Neo4j overview

Benoit Simard (@logisima)

History

Leonhard Euler 1707-1783

Konigsbergs

Why Graph database ?

neo4j

Data complexity

neo4j

Linked Data

Relation in RDBMS

- schema not flexible
- poor performances on linked data (@see snowflake schema)
- complexe queries

Relation in No-SQL

0x235C	{name:Philip, UID: PPR, Groups: [CHI,SFO,BOS]}
0xCD21	{name:Neo4j Chicago, UID: PPR, Members:[PPR,RB,NL], where:{city:Chicago, State: IL}}

Document DB

MongoDB CouchDB

	Name	UID	Members	Groups	Photo
0x235C	Philip	PPR		CHI, SFO, BOS	B75DD108A893A
0xCD21	Neo4j Chicago	CHI	PPR,RB, NL		218758D88E901

Column Familv

HBase Cassandra

0x235C	Philip
0xCD21	Neo4j Chicago
0x2014	[PPR,RB,NL]
0x3821	[CHI, SFO, BOS]
0x3890	B75DD108A

Kev-Value

membase Riak Redis

Graph database

neo4j

When to use it ?

- Do you need a flexible schema ?
- Do your data are highly connected ?
- Do you need near real time ?

Recommendation, fraud detection, MDM, network analysis, impact analysis, ACL, …

neo4j

What is a Graph database ?

neo4j

Neo4j

- Since 2010
- License under GPLv3
- Full ACID
- HA
- Schemaless

Concept

```
Failed to generate image: Could not find the 'dot' executable in PATH; add it to the PATH or specify its location using the 'graphvi
zdot' document attribute
digraph finite_state_machine {
  rankdir=TB;
  node [
    shape = Mrecord,
  ];

    "Graph";
    "Nodes";
    "Relations";
    "Properties";

    "Graph" -> "Nodes" [ label = "STOCK DATA INTO" ];
    "Graph" -> "Relations" [ label = "STOCK DATA INTO"];
    "Nodes" -> "Properties" [label = "HAVE" ];
    "Relations" -> "Properties" [ label = "HAVE" ];
    "Relations" -> "Nodes" [label = "LINK" ];
}
```

Nodes

```
Failed to generate image: no implicit conversion of nil into String
digraph finite_state_machine {
  rankdir=LR;
  node [
    shape = Mrecord,
  ];

  "Nodes";
  "Properties";
  "Labels";

  Nodes -> Properties [ label = "CAN HAVE"];
  Nodes -> Labels [ label = "CAN HAVE"];
}
```

```
Failed to generate image: no implicit conversion of nil into String
digraph finite_state_machine {

    rankdir=LR;
    node [
         shape = Mrecord
    ];

    ACTOR [ label = "Person, Actor | name = \'Keanu Reeves\' \l birth = 1964" ];
}
```

Relationships

```
Failed to generate image: no implicit conversion of nil into String
digraph finite_state_machine {
  rankdir=LR;
  node [
    shape = Mrecord,
  ];

  "Relation";
  "Start node";
  "End node";
  "Properties";
  "Type";

  Relation -> Type [label = "HAS"];
  Relation -> "Start node" [label = "HAS"];
  Relation -> "End node" [label = "HAS"];
  Relation -> Properties [label = "CAN HAVE"];
}
```

```
Failed to generate image: no implicit conversion of nil into String
digraph finite_state_machine {

    rankdir=LR;
    node [
         shape = Mrecord
    ];

    ACTOR [ label = "Person, Actor | name = \'Keanu Reeves\' \l birth = 1964" ];
    MOVIE [ label = "Movie | nom = \'Matrix\' \l released = 1999 " ];

    ACTOR -> MOVIE [ label = "ACTED IN \l role = 'Thomas Anderson' " ];
}
```

Index & constraint

Indexes

- on a property for all nodes that have a label

Constraint

- for the uniqueness of a property
- for the existence of a property

Top use-cases

neo4j

Yeah !

(Graph)-[:ARE]→(Everywhere)

neo4j

Moteur de recommandation en temps réel

neo4j

La recommandation

Des réseaux sociaux, à la vente au détail, de service ou dans le secteur des médias, la recommandation est partout !

Objectif : offrir une recommandation en temps réel, hautement ciblés et contextuelles afin quelle soit pertinentes pour l'utilisateur

Etude de cas : Meetic

neo4j

La détection de fraudes

neo4j

Pourquoi du graphe ?

Bien qu’aucune mesure de détection de fraudes ne soit parfaite, il est possible d’introduire des améliorations considérables si l’on se penche sur les connexions qui relient les points de données individuels.

Il faut étudier l’interconnexion des données

Etude de cas : Fraude dans le e-commerce

Graphe d’une série de transactions effectuée avec fraude probable se produisant à l’adresse IP1.

