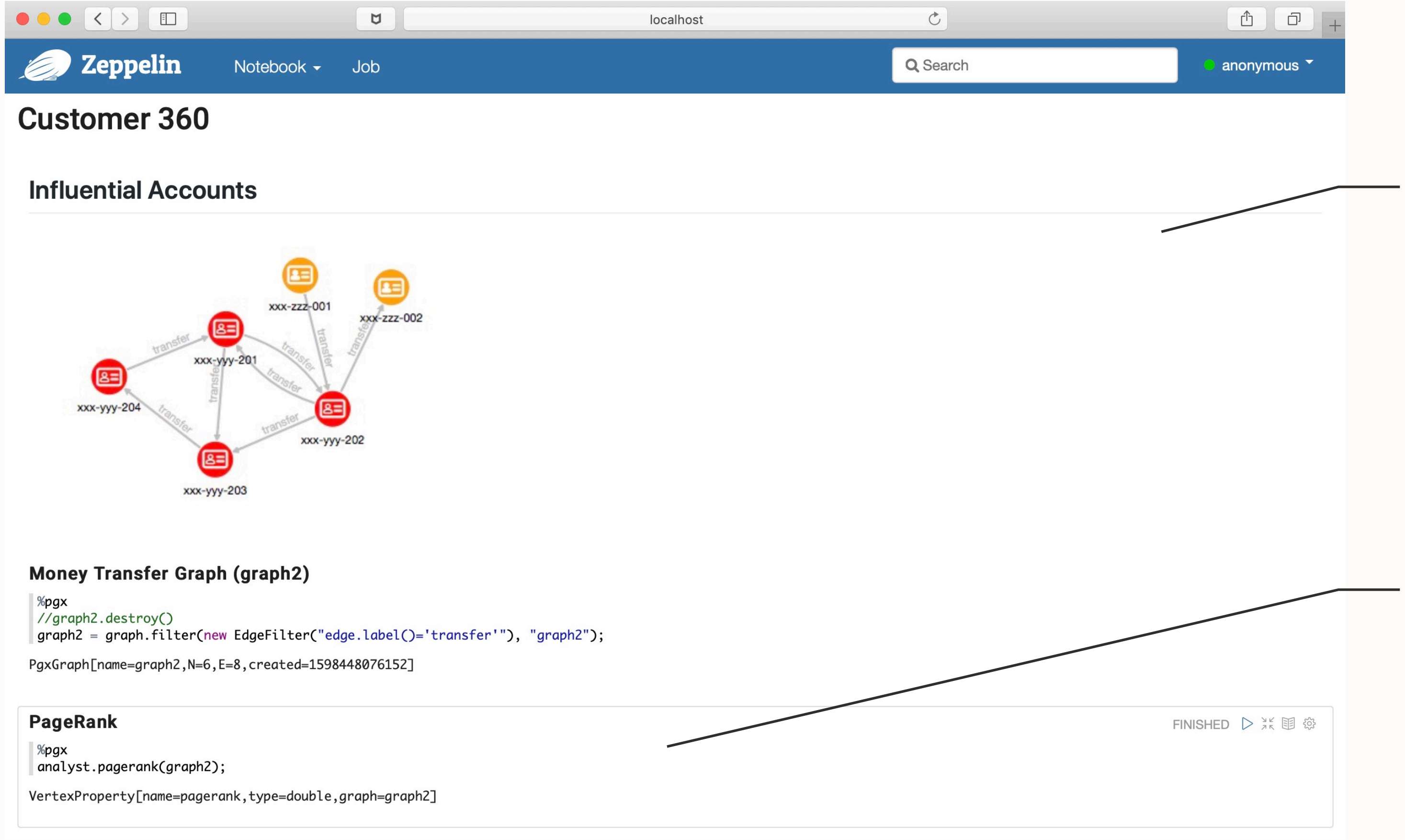
- To use Java API interactively, run opg-jshell:

```
$ ./bin/opg-jshell -b http://graph-server:7007 --username graph_dev
enter password for user graph_dev (press Enter for no password):
opg-jshell>
```

- To use Python API interactively, run opgpy:

```
$ ./bin/opgpy -b http://graph-server:7007 --username graph_dev
enter password for user graph_dev (press Enter for no password):
>>>
```

Zeppelin Notebook



Customer 360

Influential Accounts

Money Transfer Graph (graph2)

```
%pgx
//graph2.destroy()
graph2 = graph.filter(new EdgeFilter("edge.label()='transfer'"), "graph2");
```

```
PgxGraph[name=graph2,N=6,E=8,created=1598448076152]
```

PageRank

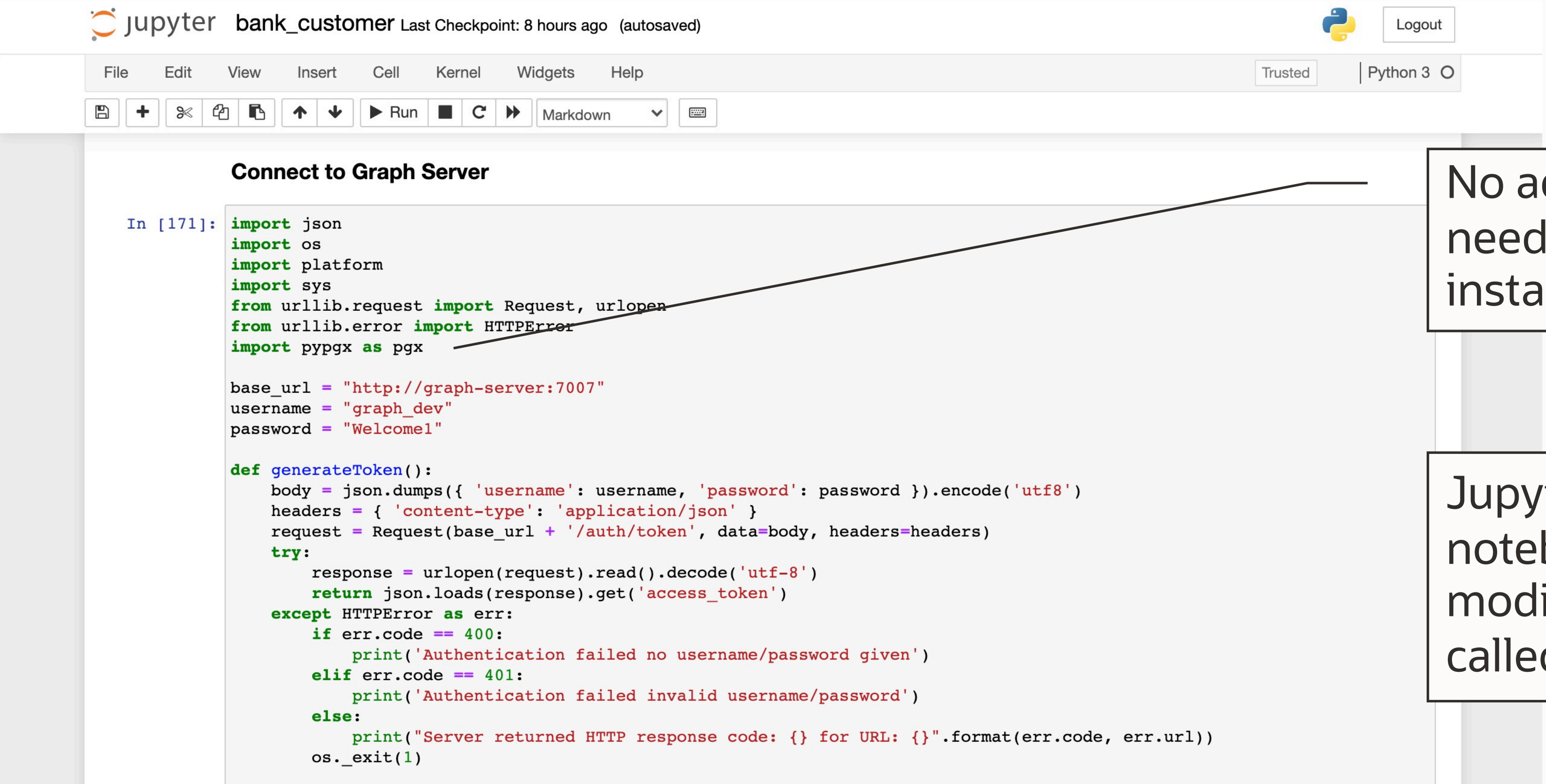
```
%pgx
analyst.pagerank(graph2);
```

```
VertexProperty[name=pagerank,type=double,graph=graph2]
```

