



Machine Learning with Oracle Graph

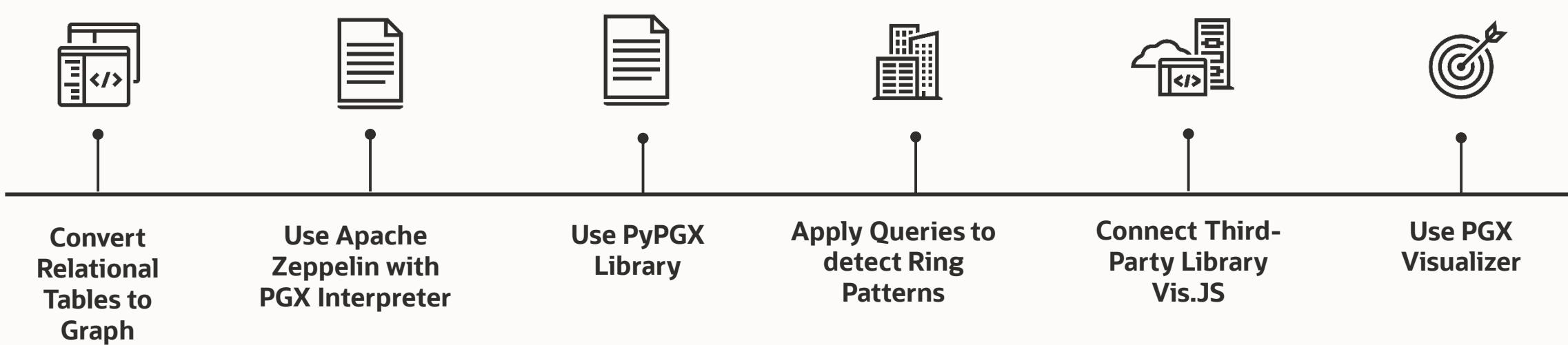
Natwest Developer Session

—
March 2021

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.

Developer Journey Map



Oracle Environment

Users	Jupyter URL	Zeppelin URL
Workshop001 to workshop020	https://bdabastion.sceceps.com/wsXXX	https://bdabastion.sceceps.com/zp1/
Workshop021 to workshop040	https://bdabastion.sceceps.com/wsXXX	https://bdabastion.sceceps.com/zp2/
Workshop041 to workshop060	https://bdabastion.sceceps.com/wsXXX	https://bdabastion.sceceps.com/zp3/
Workshop061 to workshop080	https://bdabastion.sceceps.com/wsXXX	https://bdabastion.sceceps.com/zp4/
Workshop081 to workshop100	https://bdabastion.sceceps.com/wsXXX	https://bdabastion.sceceps.com/zp5/

pwd: welcome1

Environment to download Documentation / Scripts

GitHub Repository: <https://github.com/operard/mlgraph>

Documentation for Natwest workshop: <https://github.com/operard/mlgraph/tree/main/doc>

Documentation of Oracle Graph:

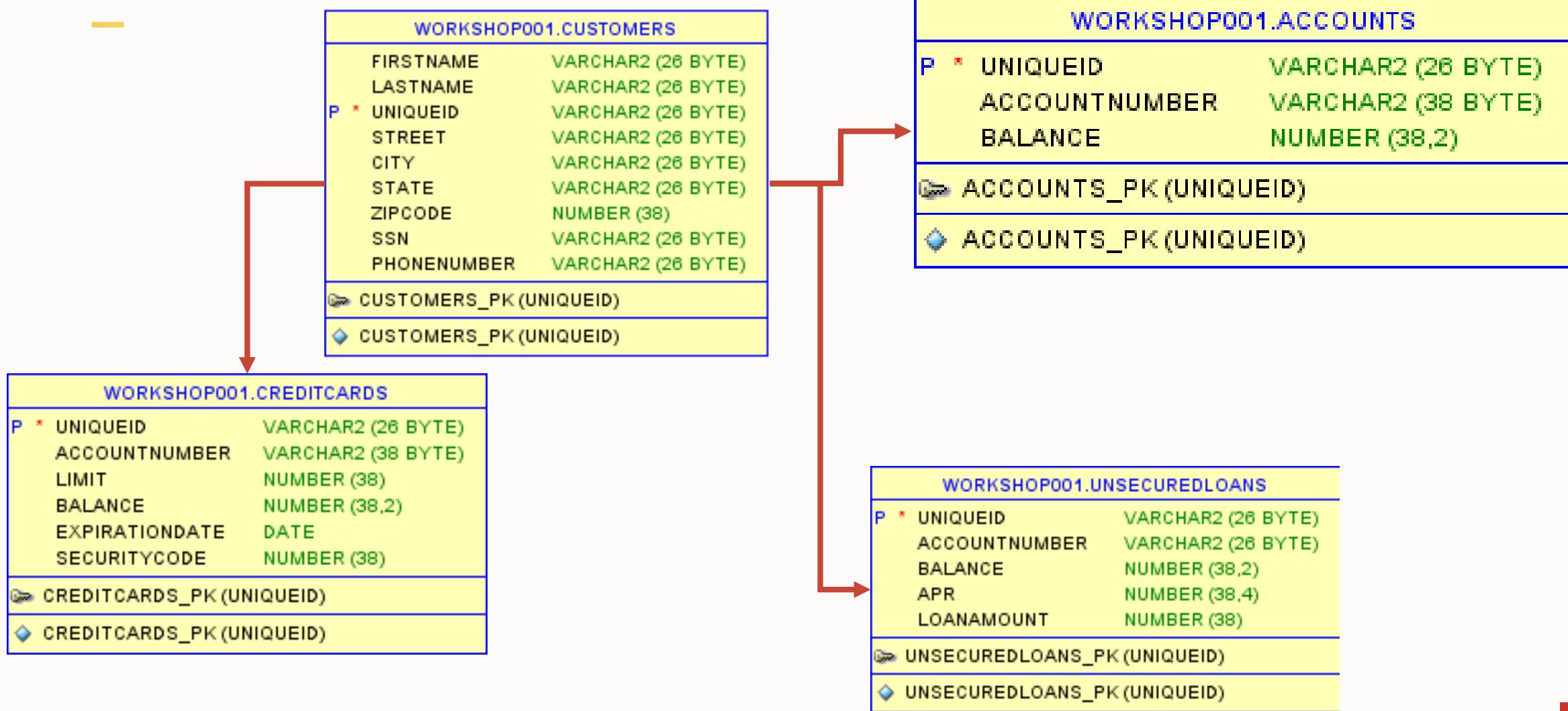
https://docs.oracle.com/cd/E56133_01/latest/proc-guides/index.html

