



このコンテンツは公開から3年以上経過しており内容が古い可能性があります
最新情報については[サービス別資料](#)もしくはサービスのドキュメントをご確認ください

[AWS Black Belt Online Seminar]

Amazon Neptune

サービスカットシリーズ

Solutions Architect

秋田 仁雅

2020/7/14

AWS 公式 Webinar

<https://amzn.to/JPWebinar>



過去資料

<https://amzn.to/JPArchive>



自己紹介

秋田仁雅

技術統括本部 ISV/SaaSソリューション本部
ソリューションアーキテクト

好きなサービス

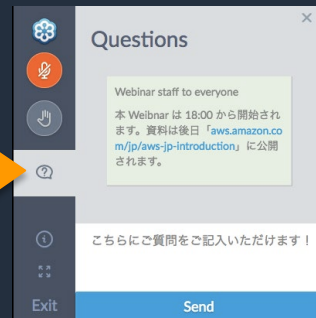
Amazon Neptune



AWS Black Belt Online Seminar とは

- 「サービス別」「ソリューション別」「業種別」のそれぞれのテーマに分かれて、アマゾンウェブ サービス ジャパン株式会社が主催するオンラインセミナーシリーズです。
- **質問を投げることができます！**
- 書き込んだ質問は、主催者にしか見えません
- 今後のロードマップに関するご質問はお答えできませんのでご了承下さい

- ① 吹き出しをクリック
- ② 質問を入力
- ③ Sendをクリック



Twitter ハッシュタグは以下をご利用ください
#awsblackbelt

内容についての注意点

- 本資料では2020年7月14日時点のサービス内容および価格についてご説明しています。最新の情報はAWS公式ウェブサイト(<http://aws.amazon.com>)にてご確認ください。
- 資料作成には十分注意しておりますが、資料内の価格とAWS公式ウェブサイト記載の価格に相違があった場合、AWS公式ウェブサイトの価格を優先とさせていただきます。
- 価格は税抜表記となっております。日本居住者のお客様が東京リージョンを使用する場合、別途消費税をご請求させていただきます。
- AWS does not offer binding price quotes. AWS pricing is publicly available and is subject to change in accordance with the AWS Customer Agreement available at <http://aws.amazon.com/agreement/>. Any pricing information included in this document is provided only as an estimate of usage charges for AWS services based on certain information that you have provided. Monthly charges will be based on your actual use of AWS services, and may vary from the estimates provided.

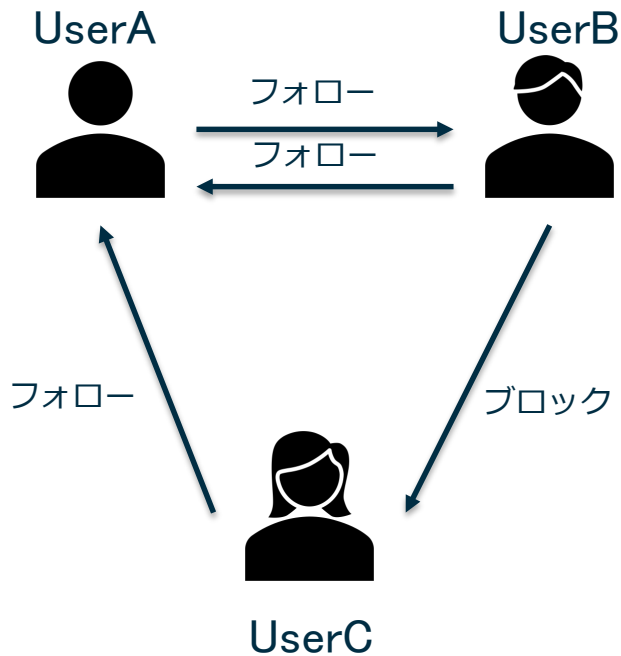
アジェンダ

1. グラフとは
2. グラフDBとは
3. Amazon Neptuneのご紹介
4. はじめるためには
5. まとめ

アジェンダ

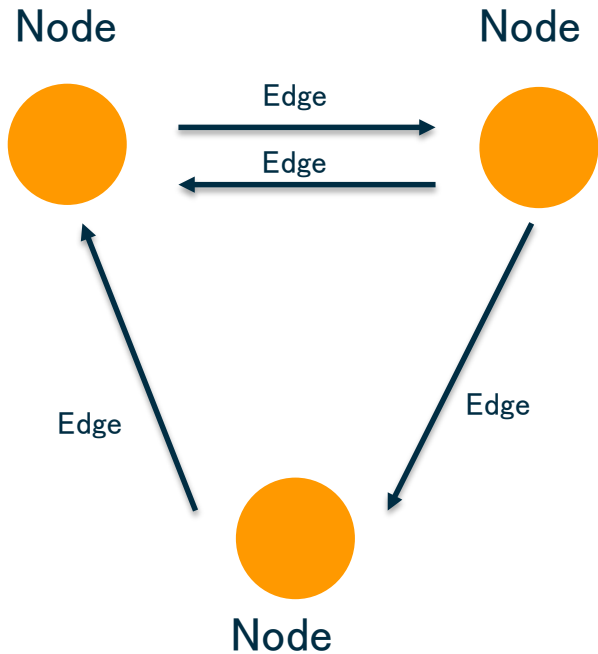
1. グラフとは
2. グラフDBとは
3. Amazon Neptuneのご紹介
4. はじめるためには
5. まとめ