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

Users can run PGQL queries and graph algorithms

Jupyter Notebook



Connect to Graph Server

```
In [171]: import json
import os
import platform
import sys
from urllib.request import Request, urlopen
from urllib.error import HTTPError
import pypgx as pgx

base_url = "http://graph-server:7007"
username = "graph_dev"
password = "Welcome1"

def generateToken():
    body = json.dumps({ 'username': username, 'password': password }).encode('utf8')
    headers = { 'content-type': 'application/json' }
    request = Request(base_url + '/auth/token', data=body, headers=headers)
    try:
        response = urlopen(request).read().decode('utf-8')
        return json.loads(response).get('access_token')
    except HTTPError as err:
        if err.code == 400:
            print('Authentication failed no username/password given')
        elif err.code == 401:
            print('Authentication failed invalid username/password')
        else:
            print("Server returned HTTP response code: {} for URL: {}".format(err.code, err.url))
        os._exit(1)
```

No additional interpreter is needed. But **pypgx** should be installed on the notebook server.

Jupyter is a popular open-source notebook interface (under modified BSD license, formally called IPython)

Basic Operations of Python API

Installation of Python API

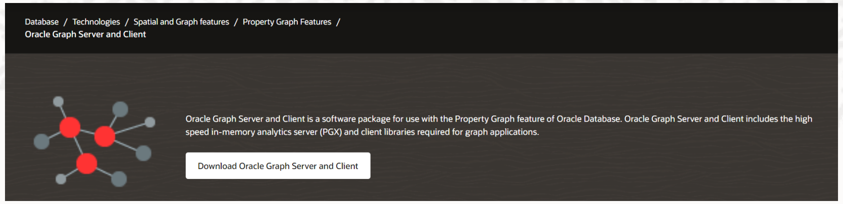
- Python version 3.5 or later is required

```
$ python3 --version
Python 3.6.1
```

- JDK 8 or later is required

```
$ java --version
java 11.0.8 2020-07-14 LTS
```

Installation of Python API



Database / Technologies / Spatial and Graph features / Property Graph Features /
Oracle Graph Server and Client

Oracle Graph Server and Client is a software package for use with the Property Graph feature of Oracle Database. Oracle Graph Server and Client includes the high speed in-memory analytics server (PGX) and client libraries required for graph applications.

- Download Oracle Graph Client zip file from: https://www.oracle.com/database/technologies/spatialandgraph/property-graph-features/graph-server-and-client/graph-server-and-client-downloads.html

- Install the required dependencies

```
$ pip3 install Cython six pyjnius
```

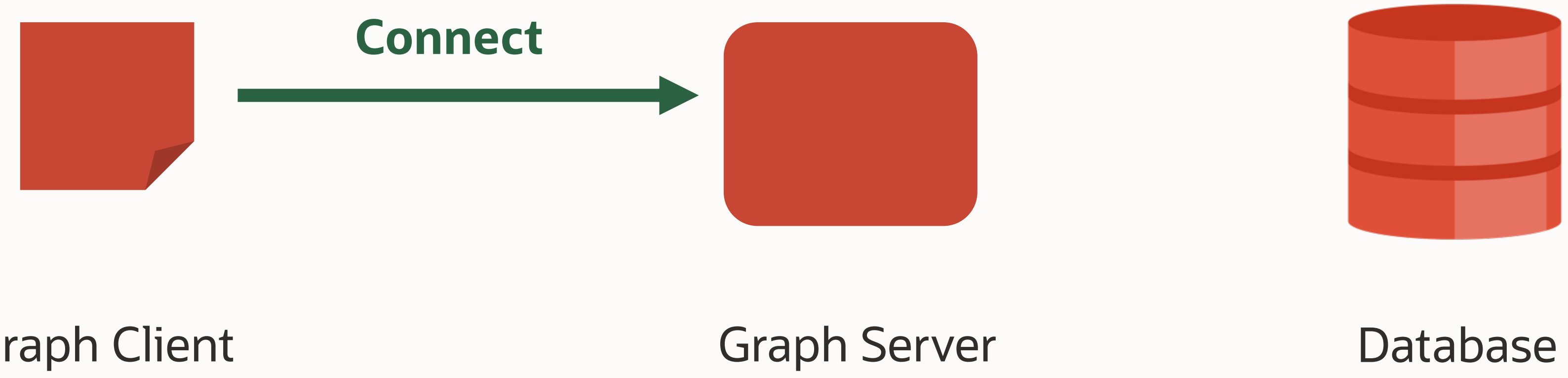
- Install the client

```
$ pip3 install oracle-graph-client-20.4.0.zip
```

Connect to Graph Server

- Connect to Graph Server using the interactive shell
  - Authenticate as a database

```
$ ./bin/opgpy -b https://graph-server:7007 --username graph_dev
```



Connect

Graph Client

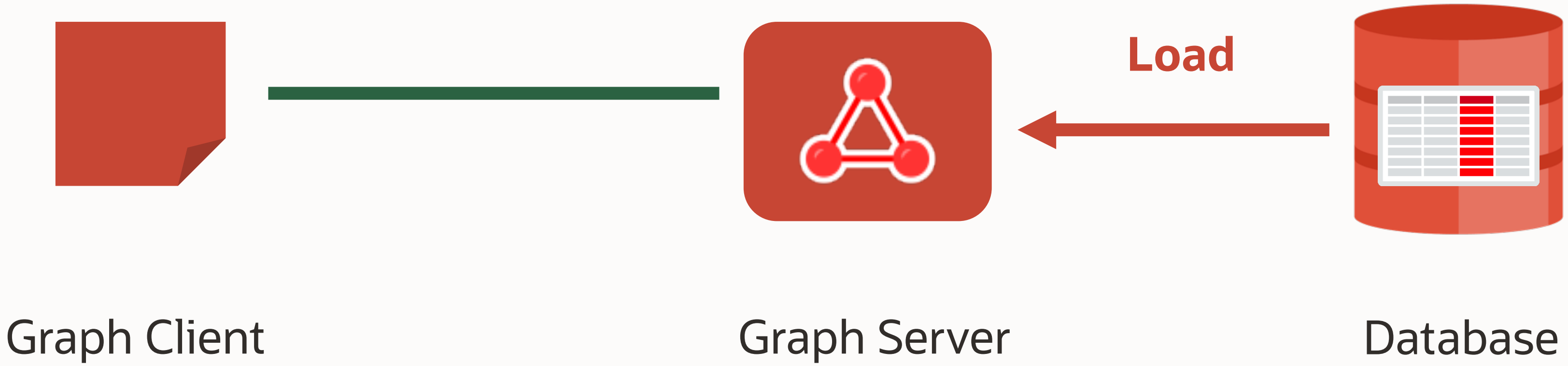
Graph Server

Database

Create Graph using DDL

- Load data from database tables and create graph in graph server

```
>>> statement = 'CREATE PROPERTY GRAPH "My Graph" ...'
>>> session.prepare_pgql(statement).execute()
```