<https://pgql-lang.org/>

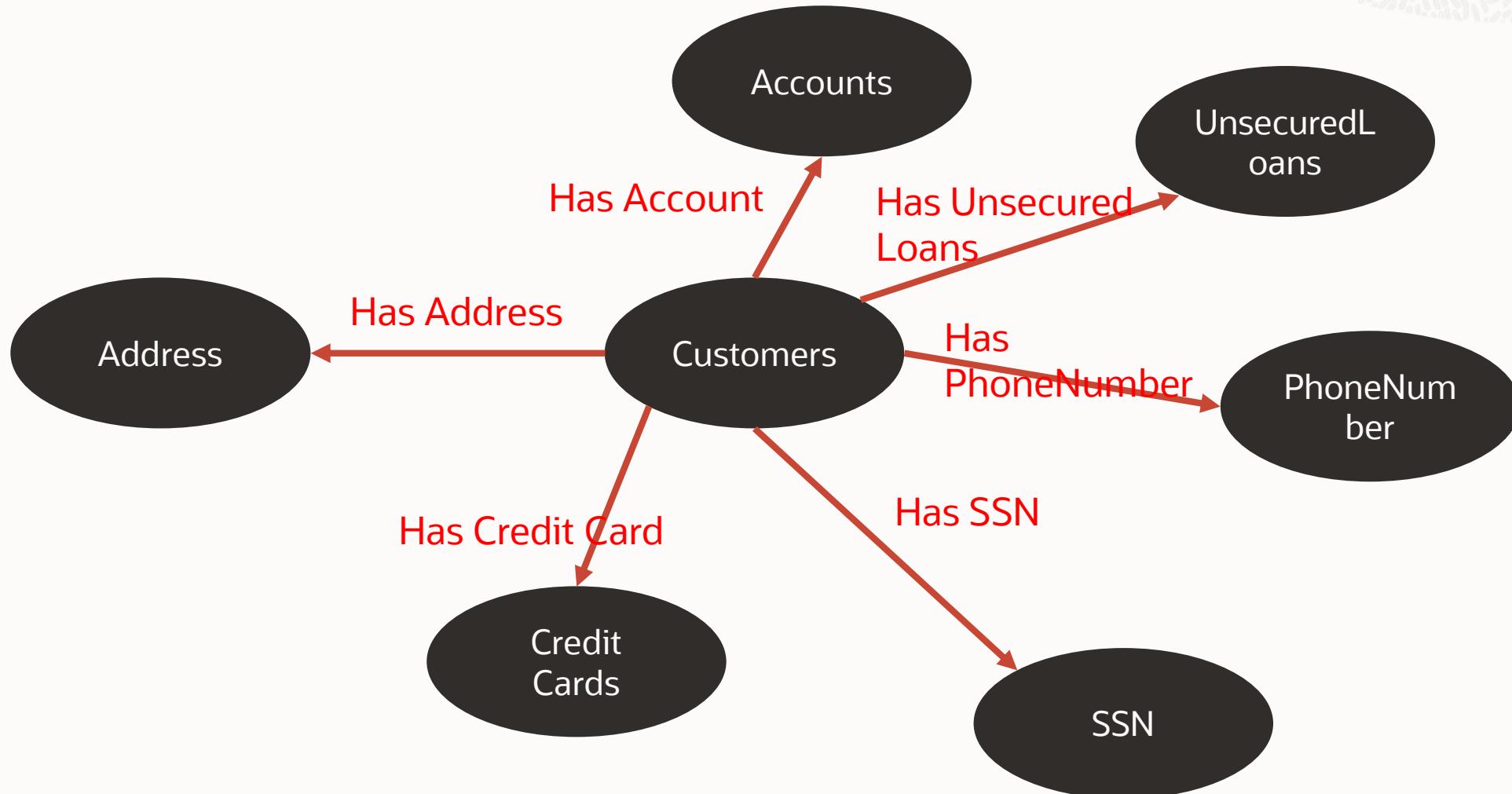
<https://github.com/oracle/pgql-lang>

STEP1: Convert from Relational Data Model to Property Graph Data Model

Bank Relational Data Model



Transform to Property Graph Data Model



How to Convert with Oracle Graph

CREATE PROPERTY GRAPH bank

VERTEX TABLES (

ACCOUNTS KEY(UNIQUEID) PROPERTIES ALL COLUMNS,

CUSTOMERS as CUST KEY(UNIQUEID) PROPERTIES (FIRSTNAME,LASTNAME),

ADDRESS KEY(STREET,CITY,STATE,ZIPCODE) PROPERTIES (STREET,CITY,STATE,ZIPCODE),

PHONENUMBERS KEY(PHONENUMBER) PROPERTIES (PHONENUMBER),

SSN KEY(SSN) PROPERTIES ALL COLUMNS,

CREDITCARDS KEY(UNIQUEID) PROPERTIES ALL COLUMNS,

UNSECUREDLOANS KEY(UNIQUEID) PROPERTIES ALL COLUMNS

)

EDGE TABLES (

How to Convert with Oracle Graph

EDGE TABLES (

ACCOUNTS SOURCE KEY(UNIQUEID) REFERENCES CUST DESTINATION KEY(UNIQUEID)
REFERENCES ACCOUNTS LABEL HAS_BANKACCOUNT,

CUST_ADDRESS SOURCE KEY(UNIQUEID) REFERENCES CUST DESTINATION
KEY(STREET,CITY,STATE,ZIPCODE) REFERENCES ADDRESS LABEL HAS_ADDRESS,

CUST_PHONENUMBERS SOURCE KEY(UNIQUEID) REFERENCES CUST DESTINATION
KEY(PHONENUMBER) REFERENCES PHONENUMBERS

LABEL HAS_PHONENUMBER,

CUST_SSN SOURCE KEY(UNIQUEID) REFERENCES CUST DESTINATION KEY(SSN) REFERENCES SSN
LABEL HAS_SSN,

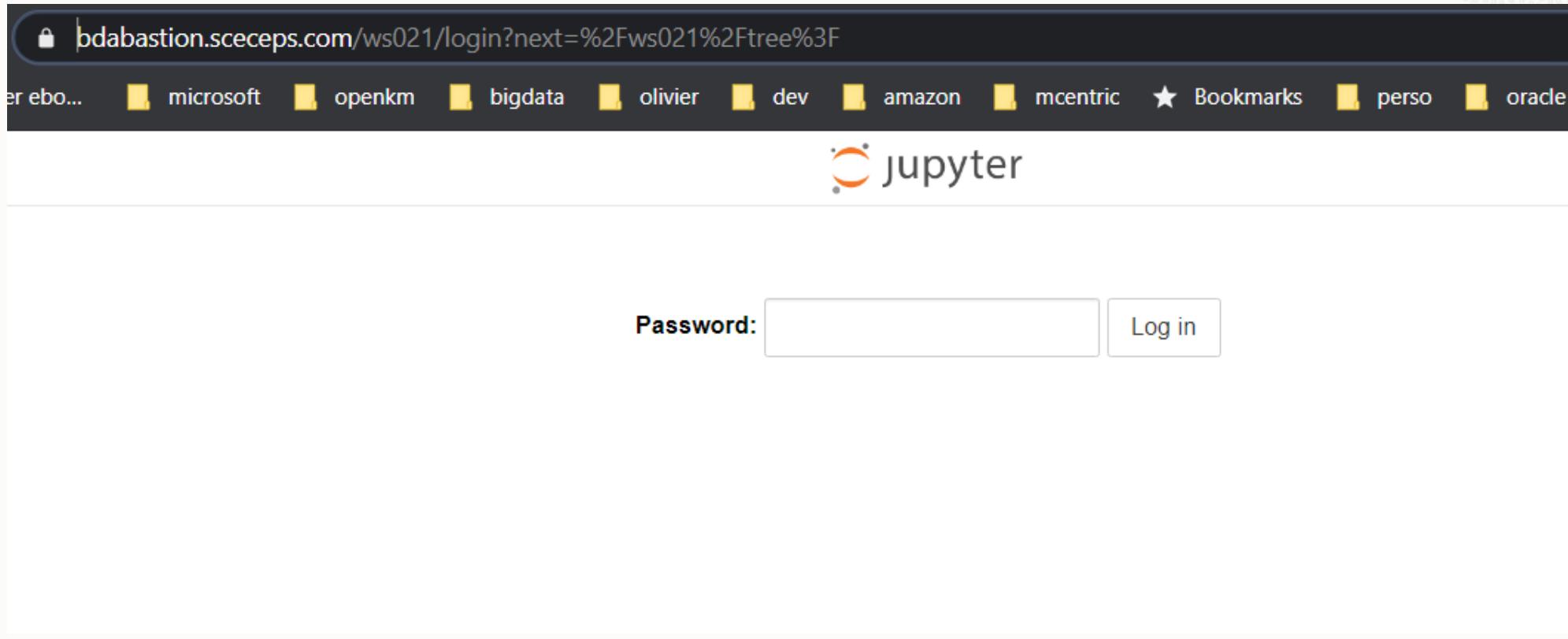
CREDITCARDS SOURCE KEY(UNIQUEID) REFERENCES CUST DESTINATION KEY(UNIQUEID)
REFERENCES CREDITCARDS

LABEL HAS_CREDITCARDS,

UNSECUREDLOANS SOURCE KEY(UNIQUEID) REFERENCES CUST DESTINATION KEY(UNIQUEID)
REFERENCES UNSECUREDLOANS

LABEL HAS_UNSECUREDLOANS

Connect to Jupyter from your browser



Open Terminal from Jupyter

Select NEW / Terminal

The screenshot shows the Jupyter Notebook interface. At the top, there is a navigation bar with a logo, 'jupyter', 'Quit', and 'Logout' buttons. Below the navigation bar, there are three tabs: 'Files' (selected), 'Running', and 'Clusters'. On the left, there is a file browser showing files like 'bank.json', 'jupyter.pid', 'launchpy.sh', 'nohup.log', and 'nohup.out'. On the right, there is a sidebar with 'Upload' and 'New' buttons. The 'New' button dropdown is open, showing options: 'Notebook: Python 3', 'Other: Text File', 'Folder', and 'Terminal'. The 'Terminal' option is circled in red.

