Oracle Database 12*c* Release 2 CoreTech Seminar

12.2.0.1

Oracle Spatial and Graph

日本オラクル株式会社
クラウド・テクノロジー事業統括
Cloud/Big Data/DISプロダクト本部
中井 亮矢
山中 遼太
2016/10

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, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

本日、お伝えしたいこと

- Spatial and Graph という面白いオプションがあること
- 新機能 プロパティグラフモデル の良さ

- R12.1 / R12.2 での各機能のアップデート
  - 既存ユーザー様向け

Agenda

1. Spatial and Graph のご紹介
2. 新機能 プロパティグラフのご紹介
3. その他のアップデート (Spatial/NDM/RDF)
4. まとめ

Agenda

1. Spatial and Graph のご紹介
2. 新機能 プロパティグラフのご紹介
3. その他のアップデート (Spatial/NDM/RDF)
4. まとめ

Spatial and Graph のご紹介

全体感

Spatial and Graph のご紹介

- Oracle Database Enterprise Editionのオプションです
- 非常に多機能なオプション製品です
  - 1つのオプションですが、12.1 時点で マニュアルが5冊あります

Spatial and Graph開発者ガイド
Spatial and Graph GeoRaster開発者ガイド
Spatial and Graphトポロジ・データ・モデルおよびネットワーク・データ・モデル・グラフ開発者ガイド
Spatial and Graph Java API Reference (Javadoc)
Spatial and Graph RDFセマンティック・グラフ開発者ガイド

- 個々の機能は非常に奥深くマニアックなものになっています

Oracle 在籍のマニアック研究者たちがお届けする
超強力な大規模ハイテクオプション

Slide Onlyの為
内容削除

Oracle Spatial and Graph
継続的に製品機能を改善、強化

20th +1 ANNIVERSARY

1995～ Oracle7
Spatialの初リリース
ポイント情報、ポリゴン
Spatial演算子

1999～ Oracle8i
オブジェクト・サポート
円、円弧
R-Tree索引
Spatial関数のサポート

2001～ Oracle9i
空間参照システム
(座標系のサポート)
線形参照システム
Spatial Partitioning
Spatial Replication(Adv.rep)

2004～
Oracle Database 10g
ラスター・データのサポート
トポロジ・データモデル
ネットワーク・データモデル
ジオコーディングとルーティング

2007～
Oracle Database 11gR1
3Dデータサポート
Spatial Webサービス
Oracle Database 11gR2
Google Maps対応

2013～
Oracle Database 12c R1
vector_acceleration
ラスター並列演算

2016～
Oracle Database 12c R2
プロパティグラフモデル
Composite B-Tree索引
点群Hilbert系モデル

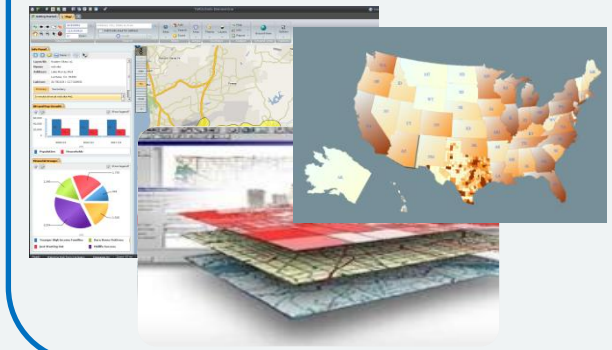
機能実装は Oracle 4 から
オプションとして Oracle 7 から
グラフの実装は Oracle 10g から

Oracle Spatial and Graph Option

12c R1

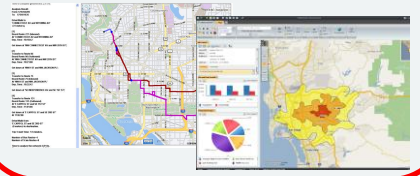
Oracle Database Enterprise Edition
Spatial and Graph Option

Spatial

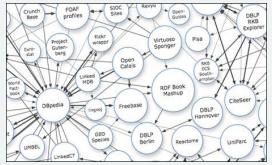


Graph

Topology & Network Data Model Graph



RDF Semantic Graph



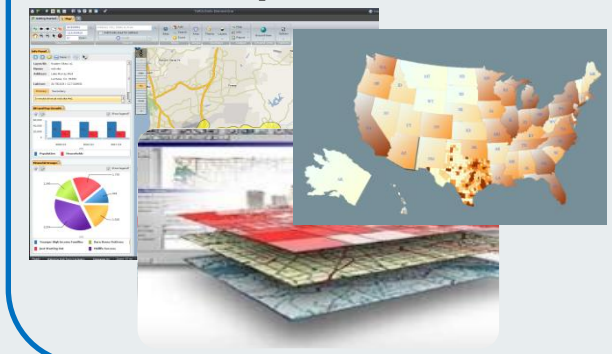
Oracle Spatial and Graph Option

12c R2

Oracle Database Enterprise Edition
Spatial and Graph Option

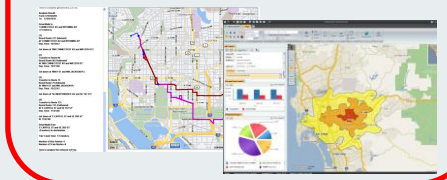
更に
肥大化

Spatial

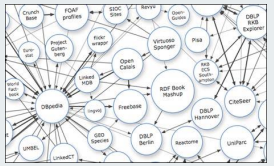


Graph

Topology & Network Data Model Graph



RDF Semantic Graph



RDF

Property Graph

Spatial and Graph のご紹介

Spatial

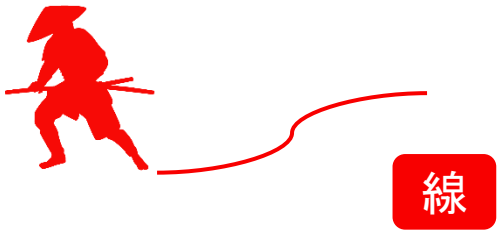
空間データって何？

空間データってなに

いろんな形や場所を表すデータ

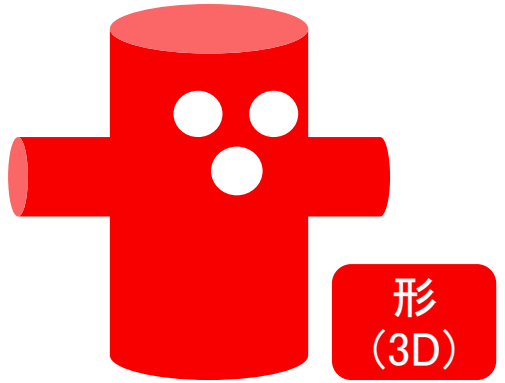
地球上のデータ

- **地球上の形や場所を表現(2D/3D)** -- たいてい**緯度経度**
  - 地球上の**点**のデータ（GPS,位置,所在地,店舗,駅...）
  - 地球上の**線**のデータ（道路,線路,河川,動線...）
  - 地球上の**形**（ポリゴン）のデータ(行政区画,国,海,湖,農地..)
  - 地球上の**面**（点群/TIN)のデータ(地表面,地層, 海水面..)



デカルト座標系のデータ

- X/Y/(Z)が**垂直に交差**する仮想的な座標系(2D/3D)
  - 機械,電子部品,ビル,建造物,（多くの人工物が該当）



空間データって何？

何がむつかしいの？

地球は丸いです

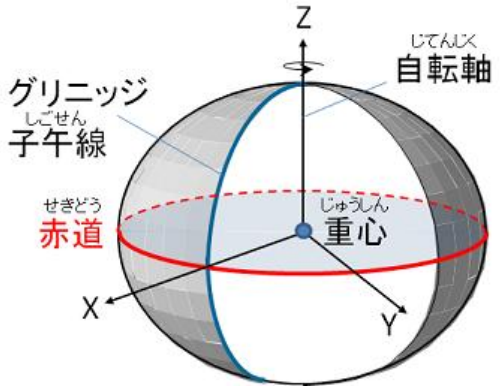
- 緯度経度は直交座標系じゃないです
  - 経線は南極と北極で交わります
  - 緯度経度の世界に特有のことがいろいろあります

北緯20度東経20度 – 北緯21度東経21度 の距離 (152.110 km)
北緯30度東経30度 – 北緯31度東経31度 の距離 (146.647 km)

地球は丸いけど真球じゃないです

  - ちょびっと極がでっぱって　横もひずんでます
  - 緯度経度は想定する回転楕円体と標準の地表面を定義して決定されます
- 国や用途によって座標や投影の取り方が違います（各国の測地座標系）

これらを考慮して**一から実装するのは結構大変**です



出典: 国土地理院こどものページ
http://www.gsi.go.jp/KIDS/KIDS13.html

空間データって何？

他にもいろんな関連データがあります

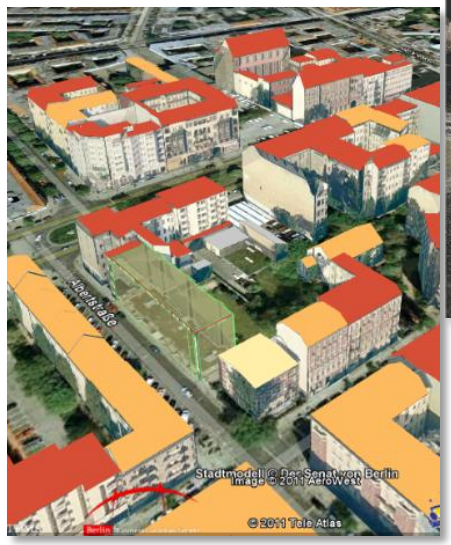
電子地図の背景データ

- ラスターデータ(衛星写真,航空写真、画像化したベクタデータ)

データ:
DigitalGlobe

地形、ビルなどの形状データ,LiDARデータ

- 点群/不規則三角網(las,laz,pcd..)

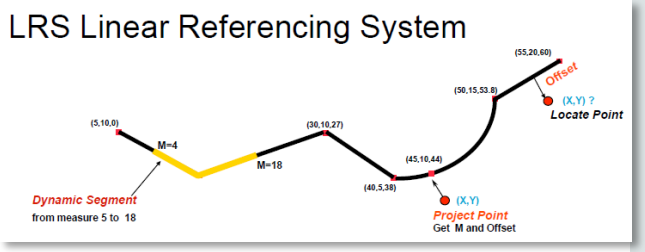


3D都市データ

- CityGML

動線データ

- GPS(KML,GPX..)



空間データって何？

他にもいろんな関連データがあります

電子地図

- ラスター

地形、ビ

- 点群/不

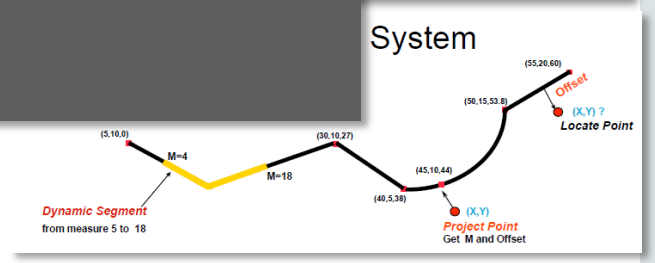
3D都市

- CityGML

動線デー

- GPS(KML,GPX..)

全て Oracle Spatial で管理できます

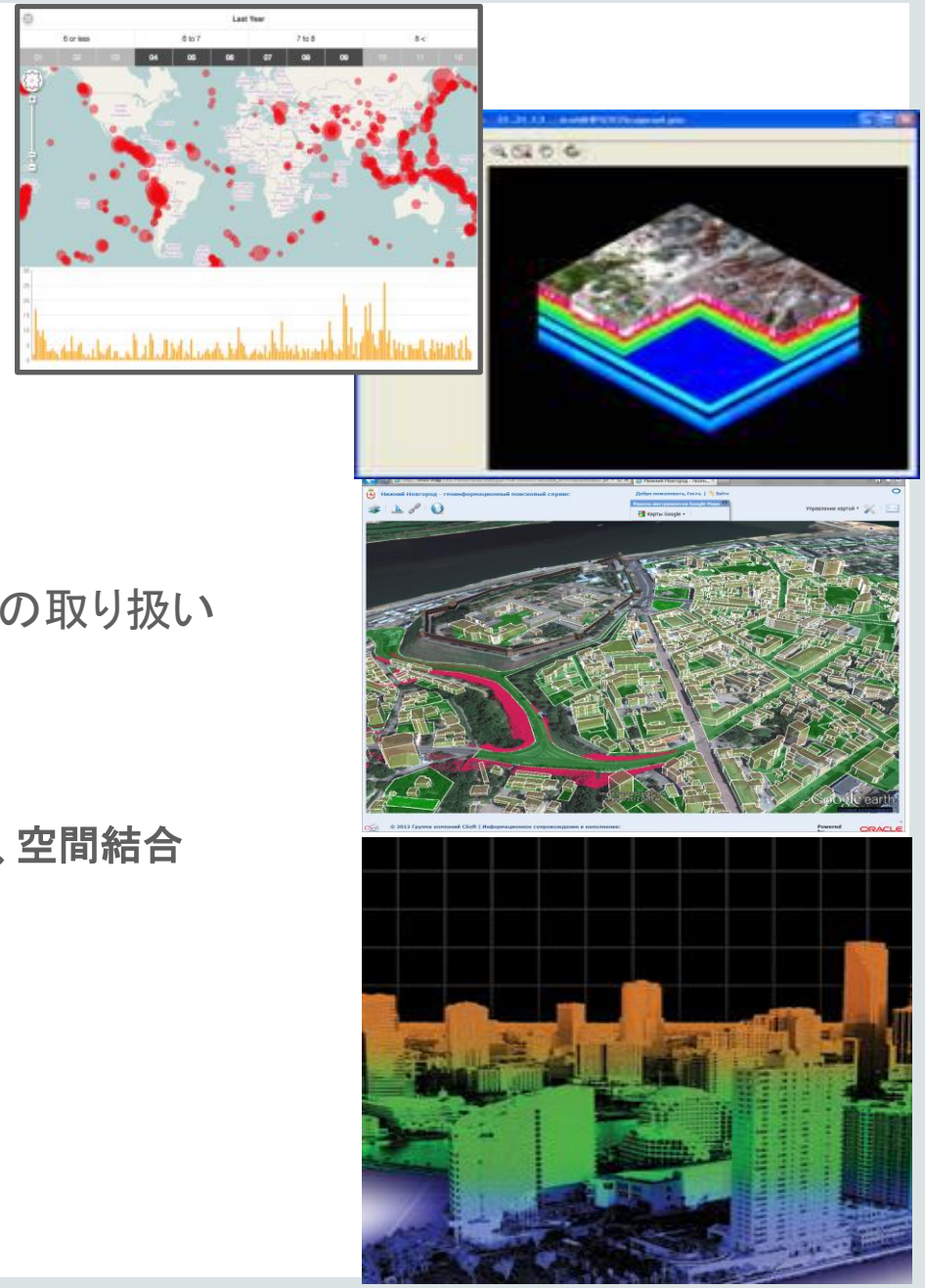


Spatial機能紹介

概要

地理空間、地図、位置情報を高度に活用

- **機能**
  - **地理空間データ**(緯度経度、測地座標系),**3D,LIDARデータ**の取り扱い
  - **地図・衛星画像・リモートセンシングデータ**の取り扱い
  - 空間索引による高速検索
  - 高度な**分析機能** – 線形参照システム、空間演算子、空間集計、空間分析、空間結合
  - 空間データの**変換、検査、補正、修正などの関数**も豊富に用意
- **特徴**
  - 高い機能性
  - 高い性能とスケーラビリティ



Spatial機能紹介

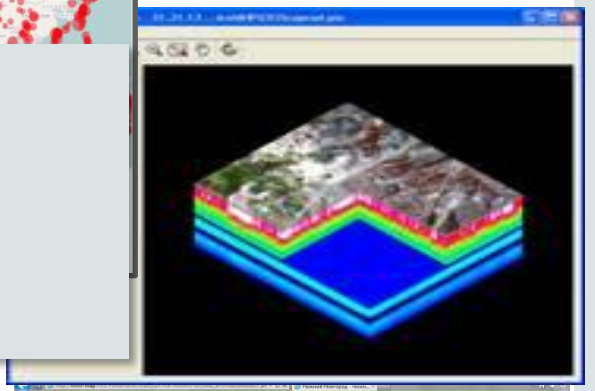
概要

地理空間、地

- **機能**
  - 地理空間デ
  - 地図・衛星画
  - 空間索引による
  - 高度な**分析機能** – 線形参照システム、空間演算子、空間集計、空間分析、空間結合
  - 空間データの**変**
- **特徴**
  - 高い機能性
  - 高い性能とス

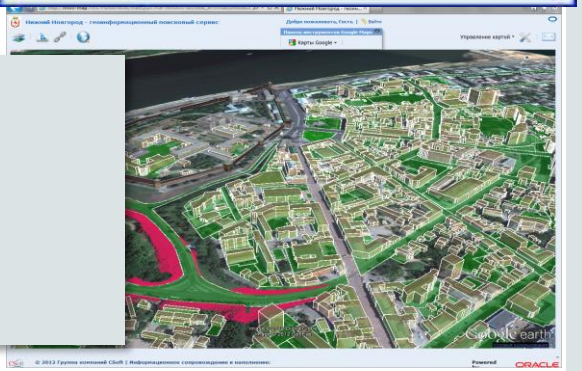
ベクトルデータ関連の実装関数 約250種

- 豊富な空間位相判定
- 空間データの変換、分析関数、測地系変換
- ラスター、3D、点群、不規則三角網(TIN)にも対応



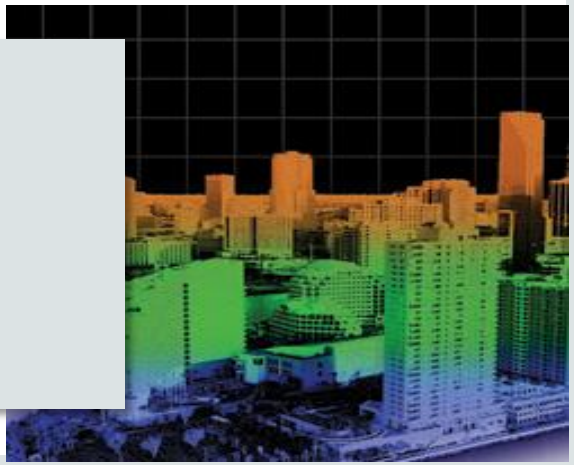
ラスターデータ関連の実装関数 約190種

- オルソ補正,光学補正,モザイク化,ピラミッド化
- 演算(ラスター代数)



一般的な空間DBの限界を突破

- 検索範囲を範囲階層で絞り込む索引構造
- パーティションによるパラレルクエリ対応
- 索引のパーティション化にも対応
- 12c新機能SVAにより更なる高速化



空間データって何？

位置情報、空間データの問い

- 自社の店舗の**半径200m以内**にある競合店舗は？

- **高所得エリアに住んでいる**が自社の火災保険に入っていない顧客は？

- **現在地から最も近い道路2つとその距離**は？

Spatial 機能紹介

位置情報、空間データの問い

- 自社の店舗の**半径200m以内**にある競合店舗は？

```
SELECT  a.TENPO_ID , b.TENPO_ID  FROM TENPO a, COMP_TENPO b
WHERE  SDO_WITHIN_DISTANCE ( a.LOCATION, b.LOCATION,'distance=200,unit=m') = 'TRUE';
```

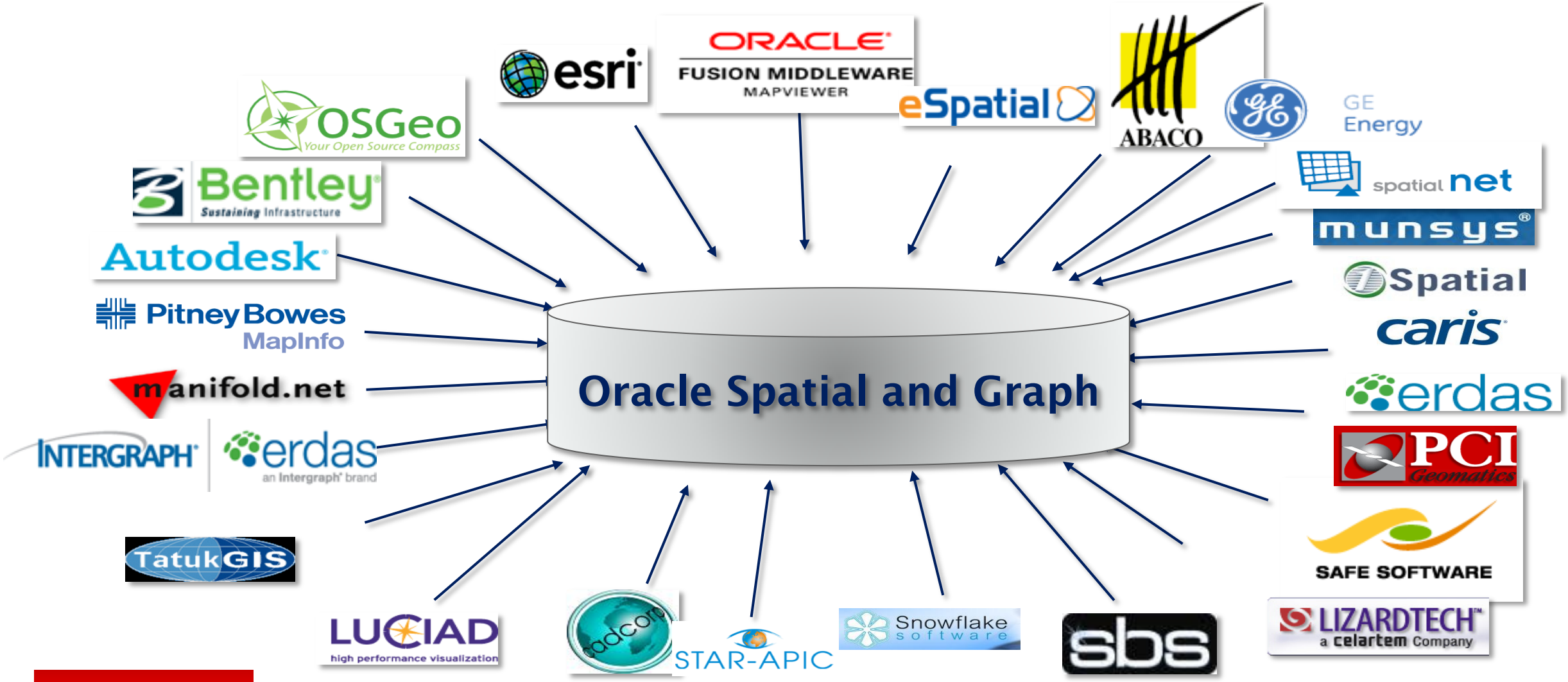
- **高所得エリアに住んでいる**が自社の火災保険に入っていない顧客は？

```
SELECT  a.CUST_ID  FROM CUSTOMER a, CUST_KEIYAKU b,  SYUNYU_LAYER c
WHERE  a.CUST_ID = b.CUST_ID AND b.KASAI_KEIYAKU = FALSE
AND SDO_INSIDE( a.ADDRESS_POINT, c.AREA ) = 'TRUE';
```

- **現在地から最も近い道路2つとその距離**は？

```
SELECT  a.ROAD_ID,SDO_NN_DISTANCE(1)  FROM ROAD a
WHERE  SDO_NN( a.ROAD_LINE,  ? ,  'sdo_num_res=2',1) = 'TRUE' ;
```

様々な分野のアプリケーションでサポートされています



Oracle Spatial の導入顧客(一部抜粋)

既に欧米では大規模空間DBとしてデファクトの域

電力/水道/ガス	Omaha Public Power, Reliant, Southern, US DoE, Western Power Corp, Severn Trent, Bejing Power, Georgia Power, Czech Telem, Copenhagen Energy, Electrable, Gaz de France, Hydro-Quebec, Equitable Resources, Nova Naturgas, Sao Paulo Electric, Xcel Energy, Pemex, Romande Energie, Societe du Canal de Provence, Burlington Hydro, Santos Oil and Gas,
通信	AT&T, Bell South, British Telecom, Cingular, DoCoMo, Intrado, Nextel, Sprint, T-Mobile, Telkom, Telenor, Telstra, Telus, Telia, Cellcom, Verizon, VIAG, Vodaphone, Wind, TurkCell, Geodan Mobile Solutions,
輸送/運輸	German Rail, Austrian Rail, California, Iowa, Florida, Maine, Maryland, Minnesota, New York, Oklahoma, Pennsylvania, Alabama, Alberta, London Rail, Netherlands Transport, Australia, CSX transport, COTRAL SpA, BRAVO, Dublin Bus,
自治体	Berlin, Dutch Police, New York City, Chicago, Los Angeles, San Jose, San Mateo, Washington DC, Cleveland, Detroit, Phoenix, Winnipeg, Vancouver, Edmonton, Stockholm, Las Vegas, Sun Francisco MTA, Moscow, Beijing,Dongcheng , Hague, Luton Borough Council, Ohaio, Hull, Nanjing Land Resource,
国土地理, 土地台帳 & 治水/利水,国防	Ordnance Survey (UK, IR, NI), US Census, NIMA, USGS, US Army, Denmark, Sweden, The Netherlands, Poland, Australia, Singapore Land Authority, Las Vegas, NDPPC, Forestry Commission(Eng), The Barletta, Andria, and Trani Public Health Unit, Servicio Geologico Mexicano,, Arma dei Carabinieri, Instituto Nacional de _Defensa Civil(PE), Kort & Matrikelstyrelsen(DE)
民間企業	LocationBox (GeoMarketing SaaS), Geofusion(GeoMarketing SaaS), Unicoop Firenze (GeoMarketing), Oracle Real Estate, Garmin, DigitalGlobe, Intermap Technologies, Where 2 get It, Oct telematics, Exor Corp, AngloGold Ashanti, Burger King, kredi Kayit Burosu A.S. , Cleveland Clinic, Brazilian Superior Electoral Court, INFOTECH,

Spatialの技術:測地系の話

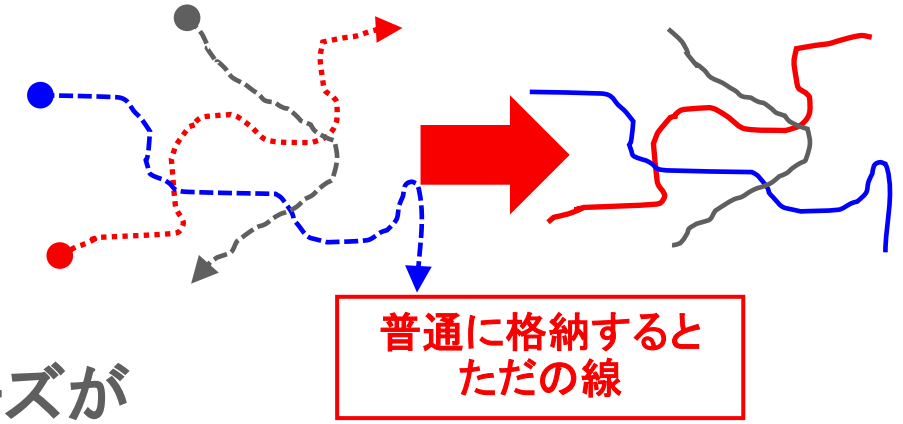
- **測地系**は地球上のデータを表現するための基準になるものです
  - 地球の形(回転楕円体)、標高の基準、座標の基準の取り方で変わります
- 現在も各国で自国の国土を正しく計測するため多くの測地系があります
- 国内外の地理情報を広く活用する際の障壁にもなっています
- Oracle Spatialでは測地系の変換を暗黙的/明示的に行う機能を持っているので、この壁を突破できます