Load

Graph Client

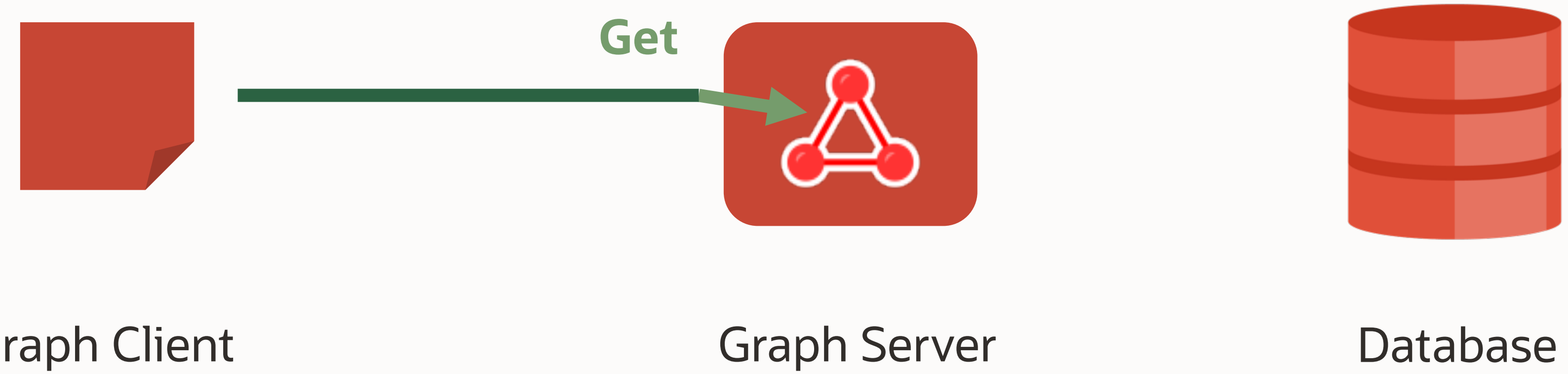
Graph Server

Database

Get Graph on Graph Server

- Use graphs already loaded on Graph Server

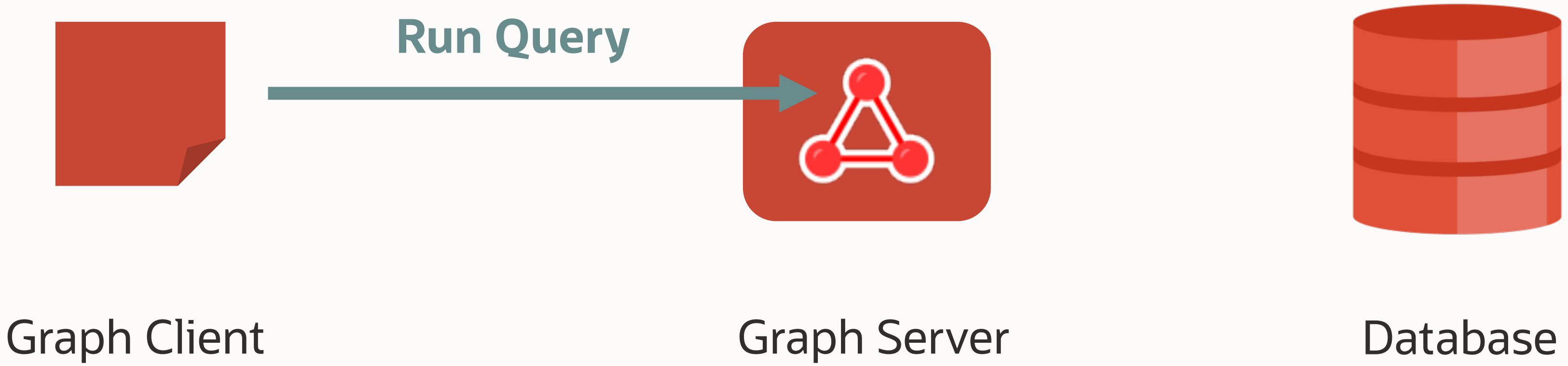
```
>>> my_graph = session.get_graph("My Graph")
```



Run PGQL Queries

- Run Query against the graph and retrieve the result as result_set

```
>>> my_graph.query_pgql("SELECT m.name FROM MATCH (n)-[:likes]->(m) ...")
```



Run Query

Graph Client

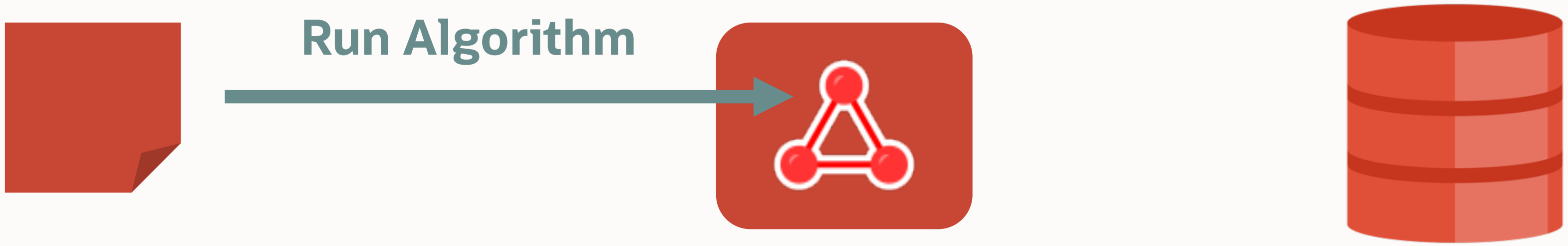
Graph Server

Database

Run Graph Algorithms

- Run Query against the graph and update the graph

```
>>> analyst = session.create_analyst()
>>> result = analyst.pagerank(my_graph);
```



Demo: Convert and Load Data from Database

Setup Quickstart Environment

Installation: https://github.com/ryotayamanaka/oracle-pg/tree/20.4

Clone repository (Note, the branch is **20.4**)

```
$ git clone https://github.com/ryotayamanaka/oracle-pg.git -b 20.4
```

Download and extract packages (Note, the packages are version **20.4**)

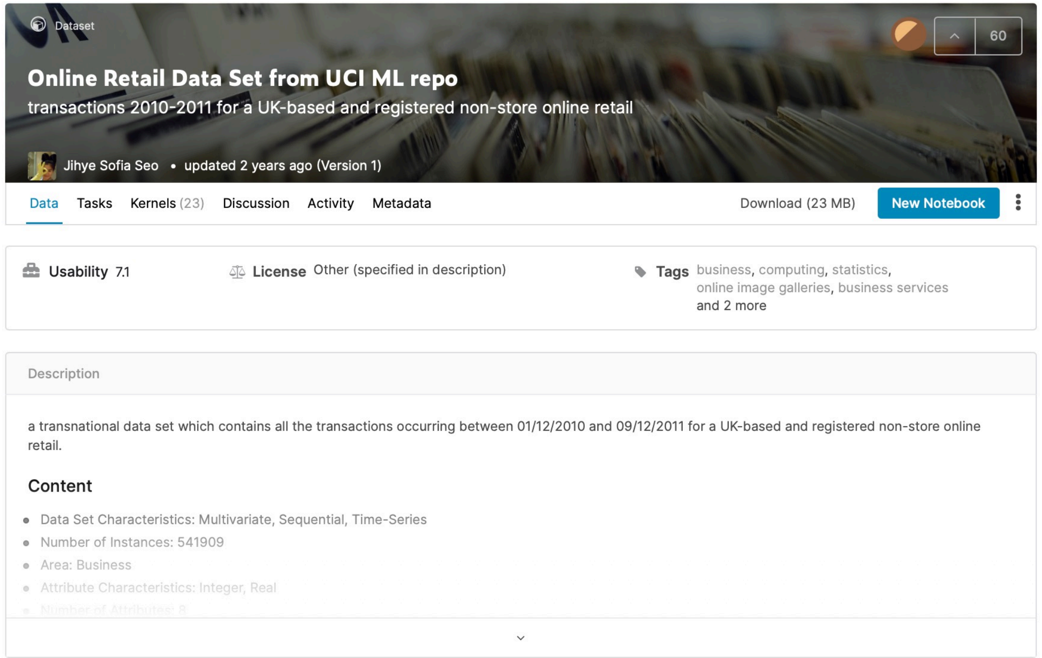
```
$ sh extract.sh
```

Build and start the docker containers

```
$ docker-compose up
```

Example - Product Purchase Dataset

- Online Retail dataset (Kaggle)
  - 4,339 customers
  - 3,919 products
  - 396,370 purchases



Online Retail Data Set from UCI ML repo

transactions 2010-2011 for a UK-based and registered non-store online retail

Jihye Sofia Seo • updated 2 years ago (Version 1)

Usability 7.1

License Other (specified in description)

Tags business, computing, statistics, online image galleries, business services and 2 more

Description

a transnational data set which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail.

