ORACLE

PGQL Introduction and Deep Dive

Melli Annamalai and Ryota Yamanaka, Product Management, Oracle

July 30, 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

Copyright © 2020, Oracle and/or its affiliates

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.

Copyright © 2020, Oracle and/or its affiliates

Agenda

1. Introduction to PGQL	Melli
2. Basic PGQL Syntax	Melli
3. Try Running PGQL	Ryota
4. Advanced Topics	Ryota

Melli

Nashua, New Hampshire, USA
@AnnamalaiMelli

Ryota

Bangkok, Thailand
@ryotaymnk

Copyright © 2020, Oracle and/or its affiliates

Introduction to PGQL

Copyright © 2020, Oracle and/or its affiliates

Recap: 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:

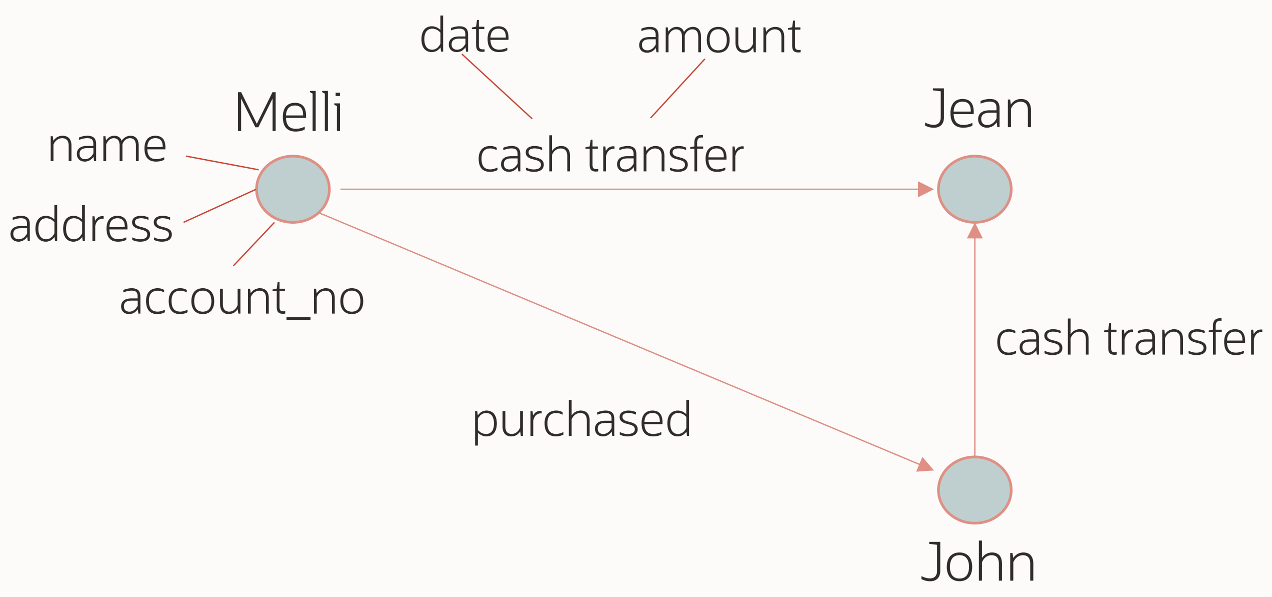
- Financial
- Law enforcement and security
- Manufacturing
- Public sector
- Pharma

and more

Copyright © 2020, Oracle and/or its affiliates

What is a Graph?

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



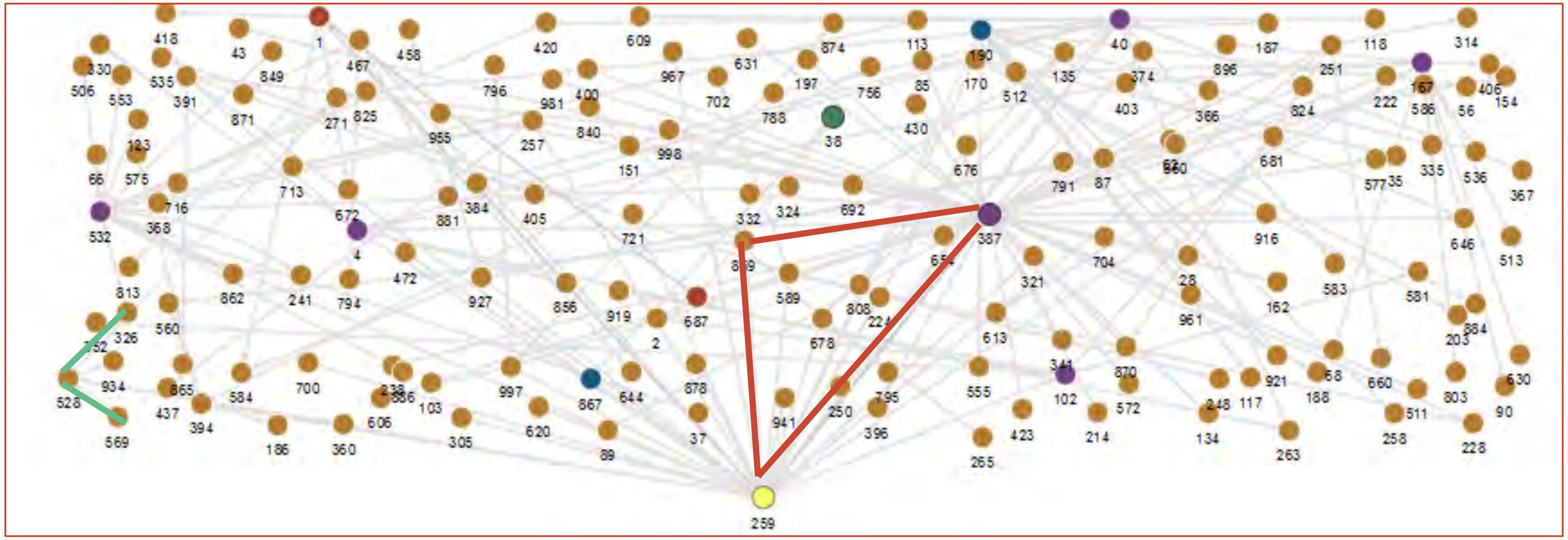
Copyright © 2020, Oracle and/or its affiliates

What is PGQL?

Property Graph Query Language

Is there a pattern that connects 259 to 869 to 387 and back?

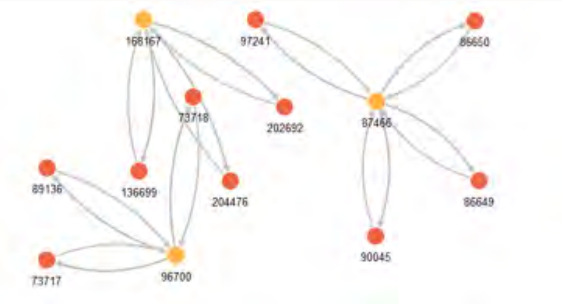
Is there a pattern that connects 528 to 326 and 569?

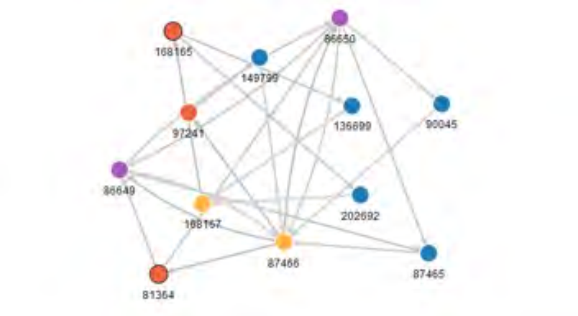


Copyright © 2020, Oracle and/or its affiliates

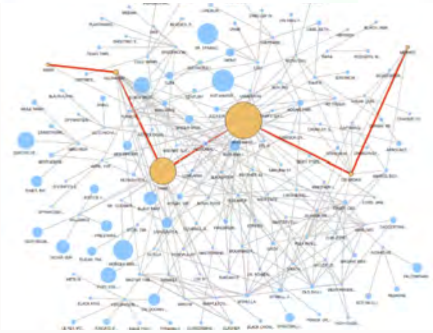
Graph Queries are Pattern Queries

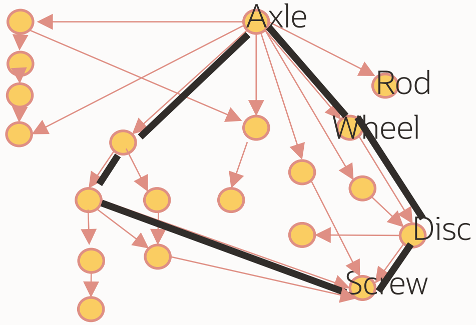
Cycles



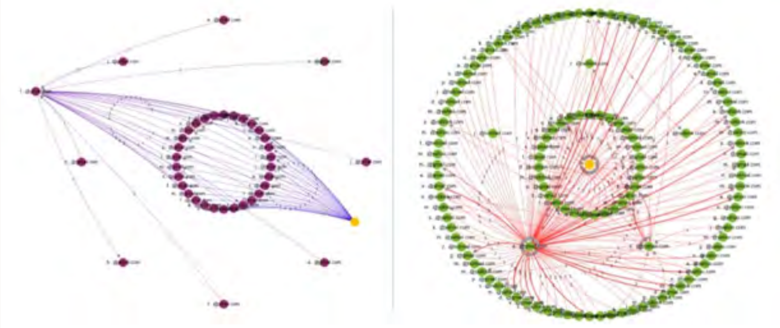


Paths





Patterns



Copyright © 2020, Oracle and/or its affiliates

PGQL

- SQL-like graph query language

- Add graph pattern to SQL constructs: `SELECT, FROM, WHERE, ORDER BY,` etc.
- Include functions in SQL: `SUM, AVG, MIN, MAX,` etc.

