ORACLE

Property Graph Support in Relational Database

Data Community Conference
November, 3rd 2022
Bern

Presenter Introduction

Marco Arnaboldi

Senior Member of Technical Staff

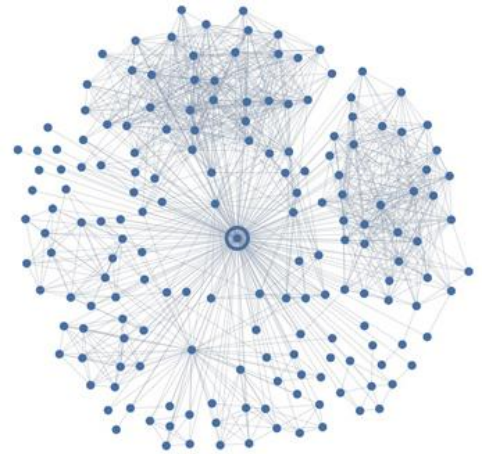
Oracle Labs

marco.arnaboldi@oracle.com

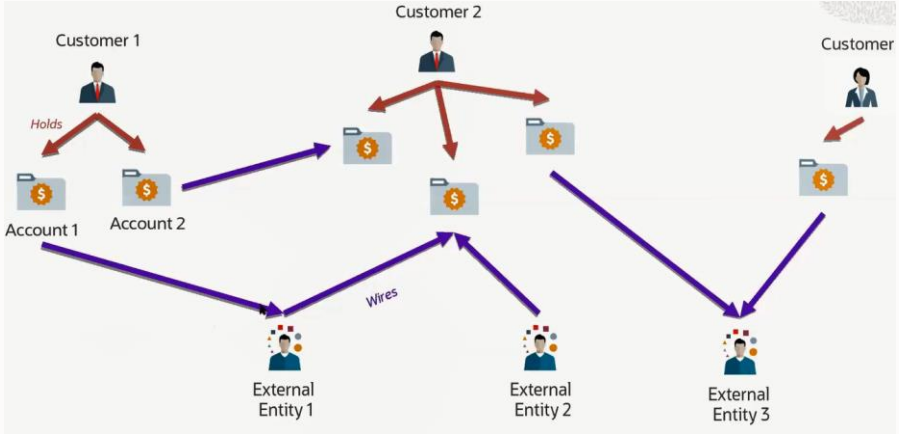
Marco's main area of work is on scalable graph processing and DB native graph technology. He has received his Master degree from Politecnico di Milano and University of Illinois. He has been working for Oracle Labs for more than 3 years.

Connections are everywhere around us

Your social network …



Your bank transactions …



The transportation network …

The supply chain network…

The interactions between us …

…

Graphs are a powerful tool to leverage information stored in connections of your data.

Graph is an important trend in Data and Analytics

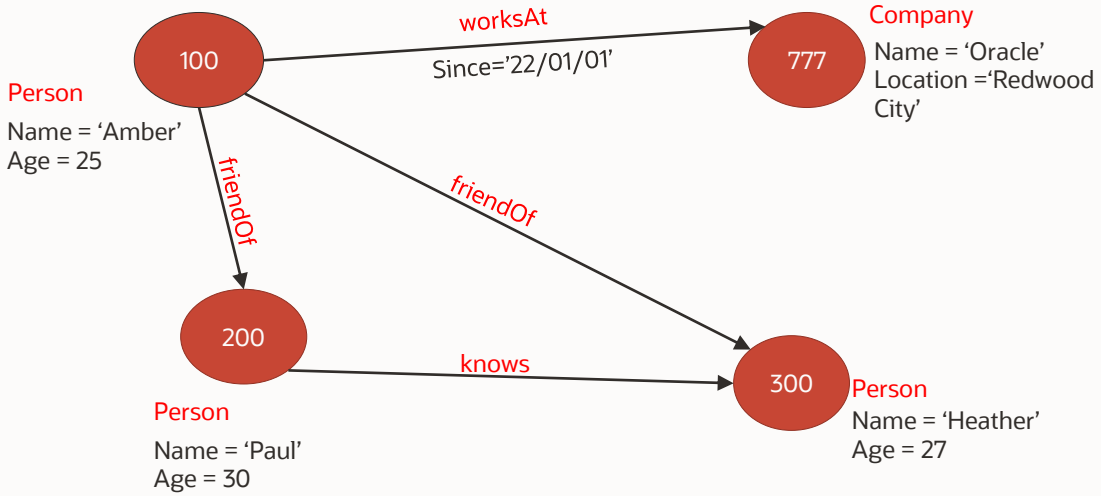
This is because graphs provide intuitive abstractions
… for integrating various data sources
… for extracting signals stored in the data connections

Trend No. 8: Graph relates everything

Graph forms the foundation of modern data and analytics with capabilities to enhance and improve user collaboration, machine learning models and explainable AI. Although graph technologies are not new to data and analytics, there has been a shift in the thinking around them as organizations identify an increasing number of use cases. In fact, as many as 50% of Gartner client inquiries around the topic of AI involve a discussion around the use of graph technology.

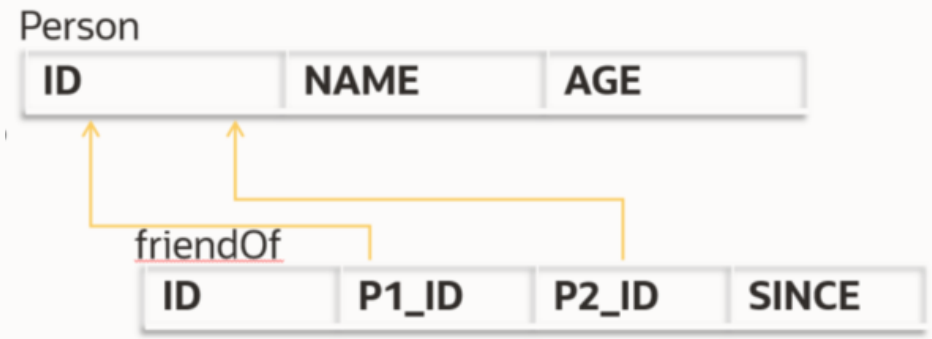
What is a property graph? (simplified)

- A set of **vertices** (sometimes called nodes)
- And **edges** that connect vertices
- Vertices and edges can
  - Have **labels** (one or more)
    - A label is an identifier
    - That also provided typing information
  - Have **properties**/attributes (zero or more)
    - By virtue of properties being associated with labels
- A property is a typed key/value pair
- Vertices and edges collectively are called graph elements