グラフとは



Userが「フォロー」「ブロック」
などの矢印で結ばれている

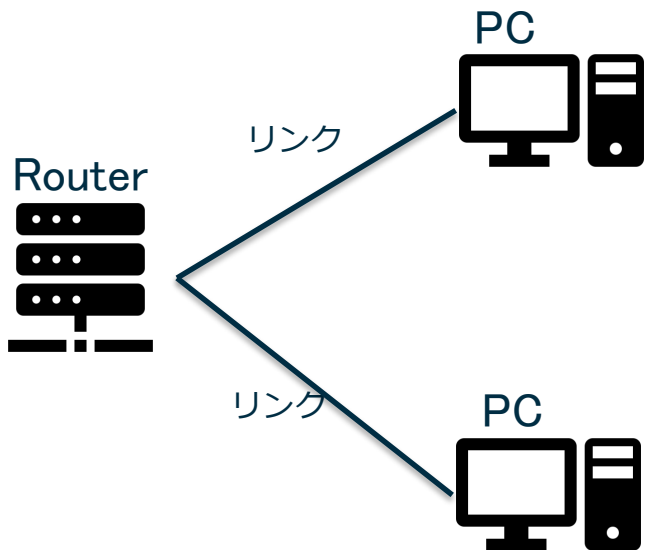
グラフとは



一般化すると、
NodeがEdgeで結ばれている

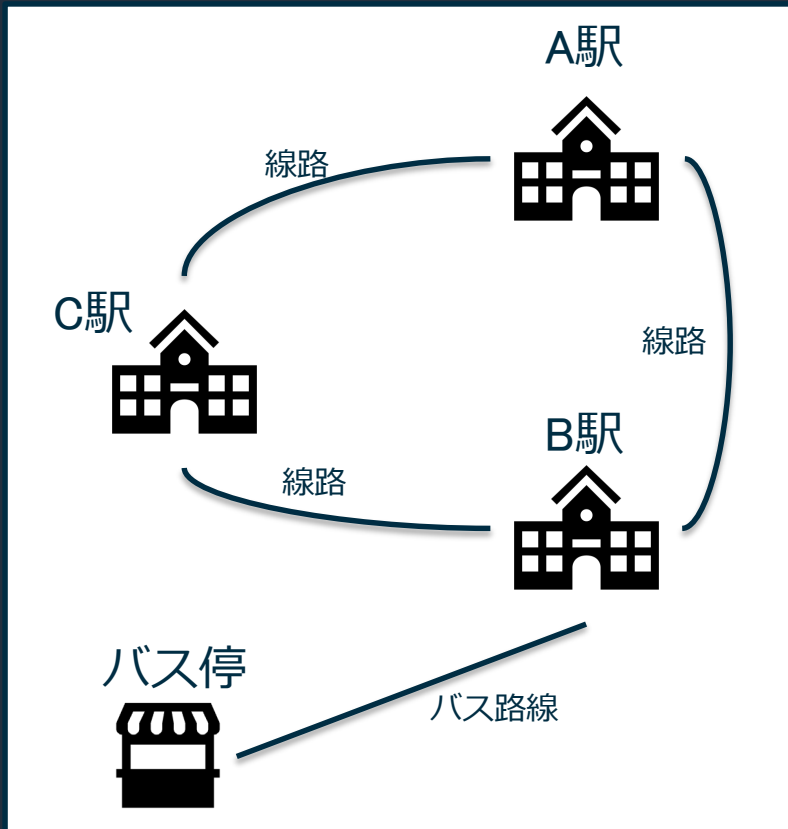
※NodeはVertexとも呼ばれる

グラフの例：IP Network



Node: ルータやPC
Edge: リンク

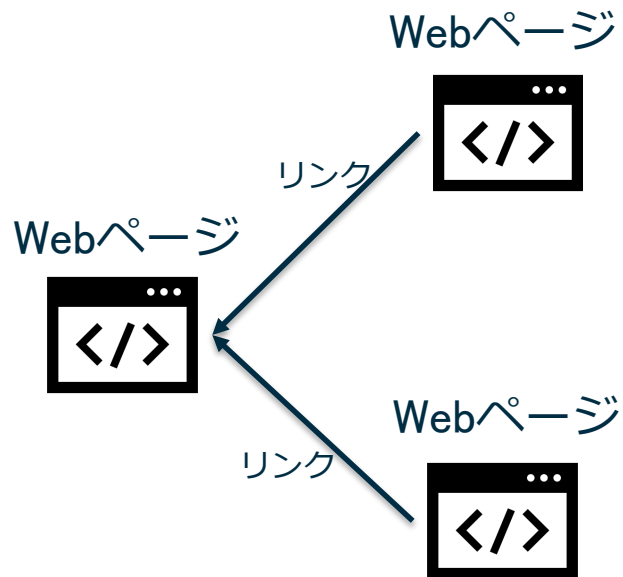
グラフの例：路線図



Node: 駅、バス停

Edge: 線路、バスルート

グラフの例：World Wide Web



Node: Webページ
Edge: ハイパーリンク

アジェンダ

1. グラフとは
2. グラフDBとは
3. Amazon Neptuneのご紹介
4. はじめるためには
5. まとめ

グラフDBとは

“グラフモデル”を効率的に格納し、検索するためのDB

2つのグラフモデルとフレームワーク

プロパティグラフ

Apache TinkerPop™ (OSS)
Gremlin Traversal Language

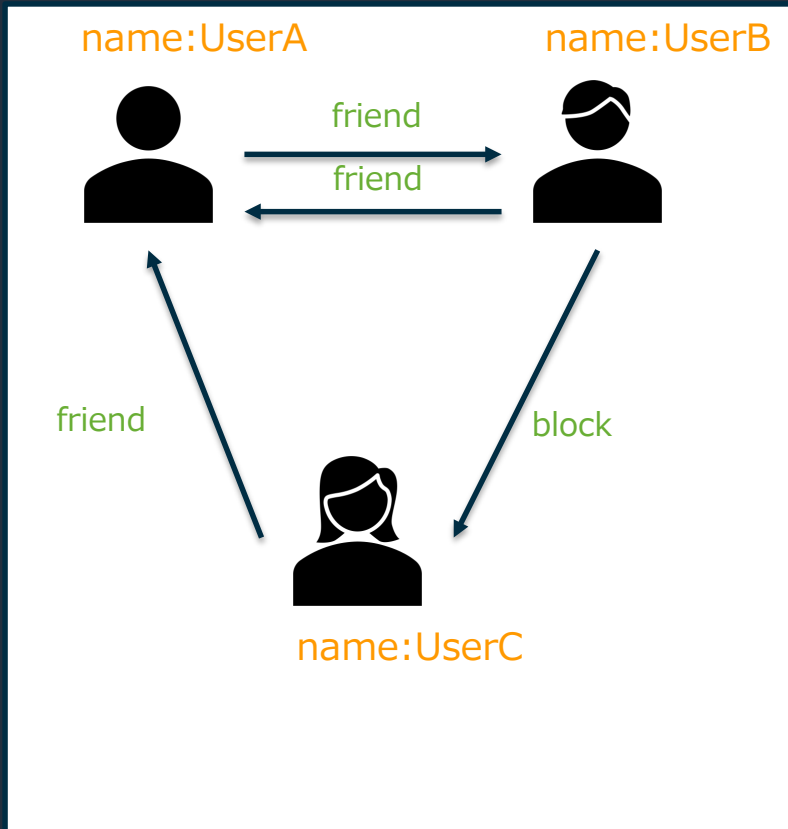


RDF

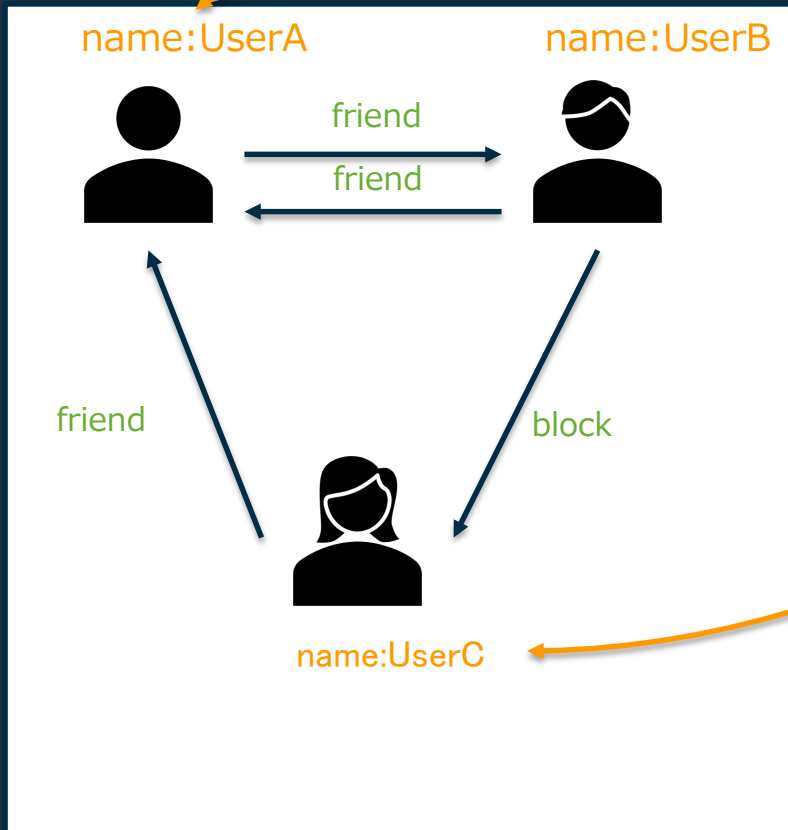
W3C標準
SPARQL Query Language



プロパティグラフモデル

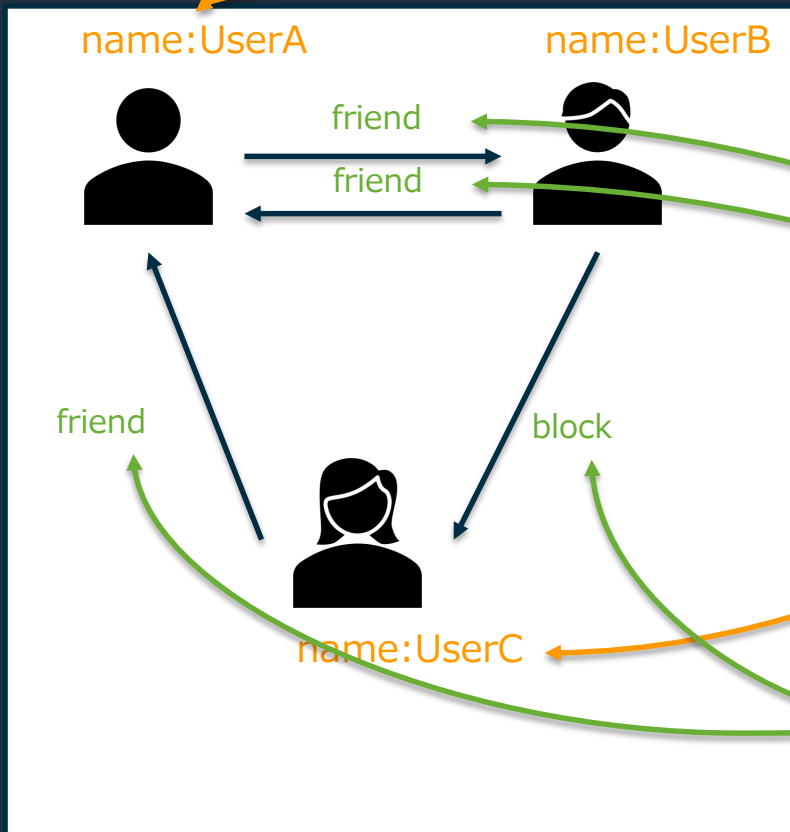


プロパティグラフモデル



Nodeのプロパティ

プロパティグラフモデル



Nodeのプロパティ

Edgeのプロパティ (ラベル)

