ORACLE

グラフ 関連新機能

Oracle Database 23c新機能セミナー

中井 亮矢

日本オラクル株式会社

2023年10月20日

Agenda

Oracle Graph

- グラフデータベースとは
- Oracle Property Graph 概要
- Oracle Property Graph 新機能
- Oracle RDF Graph 新機能

中井

Oracle Machine Learning & Oracle Spatial Updates

- Oracle Machine Learning 新機能
- Oracle Spatial 新機能

出口

このセッションの各機能について

Oracle Graph

Oracle Property Graph

- オラクルデータベースと統合された**グラフデータベース**機能
- パターンマッチング、グラフアルゴリズム、アルゴリズム開発APIなどを持

Oracle RDF Graph

- オラクルデータベースと統合されたW3C勧告の**RDFストア**
- 推論機構やデータベース内データのRDF化等の機能を持つ

W3C Semantic Web

Oracle Machine Learning

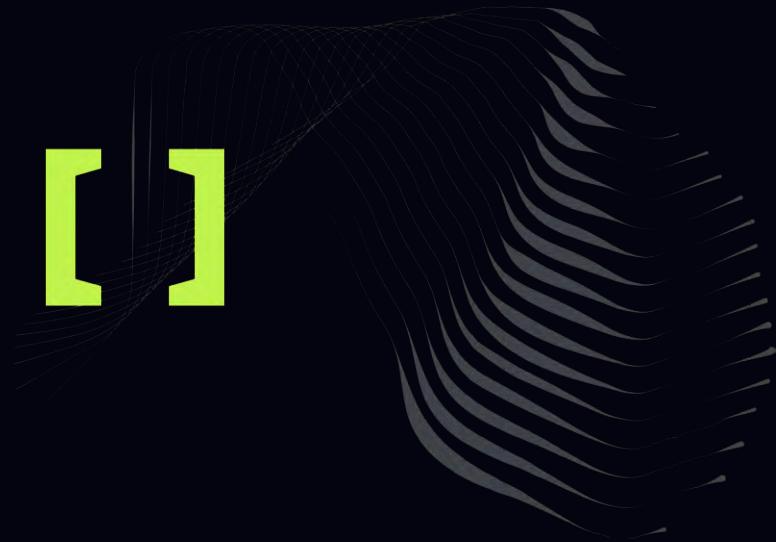
- **オラクルデータベース内**に実装された**機械学習機能**
- データベース内で機械学習プロセスを完結できる

Oracle Spatial

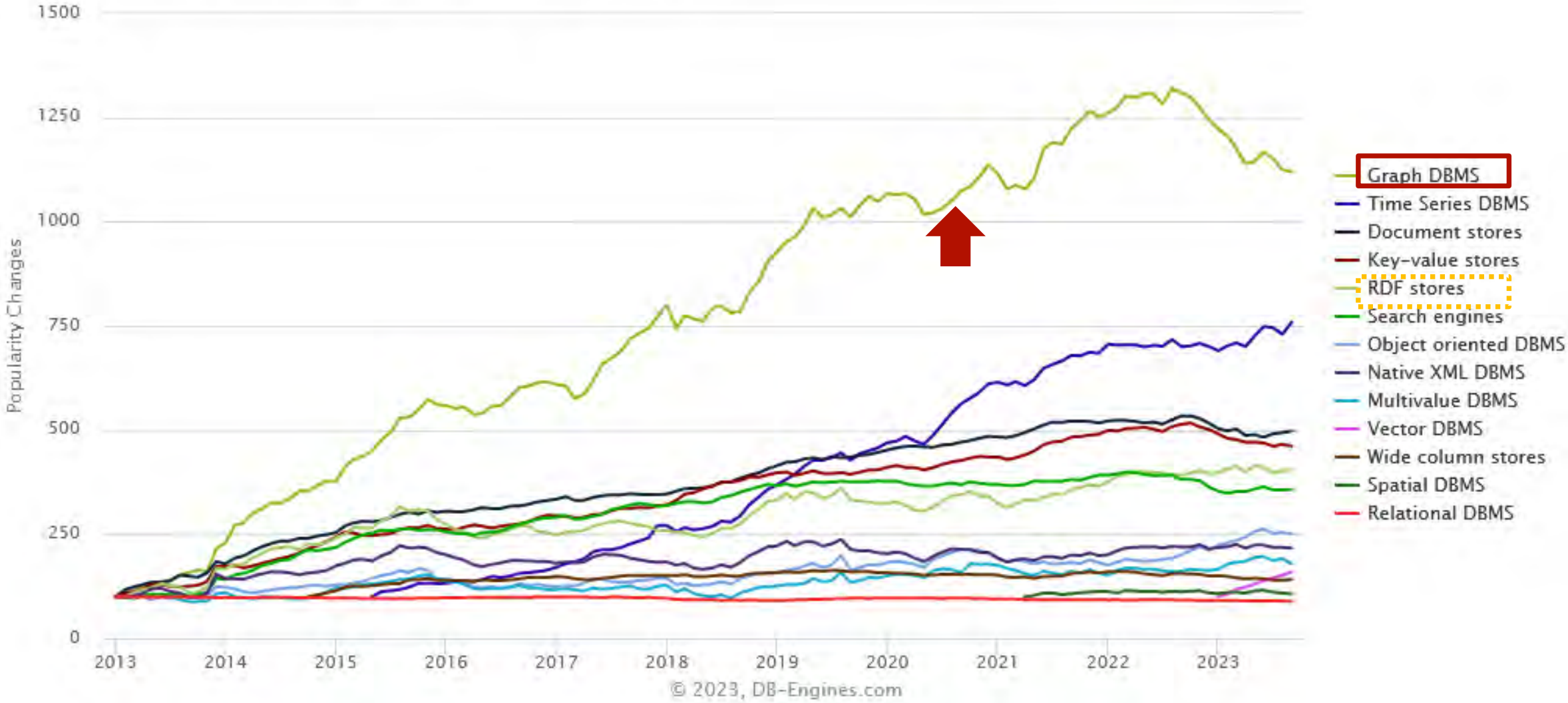
- オラクルデータベース内の**地理空間機能**
- 地図データ、位置情報、衛星画像、３D点群等の空間データを統合管理