測地成果	測地系	楕円体	投影座標	EPSG	Oracle SRID
旧測地成果	日本測地系 TOKYO	ベッセル	緯度経度	4301	4301
			UTM座標系(51N-55N)	3092-3096	3092-3096
			平面直角座標系	30161-30179	30161-30179
新測地成果2000	日本測地系 JGD2000	GRS1980	緯度経度	4612	4612(8307)
			UTM座標系	3097-3101	3097-3101
			平面直角座標系	2443-2461	2443-2461
米国	世界測地系	WGS84	緯度経度	4326	4326 (8307)
			UTM座標系(51N-56N)	32651-32656	32651-32656

```
SQL> SELECT count(1) FROM MDSYS.CS_SRS ;
  COUNT(1)
----------
      5982
```

※日本だけでも旧測地系、世界測地系の2種があり、更に投影の方法によってさらにデータ表現が変わります。右表は、比較的国内でよくa利用されている測地系になります

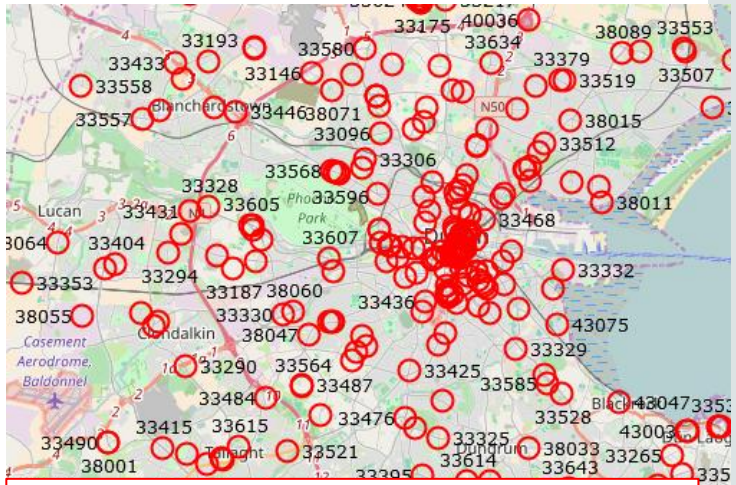
Spatialの技術:動線の管理について



線形参照システム(LRS)

- 測位技術の進歩とともに動線データの活用/管理ニーズが高まっています
- 一般的な地理空間データ表現では動線は線として格納され、時間と切り離されてしまいます
- GPSはバラバラな時刻で取得されるので大量のGPSデータから指定時刻の全員の位置を取り出だけでも一苦労です
  - GPSデータは取得した時間で入ってきます
    - Aさんは 07:59:52, 08:00:02, の位置, Bさんは 07:59:57, 07:59:07 の位置...
- Oracle Spatialでは、線形参照システムを使うことで動線を動線として管理でき、上述の問題も解決できます

例)ダブリンのバス400台の動線から
20:38:59の位置を一括取り出し

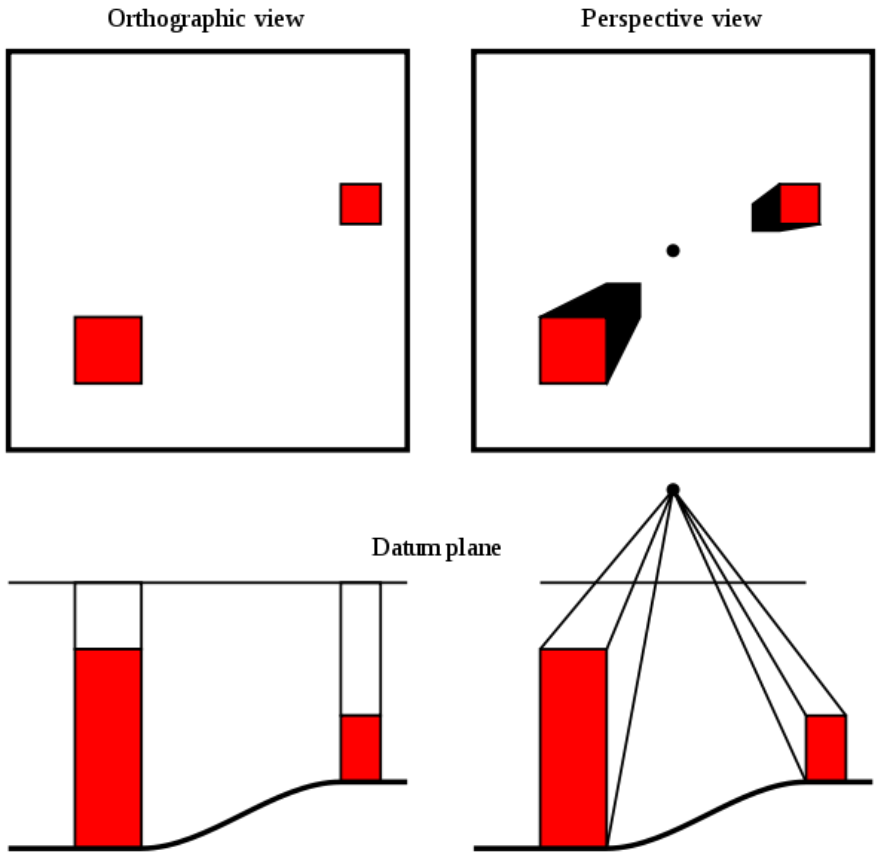


```
SELECT  vehicle_id,
  SDO_LRS.GET_NEXT_SHAPE_PT(
  lrs, 1357040339500000)
FROM  dublin_lrs
```

Spatialの技術:衛星写真/航空写真の補正の話

オルソ補正

- 衛星写真、航空写真は上空から撮った2次元のデータです
- 衛星、飛行機の位置や地形の標高によって写真にはひずみが発生します
  - 結果的にこのままでは実際の地形データと重ねることができません
- どこからでも真上からみたようにするには投影しなおす必要があります
- この処理をDB内で実行することができます



出典: Wikipedia:オルソ補正

Spatial and Graph のご紹介

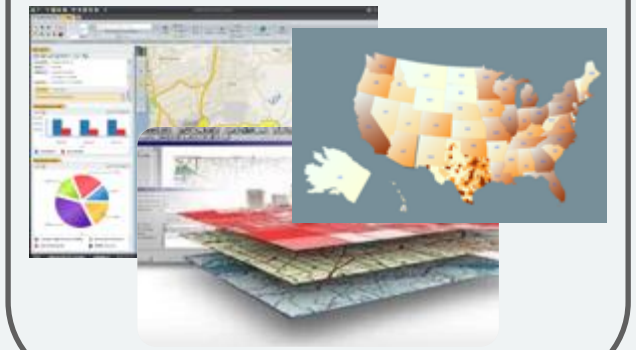
Graph

Spatial and Graph オプション

Oracle Database Enterprise Edition 12c R2

Spatial and Graph オプション

Spatial機能



Graph機能

ネットワーク・
データ・モデル

RDF セマンティック
・グラフ

プロパティ・グラフ

グラフとは

グラフって何？

円グラフ・棒グラフ
折れ線グラフのグラフ?

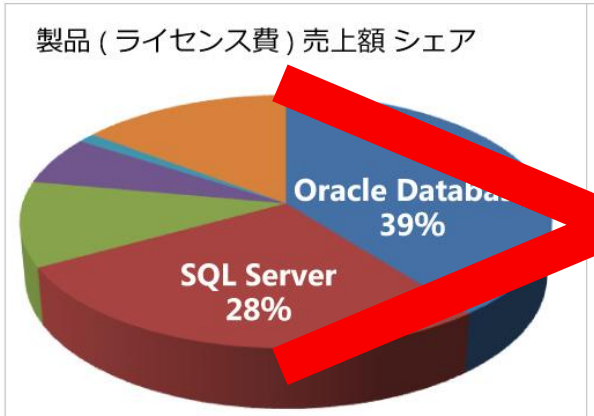


図 4 : 国内 RDB/DWH 市場規模 (製品別ライセンス金額)
出典 : 株式会社テクノ・システム・リサーチ (2010 年 8 月調査) より

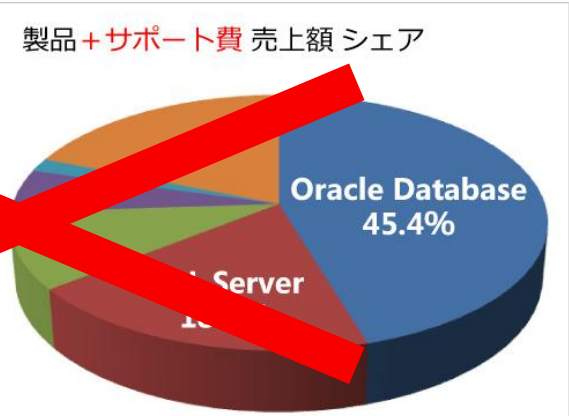
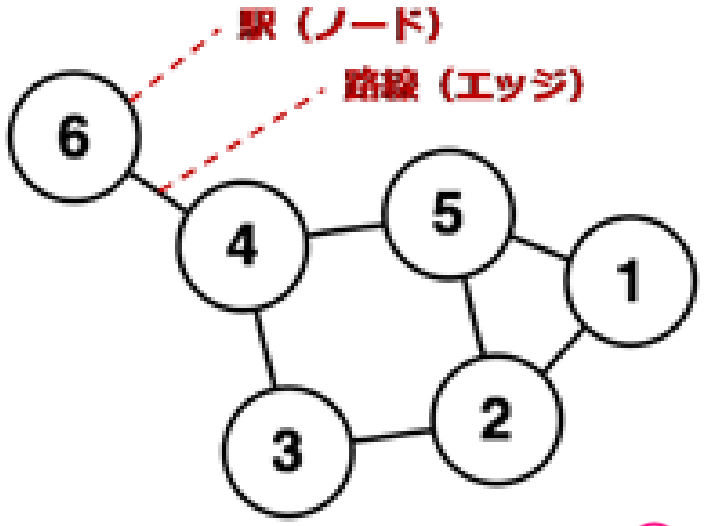


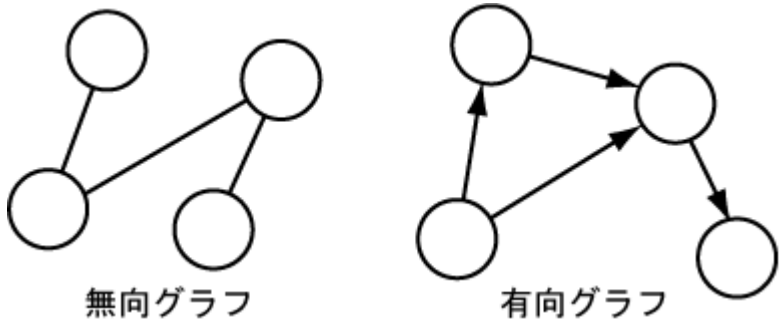
図 5 : 国内 RDB/DWH 市場規模 (製品別ライセンス & サービス金額)
出典 : 株式会社テクノ・システム・リサーチ (2010 年 8 月調査) より

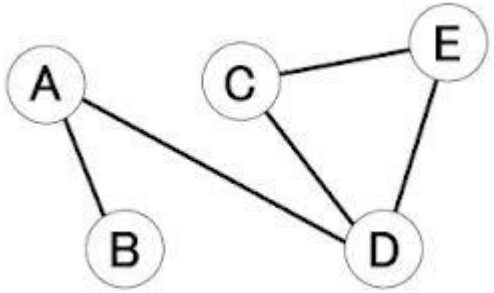
グラフとは

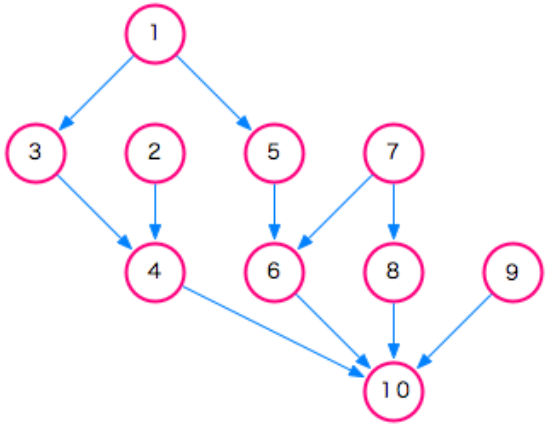
グラフって何？

数学の集合論のグラフ
点と線の集合を扱う



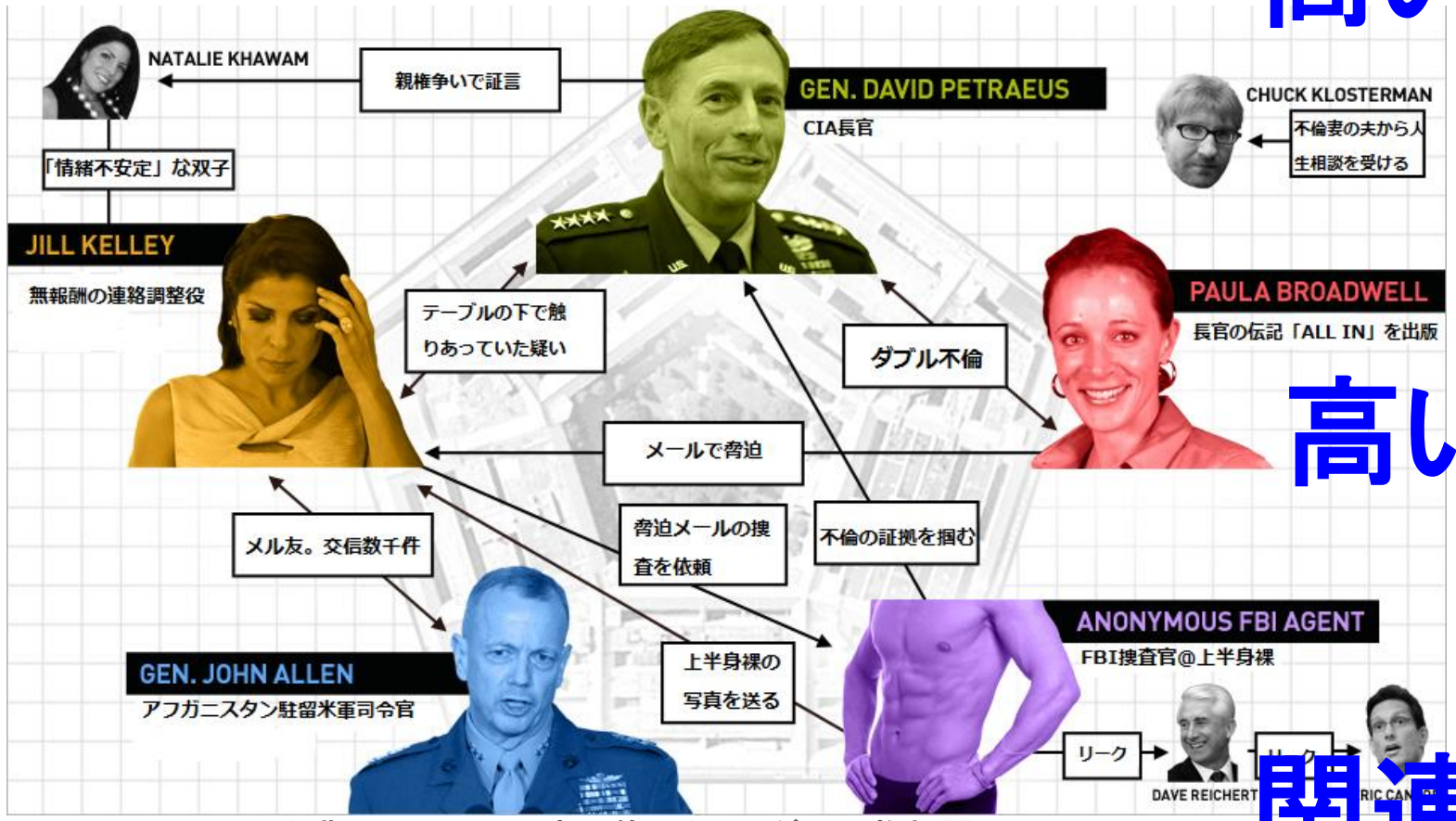






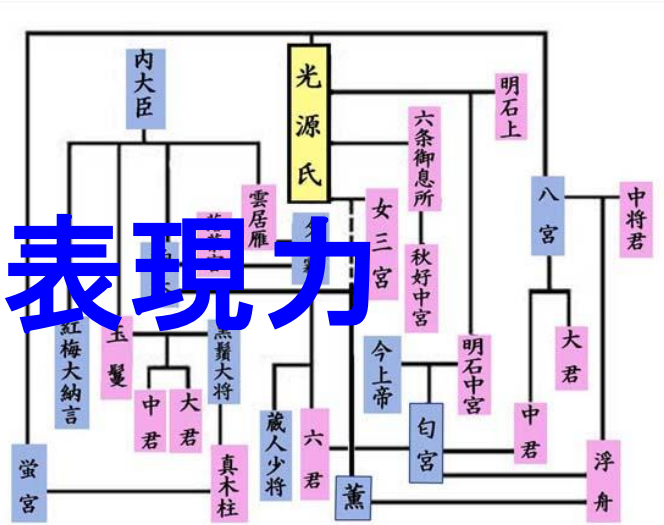
グラフとは

グラフで関係を表現した例



出典：不明...CIA長官不倫スキャンダル人物相関図
http://www.gizmodo.jp/2012/11/cia_3.html

高い表現力



高い柔軟性

関連を辿る検索

グラフとは

関係性を表現・分析できる**グラフ**に**注目**が集まっています。

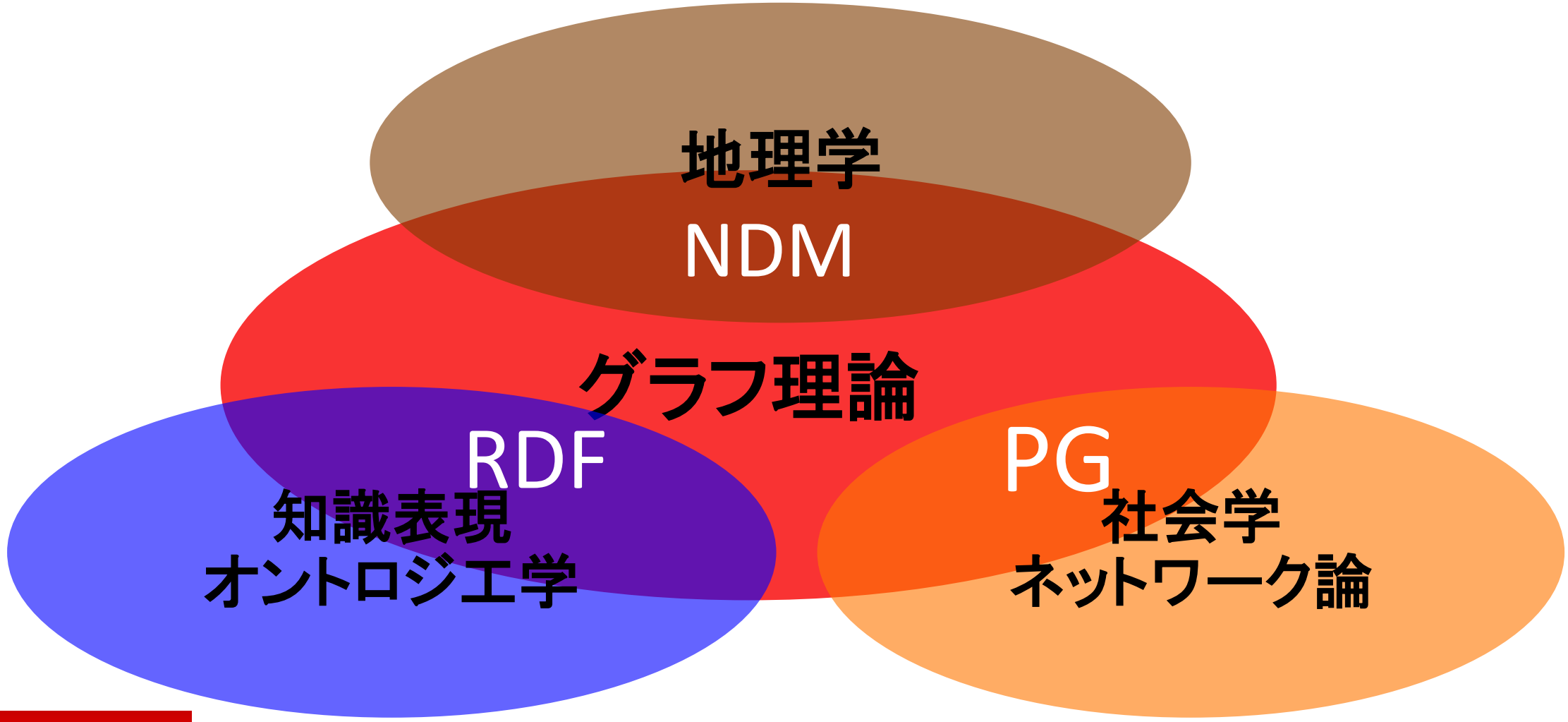
出典:TechTarget Japan

出典:日経コンピュータ,ITPro

出典:IT Leaders

出典:@IT

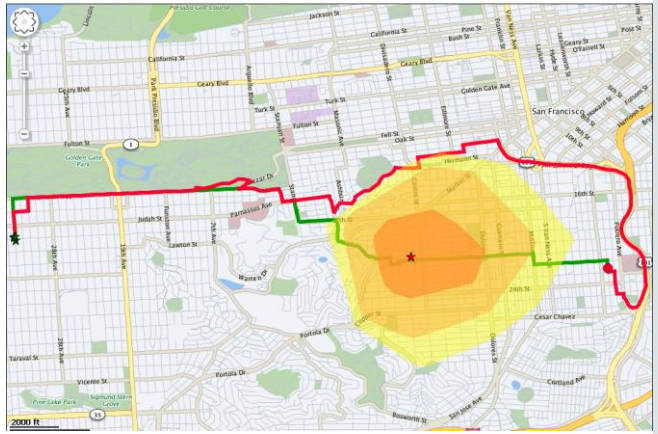
Oracleが提供する3つのグラフ

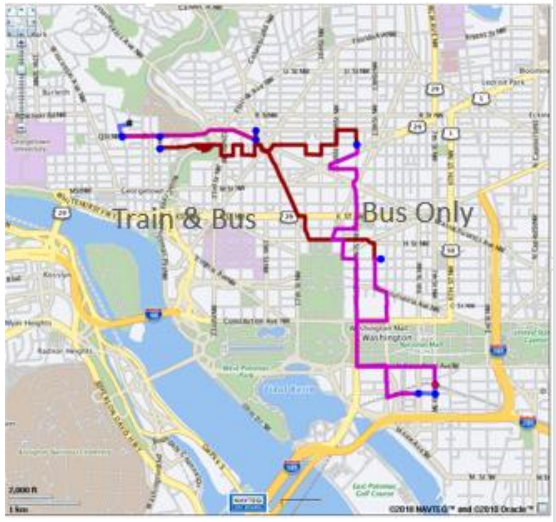


Network Data Model

地理的ネットワークの特徴

- 道路には経路上の様々な制約があります
  - 右折/左折禁止, 高さ制限, 重量制限,速度制限,
- 時間によって発生する制限もあります
  - 夜間一方通行, 夏だけ一方通行,学校の下校/登校時間による車両制限
  - 渋滞、災害
- 複数の交通手段が可能な場合もあります
  - 電車→徒歩→バス→電車
- その位置に紐づく特徴があります
  - ショッピングモールの駐車場の入口、バス停、タクシー乗り場,,
- リアルな経路分析にはこのような制約や条件を柔軟に定義・加味できるデータモデルが必要とされます





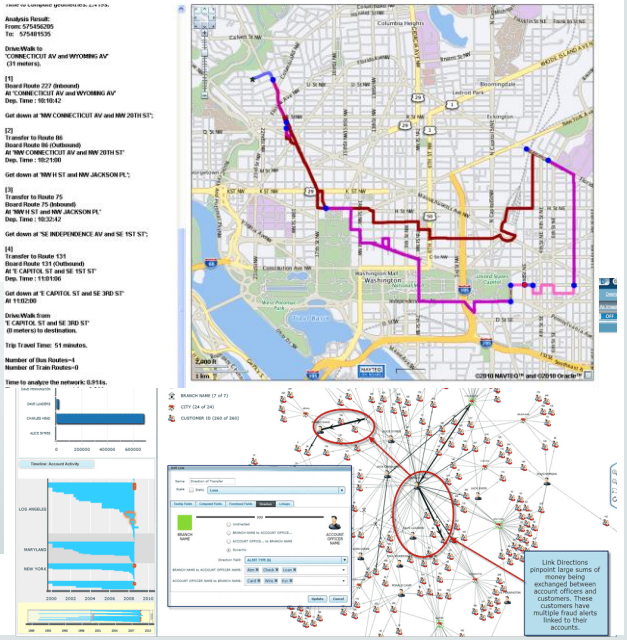
Graph Features - Network Data Model

地理空間や論理空間での経路や関係性を分析

- **機能**
  - **ネットワーク/トポロジーデータモデル**（位相関係の固定が可能）
  - **ロードオンデマンド**（大規模ネットワークへの対応）
  - **様々な地理的ネットワークの制約を取り込んだリアルな分析が可能**
    - 電車・バス・自動車などの複数の交通手段を跨いだグラフ
    - 時間帯による制限、右折禁止、高度制限など経路上に発生する制約
  - **Spatialの地理空間機能**との連携(距離/位置/範囲検索など)
- **特長**
  - **地理空間系ネットワークに特化**した機能が豊富
  - 地球規模の**大規模なネットワーク分析**が可能
  - **細かなネットワークの制約**を設定、変更可能

利用用途:
- 電力・水道などライフライン管理
  - 断水時の影響範囲分析など
- 輸送経路の最適化
- ネットワーク経路網のモデル化
- 交通網のシミュレーション