Name	Type	Last Modified	Size
0	Folder	a day ago	4.02 kB
bank.json	Text File	a day ago	5 kB
jupyter.pid	Text File	a day ago	4 kB
launchpy.sh	Text File	a day ago	4 kB
nohup.log	Text File	a day ago	4 kB
nohup.out	Text File	a day ago	4 kB

Terminal Access for your user “workshopXXX”



Check Relational Tables in your database

Use the tool SQLCL:

<https://www.oracle.com/es/database/technologies/appdev/sqlcl.html>

<https://docs.oracle.com/en/database/oracle/sql-developer-command-line/19.2/sqcug/working-sqlcl.html#GUID-1343FA2B-BDB4-4645-B4D4-CD7C3E200AC9>

Property Graph embedded in SQLCL:

<https://docs.oracle.com/en/database/oracle/sql-developer-command-line/20.2/sqcug/using-pgql-plug-sqlcl.html#GUID-EOEFA43F-003F-4C8C-8056-54E9A428B8B7>

Check Relational Tables in your database



Logout

```
[workshop021@graph02 ~]$ ls  
bank.json jupyter.pid launchpy.sh nohup.log nohup.out  
[workshop021@graph02 ~]$ /opt/sqlcl/bin/sql workshop021/welcome1@orclpdb
```

```
SQLcl: Release 20.4 Production on Mon Mar 22 05:57:15 2021
```

```
Copyright (c) 1982, 2021, Oracle. All rights reserved.
```

```
Connected to:  
oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production  
version 19.3.0.0.0
```

```
SQL> select table_name from user_tables;
```

TABLE_NAME
ACCOUNTS
CREDITCARDS
CUSTOMERS
UNSECUREDLOANS

```
SQL> ■
```

Execute the PGQL Command to create the Property Graph Model

```
SQL> pgql auto on
```

```
PGQL Auto enabled for graph=[null], execute=[true], translate=[false]
PGQL> CREATE PROPERTY GRAPH bank VERTEX TABLES ( ACCOUNTS KEY(UNIQUEID) PROPERTIES ALL COLUMNS, CUSTOMERS as CUST KEY(UNIQUEID) PROPERTIES (FIRSTNAME,LASTNAME), ADDRESS KEY(STREET,CITY,STATE,ZIPCODE) PROPERTIES (STREET,CITY,STATE,ZIPCODE), PHONENUMBERS KEY(PHONENUMBER) PROPERTIES (PHONENUMBER), SSN KEY(SSN) PROPERTIES ALL COLUMNS, CREDITCARDS KEY(UNIQUEID) PROPERTIES ALL COLUMNS, UNSECUREDLOANS KEY(UNIQUEID) PROPERTIES ALL COLUMNS ) EDGE TABLES ( ACCOUNTS SOURCE KEY(UNIQUEID) REFERENCES CUST DESTINATION KEY(UNIQUEID) REFERENCES ACCOUNTS LABEL HAS_BANKACCOUNT, CUST_ADDRESS SOURCE KEY(UNIQUEID) REFERENCES CUST DESTINATION KEY(STREET,CITY,STATE,ZIPCODE) REFERENCES ADDRESS LABEL HAS_ADDRESS, CUST_PHONENUMBERS SOURCE KEY(UNIQUEID) REFERENCES CUST DESTINATION KEY(PHONENUMBER) REFERENCES PHONENUMBERS LABEL HAS_PHONENUMBER, CUST_SSN SOURCE KEY(UNIQUEID) REFERENCES CUST DESTINATION KEY(SSN) REFERENCES SSN LABEL HAS_SSN, CREDITCARDS SOURCE KEY(UNIQUEID) REFERENCES CUST DESTINATION KEY(UNIQUEID) REFERENCES CREDITCARDS LABEL HAS_CREDITCARDS, UNSECUREDLOANS SOURCE KEY(UNIQUEID) REFERENCES CUST DESTINATION KEY(UNIQUEID) REFERENCES UNSECUREDLOANS LABEL AS_UNSECUREDLOANS );
```

```
Graph created
```

```
PGQL> █
```

Check Tables created with SQL commands

```
PGQL> pgql auto off
```

```
PGQL Auto disabled
```

```
SQL> select table_name from user_tables;
```

TABLE_NAME
ACCOUNTS
BANKGE\$
BANKGT\$
BANKIT\$
BANKSS\$
BANKVD\$
BANKVT\$
CREDITCARDS
CUSTOMERS
UNSECUREDLOANS

```
10 rows selected.
```

Other commands to test PGQL

- Change **XXX** in the next command for your workshop user:
- You can delete the Graph Data Model using the next commands:

```
/opt/sqlcl/bin/sql workshopXXX/welcome1@orclpdb
```

```
PGQL AUTO ON  
DROP PROPERTY GRAPH BANK;
```

- Execute the BANK Property Graph from command line:

```
/opt/sqlcl/bin/sql workshopXXX/welcome1@orclpdb @pgbank_create.sql
```

STEP2: Use Open Source Apache Zeppelin with PGX Interpreter and Groovy.

Oracle Environment

Users	Jupyter URL	Zeppelin URL
Workshop001 to workshop020	https://bdabastion.sceceps.com/wsXXX	https://bdabastion.sceceps.com/zp1/
Workshop021 to workshop040	https://bdabastion.sceceps.com/wsXXX	https://bdabastion.sceceps.com/zp2/
Workshop041 to workshop060	https://bdabastion.sceceps.com/wsXXX	https://bdabastion.sceceps.com/zp3/
Workshop061 to workshop080	https://bdabastion.sceceps.com/wsXXX	https://bdabastion.sceceps.com/zp4/
Workshop081 to workshop100	https://bdabastion.sceceps.com/wsXXX	https://bdabastion.sceceps.com/zp5/

pwd: welcome1

Access to Apache Zeppelin Open Source

Click on “login” button and Use your user “workshopXXX”



Welcome to Zeppelin

Zeppelin is web-based notebook that enables interactive data analysis. You can make beautiful data-driven, interactive, collaborative reports.

Help

Get started with [Zeppelin documentation](#)

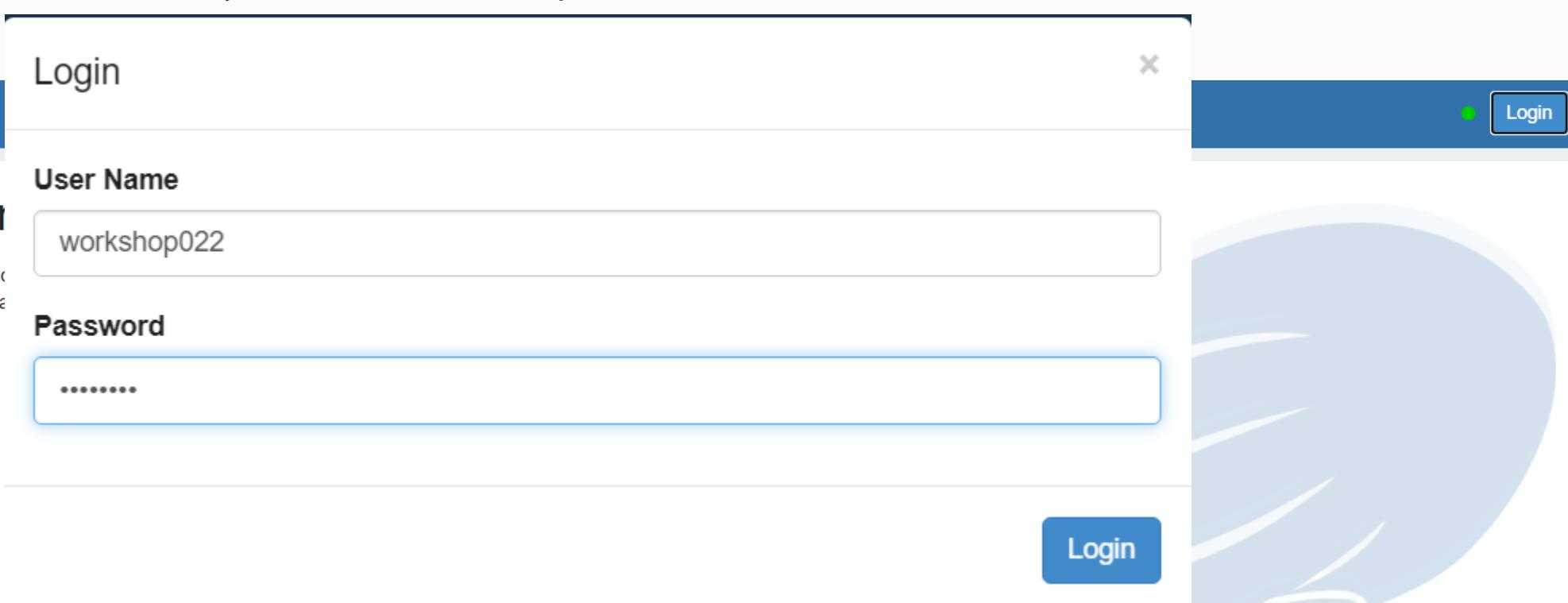
Community

Please feel free to help us to improve Zeppelin, Any contribution are welcome!