Content

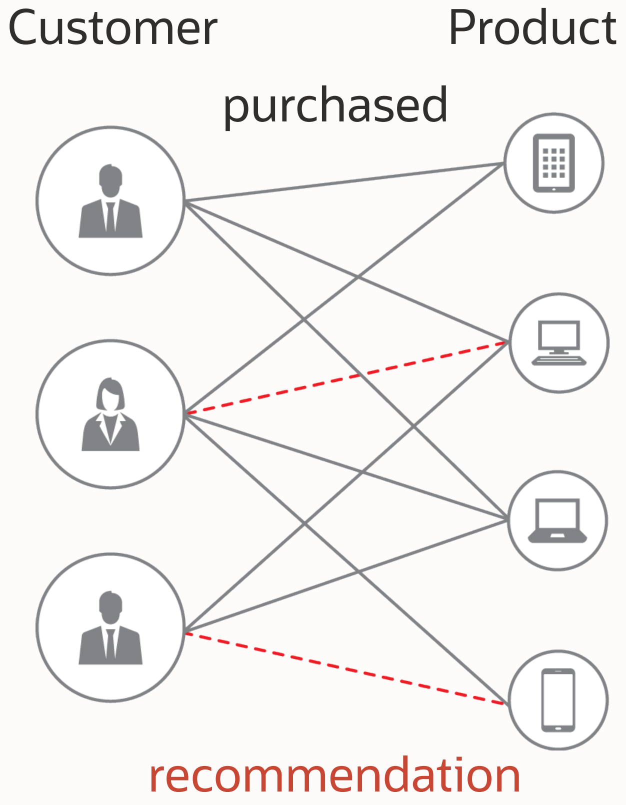
- Data Set Characteristics: Multivariate, Sequential, Time-Series
- Number of Instances: 541909
- Area: Business
- Attribute Characteristics: Integer, Real
- Number of Attributes: 8

https://www.kaggle.com/jihyeseo/online-retail-data-set-from-uci-ml-repo

Example - Product Recommendation in Online Retail

When you need to analyze the product purchase activity of your customers and generate recommendation, your graph should contain the following information:

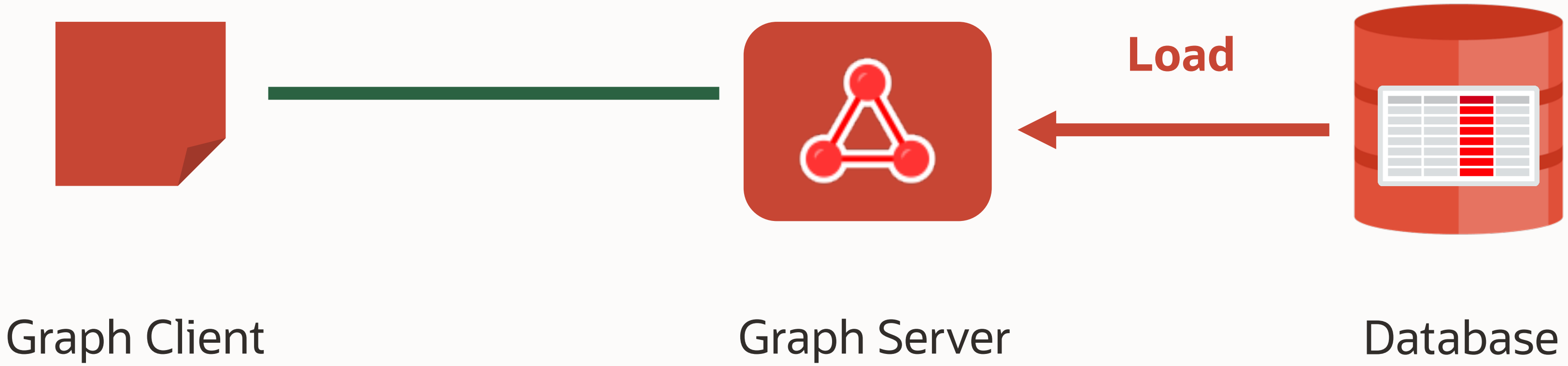
- **Customer** entities
- **Product** entities
- **Purchased** relationships



Convert and Load Data from Database

- Load data from database tables and create graph in graph server

```
>>> statement = 'CREATE PROPERTY GRAPH "Online Retail" ...'
>>> session.prepare_pgql(statement).execute()
```



CREATE PROPERTY GRAPH Statement

- Definition of vertices

```
CREATE PROPERTY GRAPH "Online Retail"
  VERTEX TABLES (
    online_retail.customers
      LABEL "Customer"
      PROPERTIES (customer_id AS "customer_id", "country")
  , online_retail.products
      LABEL "Product"
      PROPERTIES (stock_code AS "stock_code", "description")
  )
  EDGE TABLES (
    ...
  )
```

CREATE PROPERTY GRAPH Statement

- Definition of edges

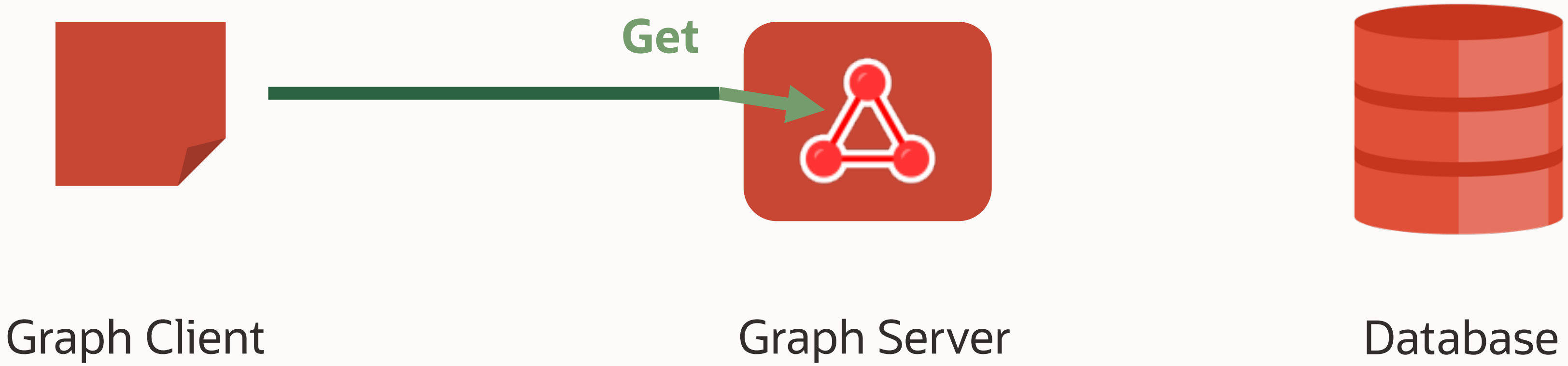
```
CREATE PROPERTY GRAPH "Online Retail"
  VERTEX TABLES (
    ...
  )
  EDGE TABLES (
    online_retail.purchases_distinct
      KEY (purchase_id)
      SOURCE KEY(customer_id) REFERENCES customers
      DESTINATION KEY(stock_code) REFERENCES products
      LABEL "has_purchased"
      PROPERTIES (purchase_id)
)
```

Demo: Query, Run Algorithms, and Visualize

Get Graph on Graph Server

- Attach the graphs already loaded on Graph Server

```
>>> graph = session.get_graph("Online Retail")
```



Get

Graph Client

Graph Server

Database

Run Algorithm

For calculating the recommended products for a specific customer "cust_12353", we use **personalize pagerank** algorithm here.