プロパティグラフモデル

name:UserA
age:26
gender:male

age:25
job: engineer



friend

Since:2019-07-14



プロパティは、この図のように複数もつことも可能です

ノード毎、エッジ毎に異なるプロパティをもつことも可能です

RDF

主語



述語

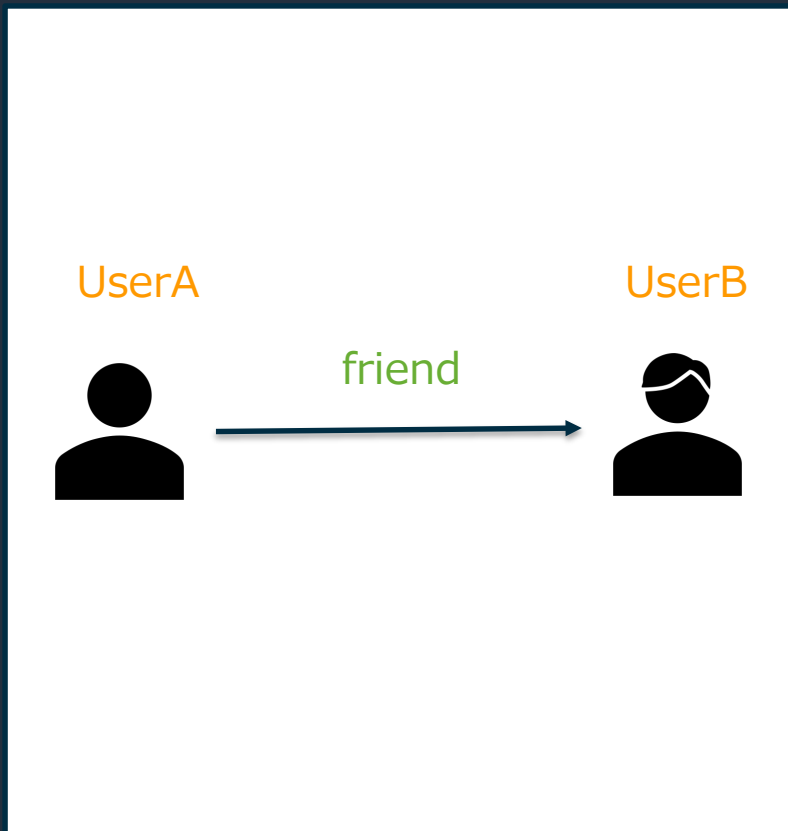


目的語



- ノード=>主語、目的語
- エッジ=>述語

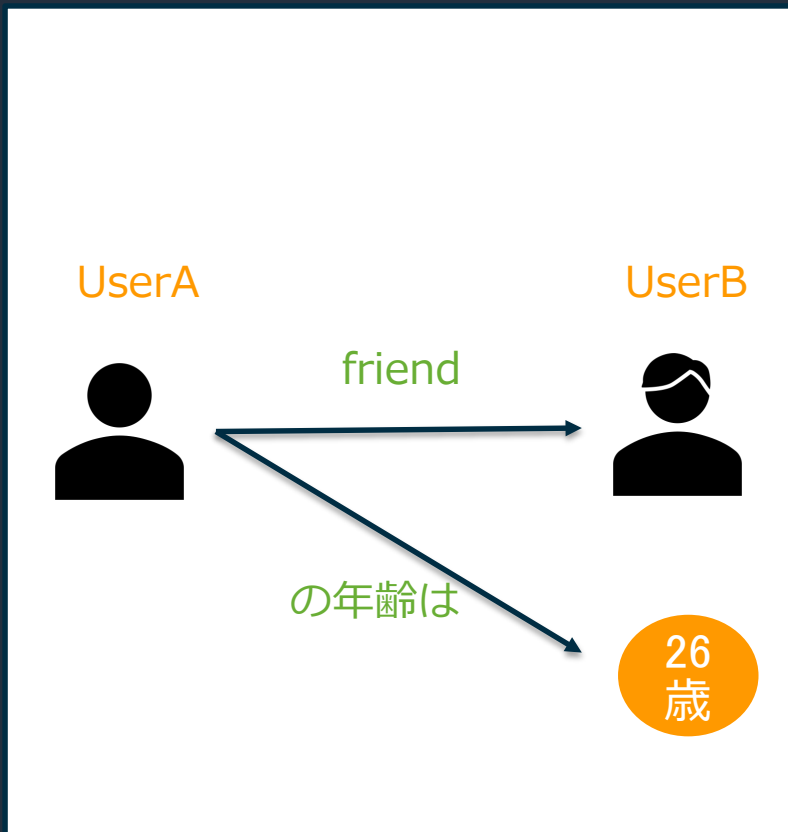
RDF



“主語” “述語” “目的語”

UserA friend UserB.

RDF



“主語” “述語” “目的語”

UserA friend UserB.

UserA の年齢は 26歳.

Gremlin 101



Gremlin 101

name:UserA



friend



name:UserB



friend



friend



name:UserC

グラフから全ノードを探したい

`g.V()`

`=> ['UserA', 'UserB', 'UserC']`

Gremlin 101

name:UserA



friend



name:UserB



friend



friend



name:UserC

Nameが“UserA”のノードを探したい

```
g.V().has('name', 'UserA')
```

=> ['UserA']

Gremlin 101

name:UserA



friend



name:UserB



friend



friend



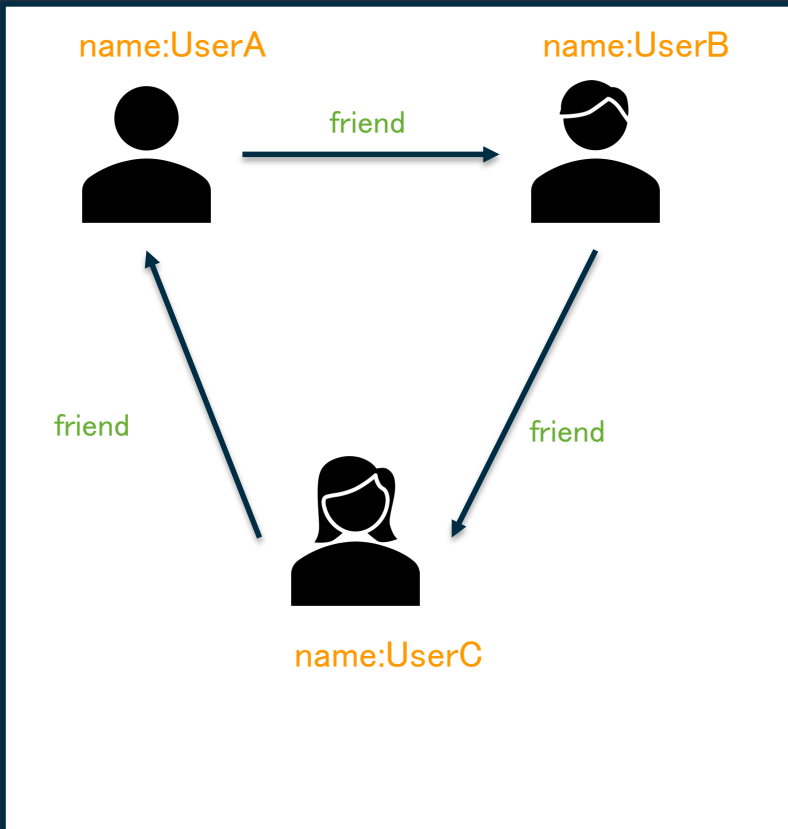
name:UserC

“UserA”の友達であるユーザを探したい

```
g.V().has('name', 'UserA').out('friend')
```

⇒ ['UserB']

Gremlin 101

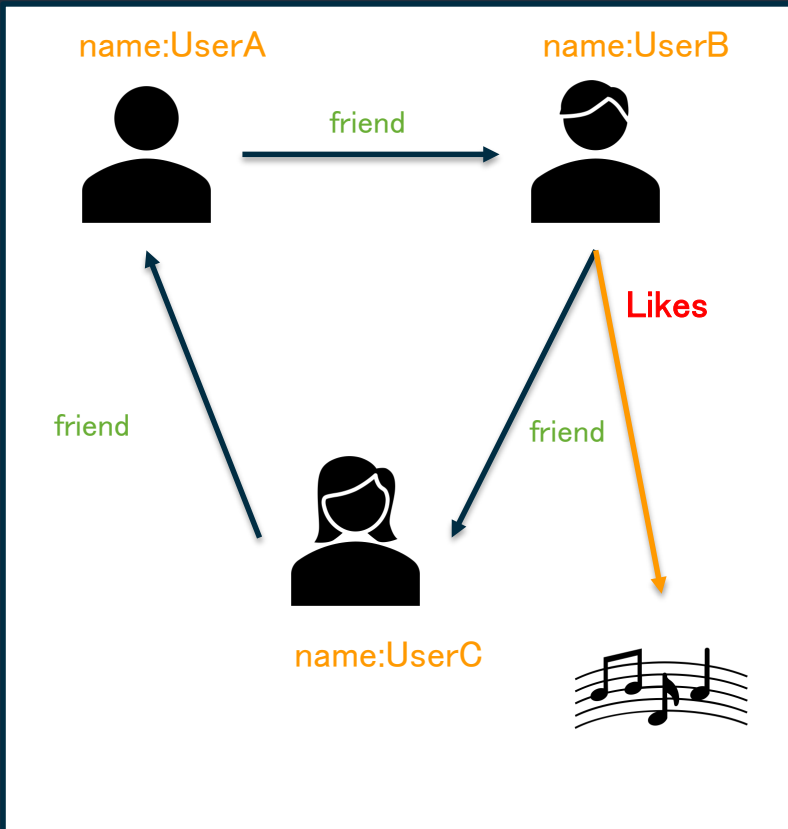


“UserA”にとって、友達の友達であるユーザを探したい

```
g.V().has('name': 'UserA').out('friend')  
.out('friend')
```