主要な経路探索アルゴリズムが実装済み
最短経路（ダイクストラ法,A*Star）
K短経路
巡回セールスマン

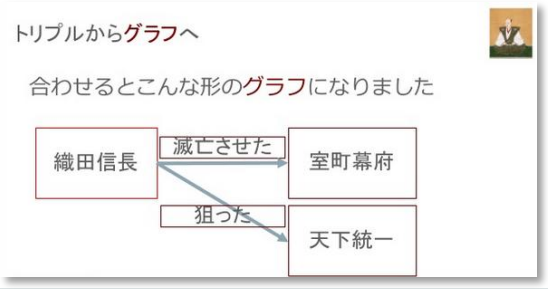


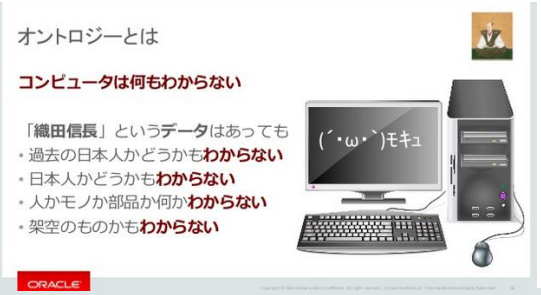
RDF Semantic Graph

RDF Semantic Web とは

- RDFはW3Cで標準化されているSemantic Webの規格になります
  - W3C Resource Description Framework　https://www.w3.org/RDF/
- メタデータの意味的統合やリンクトオープンデータ等で利用されています
  - 推論機構を含む高度なデータ管理を実現するデータモデルです
- RDF/OracleのRDF機能にご興味がございましたら弊社担当営業までご連絡下さいませ
  - RDF Semantic Graph「RDF 超入門」を slideshareにて公開しています

    http://www.slideshare.net/oracle4engineer/rdf-semantic-graph-intro

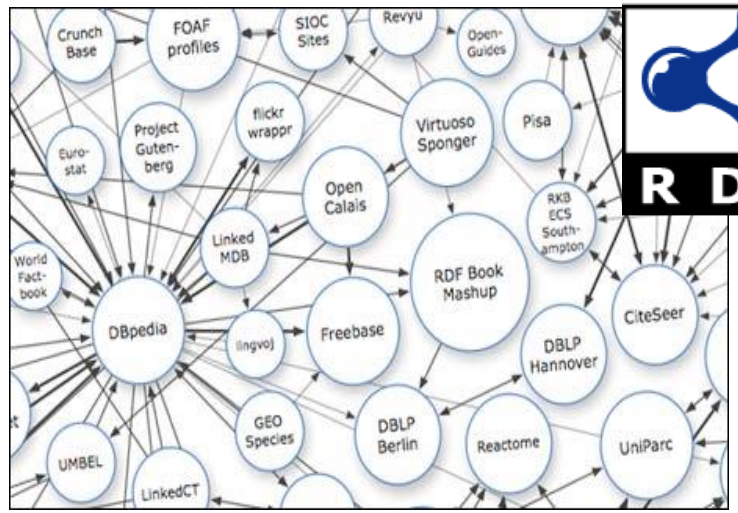




Graph Features - RDF Semantics Graph

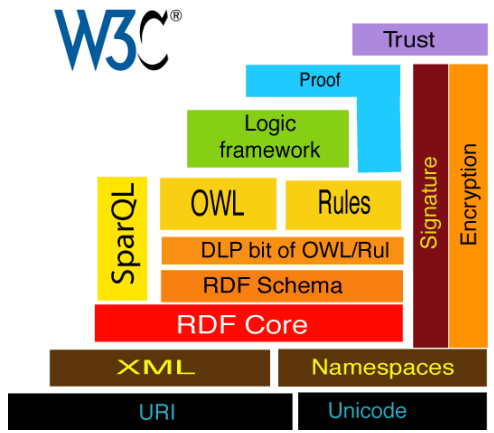
RDFセマンティックテクノロジーの実装

- **機能**
  - **最新のSPARQL 1.1対応の高性能RDFトリプルストア**
  - **RDB 2 RDF の変換プロセッサ(DM,R2RML)とビュー機能**
  - **データベース内のフォワードチェイニング型推論エンジン**
    - RDF/RDFS/OWL/SKOS標準ボキャブラリに対応
    - ユーザー定義での推論も可能
- **特長**
  - 1兆トリプルを捌く #1 **大規模トリプルストア**
  - **既存データ、RDBユーザーに優しい**
  - **企業・軍情報機関での利用に耐えるスケーラビリティ、セキュリティ**
  - **標準的なOSSツールとの連携,アダプタの提供**



R2RML

利用用途:
- **リンクト・オープンデータ**
- **メタデータの意味的統合**
- **共通語彙基盤の構築**



Oracle Spatial and Graph / RDF Semantic Graph

- **大規模トリプルストアとしてナンバーワン**
  - 1兆トリプルを取り扱えるキャパシティ

    https://www.w3.org/wiki/LargeTripleStores
- **DB内の推論エンジン**
  - RDFS/OWL/SKOS標準推論をDB側に標準搭載
  - ユーザ定義の推論も自由に記述が可能
  - 推論結果を既存トリプルと分けて保持
- **高いセキュリティ**
  - モデル～トリプル単位でアクセス権を管理
- **標準技術の強力な実装**

並列ローディング、並列クエリなどDBの機能をフル活用できる設計

**大規模なデータアクセスとなる推論をデータの在処で実行できる。
推論結果はパーティションに格納。**

**Oracle Label Securityを
トリプルストアに適用可能**

**SPARQL 1.1,GeoSPARQL,
SPARQL 1.1 updates,
OWL EL, R2RML**

Agenda

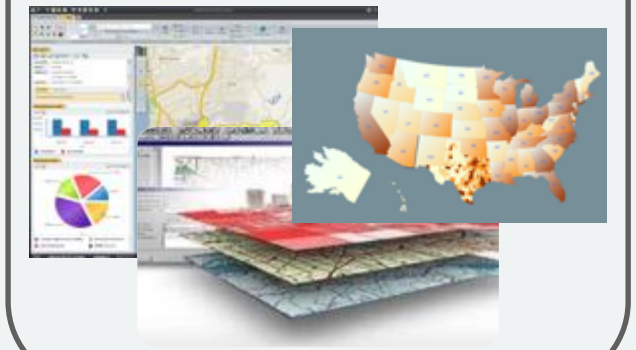
1. Spatial and Graph のご紹介
2. **新機能 プロパティグラフのご紹介**
3. その他のアップデート (Spatial/NDM/RDF)
4. まとめ

Spatial and Graph オプション

Oracle Database Enterprise Edition 12c R2

Spatial and Graph オプション

Spatial機能



Graph機能

ネットワーク・
データ・モデル

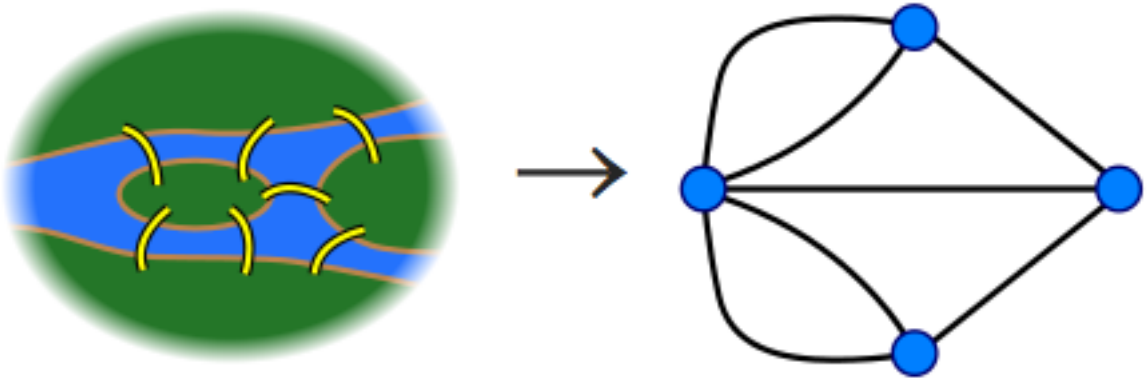
RDF セマンティック
・グラフ

プロパティ・グラフ

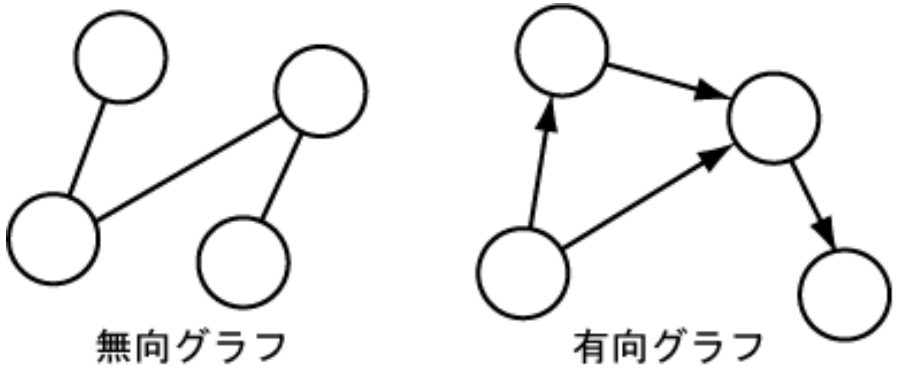
グラフ・データベース

ネットワーク分析とグラフ

- さまざまな関係構造をネットワークとして扱うことができる
- このネットワークを点と線の集合として抽象化したものがグラフ
- グラフ理論は数学の一つとして古くから発展してきた



1736年 ケーニヒスベルグの問題(オイラー)

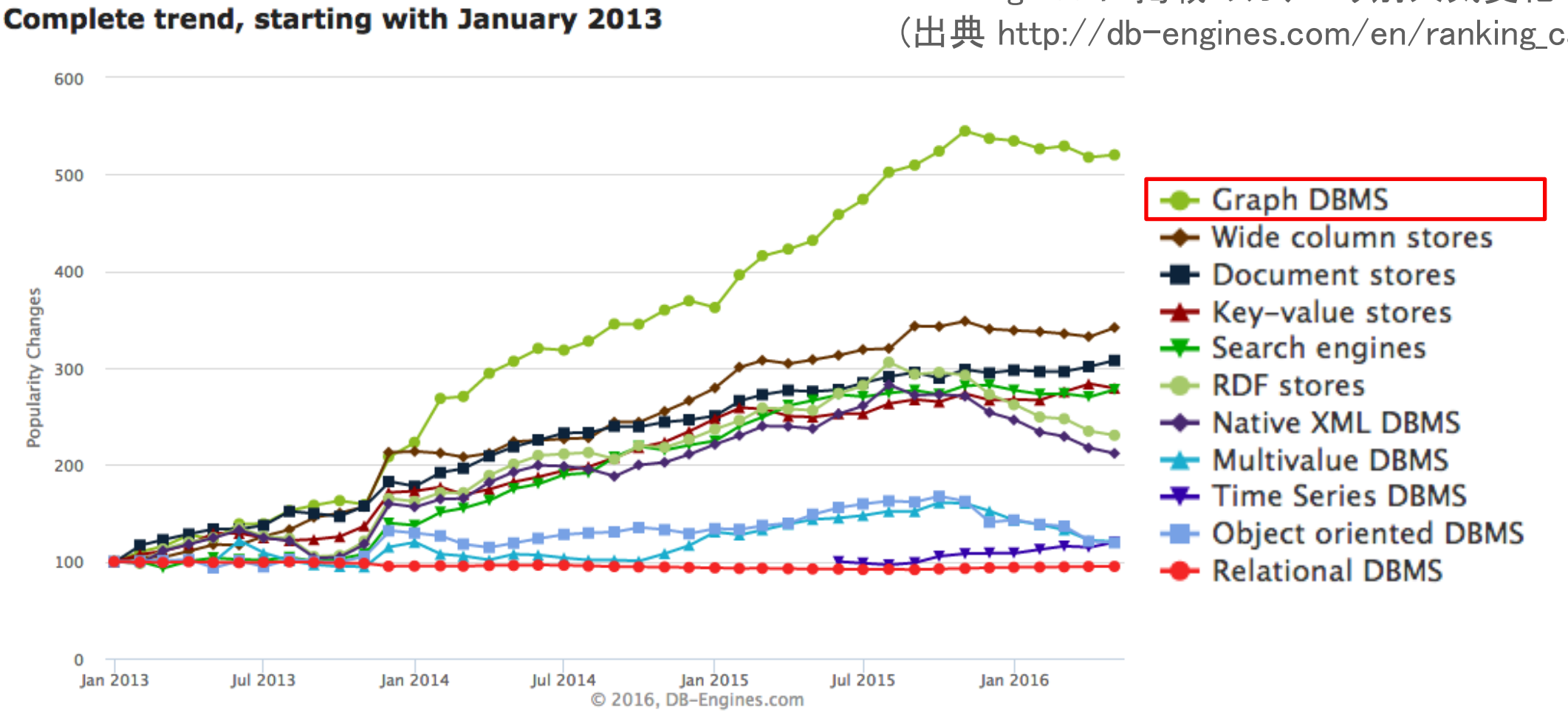


無向／有向グラフ、多重グラフ、重み、…

※ 出典 https://ja.wikipedia.org/wiki/一筆書き (CC BY-SA 3.0)

グラフ・データベース

DB-Engines に掲載のカテゴリ別人気変化率
(出典 http://db-engines.com/en/ranking_categories)



データを表に格納する

社員表

empno	ename	job	deptno
1	アルバート	課長	1
2	バクスター	主任研究員	2
3	チェン	係長	3
4	デイヴィス	エンジニア	2

主キー

外部キー

部門表

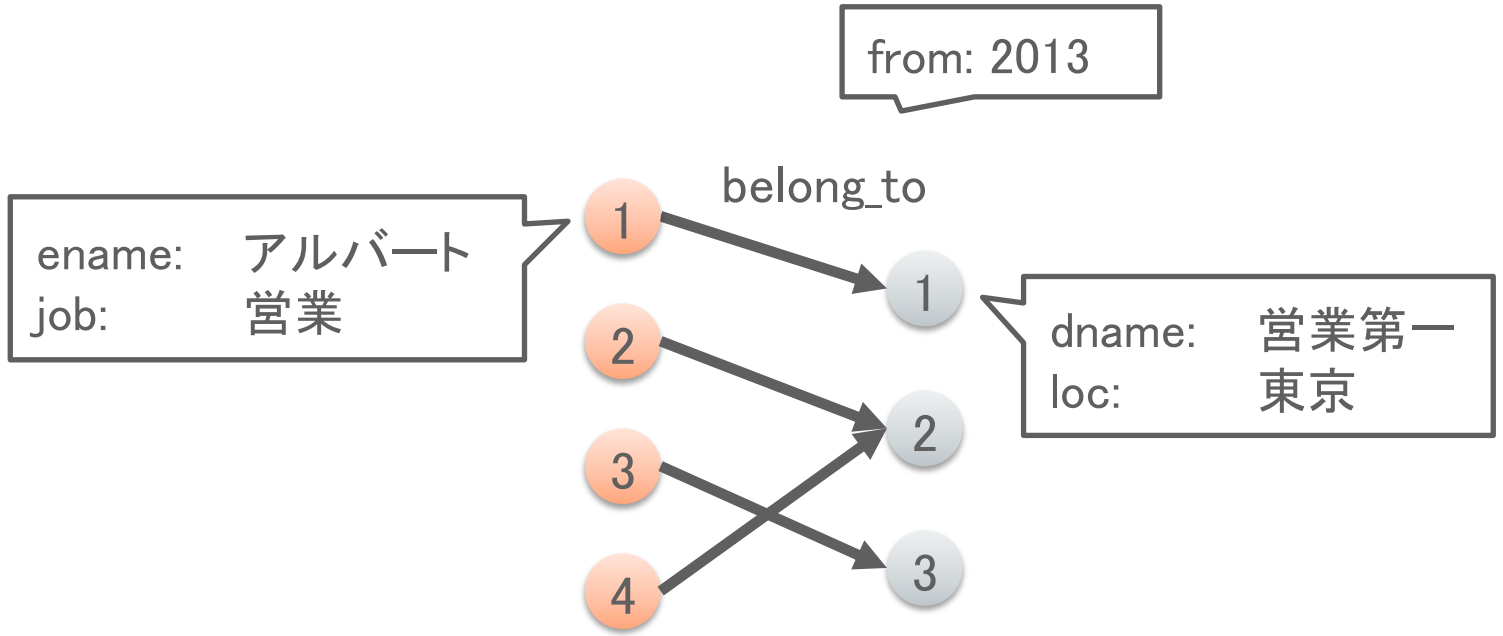
deptno	dname	loc
1	営業第一	東京
2	研究開発	大阪
3	マーケティング	東京

主キー

```
SELECT ename
FROM emp e, dept d
WHERE
    e.deptno = d.deptno
AND d.loc = 'Tokyo'
```

データをグラフに格納する

プロパティ・グラフなら、簡単かつ柔軟に表現することができ、
さらに、グラフ用の直感的なクエリ言語で検索することができる




```
SELECT e.ename
WHERE
    (e)-[belong_to]->(d)
  , d.loc = 'Tokyo'
```

誰が最重要人物か？ — つながりの定量化

- **集計**を用いた手法
  - 誰が多く支払ったか？
  - 誰が高いマージンの商品を買ったか？
  - 誰が継続的に購入しているか？

表に対する問い

SQLのような計算方法が適している

- **つながり**を用いた手法
  - 誰が強い影響力を周りに対して持っているか？
  - 誰が特定人物と同じ商品を購入しているか？
  - 誰が若い世代のコミュニティに属しているか？

グラフに対する問い

なにか別の計算方法が必要とされている！

グラフ分析エンジン「PGX」

グラフ分析エンジン「Oracle Labs PGX」

Oracle Labs（旧 Sun Labs）から OTN ライセンスで配布

- 商用利用はできませんが、無償で検証や開発ができます。

Parallel Graph Analytics (PGX)

OVERVIEW

PGX is a fast, parallel, in-memory graph analytic framework. Using PGX, the users can load up their graphs into main-memory, run various graph algorithms on them very efficiently, explore their results, and export them back into the file system.

For more information see Project Site

TEAM MEMBERS

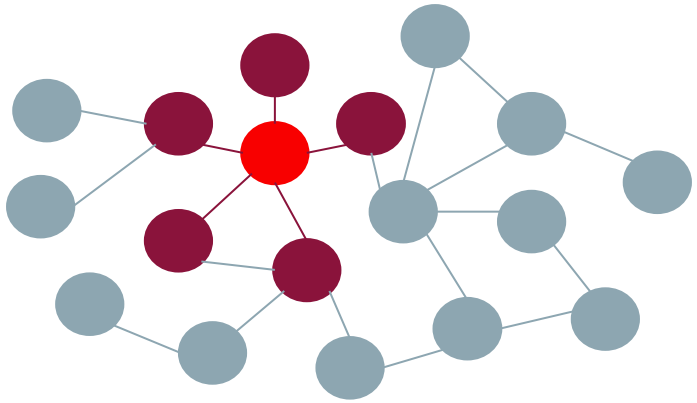

Sungpack Hong


Hassan Chafi

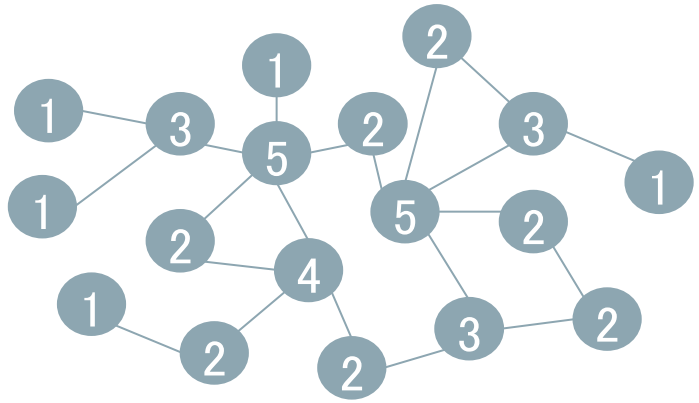

Felix Kaser

グラフ分析における2種類の処理

- 参照系処理
  - あるノードの周辺ノードの参照
  - プロパティ・パスによる探索
  - パターン・マッチング
  - サブグラフの抽出



- 演算系処理
  - コンポーネントやコミュニティの検出
  - コミュニティ構造の評価
  - ランキングとウォーキング
  - 経路探索

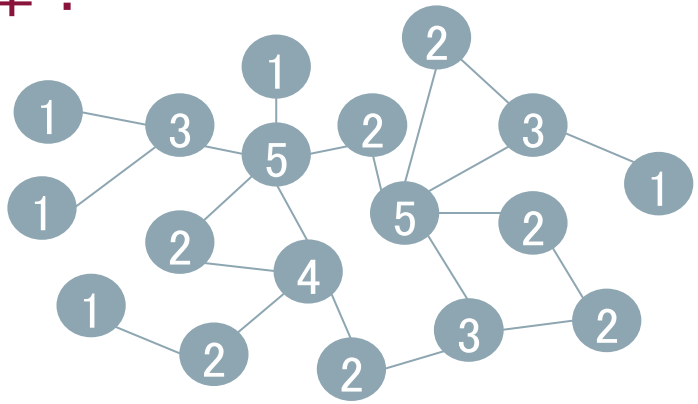


PGX のコンセプト

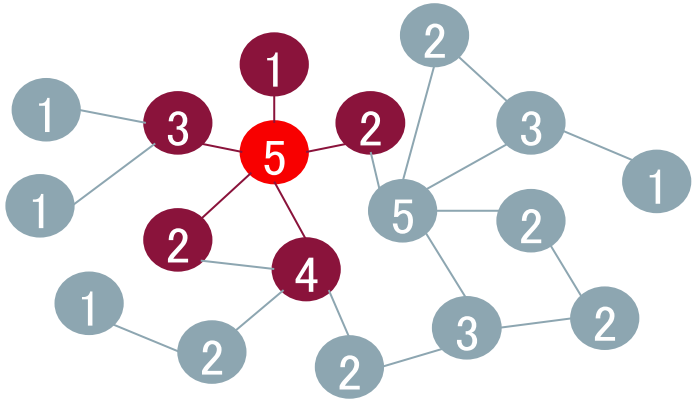
- 参照系処理と演算系処理の双方、これらを組み合わせた処理に対応
  - グラフのクエリによる参照（PGQL）
  - グラフへのアルゴリズムの適用
  - グラフの変換（Mutation）

「あるノードから3ステップ以内にあるページランクの高いノード」を探す、など

演算！

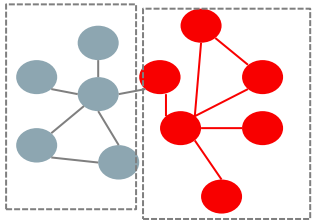


参照！



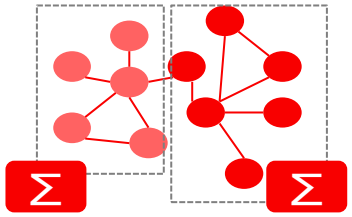
演算系処理 — 定義済みの35のアルゴリズム

コンポーネントやコミュニティの検出



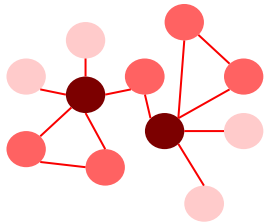
強連結成分分解
（Tarjan法、Kosaraju法）
弱連結コンポーネント
ラベル伝搬法

コミュニティ構造の評価



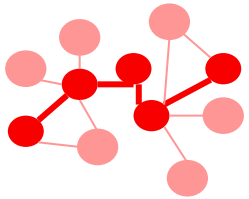
コンダクタンス
モジュール性
クラスタ係数
トライアングル数え上げ

ランキングとウォーキング



ページランク、
パーソナライズドページランク
中心媒介性、次数分布、次数中心性
近接中心性、固有ベクトル中心性
HITS、ランダムウォークサンプリング

経路探索



幅優先探索、ダイクストラ法
双方向ダイクストラ法
ベルマン・フォード法

その他

リンク予測（SALSA）

参照系処理 — クエリ言語によるグラフの参照

- Neo4j Cypher に似たクエリ言語「PGQL」によってグラフを参照可能
- グラフのパターンを直感的に記述できる： (a)-[likes]->(b)<-[likes]-(c)

「年齢がマリオより2歳下より若く、マリオとルイージが共に好いている
全ての人物の名前と年齢を、名前順に教えてください」

```
SELECT person.name, person.age
WHERE
  (m WITH name='Mario') -[WITH type='likes']-> (person),
  (l WITH name='Luigi') -[WITH type='likes']-> (person),
  person.age < m.age - 2
ORDER BY person.name
```

- バージョン 0.9 の仕様書はこちら

グラフ・データベースが最適？

Graph Analysis – Do We Have to Reinvent the Wheel?

Adam Welc
Oracle Labs
adam.welc@oracle.com

Raghavan Raman
Oracle Labs
raghavan.raman@oracle.com

Zhe Wu
Oracle
alan.wu@oracle.com

Sungpack Hong
Oracle Labs
sungpack.hong@oracle.com

Hassan Chafi
Oracle Labs
hassan.chafi@oracle.com

Jay Banerjee
Oracle
jayanta.banerjee@oracle.com

ABSTRACT

The problem of efficiently analyzing graphs of various shapes and sizes has been recently enjoying an increased level of attention both in the academia and in the industry. This trend prompted creation of specialized graph databases that have been rapidly gaining popularity of late. In this paper we argue that there exist alternatives to graph databases, providing competitive or superior performance, that do not require replacement of the entire existing storage infrastructure by the companies wishing to deploy them.

ing or exceeding that of the dedicated graph databases. We will also demonstrate that even larger performance advantage over graph databases can be achieved using in-memory graph analysis engines that can utilize relational database as the storage solution.