Firstly, get the vertex object represents the customer.

```
rs = graph.query_pgql(
       "SELECT ID(c) FROM MATCH (c) WHERE c.customer_id = 'cust_12353'")
vertex = graph.get_vertex(rs.get_row(0))
```

Run the algorithm. The result is stored as "ppr" new node property.

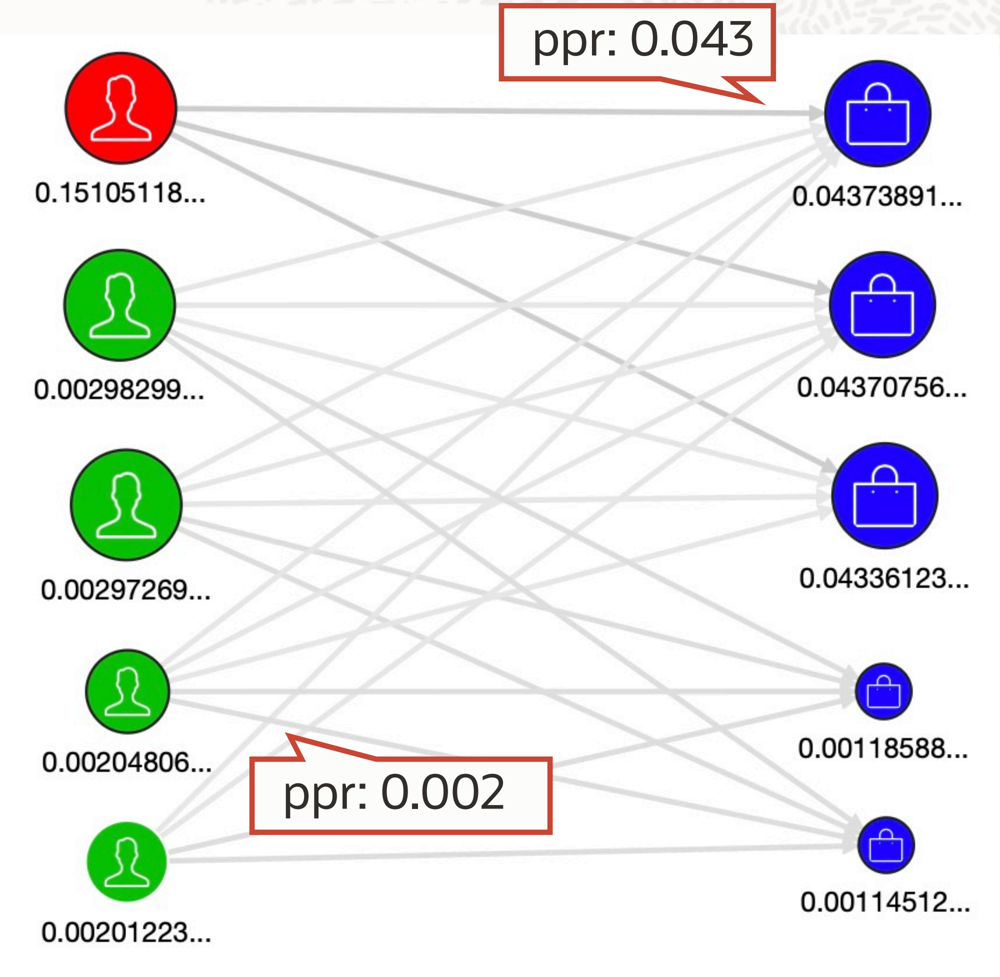
```
analyst.personalized_pagerank(graph, vertex, rank="ppr")
```

Query the Algorithm Results

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

Users can access the results using PGQL queries:

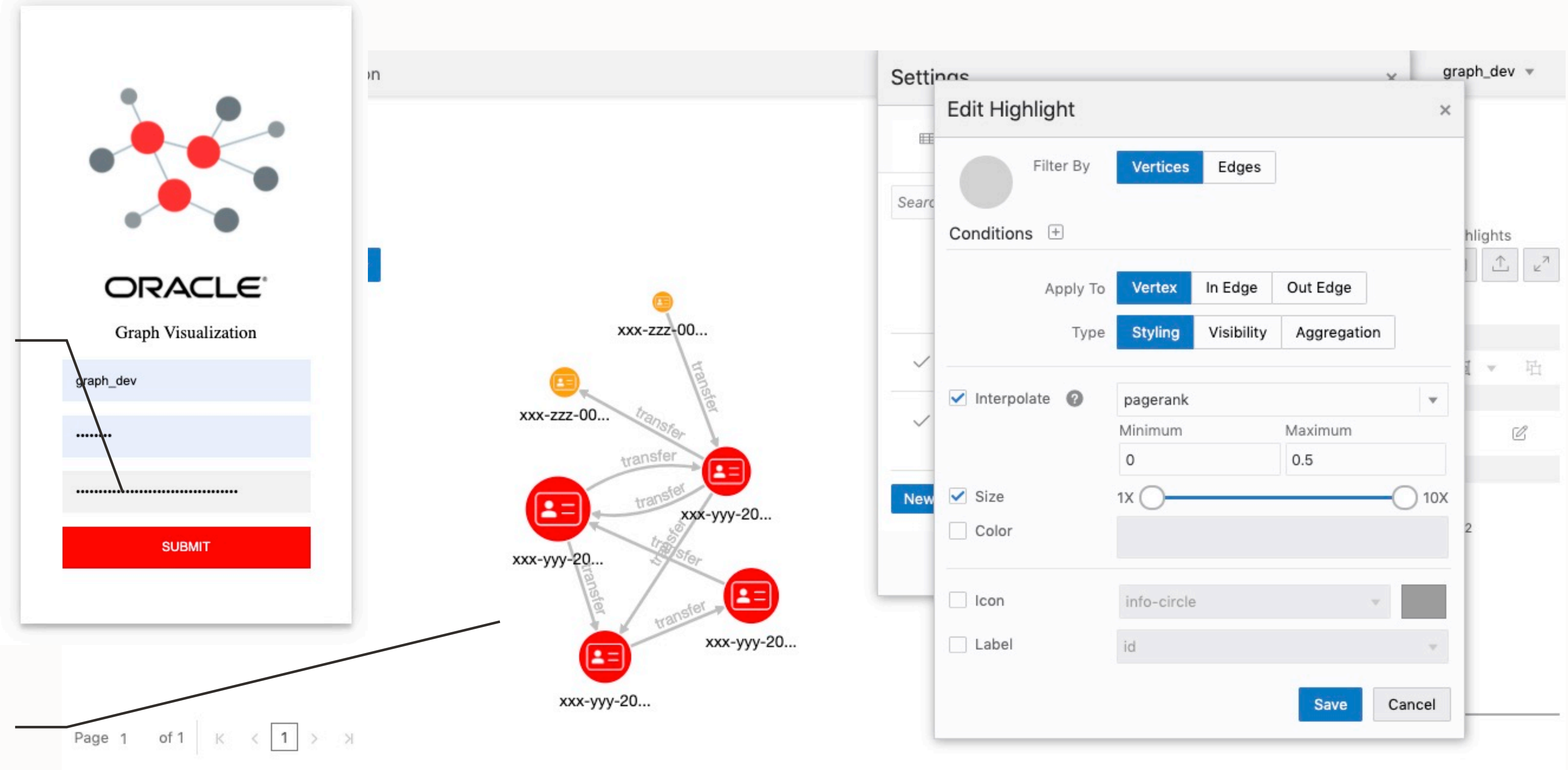
```
SELECT p.description, p.ppr
FROM MATCH (p)
ORDER BY p.ppr DESC
```



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.