[Mailing list](#)

[Issues tracking](#)

[Github](#)



The image shows the Zeppelin login interface. It features a header with the Zeppelin logo and a "Login" button. Below the header is a "User Name" field containing "workshop022". Below that is a "Password" field with several dots indicating the password. At the bottom right is a blue "Login" button. The background of the slide features a large, stylized graphic of a brain and a steering wheel.

Login

User Name

workshop022

Password

.....

Login

Select the notebook in the Folder for your user “workshopXXX”

Zeppelin is web-based notebook that enables interactive data analytics.

You can make beautiful data-driven, interactive, collaborative document with SQL, code and even more!

Notebook

 Import note

 Create new note

 Flink Tutorial   

 Miscellaneous Tutorial

 PGX

 Python Tutorial

 R Tutorial

 Spark Tutorial

 workshop021

 Bank Ring Analysis

 workshop022

 workshop023

 workshop024

 workshop025

 workshop026

 workshop027

 workshop028

 workshop029

 workshop030

 workshop031

Help

Get started with [Zeppelin documentation](#)

Community

Please feel free to help us to improve Zeppelin,
Any contribution are welcome!

 Mailing list

 Issues tracking

 Github

Execute each paragraph Step By Step

Bank Ring Analysis

         Head  

default ▾

Connect to PGX

FINISHED

```
%pgx
import oracle.pgx.api.*
import groovy.json.*

baseUrl = 'https://localhost:7007'
username = 'workshop001'
password = 'welcome1'

conn = new URL("$baseUrl/auth/token").openConnection()
conn.setRequestProperty('Content-Type', 'application/json')
token = conn.with {
    doOutput = true
    requestMethod = 'POST'
    outputStream.withWriter { writer ->
        writer << JsonOutput.toJson([username: username, password: password])
    }
    return new JsonSlurper().parseText(content.text).access_token
}

instance = Pgx.getInstance(baseUrl, token)
-----
```

Took 7 sec. Last updated by oracle at March 22 2021, 11:56:22 AM. (outdated)

Database credentials

FINISHED

```
%pgx  
  
jdbcUrl = "jdbc:oracle:thin:@localhost:1521/orclpdb"  
user = "workshop001"  
pass = "welcome1"
```

Step by Step

Bank Ring Analysis

A set of small, light-gray icons used for navigating through the document or application. From left to right, they include: a right-pointing arrow, a double right-pointing arrow, a document icon, a pencil icon, a double document icon, a download icon, a double download icon, and a user group icon.A magnifying glass icon, typically used for search functions.A simple trash bin icon, often used for deleting or removing items.

default ▾

```
%pgx

import oracle.pgx.common.types.*;

builder.setUsername(user);
builder.setPassword(pass);
// Read the Graph Name created previously
builder.setName("bank");
builder.addVertexProperty("FIRSTNAME", PropertyType.STRING);
builder.addVertexProperty("LASTNAME", PropertyType.STRING);
builder.addVertexProperty("UNIQUEID", PropertyType.STRING);
builder.addVertexProperty("SSN", PropertyType.STRING);
builder.addVertexProperty("PHONENUMBER", PropertyType.STRING);
builder.addVertexProperty("BALANCE", PropertyType.DOUBLE);
builder.addVertexProperty("APR", PropertyType.DOUBLE);
builder.addVertexProperty("LOANAMOUNT", PropertyType.DOUBLE);
builder.addVertexProperty("STREET", PropertyType.STRING);
builder.addVertexProperty("STATE", PropertyType.STRING);
builder.addVertexProperty("ZIPCODE", PropertyType.STRING);
builder.addVertexProperty("CITY", PropertyType.STRING);
builder.addVertexProperty("ACCOUNTNUMBER", PropertyType.STRING);
builder.addVertexProperty("LIMIT", PropertyType.DOUBLE);
builder.addVertexProperty("SECURITYCODE", PropertyType.STRING);
builder.addVertexProperty("EXPIRATIONDATE", PropertyType.TIMESTAMP);

builder.setLoadVertexLabels(true);
builder.setLoadEdgeLabel(true);
builder.setKeystoreAlias("alias");

oracle.pgx.config.PgRdbmsGraphConfigBuilder@3991c5fa
```

This name is the table name used in CREATE PROPERTY GRAPH xxxx

Took 0 sec. Last updated by oracle at March 22 2021, 11:56:47 AM.



STEP3: Use Python to access to Oracle Graph “PyPGX”.

Connect to <https://bdabastion.sceceps.com/wsXXX>

bdabastion.sceceps.com/ws022/tree? ★

... [microsoft](#) [openkm](#) [bigdata](#) [olivier](#) [dev](#) [amazon](#) [mcntric](#) [Bookmarks](#) [perso](#) [oracle](#) [teradata](#) [gartner](#) [iebusiness](#) [customers](#)

 jupyter Quit Logout

[Files](#) [Running](#) [Clusters](#)

Select items to perform actions on them. Upload New ▾ 

<input type="checkbox"/>	Name	Last Modified	File size
<input type="checkbox"/>	/		
<input type="checkbox"/>	PGX.analysis.Bank.ipynb	Running 4 hours ago	13.9 kB
<input type="checkbox"/>	PGX.visjs.Bank.ipynb	Running 4 hours ago	18.7 kB
<input type="checkbox"/>	bank.json	2 days ago	2.05 kB
<input type="checkbox"/>	graph.html	4 days ago	5.19 kB
<input type="checkbox"/>	jupyter.pid	a day ago	5 B
<input type="checkbox"/>	launchpy.sh	5 days ago	314 B
<input type="checkbox"/>	nohup.log	4 hours ago	10.3 kB
<input type="checkbox"/>	nohup.out	2 days ago	1 B

26 Copyright © 2021, Oracle and/or its affiliates | Confidential: Internal/Restricted/Highly Restricted [Nov. 2020] 

Select the notebook “PGX.analysis.bank.ipynb”



jupyter PGX.analysis.Bank Last Checkpoint: 4 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

Ring Algorithm using PGX ¶

First Step: Get a session in PGX server.

```
In [1]: import json
import ssl

ssl._create_default_https_context = ssl._create_unverified_context

def generateToken(base_url, username, password):
    from urllib.request import Request, urlopen
    from urllib.error import HTTPError

    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:
```

Step by Step in the notebook “PGX.analysis.bank.ipynb”

In [4]: ► `print(session)`

```
PgxSession(id: 1a5cadf9-2b36-4729-a6d2-4e10f412e6b1, name: pypgx)
```

Load JSON File with Property Graph Definition

In [6]: ► `# read Graph
graph = session.read_graph_with_properties("./bank.json")`

Bank.json Definition

```
{  
  "db_engine": "RDBMS",  
  "vertex_id_type": "long",  
  "error_handling": {},  
  "name": "bank",  
  "vertex_props": [  
    {  
      "dimension": 0,  
      "name": "FIRSTNAME",  
      "type": "string"  
    },  
    {  
      "dimension": 0,  
      "name": "LASTNAME",  
      "type": "string"  
    },  
    {  
      "dimension": 0,  
      "name": "UNIQUEID",  
      "type": "string"  
    },  
    {  
      "dimension": 0,  
      "name": "SSN",  
      "type": "string"  
    },  
    {  
      "dimension": 0,  
      "name": "PHONENUMBER",  
      "type": "string"  
    },  
    {  
      "dimension": 0,  
      "name": "BALANCE",  
      "type": "double"  
    },  
    .....etc.....  
  ],  
  "loading": {  
    "load_vertex_labels": true,  
    "load_edge_label": true  
  },  
  "edge_props": [],  
  "attributes": {},  
  "format": "pg"  
}
```