One of the prime examples of the graph database technology which we use for a limited performance study presented in this paper is Neo4j [11], "The World's Leading Graph Database" [1] - this claim backed up by a multitude of customers [10] and descriptions of extremely favorable comparisons with solutions based on the relational database

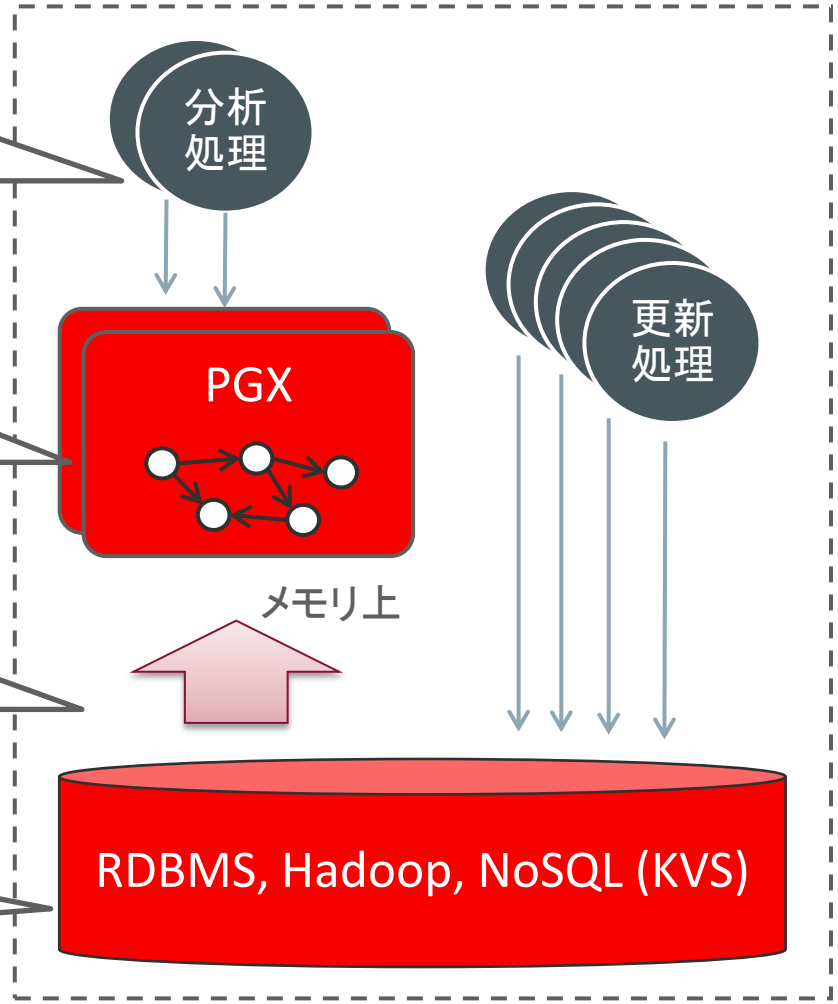
インメモリ解析 ＋ 永続化ストレージ

グラフ用クエリ言語 PGQL や、定義済みグラフ分析アルゴリズムを API から使用可能
（ただし、PGQL は 12.2.0.1 では未サポート）

Oracle Labs で開発された高速なインメモリのグラフ分析エンジン

データを並列処理で読み込み
ノードとエッジとしてメモリ上に展開

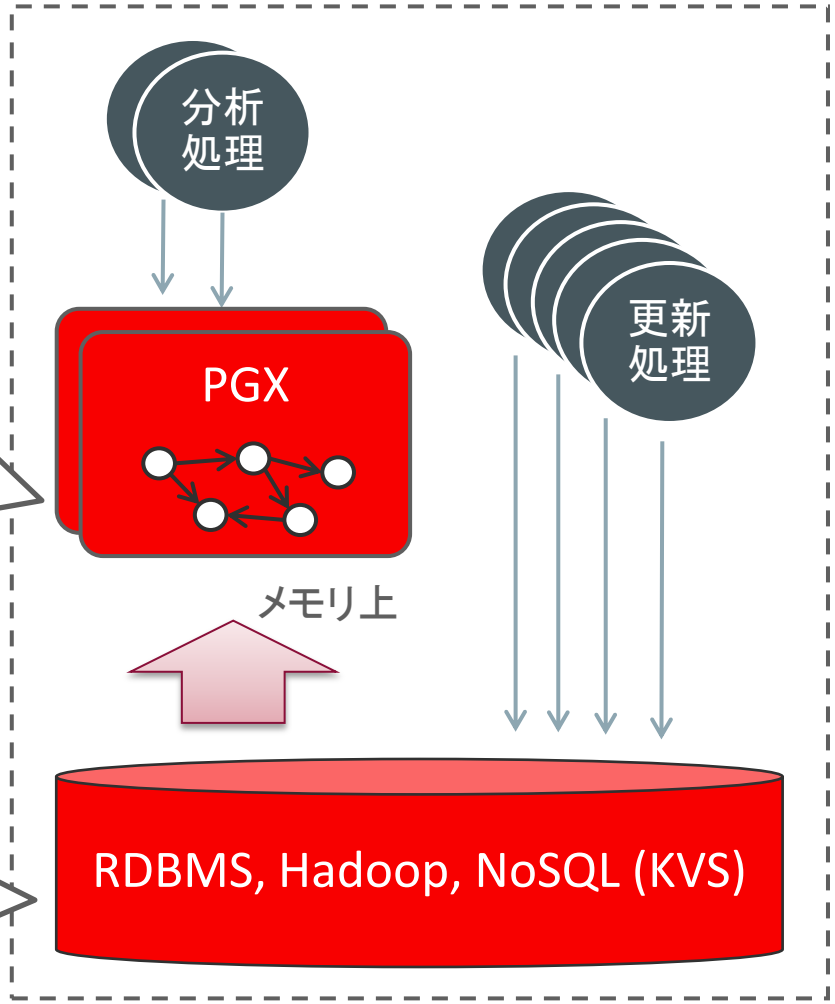
トランザクションはデータベースやHadoopで処理



今回ご紹介する PGX とデータベース機能

1. 今回は、より新しい技術を紹介するため、グラフ用クエリ言語「PGQL」をサポートしている Oracle Labs の PGX 1.2.1 を用いてグラフ分析を紹介します。(PGX 1.2.1 は OTN よりダウンロード可能ですが、現時点では 12.2.0.1 との連携はできません。)

2. さらに、データベース 12.2.0.1 のプロパティ・グラフ機能を使う際に、どのようにグラフを作成、データを追加、グラフをPGX にロードするかを紹介します。



使ってみる

PGX のセットアップ（Linux）

最新版の PGX release 1.2.1 をこちらからダウンロードします

JDK7（最新は 7u80）もこちらからダウンロードします

ファイルを展開します

```
$ cd $HOME/pgx   # ここにファイルを置くとします
$ unzip pgx-1.2.1-otn-linux-x86-64bit.zip -d pgx-1.2.1
$ tar xvzf jdk-7u80-linux-x64.gz
```

環境変数とエイリアスを設定します

```
$ export JAVA_HOME=/home/oracle/pgx/jdk1.7.0_80
$ export PATH=$JAVA_HOME/bin:$PATH
$ export PGX_HOME=$HOME/pgx/pgx-1.2.1
$ alias pgx='$PGX_HOME/bin/pgx'
```

参考　http://docs.oracle.com/cd/E56133_01/installation.html

PGX のセットアップ（Mac）

最新版の PGX release 1.2.1 をこちらからダウンロードします

JDK7（最新は 7u80）もこちらからダウンロードしてインストールします

ファイルを展開します

```
$ cd $HOME/pgx   # ここにファイルを置くとします
$ unzip pgx-1.2.1-otn-mac-x86-64bit.zip -d pgx-1.2.1
```

環境変数とエイリアスを設定します

```
$ export ¥
  JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.7.0_80.jdk/Contents/Home/
$ export PATH=$JAVA_HOME/bin:$PATH
$ export PGX_HOME=$HOME/pgx/pgx-1.2.1
$ alias pgx='$PGX_HOME/bin/pgx'
```

参考　http://docs.oracle.com/cd/E56133_01/installation.html

PGX シェルの起動

PGXシェルを起動します

```
$ pgx
..
pgx>
```

PGXシェルを閉じるとき

```
pgx> :exit
```

PGXシェルの誤入力でエラーが発生した際はヒストリーをクリアしてください

```
pgx> :clear
```

グラフの読み込み

PGXシェルを起動します

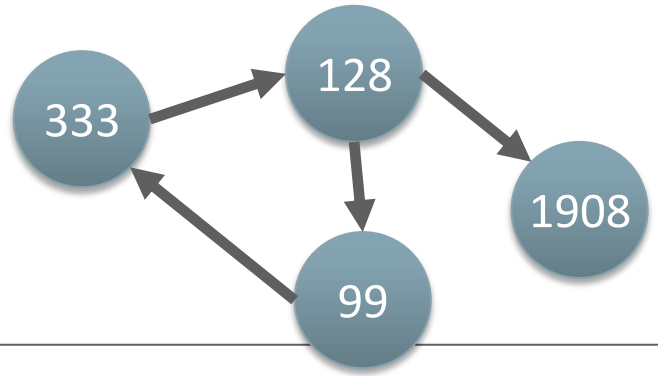
```
$ cd $PGX_HOME
$ pgx
```

サンプルのデータをロードした後、全てのエッジを参照します

```
G = session.readGraphWithProperties("examples/graphs/sample.adj.json")
```

```
G.queryPgql(" SELECT x.id(), y.id() WHERE (x)->(y) ").print()
```

```
                x.id()          y.id()
==========================================
                333             128
                128             1908
                128             99
                99              333
```



ヒーローたちの共演

Marvel ヒーローのグラフ

- Marvel のコミックには
  多くのキャラクターが登場
- さらに、ヒーロー達が集まって敵と
  戦ったり、他のコミックに友情出演
  することが多々ある
- そこでキャラクターの共演を
  グラフにして解析する

※ 出典 http://www.oracle.com/us/theavengers
/avengers-main-body-1571184.html

（壁紙もダウンロードできます）

データの入手

Exposedata.com (http://exposedata.com/marvel/) のウェブ・アーカイブの「Hero Social Network Data」からCSVデータを入手します

このデータは、キャラクターをノードとして、コミックの同じ号に登場しているキャラクター同士をカンマ区切りで対にしただけのものです

```
$ more hero-network.csv

"IRON MAN/TONY STARK ","CHAIN"
"IRON MAN/TONY STARK ","SPIDER-WOMAN II/JULI"
"IRON MAN/TONY STARK ","CARPENTER, RACHEL"
"CARPENTER, RACHEL","CHAIN"
"CARPENTER, RACHEL","SPIDER-WOMAN II/JULI"
..
```

グラフのロード

グラフのロードのための情報をJSONで記述します

```
$ vi hero-network.csv.json
{
  "uri": "hero-network.csv"
, "format": "edge_list"
, "node_id_type": "string"
, "separator": ","
}
```

PGXシェルから上のファイルを指定してグラフをロードします
（無向グラフとして読み込んでいます）

```
$ pgx
```

```
G = session.readGraphWithProperties("hero/hero-network.csv.json").undirect()
```

グラフのロード

キャラクター（ノード）を表示します

```
G.queryPgql(" SELECT x WHERE (x) ").print()
```

キャラクター（ノード）の数を表示します

```
G.queryPgql(" SELECT x WHERE (x) ").getNumResults()
```

```
==> 6426
```

一緒に登場するキャラクターの組（エッジ）の数を表示します

```
G.queryPgql(" SELECT x, y WHERE (x)->(y) ").getNumResults()
```

```
==> 792314
```

グラフのロード

ロードしたグラフに前処理を施します

```
G = G.simplify( ¥
      MultiEdges.REMOVE_MULTI_EDGES ¥
    , SelfEdges.REMOVE_SELF_EDGES ¥
    , TrivialVertices.REMOVE_TRIVIAL_VERTICES ¥
    , Mode.CREATE_COPY ¥
    , null ¥
    )
```

```
# 同じ2つのノードを結ぶ複数のエッジを除去
# 始点と終点が同じノードのエッジを除去
        # エッジを持たないノードを除去
        # 内部的にグラフのコピーを作成
        # グラフ名（任意）
```

一緒に登場するキャラクターの組（エッジ）の数を表示します

```
G.queryPgql(" SELECT x, y WHERE (x)->(y) ").getNumResults()
```

```
==> 334414
```

参照型処理の実行

PGQLは参照型処理（パターン・マッチング）に便利です

「アイアンマンとスパイダーマンとウルヴァリンの全てと
一緒に登場したことのあるキャラクターは？」

```
G.queryPgql(" ¥
  SELECT x ¥
  WHERE ¥
    (x)--(a @'IRON MAN/TONY STARK ') ¥
  , (x)--(b @'SPIDER-MAN/PETER PAR') ¥
  , (x)--(c @'WOLVERINE/LOGAN ') ¥
").print()
```

参照型処理の実行

結果の件数が多いので、一度、結果データを変数に渡して件数を数えます

```
Result = G.queryPgql(" ¥
 SELECT x ¥
  WHERE ¥
    (x)--(a @'IRON MAN/TONY STARK ') ¥
   , (x)--(b @'SPIDER-MAN/PETER PAR') ¥
   , (x)--(c @'WOLVERINE/LOGAN ') ¥
")
```

```
Result.getNumResults()
```

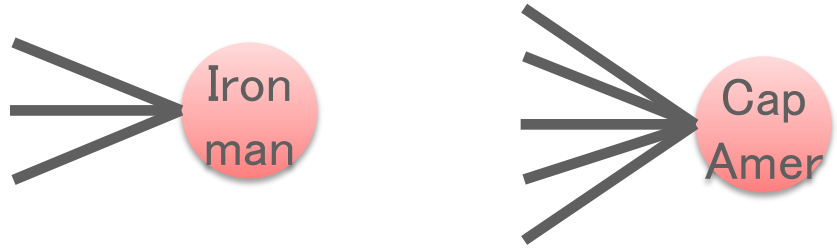
```
==> 504
```

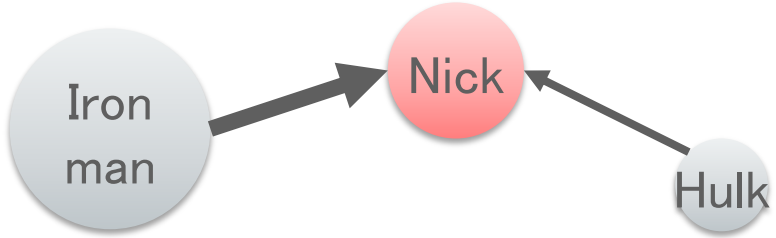
10件だけ見てみます

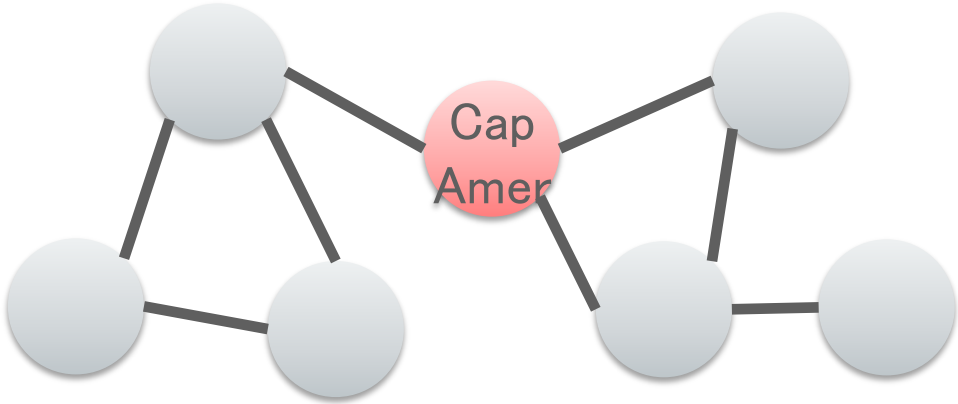
```
Result.print(5)
```

ヒーローの重要度の評価

- 次数中心性
  - より多くのエッジを持つノードがより重要
  - 友だちが多い人は重要人物？
- ページランク
  - 重要性はグラフ上で伝播するとすれば…
  - 重要なノードに繋がるノードは重要
  - 重要人物の友だちは重要人物？
- 媒介中心性
  - 多くのノード同士の最短経路にいると重要
  - その人がいないとみんなが困る？







演算型処理の実行

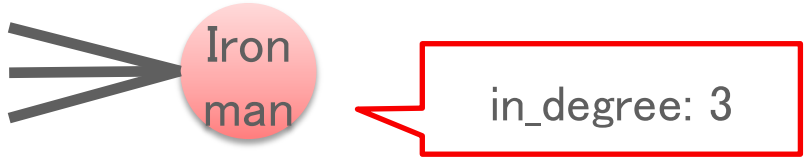
まず、各ノードの次数中心性を求めます

```
analyst.inDegreeCentrality(G)
```

```
==> Vertex Property named 'in_degree' of type integer belonging to graph ..
```

引数として、グラフ(G)が渡されています。計算量はノードの数 N、エッジ数を E としたとき、計算量は N に依存して増加するため O (N) になります。詳細はリファレンスに記載されています。

計算されたスコアは各ノードの
プロパティ in_degree に格納されます



演算型処理の実行

計算結果を参照するためには、PGQLを使います

```
G.queryPgql(" ¥
  SELECT n, n.in_degree WHERE (n) ¥
  ORDER BY n.in_degree DESC ¥
").print(5)
```

５人の重要なキャラクターが得られました！

```
n                                    n.degree
=============================================================
PgxVertex with ID CAPTAIN AMERICA          1906
PgxVertex with ID SPIDER-MAN/PETER PAR      1737
PgxVertex with ID IRON MAN/TONY STARK       1522
PgxVertex with ID THING/BENJAMIN J. GR      1416
PgxVertex with ID MR. FANTASTIC/REED R      1379
```

演算型処理の実行

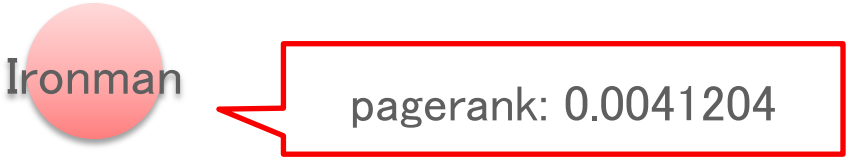
次に、各ノードのページランクを求めます

```
analyst.pagerank(G, 0.0001, 0.85, 100)
```

```
==> Vertex Property named 'pagerank' of type double belonging to graph ..
```

引数として、グラフ（G）および3つのパラメータ（許容される最大エラー値、ダンピング・ファクター、計算の反復回数）が渡されています。計算量は反復回数を k として O (k*E) になります。詳細はリファレンスに記載されています。

計算されたスコアは各ノードの
プロパティ pagerank に格納されます



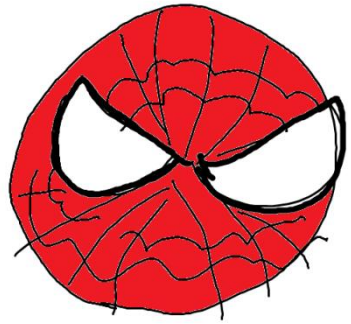
演算型処理の実行

計算結果を参照するためには、PGQLを使います

```
G.queryPgql(" ¥
  SELECT n, n.pagerank WHERE (n) ¥
  ORDER BY n.pagerank DESC ¥
").print(5)
```

スパイダーマンが1位になりました！

```
n                                                  n.pagerank
==========================================================================
PgxVertex with ID SPIDER-MAN/PETER PAR             0.005295697226042497
PgxVertex with ID CAPTAIN AMERICA                  0.00512599614036231 5
PgxVertex with ID IRON MAN/TONY STARK              0.004120441575337775 5
PgxVertex with ID WOLVERINE/LOGAN                  0.003866957384352529 7
PgxVertex with ID THING/BENJAMIN J. GR             0.003682406047200 28
```



演算型処理の実行

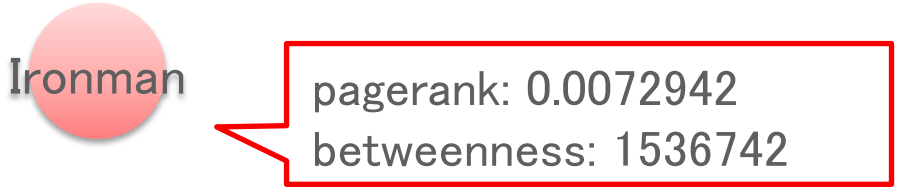
同様に、各ノードの媒介中心性も求めてみます

```
analyst.vertexBetweennessCentrality(G)
```

```
==> Vertex Property named 'betweenness' of type double belonging to graph ..
```

媒介中心性は、全ての頂点間の最短経路の中でその頂点が経路上にいる割合をスコアとしています。媒介中心性の計算量は O (N*E) であり、ページランクの計算量 O (k*E) と比較して計算コストが高く、やや実行時間が長くかかります。
詳細はリファレンスに記載されています。

計算されたスコアは各ノードの
プロパティ betweenness に格納されます



演算型処理の実行

計算結果を参照するためには、PGQLを使います

```
G.queryPgql(" ¥
  SELECT n, n.betweenness WHERE (n) ¥
  ORDER BY n.betweenness DESC ¥
").print(5)
```

トップ4まではページランクとも順位が一致しています

```
n                                                    n.betweenness
=========================================================================
PgxVertex with ID SPIDER-MAN/PETER PAR               3035278.742514678
PgxVertex with ID CAPTAIN AMERICA          2351348.199987796
PgxVertex with ID IRON MAN/TONY STARK                1536742.6554027186
PgxVertex with ID WOLVERINE/LOGAN          1473595.2238889225
PgxVertex with ID HAVOK/ALEX SUMMERS                 1471842.538198292
```

参照と演算の併用

演算型処理の結果（ここではページランクのスコア）を参照型処理（パターン・マッチング）の対象としてそのまま使えてしまうのがPGQLの便利なところ！

前出の検索をこのように変更してみます。

「アイアンマンとスパイダーマンとウルヴァリンの全てと一緒に登場したことのあるキャラクターのうちマイナーなキャラクターは？」

```
Result = G.queryPgql(" ¥
  SELECT x, x.pagerank ¥
  WHERE ¥
    (x)--(a @'IRON MAN/TONY STARK ') ¥
  , (x)--(b @'SPIDER-MAN/PETER PAR') ¥
  , (x)--(c @'WOLVERINE/LOGAN ') ¥
  ORDER BY x.pagerank ASC ¥
")
```

参照と演算の併用

こちらがマイナーなキャラクター10名です

```
Result.print(10)
```

```
x                                                    x.pagerank
==========================================================================
PgxVertex with ID TYCHO                                  5.611553899059807E-5
PgxVertex with ID ISBISA/DR. SANDERSON                   7.622033978468879E-5
PgxVertex with ID MASON, WANDA                           9.521036405479172E-5
PgxVertex with ID STORM, CHILI                       1.08794020519576 3E-4
PgxVertex with ID MAGNUM, MOSES                          1.098942967308010 1E-4
PgxVertex with ID CAT MAN/HORGAN                     1.206812449804566 8E-4
PgxVertex with ID BIRD MAN/HENRY HAWK                    1.206812449804566 8E-4
PgxVertex with ID TERRAXIA                           1.213747846059133 8E-4
PgxVertex with ID RAZORFIST III    1.234956960666 52E-4
PgxVertex with ID SVAROG                                 1.272774879016462 4E-4
```

参照と演算の併用

他にもこのようなクエリを書くことができます

「ドクターオクトパスと共演したマイナーなキャラクターの中で、スパイダーマン以外の重要キャラクターと共演したことのあるキャラクターは？」

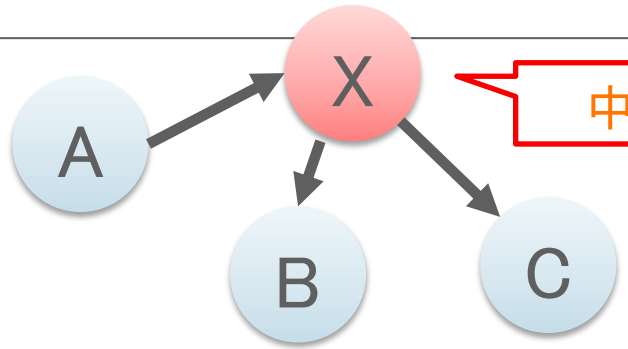
```
G.queryPgql(" ¥
  SELECT x, h ¥
  WHERE ¥
    (o @'DR. OCTOPUS/OTTO OCT')--(x)--(h WITH id()!='SPIDER-MAN/PETER PAR') ¥
  , x.pagerank < 0.0001 ¥
  , h.pagerank > 0.003 ¥
").print(5)
```

```
m                          h
==========================================================================
PgxVertex with ID PARKER, MARY  PgxVertex with ID MR. FANTASTIC/REED R
```

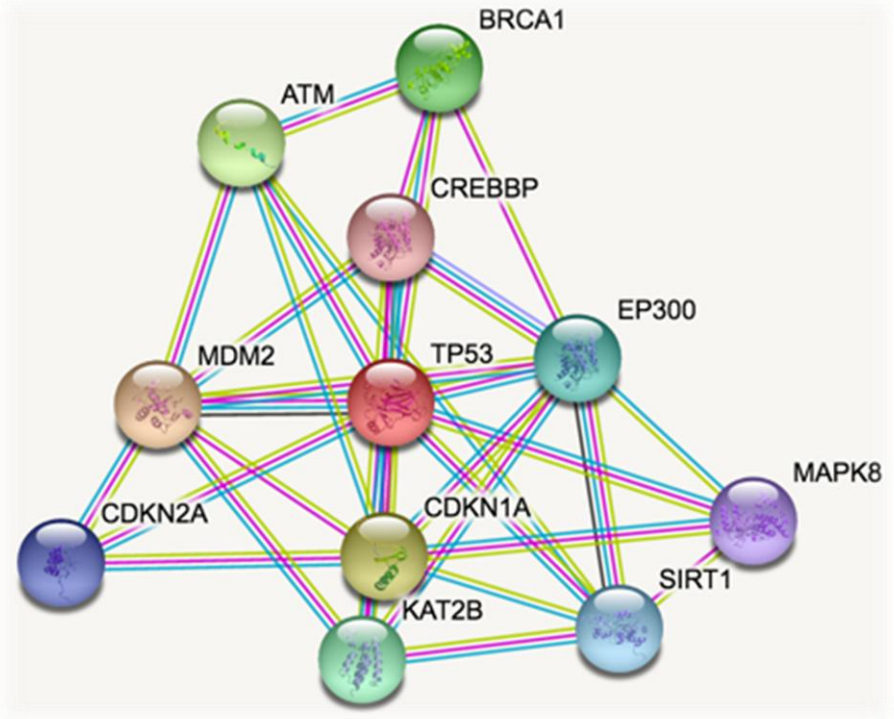
タンパク質のネットワークであれば…

「タンパク質 A、B、C の全てと相互作用があるタンパク質のうちネットワーク上の重要性が低いタンパク質（ロングテールの標的の可能性がある）は？」

```
Res = G.queryPgql(" ¥
  SELECT x, x.pagerank ¥
  WHERE ¥
    (x)--(a @'Protein A') ¥
  , (x)--(b @'Protein B') ¥
  , (x)--(c @'Protein C') ¥
  ORDER BY x.betweenness ASC ¥
")
```