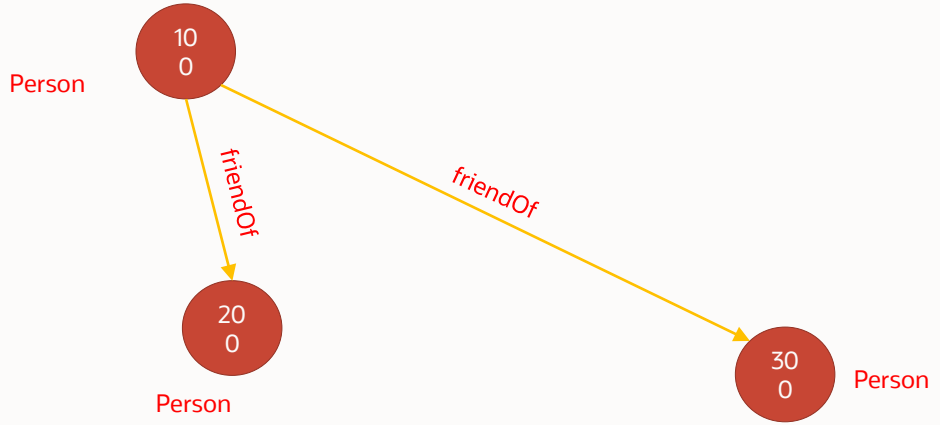


How is property graph data model different from relational?

- In relational data model, **connections** are encoded **implicitly** (**FK/PK**, joins)
- In property graph data model , connections are **explicit** part of the data model in form of **edges** between entities (**vertices**)



+



Graphs enable a variety of new use-cases

Money laundering detection for financial services

Create a graph of all bank accounts, then run graph queries to find all customers who have information that reveals criminal activity.

Data lineage tracking for GDPR

The various steps in the data lifecycle can be tracked and navigated, vertex by vertex, by following the edges in a graph. Follow the data's path, see where the information originally resided, was copied, and utilized, so data professionals can fulfil GDPR requests.

Feature engineering for machine learning

Building machine learning models requires augmented data, which can be created by running graph queries and creating enriched data which can then be used for machine learning.

...

Increasingly used in many industry domains

Finance

- Fraud detection
- Risk analytics
- Identify communities among customers

Transportation

- Identify vulnerable points in infrastructure, single points of failure
- Aggregate costs along paths

Manufacturing

- Simplify application that enables users to navigate full product hierarchies in their e-commerce system

Telecommunications

- Analyze and optimize network topologies

Graph View vs. Relational View

Relational model is widely used for general data management

However, the model is cumbersome for answering certain questions

Are Bob and Charlie related (and how)?
Is there any money flow between them?

Is there any money flowing in cycles back to the originating owner (laundering)?

Account Table

Account ID	Owner ID	Creation Date
1111	200	2010-3-10
2222	100	2011-2-13
3333	400	2015-9-16
4444	300	2012-5-25
5555	100	...

Customer Table

Owner ID	Name
100	Alice
200	Bob
300	Charlie
400	Dave
...	

Transfer Table

SRC	DEST	Type	Amount
1111	3333	Wire	$20,000
5555	4444	Wire	$30,000
4444	2222	Recurring	$10,000
3333	5555	Wire	$20,000
....			

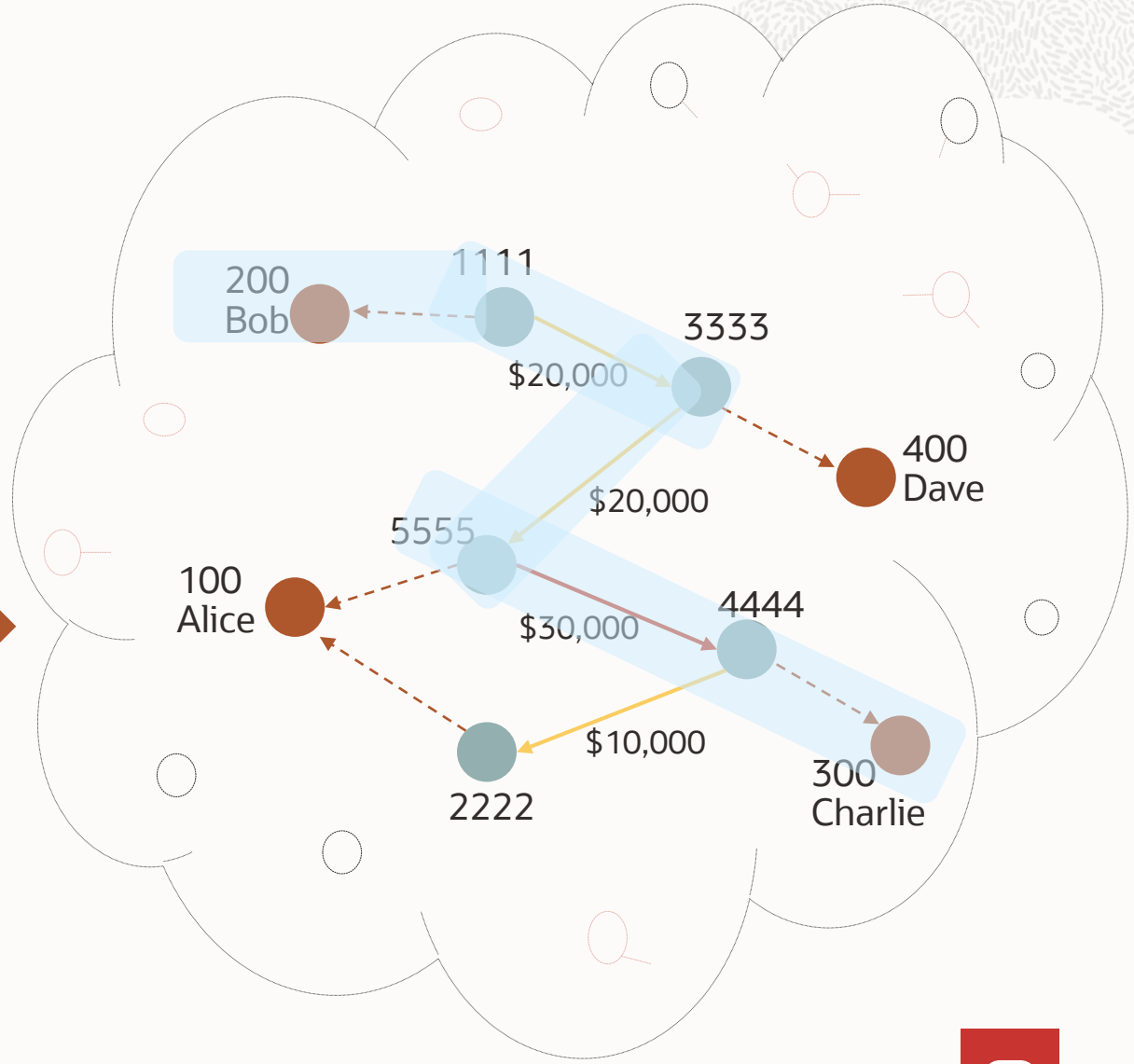
Benefits from Graph View

Represent the dataset as a graph

- Entities become vertices
- Relationships become edges

Previous Qs more intuitive to answer with graph representation

Are Bob and Charlie related (and how)?
Is there any money flow between them?