- DDL: `CREATE PROPERTY GRAPH, DROP PROPERTY GRAPH`

- DML: `INSERT, UPDATE, DELETE` graph vertices, edges, properties

Make it easy for SQL developers to write graph queries

Copyright © 2020, Oracle and/or its affiliates

Executing PGQL

- **GraphViz**

- **SQLcl**

- **Java**
  - **Notebook:** Using graph server (PGX) interpreter

  - **JShell CLI:** For quick tests and prototypes

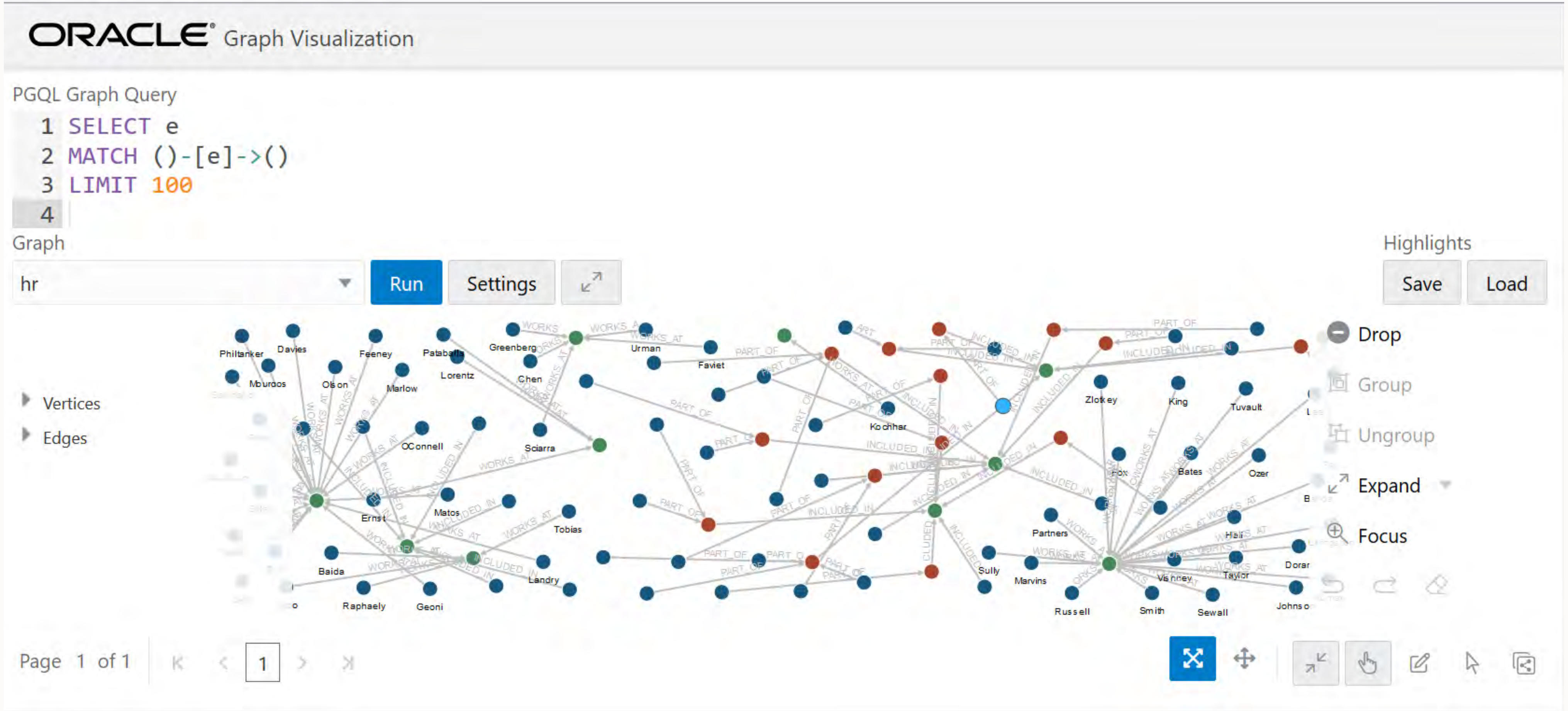
    ```
    opg-jshell> session.queryPgql(“SELECT e from MATCH ()-[e]->()”)
    ```

  - **Java application**

    ```
    String pgql = "SELECT e.\“amount\" AS AMOUNT "+ "FROM MATCH ()-[e]->()";
    rs = ps.executeQuery(pgql, /* query string */ "" /* options */);
    rs.print();
    ```