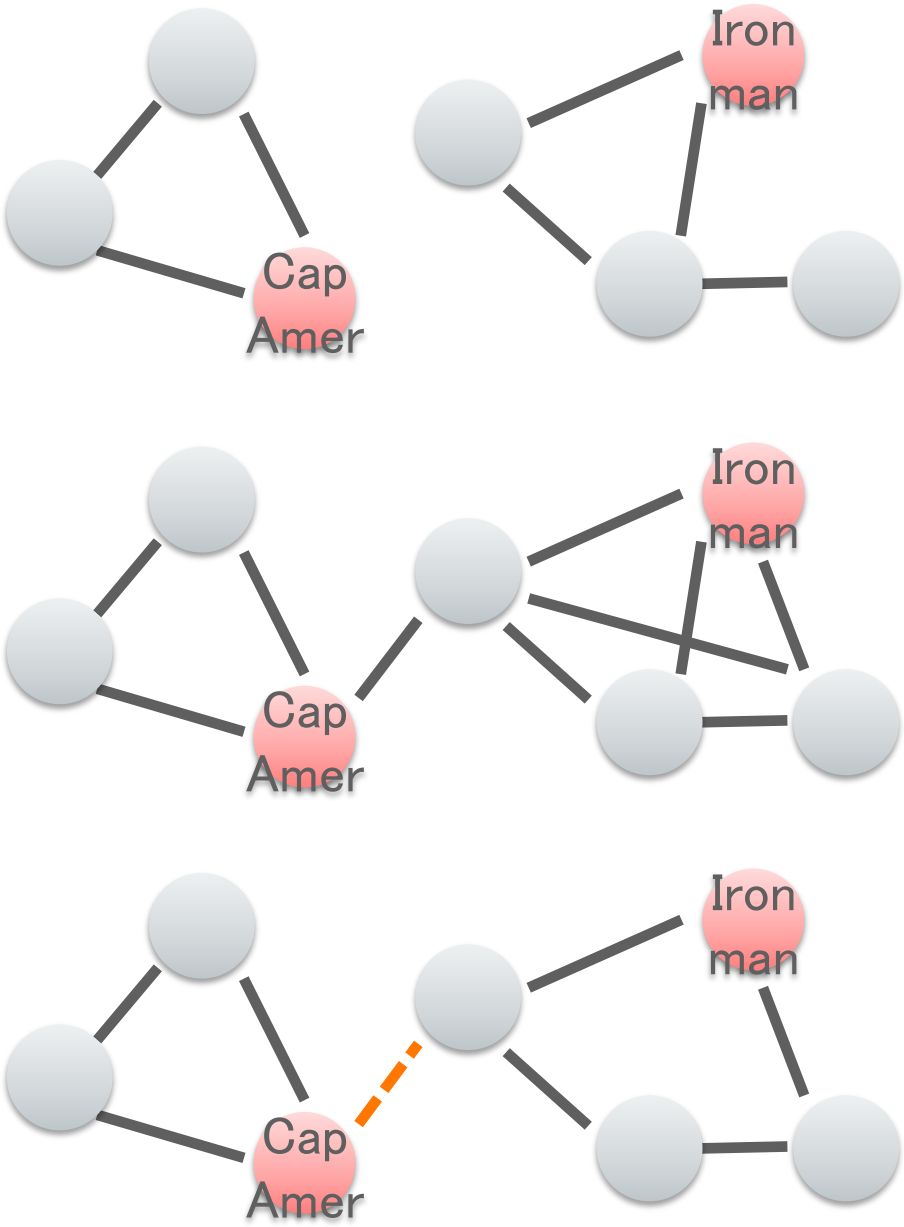
中心媒介性：0.00129（低）



※ 出典：STRING Database（http://string-db.org/）

コミュニティの抽出

- 連結成分
  - 接続されたノードの集合
  - 当然ながらコミュニティとして分割
- クリーク
  - 全てのノードがお互いに接続されている
  - 皆が知り合いならばコミュニティとする
- 辺媒介中心性
  - そのエッジがないと最短経路がなくなる
  - 重要な経路から除去していく
  - 橋渡し役を見分けて分割する



乗換案内

データの入手

駅データ.jp から、駅データ、接続駅データ、路線データを入手します。

駅データはノードとそのプロパティです。

```
station_cd,station_g_cd,station_name, ..
1110101,1110101,函館, ..
1110102,1110102,五稜郭, ..
1110103,1110103,桔梗, ..
```

接続駅データはエッジです。

```
line_cd,station_cd1,station_cd2
1002,100201,100202
1002,100202,100203
1002,100203,100204
```

路線データはエッジのプロパティです。

```
line_cd,company_cd,line_name,line_name_k, ..
1001,3,中央新幹線,チュウオウシンカンセン, ..
1002,3,東海道新幹線,トウカイドウシンカンセン, ..
1003,4,山陽新幹線,サンヨウシンカンセン, ..
```

データの加工

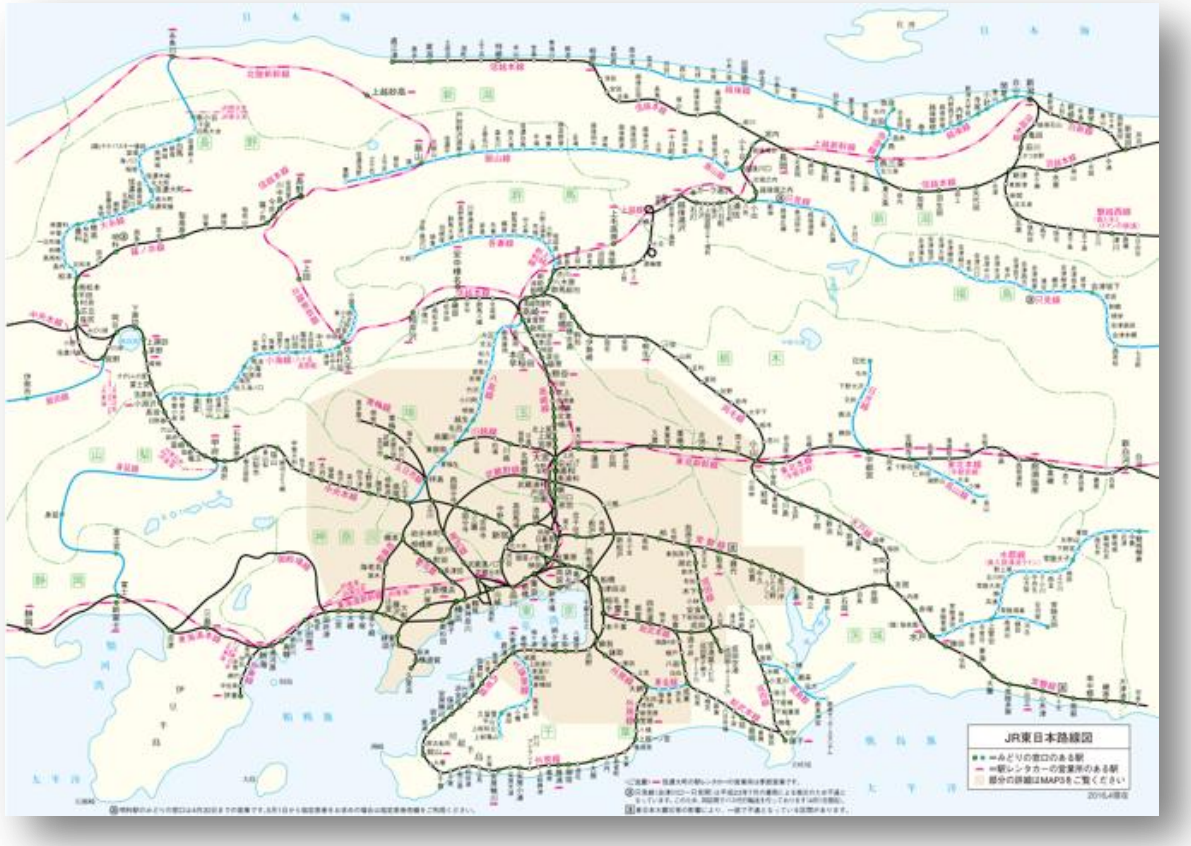
データを Edge List 形式に加工します。

```
$ sort -t ',' -k 1 station20160401free.csv > station.csv
$ sort -t ',' -k 1 line20160402free.csv > line.csv
$ sort -t ',' -k 2 join20160401.csv > join01.csv
$ join -t ',' join01.csv station.csv -1 2 -2 1 -o 1.1,2.2,1.3 > join02.csv
$ sort -t ',' -k 3 join02.csv > join03.csv
$ join -t ',' join03.csv station.csv -1 3 -2 1 -o 1.1,1.2,2.2 > join04.csv
$ sort -t ',' -k 1 join04.csv > join05.csv
$ join -t ',' join05.csv line.csv -1 1 -2 1 -o 2.3,1.2,1.3 > join06.csv
$ cat station20160401free.csv | ¥
  awk -v FS=',' -v OFS='¥t' '{if (NR != 1) print $2, "*", $1, $3}' >> rail.tsv
$ cat join06.csv | ¥
  awk -v FS=',' -v OFS='¥t' '{print $2, $3, $1, "1"}' >> rail.tsv
$ cat join06.csv | ¥
  awk -v FS=',' -v OFS='¥t' '{print $3, $2, $1, "1"}' >> rail.tsv
```

グラフのロード

グラフのロードのための情報をJSONで記述します

```
$ vi rail.tsv.json
{
  "uri": "rail.tsv"
, "format": "edge_list"
, "node_id_type": "integer"
, "vertex_props":[
    {"name":"cd", "type":"integer"}
  , {"name":"name", "type":"string"}
  ]
, "edge_props":[
    {"name":"name", "type":"string"}
  , {"name":"score", "type":"double"}
  ]
, "separator": "¥t"
}
```



※ 出典 JR東日本ウェブサイト – http://www.jreast.co.jp/map/pdf/kanto.pdf

参照型処理の実行

PGXシェルを開き、グラフを（有向グラフのまま）ロードします

```
G = session.readGraphWithProperties("rail/rail.tsv.json")
```

外苑前のとなりの駅と路線

```
G.queryPgql(" ¥
  SELECT x.name, r.name, y.name WHERE (x)-[r]->(y), x.name='外苑前' ¥
").print()
```

外苑前から3ホップの駅

```
G.queryPgql(" ¥
  SELECT y.name, COUNT(*) AS cnt WHERE (x)-->()-->()-->(y), x.name='外苑前' ¥
  GROUP BY y.name ORDER BY cnt DESC ¥
").print()
```

重要度の高い駅は？

各駅の次数中心性、ページランク、媒介中心性も求めてみます。

前処理として、グラフを無向グラフに変換後、重複しているエッジを削除します。

```
G = G.undirect()
G = G.simplify( ¥
      MultiEdges.REMOVE_MULTI_EDGES ¥
    , SelfEdges.REMOVE_SELF_EDGES ¥
    , TrivialVertices.REMOVE_TRIVIAL_VERTICES ¥
    , Mode.CREATE_COPY ¥
    , null ¥
    )
```

```
analyst.inDegreeCentrality(G)
analyst.pagerank(G, 0.0001, 0.85, 100)
analyst.vertexBetweennessCentrality(G)
```

※ 出典 (ライセンス: CC BY-SA 4.0) https://ja.wikipedia.org/wiki/新宿駅#/media/File:JR_Shinjuku_Miraina_TowerB.JPG

重要度の高い駅は？

次数中心性の高い駅を参照します

```
G.queryPgql(" ¥
  SELECT n.name, n.in_degree ¥
  WHERE (n) ORDER BY n.in_degree DESC ¥
").print(10)
```

n.name	n.in_degree
横浜	14
新宿	13
池袋	13
西梅田	12
東京	11
品川	11
千葉	11

※ 出典 (ライセンス: CC BY 3.0) https://ja.wikipedia.org/wiki/横浜駅#/media/File:横浜駅西口方駅舎.jpg

重要度の高い駅は？

ページランクの高い駅を参照します

```
G.queryPgql(" ¥
  SELECT n.name, n.pagerank ¥
  WHERE (n) ORDER BY n.pagerank DESC ¥
").print(10)
```

```
n.name              n.pagerank
==========================================
横浜                4.9988298663078 69E-4
池袋                4.5247879537142 31E-4
新宿                4.2281017462182717E-4
名古屋    4.0836351929643747E-4
千葉                4.0212473821223 9E-4
西梅田    3.8060529208011 82E-4
三宮                3.6194387995749 44E-4
```

※ 出典（ライセンス：CC BY-SA 3.0）https://ja.wikipedia.org/wiki/池袋駅#/media/File:Ikebukuro_station_west_2012.JPG

重要度の高い駅は？

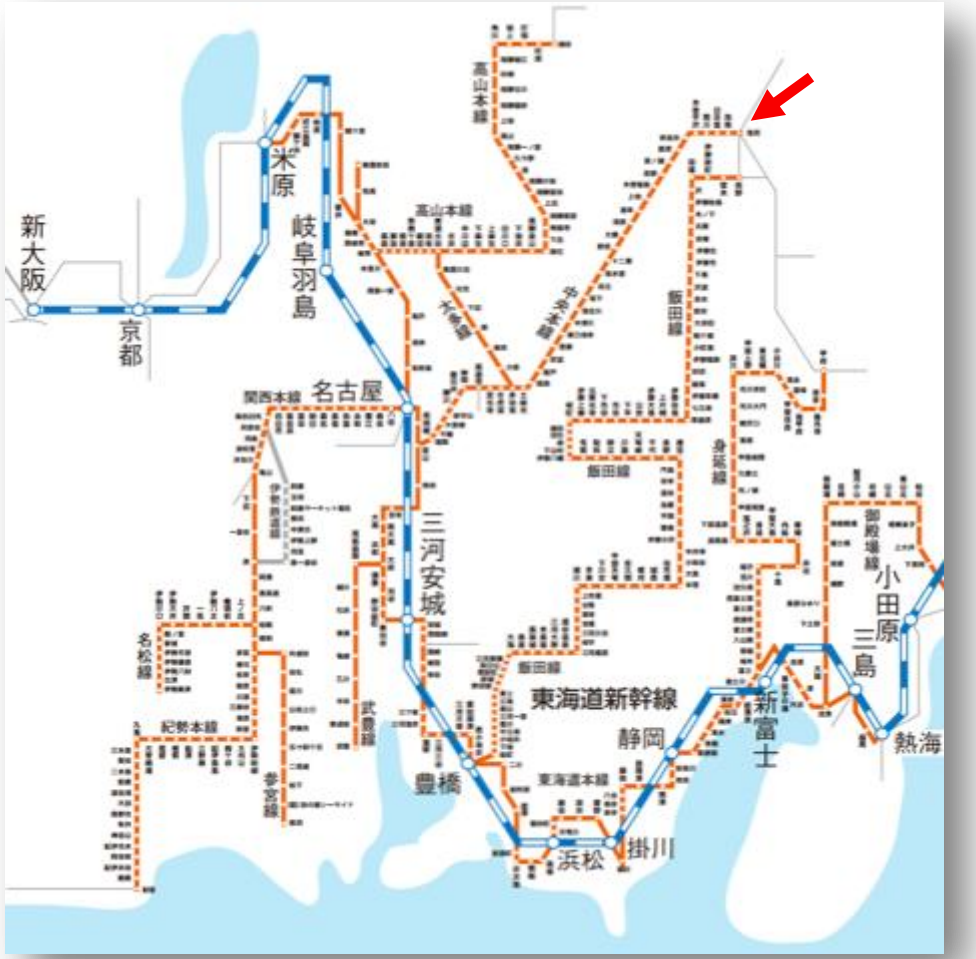
媒介中心性の高い駅を参照します

```
G.queryPgql(" ¥
  SELECT n.name, n.betweenness ¥
  WHERE (n) ORDER BY n.betweenness DESC ¥
").print(10)
```

n.name	n.betweenness
塩尻	3.505639812591153E7
恵那	3.44183248618756 4E7
多治見	3.441294421393956E7
土岐市	3.43489948618755 8E7
瑞浪	3.43467098618755 7E7
釜戸	3.43444968618755 7E7
武並	3.43423248618755 64E7

※ 出典（ライセンス：GFDL）https://ja.wikipedia.org/wiki/塩尻駅#/media/File:ShiojiriStnishi.jpg

重要度の高い駅は？



※ 出典 JR東海ウェブサイト http://railway.jr-central.co.jp/route-map/_pdf/entire.pdf , JR東日本ウェブサイト http://www.jreast.co.jp/map/pdf/kanto.pdf

演算型処理の実行（最短経路）

グラフを（有向グラフで）再度ロードします

```
G = session.readGraphWithProperties("rail/rail.tsv.json")
```

出発と到着の駅のノードIDを確認しておきます

```
G.queryPgql(" SELECT x.id() WHERE (x), x.name='外苑前' ").print()
G.queryPgql(" SELECT x.id() WHERE (x), x.name='新大阪' ").print()
G.queryPgql(" SELECT x.id() WHERE (x), x.name='長崎' ").print()
```

最短経路の計算

エッジのスコア（今回は距離などのデータがないため全て 1 としています）を
指定してダイクストラ法で最短経路を計算します。計算量は *O*（N log(N)）です。

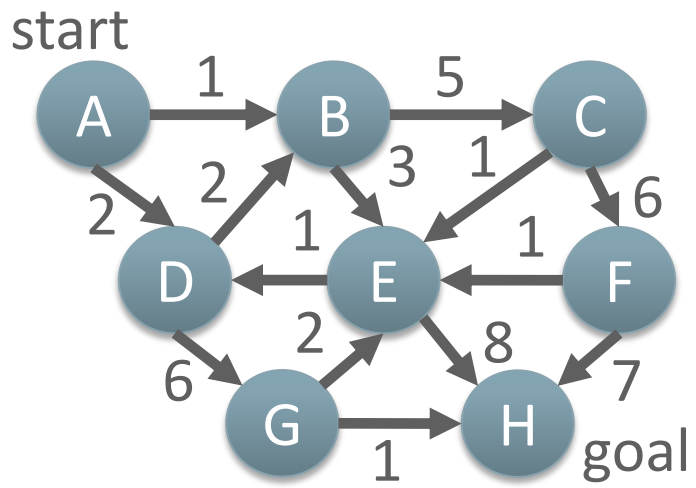
```
node1 = G.getVertex(2800117)
node2 = G.getVertex(1160213)
score = G.getEdgeProperty("score")
Path = analyst.shortestPathDijkstra(G, node1, node2, score)
```

最短で何ホップで到達できるかがわかります。

```
Path.getPathLengthWithHop()
```

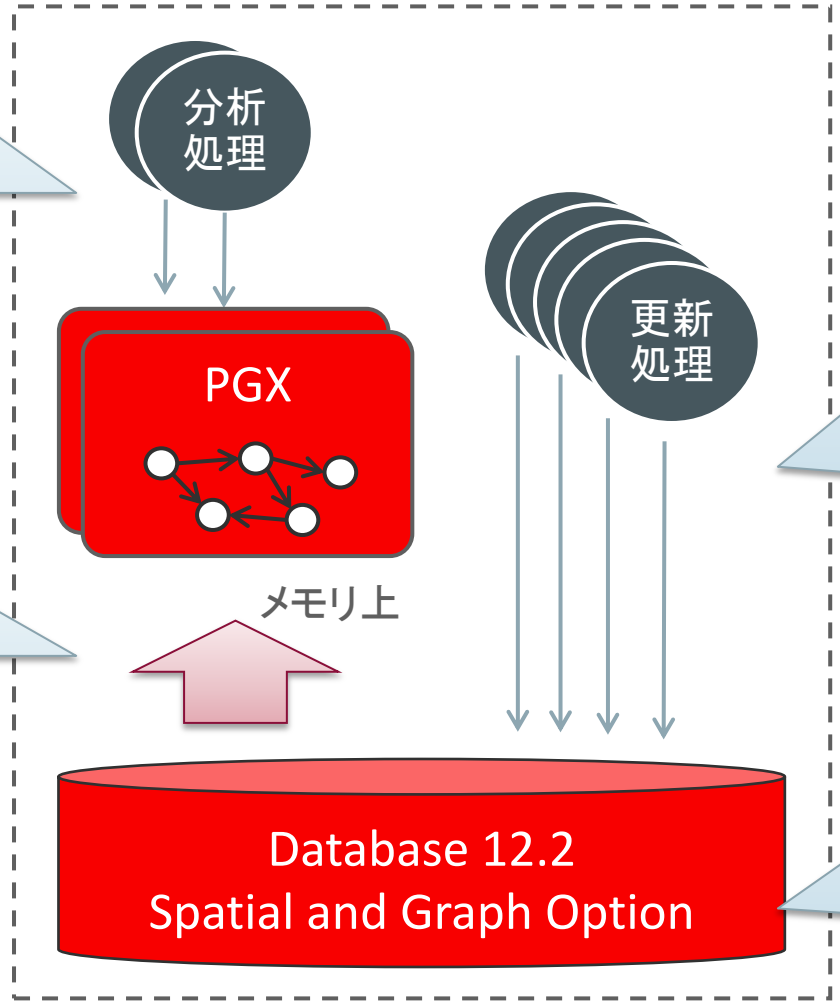
最短経路上の駅を確認することができます。

```
Path.getVertices()
```



データベースと PGX の連携

データベースに格納されたグラフ・データを分析



4. メモリ上でページランクなどのアルゴリズムを高速に実行（PGQL クエリは未サポート）

PGX Shell を使用

3. PGX Shell からデータベースに接続し、データをグラフに展開する

PGX Shell を使用

1. Java API を使用してグラフ用の表を作成しデータを格納する。トランザクションはデータベースで管理される

Java API を使用

2. 参考）データは表に格納されているため、SQL を使用して内容を参照することも可能

SQL を使用

Property Graph を使うための準備

初期化パラメータ（max_string_size）の変更

```
sqlplus / as sysdba
```

```
SQL> SHOW PARAMETER max_string_size
.. STANDARD
SQL> ALTER SYSTEM SET max_string_size='EXTENDED' scope=spfile;
SQL> SHUTDOWN IMMEDIATE
SQL> STARTUP UPGRADE
SQL> @?/rdbms/admin/utl32k.sql
SQL> SHUTDOWN IMMEDIATE
SQL> STARTUP
```

参考）2.3 Property Graph Schema Objects for Oracle Database

- In addition, the V column has a size of 15000, which requires the enabling of 32K VARCHAR (MAX_STRING_SIZE = EXTENDED).

Property Graph を使うための準備

データベース設定用のスクリプトを実行します。

```
cd $ORACLE_HOME/md/admin
sqlplus / as sysdba
```

```
@catopg.sql
```

データベースに一般ユーザーを作成します。

```
CREATE USER opg_user IDENTIFIED BY oracle;
GRANT connect, resource TO opg_user;
CONNECT opg_user/oracle;
```

Property Graph を使うための準備

PGX Shell を使用するために pgx コマンドに実行権限を与えます

```
cd $ORACLE_HOME/md/property_graph/pgx/bin
chmod +x pgx
pgx
```

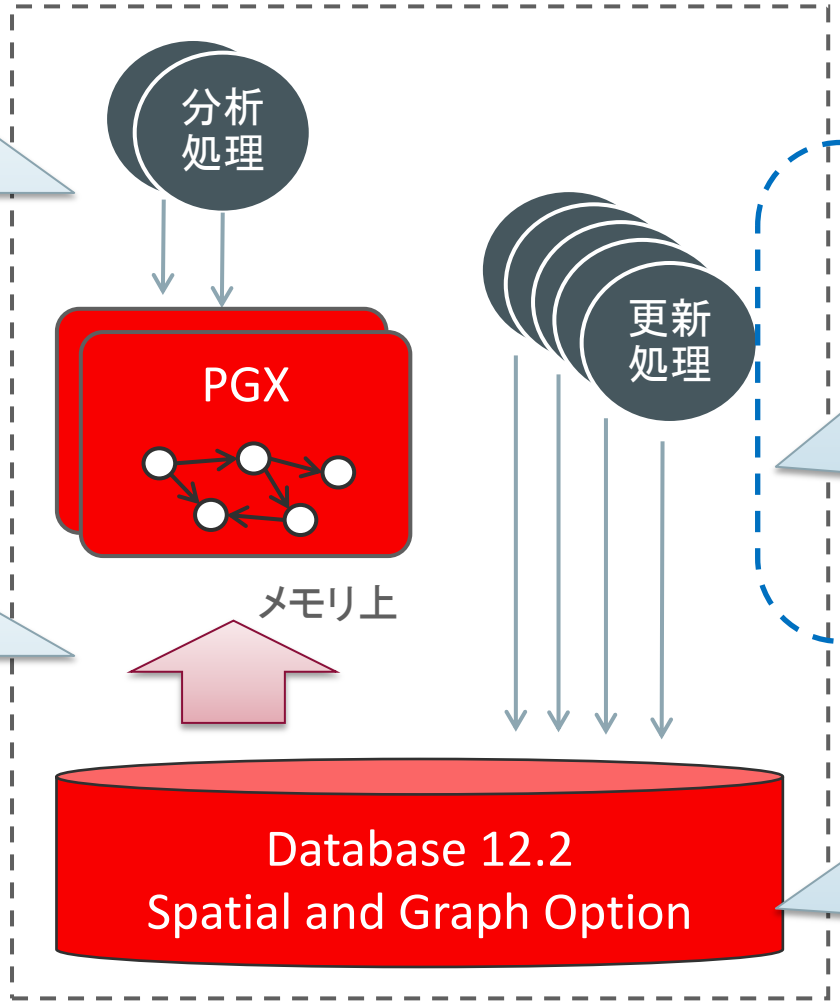
JAVA のパスを設定します。

```
export JAVA_HOME=${ORACLE_HOME}/jdk/jre/
export PATH=${JAVA_HOME}/bin:${PATH}
```

Java API を使用する操作は、ここでは Groovy を使用して対話的に実行します

```
cd $ORACLE_HOME/md/property_graph/dal/groovy
sh gremlin-opg-rdbms.sh
```

データベースに格納されたグラフ・データを分析



4. メモリ上でページランクなどのアルゴリズムを高速に実行（PGQL クエリは未サポート）

PGX Shell を使用

3. PGX Shell からデータベースに接続し、データをグラフに展開する

PGX Shell を使用

1. Java API を使用してグラフ用の表を作成しデータを格納する。トランザクションはデータベースで管理される

Java API を使用

2. 参考）データは表に格納されているため、SQL を使用して内容を参照することも可能

SQL を使用

グラフの作成

Java API を使用

データベースへの接続を作成します。

```
oracle =
  new Oracle("jdbc:oracle:thin:@127.0.0.1:1521:orcl","opg_user","oracle");
```

connections という名前のプロパティ・グラフを作成します。

```
opg = OraclePropertyGraph.getInstance(oracle, "connections");
```

新しいグラフにノードが格納されていないことを確認します。

```
opg.countVertices();
```

グラフに既にデータがロードされている場合は消去します。

```
opg.clearRepository();
```

グラフへの初期データのロード

Java API を使用

データベース内にはノード用とエッジ用の2つの表が作成されます。
このため、入力データも2つの CSV ファイルを用意しておく必要があります。

vi ../../data/connections.opv

```
1,name,1,Barack Obama,,
1,role,1,political authority,,
1,occupation,1,44th president of ..
1,country,1,United States,,
1,political party,1,Democratic,,
1,religion,1,Christianity,,
2,name,1,Beyonce,,
2,role,1,singer actress,,
2,country,1,United States,,
2,music genre,1,pop soul ,,
3,name,1,Charlie Rose,,
3,role,1,talk show host journalist,,
```

vi ../../data/connections.ope

```
1000,1,2,collaborates,weight,3,,1.0,
1001,1,3,collaborates,weight,3,,1.0,
1002,1,4,admires,weight,3,,1.0,
1003,1,5,admires,weight,3,,1.0,
1004,1,6,admires,weight,3,,1.0,
1005,1,7,admires,weight,3,,1.0,
1006,6,1,admires,weight,3,,1.0,
1007,6,7,collaborates,weight,3,,1.0,
1008,7,1,admires,weight,3,,1.0,
1009,7,6,collaborates,weight,3,,1.0,
1010,1,8,feuds,weight,3,,1000.0,
1011,8,1,feuds,weight,3,,1000.0,
```

グラフへの初期データのロード

Java API を使用

ローダーを作成します。

```
opgdl = OraclePropertyGraphDataLoader.getInstance();
```

入力ファイルを指定します。

```
vfile = "../../data/connections.opv";
efile = "../../data/connections.ope";
```