2019年12月5日より上記機能は全て無償化（オラクルデータベースのライセンスに付随）

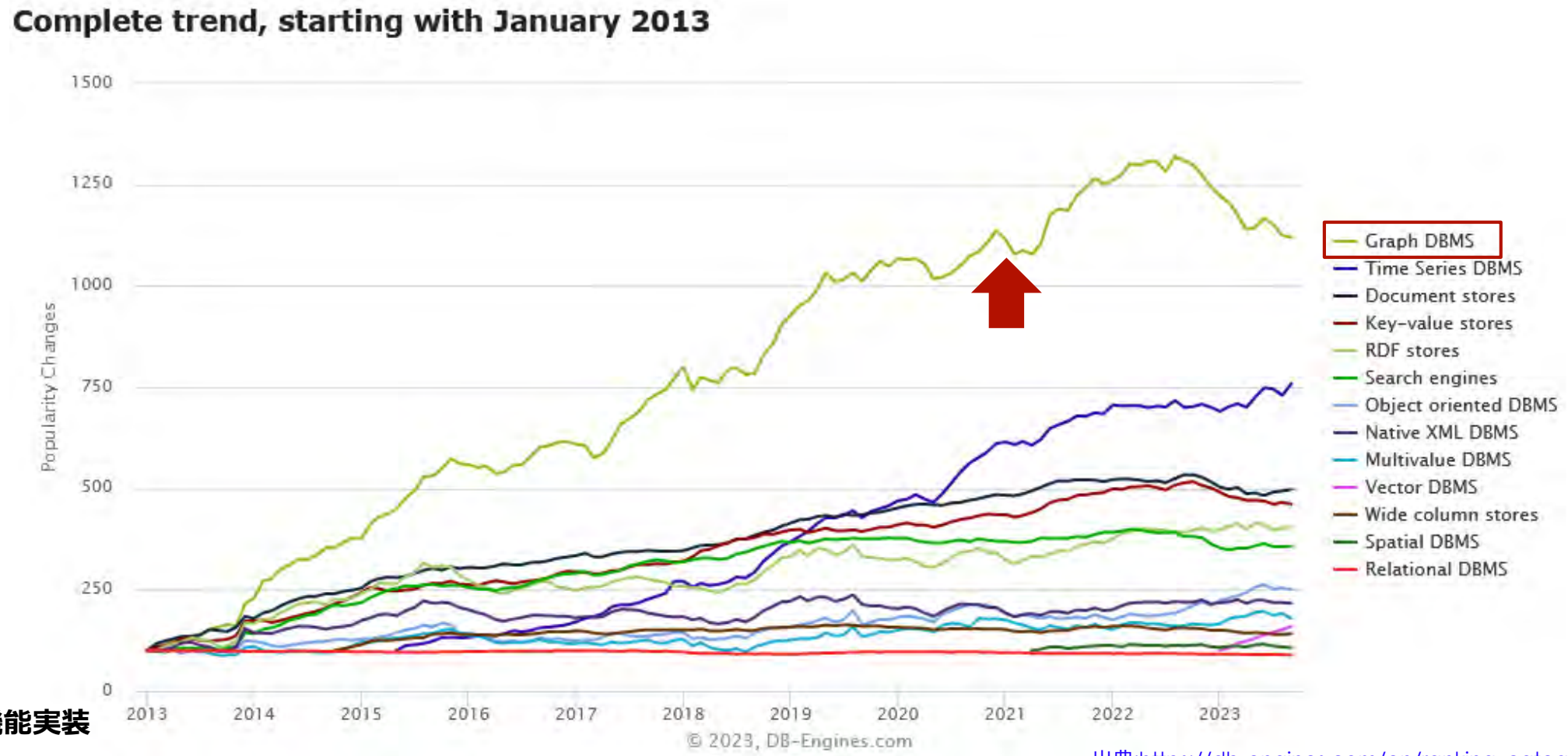
グラフデータベースとは



Complete trend, starting with January 2013



出典:https://db-engines.com/en/ranking_categories



出典:https://db-engines.com/en/ranking_categories

オラクルのグラフ機能実装

2004 ORACLE DATABASE 10g
ネットワーク
データモデル

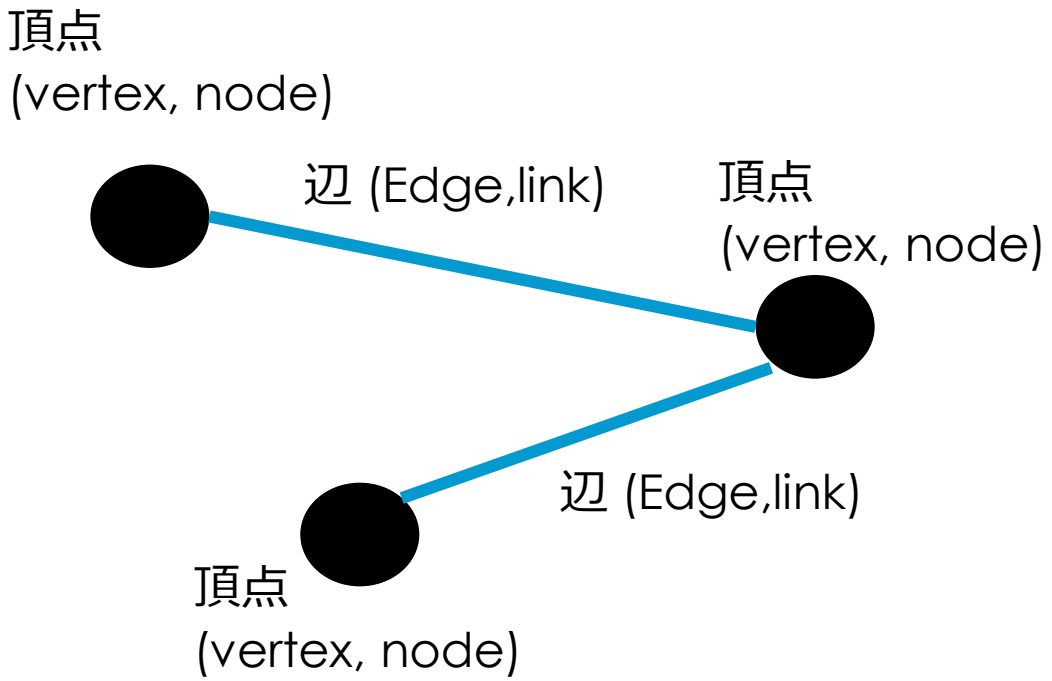
2007 ORACLE DATABASE 11g
RDFグラフ

2016 ORACLE DATABASE 12c
プロパティグラフ(12cR2～)

グラフデータベースとは

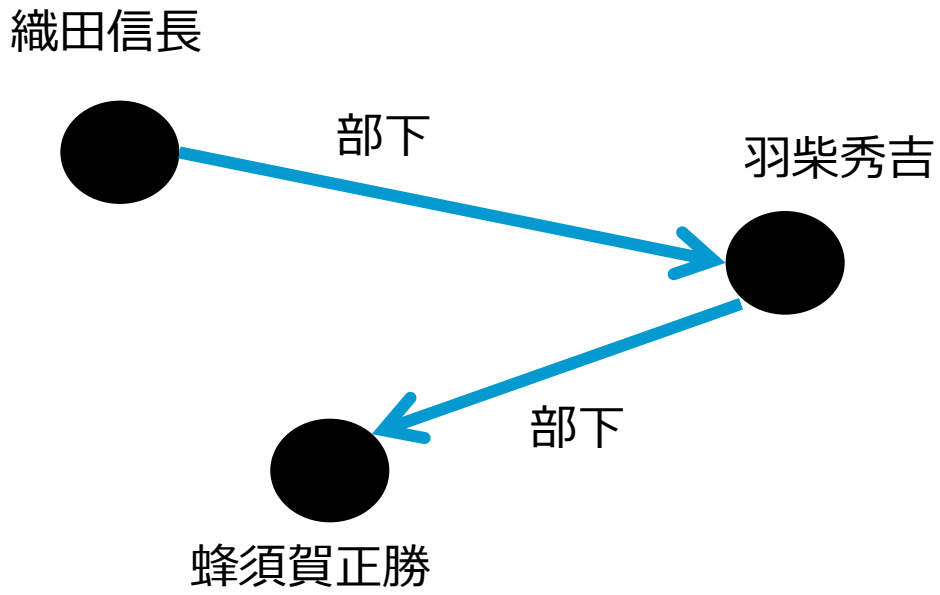
- グラフ形式のデータモデルを扱うデータベース

- グラフ形式
  - 頂点と辺(エッジ)で構成される
  - 頂点と頂点は辺で接続される



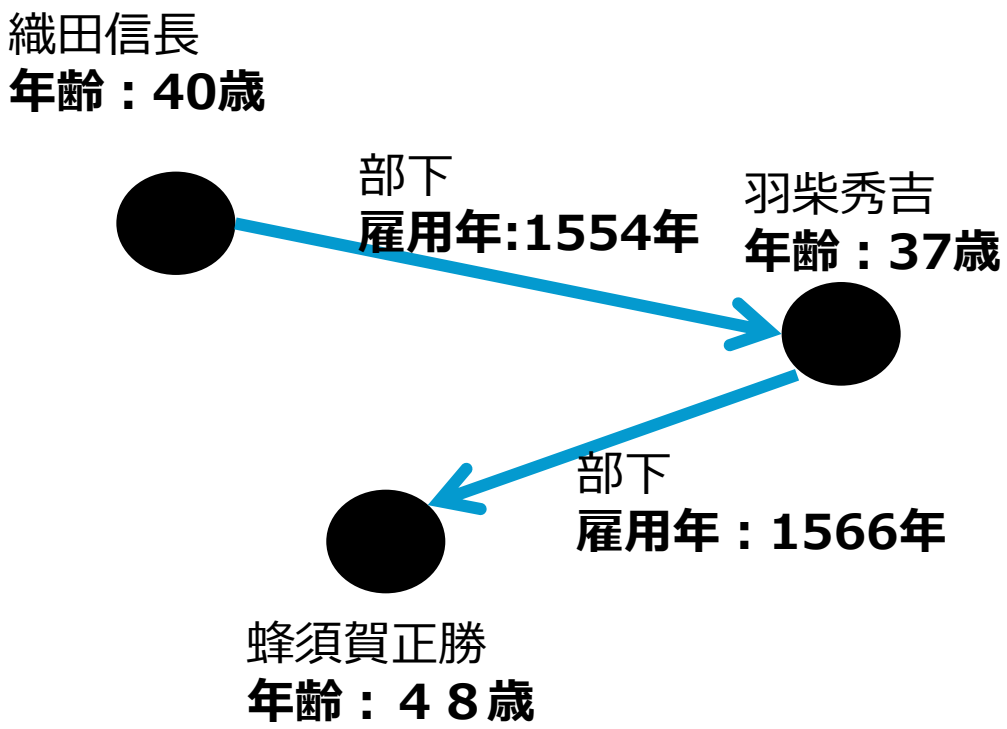