Copyright © 2020, Oracle and/or its affiliates

Execute PGQL in GraphViz



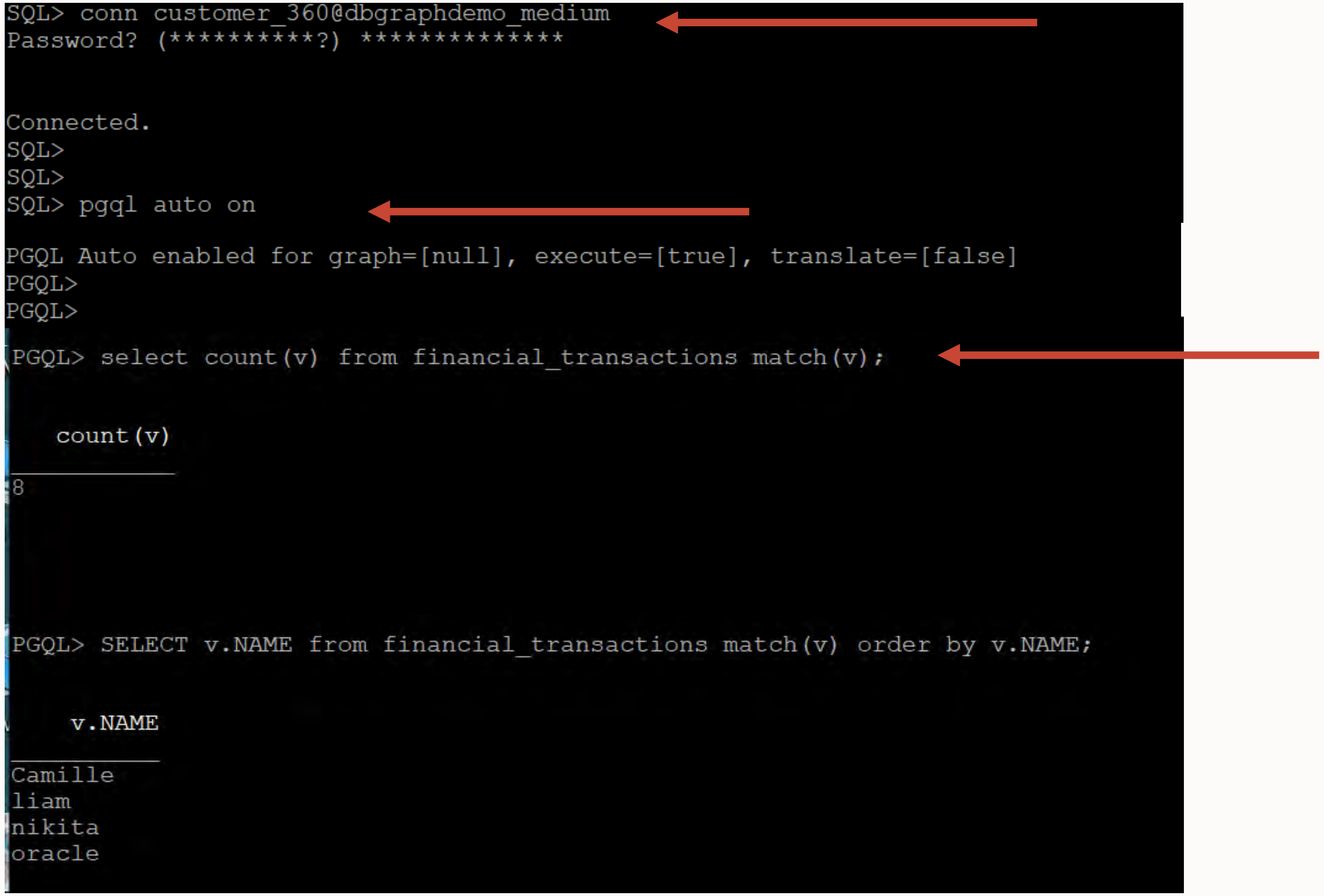
PGQL Graph Query

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

Graph: hr

Copyright © 2020, Oracle and/or its affiliates

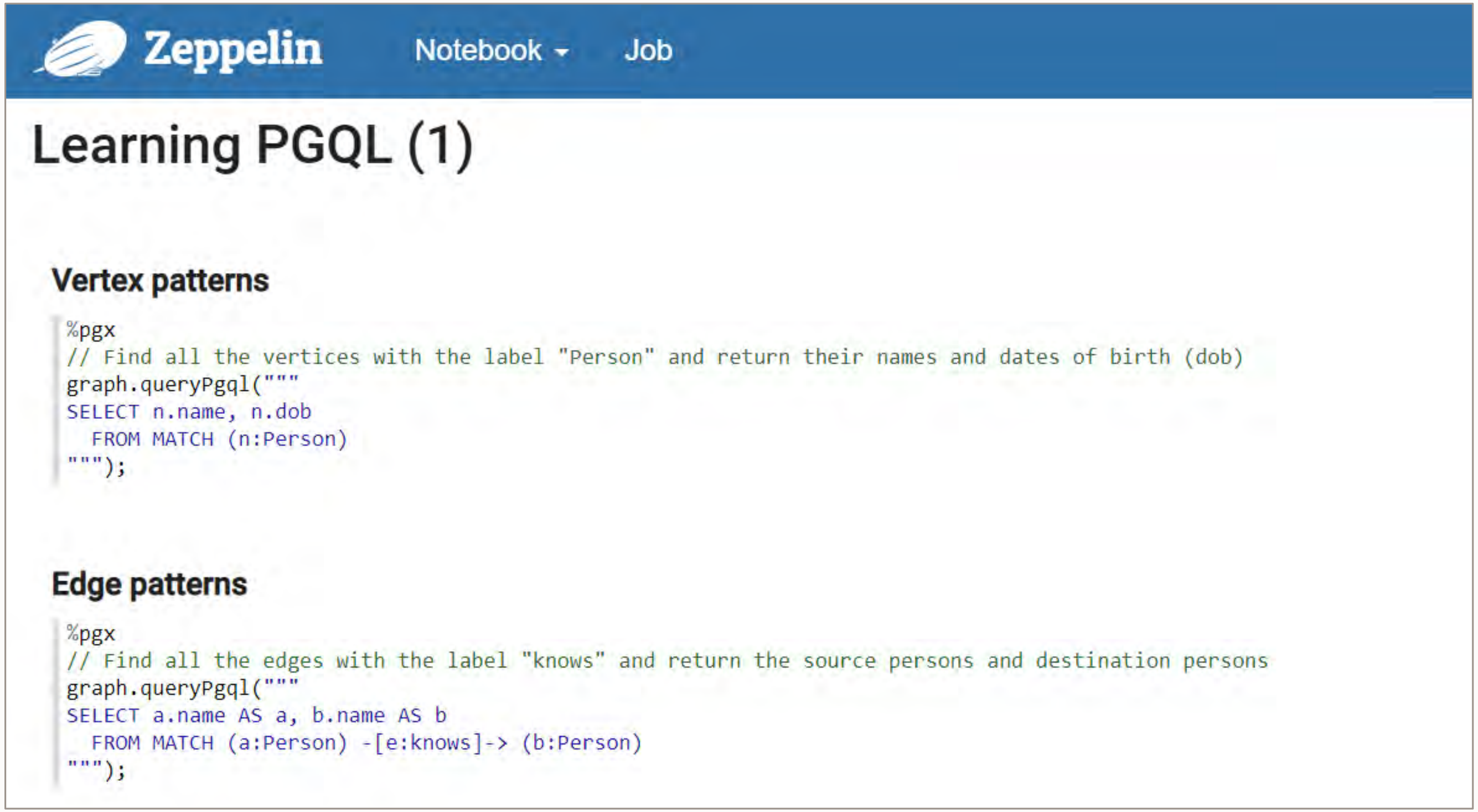
Execute PGQL in SQLcl



```
SQL> conn customer_360@dbgraphdemo_medium
Password? (**********?) **************


Connected.
SQL>
SQL>
SQL> pgql auto on

PGQL Auto enabled for graph=[null], execute=[true], translate=[false]
PGQL>
PGQL>

PGQL> select count(v) from financial_transactions match(v);


    count(v)
  ___________
8





PGQL> SELECT v.NAME from financial_transactions match(v) order by v.NAME;


     v.NAME
  __________
Camille
liam
nikita
oracle
```

Copyright © 2020, Oracle and/or its affiliates

Execute PGQL in Zeppelin



Learning PGQL (1)

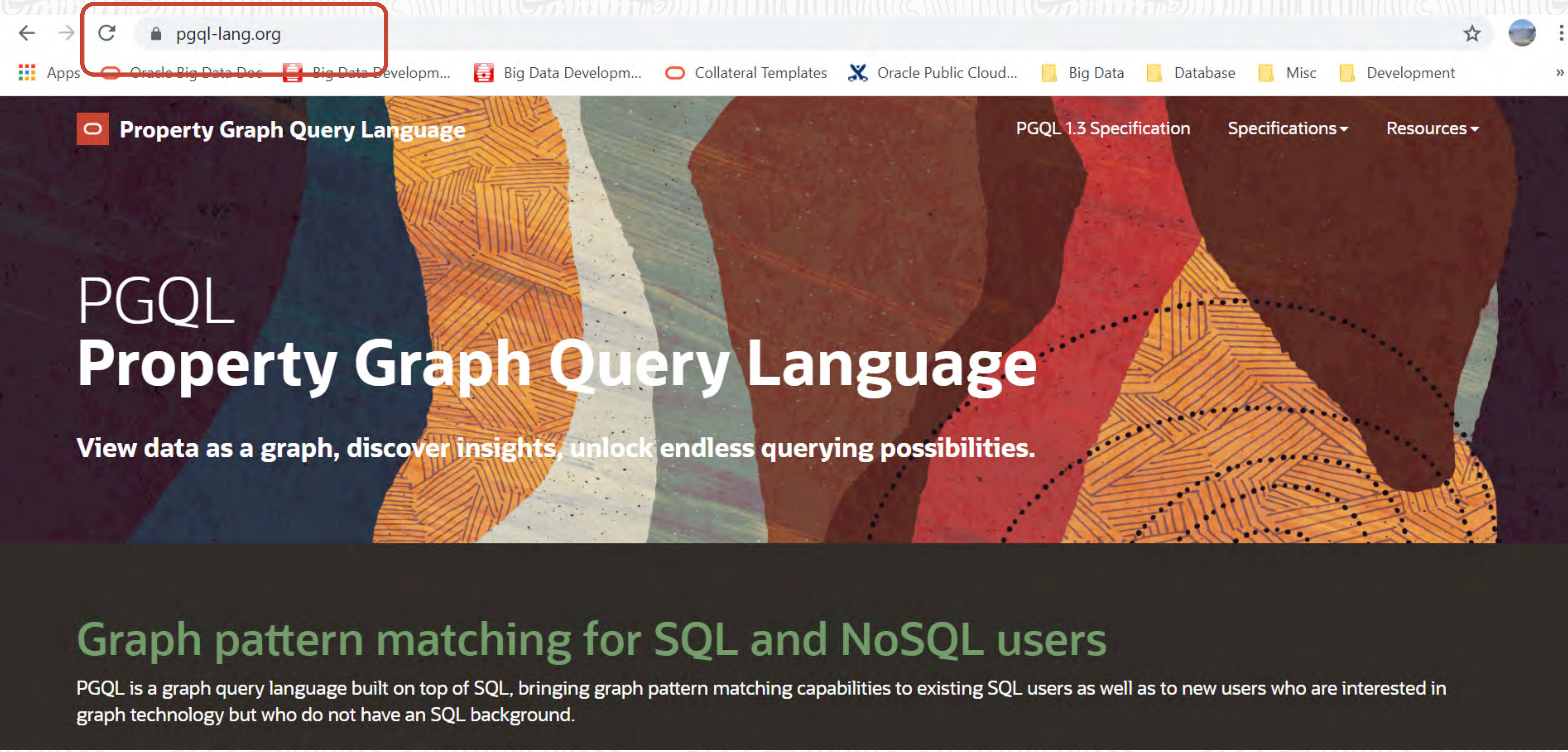
Vertex patterns

```
%pgx
// Find all the vertices with the label "Person" and return their names and dates of birth (dob)
graph.queryPgql("""
SELECT n.name, n.dob
  FROM MATCH (n:Person)
""");
```