Benefits from Graph View

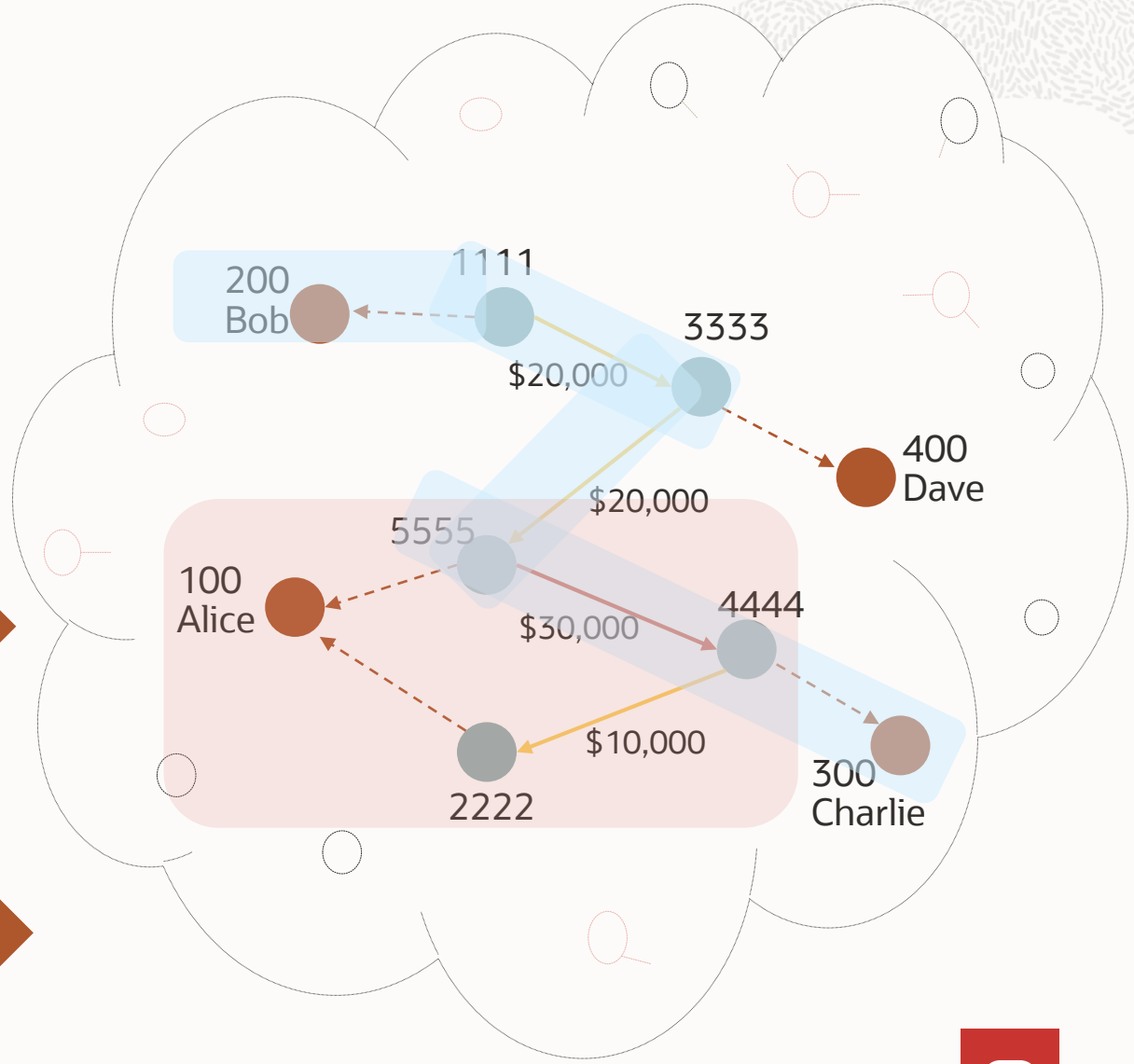
Represent the dataset as a graph

- Entities become vertices
- Relationships become edges

Previous Qs more intuitive to answer with graph representation

Are Bob and Charlie related (and how)?
Is there any money flow between them?

Is there any money flowing in cycles back to the originating owner (laundering)?



Property Graph support in Oracle Database

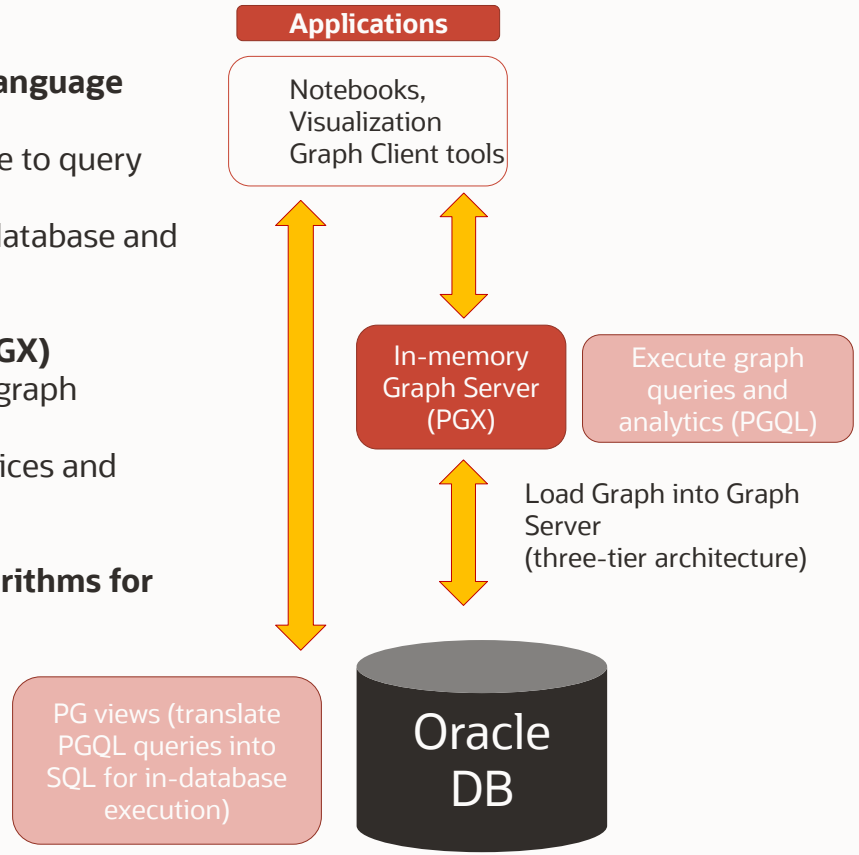
Property Graph Query Language (PGQL)
- SQL-like query language to query graphs
- Execute queries in the database and in PGX