グラフデータベースとは

- グラフモデルの実装
  - 辺は**向き**を持ってもよい



グラフデータベースとは

- グラフモデルの実装
  - 辺は向きを持ってもよい
  - 頂点、辺は**属性**を持ってもよい

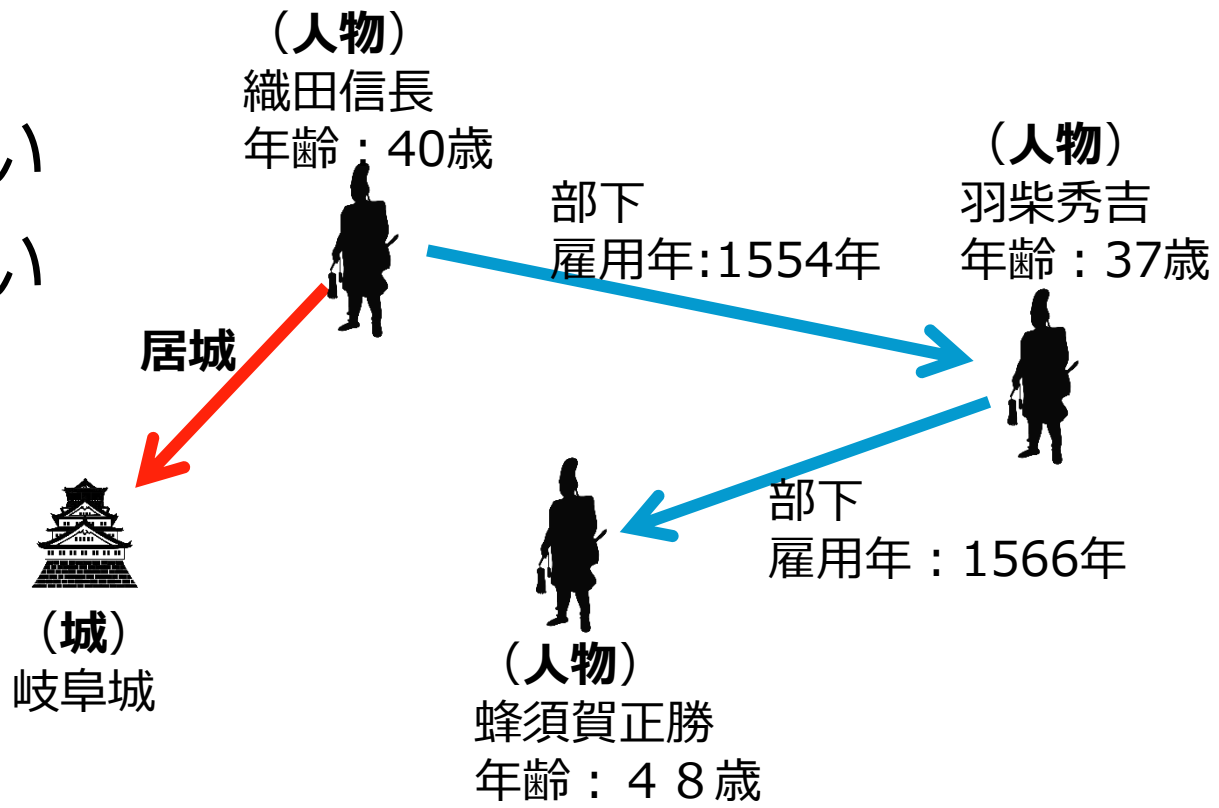


属性(プロパティ)を持つため
プロパティグラフモデル
とも呼ばれる

グラフデータベースとは

- グラフモデルの実装
  - 辺は向きを持ってもよい
  - 頂点、辺は属性を持ってもよい
  - 頂点、辺は**種類**を持ってもよい

種類のラベルを持つため
ラベル付きプロパティグラフモデル
とも呼ばれる

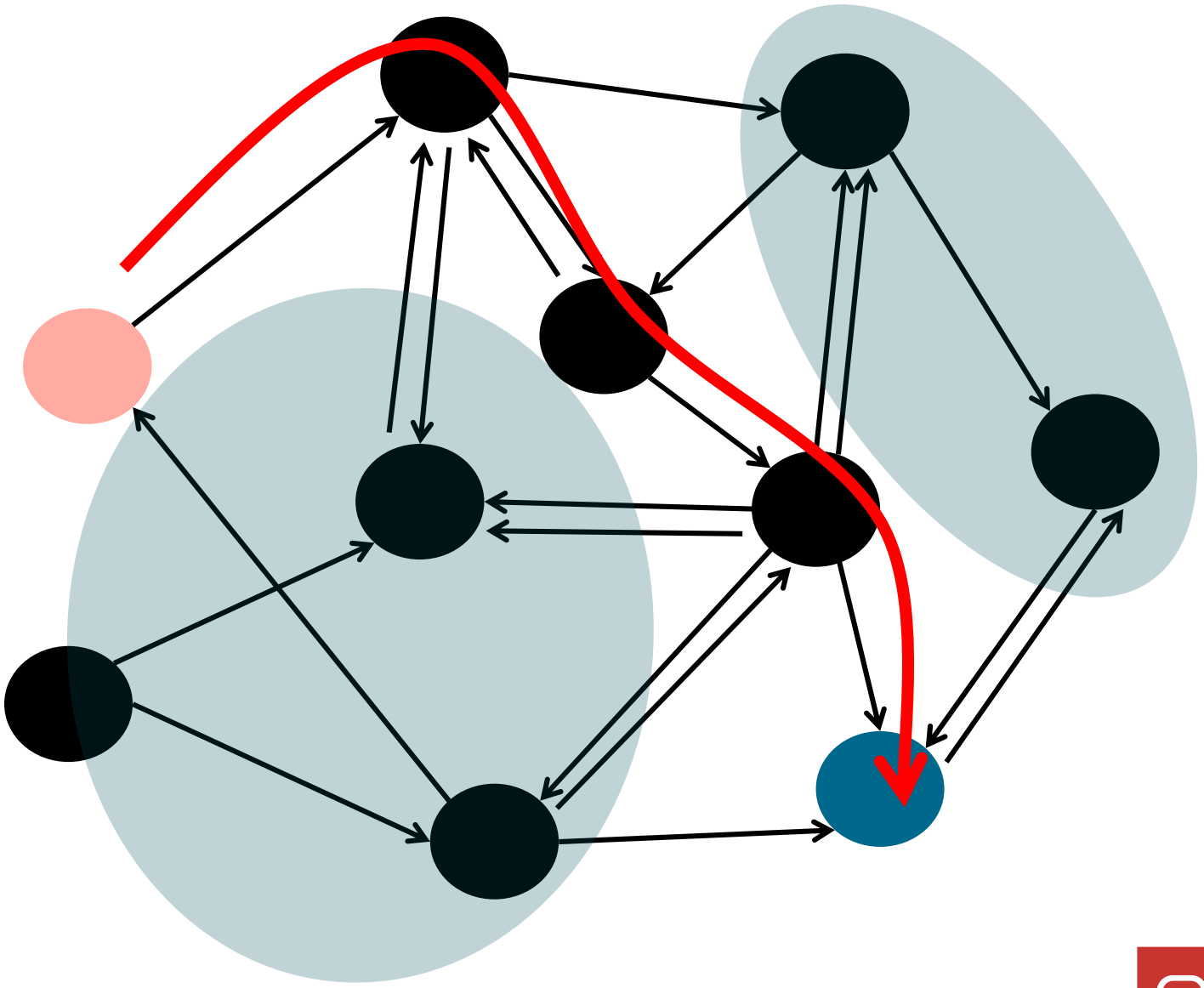


グラフデータベースの特徴

✓ 辿る検索が高速

- 各頂点は接続先の辺の情報を持つ。
- 探索時に経路に関係するデータのみのアクセスで辿ることができるため、高速に結果を取得できる

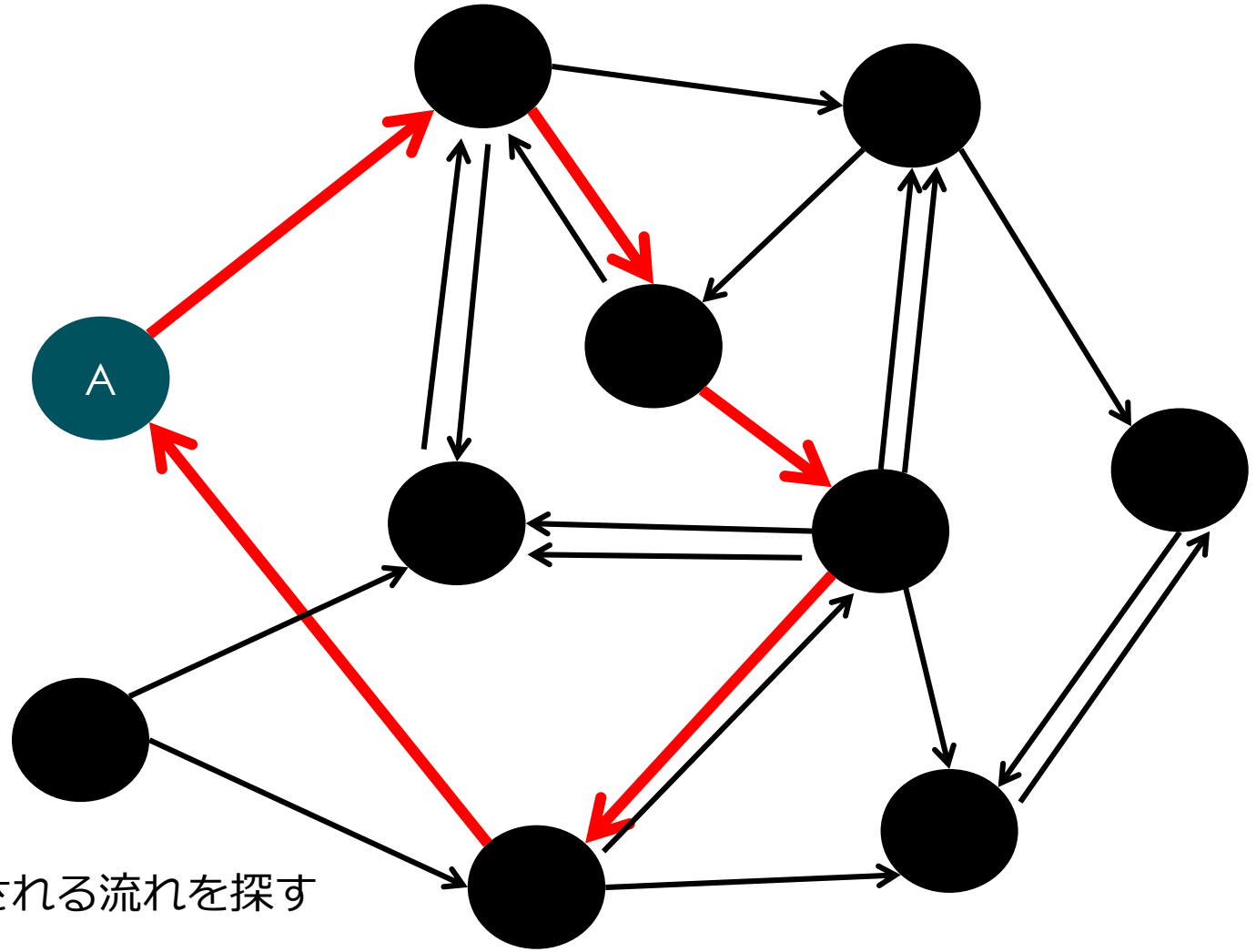
※この手の処理をリレーショナルモデルで実装するとSQLで記述すると複数回のjoinが発生し、非常に重くなることが多い