ロードします。4つめの引数は並列度です。

```
opgdl.loadData(opg, vfile, efile, 4/*dop*/);
```

ノードの数を確認します。

```
opg.countVertices();
```

```
==>78
```

グラフへのデータの追加

Java API を使用

ノード ID の最大値を確認しておきます。

```
opg.getMaxVertexID()
```

```
==>78
```

2つのノードを追加します。各ノードに任意の数のプロパティを追加することができます。

```
v1 = opg.addVertex(79l);
v1.setProperty("name", "John");
```

```
v2 = opg.addVertex(80l);
v2.setProperty("name", "Mary");
v2.setProperty("age", 30i);
```

グラフへのデータの追加

Java API を使用

ノードの数を確認します。

```
opg.countVertices();
```

```
==>80
```

さらに、1つのエッジを追加します。

```
opg.getMaxEdgeID()
```

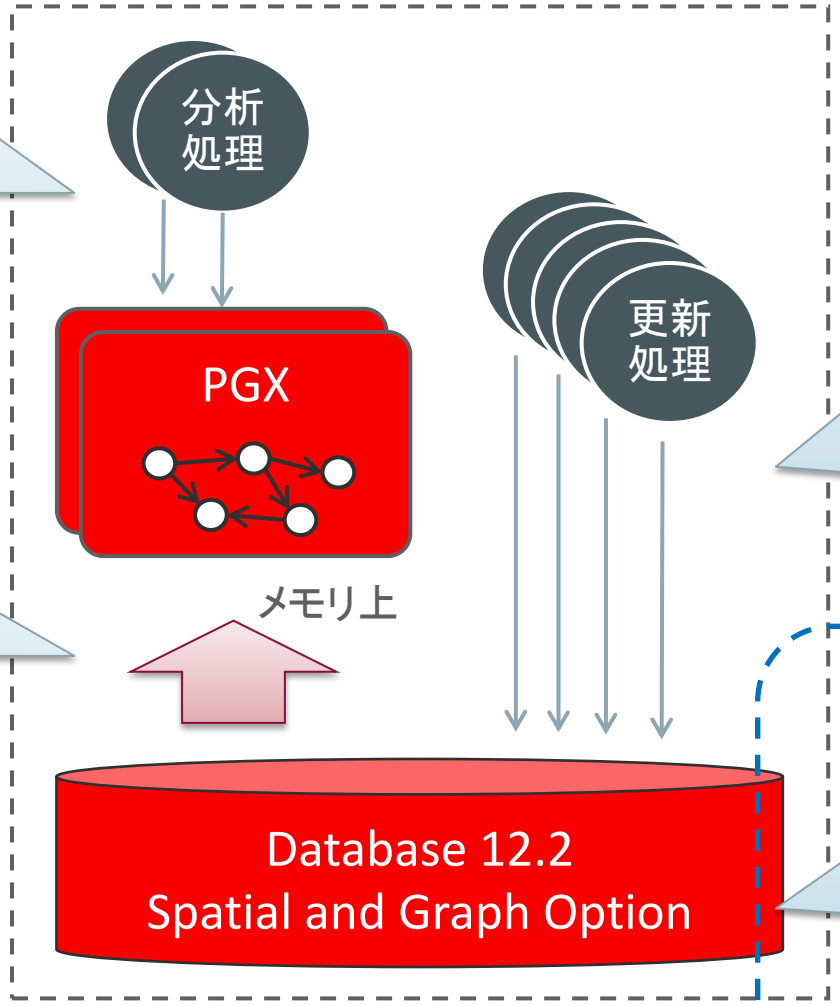
```
==>1163
```

```
e = opg.addEdge(1164l, v1, v2, "likes");
```

データベース上の操作なのでコミットが必要です。

```
opg.commit();
```

データベースに格納されたグラフ・データを分析



4. メモリ上でページランクなどのアルゴリズムを高速に実行（PGQL クエリは未サポート）

PGX Shell を使用

3. PGX Shell からデータベースに接続し、データをグラフに展開する

PGX Shell を使用

1. Java API を使用してグラフ用の表を作成しデータを格納する。トランザクションはデータベースで管理される

Java API を使用

2. 参考）データは表に格納されているため、SQL を使用して内容を参照することも可能

SQL を使用

参考）グラフのデータの参照

SQL を使用

グラフの実体は表なので、SQL でデータを参照することができます。

```
SELECT COUNT(DISTINCT vid) FROM connectionsVT$;
```

```
COUNT(DISTINCTVID)
------------------
                80
```

プロパティが表の値として格納されています。

```
SELECT vid, k, t, v FROM connectionsVT$ WHERE vid=80;
```

```
       VID K                  T V
---------- ---------- ---------- ----------
        80 name                1 Mary
        80 age                 2 30
```

参考）グラフのデータの参照

SQL を使用

同様にエッジも確認します。

```
SELECT COUNT(DISTINCT eid) FROM connectionsGE$;
```

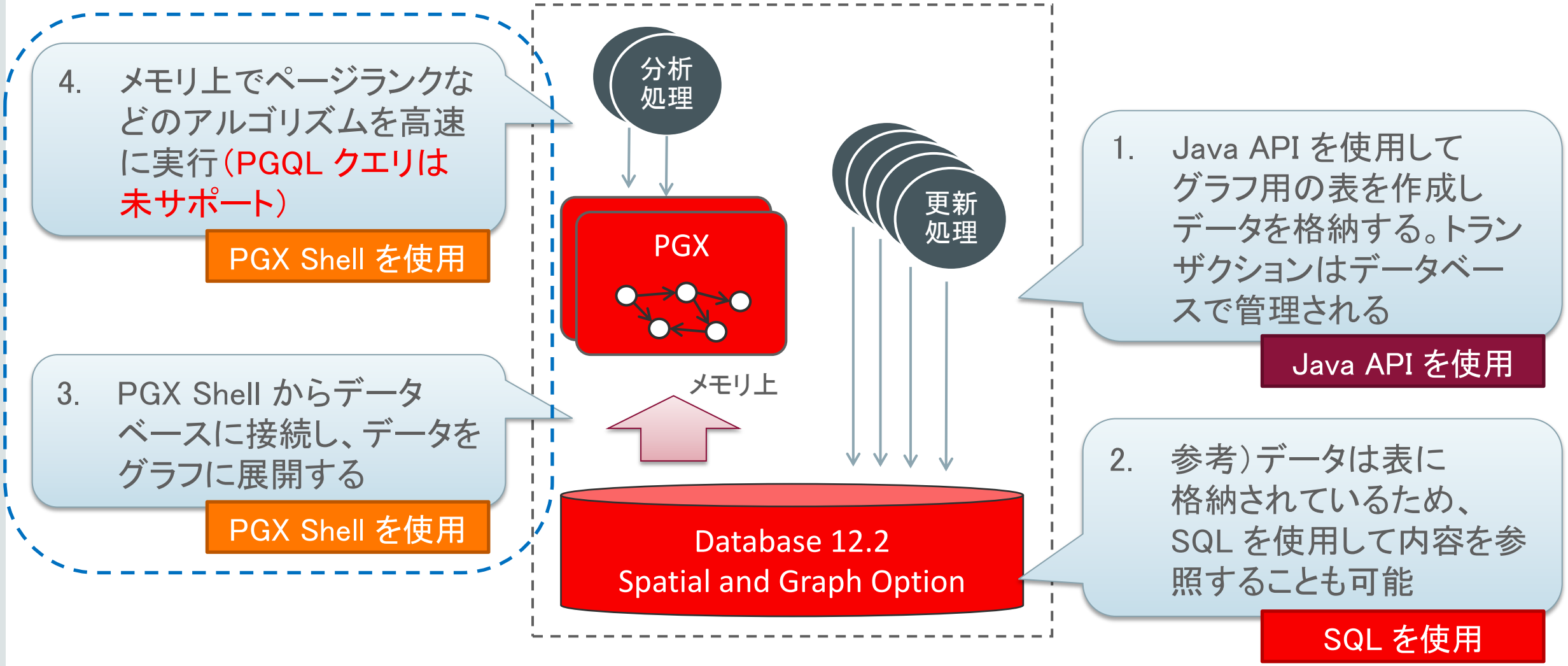
```
COUNT(DISTINCTEID)
------------------
               165
```

プロパティが表の値として格納されています。

```
SELECT eid, svid, dvid, el, k, t FROM connectionsGE$ WHERE eid=1001;
```

```
 EID       SVID       DVID       EL                 K                  T
---------- ---------- ---------- ------------------ ---------- ----------
      1001          1          3 collaborates       weight              3
```

データベースに格納されたグラフ・データを分析



DB から PGX へのデータのロード

PGX Shell を使用

グラフをロードするための定義ファイルを作成します。

```
$ vi /tmp/connections.json
```

```
{
  "db_engine":"RDBMS",
  "jdbc_url":"jdbc:oracle:thin:@127.0.0.1:1521:orcl",
  "username":"opg_user",
  "password":"oracle",
  "max_num_connections":8,
  "error_handling":{},
  "format":"pg",
  "name":"connections",
  "vertex_props":[{"name":"name","type":"string","default":"no_name"}],
  "edge_props":[{"name":"weight","type":"integer","default":"1"}],
  "loading":{"load_edge_label":false}
}
```

DB から PGX へのデータのロード

PGX Shell を使用

データを PGX にロードします。

```
G = session.readGraphWithProperties("/tmp/connections.json")
```

ノード数を確認します。

```
G.getNumVertices()
```

```
==> 80
```

分析アルゴリズムの実行

PGX Shell を使用

全てのノードの媒介中心性を計算します。

```
Result = analyst.vertexBetweennessCentrality(G)
```

媒介中心性の上位5つのノードを出力します。

```
Result.getTopKValues(5)
```

```
==> PgxVertex with ID 1=2225.6666666666665
==> PgxVertex with ID 2=1029.0
==> PgxVertex with ID 3=876.5
==> PgxVertex with ID 37=797.0
==> PgxVertex with ID 45=456.0
```

分析アルゴリズムの実行

PGX Shell を使用

ノードの名前（name プロパティ）を確認します。

```
G.getVertex(1l).getProperty("name")
```

```
==> Barack Obama
```

```
G.getVertex(2l).getProperty("name")
```

```
==> Beyonce
```

参考）PGQL がサポートされることで、以下のクエリで結果を得ることができます。

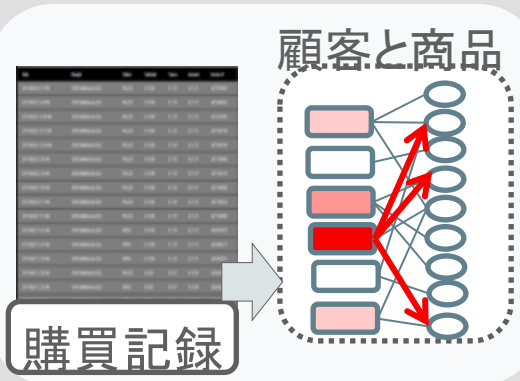
```
G.queryPgql(" ¥
  SELECT n.name, n.betweenness ¥
  WHERE (n) ORDER BY n.betweenness DESC ¥
").print(5)
```

最後に

グラフが利用されるビッグデータ課題

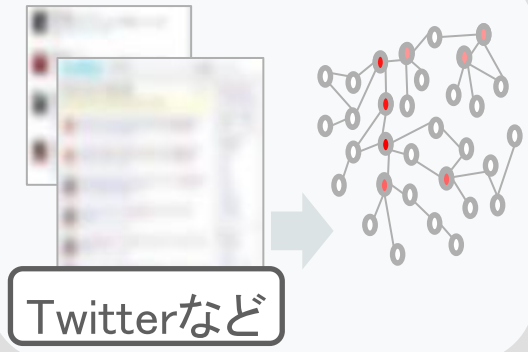
私と買い物が似ている人物によって購入された商品と、その商品と一緒に購入されている商品を教えてください

リコメンデーション



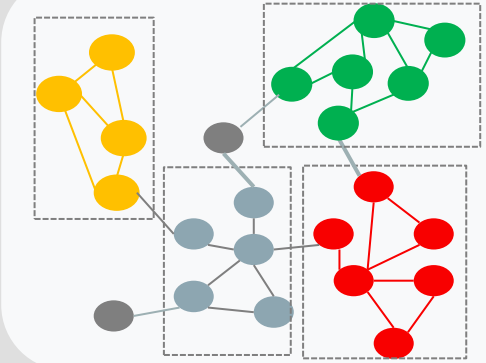
ソーシャル・ネットワーク上で中心的な役割を担っている人物を教えてください（マーケティング分析など）

影響力のある人物の特定



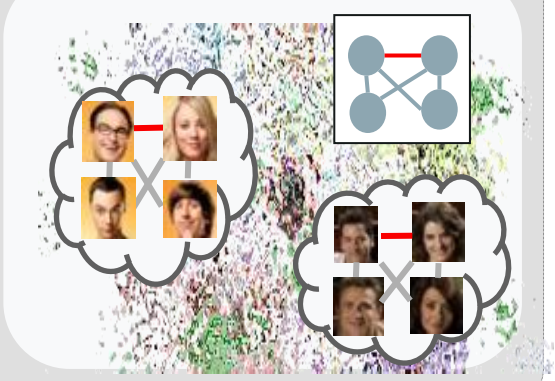
似通った人達やつながりのある人達のグループを教えてください（ターゲット・マーケティングなど）

コミュニティの検出



ある不正取引と同様の取引パターンがみられる全ての預金口座を探してください（不正検出や行動分析など）

パターン・マッチング



まとめ

- 12c R2 では**プロパティ・グラフ**機能が追加されます
- この機能は一般に**グラフ・データベース**と呼ばれるものです

- **グラフ・データベース**と**ネットワーク分析**はそれぞれ
  活用され始めているものの、これらの技術を組合わた
  プロパティ・グラフ機能の**可能性は未知**です！

- 是非、活用方法を一緒に検討させてください！

Agenda

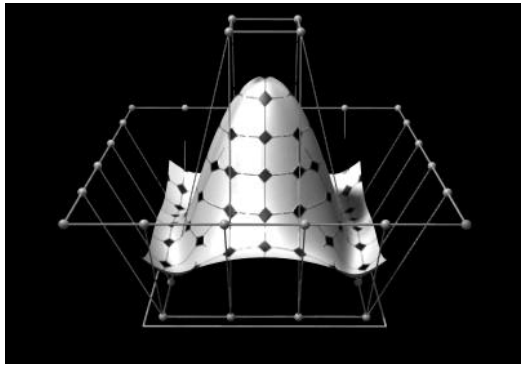
1. Spatial and Graph のご紹介
2. 新機能 プロパティグラフのご紹介
3. **その他のアップデート (Spatial/NDM/RDF)**
4. まとめ

Spatial and Graph 12.2 updates
Spatial

Spatial 12c R1 新機能まとめ

12c R1

- **Spatial Vector Acceleration**
- **NURBS曲線のサポート**
- **ラスターデータ機能の拡充**
- 3D機能の拡張(3D測地操作の近似処理, 3Dメタデータ用のビュー,関数の3D対応など)
- 点群(Point Cloud),不規則三角網機能の拡張
  - 点群の属性次元を含む検索、等高線の生成,DEMデータのラスターデータへの反映など
- 新規関数の追加,変更,性能改善(約36種(12.1.0.2含む))
- Web Feature Server コンソール



出典:wikiepdia

Spatial Vector Acceleration

12c R1

12cから追加された新規初期化パラメータ

空間処理が高速化
(クエリにより数倍～100倍高速化)

空間処理のパフォーマンス改良

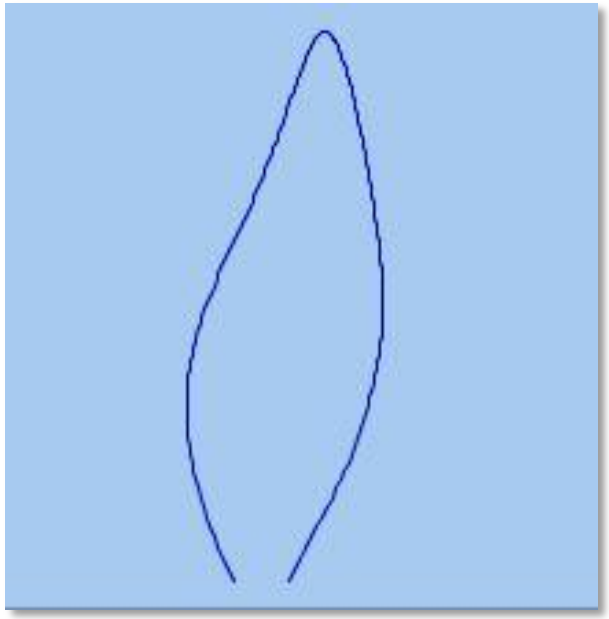
- Join: 50-100倍
- Touch: 50倍
- Contains, Overlaps: 50倍
- 複雑なマスク判定: 50倍

```
SQL>  show parameter spatial
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------
spatial_vector_acceleration          boolean     TRUE
```

NURBS曲線のサポート

12c R1

- NURBS曲線のサポートによりなめらかな曲線や曲面をより端的に表現できるようになりました。



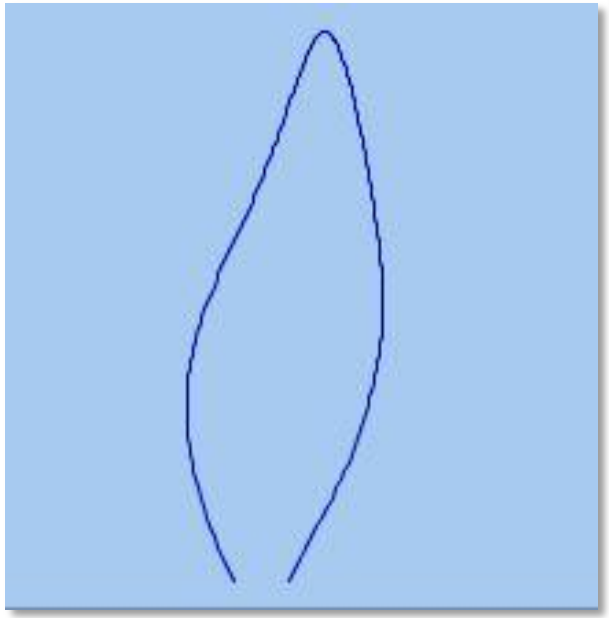
従来の表現

```
SDO_GEOMETRY(2002, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 2, 1),
 SDO_ORDINATE_ARRAY(0, 0,
  -.02912839, .059699523, -.05624374, .118211319, -.08139356, .175559751, -.10462535, .231769184,
  -.12598662, .286863981, -.14552488, .340868505, -.16328764, .39380712, -.17932241, .445704191,
  -.1936767, .496584079, -.20639802, .54647115, -.21753387, .595389767, -.22713177, .643364292,
  -.23523922, .690419091, -.24190374, .736578527, -.24717284, .781866962, -.25109401, .826308762,
  -.25371477, .869928288, -.25508264, .912749906, -.25524512, .954797979, -.25424971, .99609687,
  -.25214393, 1.03667094, -.24897529,  1.07654456, -.24479129, 1.11574209, -.23963945, 1.15428789,
  -.23356727, 1.19220633, -.22662227, 1.22952177, -.21885194, 1.26625857, -.21030381, 1.3024411,
  -.20102538, 1.33809372, -.19106416, 1.37324079, -.18046765, 1.40790668, -.16928338, 1.44211576,
  -.15755884, 1.47589238, -.14534154, 1.50926091, -.132679, 1.54224571, -.11961872, 1.57487115,
  -.10620822, 1.60716159, -.092495, 1.63914139, -.07852657, 1.67083492, -.06435044, 1.70226654,
  -.05001412, 1.73346062, -.03556511, 1.76444151, -.02105094, 1.79523359, -.0065191, 1.82586121,
   .007982896, 1.85634874, .022407535, 1.88672054, .036707311, 1.91700098, .050834714, 1.94721442,
   .064742236, 1.97738522, .078382506, 2.00753762, .091725045, 2.03768051,.104772175, 2.06779294,
   .117529987, 2.09785056, .130004572, 2.12782899, .14220202, 2.15770388, .154128423, 2.18745085,
   .165789872, 2.21704556, .177192457, 2.24646363, .188342269, 2.27568069, .1992454,  2.3046724,
   .20990794, 2.33341438, .22033598, 2.36188226, .23053561, 2.3900517, .240512923, 2.41789831,
   .250274008, 2.44539774, .259824957, 2.47252563, .269171861, 2.49925761, .27832081, 2.52556931,
   .287277896, 2.55143638, .296049209, 2.57683445, .30464084, 2.60173916, .31305888, 2.62612614,
   .321309421, 2.64997102, .329398552, 2.67324946, .337332365, 2.69593707, .345116951, 2.71800951,
  .352758401,  2.7394424, .360262805, 2.76021137, .367636255, 2.78029208, .374884841, 2.79966015,
  .382014654, 2.81829122, .389031786, 2.83616093, .395942326, 2.85324491, .402752367, 2.8695188,
  .409467999, 2.88495824, .416095312, 2.89953885, .422640398, 2.91323629, .429109348, 2.92602618,
  .435508253, 2.93788416, .441843203, 2.94878587, .448120289, 2.95870695, .454345602, 2.96762302,
  .460525234, 2.97550973, .466665275, 2.98234271, .472771816, 2.98809761, .478850948, 2.99275004,
  .484908761, 2.99627566, .490951348, 2.9986501,  .496984798, 2.999849,   .50301505,  2.999849,
  .509044541, 2.9986501, .515072205, 2.99627566, .521096823, 2.99275004, .527117177, 2.98809761,
  .53313205, 2.98234271, .539140223, 2.97550973, .545140477, 2.96762302, .551131595, 2.95870695,
  .557112359, 2.94878587, .56308155, 2.93788416, .56903795, 2.92602618, .574980341, 2.91323629,
  .580907505, 2.89953885, .586818223, 2.88495824, .592711277, 2.8695188, .59858545, 2.85324491,
  .604439523, 2.83616093, .610272278, 2.81829122, .616082496, 2.79966015, .621868959, 2.78029208,
  .62763045, 2.76021137, .63336575, 2.7394424, .639073641, 2.71800951, .644752905, 2.69593707,
  .650402323, 2.67324946, .656020678, 2.64997102, .661606751, 2.62612614, .667159324, 2.60173916,
  .672677178, 2.57683445, .678159097, 2.55143638, .683603861, 2.52556931, .689010252, 2.49925761,
  .694377052, 2.47252563, .699703043, 2.44539774, .704987007, 2.41789831, .710227725, 2.3900517,
  .71542398, 2.36188226, .720574553, 2.33341438, .725678226, 2.3046724, .730733781, 2.27568069,
  .735739999, 2.24646363, .740695663, 2.21704556, .745599554, 2.18745085, .750450454, 2.15770388,
  .755247146, 2.12782899, .75998841, 2.09785056, .764673028, 2.06779294, .769299783, 2.03768051,
  .773867456, 2.00753762, .778372773, 1.97738522, .782794569, 1.94721442, .787102465, 1.91700098,
  .791266007, 1.88672054, .795254739, 1.85634874, .799038208, 1.82586121, .802585958, 1.79523359,
  .805867536, 1.76444151, .808852487, 1.73346062, .811510356, 1.70226654, .813810689, 1.67083492,
  .815723031, 1.63914139, .817216927, 1.60716159, .818261924, 1.57487115, .818827566, 1.54224571,
  .8188834,   1.50926091, .818398969, 1.47589238, .817343821, 1.44211576, .8156875,   1.40790668,
  .813399552, 1.37324079, .810449523, 1.33809372, .806806957, 1.3024411,  .802441401, 1.26625857,
  .797322399, 1.22952177, .791419497, 1.19220633, .784702242, 1.15428789, .777140177, 1.11574209,
  .768702849, 1.07654456, .759359802, 1.03667094, .749080584, .99609687, .737834738, .954797979,
  .725591811, .912749906, .712321348, .869928288, .697992894, .826308762, .682575995, .781866962,
  .666040196, .736578527, .648355043, .690419091, .629490081, .643364292, .609414855, .595389767,
  .588098912, .54647115, .565511797, .496584079, .541623054, .445704191, .51640223, .39380712,
  .48981887, .340868505, .46184252, .286863981, .432442724,  .231769184, .401589029, .175559751,
  .36925098, .118211319, .335398121, .059699523, .3, 0))
```

NURBS曲線のサポート

12c R1

- NURBS曲線のサポートによりなめらかな曲線や曲面をより端的に表現できるようになりました。



新機能

```
SDO_GEOMETRY(
   2002,   NULL,   NULL,
   SDO_ELEM_INFO_ARRAY(1, 2, 3),
   SDO_ORDINATE_ARRAY
     (3,   7,
      0, 0, 1,   -0.5, 1, 1,  0.2, 2, 1,    0.5, 3.5, 1,
      0.8, 2, 1,  0.9, 1, 1,    0.3, 0, 1,
      11, 0, 0, 0, 0,  0.25, 0.5, 0.75, 1.0, 1.0, 1.0, 1.0)));
```

※NURBS曲線では曲線や曲面を次数、重みつき制御点、ノットベクトルの要素でコンパクトに表現できます

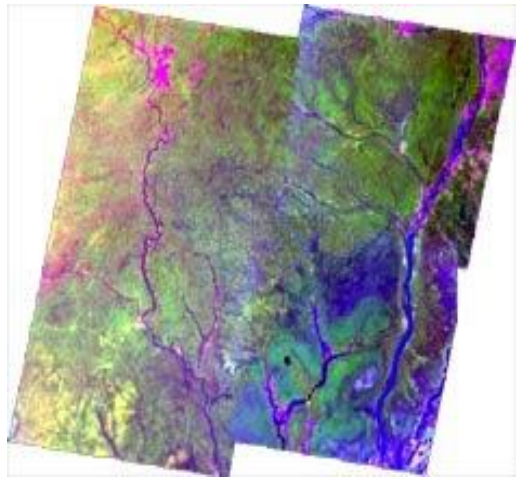
従来の表現

```
SDO_GEOMETRY(2002, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 2, 1),
 SDO_ORDINATE_ARRAY(0, 0,
  -.02912839, .059699523, -.05624374, .118211319, -.08139356, .175559751, -.10462535, .231769184,
  -.12598662, .286863981, -.14552488, .340868505, -.16328764, .39380712, -.17932241, .445704191,
  -.1936767, .496584079, -.20639802, .54647115, -.21753387, .595389767, -.22713177, .643364292,
  -.23523922, .690419091, -.24190374, .736578527, -.24717284, .781866962, -.25109401, .826308762,
  .580907505, 2.89953885, .586818223, 2.88495824, .592711277, 2.8695188, .59858545, 2.85324491,
  .604439523, 2.83616093, .610272278, 2.81829122, .616082496, 2.79966015, .621868959, 2.78029208,
  .62763045, 2.76021137, .63336575, 2.7394424, .639073641, 2.71800951, .644752905, 2.69593707,
  .650402323, 2.67324946, .656020678, 2.64997102, .661606751, 2.62612614, .667159324, 2.60173916,
  .672677178, 2.57683445, .678159097, 2.55143638, .683603861, 2.52556931, .689010252, 2.49925761,
  .694377052, 2.47252563, .699703043, 2.44539774, .704987007, 2.41789831, .710227725, 2.3900517,
  .71542398, 2.36188226, .720574553, 2.33341438, .725678226, 2.3046724, .730733781, 2.27568069,
  .735739999, 2.24646363, .740695663, 2.21704556, .745599554, 2.18745085, .750450454, 2.15770388,
  .755247146, 2.12782899, .75998841, 2.09785056, .764673028, 2.06779294, .769299783, 2.03768051,
  .773867456, 2.00753762, .778372773, 1.97738522, .782794569, 1.94721442, .787102465, 1.91700098,
  .791266007, 1.88672054, .795254739, 1.85634874, .799038208, 1.82586121, .802585958, 1.79523359,
  .805867536, 1.76444151, .808852487, 1.73346062, .811510356, 1.70226654, .813810689, 1.67083492,
  .815723031, 1.63914139, .817216927, 1.60716159, .818261924, 1.57487115, .818827566, 1.54224571,
  .8188834,   1.50926091, .818398969, 1.47589238, .817343821, 1.44211576, .8156875,   1.40790668,
  .813399552, 1.37324079, .810449523, 1.33809372, .806806957, 1.3024411,  .802441401, 1.26625857,
  .797322399, 1.22952177, .791419497, 1.19220633, .784702242, 1.15428789, .777140177, 1.11574209,
  .768702849, 1.07654456, .759359802, 1.03667094, .749080584, .99609687, .737834738, .954797979,
  .725591811, .912749906, .712321348, .869928288, .697992894, .826308762, .682575995, .781866962,
  .666040196, .736578527, .648355043, .690419091, .629490081, .643364292, .609414855, .595389767,
  .588098912, .54647115, .565511797, .496584079, .541623054, .445704191, .51640223, .39380712,
  .48981887, .340868505, .46184252, .286863981, .432442724,  .231769184, .401589029, .175559751,
  .36925098, .118211319, .335398121, .059699523, .3, 0))
```