=> [UserC]

Gremlin 101



“UserA”の友達がLikesをつけているコンテンツを探したい

```
g.V().has('name': 'UserA').out('friend')  
.out('likes')
```

アジェンダ

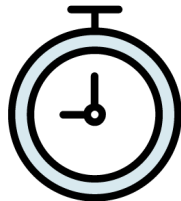
1. グラフとは
2. グラフDBとは
3. Amazon Neptuneのご紹介
4. はじめるためには
5. まとめ

Amazon Neptune の特徴

高速



高信頼性、セキュア



簡単



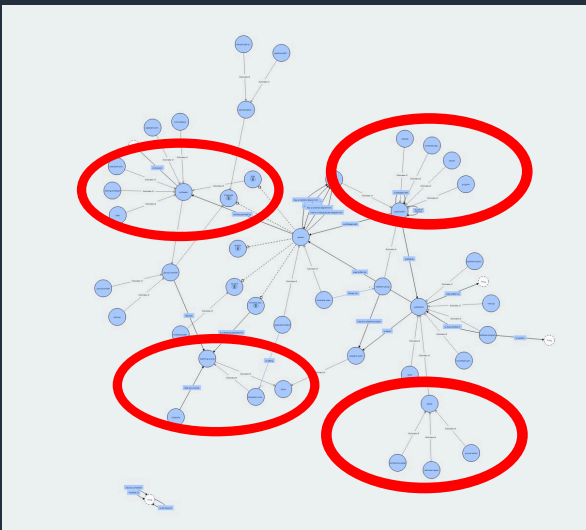
オープン



グラフ処理の分類

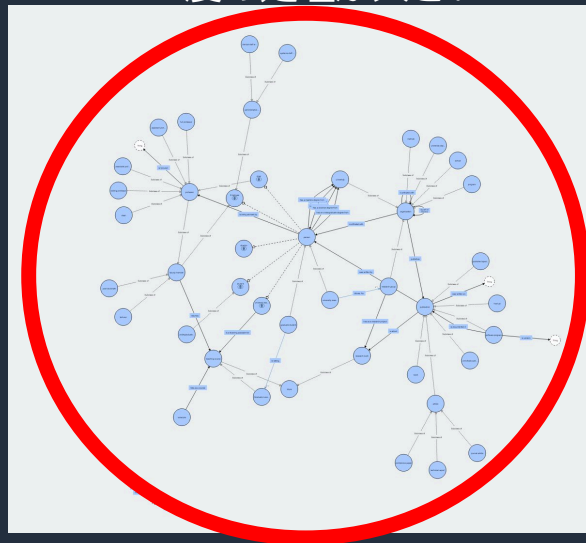
• OLTP

- 1処理単位でグラフの一部を探索
- リコメンデーション、不正検知など
並列かつ大量に実行される



• OLAP

- 1処理単位でグラフのすべてを探索
- クラスタリング、全ノードの重みバランスの調整
など
1度の処理が大きい



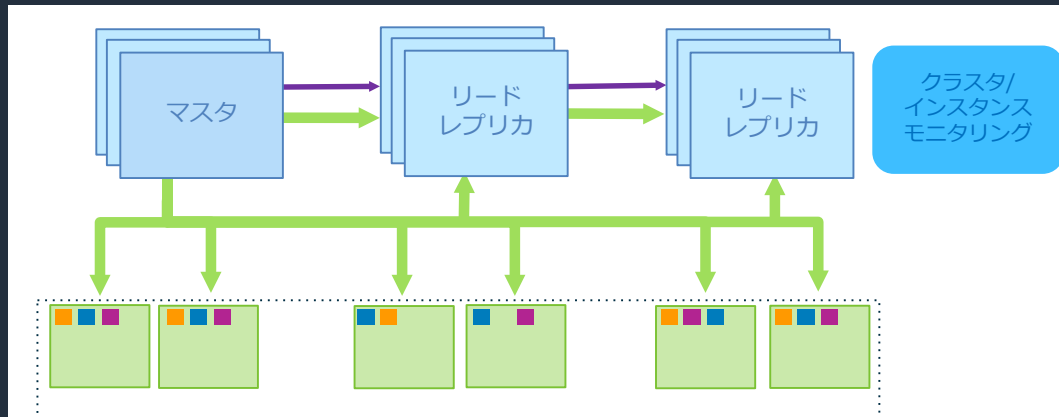
クエリ性能

- OLTPのグラフワークロードにおいて
毎秒100,000件のクエリをサポートするよう設計
 - 最大15個のリードレプリカ
 - 専用に設計されたクラウドネイティブなストレージサービス
 - 最適化されたインメモリアーキテクチャ
 - クエリの最適化
- 索引（インデックス）はNeptune側で管理される
 - クエリ実行計画やプロファイルの取得も可能

リードレプリカ

可用性

- データベースノードの障害は自動的に検知され、交換される
- データベース処理の障害は自動的に検知され、リソースはリサイクルされる
- リードレプリカは必要に応じて自動的にマスタに昇格する
- どのリードレプリカに対して優先的にフェイルオーバーさせるかを指定できる



パフォーマンス

- リードレプリカによってアプリケーションの読み込みトラフィックをスケールアウトさせることができる
- 読み込みエンドポイントによってリードレプリカを跨がって負荷が分散される