グラフデータベースの特徴

✓ パターンマッチング

- 多くのグラフデータベースで、辿る検索の高速性を生かし、パターン照合を行うクエリが実装されている

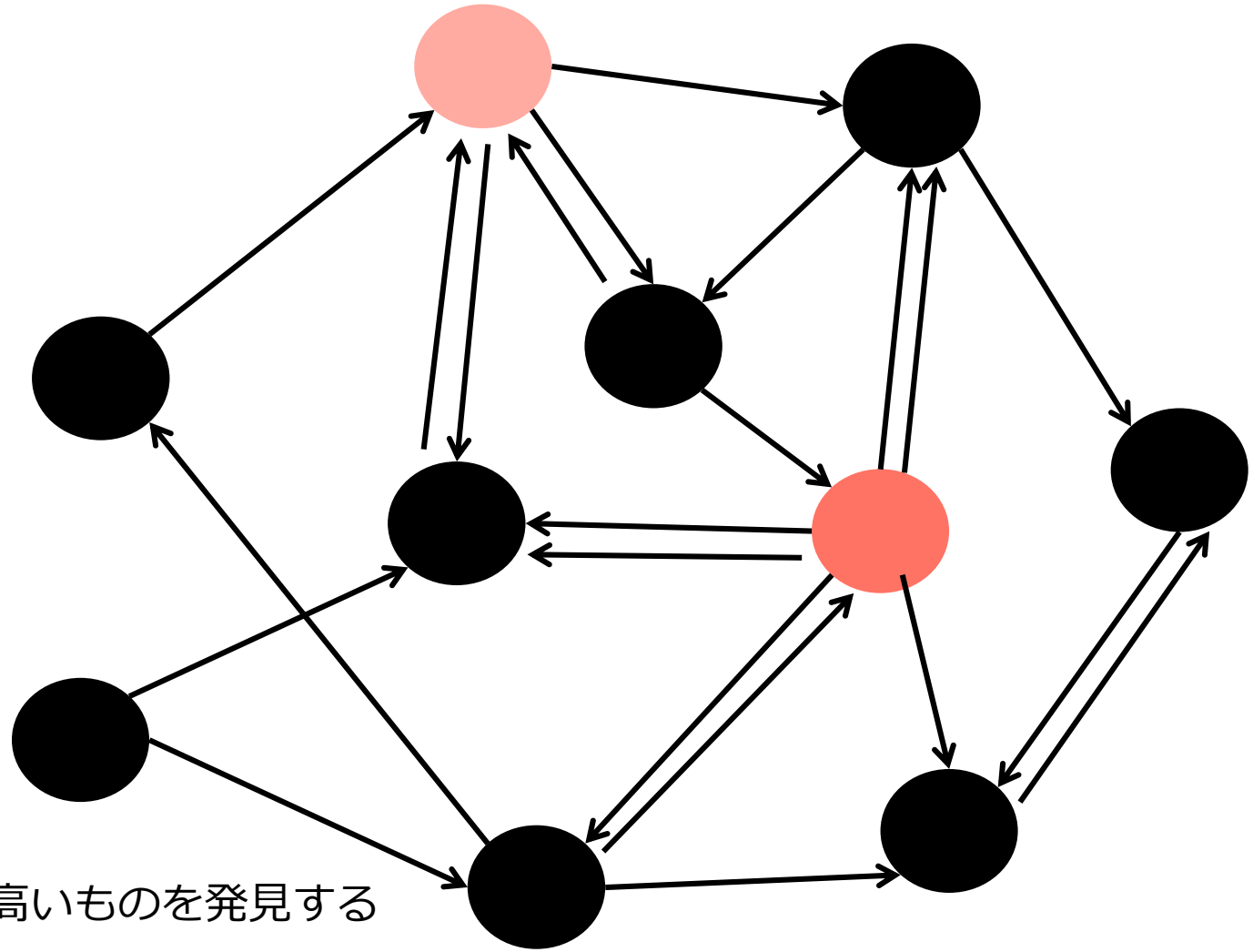


例：送金ネットワーク内で
５回の送金で自分の口座に送金される流れを探す

グラフデータベースの特徴

✓ グラフアルゴリズム

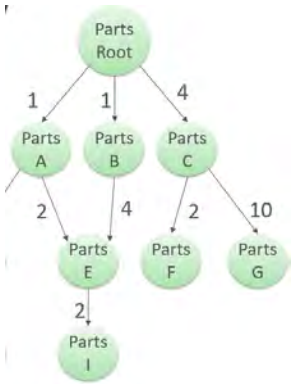
- グラフの構造やその類似性を定量的に指標化する様々なグラフアルゴリズムが存在しています
- Graph Embedding/Graph Neural Network (グラフ内の関係構造を埋め込み、ベクトル表現を得る等）も非常に注目されている手法の１つ



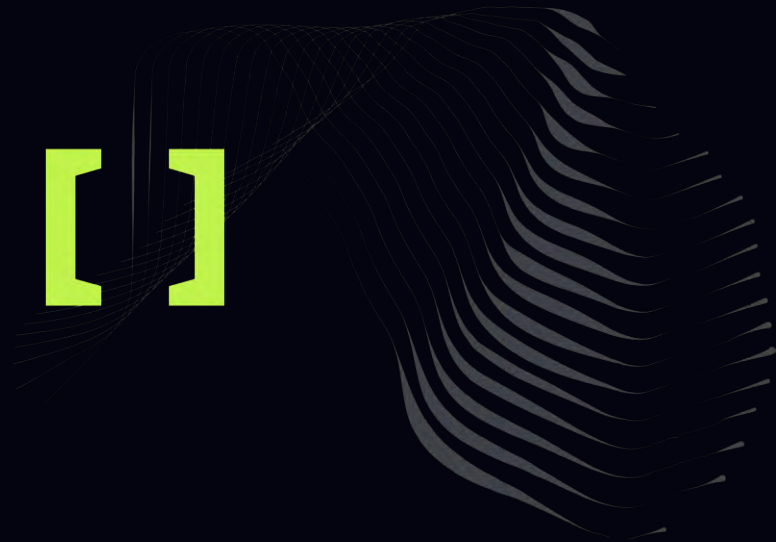
例：送金ネットワーク内で中心性の高いものを発見する

グラフデータベースのユースケース例

- マッチング/リコメンデーション
  - 商品、顧客、購買データから**パターン抽出や類似性評価**を行う
- Customer360
  - 複数システムに点在する顧客に関連したデータを**グラフでつなぐ**
- 不正検知
  - 送金、ネットワークトラフィックのような構造のデータから**パターンを抽出**
- BOM管理
  - トレーサビリティ、設計/製造BOM/調達情報を**グラフでつなぐ**



Oracle Property Graph 概要



Oracle Property Graph

Oracle Labs PGX

- Oracle Labs(旧Sun Microsystems Laboratories)で研究開発された並列グラフ探索エンジン (**P**arallel **G**raph Analyti**X**)