Mid-tier graph server (PGX)
- Extremely fast, parallel graph analytics
- Scales to billions of vertices and edges

60+ pre-built graph algorithms for powerful analytics
- Java and Python API
- Execute in PGX

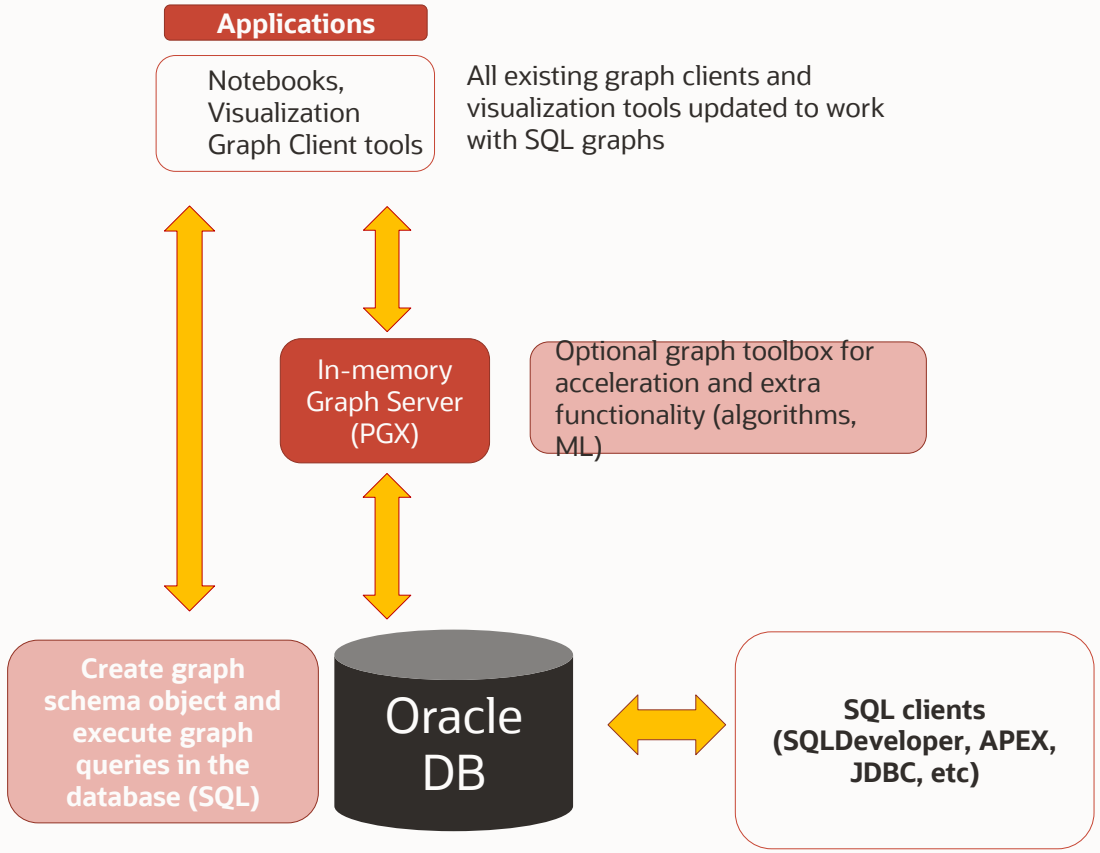
Products
- ADB-Graph
- Oracle Graph Server and Client

Since Oracle 12c



Starting Oracle 23c

Graphs can be used directly in SQL !



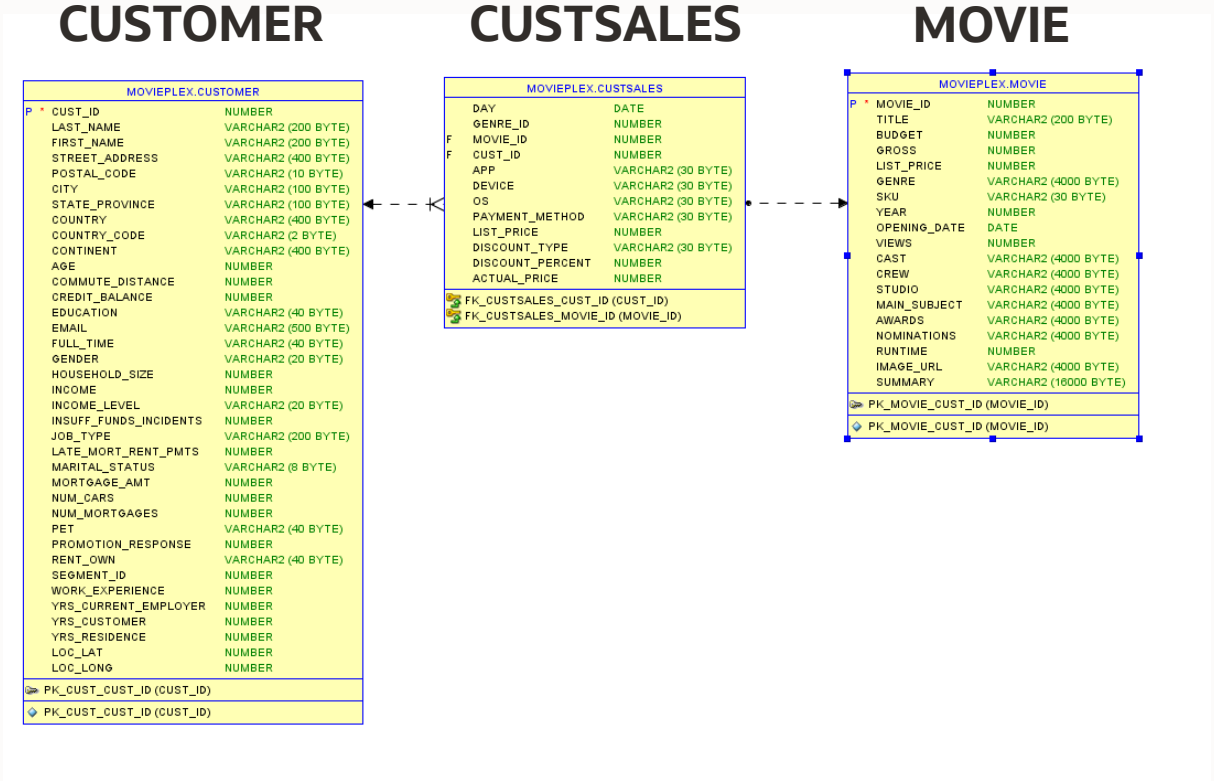
Property Graph Support in SQL – starting Oracle 23c