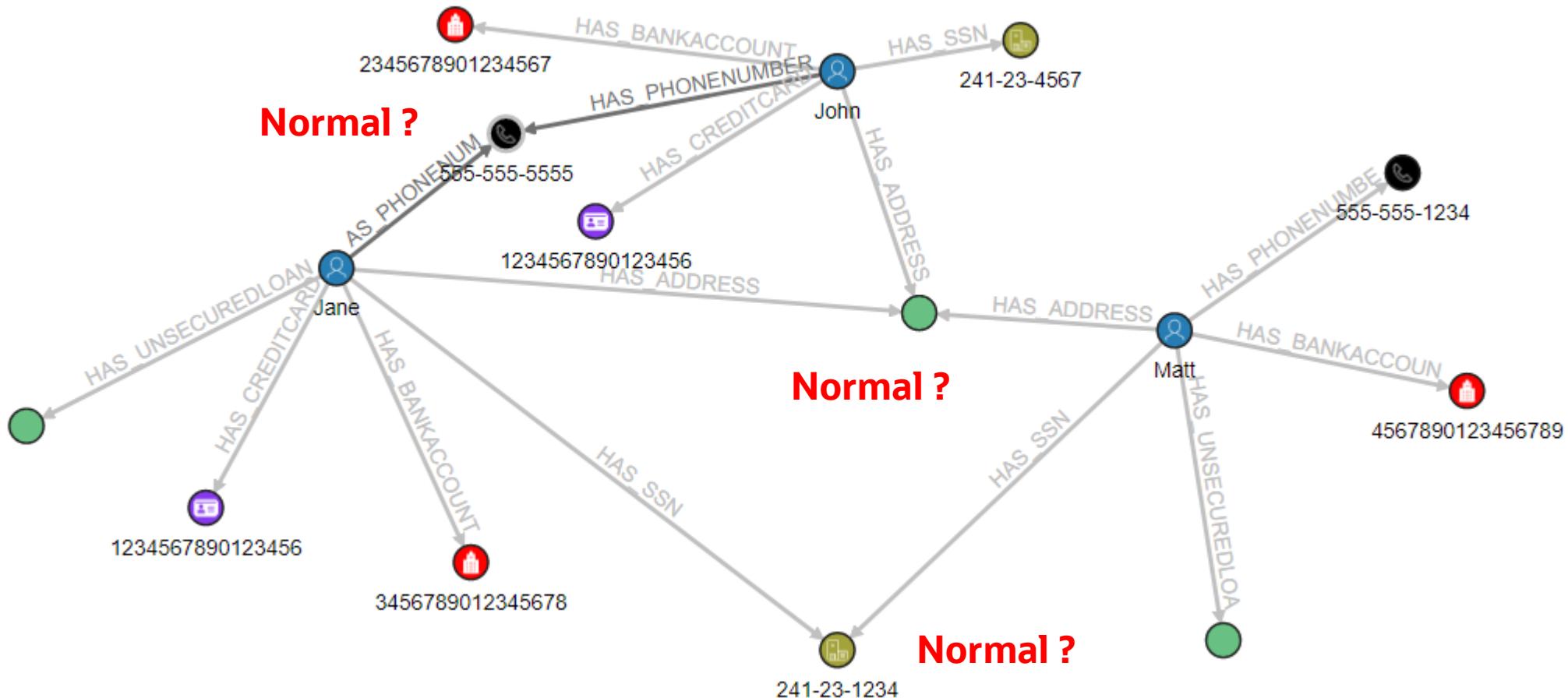
Compare with Apache Zeppelin Definition

```
%pgx
```

```
import oracle.pgx.common.types.*  
builder.setUsername(user);  
builder.setPassword(pass);  
// Read the Graph Name created previously  
builder.setName("bank");  
builder.addVertexProperty("FIRSTNAME", PropertyType.STRING);  
builder.addVertexProperty("LASTNAME", PropertyType.STRING);  
builder.addVertexProperty("UNIQUEID", PropertyType.STRING);  
builder.addVertexProperty("SSN", PropertyType.STRING);  
builder.addVertexProperty("PHONENUMBER", PropertyType.STRING);  
builder.addVertexProperty("BALANCE", PropertyType.DOUBLE);  
builder.addVertexProperty("APR", PropertyType.DOUBLE);  
  
.....etc.....
```

STEP4: Apply Queries to detect Ring Patterns in the Graph Data Model.

Checking the Data with Graph Visualizer



Analysis in the different Vertex in order to check Ring Data Fraud (PyPGX)

Entity Link Analysis

Performing entity link analysis on the above Graph data model is demonstrated below. We use brackets in the below table is to isolate individual elements of a collection.

Find account who share more than one piece of legitimate contact

```
In [ ]: pgxResultSetNode = graph.query_pgql("""
    select label(contact), listagg(acct.FIRSTNAME,','), count(*) as size
    from match (acct)-[]->(contact)
    where exists (
        SELECT count(*) as ringsize, contact
        from MATCH (account:CUST)-[]->(contact)
        group by contact
        having ringsize > 1
        order by ringsize desc
    )
    group by label(contact)
""")

for i in pgxResultSetNode:
    print (i)
```



Check the Financial Risk for possible Fraud Ring (PyPGX)

Determine the financial risk of a possible fraud ring

```
In [ ]: pgxResultSetNode = graph.query_pgql("""
    select label(contact), listagg(acct.FIRSTNAME,','),
           count(*) as size, round(sum(CASE
                                         WHEN label(r)='HAS_CREDITCARDS' THEN unsecuredAccount.LIMIT
                                         WHEN label(r)='HAS_UNSECUREDLOANS' THEN unsecuredAccount.BALANCE
                                         ELSE 0
                                       END)) as FINANCIALRISK
    from match (acct)-[]->(contact),
         match (acct)-[r:HAS_CREDITCARDS|HAS_UNSECUREDLOANS]->(unsecuredAccount)
    where exists (
        SELECT count(*) as ringsize, contact
        from MATCH (account:CUST)-[]->(contact)
        group by contact
        having ringsize > 1
        order by ringsize desc
    )
    group by label(contact)
""")

for i in pgxResultSetNode:
    print (i)
```



Analysis in the different Vertex in order to check Ring Data Fraud (Zeppelin)

This is another way to find the same list using a pattern.

FINISHED

Took 0 sec. Last updated by oracle at March 22 2021, 11:57:10 AM.

%pgx

```
graph.queryPgql("""  
select label(contact), count(*) as ringsize  
match (accta:CUST)-[e1]-(contact)<-[e2]-(acctb:CUST)  
where accta != acctb  
group by contact  
having count(*) >1  
""")
```

FINISHED



settings ▾

label(contact)

ringsize

PHONENUMBERS

2

SSN

2

ADDRESS

6

Check the Financial Risk for possible Fraud Ring (Zeppelin)

Bank Ring Analysis  Head  

Find account who share more than one piece of legitimate contact and make groups with them. FINISHED 

Took 0 sec. Last updated by oracle at March 22 2021, 11:57:27 AM.

```
%pgx
graph.queryPgql("""
select label(contact), listagg(acct.FIRSTNAME,','), count(*) as size
from match (acct)-[]-(contact)
where exists (
  SELECT count(*) as ringsize, contact
  from MATCH (account:CUST)-[]-(contact)
  group by contact
  having ringsize > 1
  order by ringsize desc
)
group by label(contact)
""")
```

  settings ▾

label(contact)	listagg(acct.FIRSTNAME,',')	size
ADDRESS	John,Matt,Jane	3
PHONENUMBERS	John,Jane	2
SSN	Matt,Jane	2



STEP5: Connect Third-Party Library Vis.JS from Python.