(IP : Adresse IP, CC : carte de crédit, ID : Identifiant de l’utilisateur, CK : Adresse de livraison)

Etude de cas : La détection d’anneau

Constituer un ensemble de fausses identités à partir d’identités volées, dans l’objectif de commettre un acte frauduleux

La gestion de données de référence

neo4j

Reconciliation des données

Réconcialiation des données par degré de similitude

La modélisation RH d’une entreprise

Hiérarchie de données de référence illustrant des structures de rapport et de supervision d’employé.

La modélisation RH d’une entreprise

Réseau de données de référence détaillant les relations de rapport et de supervision d’employé, ici avec une plus grande complexité du **monde réel**.

neo4j

La gestion des identités et des accès

neo4j

La gestion des identités et des accès

Pourquoi du graphe ?

C’est un graphe !

Les réseaux informatiques

neo4j

Analyse d’impact chez SFR

Modélisation de leur reseau physique dans Neo4j.

How to query ?

neo4j

Cypher

Cypher is Neo4j’s graph query language

- Declarative Pattern-Matching language
- SQL-like syntax
- Designed for graphs

neo4j

All is pattern

neo4j

Ascii Art

Cypher using relationship 'likes'

Cypher

```
(a) -[:LIKES]-> (b)
```

© All Rights Reserved 2013 | Neo Technology, Inc.

Let see it in action

Indexes

```
CREATE INDEX ON :Person(name)
CREATE INDEX ON :Movie(title)
```

Query

```
MATCH (a)-[:ACTED_IN]->(m)<-[:DIRECTED]-(a)
RETURN a.name, m.title;
```

List all the characters in the movie “The Matrix”

```
MATCH (matrix:Movie)<-[r:ACTED_IN]-(actor)
WHERE matrix.title="The Matrix"
RETURN r.roles, actor.name
```

Query

```
MATCH (tom:Person)-[:ACTED_IN]->(movie:Movie)
WHERE tom.name="Tom Hanks"
AND movie.released < 1992
RETURN movie.title
```

Query

```
MATCH (actor:Person)-[r:ACTED_IN]->(movie)
WHERE actor.name="Keanu Reeves"
AND "Neo" IN (r.roles)
RETURN movie.title
```

Query

```
MATCH (tom:Person)-[:ACTED_IN]->(movie),
      (a:Person)-[:ACTED_IN]->(movie)
WHERE tom.name="Tom Hanks"
  AND a.born < tom.born
RETURN DISTINCT a.name, (tom.born - a.born) AS diff
```

Query

```
MATCH (keanu:Person)-[:ACTED_IN]->(movie),
      (n)-[:ACTED_IN]->(movie),
      (hugo:Person)
WHERE keanu.name="Keanu Reeves" AND
      hugo.name="Hugo Weaving"
AND NOT (hugo)-[:ACTED_IN]->(movie)
RETURN DISTINCT n.name
```

Who are the five busiest actors?

```
MATCH (a:Person)-[:ACTED_IN]->()
RETURN a.name, count(*) AS count
ORDER BY count DESC
LIMIT 5
```

Recommend 3 actors that Keanu Reeves should work with (but hasn't).

```
MATCH (keanu:Person)-[:ACTED_IN]->()<-[:ACTED_IN]-(c),
      (c)-[:ACTED_IN]->()<-[:ACTED_IN]-(coc)
WHERE keanu.name="Keanu Reeves"
  AND coc <> keanu
  AND NOT((keanu)-[:ACTED_IN]->()<-[:ACTED_IN]-(coc))
RETURN coc.name, count(coc)
ORDER BY count(coc) DESC
LIMIT 3
```

Add KNOWS relationships between all actors who acted in the same movie

```
MATCH (a:Person)-[:ACTED_IN]->()<-[:ACTED_IN]-(b:Person)
MERGE (a)-[:KNOWS]-(b);
```

Friends-of-Friends

```
MATCH (keanu:Person)-[:KNOWS*2]-(fof)
WHERE keanu.name="Keanu Reeves"
RETURN DISTINCT fof.name
```

Query

```
MATCH (keanu:Person)-[:KNOWS*2]-(fof)
WHERE keanu.name="Keanu Reeves"
  AND keanu <> fof AND NOT (keanu)-[:KNOWS]-(fof)
RETURN DISTINCT fof.name
```

Query

```
MATCH p=shortestPath(
  (charlize:Person)-[:KNOWS*]-(bacon:Person)
)
WHERE charlize.name="Charlize Theron" AND
      bacon.name="Kevin Bacon"
RETURN length(rels(p))
```

Return the names of the people joining Charlize to Kevin.

```
MATCH p=shortestPath((charlize:Person)-[:KNOWS*]-(bacon:Person))
WHERE charlize.name="Charlize Theron" AND
      bacon.name="Kevin Bacon"
RETURN [n in nodes(p) | n.name] AS names;
```

QUESTION ?

neo4j

PROCHAIN ÉVÈNEMENTS

- **Graph Day Paris :** mercredi 21 Septembre
- **Training Graph Modeling :** mardi 25 Octobre à Paris

neo4j