Quick facts

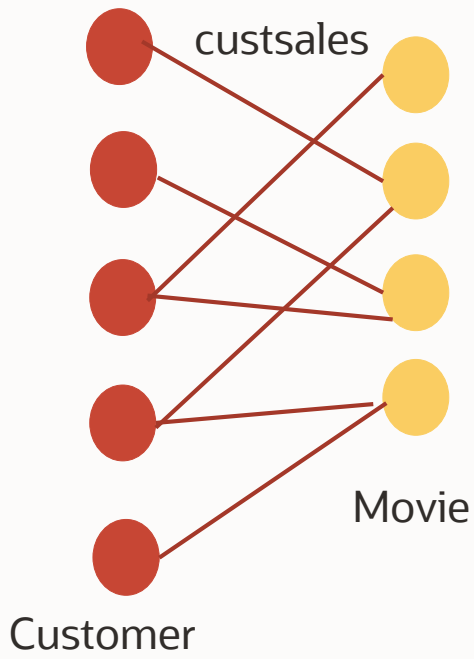
- SQL property graphs are **part of ISO** standard: ISO/IEC 9075-16 (SQL/PGQ language)
- SQL property graphs specify two major capabilities:
  - **Creating property graph** on top of existing tables in an relational database
  - Querying property graphs using a **GRAPH_TABLE** operator in an SQL FROM clause
- SQL graphs are created as metadata object over original data:
  - No data copy
  - Transactional consistency
- SQL property graphs **strengthen Oracle's converged database vision:** graph from relational data as another form of business object view for 23c
- SQL property graphs achieve **extreme scalability** by leveraging existing SQL execution engine.

SQL / PGQ – Let's start with a simple example

Relational schema



Graph Schema



SQL Property Graph Creation

Concise syntax when required metadata exists.

i.e. primary and foreign keys, uniqueness constraints

```
CREATE PROPERTY GRAPH MovieRentals
  VERTEX TABLES (
    Customer, Movie
  )
  EDGE TABLES (
    CustSales SOURCE Customer DESTINATION Movie
  );
```

Graph created as a metadata object over original data

- No data copy
- Transactional consistency

SQL Property Graph Creation – Explicit Syntax

```
CREATE PROPERTY GRAPH MovieRentals
  VERTEX TABLES (
   Customer KEYS (Cust_ID)
     PROPERTIES (First_Name, Last_Name, Gender),
   Movie KEYS (Movie_ID)
     PROPERTIES (Title, Genre)
 )
 EDGE TABLES (
  Custsales
    PROPERTIES (Day AS Date_Rented)
    SOURCE      Custsales KEYS(Cust_ID)  REFERENCES Customers (Cust_ID)
    DESTINATION Custsales KEYS(Movie_ID) REFERENCES Movie (Movie_ID)
 );
```

Syntax for explicitly defining keys

Syntax for explicitly defining properties

Syntax for explicitly naming properties

Syntax for explicitly defining edge relationships

Querying SQL Graphs

```
SELECT … FROM
GRAPH_TABLE (
    <graph name>                              -- input graph
    MATCH <graph pattern>                     -- pattern to match
    WHERE <conditions>                        -- conditions to satisfy
    COLUMNS (<columns to return>)             -- return type of result table
)
GROUP BY …
ORDER BY …
```

Querying SQL Graphs - Example

Find all two customers who rented the same romantic comedy movie one after the other and after the 14th Feb 2022.

```
SELECT *
FROM GRAPH_TABLE(MovieRentals
  MATCH
    (cust1 IS Customer)-[e1]->
    (movie IS Movie)<-[e2]-(cust2 IS Customer)
  WHERE e1.Date_Rented > TO_DATE('14-Feb-2022')
    AND movie.genre='Romantic Comedy'
    AND e2.Date_Rented > e1.Date_Rented AND cust1.Last_Name!= cust2.Last_Name
  COLUMNS(cust1.Last_Name AS EarlierRenter, cust2.Last_Name AS LaterRenter,
          e1.Date_Rented))
ORDER BY Date_Rented;
```

Property Graph

Vertex

Label

Edge

Path Pattern

Simple Predicate

Cross-variable predicate

Row type of the result table

A real life example made easy with graph

Assume Bob wants to take a loan to buy an expensive house in New York

The bank would like to investigate the credit worthiness of Bob by:

- Analyzing the credit worthiness of Bob's social circle
- Investigating credit worthy work colleagues who can recommend Bob
- Finding friends of friends of friends of Bob working in New York, in order to check his ability to socially integrate in the new city.

Let's assume the following relational schema

Vertex tables

```
CREATE TABLE person (
  id NUMBER (5) PRIMARY KEY ,
  details JSON
);
```

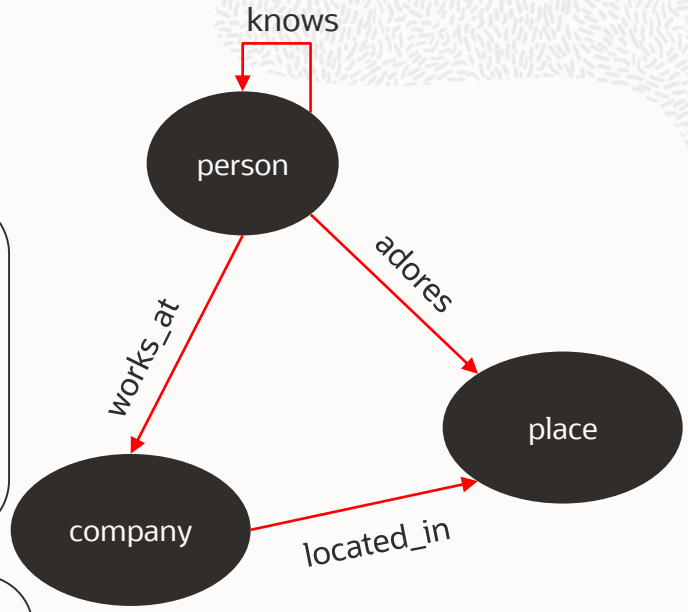
```
CREATE TABLE company (
  id NUMBER (5) PRIMARY KEY ,
  name VARCHAR (100) ,
  age NUMBER (5) ,
  size_c NUMBER (10)
);
```

```
CREATE TABLE place(
  id NUMBER (5) PRIMARY KEY ,
  name VARCHAR (100) ,
  size_p NUMBER (10)
);
```