PGXの特徴

- 高速なグラフ探索
- 高速なパターンマッチングクエリ
- 高速なアルゴリズム実行
- アルゴリズム開発用API

＋

Oracle Database技術

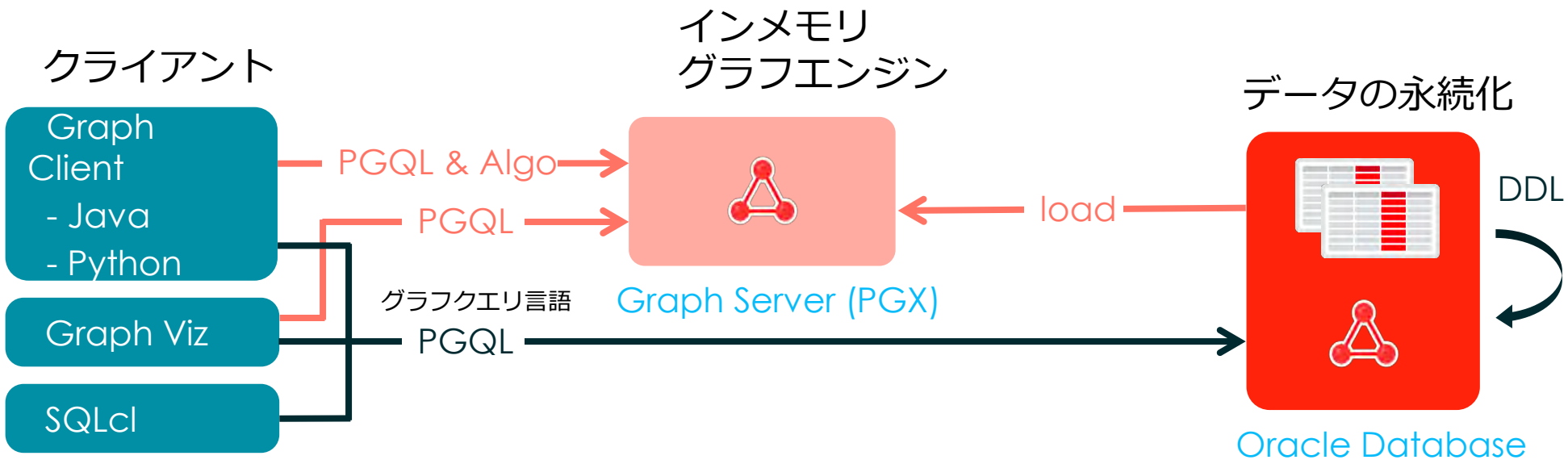
- オラクルデータベースに宿るオラクルのデータマネジメント技術

一般のグラフデータベースの苦手科目

- ACID・トランザクション機能
- 参照、更新の混合ワークロード
- セキュリティ機能
- 運用管理機能　等

アーキテクチャ

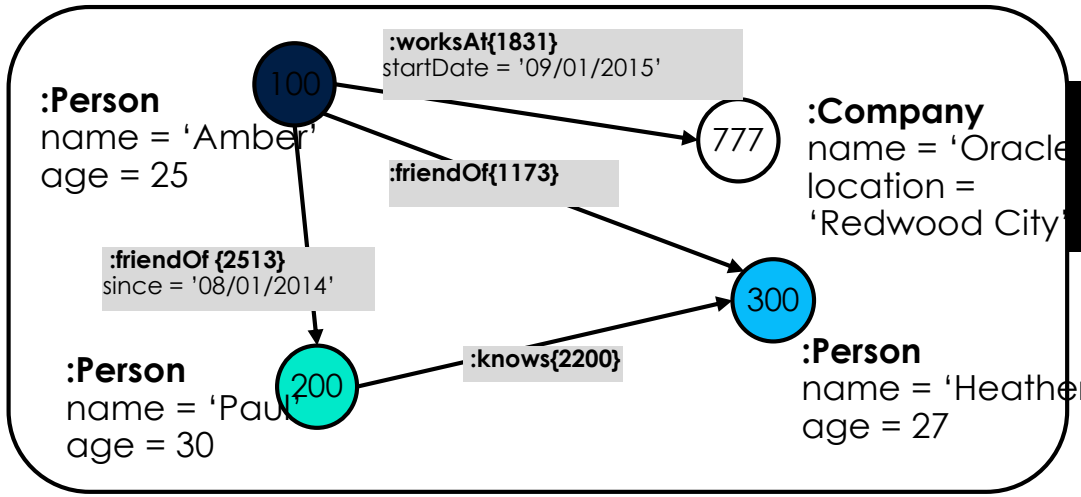
- 高速グラフ処理はインメモリのグラフエンジン(PGX)で
- データの永続化はデータベースで



PGQL: グラフパターンマッチング

対象のグラフから指定のパターンに適合するすべてのデータを検索

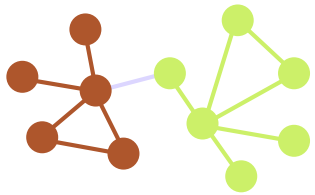
```
SELECT v3.name, v3.age
  FROM socialNetworkGraph
 MATCH (v1:Person) -[:friendOf]-> (v2:Person) -[:knows]-> (v3:Person)
 WHERE v1.name = 'Amber'
```



socialNetwork
Graph

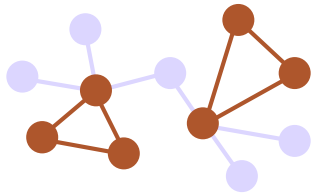
Amberさん
の友達
が知っているすべての人
の名前と年齢

Built-in Graph Algorithms



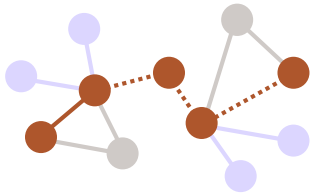
Detecting communities

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



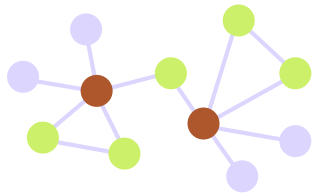
Topology analysis

Conductance, Cycle Detection, Degree Distribution, Eccentricity, K-Core, LCC, Modularity, Reachability Topological Ordering, Triangle Counting, Bipartite Check, Partition conductance



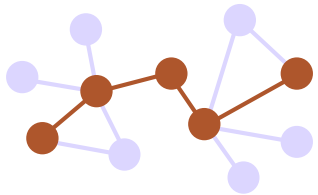
Link prediction and others

Twitter Whom-to-follow, SALSA, Adamic-Adar Index



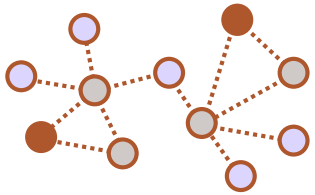
Ranking and walking

PageRank, Personalized PageRank, Degree Centrality, Closeness Centrality, Vertex Betweenness Centrality, Eigenvector Centrality, HITS, Minimum Spanning-Tree (Prim's), Breadth-First Search, Depth-First Search, Random Walk with Restart



Path-finding

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



Machine learning

DeepWalk, Supervised GraphWise, Unsupervised GraphWise, Pg2Vec, Matrix Factorization, GNNExplainer

PGX Algorithm API

グラフアルゴリズム開発用のAPI

- JavaベースのグラフアルゴリズムDSLセットを含むAPI
- グラフにありがちな処理(幅優先探索,フィルタ処理等)の簡潔な記法を提供
- Javaコード内で @GraphAlgorithm アノテーションを利用

メリット：

- アルゴリズムを簡潔に記述
  グラフ系アルゴリズムの記述でコードを複雑化させがちな探索のための多重ループ処理やフィルタを**簡潔に記述**できる
- 高速な実装
  グラフアルゴリズムに特化して解釈されるため、記述したアルゴリズムは**グラフ上での実行に最適化**された形でバイトコード化されるため非常に高速に動作する

コードイメージ (指定した頂点からの最短経路の Hop数を導出する)

```
@GraphAlgorithm
public class HopDistance {
 public void hop_dist(PgxGraph g, PgxVertex root, @Out VertexProperty<Double> dist,
   @Out VertexProperty<PgxVertex> prev, @Out VertexProperty<PgxEdge> prev_edge) {
  if (g.getNumVertices() == 0) {
   return;
  }

  dist.setAll(Double.POSITIVE_INFINITY);
  prev.setAll(PgxVertex.NONE);
  prev_edge.setAll(PgxEdge.NONE);
  dist.set(root, 0d);

  inBFS(g, root).forward(n ->{
   dist.set(n, (double) currentLevel());
   prev.set(n, n.parentVertex());
   prev_edge.set(n, n.parentEdge());
  });
 }
}
```

プロパティの初期化

Rootからの幅優先探索(BFS)、ループ的に動作し、 nに、幅優先で探索した順にノードがセットされる

currentLevel()は現在の幅優先探索の階層レベル

※この例の処理内では、最短経路のホップ数(到達時点の幅優先の深さ) をdist プロパティにセットし、ついでにその際の親ノードと親エッジをプロパティにセットしています。

リレーショナル表からのグラフ定義

create property graph文

リレーショナル表からのグラフデータ生成

CREATE PROPERTY GRAPH　文によって、リレーショナル表をベースとしたグラフを宣言的に生成することができます。

```
CREATE PROPERTY GRAPH bank_transfers
 VERTEX TABLES ( persons KEY(account_number))
 EDGE TABLES
 ( transactions KEY
  (from_acct, to_acct, date, amount)
   SOURCE KEY (from_account) REFERENCES persons
   DESTINATION KEY (to_account) REFERENCES persons
   PROPERTIES (date, amount) )
```