Edge patterns

```
%pgx
// Find all the edges with the label "knows" and return the source persons and destination persons
graph.queryPgql("""
SELECT a.name AS a, b.name AS b
  FROM MATCH (a:Person) -[e:knows]-> (b:Person)
""");
```

Copyright © 2020, Oracle and/or its affiliates



PGQL
Property Graph Query Language

View data as a graph, discover insights, unlock endless querying possibilities.

Graph pattern matching for SQL and NoSQL users

PGQL is a graph query language built on top of SQL, bringing graph pattern matching capabilities to existing SQL users as well as to new users who are interested in graph technology but who do not have an SQL background.

Copyright © 2020, Oracle and/or its affiliates

Basic PGQL Syntax

Copyright © 2020, Oracle and/or its affiliates

PGQL Graph Pattern: A Few Syntax Specifications

()	Vertex Ex: (), (v), (name) represent vertices. v and name are variables
[]	Edge Ex: [], [e], [name] represent edges. e, owner are variables
-	Undirected edge
->	Directed edge
:	Specify a label (type of vertex/edge) Ex: (v:PERSON), [e:TRANSFER]
/ */	Existence of a path (zero or more hops), use / +/ for one or more paths

```
SELECT a,b,e1
FROM graph
MATCH (a)-[e1]->(b)
```

```
SELECT a,b,c,e1,e2,e3
FROM graph
MATCH (a)-[e1]->(b)-[e2]->(c)-[e3]->(a)
```

```
SELECT a,b,e1
FROM graph
MATCH (a:PERSON)-[e1:TRANSFER]->(b)
```

```
SELECT a,b
FROM graph
MATCH (a:PERSON)-/:TRANSFER*/->(b)
WHERE a.NAME=‘nikita’
```

Copyright © 2020, Oracle and/or its affiliates

Writing a PGQL Query with SQL Constructs

```
SELECT v
FROM graph
MATCH (v)
```

```
SELECT e
FROM graph
MATCH ()-[e]-()
```

```
SELECT e
FROM graph
MATCH ()-[e]->()
```

```
SELECT v
FROM graph
MATCH (v)
WHERE v.NAME=‘nikita’
```

```
SELECT e
FROM graph
MATCH ()-[e]->()
WHERE e.AMOUNT > 1000
```

```
SELECT v
FROM graph
MATCH (v)
ORDER BY v.NAME
```

```
SELECT v.NAME, e
FROM graph
MATCH (v)-[e]->()
WHERE e.amount > 1000
```

```
SELECT e
FROM graph
MATCH ()-[e:TRANSFER]->()
WHERE e.AMOUNT > 1000
```

```
SELECT v.NAME, e
FROM graph
MATCH (v:PERSON)-[e:TRANSFER]->()
WHERE e.AMOUNT > 1000
```

Copyright © 2020, Oracle and/or its affiliates

Writing a PGQL Query with SQL Functions (Aggregation)

```
SELECT COUNT(v)
FROM graph
MATCH (v)
```

```
SELECT SUM(e.AMOUNT)
FROM graph
MATCH ()-[e]-()
```

```
SELECT MAX(e.AMOUNT)
FROM graph
MATCH ()-[e]->()
```

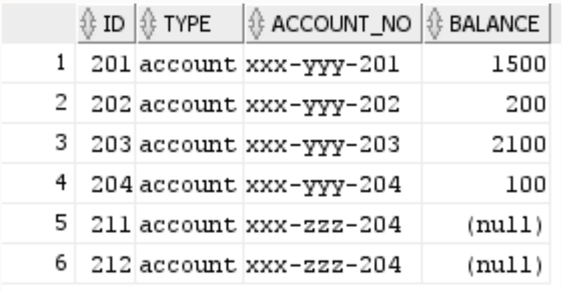
Copyright © 2020, Oracle and/or its affiliates

PGQL DDL

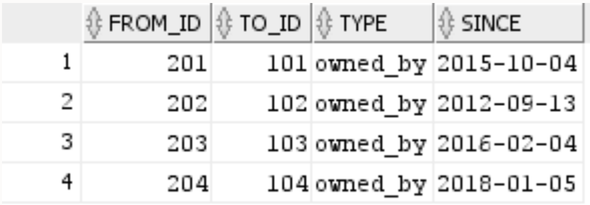
Copyright © 2020, Oracle and/or its affiliates

CREATE PROPERTY GRAPH

Accounts

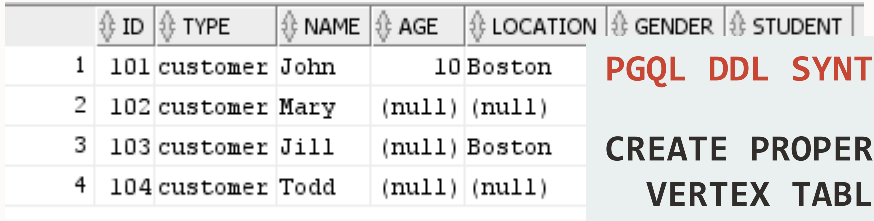


	ID	TYPE	ACCOUNT_NO	BALANCE
1	201	account	xxx-yyy-201	1500
2	202	account	xxx-yyy-202	200
3	203	account	xxx-yyy-203	2100
4	204	account	xxx-yyy-204	100
5	211	account	xxx-zzz-204	(null)
6	212	account	xxx-zzz-204	(null)

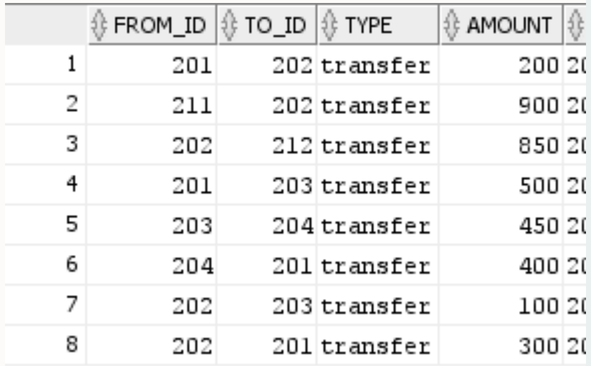


	FROM_ID	TO_ID	TYPE	SINCE
1	201	101	owned_by	2015-10-04
2	202	102	owned_by	2012-09-13
3	203	103	owned_by	2016-02-04
4	204	104	owned_by	2018-01-05

Customers



	ID	TYPE	NAME	AGE	LOCATION	GENDER	STUDENT
1	101	customer	John	10	Boston		
2	102	customer	Mary	(null)	(null)		
3	103	customer	Jill	(null)	Boston		
4	104	customer	Todd	(null)	(null)		



	FROM_ID	TO_ID	TYPE	AMOUNT	
1	201	202	transfer	200	20
2	211	202	transfer	900	20
3	202	212	transfer	850	20
4	201	203	transfer	500	20
5	203	204	transfer	450	20
6	204	201	transfer	400	20
7	202	203	transfer	100	20
8	202	201	transfer	300	20

PGQL DDL SYNTAX:

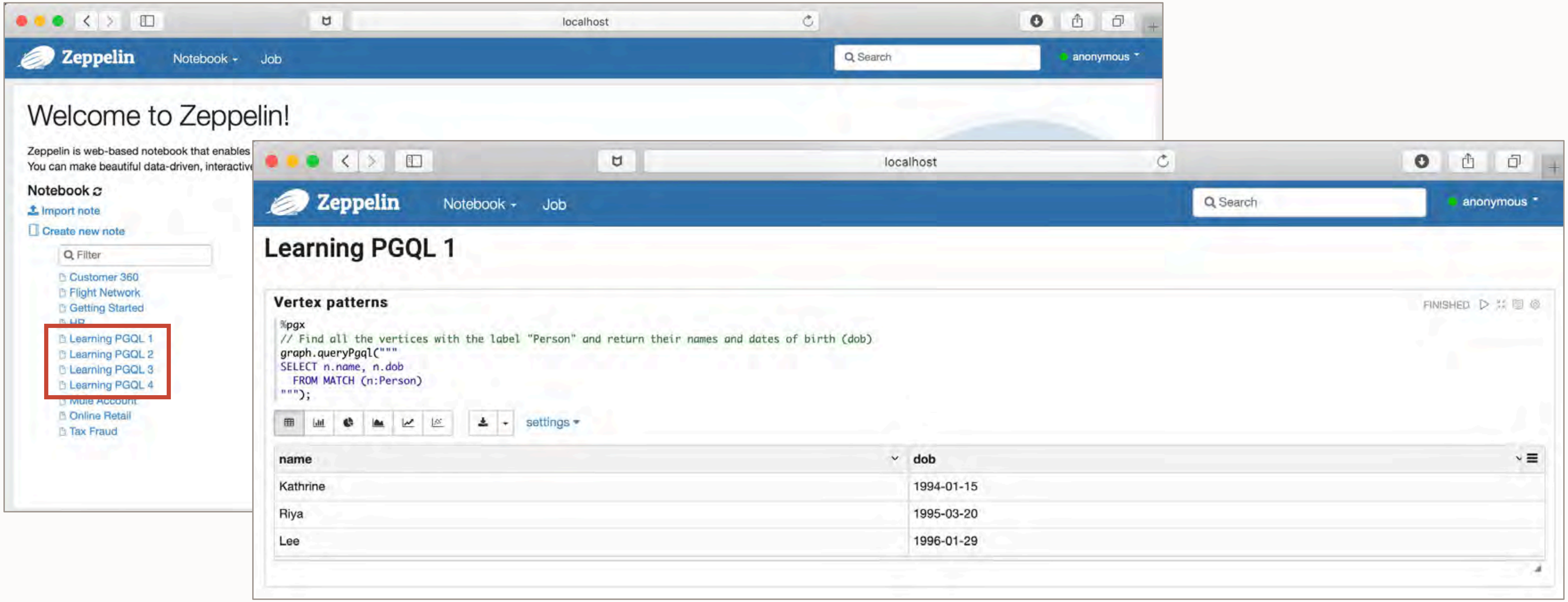
```
CREATE PROPERTY GRAPH customer_360
  VERTEX TABLES (
    customers PROPERTIES ALL COLUMNS EXCEPT(id)
  , accounts PROPERTIES ALL COLUMNS EXCEPT(id)
  )
  EDGE TABLES (
    owned_by
      SOURCE KEY(from_id) REFERENCES accounts
      DESTINATION KEY(to_id) REFERENCES customers
  , transfer
      SOURCE KEY(from_id) REFERENCES accounts
      DESTINATION KEY(to_id) REFERENCES accounts
  )
```

Copyright © 2020, Oracle and/or its affiliates

Try Running PGQL

Copyright © 2020, Oracle and/or its affiliates

Setup Your Graph Server



Welcome to Zeppelin!

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

Notebook

- Customer 360
- Flight Network
- Getting Started
- HR
- Learning PGQL 1
- Learning PGQL 2
- Learning PGQL 3
- Learning PGQL 4
- Mule Account
- Online Retail
- Tax Fraud

Learning PGQL 1

Vertex patterns

```
%pgx
// Find all the vertices with the label "Person" and return their names and dates of birth (dob)
graph.queryPgql("""
SELECT n.name, n.dob
  FROM MATCH (n:Person)
""");
```

name	dob
Kathrine	1994-01-15
Riya	1995-03-20
Lee	1996-01-29

Copyright © 2020, Oracle and/or its affiliates

Setup Your Graph Server

Installation: https://github.com/ryotayamanaka/oracle-pg/tree/20200730

Clone repository (Note, the tag name is 20200730)

```
$ git clone https://github.com/ryotayamanaka/oracle-pg.git -b 20200730
```

Download and extract packages (Note, the packages are version 20.2, not 20.3 yet)

```
$ sh extract.sh
```

Build and start the docker containers

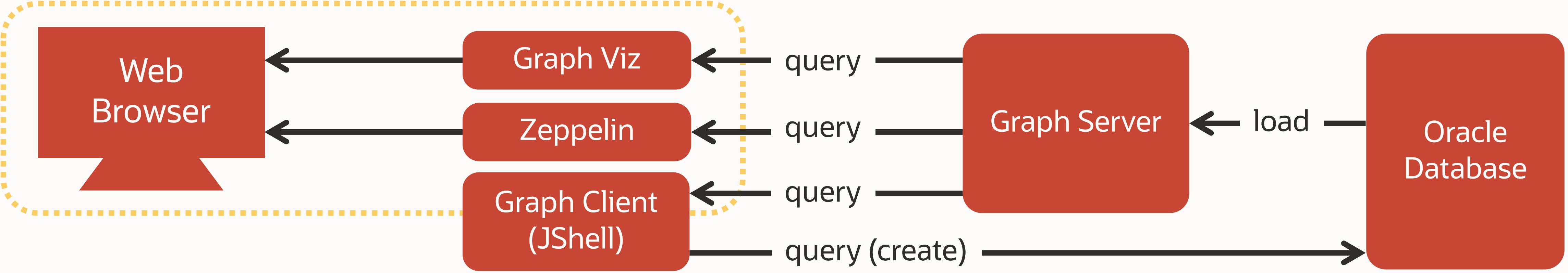
```
$ docker-compose up -d
```

Copyright © 2020, Oracle and/or its affiliates

Setup Your Graph Server

The containers includes **Graph Server**, **Graph Viz**, and **Zeppelin** (with the **tutorials**).

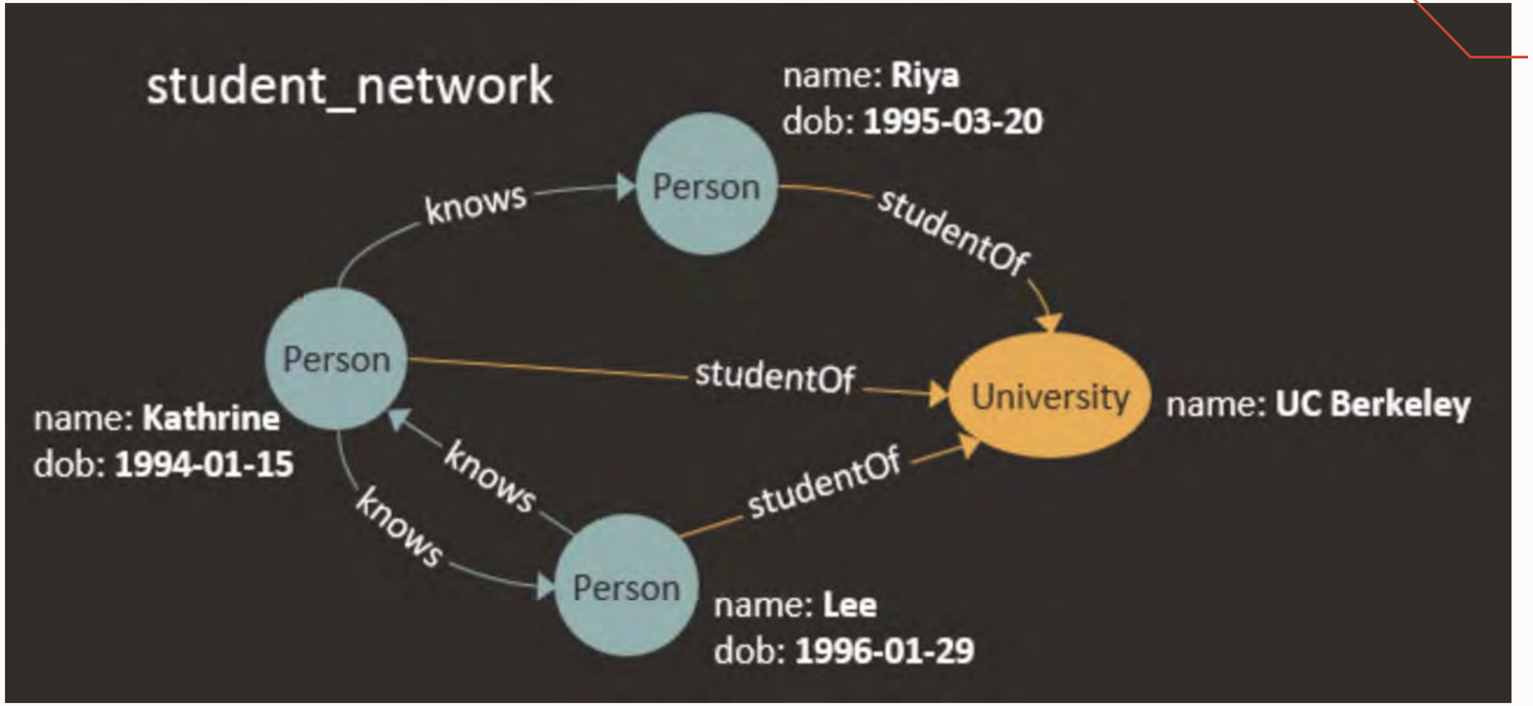
Oracle Database is optional, as the small sample datasets can be loaded from files.