Edge tables

```
CREATE TABLE knows (
  e_src NUMBER (5) NOT NULL ,
  e_dst NUMBER (5) NOT NULL ,
  since NUMBER (5),
  CONSTRAINT pk1 PRIMARY KEY(e_src, e_dst),
  CONSTRAINT fk1 FOREIGN KEY (e_src) REFERENCES person(id),
  CONSTRAINT fk2 FOREIGN KEY (e_dst) REFERENCES person(id)
);
```

```
CREATE TABLE adores (
  e_src NUMBER (5) NOT NULL ,
  e_dst NUMBER (5) NOT NULL ,
  CONSTRAINT pk_a PRIMARY KEY(e_src, e_dst),
  CONSTRAINT fk_a1 FOREIGN KEY (e_src) REFERENCES person(id),
  CONSTRAINT fk_a2 FOREIGN KEY (e_dst) REFERENCES place(id)
);
```

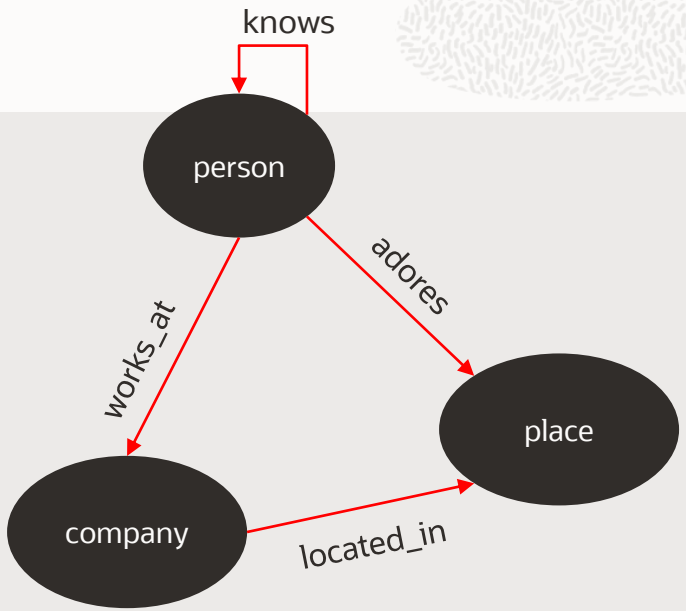


Graph Creation

```
CREATE PROPERTY GRAPH MY_GRAPH
  VERTEX TABLES (
    person as p key(id) label person PROPERTIES(
       p.id,
       p.details.name.string() AS name,
       p.details.address.city.string() AS city,
       p.details.address.zip.number() AS zip,
       p.details.birthdate.date() AS birthdate,
       p.details.creditScore[*].avg() AS avg_credit_score),
    company,
    place
  )
  EDGE TABLES (
    knows SOURCE KEY(E_SRC) REFERENCES p(ID) DESTINATION KEY(E_DST) REFERENCES p(ID),
    person as works_at SOURCE KEY(id) REFERENCES p(id) DESTINATION key(works_at) REFERENCES
company(id),
    adores SOURCE p DESTINATION place,
    company as located_in SOURCE KEY(ID) REFERENCES company(ID) DESTINATION KEY(located_in)
REFERENCES place(id)
 );
```

JSON simplified syntax used to define properties from schema-less JSON column values

Use explicit syntax to disambiguate between source and destination



Graph element tables can be external tables or materialized views

Credit worthiness of Bob's social circle

Bob would like to take a loan from his bank. In order to assess his credit worthiness

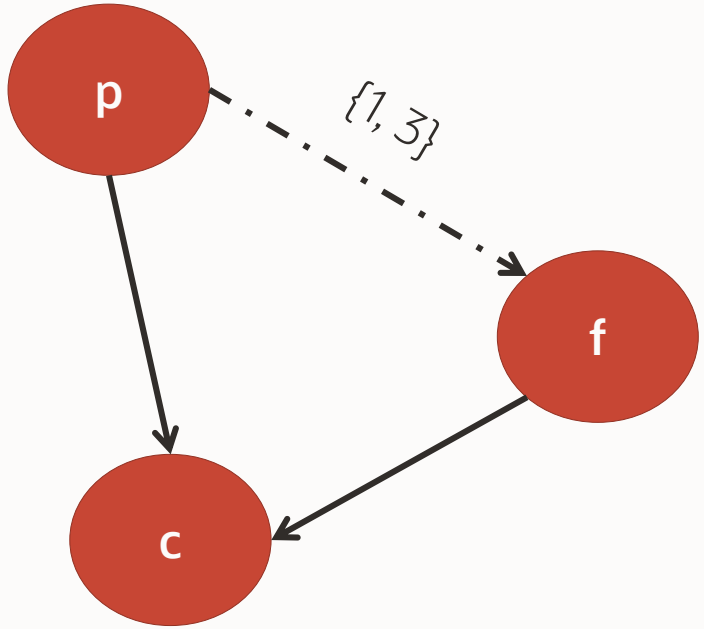
The bank will investigate the credit scores of his friends, friends of friends, etc

```
SELECT DISTINCT id, name, credit_score
FROM GRAPH_TABLE (MY_GRAPH
  MATCH (p)(-[is knows]-){1,3}(f)
  WHERE p.name = 'Bob'
  COLUMNS (f.id as id,
           f.name as name,
           f.avg_credit_score as credit_score)
);
```

Credit worthy work colleagues who can recommend Bob

The bank would also like to find a set of credit worthy work colleagues, in the social circle of Bob, who could recommend Bob.

```
SELECT name, city, avg_credit_score
FROM GRAPH_TABLE (MY_GRAPH
  MATCH (p)(-[is knows]-){1,3}(f),
        (p)-[is works_at]->(c is company),
        (f)-[is works_at]->(c is company)
  WHERE p.name = 'Bob' AND
        f.avg_credit_score > 750 AND
        p.id <> f.id
  COLUMNS (f.name, f.city, f.avg_credit_score)
)
ORDER BY avg_credit_score DESC;
```



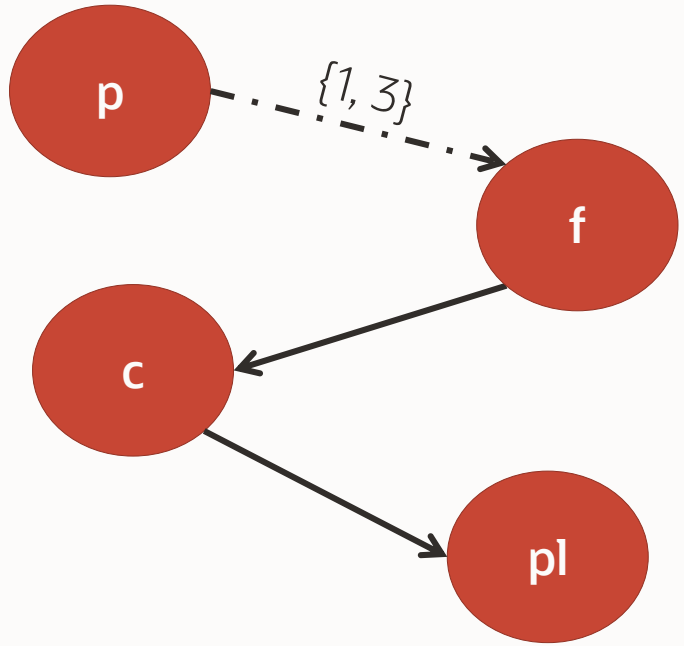
Friends of friends of friends working in New York

Bob needs his loan to buy a new house in New York.