クラウドネイティブストレージエンジンの概要

データは**3つのアベイラビリティゾーン**に跨った**6つのレプリカ**にコピーされる

継続的に堅牢な **Amazon S3** へバックアップされる

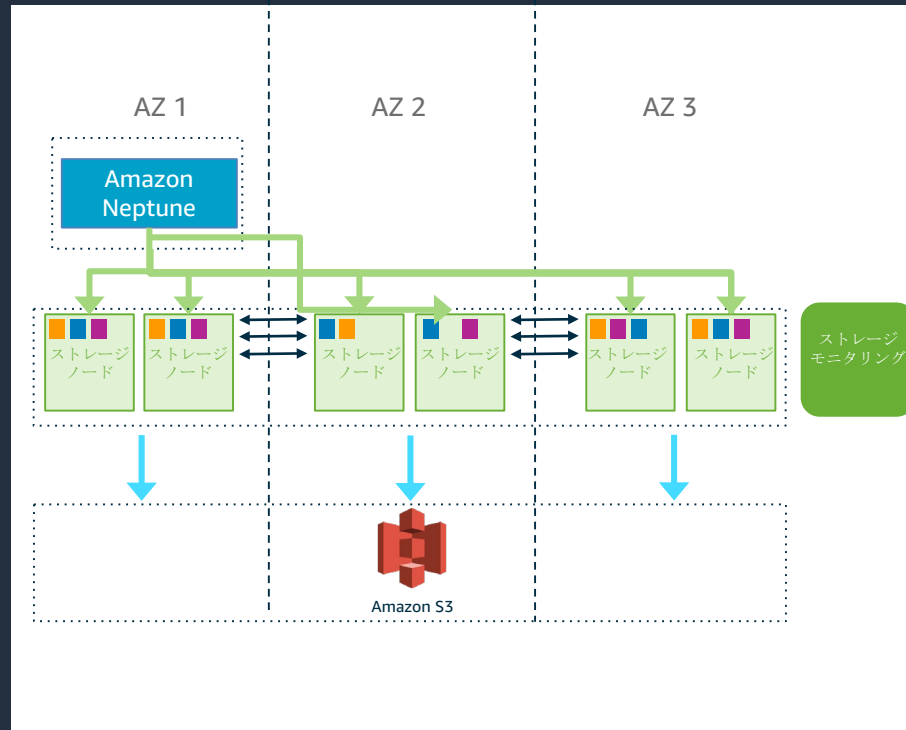
継続的に**ノードやディスクが修復**される

修復やホットスポットのリバランスのために**10GBのセグメントユニット**で管理されている

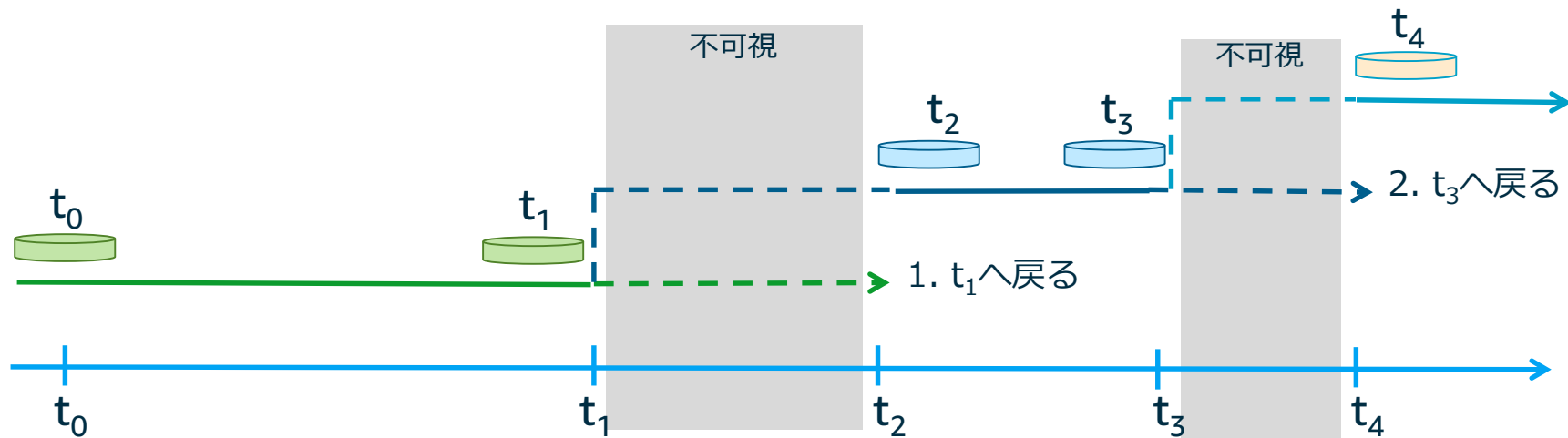
読み書きには**レイテンシ耐性を持つクォーラムシステム**を使用している

クォーラムのメンバーシップが変更されたとしても書き込みは**阻害されない**

ストレージは使用に応じて**自動的に64TBまで拡張**される



Point-in-Time リストア



オンラインでの point-in-time リストアにより、バックアップからリストアすることなく指定した時間にデータベースの状態を戻すことができる

セキュリティ

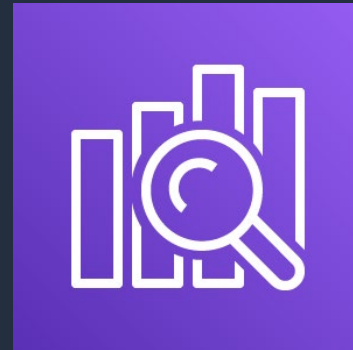
- TLSによるクライアント/サーバ間の通信暗号化
- KMSによるストレージの暗号化
- セキュリティグループによるアクセス制御
- AWS IAMロールによる認証
- 監査ログをCloudWatch Logsに発行
- SOC, PCI-DSSなど各種コンプライアンス準拠

https://docs.aws.amazon.com/ja_jp/neptune/latest/userguide/neptune-compliance.html

Amazon Neptuneでサポートされるインターフェイス

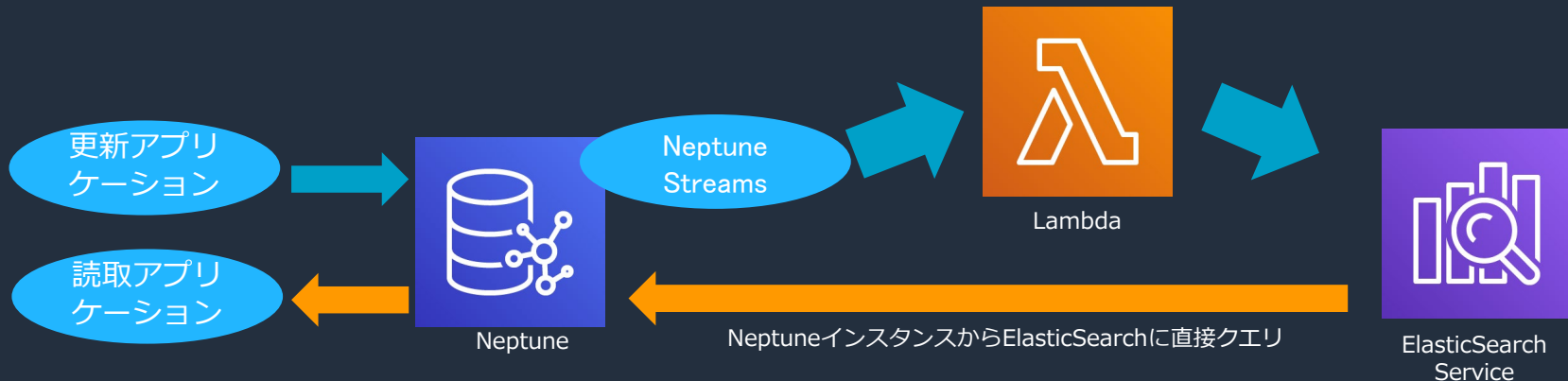
	Tinkerpop/Gremlin	RDF/SPARQL
CLI	Gremlin Console	RDF4J Console RDF4J Workbench
Java	Gremlin Driver	RDF4J
Python	Gremlin-Python Driver	sparqlwrapper
.NET	Gremlin.NET	-
Node.js	gremlin-javascript	-
HTTP REST	HTTP REST (Gremlin)	HTTP REST (SPARQL)
REST以外の HTTP	Gremlin HTTP WebSocket API	SPARQL HTTP

Amazon ElasticSearch Service連携



- Amazon ElasticSearch Serviceにデータをストリームで連携
- Amazon NeptuneからElasticSearchによる文字列検索が可能

Amazon ElasticSearch Serviceとの連携



グラフデータの更新をNeptune Streamsから検知、ElasticSearch Serviceを更新

この構成を構築するCloudFormationのテンプレートが用意

https://docs.aws.amazon.com/ja_jp/neptune/latest/userguide/full-text-search-cfn-create.html

neptune_streams パラメータの有効化が必要

ElasticSearch Service連携でLIKE検索やfuzzy検索

Ex1)

```
g.withSideEffect("Neptune#fts.endpoint", "your-es-endpoint-URL")  
  .withSideEffect("Neptune#fts.queryType", "query_string") Query String 検索  
  .withSideEffect("Neptune#fts.minScore", 1.5) Score 1.5以上のDocumentのみ  
  .V().has("name", "Neptune#fts m*k*"); ワイルドカード
```

Ex2)

```
g.withSideEffect("Neptune#fts.endpoint", "your-es-endpoint-URL")  
  .withSideEffect("Neptune#fts.queryType", "fuzzy") Fuzzy検索  
  .V().has("name", "Neptune#fts mike");
```

https://docs.aws.amazon.com/ja_jp/neptune/latest/userguide/full-text-search.html

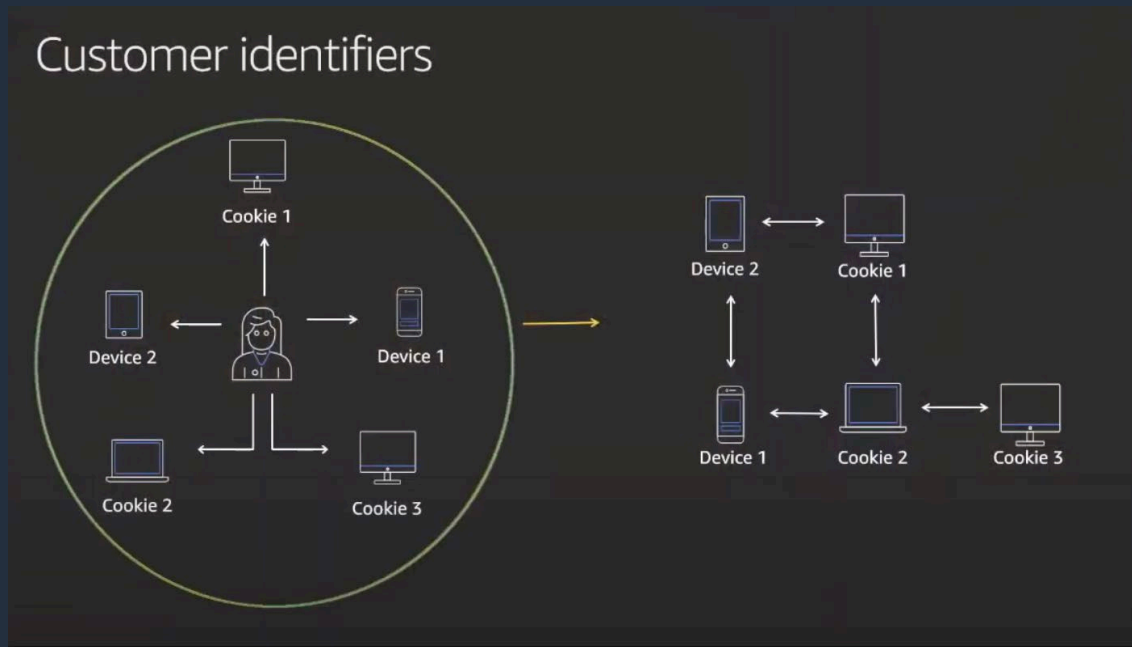
その他最近のアップデート

- データベースクラスタの停止が可能に
 - 最大停止期間は7日間
 - 7日間を超える場合はスナップショットを取得して削除を

- db.t3.mediumインスタンスが利用可能に
 - 0.15USD/時間で利用可能（東京リージョン）

事例: Zeta Global 顧客識別グラフ

- 顧客を識別するためのブラウザCookieなどの情報を管理
- 異なるCookieやDeviceIDが同じ顧客を表した場合にEdgeで結合
- 1日あたり4億5000万クエリ
- 平均35msecのレイテンシ