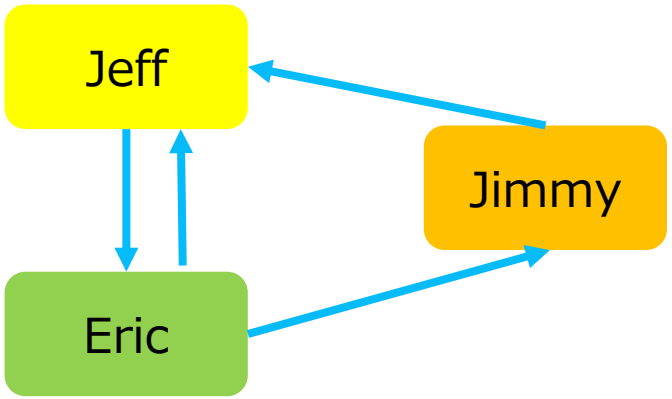
※Autonomous databaseでは、Graph Studio というツールの中で
GUI でリレーショナル表を選択するだけで外部キー制約などから
半自動的にcreate property graph文を生成するツールをご用意しております

Persons表

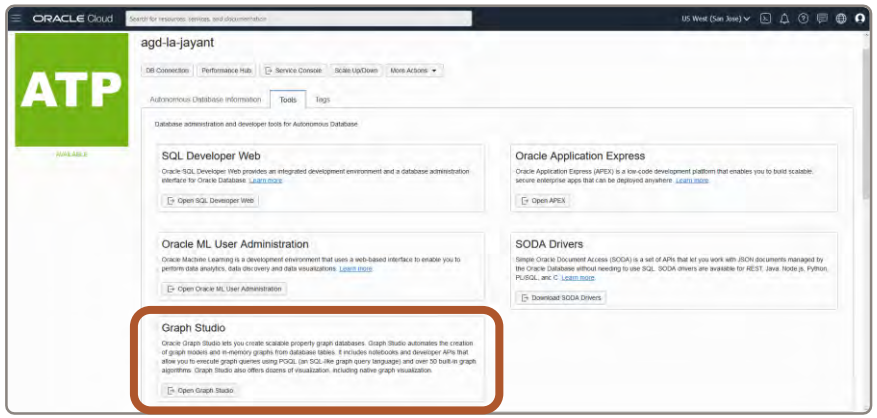
account_number	name	phone	...
333394584	Jeff	..	
327324852	Eric		
879743653	Jimmy		

transactions表

from_account	to_account	date	amount
333394584	327324852	2023/3/1	32000
327324852	879743653	2023/3/15	7500
879743653	333394584	2023/3/22	21400
327324852	333394584	2023/3/24	11000



Graph Studio in Autonomous Database

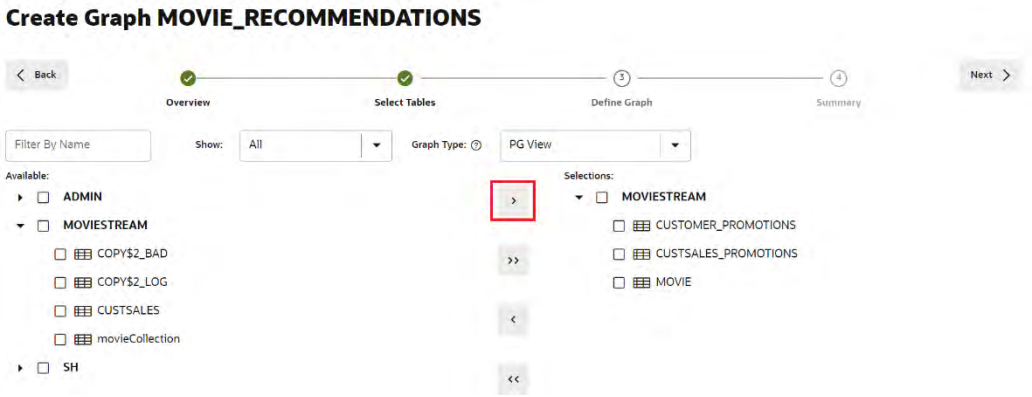


Autonomous Databaseでセットアップ作業不要で
使える、グラフデータベースとその分析環境

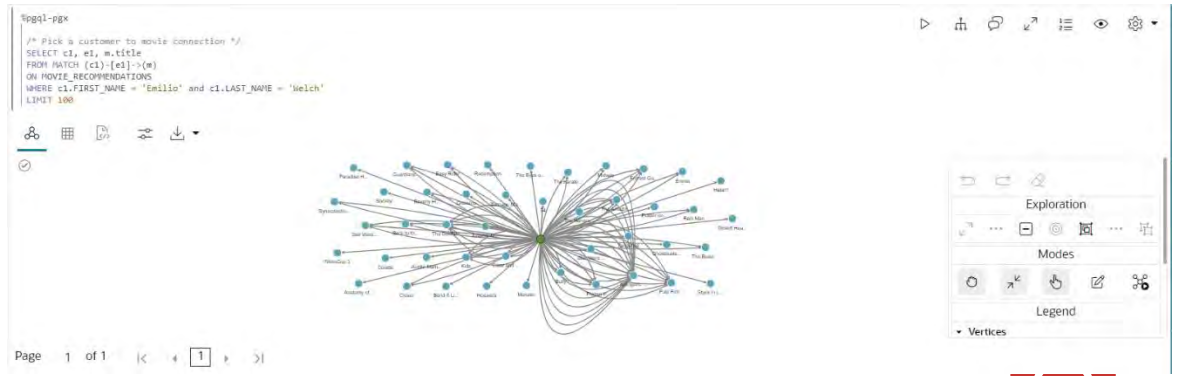
グラフ分析のための包括的なツール

- **グラフモデリングツール**
  - リレーショナル表のデータをグラフにマッピング
- **ノートブック機能**
  - ブラウザ上から利用可能
  - グラフデータに対するクエリ(PGQL)やクエリ結果の描画機能、アルゴリズムの実行などグラフに対する操作を一元化できるノートブック
- **管理性**
  - グラフモデル管理
  - 高速なインメモリグラフエンジン(PGX)へのロードもGUI設定だけで可能

グラフモデリングツール

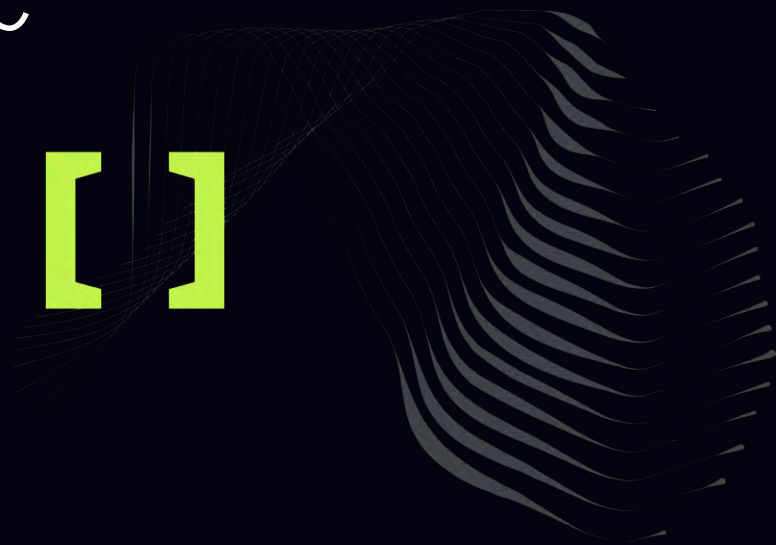


ノートブック



Oracle Property Graph 新機能

～ *Operational Property Graph* ～



Property Graph 新機能/ Database 23c

- ✓ Oracle Property Graph は 現在 Oracle Graph Server & Client というデータベースとは**別のコンポーネントとしてバイナリ管理されています**
  - ✓ 最新のProperty Graphでも Database 19cもサポート
  - ✓ バイナリは分かれていますが、ライセンスはデータベースに許諾されており、データベースがあれば追加のライセンス費用なしで利用可能
- ✓ Database 23c のみで有効な新機能もあるため、以下を分けて記載する
  - ✓「Database 23c と併用による新機能」
  - ✓「Property Graph 23.3 としての新機能」