The bank would like to check how many friends of Bob work in New York in order to check the likelihood of his social integration.

SQL with graphs

```
SELECT * FROM GRAPH_TABLE (MY_GRAPH
  MATCH (p)(-[is knows]-){1,3}(f),
        (f)-[is works_at]->(c is company),
        (c)-[is located_in]->(pl is place)
  WHERE p.name = 'Bob' AND
        pl.name = 'New York'
  COLUMNS (f.name, f.zip as address)
);
```



Graphs are a powerful tool to traverse connections in your data !

Graph is now just another SQL object

Security

- Privileges, DataGuard, DataVault, RAS, Redaction, …
- Auditing

SQL interoperability

- Views, Materialized views, Joins, …
- Triggers

Data pump support

- Import / Export

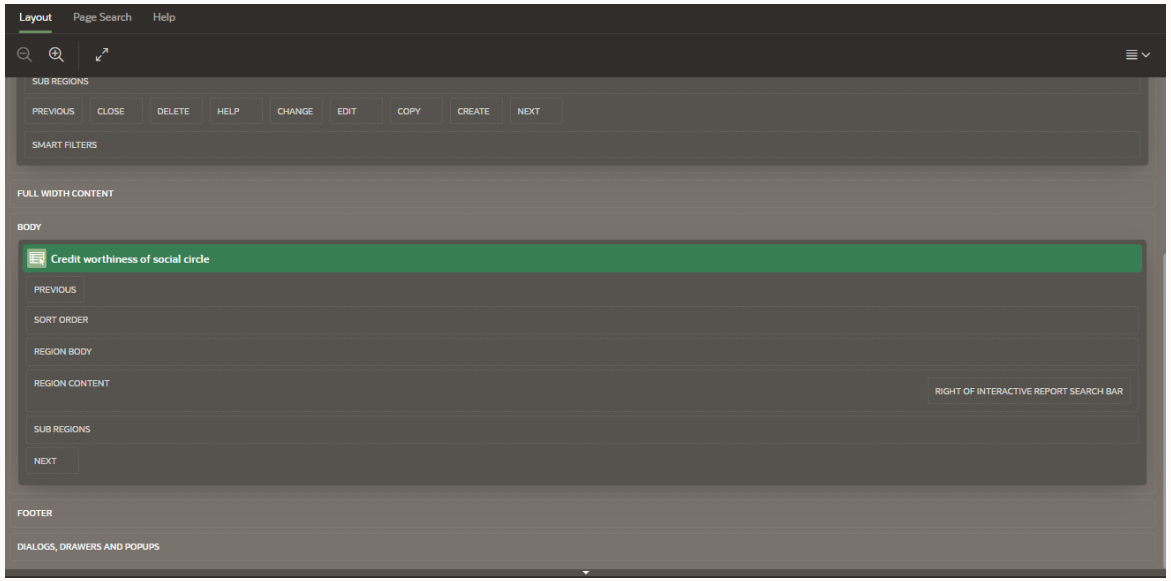
…

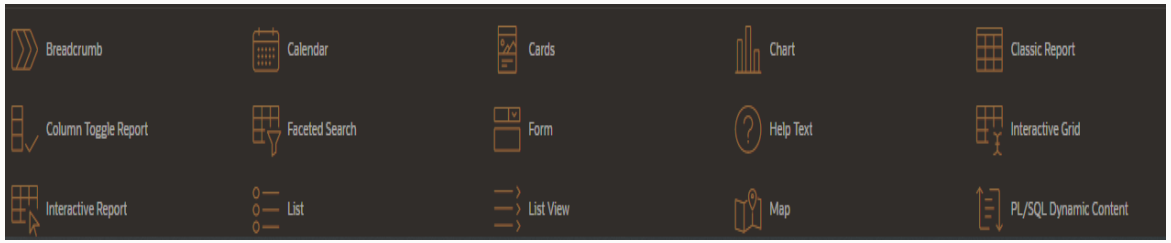
Graph is now just another SQL object

All SQL applications are able to leverage graphs out of the box.

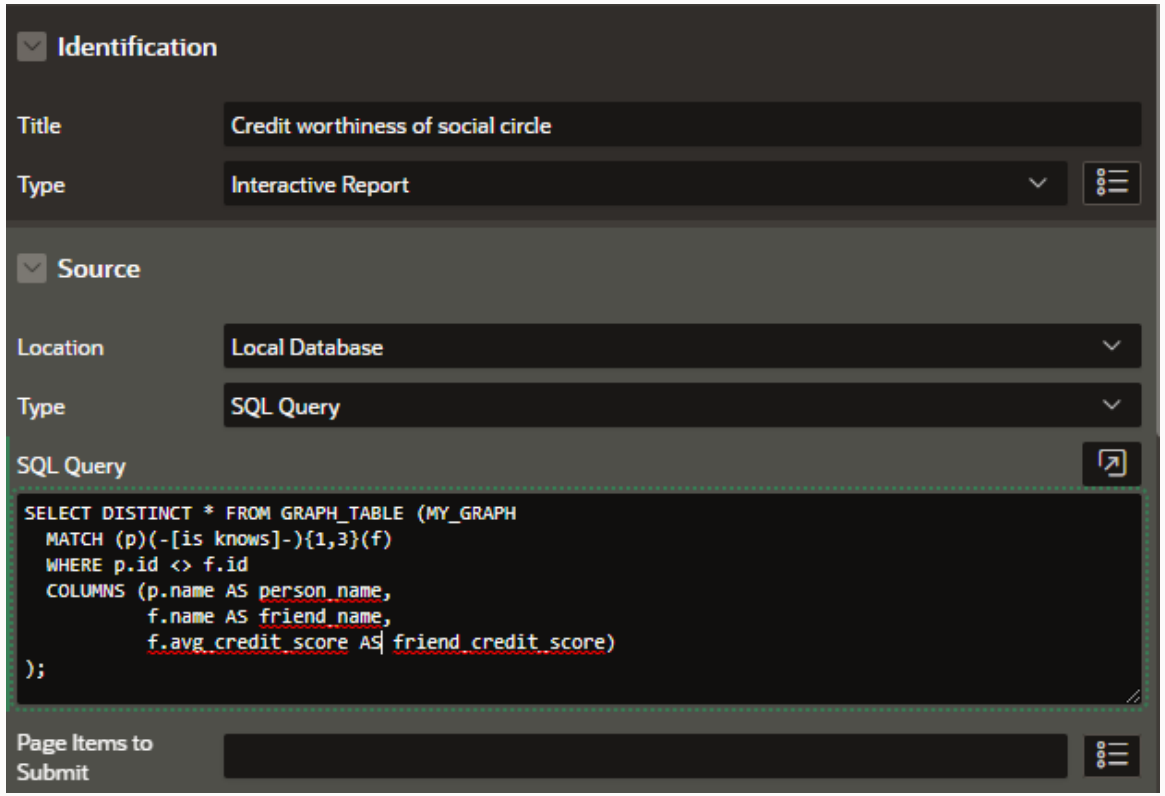
For example we can use APEX to build a graph application

Decide on page layout and widget to include as in any APEX page.