<https://aws.amazon.com/jp/blogs/news/building-a-customer-identity-graph-with-amazon-neptune/>

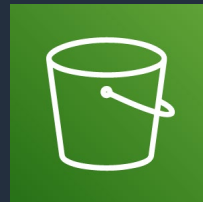
アジェンダ

1. グラフとは
2. グラフDBとは
3. Amazon Neptuneのご紹介
4. はじめるためには
5. まとめ

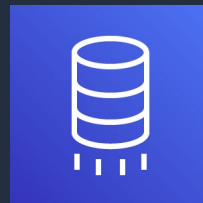
データの投入

バルクでデータを投入する方法は2種類

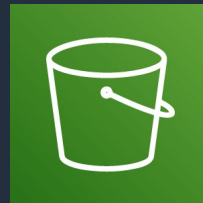
- S3上にCSVファイルを用意し、Neptune Loaderでロード



- AWS Database Migration Serviceを使用して他のDBから移行



S3上のCSVファイルによるデータロード



NodeのCSVファイル例

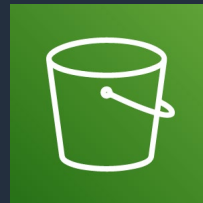
```
~id, name:String, age:Int, ~label  
v1, UserA, 29, person  
v2, UserB, 30, person
```

EdgeのCSVファイル例

```
~id,~from,~to,~label  
e1,v1,v2, follow  
e2,v2,v1, follow
```

https://docs.aws.amazon.com/ja_jp/neptune/latest/userguide/bulk-load.html

S3上のCSVファイルによるデータロード



NodeのCSVファイル例

```
~id, name:String, age:Int, ~label  
v1, UserA, 29, person  
v2, UserB, 30, person
```

EdgeのCSVファイル例

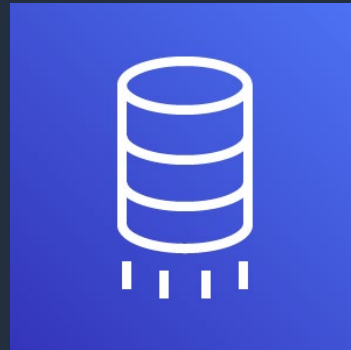
```
~id,~from,~to,~label  
e1,v1,v2, follow  
e2,v2,v1, follow
```

ロード実行例

```
curl -X POST -H 'Content-Type:  
application/json' https://<neptune-endpoint>:8182/loader -d  
{  
  "source" : "s3://<S3 URI>/",  
  "format" : "csv",  
  "iamRoleArn" : "<IAM Role ARN>",  
  "region" : "<region>",  
  "failOnError" : "FALSE",  
  "parallelism" : "MEDIUM",  
  "updateSingleCardinalityProperties" : "FALSE",  
  "queueRequest" : "TRUE"  
}
```

https://docs.aws.amazon.com/ja_jp/neptune/latest/userguide/bulk-load.html

AWS Database Migration Serviceによるデータ移行



RDSや、オンプレミスのRDBなどからデータの移行が可能
RDBのレコードを、ノードやエッジにマッピング
各カラムの値をプロパティに変換

https://docs.aws.amazon.com/ja_jp/neptune/latest/userguide/dms-neptune-replication.html

CLI (Gremlin Console)

Groovyベースのコンソール

用途: Gremlinの学習/アプリケーション開発/アドホック分析/データベース運用

```
$ cat conf/neptune-remote.yaml
hosts: [<Neptune end point>]
port: 8182
serializer: { className: (省略) }
```

サーバー定義

```
$ cat neptune.groovy
:remote connect tinkerpop.server conf/neptune-remote.yaml
:remote console
```

Neptuneに接続する初期化スクリプト

```
$ cat bin/mygremlin.sh
bin/gremlin.sh -i neptune.groovy
```

初期化スクリプトを呼び出す起動シェル

```
$ bin/mygremlin.sh
      ¥,,,/
      (o o)
-----o00o-(3)-o00o-----
plugin activated: tinkerpop.server
plugin activated:
tinkerpop.utilities
plugin activated:
tinkerpop.tinkergraph
gremlin> g.V().limit(1)
==>v[Luke]
gremlin>
```

起動シェルの実行

Neptuneへのクエリ



https://docs.aws.amazon.com/ja_jp/neptune/latest/userguide/access-graph-gremlin-console.html

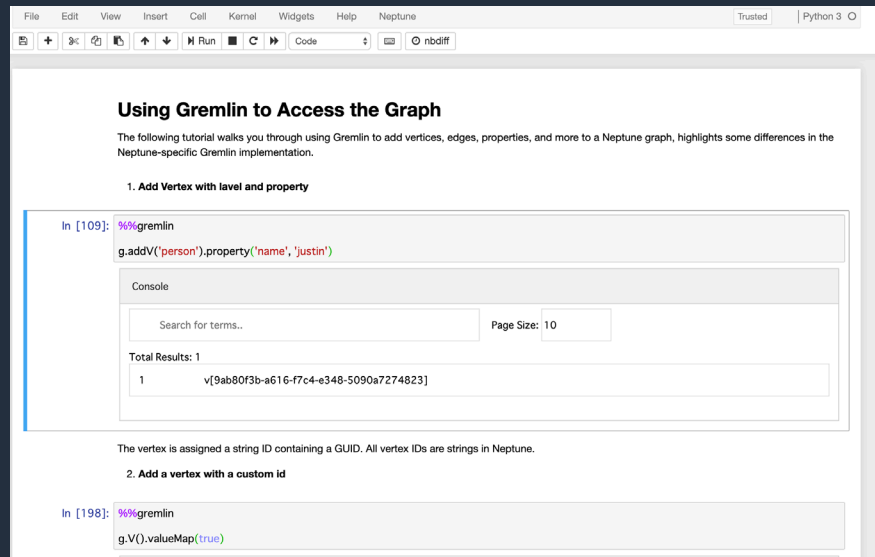
<http://tinkerpop.apache.org/docs/current/reference/#connecting-via-console>

<https://github.com/tinkerpop/gremlin/wiki/Getting-Started>



Neptune Workbench : Jupyter Notebookからクエリ実行

“%%sparql”や“%%gremlin”を記述したセルでクエリを実行



The screenshot shows a Jupyter Notebook interface within Neptune Workbench. The notebook title is "Using Gremlin to Access the Graph". The content includes an introductory paragraph and two numbered steps. Step 1, "Add Vertex with level and property", shows a code cell with the following code: `%%gremlin` and `g.addV('person').property('name', 'justin')`. Below the code is a console output showing a search for terms with a page size of 10, resulting in one vertex with ID `v[9ab80f3b-a616-f7c4-e348-5090a7274823]`. Step 2, "Add a vertex with a custom id", shows a code cell with the following code: `%%gremlin` and `g.V().valueMap(true)`.

File Edit View Insert Cell Kernel Widgets Help Neptune Trusted Python 3

Using Gremlin to Access the Graph

The following tutorial walks you through using Gremlin to add vertices, edges, properties, and more to a Neptune graph, highlights some differences in the Neptune-specific Gremlin implementation.

1. Add Vertex with level and property

```
In [109]: %%gremlin
g.addV('person').property('name', 'justin')
```

Console

Search for terms... Page Size: 10

Total Results: 1

1	v[9ab80f3b-a616-f7c4-e348-5090a7274823]
---	-----------------------------------------

The vertex is assigned a string ID containing a GUID. All vertex IDs are strings in Neptune.