ORACLE
Graph Visualization

graph_dev

....

....

SUBMIT

Edit Highlight

Filter By: Vertices | Edges

Conditions

Apply To: Vertex | In Edge | Out Edge

Type: Styling | Visibility | Aggregation

☑ Interpolate: pagerank

Minimum: 0 Maximum: 0.5

☑ Size: 1X — 10X

□ Color

□ Icon: info-circle

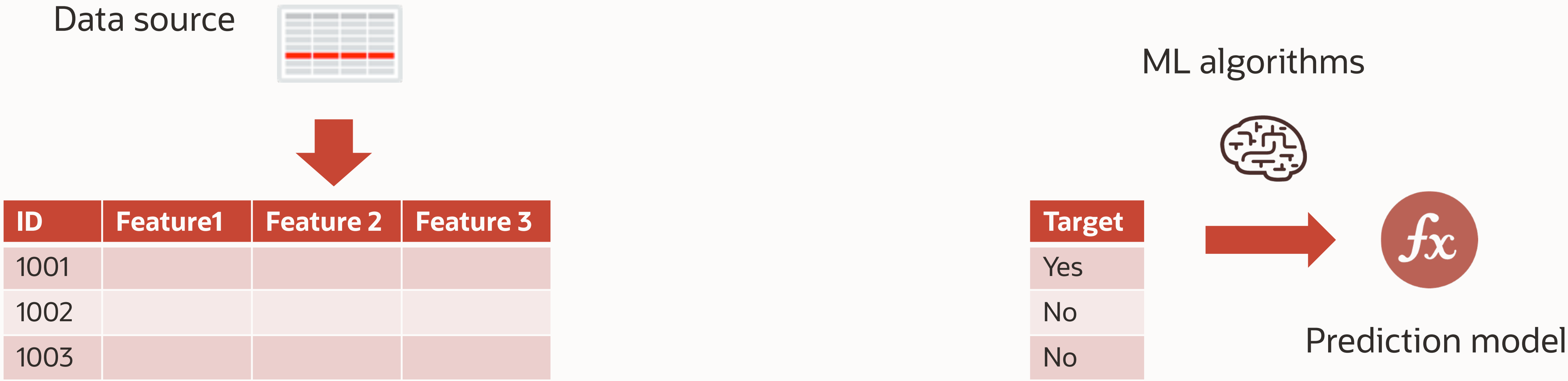
□ Label: id

Save Cancel

Demo: Combine with Machine Learning

How Graph Enhances Machine Learning

- **Information coverage** of the training dataset is important to make good predictive models
- New features are generated based on relationships using **graph queries** and **graph algorithms**

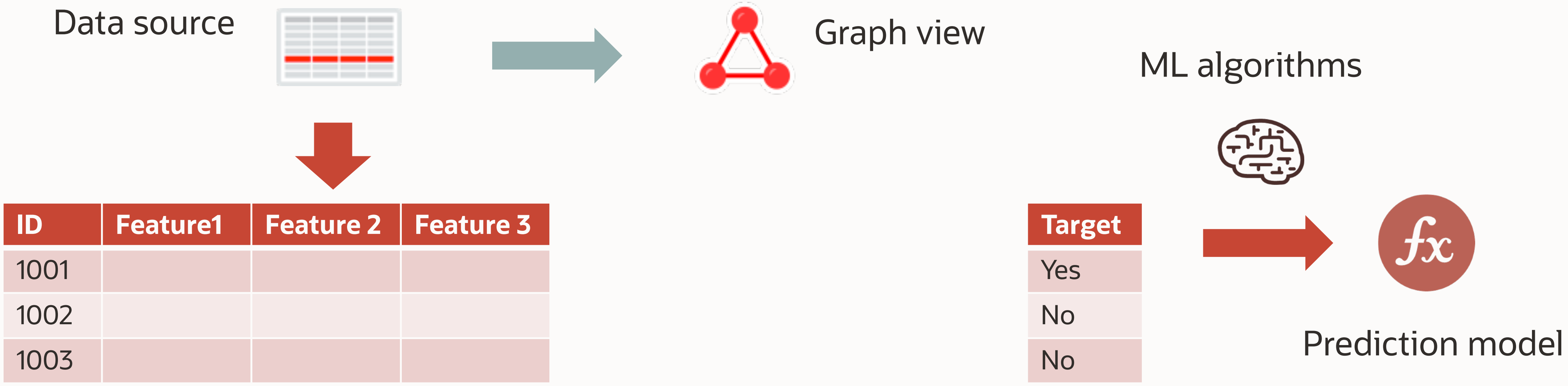


ID	Feature1	Feature 2	Feature 3
1001			
1002			
1003			

Target
Yes
No
No

How Graph Enhances Machine Learning

- **Information coverage** of the training dataset is important to make good predictive models
- New features are generated based on relationships using **graph queries** and **graph algorithms**

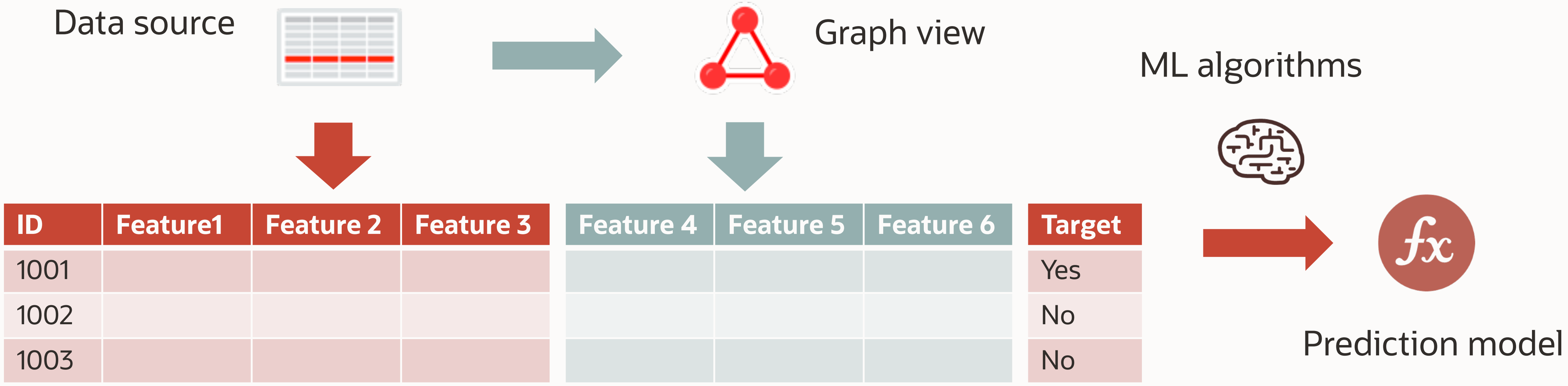


ID	Feature1	Feature 2	Feature 3
1001			
1002			
1003			

Target
Yes
No
No

How Graph Enhances Machine Learning

- **Information coverage** of the training dataset is important to make good predictive models
- New features are generated based on relationships using **graph queries** and **graph algorithms**



Data source

Graph view

ML algorithms

Prediction model

ID	Feature1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6	Target
1001							Yes
1002							No
1003							No

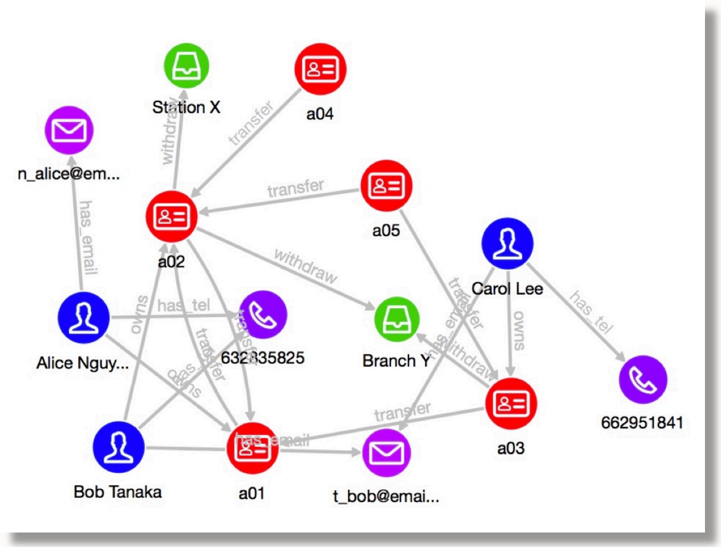
Example - Mule Account Detection

- **Mule accounts** are often stolen accounts and transfers money illegally
- Question:
  Is it possible to generate **more features** based on the relationships between accounts (e.g. transaction patterns, family relationships, ...) ?