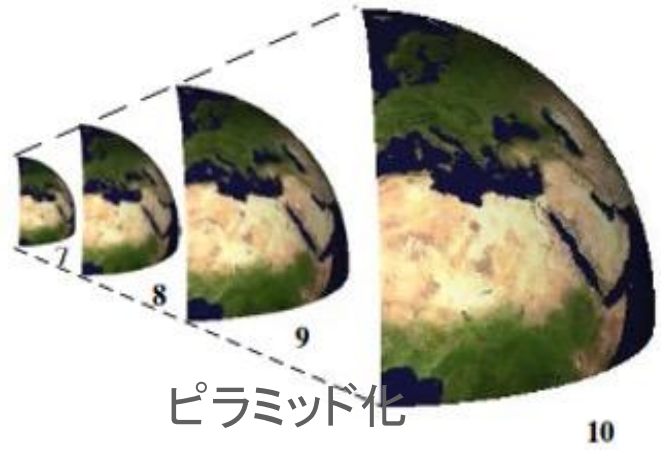
ラスターデータ機能の拡充

12c R1

- ラスター代数演算の対応
  - ラスター代数言語によるラスター処理をDB内で並列実行
- 高度な画像処理
  - 地理参照される画像データの補正処理(幾何学補正、オルソ補正など)、
  - モザイク処理、ピラミッド化の並列化、高速化
  - イメージマスキング処理、セグメント化、線形ストレッチ処理
  - 内部での再投影処理と幾何学補正
  - NDVI計算、タッセルドキャップ変換



モザイク化



ピラミッド化

Spatial 12.2 新機能サマリー

12c R2

- **空間索引の機能拡充**
  - 新しい空間索引
  - パーティション方式の追加対応
  - 点データ用の軽量な空間索引の追加
- **GeoJSON対応**
- **Location Tracking Service**
- SVA高速化
  - AND演算, MINUS演算
- 標準WebServiceの追加対応
  - Catalog Services for the Web (CSW) 2.0.2
  - WCS (Web Coverage Service)

- 3D 関連機能
  - 点群/TIN
    - 点群関連関数追加(Hilbert数算出関数など)
    - 点群ファイルフォーマットの対応拡大
    - CityGML対応強化
- ラスター機能
  - 機能追加
  - ラスター代数式の機能拡張
  - 画像演算機能の強化
  - JPEG 2000対応
- GoldenGate対応データの追加
  - トポロジーデータ,ラスターデータが新たに対応

空間索引の機能拡充 (新しい空間索引)

12c R2

- **新しい空間索引（ SPATIAL_INDEX_V2 ）**
  - システム管理の空間索引
    - パーティショニング
  - 索引作成/統計情報取得の性能向上
    - REDO生成量が1/3
    - ２－３倍の高速化
      - グローバル一時表(GTT)利用

※今後はこちらの空間索引の利用が推奨となります

```
CREATE INDEX spidx_area ON area (geom)
    INDEXTYPE IS MDSYS.SPATIAL_INDEX_V2;
```

- **パーティション方式の追加対応**
  - 新たにハッシュ, リスト,インターバル,仮想列でのパーティションに対応
    - 空間索引をローカル索引として作成が可能になります
    - （ 12.1まではレンジパーティションのみ）

空間索引の機能拡充(コンポジットB-Tree索引)

12c R2

- 点データ専用の軽量な空間索引(**コンポジットB-Tree索引**)の追加

```
CREATE INDEX pt_idx ON PT_CB(c2)
   indextype is MDSYS.spatial_index_v2
    PARAMETERS ('layer_gtype=POINT cbtree_index=true');
```

  - **空間索引の作成コストが小さい**のが特徴
    - 1000万件の点データに対する索引作成時間
      - 従来型索引：13分　に対して cbtreeでは 15秒で作成が完了
  - ユースケースに応じた選択が可能になります
    - R-Tree型の索引を置き換えるものではないです
    - クエリ性能はR-Tree索引より遅くなる可能性があります
      - データ量、処理内容(1次フィルター処理のみか)、地理参照の有無などに影響されます

cbtreeを使うことを検討するパターン
◎1次フィルター処理が主
◎データ更新頻度が高い
◎地理参照を使わない
※2次フィルター処理がある場合にはcbtreeはクエリ劣化が激しくなる可能性があります

空間索引の機能拡充(コンポジットB-Tree索引)

12c R2

```
SQL> select count(1) from densha_eki_b ;
  COUNT(1)
----------
    216680
```

- **通常の空間索引(R-Tree) の作成**

```
CREATE INDEX spidx_densha_eki_b ON DENSHA_EKI_b(GEOM)
  INDEXTYPE IS MDSYS.SPATIAL_INDEX_V2
    PARAMETERS('layer_gtype=POINT');
索引が作成されました。
経過: 00:00:09.68
```

- **コンポジットB-Tree索引の作成**

```
CREATE INDEX spidx_densha_eki_b ON DENSHA_EKI_b(GEOM)
  INDEXTYPE IS MDSYS.SPATIAL_INDEX_V2
    PARAMETERS('layer_gtype=POINT cbtree_index=true');
索引が作成されました。
経過: 00:00:00.24
```

空間索引の機能拡充(コンポジットB-Tree索引)

12c R2

クエリ性能

- **SDO_FILTER処理のみでデータ量が少ない場合には、cbtreeの方が高速。**
  - データ量が増えると、R-Treeと同程度
- **2次フィルターまで処理を行う場合で、かつ地理参照を含む場合、R-Treeの方が高速**

SQLクエリ	索引
SELECT count(*) FROM abi, **states** WHERE **SDO_FILTER(geometry, geom) = 'TRUE';**	**C-BTree**
	R-Tree
	C-BTree
	R-Tree
SELECT count(*) FROM abi, **counties** WHERE **SDO_FILTER(geometry, geom) = 'TRUE';**	**C-BTree**
	R-Tree
	C-BTree
	R-Tree
SELECT count(*) FROM abi, states WHERE **SDO_ANYINTERACT(geometry, geom) = 'TRUE';**	C-BTree
	R-Tree
	C-BTree
	R-Tree
SELECT count(*) FROM abi, countries WHERE **SDO_ANYINTERACT(geometry, geom) = 'TRUE';**	C-BTree
	R-Tree
	C-BTree
	R-Tree

Slide Onlyの為内容削除

ご参考：空間クエリの２段階フィルター処理

SDO_RELATE

SDO_FILTER

簡易絞り込み用のSDO_FILTER関数は1次フィルタ処理のみで結果を戻します

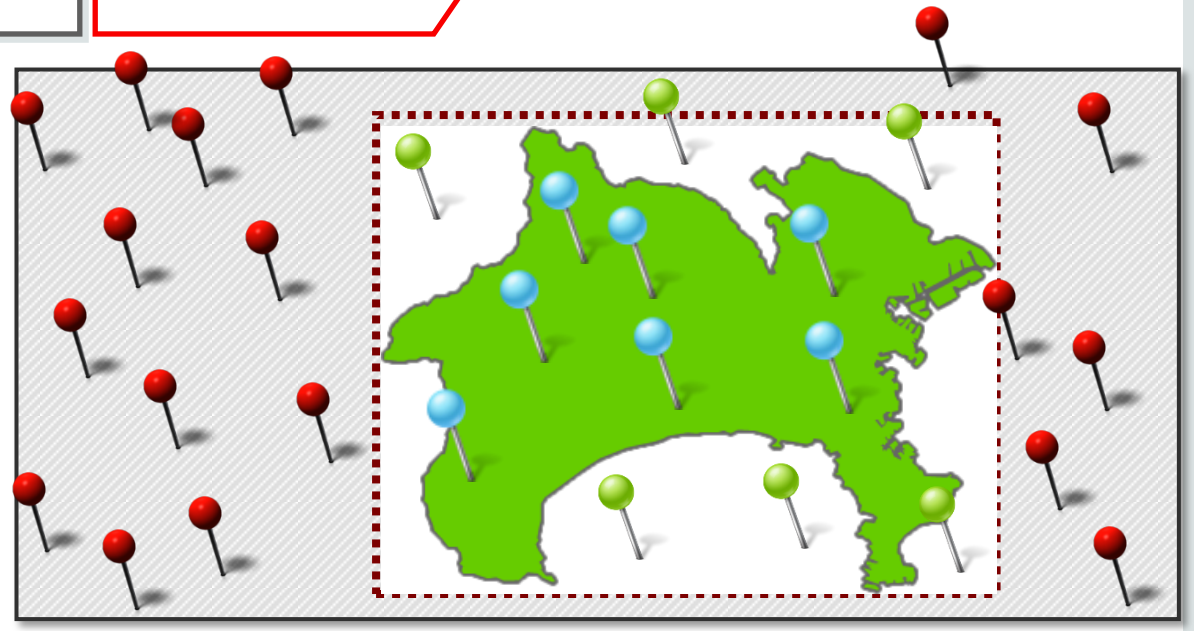
空間データ

1次フィルタ

空間データ

２次フィルタ

結果データ



空間索引の利用

コストの高い厳密な空間判定演算の範囲を最小境界矩形(MBR)をベースとした空間索引で高速に絞り込みます。

GeoJSON のサポート

12c R2

- DB内のJSONデータに対する空間操作をサポート

```
SQL> SELECT  JSON_VALUE(
 '{"type": "Point",
   "coordinates": [125.6,  10.1]}',
        '$' returning sdo_geometry)
FROM  dual ;
```

```
SDO_GEOMETRY(2001, 4326,
   SDO_POINT_TYPE(125.6, 10.1, NULL), NULL, NULL)
```

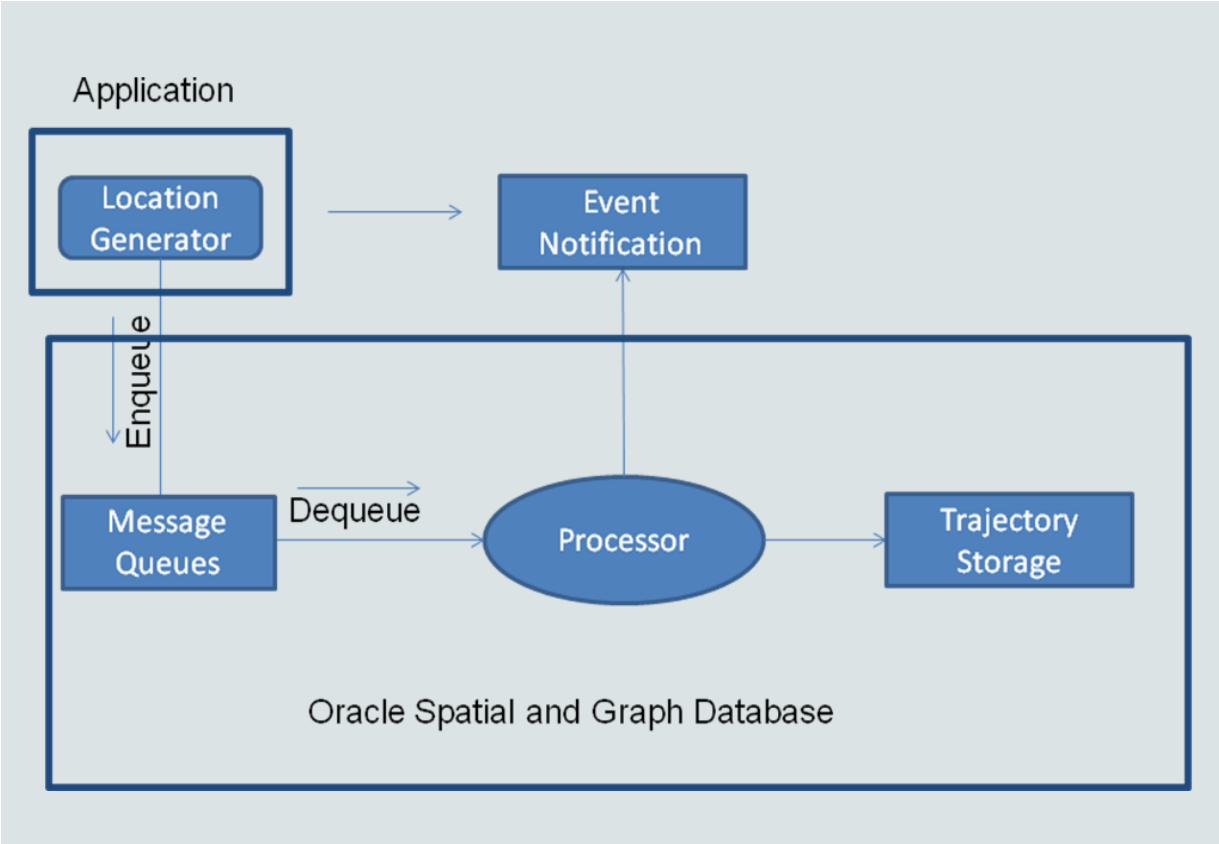
GeoJSON 出典:Wikipedia

GeoJSON[1]はJavaScript Object Notationを用いて空間データをエンコードし非空間属性を関連付けるファイル形式である。

- SDO_GEOMETRY コンストラクタにJSONでの入力が可能に
  - JSON_VALUE関数はGeoJSONとSDO_GEOMETRYに対応
- DB内のJSONドキュメントに直接空間索引を作成し、空間検索が可能
- SDO_GEOMETRY, GeoJSON間の動的な変換関数の追加
  - SDO_UTIL.TO_GEOJSON
  - SDO_UTIL.FROM_GEOJSON

ロケーション・トラッキング・サービス

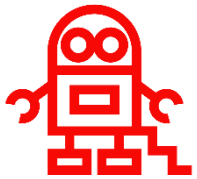
12c R2

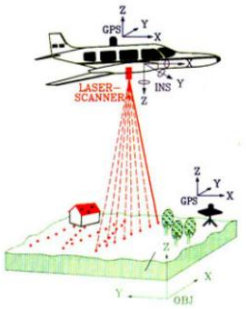


- 位置情報や地理的特性によるイベント処理をトラッキングするためのJMXを利用したJava API を提供
- **Point in Polygon分析の拡張**
  - 複数の指定範囲内の複数の移動する物体を分析する
  - 数千のエリアでの数百万に及ぶ移動物体をトラッキングできます
- 民間、政府のクラウド上での大規模な利用を想定して設計
  - DB内ではSDO_TRKRパッケージで関数を提供。AQ,JavaAPIと連携。

点群/不規則三角網(TIN)

12c R2





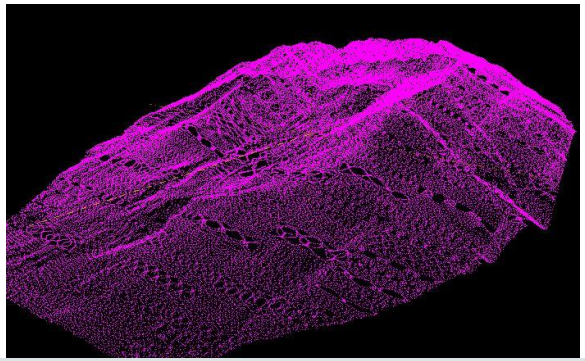
新機能サマリ

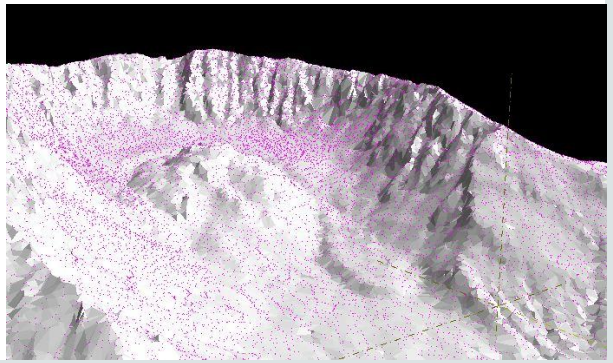
点群: 点の集合で3Dを表現するデータ（LiDARなど）

不規則三角網: 三角形で表面形状を表現

- 性能、スケーラビリティの改善
  - **ヒルベルトR-Treeモデル/ヒルベルト数の算出プログラムなど**
  - 大規模点群データに対するパラレルクエリの改善
  - 属性データに対応する新しいストレージモデルの追加
  - 点群データ用ローダーの性能改善
  - 等高線の算出

SDO_PC_PKGに新規追加
ADD_HILBERT_TO_FLAT_MODEL_PC
GETNTHHILBERTVALUE
GENERATE_HILBERT_VALS
CLIP_PC_INTO_LAS
.. 他数十個新規追加





点群/3D

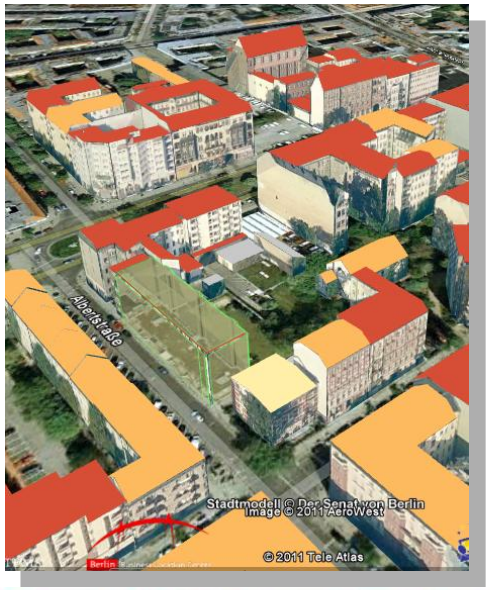
12c R2

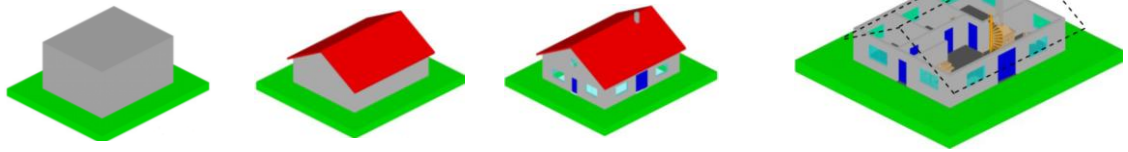
CityGML 機能拡張

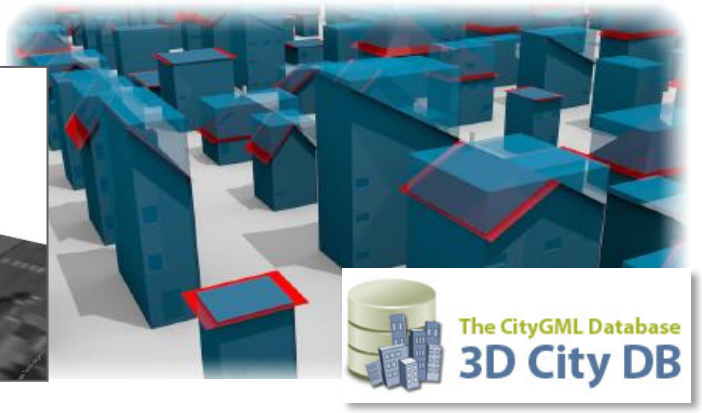
CityGML 2.0対応
3dCity 2.0.1

CityGML: 都市モデルを記述する言語

- **3DCityDB (OSS)との連携強化**
- CityGMLモデルからネイティブに格納する3Dスキーマの提供
- CityGML全域にわたるフル3Dでの格納およびクエリ
- CityGML用のネイティブローダー
  - Oracle用のCityGMLドキュメントの高性能ローダーの提供







補足：Hilbert R Tree (ヒルベルトRツリー)

- **ヒルベルト曲線は空間充填曲線のひとつです**
  - 3次元(N次も)空間にも対応しています
  - ヒルベルト曲線を用いて空間全体に順序付けができます
- R-Treeを併用した手法(Hilbert-R-tree)での索引づけが可能です
  - **近傍検索などに威力を発揮します**

ヒルベルトRツリーは、線、領域、3-Dオブジェクト、高次元の特徴ベースのパラメトリックオブジェクトなどの多次元オブジェクトのインデックスです。
ヒルベルトRツリーは空間充填曲線を利用しヒルベルト曲線はデータの長方形の線形な順序を課します

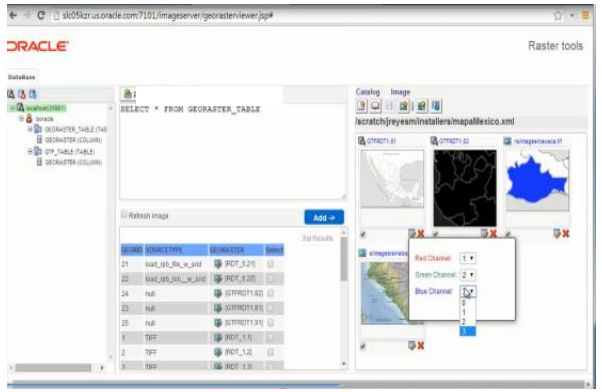
出典:Wikipedia (en): Hilbert R-Tree:
https://en.wikipedia.org/wiki/Hilbert_R-tree

ラスター機能の向上

12c R2

機能追加

- イメージ演算機能の強化
  - モザイク化の際の新しいルール定義の追加(color balancing)
  - 新規演算パッケージの追加 (SDO_GEOR_IP)
    - linear stretching,piecewise stretching,equalization ,normalization
    - histogram matching, dodging , filtering ,
    - warping (sdo_geor.warp in SDO_GEOR package)



- ラスター代数式の拡張
  - 25種の演算子の追加: conditionalExpr (IF-THEN-ELSE), ^ (XOR), % (MODULO), POWER, FACTORIALなど
- JPEG2000圧縮への対応
- 新しいGeoRASTER Viewer&ETLツールの提供

おまけ:ちょっとうれしい拡張

12c R2

- **楕円体距離演算の拡張**
  - 点や複数点間以外でも楕円体近似での距離計算が可能になりました。
    - より正確な距離計算が可能になりました

SDO_WITHIN_DISTANCEや SDO_GEOM.SDO_DISTANCE などの距離計算や範囲内での検索時の距離計算が楕円体距離演算になることでより高い精度での検索や演算が可能になります

(例) 港区から遠い東京23区内の丁目と距離(m)

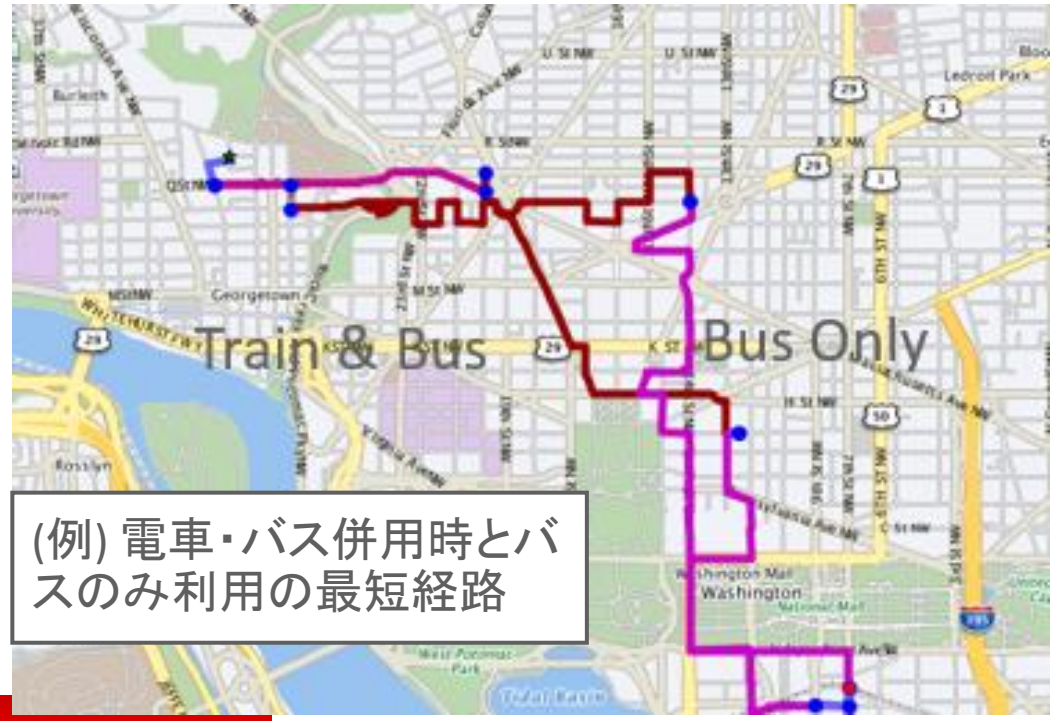
```
SELECT a.gst_name, a.moji, a.key_code,
   SDO_GEOM.SDO_DISTANCE( a.geometry, b.geom, 0.0005,
       'unit=m','ellipsoidal=true') dist
FROM tokyo02 a, minatoku b
ORDER BY dist;
```

足立区	花畑８丁目	13121067008	16262.9429
葛飾区	東水元５丁目	13122028005	16369.5831
葛飾区	東水元６丁目	13122028006	16403.4701
葛飾区	東金町８丁目	13122025008	16490.081
練馬区	西大泉町	131200420	16637.3619