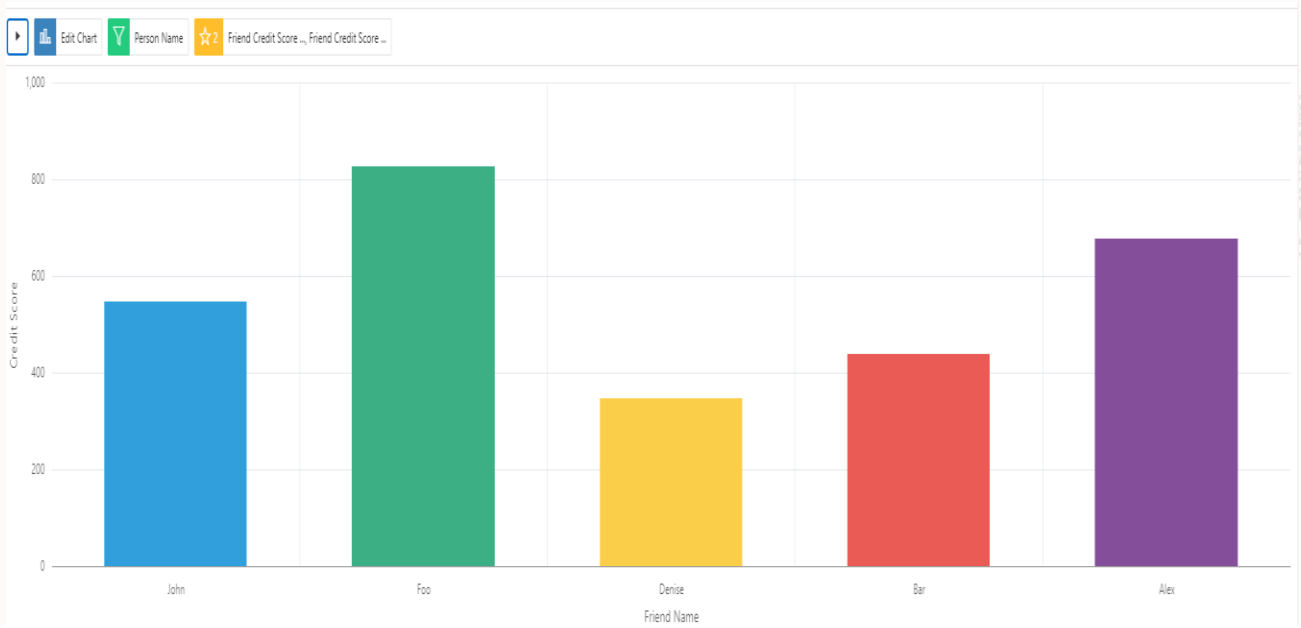




Pick graph query as data source for the widget



APEX interactive visualization of graph query results



Credit worthiness of social circle

Go

Actions

Person Name = 'Bob'

Friend Credit Score >= 750

Friend Credit Score <= 500

Person Name	Friend Name	Friend Credit Score
Bob	John	550
Bob	Foo	830
Bob	Denise	350
Bob	Bar	440
Bob	Alex	680

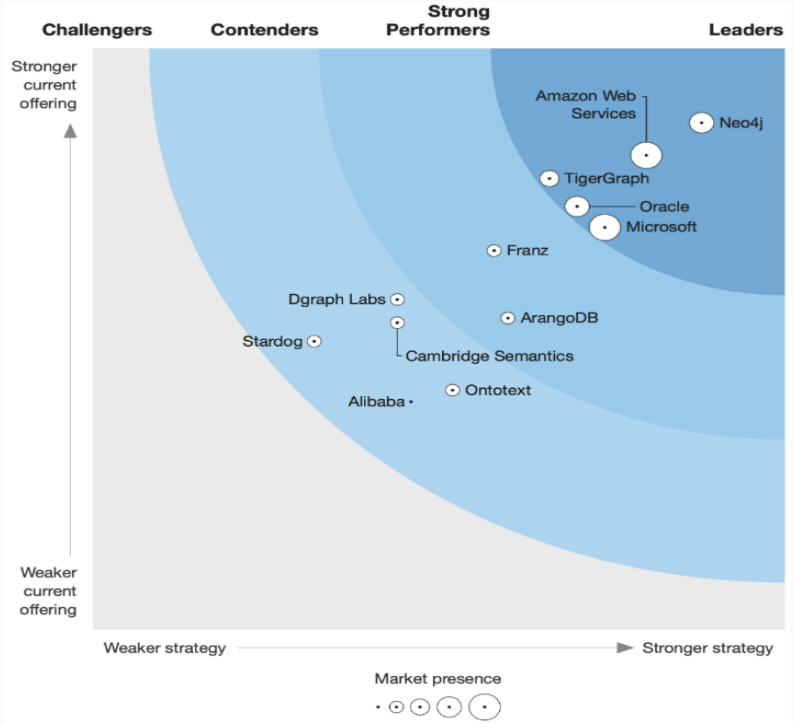
1 - 5

Oracle named as a leader in Graph Data Platforms

The Forrester Wave™: Graph Data Platforms, Q4 2020

The 12 Providers That Matter Most And How They Stack Up

by Noel Yuhann
November 16, 2020



Key Takeaways

Neo4j, Amazon Web Services, TigerGraph Microsoft, And Oracle Lead The Pack

Conclusion of the evaluation

- "Oracle's graph offering is a viable option, especially for oracle customers."
- Oracle supports both RDF and property graph models.
- Benefits are Oracle's capabilities for technical support, cloud offering, PGQL, ease to start with SQL-like syntax, and performance for moderately sized deployments.

To add on top

- Oracle is now the first company to implement standardized SQL extension for property graphs. (SQL/PGQ)

Summary

- Graphs can be created and queried in SQL.
- Graphs simplifies queries for identifying connections or dependencies.
- Since graphs are part of the SQL engine all existing tools and programmatic interfaces will work with graphs.

SQL Property Graphs will be available starting Oracle 23c on all DB cloud services and on premise.

Try it out !

Thank you!

More details: oracle.com/database/graph

For any question feel free to contact us!