Account

name
age
occupation
branch
balance
...

is_fraud

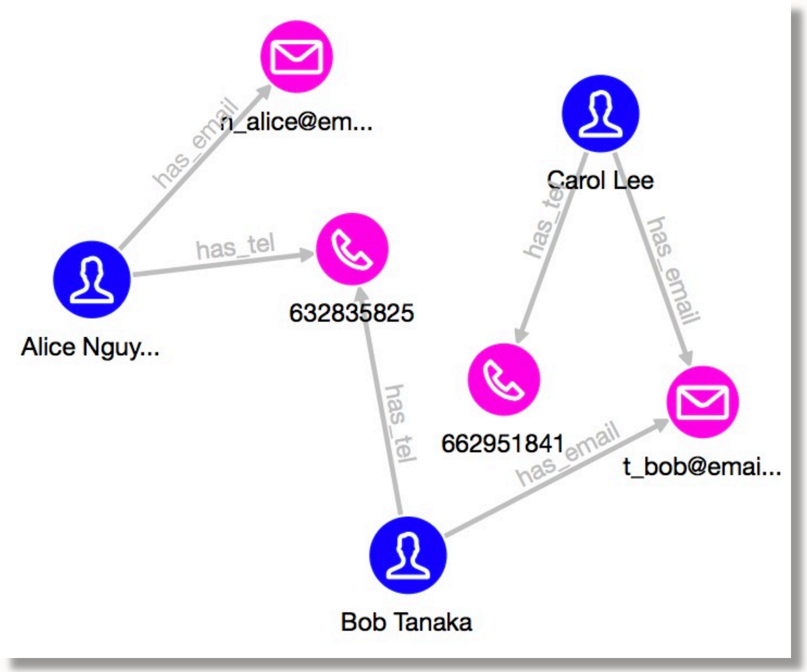


Example - Mule Account Detection

Feature 1

- If the owner of this account is sharing personal information with others

```
SELECT a1, COUNT(s)
MATCH (a1)<-[:owns]-(c1)-(s)-(c2)-[:owns]->(a2)
WHERE a1 != a2 AND c1 != c2
GROUP BY a1
```



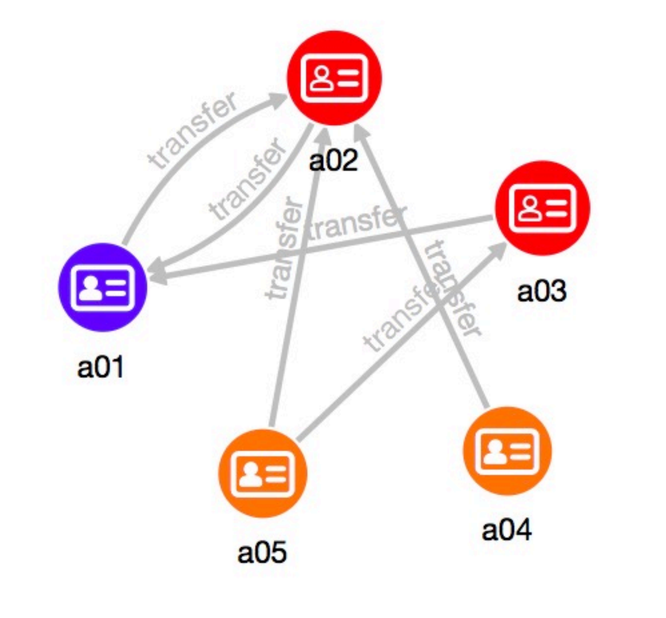
Example - Mule Account Detection

Feature 2

- How many fraud accounts exist in the same money transfer community

```
analyst.scc_kosaraju(G)
```

```
SELECT a.scc_kosaraju, COUNT(a)
MATCH (a)
WHERE a.type = 'Account'
  AND a.is_fraud = 'true'
GROUP BY a.scc_kosaraju
```



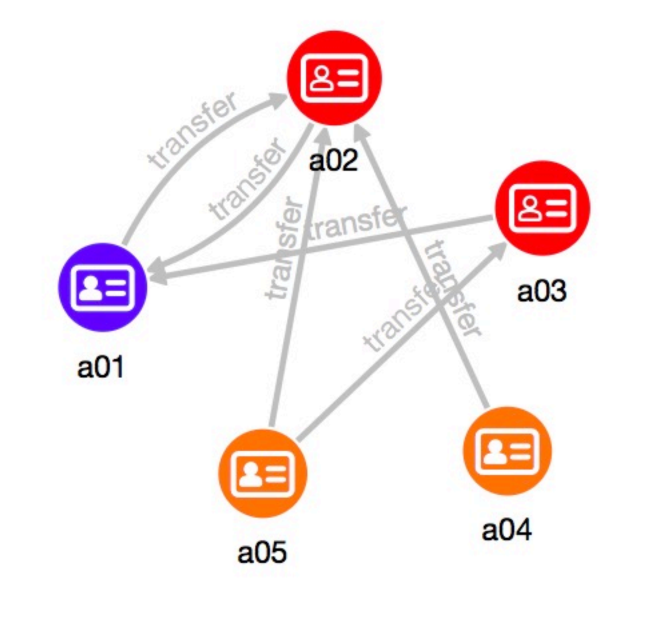
Example - Mule Account Detection

Feature 3

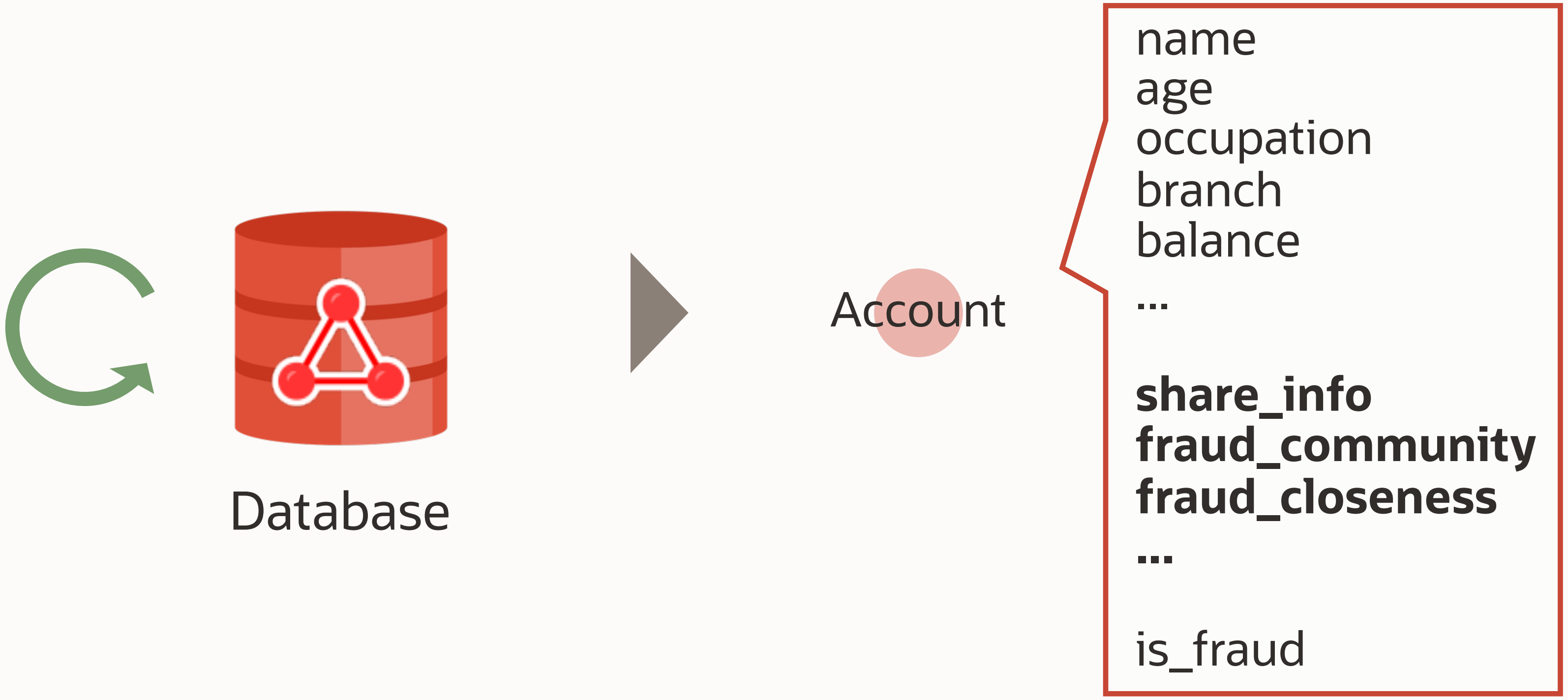
- Closeness to know fraud accounts

```
analyst.personalized_pagerank(G, fraud_accounts)
```

```
SELECT a.account_no, a.pagerank, a.is_fraud
MATCH (a) WHERE a.type = 'Account'
ORDER BY a.pagerank DESC
```