Copyright © 2020, Oracle and/or its affiliates

Learning PGQL 1 - Graph Pattern Matching

This tutorial shows basic graph pattern matching using Student Network dataset.

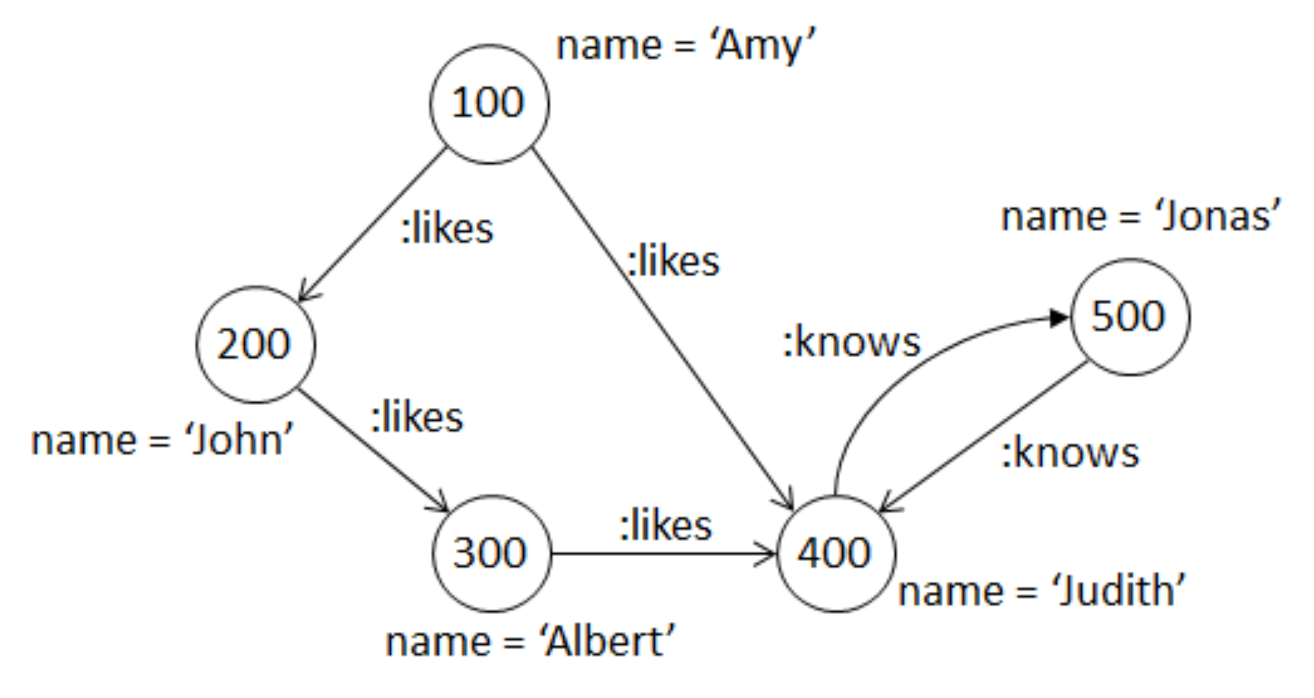


All the tutorials follow the PGQL online documentation http://pgql-lang.org

Copyright © 2020, Oracle and/or its affiliates

Learning PGQL 2 - Variable Length Paths (Reachability)

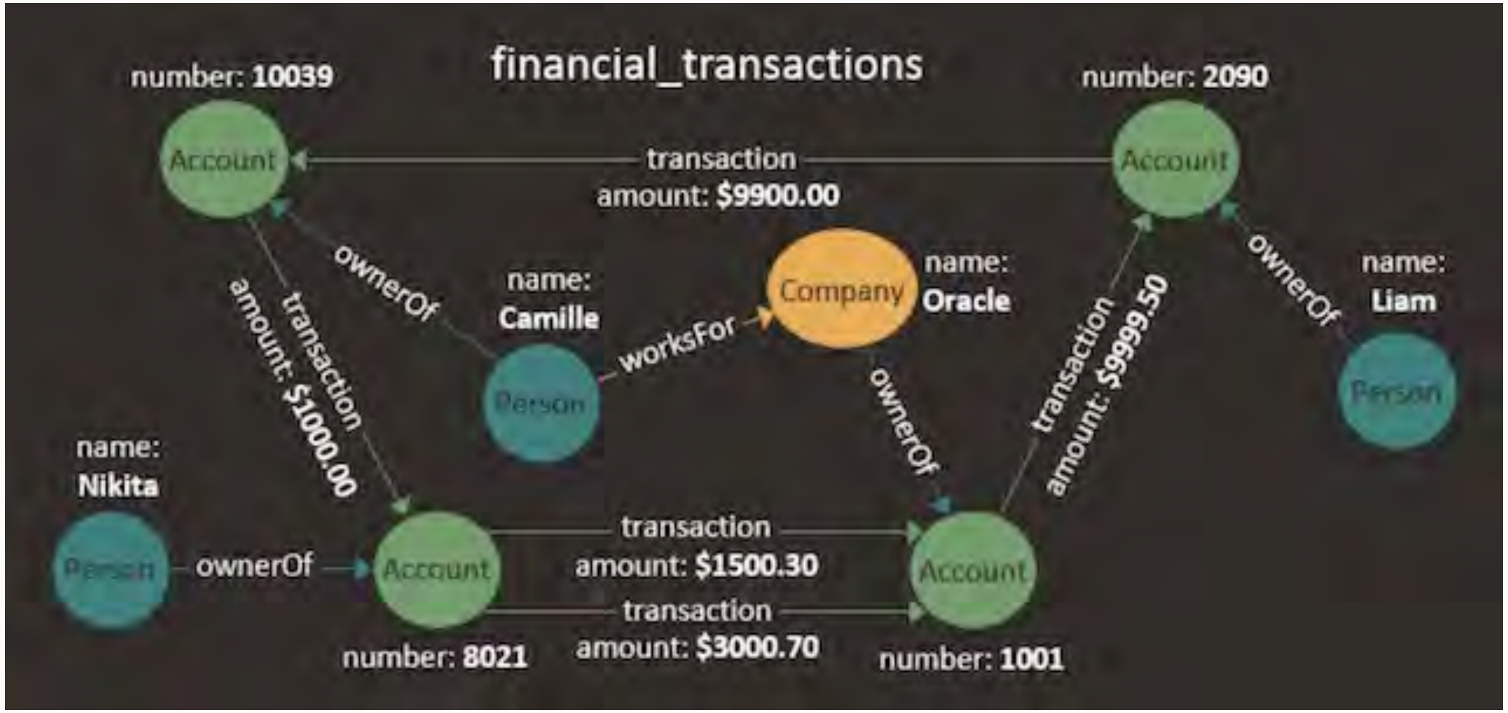
This tutorial shows how to test reachability in graph pattern matching.



Copyright © 2020, Oracle and/or its affiliates

Learning PGQL 3 - Variable Length Paths (Shortest Path)

This tutorial shows how to find shortest paths using Financial Transactions dataset.