Connect to <https://bdabastion.sceceps.com/wsXXX>

bdabastion.sceceps.com/ws022/tree? ★

... [microsoft](#) [openkm](#) [bigdata](#) [olivier](#) [dev](#) [amazon](#) [mcntric](#) [Bookmarks](#) [perso](#) [oracle](#) [teradata](#) [gartner](#) [iebusiness](#) [customers](#)

 jupyter Quit Logout

[Files](#) [Running](#) [Clusters](#)

Select items to perform actions on them. Upload New ▾ 

<input type="checkbox"/>	Name	Last Modified	File size
<input type="checkbox"/>	/		
<input type="checkbox"/>	PGX.analysis.Bank.ipynb	Running 4 hours ago	13.9 kB
<input type="checkbox"/>	PGX.visjs.Bank.ipynb	Running 4 hours ago	18.7 kB
<input type="checkbox"/>	bank.json	2 days ago	2.05 kB
<input type="checkbox"/>	graph.html	4 days ago	5.19 kB
<input type="checkbox"/>	jupyter.pid	a day ago	5 B
<input type="checkbox"/>	launchpy.sh	5 days ago	314 B
<input type="checkbox"/>	nohup.log	4 hours ago	10.3 kB
<input type="checkbox"/>	nohup.out	2 days ago	1 B

38 Copyright © 2021, Oracle and/or its affiliates | Confidential: Internal/Restricted/Highly Restricted [Nov. 2020] 

Select the notebook “PGX.visjs.bank.ipynb”

Query to prepare Data for Vis.JS, Network and NetworkX (Python)

```
In [ ]: # get result
# node data
pgxResultSetNode = graph.query_pgql("""
    SELECT id(x), label(x), x.FIRST_NAME, x.CITY
    MATCH (x)-[]-()
""")

#[ 'size', 'value', 'title', 'x', 'y', 'label', 'color']

node_id = []
node_title =[]
node_value = []
node_label = []

for i in pgxResultSetNode:
    size = i[0]
    if size not in node_id:
        node_id.append(size)
        node_label.append(i[1])
        if i[2] != '':
            node_title.append(i[2])
            print('Node: size: ' + str(i[0]) + ' label: ' + i[1] + ' FIRST_NAME: ' + i[2])
        elif i[3] != '':
            node_title.append(i[3])
            print('Node: size: ' + str(i[0]) + ' label: ' + i[1] + ' CITY: ' + i[3])
        else:
            node_title.append('n/a')
            print('Node: size: ' + str(i[0]) + ' label: ' + i[1] + ' title: n/a')
```

Use Vis.JS Library (PYVIS Library in Python)

Open Source Vis.JS Visualization using Network Python Library

```
In [ ]: ┌─▶ from pyvis.network import Network  
  
g = Network(notebook=True, height = '800px', width = '100%', directed = True)  
g.options = {  
    "nodes": {  
        "scaling": {  
            "min": 16,  
            "max": 32,  
        },  
    },  
    "edges": {  
        "color": "GRAY",  
        "smooth": "false",  
    },  
    "physics": {  
        "barnesHut": { "gravitationalConstant": -30000 },  
        "stabilization": { "iterations": 2500 },  
    }  
}  
  
g.add_nodes(node_id,label=node_label,title=node_title)  
g.add_edges(edge_list)  
#g.show_buttons()  
g.show('graph.html')
```

Use NetworkX with Vis.JS

Another Vis.JS Visualization using NetworkX Python Library

```
In [ ]: └─ from pyvis.network import Network
      import networkx as nx

      nx_graph = nx.Graph()

      pgxResultSetNode = graph.query_pgql("""
          SELECT id(x), label(x), x.FIRSTNAME, x.CITY
          MATCH (x)-[]-()
      """)

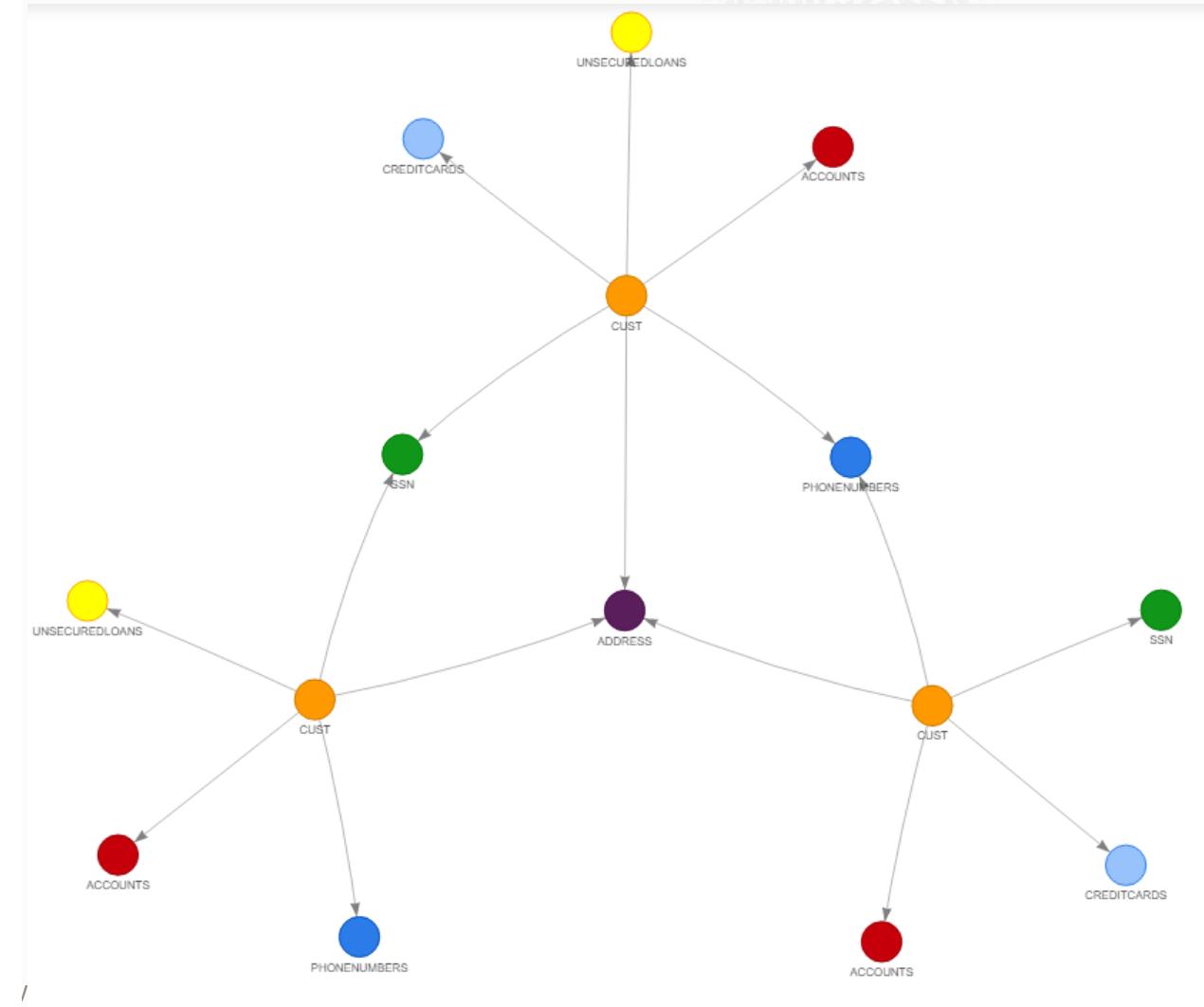
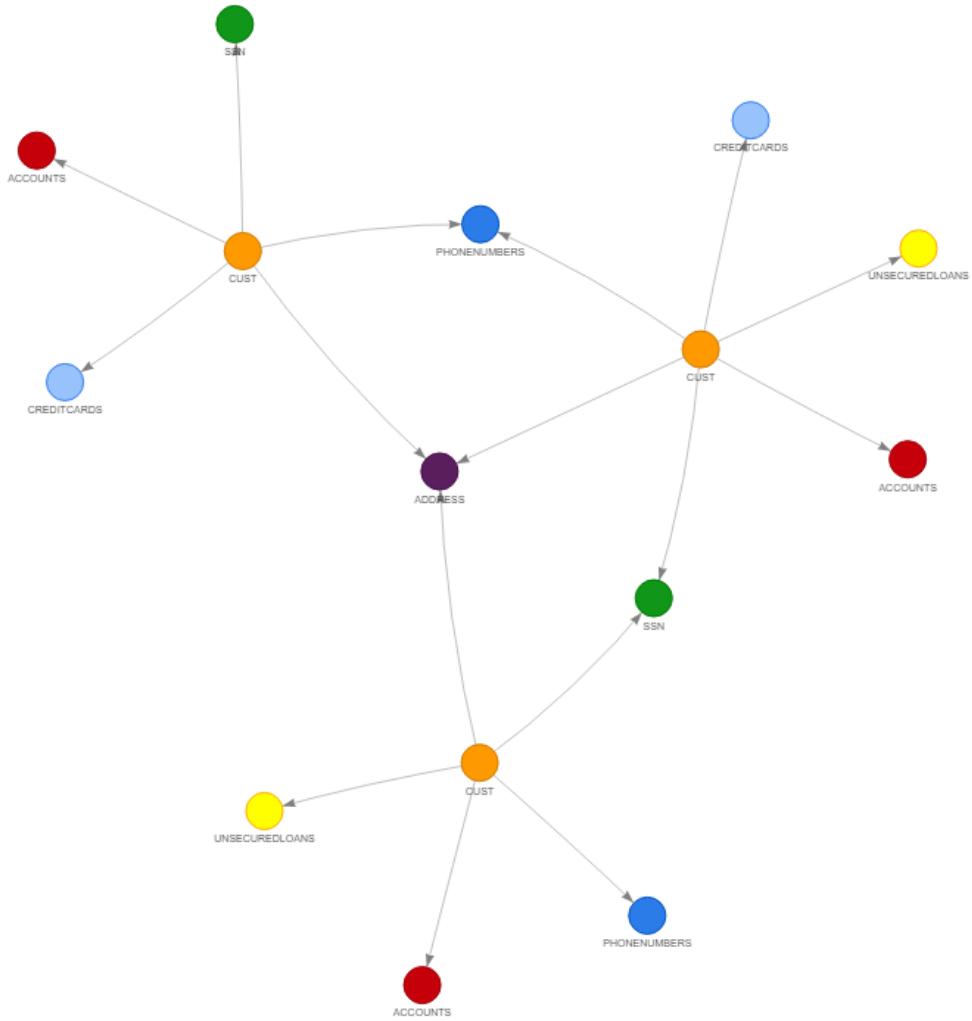
      for i in pgxResultSetNode:
          if i[2]:
              nx_graph.add_node(i[0], size=20, title=i[2], group=i[1])
          else:
              nx_graph.add_node(i[0], size=20, title=i[3], group=i[1])

      # edge data
      pgxResultSetEdge = graph.query_pgql("""
          SELECT id(x), id(y), label(e)
          MATCH (x)-[e]->(y)
      """)

      edge_list = []
      for i in pgxResultSetEdge:
          nx_graph.add_edge(i[0], i[1], weight=5)

      nt = Network(notebook=True, height = '800px', width = '100%')
      # populates the nodes and edges data structures
      nt.from_nx(nx_graph)
      nt.show('nx.html')
```