Example - Mule Account Detection



Database

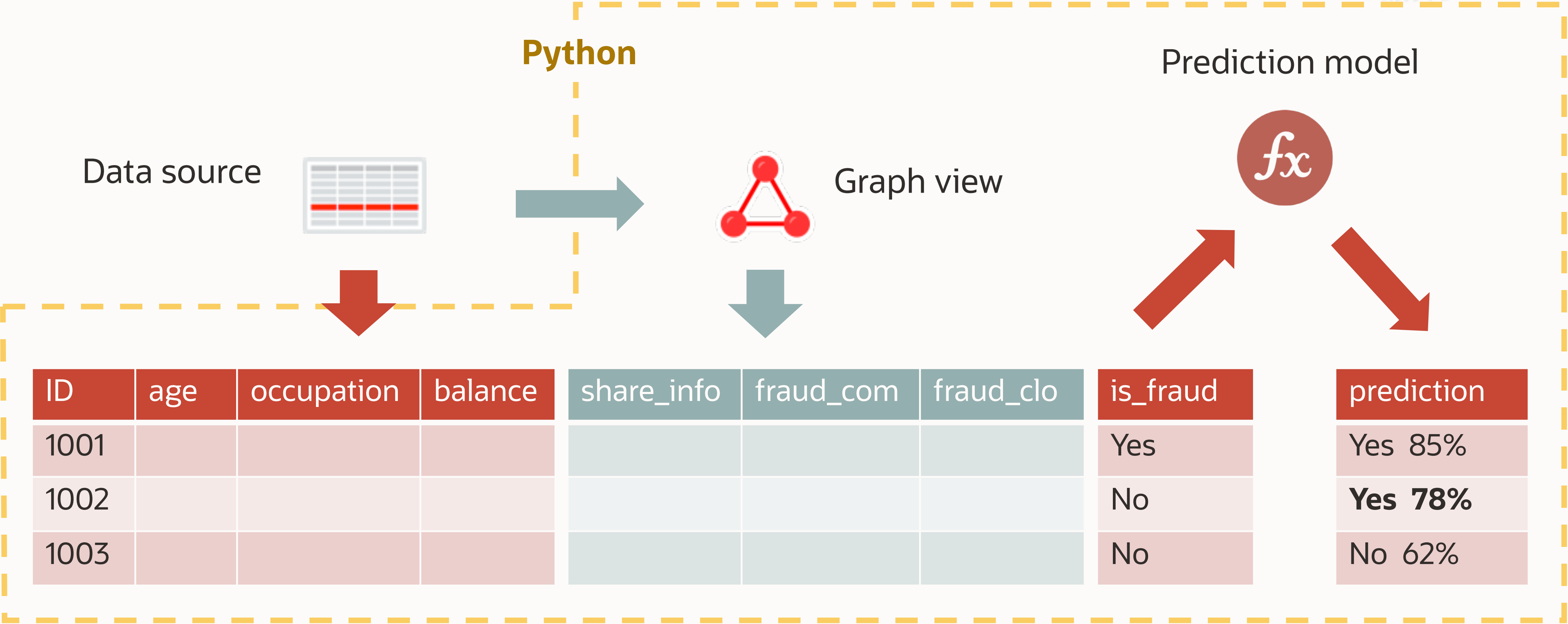
Account

name
age
occupation
branch
balance
...

share_info
fraud_community
fraud_closeness
...

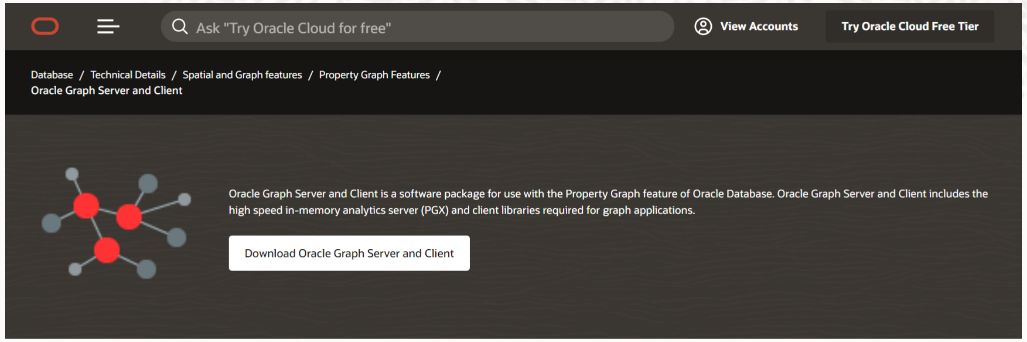
is_fraud

Example - Mule Account Detection



ID	age	occupation	balance	share_info	fraud_com	fraud_clo	is_fraud	prediction
1001							Yes	Yes 85%
1002							No	**Yes 78%**
1003							No	No 62%

Helpful Links


Ask "Try Oracle Cloud for free"
View Accounts
Try Oracle Cloud Free Tier
Download Oracle Graph Server and Client

Database / Technical Details / Spatial and Graph features / Property Graph Features /
Oracle Graph Server and Client

Oracle Graph Server and Client is a software package for use with the Property Graph feature of Oracle Database. Oracle Graph Server and Client includes the high speed in-memory analytics server (PGX) and client libraries required for graph applications.

- Oracle Property Graph: http://www.oracle.com/goto/propertygraph
- Hands-on (using Oracle Cloud)
  - with OCI Marketplace instance https://apexapps.oracle.com/pls/apex/dbpm/r/livelabs/view-workshop?p180_id=686
  - manual install of Graph Server and Client https://apexapps.oracle.com/pls/apex/dbpm/r/livelabs/view-workshop?p180_id=687
- Social Media
  - Twitter: @OracleBigData, @SpatialHannes, @JeanIhm, @ryotaymnk, @AnnamalaiMelli
  - LinkedIn: Oracle Spatial and Graph Group
  - YouTube: youtube.com/c/OracleSpatialandGraph
  - Blog - Examples, Tips and Tricks: http://bit.ly/OracleGraphBlog

AskTOM Office Hours: Graph Database and Analytics

ASK TOM

- 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