Copyright © 2020, Oracle and/or its affiliates

Learning PGQL 4 - Graph Modification

This tutorial shows how to modify graph using insert, update, and delete statements.

```
INSERT VERTEX x LABELS ( Male ) PROPERTIES ( x.age = y.age )
  FROM MATCH (y:Male)
```

```
UPDATE x SET ( x.age = 42 )
  FROM MATCH (x:Person) WHERE x.name = 'John'
```

```
DELETE x
  FROM MATCH (x)
 WHERE id(x) = 11
```

Copyright © 2020, Oracle and/or its affiliates

Methods to Run PGQL

For running PGQL queries on Zeppelin Notebook (PGX interpreter), we use two methods.

```
graph.queryPgql("""
  SELECT p.name
    FROM MATCH (p:Person)
""");
```

```
graph2 = graph.cloneAndExecutePgql("""
  INSERT VERTEX x
         LABELS ( Male )
""");
```

```
| name |
+------+
| Amy  |
```

Copyright © 2020, Oracle and/or its affiliates

Methods to Run PGQL

To see execution plans, use the following method.

```
graph.explainPgql("""
  SELECT p1.name AS p1, p2.name AS p2, p3.name AS p3
    FROM MATCH (p1:Person) -[:knows]-> (p2:Person) -[:knows]-> (p3:Person)
   WHERE p1.name = 'Lee' AND ALL_DIFFERENT(p1, p3)
""");
```

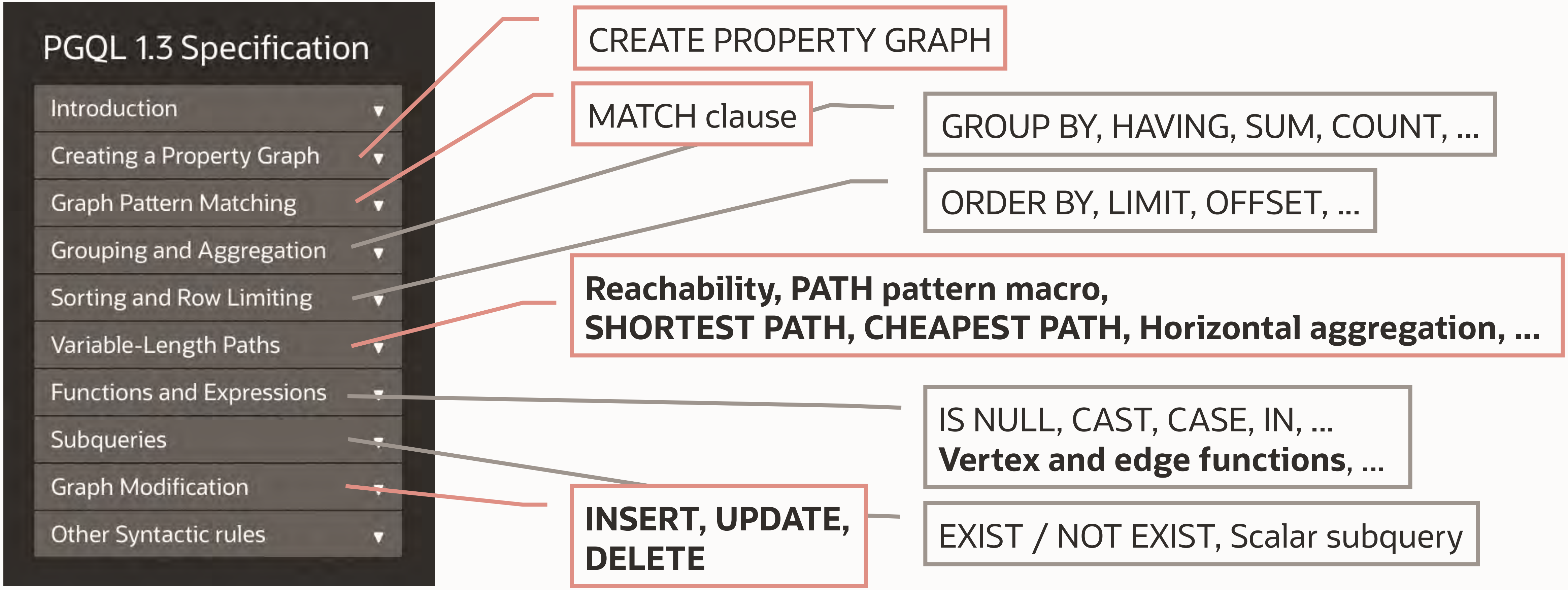
```
\--- (P2) -["anonymous_2"]-> (P3) NeighborMatch {"cardinality":"84.4E-3", ...}
 \--- (P1) -["anonymous_1"]-> (P2) NeighborMatch {"cardinality":"113E-3", ...}
  \--- (P1) RootVertexMatch {"cardinality":"150E-3", "cost":"150E-3", ...}
```

Copyright © 2020, Oracle and/or its affiliates

Advanced Topics

Copyright © 2020, Oracle and/or its affiliates

Overview of PGQL Elements



PGQL 1.3 Specification

- Introduction
- Creating a Property Graph
- Graph Pattern Matching
- Grouping and Aggregation
- Sorting and Row Limiting
- Variable-Length Paths
- Functions and Expressions
- Subqueries
- Graph Modification
- Other Syntactic rules

CREATE PROPERTY GRAPH

MATCH clause

GROUP BY, HAVING, SUM, COUNT, ...

ORDER BY, LIMIT, OFFSET, ...

Reachability, PATH pattern macro,
SHORTEST PATH, CHEAPEST PATH, Horizontal aggregation, ...

IS NULL, CAST, CASE, IN, ...
Vertex and edge functions, ...

EXIST / NOT EXIST, Scalar subquery

INSERT, UPDATE,
DELETE

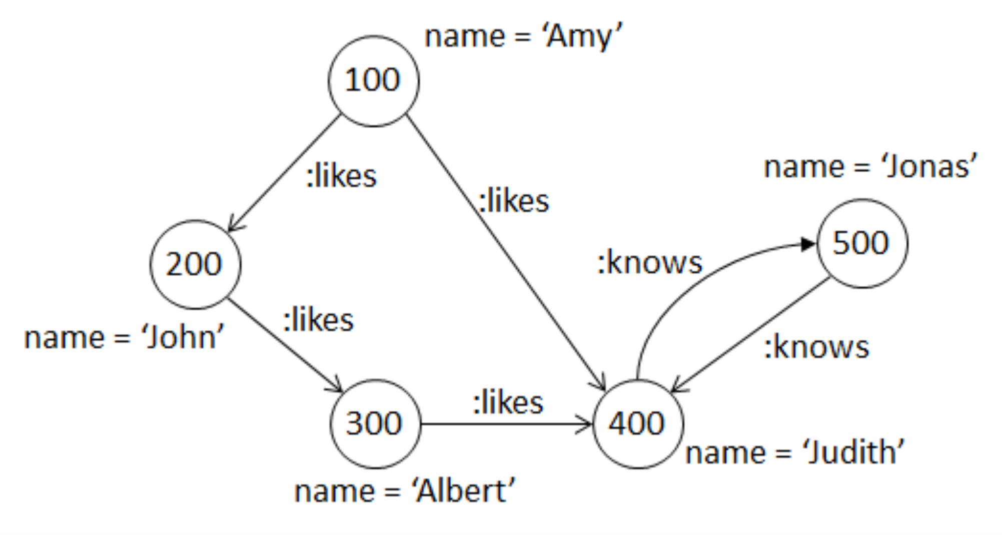
Copyright © 2020, Oracle and/or its affiliates

Reachability

To test the existence of paths (true/false) between vertices, slashes are used instead of square brackets.

```
SELECT y.name
  FROM MATCH (x:Person) -/:likes*/-> (y)
 WHERE x.name = 'Amy'
```

```
SELECT y.name
  FROM MATCH (x:Person) -/:likes{2,3}/-> (y)
 WHERE x.name = 'Amy'
```



Copyright © 2020, Oracle and/or its affiliates

Path Pattern Macro

Path patterns can be declared using PATH .. AS .. syntax.

```
PATH has_parent AS () -[:has_father|has_mother]-> (:Person)

SELECT ancestor.name
  FROM MATCH (p1:Person) -/:has_parent+/-> (ancestor)
     , MATCH (p2:Person) -/:has_parent+/-> (ancestor)
 WHERE p1.name = 'Mario'
   AND p2.name = 'Luigi'
```

Copyright © 2020, Oracle and/or its affiliates

Shortest Path

This quey finds the shortest path (= one of the shortest paths) between two vertices.

```
SELECT COUNT(e) AS num_hops
     , SUM(e.amount) AS total_amount
     , ARRAY_AGG(e.amount) AS amounts_along_path
  FROM MATCH SHORTEST ( (a:Account) -[e:transaction]->* (b:Account) )
 WHERE a.number = 10039
   AND b.number = 2090
 ORDER BY num_hops, total_amount
```

```
| num_hops | total_amount | amounts_along_path       |
+----------------------------------------------------+
| 3        | 12499.8      | [1000.0, 1500.3, 9999.5] |
```

Note, the amounts are aggregated without GROUP BY clause (= horizontal aggregation)

Copyright © 2020, Oracle and/or its affiliates

Shortest Path (Top K)

This quey finds the the k shortest paths between a source vertex and a destination vertex.

```
SELECT COUNT(e) AS num_hops
     , SUM(e.amount) AS total_amount
     , ARRAY_AGG(e.amount) AS amounts_along_path
  FROM MATCH TOP 3 SHORTEST ( (a:Account) -[e:transaction]->* (b:Account) )
 WHERE a.number = 10039
   AND b.number = 2090
 ORDER BY num_hops, total_amount
```

```
| num_hops | total_amount | amounts_along_path                                      |
+-------------------------------------------------------------------------------------+
| 3        | 12499.8      | [1000.0, 1500.3, 9999.5]                                |
| 3        | 14000.2      | [1000.0, 1500.3, 9999.5]                                |
| 7        | 34899.6      | [1000.0, 1500.3, 9999.5, 9900.0, 1000.0, 1500.3, 9999.5] |
```

Copyright © 2020, Oracle and/or its affiliates

Cheapest Path

This quey finds the cheapest path based on the cost (= edge property "amount", in this case).