Spatial and Graph 12.2 updates Network Data Model

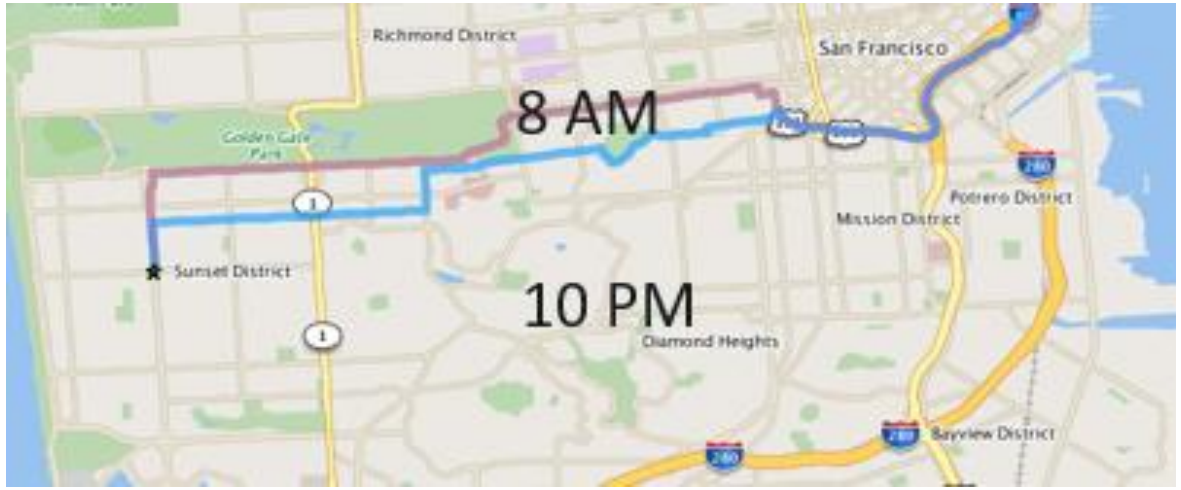
12c R1: ネットワーク・データ・モデル(Network Data Model)

- **複数モード(Multimodal)ルーティング**
  - 複数の交通手段を合わせた経路探索
    - バス,電車, 徒歩, 自家用車など



(例) 電車・バス併用時とバスのみ利用の最短経路

- **時間制約付きの経路と解析**
  - 時間帯により変化する経路を鑑みた経路探索



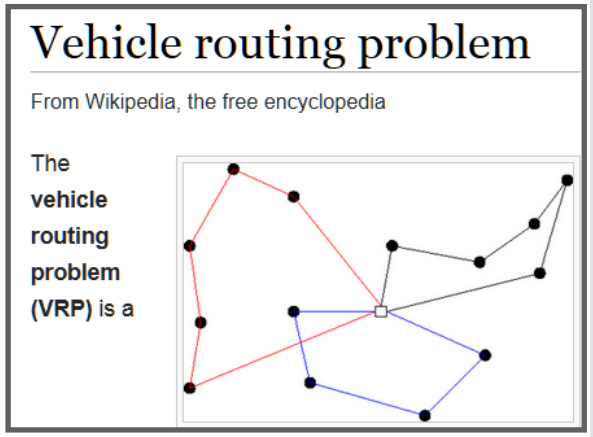
(例) 朝8時と夜10時の最短経路

ネットワーク・データ・モデル、ルーティングエンジン

12c R2

- **Network Data Model**
  - NFE
    - ネットワーク経路上に特徴を配置(飲食店など)
  - 配車ルート問題(VRP),マルチ巡回セールスマン
  - 並列処理の高速化
    - Java Concurrent Frameworkの利用
- **Routing Engine**
  - 時間帯(TZ)情報の付加が可能に
  - 交通パターンの加味
  - 国境情報も加味してコスト判断

出典:Wikipedia



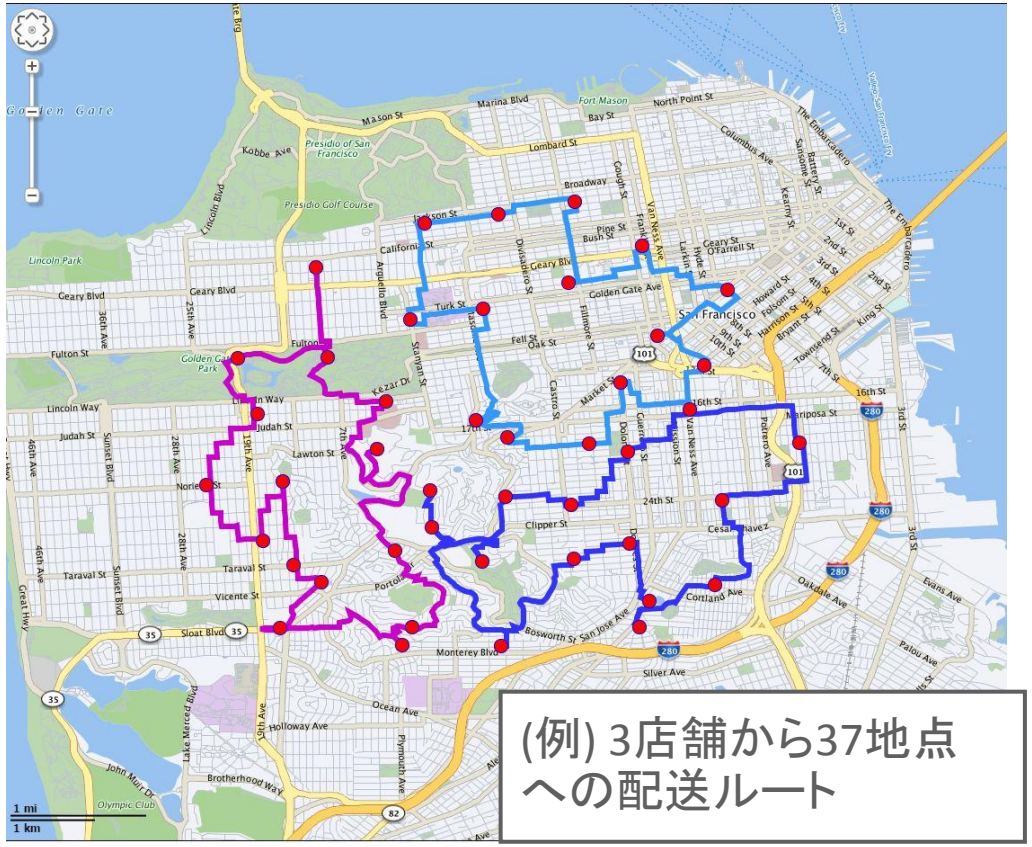
12c R2: ネットワーク・データ・モデル(Network Data M

12c R2

- **ネットワークバッファ**
  - 複数の到達可能範囲のデータ化と分析

(例) 8つの店舗から各徒歩5分圏内のデータ

- **マルチ巡回セールスマン**
  - 複数の車で複数の拠点を網羅する最適解



(例) 3店舗から37地点への配送ルート

Spatial and Graph 12.2 updates
RDF Semantic Graph

RDF Semantic Graph - 12.1 New Features

- **RDF View (R2RML, Direct Mapping)**
  - RDBデータを標準変換ルール、言語を用いて仮想的にRDF化が可能に
- **SPARQL 1.1 拡張**
  - SPARQL 1.1 構文のサポート
- **ユーザー定義推論**
- **OGC GeoSPARQLサポート**
- OWL EL サポート
- ラダーベース推論
- セマンティックネットワークのimpdp/expdpサポート
- 仮想モデルのreplace

12.1.0.2

SPARQLフェデレーテッドクエリ（service句）

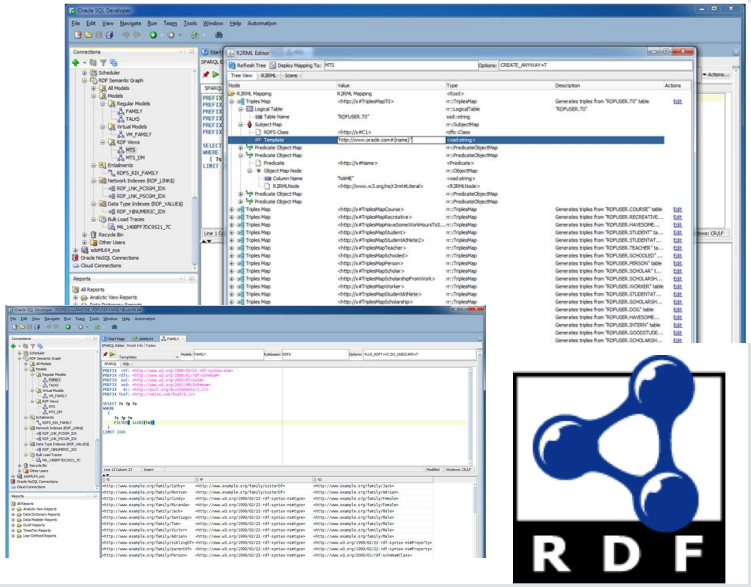
RDFビューと実トリプルストアの結合

RDF Semantic Graph - 12.2 New Feature

12c R2

- **機能強化**
  - **SPARQL機能強化**
  - 既定推論ルールの追加（OWL2 QL）
  - 性能強化
    - SPARQL 2 SQL 変換ロジックの改善
    - データタイプ索引の追加（文字列部分一致用）
- **ユーザビリティの向上**
  - **RDF Studio（on SQL Developer）**
    - RDFモデルの作成、バルクロード,SPARQLクエリエディタ , R2RML エディタ
  - **Cytoscape/Protegeプラグインのアップデート**
  - RDB 2 RDF / RDF View使い勝手向上
    - DBLink対応、RDF Viewの上書き作成(以前はdrop/create) ,Viewの性能向上など

※注意事項
12.2へのアップグレードには**セマンティックデータのマイグレーションが必要**になります(内部的に利用される表やビューの名称などが変更されました)
マイグレーションには以下のプロシージャが利用できます。
SEM_APIS.MIGRATE_DATA_TO_CURRENT



SPARQL機能強化: SEM_APIS.UPDATE_MODEL

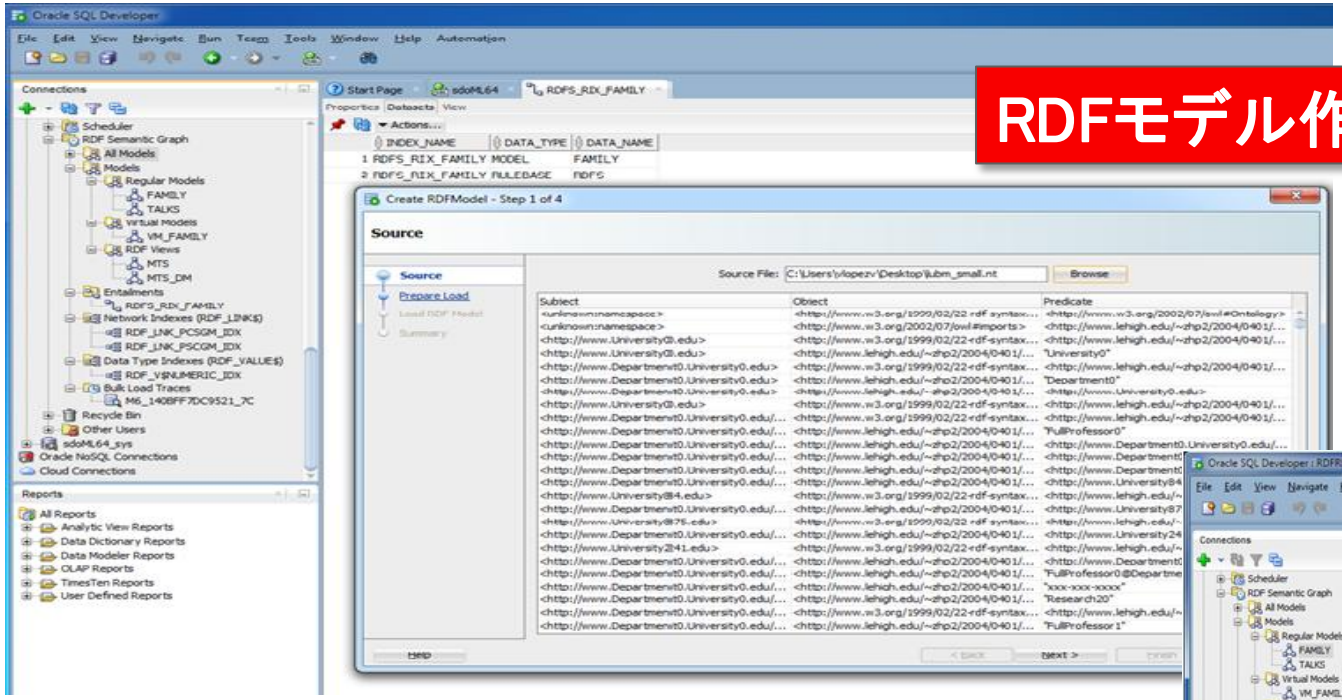
12c R2

- **SEM_APIS.UPDATE_MODELプロシージャ**
  - **SPARQL更新処理(SPARQL 1.1 updates)の構文をDB側でネイティブサポート**
- SPARQL filter句に利用可能なユーザー定義関数 (PL/SQL)
- SPARQLをSQLに変換する関数の提供 (SEM_APIS.SPARQL_TO_SQL 関数)
- SPARQL性能向上
  - プロパティパス,aggregateの性能向上など
- GeoSPARQL機能拡張
  - 判定可能な位相関係の追加
- プロパティグラフとの相互運用

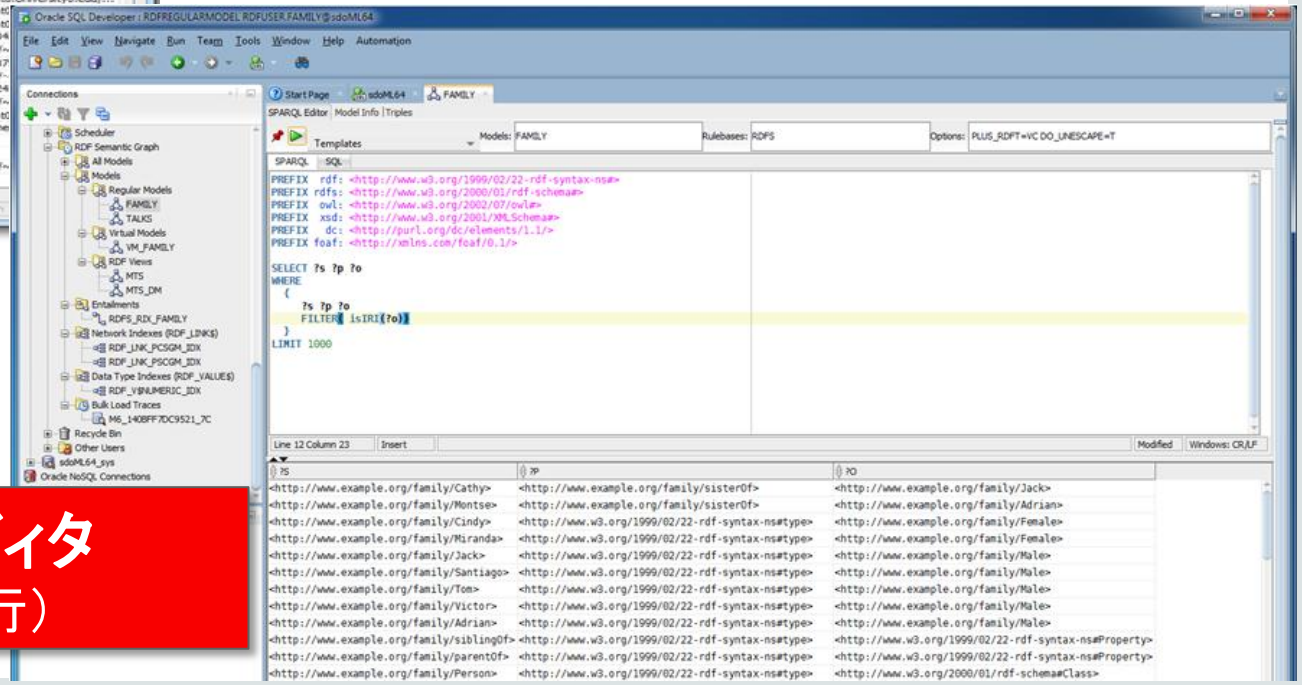
```
begin
   sem_apis.update_model(
     apply_model=>'M1',
     update_stmt=>'INSERT {?s :mbox ?n}
                   WHERE  {?s :email ?n}'
   );
end;
/
```

RDF Studio (on SQL Developer)

12c R2



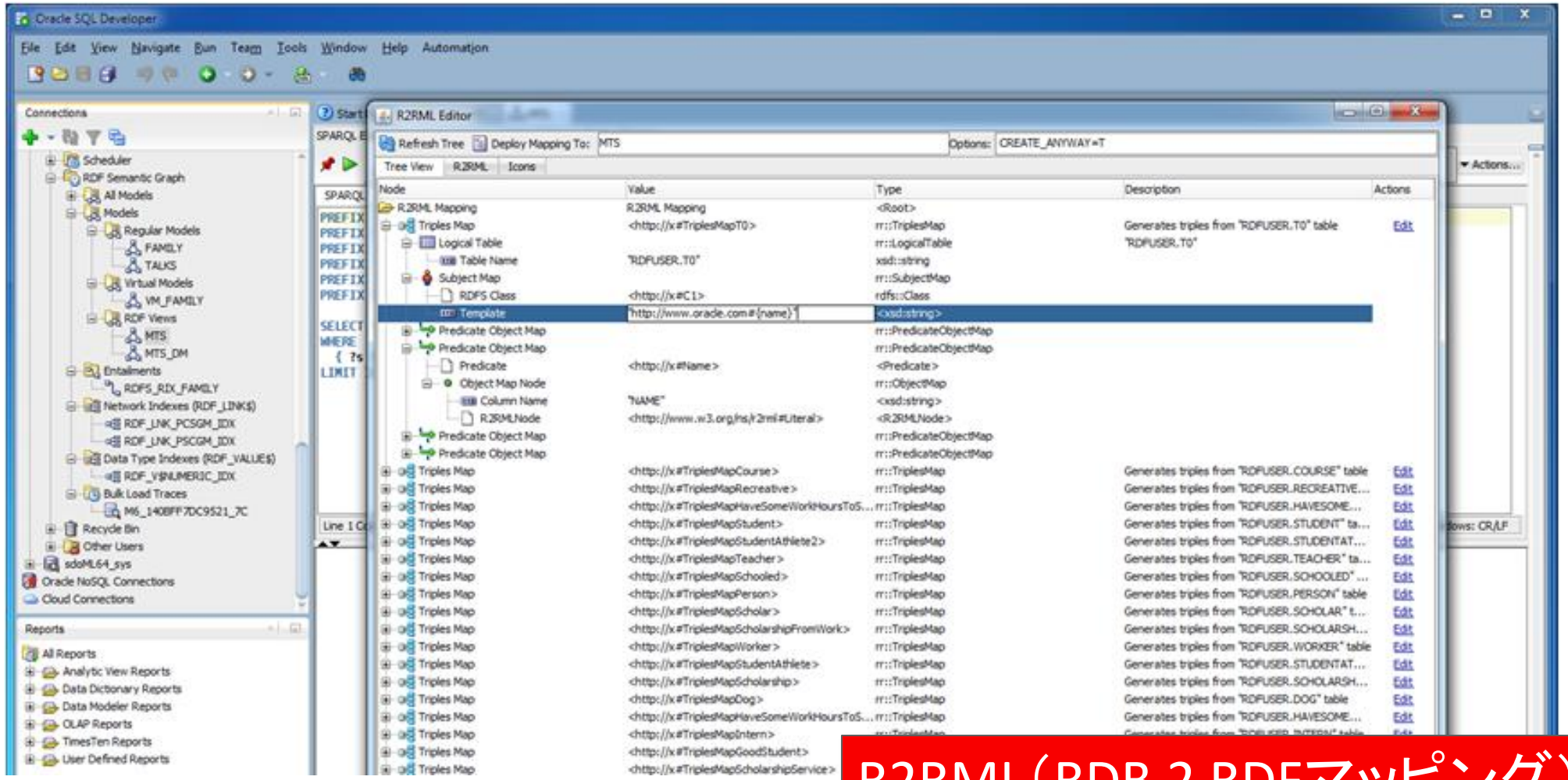
RDFモデル作成とファイルからのバルクロード



SPARQL クエリエディタ
（SPARQLクエリの実行）

RDF Studio (on SQL Developer)

12c R2



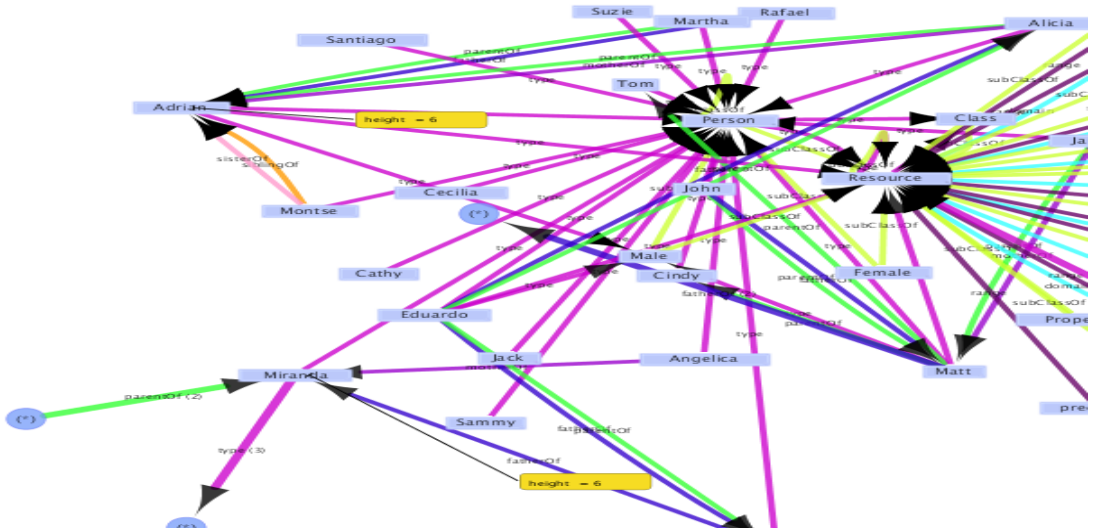
R2RML（RDB 2 RDFマッピング）エディタ

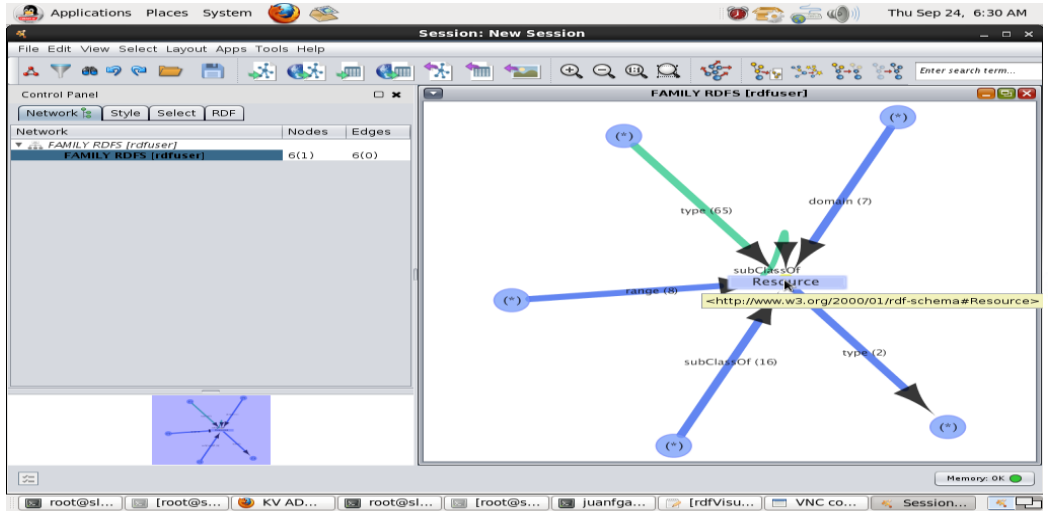
可視化機能の強化

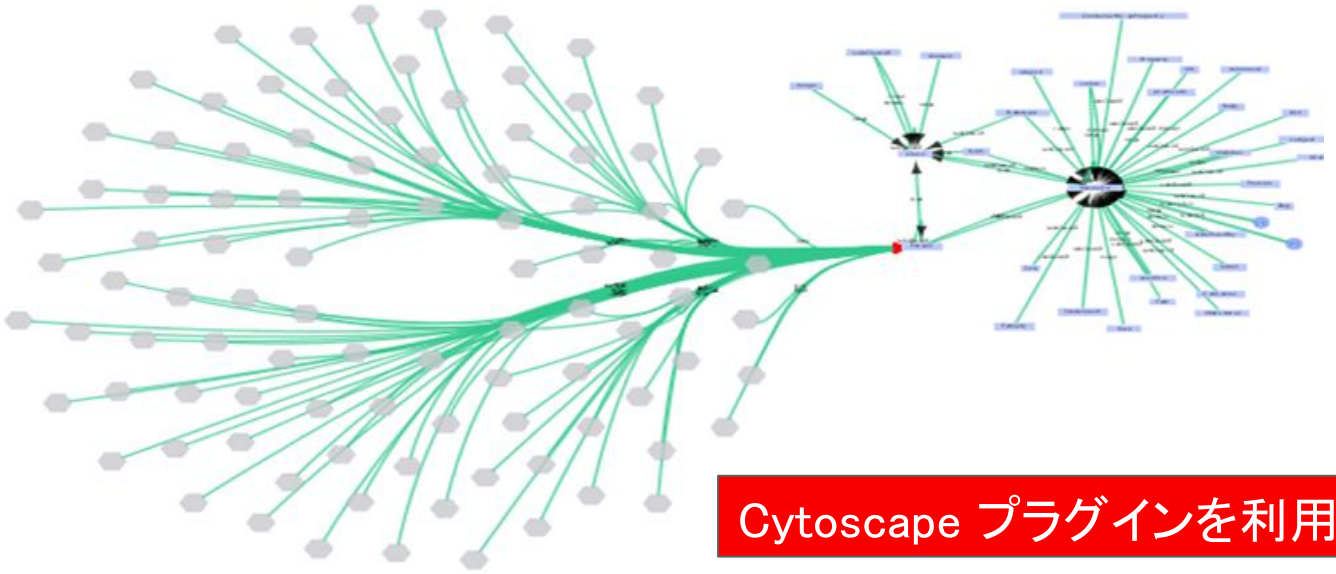
12c R2

- Cytoscape 3.2に対応したプラグイン

※OTNのSpatial and Graphページからの提供になります







Cytoscape プラグインを利用

本日、お伝えしたかったこと

- Spatial and Graph という面白いオプションがあること
- 新機能 プロパティグラフモデル の良さ

- R12.1 / R12.2 での各機能のアップデート
  - 既存ユーザ様向け

リファレンス

- Oracle Database Spatial and Graph, 12c Release 2

  http://docs.oracle.com/database/122/nav/spatial-and-graph.htm

- 関連する Oracle 製品
  - Oracle Labs PGX
  - Oracle Big Data Spatial and Graph
    - Release 1.2 ユーザーズ・ガイド in Appliance 4.6

- 関連する SlideShare 資料
  - Oracle Labs 発！ Parallel Graph Analytics（PGX）
  - Hadoop Conference Japan 2016 LT資料 グラフデータベース事始め

Safe Harbor Statement

The preceding 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, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Integrated Cloud

Applications & Platform Services


ORACLE®