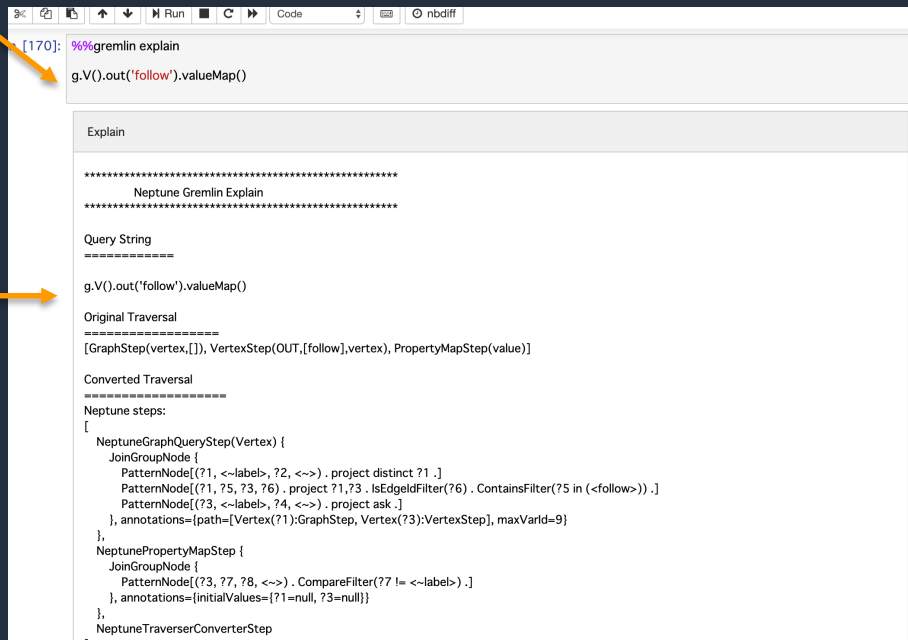
2. Add a vertex with a custom id

```
In [198]: %%gremlin
g.V().valueMap(true)
```


Gremlin 実行計画 の取得

Jupyter Notebookのセルで“%%gremlin explain”の後にクエリ記述

クエリの実行計画



```
[170]: %%gremlin explain
g.V().out('follow').valueMap()

Explain

*****
Neptune Gremlin Explain
*****

Query String
-----
g.V().out('follow').valueMap()

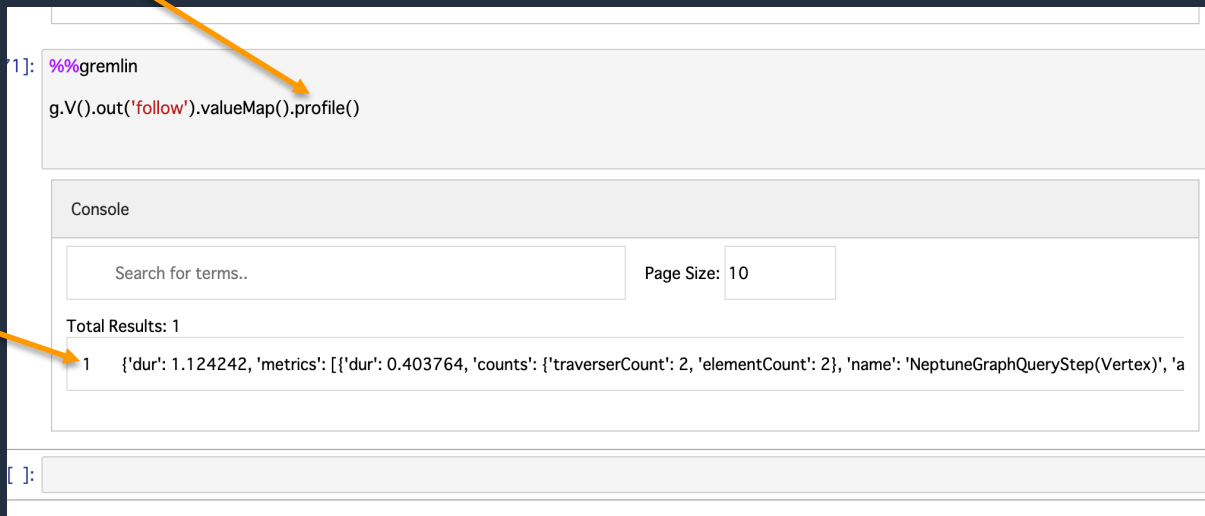
Original Traversal
-----
[GraphStep(vertex,[]), VertexStep(OUT,[follow],vertex), PropertyMapStep(value)]

Converted Traversal
-----
Neptune steps:
[
  NeptuneGraphQueryStep(Vertex) {
    JoinGroupNode {
      PatternNode[ (?1, <-label>, ?2, <->) . project distinct ?1 . ]
      PatternNode[ (?1, ?5, ?3, ?6) . project ?1,?3 . isEdgeIdFilter(?6) . ContainsFilter(?5 in (<follow>)) . ]
      PatternNode[ (?3, <-label>, ?4, <->) . project ask . ]
    }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep], maxVarId=9}
  },
  NeptunePropertyMapStep {
    JoinGroupNode {
      PatternNode[ (?3, ?7, ?8, <->) . CompareFilter(?7 != <-label> . )
    }, annotations={initialValues={?1=null, ?3=null}}
  },
  NeptuneTraverserConverterStep
```

Gremlin プロファイルの取得

クエリの最後に“.profile()”付与

クエリのプロファイル



The screenshot displays a Gremlin console interface. At the top, a query is entered: `g.V().out('follow').valueMap().profile()`. An orange arrow points from the text above to the `.profile()` part of the query. Below the query, the console shows the results of the profile. It includes a search bar, a page size of 10, and a table of results. The table has one row with the following data: `1`, `{'dur': 1.124242, 'metrics': [{'dur': 0.403764, 'counts': {'traverserCount': 2, 'elementCount': 2}, 'name': 'NeptuneGraphQueryStep(Vertex)', 'a`.

“%query_mode profile”マジックコマンドでもprofile取得可能

Neptune Workbench : その他の機能

各マジックコマンドでNeptuneの様々な機能をJupyterNotebookから実行

- `%status`
 - Neptuneクラスタの情報を取得して表示
- `%load`
 - S3上のファイルについてURI等を指定し、バルクロードを実行
- `%graph_notebook_config`
 - 使用中のJupyterNotebookの接続情報を表示
- `%query_mode (query|explain|profile)`
 - クエリモードの変更

可視化 - 何のための可視化か？



- 全体を見るための可視化

視覚的に全体的な構造を把握
クラスタリングの仮説を立てる
ドリルダウン

実トランザクションの大規模グラフ
ではすべてのデータを描画しても
情報過多となる



- 一部を見るための可視化

視覚的にターゲット周辺の構造を把握
トラバーサル仮説を立てる

大規模グラフのOLTP用途では
こちらが中心となる

OSSツールによる可視化

JavaScriptベースの可視化ツール“Graphexp”

GitHubリポジトリ以下のファイル形式をWebブラウザで開くだけ

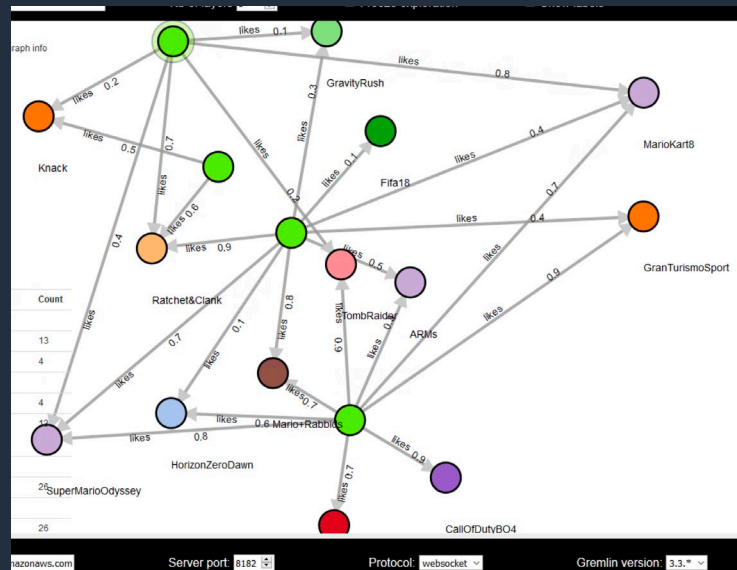
<https://github.com/bricaud/graphexp>

以下設定でNeptuneに対応

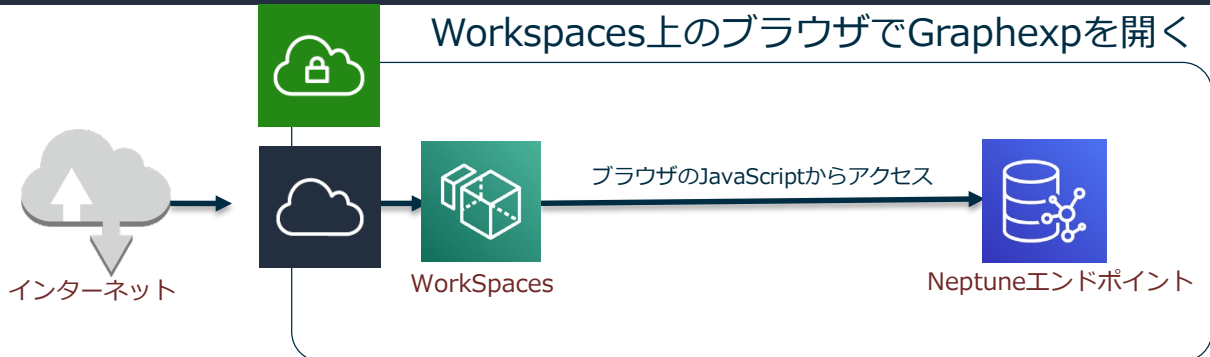
graphConf.js

```
const SINGLE_COMMANDS_AND_NO_VARS = true ;  
const REST_USE_HTTPS = true;
```