Use NetworkX with Vis.js



STEP6: Use PGX Visualizer.

Access to PGX Visualizer

<https://bdabastion.sceceps.com/ui>

User: bankuser
Pwd: welcome1



PGX Visualizer

Choose the graph “global_bank”. Execute the query.

ORACLE® Graph Visualization bankuser ▾

PGQL Graph Query

```
1 SELECT e
2 FROM MATCH ()-[e]->()
3 LIMIT 100
4
5
```

Graph Parallelism ? Settings

global_bank 0 ↕ ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋

PGX Visualizer Result

ORACLE® Graph Visualization

bankuser ▾

PGQL Graph Query

```
1 SELECT e
2 FROM MATCH ()-[e]->()
3 LIMIT 100
4
5
```

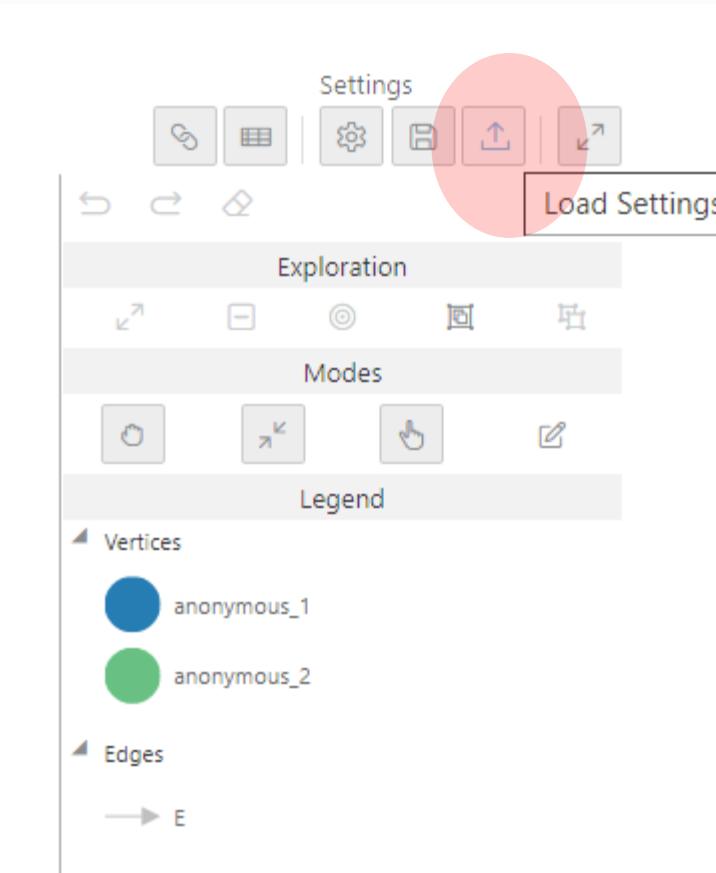
Graph global_bank ▾ Parallelism ? 0 ↕ ⏪

The graph visualization interface includes a legend on the right side:

- Vertices:**
 - anonymous_1 (blue circle)
 - anonymous_2 (green circle)
- Edges:**
 - E (grey arrow)

Exploration and Modes buttons are also present in the legend area.

Upload the JSON Configuration “bank_settings_20210315.json”



Download the JSON Configuration file:

[https://github.com/operard/mlgraph/blob/main/graphviz/bank settings 20210315.json](https://github.com/operard/mlgraph/blob/main/graphviz/bank_settings_20210315.json).

Upload the JSON configuration “bank_settings_20210315.json”.

PGQL Graph Query

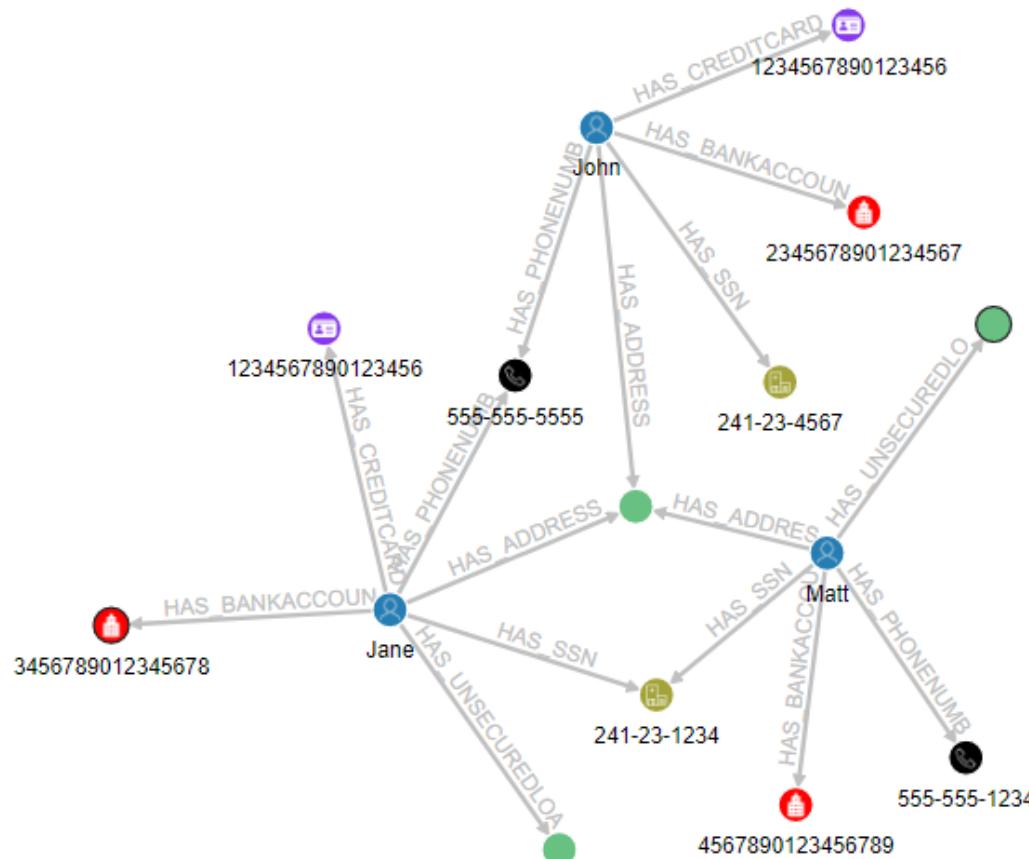
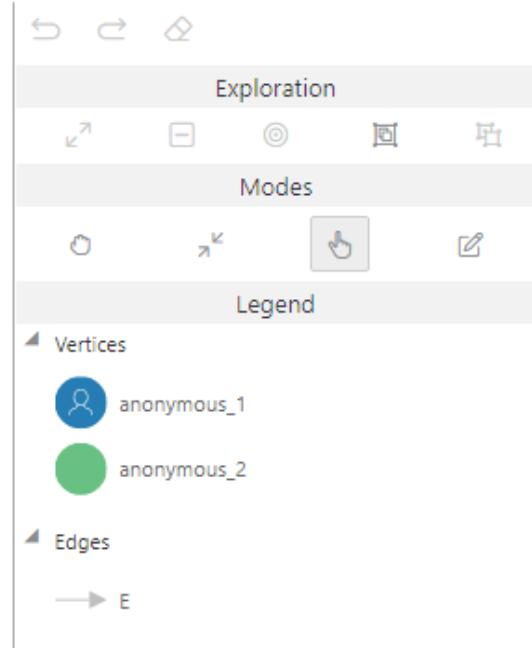
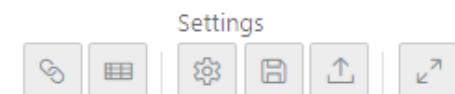
```
1 SELECT e
2 FROM MATCH ()-[e]->()
3 LIMIT 100
4
5
```