Database 23c と併用による新機能

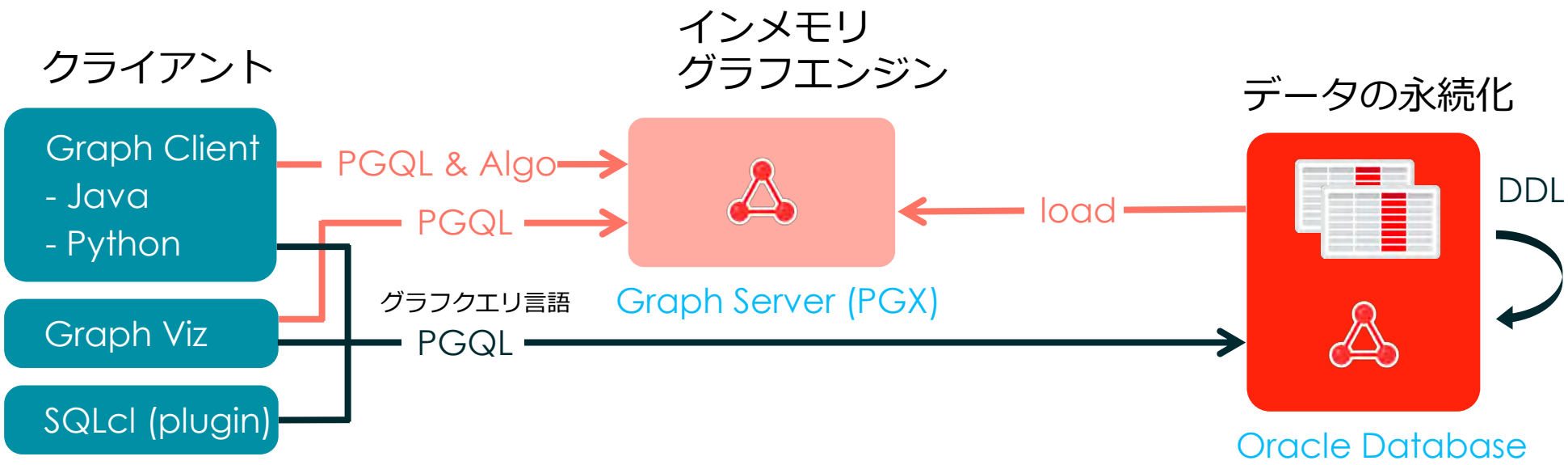
Property Graph 23.3 新機能

ISO標準 SQL:2023, SQL/PGQ サポート

- ✓ ISO SQL:2016の後継として策定されたSQL ISO標準
  - ✓ SQL:2023の中には他にもjsonサポートなどSQL標準への拡張が色々入っている
- ✓ 2023では新しい項目としてSQLのグラフサポート(SQL/PGQ)が追加された
  - ✓ SQLでのグラフの生成、削除、クエリ記述方法などが記載されている

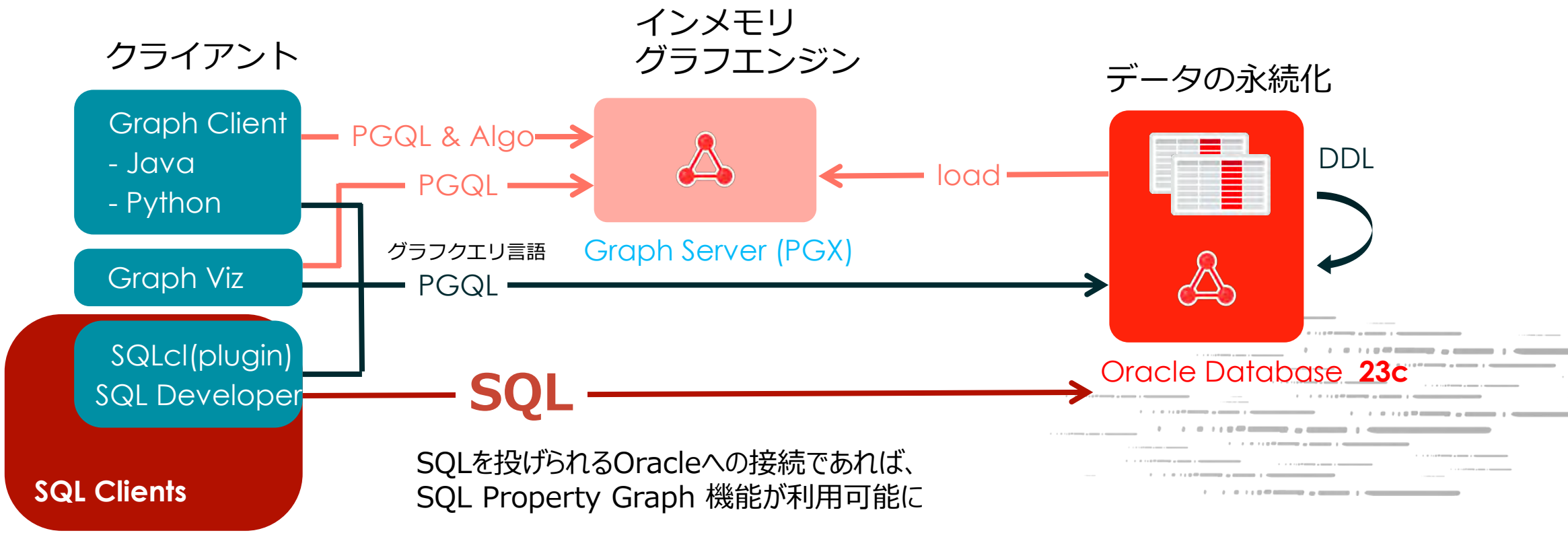
これまでのアーキテクチャ

データの永続層をデータベースが受け持ち、グラフは２層構造でPGQLを受け付ける



Database 23c でのアーキテクチャ

データの永続層をデータベースが受け持ち、グラフは 2 層構造でPGQLを受け付け、**データベースはSQLでグラフクエリを受け付ける**



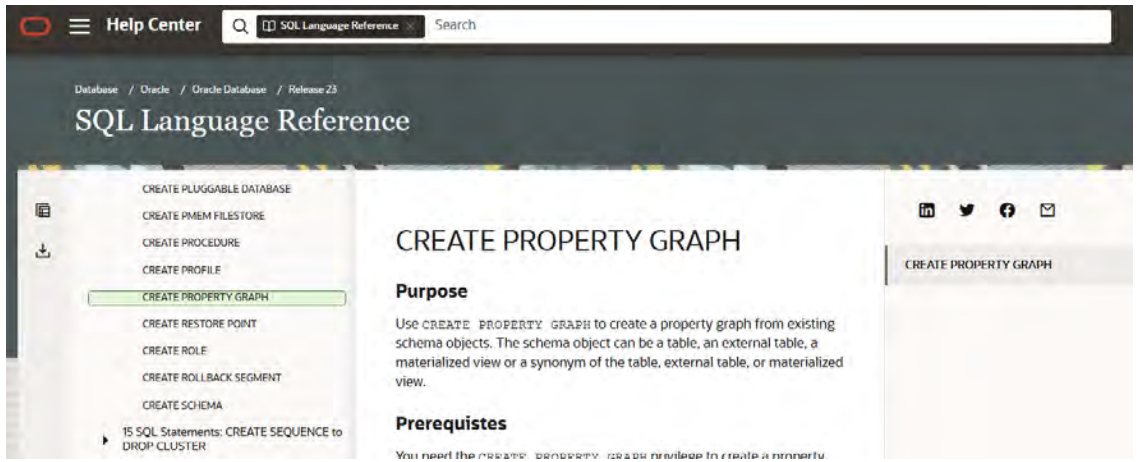
SQLでグラフの生成、削除等が可能に

- **グラフ生成のDDL文がSQLとして実行可能に**
- SQL/PGQ標準
  - CREATE PROPERTY GRAPH
  - DROP PROPERTY GRAPH
  - ALTER PROPRETY GRAPH

生成されたグラフを ***SQL Property Graph***と呼称

- **インメモリ(PGX)連携の拡充**
  DB生成されたグラフ(SQL Property Graph)のインメモリサーバ(PGX)への読み込み方式の拡充
  - グラフ名の指定、サブクエリ発行によるグラフロード等

以前は同様のステートメントをグラフクエリ言語 PGQLとしてグラフクライアントから実行する必要があった

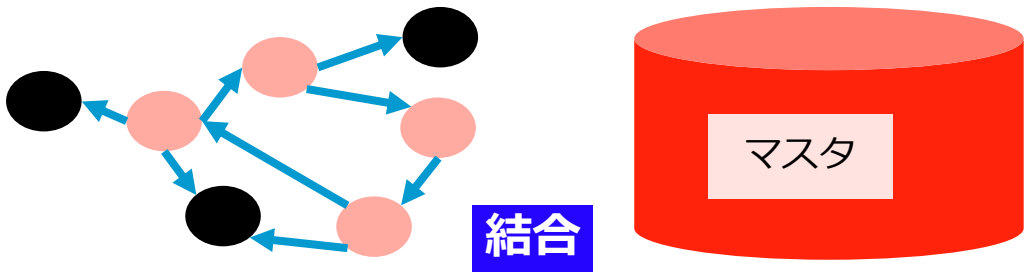


23c SQLリファレンス より

SQLでグラフのクエリ記述が可能に

- **GRAPH_TABLE演算子**によりSQLでグラフクエリを記述可能に
- **SQL/PGQ標準**

- **SQLとしてグラフクエリを利用可能に**
  - JDBC,OCI等のデータベース接続で、**グラフクエリ言語の表現力を利用したクエリが実行**できる
- **グラフデータ活用の加速**
  - SQL 1 本で、グラフクエリの結果をリレーショナル表やJSONデータなどのデータベース内データと結合するなど、**グラフデータの横断的なデータ活用が更に容易に**なる
- **適材適所でクエリを使い分け、マージできる**
  - グラフモデルアクセスに向いたクエリはグラフで
  - リレーショナルモデルに向いたクエリはリレーショナルで



GRAPH_TABLE 演算子

```
SELECT * FROM
  GRAPH_TABLE ( students_graph
    MATCH
     (a IS person) -[e IS friends]-> (b IS person WHERE b.name = 'Mary')
       WHERE a.name='John'
       COLUMNS (a.name AS person_a, b.name AS person_b )
);
```