Webブラウザから直接Neptuneエンドポイントに
アクセスできる必要がある



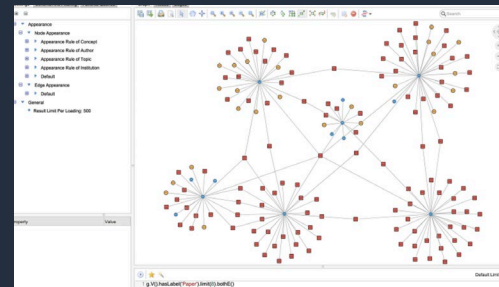
例



商用ツールによる可視化

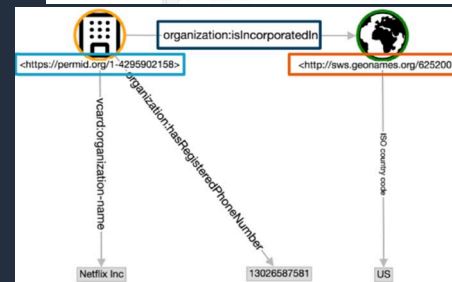
Tom Sawyer

<https://www.tomsawyer.com/aws-amazon-neptune-visualization/>
<https://aws.amazon.com/jp/blogs/news/exploring-scientific-research-on-covid-19-with-amazon-neptune-amazon-comprehend-medical-and-the-tom-sawyer-graph-database-browser/>



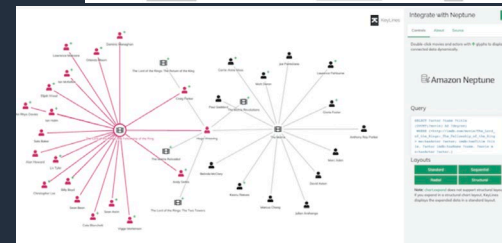
Metaphactory

<https://metaphacts.com/amazon-neptune>
<https://aws.amazon.com/jp/blogs/apn/exploring-knowledge-graphs-on-amazon-neptune-using-metaphactory/>

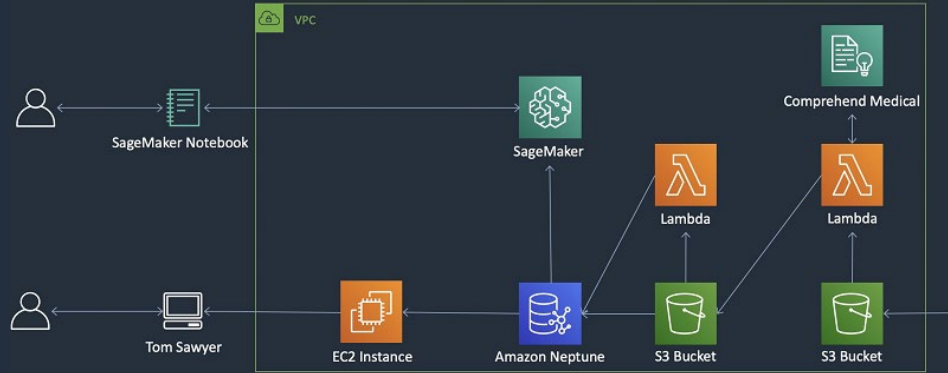


Keylines

<https://cambridge-intelligence.com/visualizing-the-amazon-neptune-database-with-keylines/>



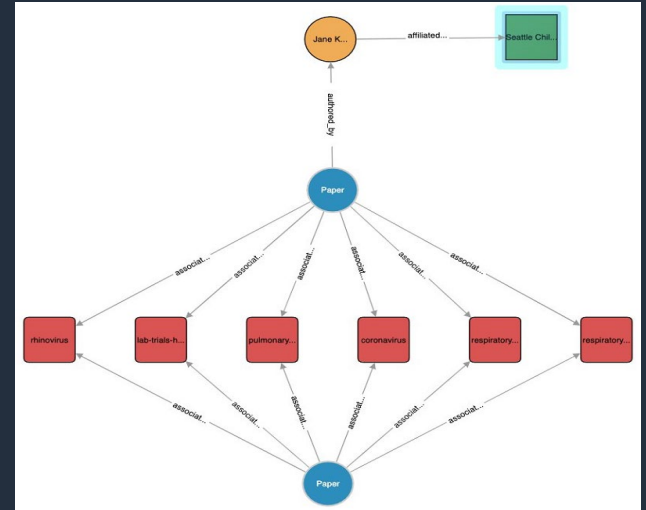
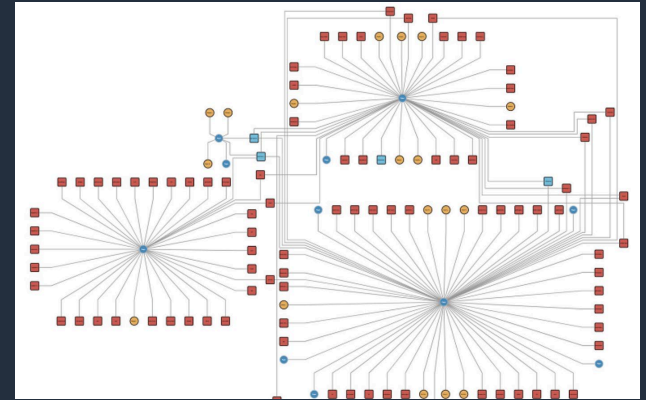
Tom Sawyer: ユースケース



Comprehend Medicalで医学論文から意味情報を抽出しつつトピックごとに分類し、Neptuneにロード

同じトピックに属する論文や、同じ意味情報の多くを共有する論文同士の類似関係を視覚化

<https://aws.amazon.com/jp/blogs/news/exploring-scientific-research-on-covid-19-with-amazon-neptune-amazon-comprehend-medical-and-the-tom-sawyer-graph-database-browser/>



運用ツール : GraphML 2 Neptune CSV

GraphML ファイルをNeptuneでのバルクロードが可能なCSV ファイルに変換するツール

<https://github.com/awslabs/amazon-neptune-tools/tree/master/graphml2csv>

TinkerpopのModernデータを変換する例

```
# TinkerpopのModernデータのうちGraphML形式の tinkerpops-modern.xml をダウンロード
$ curl https://raw.githubusercontent.com/apache/tinkerpop/master/data/tinkerpop-modern.xml -o tinkerpops-modern.xml
```

```
# Pythonスクリプトを実行してノードとエッジの2つのCSVファイルに変換
$ ./graphml2csv.py -i tinkerpops-modern.xml
infile = tinkerpops-modern.xml
Processing tinkerpops-modern.xml
Wrote 6 nodes and 18 attributes to tinkerpops-modern-nodes.csv.
Wrote 6 edges and 12 attributes to tinkerpops-modern-edges.csv.
```

```
# あとは変換されたCSVファイルをS3バケットに配置してNeptuneにバルクロードすればOK
```


運用ツール : Neptune Export

Neptuneのデータをテキストファイルにエクスポートするツール

<https://github.com/awslabs/amazon-neptune-tools/tree/master/neptune-export>

- BulkLoaderでNeptuneに取り込み可能なCSVやTurtleフォーマット
- 全データ、またはサンプリングやクエリの結果のみをエクスポート
- ツール自体をAWS Lambdaにデプロイすることも可能

```
# Mavenなどによりjarを作成
$mvn clean install
# endpointとoutputを指定して実行
$ bin/neptune-export.sh export-pg -d ~/neptune/output/ -e <neptune-endpoint> --use-ssl
```

[Gremlin] 学習パスの例

Neptune WorkbenchとGraphexpを用いてクエリを実行しながら学習

- Workbenchのサンプルノートブックを実行してみる
- AWS SamplesのTutorialに沿ってクエリを実行してみる
<https://github.com/aws-samples/amazon-neptune-samples/tree/master/gremlin/collaborative-filtering>
- TinkerpopのTutorialで学習する
<https://tinkerpop.apache.org/docs/current/tutorials/getting-started/>

※Neptuneにおける実装での相違点に留意しながら実行する

https://docs.aws.amazon.com/ja_jp/neptune/latest/userguide/access-graph-gremlin-differences.html

[RDF/SPARQL] 学習パスの例

- Workbenchのサンプルノートブックを実行してみる
- 公開されたグローバルなナレッジベースを用いてSPARQLを学習
例: DBPedia <http://ja.dbpedia.org/sparql>

既存のクエリを少しずつ改変しながら学ぶなどが有効。

アジェンダ

1. グラフとは
2. グラフDBとは
3. Amazon Neptuneのご紹介
4. はじめるためには
5. まとめ

まとめ

- グラフとは
 - SNS、路線図、WWW等、いたるところに
- Amazon Neptuneの特徴
 - 最大で毎秒100,000件のクエリをサポート
 - クラウドネイティブストレージによる高信頼性
 - 継続的バックアップ、Point-In-Timeリストア
 - TLSやKMSによる通信やストレージの暗号化
 - CloudWatchLogsへの監査ログの発行
 - ElasticSearch Serviceとの連携による、柔軟なクエリ
 - Neptune Workbenchや可視化による、高い開発効率

Q&A

- お答えできなかったご質問については
- AWS Japan Blog
「<https://aws.amazon.com/jp/blogs/news/>」にて
- 資料公開と併せて、後日掲載します。

ご清聴ありがとうございました