Graph

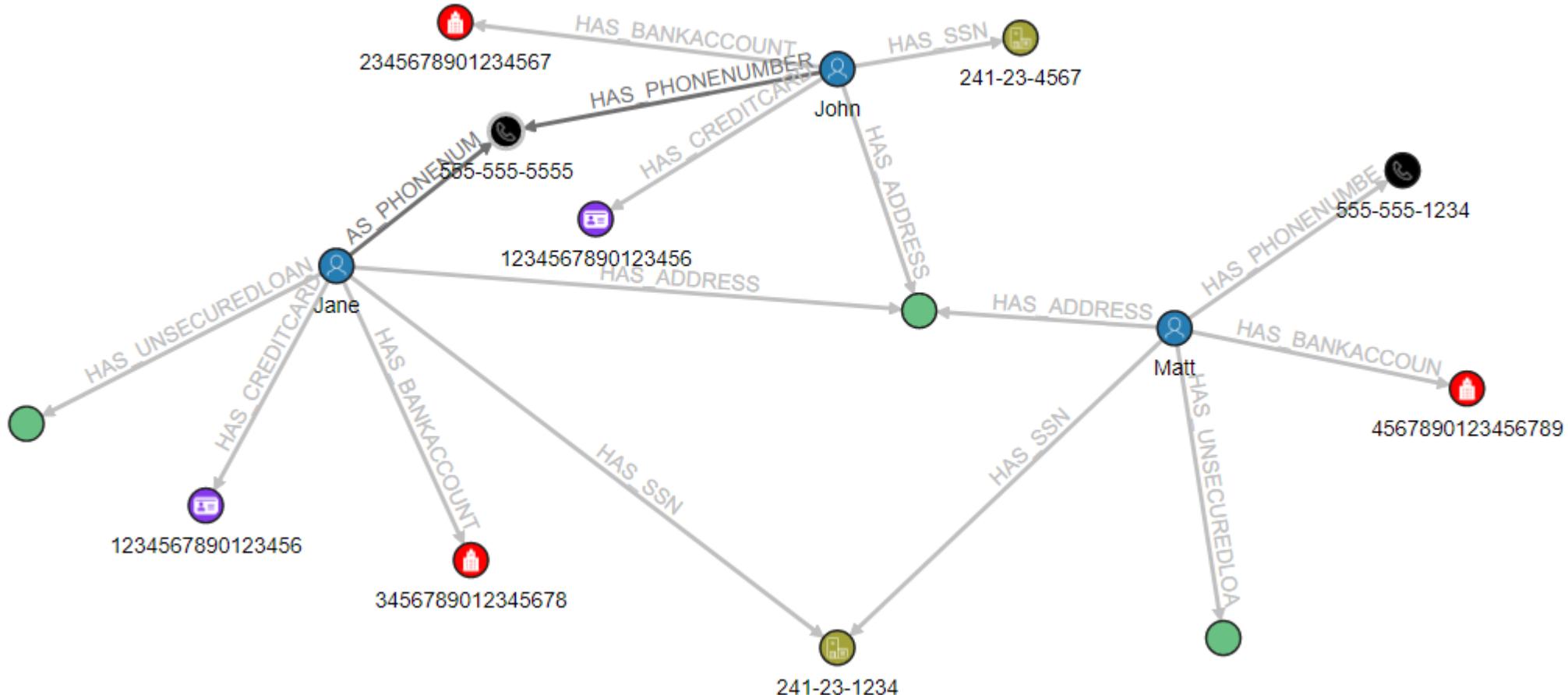
Parallelism ?

global_bank

0 ▼ ▲ ⏪ ⏩

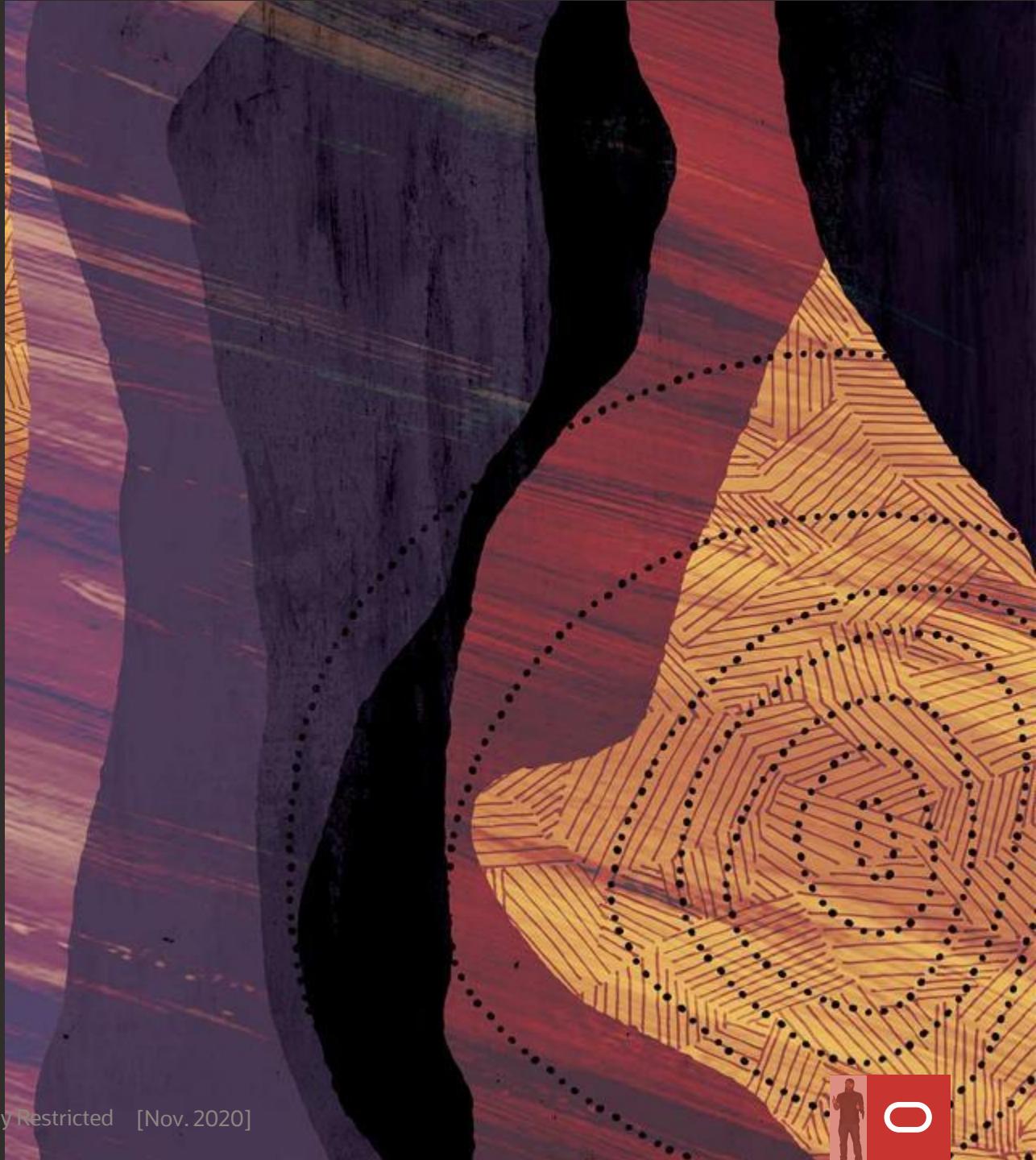


Bank Property Graph UI



Thank You

DevRel Team





ORACLE

O