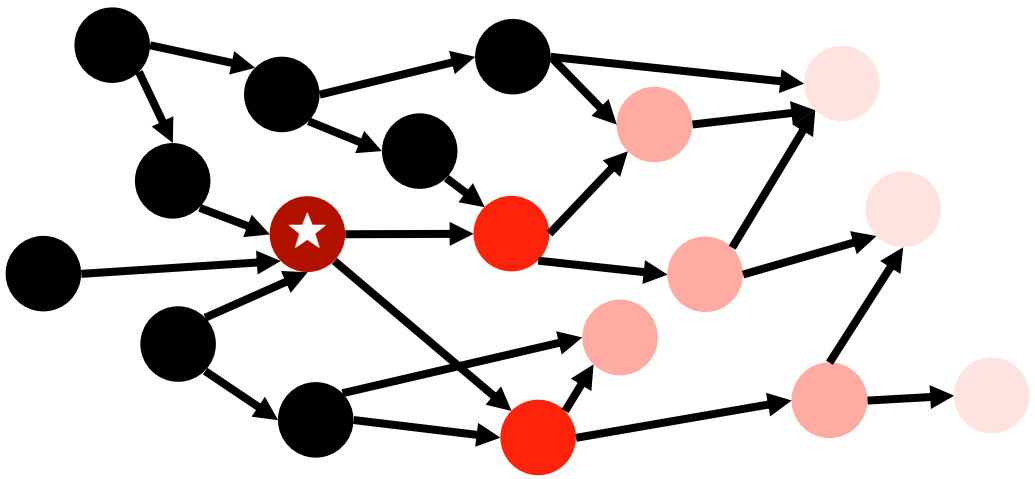
- students_graph …クエリ対象のグラフ名
- (a IS person) …ラベルがperson である ノードa
- -[e IS friends]-> …ラベルがfriendsである有向エッジe
- (b IS person WHERE b.name='Mary' …ラベルがpersonで、name属性がMaryであるノードb
- COLUMNS …戻り値の列名のbindを記述
- a.name AS person_a … ノードaのname属性をperson_a列に定義

GRAPH_TABLE による効果（表現力）

Find all accounts connected to `src` account in 1 to 3 hops

```
-- With new GRAPH_TABLE and MATCH syntax

SELECT account_id1, account_id2
   FROM GRAPH_TABLE(bank_graph
   MATCH (src)-[is bank_transfers]->{1,3}(dst)
   COLUMNS (src.id as account_id, dst.id as account_id2) );
```



```
-- With old SQL syntax (12 joins and 3 UNION ALLs)

SELECT v1.id as account_id1 , v2.id as account_id2
FROM   bank_accounts  v1 ,
       bank_transfers  btx,
       bank_accounts  v2
WHERE  (v1.id = btx.src_acct_id  AND v2.id = btx.dst_acct_id)
AND     v1.id= <src> AND v2.id= <dst>
UNION ALL
SELECT v1.id as account_id1 , v2.id as account_id2
FROM   bank_accounts  v1 ,
       bank_transfers  btx,
       bank_accounts  bc2,
       bank_transfers  btx2 ,
       bank_accounts  v2
WHERE  (v1.id = btx.src_acct_id  AND  bc2.id = btx.dst_acct_id  AND
        bc2.id = btx2.src_acct_id  AND v2.id = btx2.dst_acct_id  )
AND     v1.id= <src> AND v2.id= <dst>
UNION ALL
SELECT v1.id as account_id1 ,v2.id as account_id2
FROM   bank_accounts  v1 ,
       bank_transfers  btx,
       bank_accounts  bc2,
       bank_transfers  btx2 ,
       bank_accounts  bac4,
       bank_transfers  btx5 ,
       bank_accounts  v2
WHERE  (v1.id = btx.src_acct_id  AND bc2.id = btx.dst_acct_id  AND
        bc2.id = btx2.src_acct_id  AND bac4.id = btx2.dst_acct_id  AND
        bac4.id = btx5.src_acct_id  AND  v2.id = btx5.dst_acct_id  )
AND     v1.id= <src> AND v2.id= <dst>
;
```

SQLでできるということの意味

例：グラフモデリングの場合

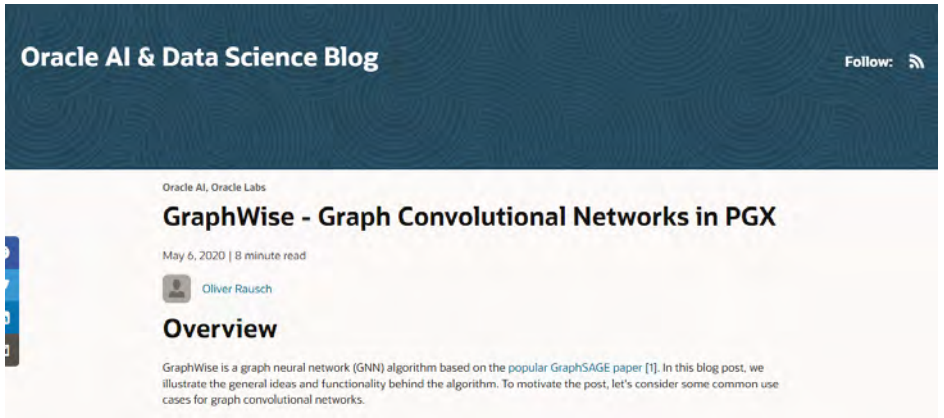
グラフデータベースなどを利用し分析作業などを行う場合、**グラフモデリング作業はトライ＆エラーを含むことが多く、タフな作業**となることが多い

例えば、右記のようなプロセスが一般に考えられる

グラフ分析対象データを一度オラクルデータベースに入れてしまいさえすれば、このようなグラフモデリングのプロセスを「元データ抜き出し」「グラフデータ変換」「グラフDBにロード」といったストッパーとなりがちな処理を挟まずに、SQLだけで回し切れるため、**モデリング作業の効率化に寄与**することができる

アルゴリズムの追加、改善

- Graph Attention Network (GAT)のサポート
  - Graph Nerural Networkアルゴリズムの１つである 教師あり/教師無し GraphWiseモデルにAttention+畳み込み層をサポートする拡張を追加
- アルゴリズムの改善
  - Adamic-Adar index
  - Local Clustering Coefficient
  - Prim's Algorithm
  - Shortest Path Hop Distance



※こちらは データベースのバージョンとは別に Property Graph 自体の機能として追加、改善されたものです。

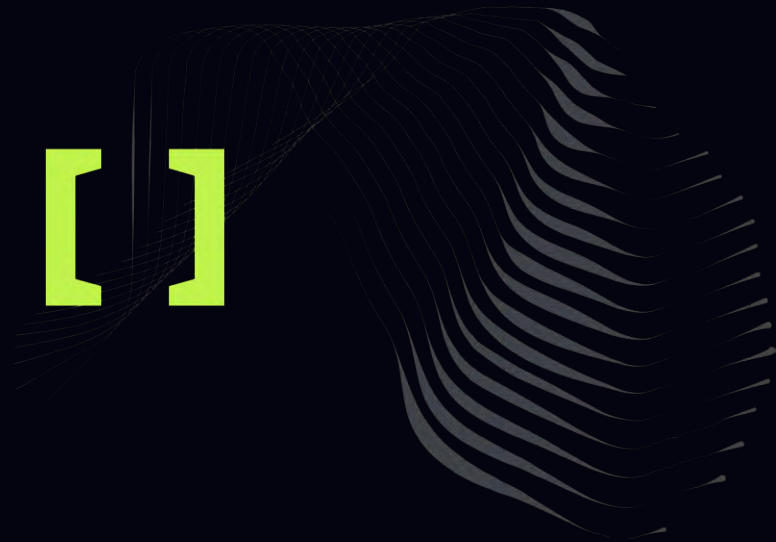
非推奨/廃止される機能等 (一部抜粋)

- Property Graph Shema が非推奨に (関連する関数群含む)
  - SQL Proprety Graph / PGQL Property Graph (create property graph 文で生成されるグラフ) への移行が推奨
- 関数等
  - opg_apis.get_version() 非推奨に
    - [対策] : OPG_APIS.GET_OPG_VERSION() を利用
  - OPG_APIS.GET_SCN 非推奨に
    - [対策] : DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER() を利用

個々の非推奨、廃止となる機能の詳細に関しては以下をご参照ください

Graph Developer's Guide for Property Graph - Changes in This Release for This Guide
https://docs.oracle.com/en/database/oracle/property-graph/23.3/spgdg/changes-in-this-release.html#GUID-75309029-ADEB-4C67-877E-F5639F1896FF

Oracle RDF Graph 新機能



Oracle RDF Graph　新機能

標準サポートのアップグレード

- GeoSPARQL 1.1 Support （1.0->1.1)

性能改善機能系のアップデート

- SPARQL->SQL変換を利用したクエリ最適化手法
- クエリ実行プランのコスト取得のサポート
  - SEM_APIS.GET_PLAN_COST関数
- RDF_LINK$ テーブルの自動リスト サブパーティション化のサポート

少し便利なアップデート

- 32K VARCHAR RDF 値のサポート
  - 4k以上はCLOBになっていたものがvarchar2で扱える

ありがとうございました