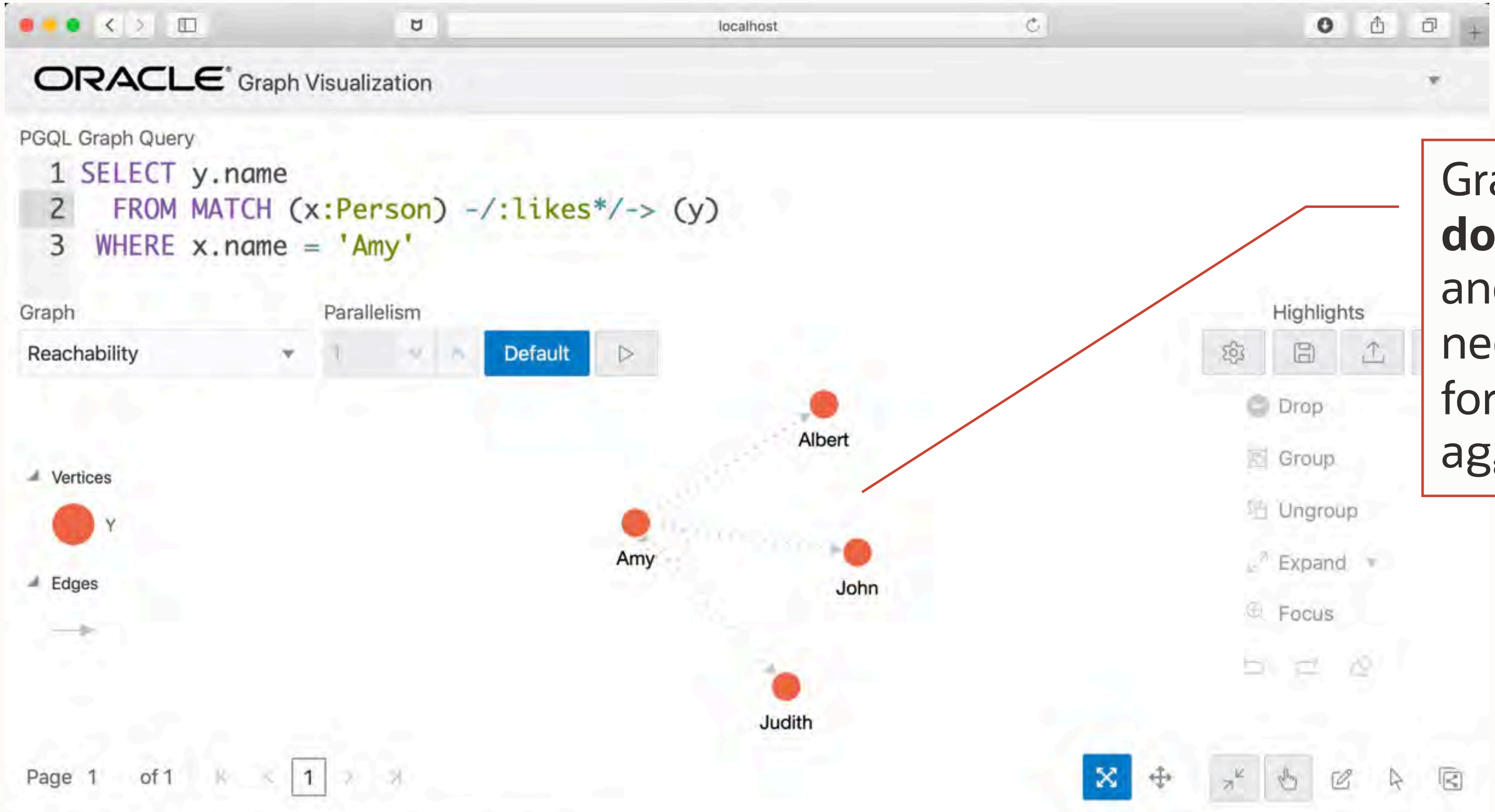
```
SELECT COUNT(e) AS num_hops
     , SUM(e.amount) AS total_amount
     , ARRAY_AGG(e.amount) AS amounts_along_path
  FROM MATCH CHEAPEST ( (a:Account) (-[e:transaction]-> COST e.amount)* (b:Account) )
 WHERE a.number = 10039
   AND b.number = 2090
 ORDER BY num_hops, total_amount
```

```
| num_hops | total_amount | amounts_along_path       |
+----------------------------------------------------+
| 3        | 12499.8      | [1000.0, 1500.3, 9999.5] |
```

The cost is the selected edge property, and the path with the the smallest total cost is returned.

Copyright © 2020, Oracle and/or its affiliates

Paths in GraphViz



```
SELECT y.name
  FROM MATCH (x:Person) -/:likes*/-> (y)
 WHERE x.name = 'Amy'
```

GraphViz shows the paths as **dotted lines**. To get the nodes and edges on the paths, you need to get the results in table format, using horizontal aggregation.

Copyright © 2020, Oracle and/or its affiliates

SQL/PGQ

SQL extensions to query property graphs

- Our team is working with ISO and ANSI committees
- Target: Next version of SQL

Create a property graph using SQL data definition

```
CREATE PROPERTY GRAPH myGraph
         VERTEX TABLES (Person, Message)
         EDGE TABLES (
     Created SOURCE Person DESTINATION Message,
     Commented SOURCE Person DESTINATION Message )
```

Query a property graph in SQL

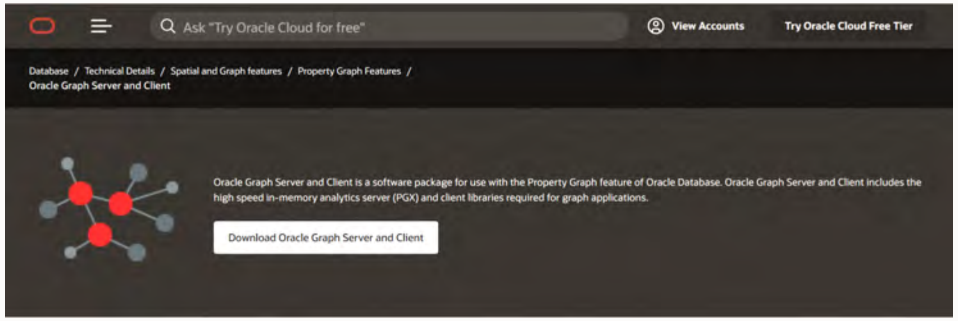
```
SELECT GT.creationDate, GT.content
FROM myGraph GRAPH_TABLE (
  MATCH
   (Creator IS Person WHERE Creator.email = :email1)
          -[ IS Created ]->
   (M IS Message)
          <-[ IS Commented ]-
   (Commenter IS Person WHERE Commenter.email = :email2)
      WHERE ALL_DIFFERENT (Creator, Commenter)
  ONE ROW PER MATCH
  COLUMNS (
          M.creationDate,
          M.content )
 ) AS GT
```

Copyright © 2020, Oracle and/or its affiliates

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, @JeanIhm, @ryotaymnk
  - LinkedIn: Oracle Spatial and Graph Group
  - YouTube: youtube.com/c/OracleSpatialandGraph

Search for "Oracle Graph Server and Client" to download from oracle.com



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.

Copyright © 2020, Oracle and/or its affiliates

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

Copyright © 2020, Oracle and/or its affiliates

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

ORACLE

- Contents
  - Contents
- Contents
  - Contents
    - item 1
    - item 2
    - item 3
    - item 4

Copyright © 2020, Oracle and/or its affiliates