Neo4j
The World's Leading Graph Database

Neo4j
Aprendendo conceitos por trás do Neo4j

Universidade Federal do Paraná - UFPR

Programa de Pós-Graduação em Informática - PPGInf

Oficina de Banco de Dados - CI829

- Profa: Dra. Carmem Hara
- Aluno: Walmir Couto

Sem SQL?

Neo4j
The World's Leading Graph Database

Apresentação

Neo4j
The World's Leading Graph Database

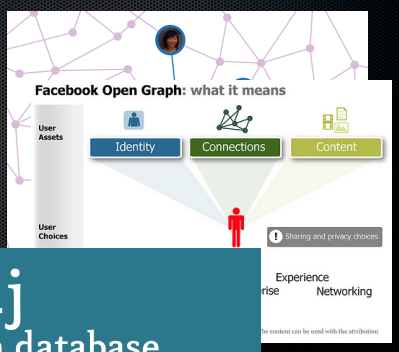
- Por que grafos, por que agora?
- O que é um graph database;
- Overview NOSQL;
- Graph databases e Neo4j;
- Como usamos o Neo4j;
- Conclusões

Por que grafos, por que agora?

Neo4j
The World's Leading Graph Database

1. Big Data é uma tendência;
2. NOSQL é a resposta;
3. Grande em volume e em densidade;

✓ Todo mundo está falando em grafos;



Neo4j
the graph database

Neo4j
The World's Leading Graph Database

Um Grafo?
Sim, um grafo!

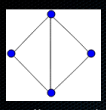
Um grafo...

Neo4j
The World's Leading Graph Database

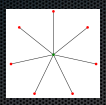
- Você conhece as estruturas de dados comuns
  - linked lists, árvores, “grafos” objeto
- Grafo é uma estrutura de dados de propósito geral
  - adequado para qualquer dado conectado
- Algoritmos e padrões bem conhecidos
  - estudado desde as 7 pontes de Leonard Euler (1736)
  - Modelo Relacional de Codd (1970)
  - não é uma ideia nova, apenas uma ideia para o momento atual

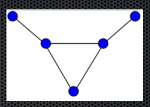
Alguns grafos bem conhecidos...

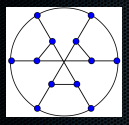
Neo4j
The World's Leading Graph Database

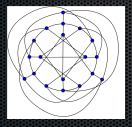

- dimond

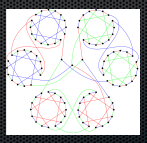

- butterfly

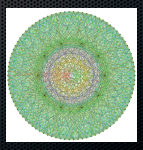

- star

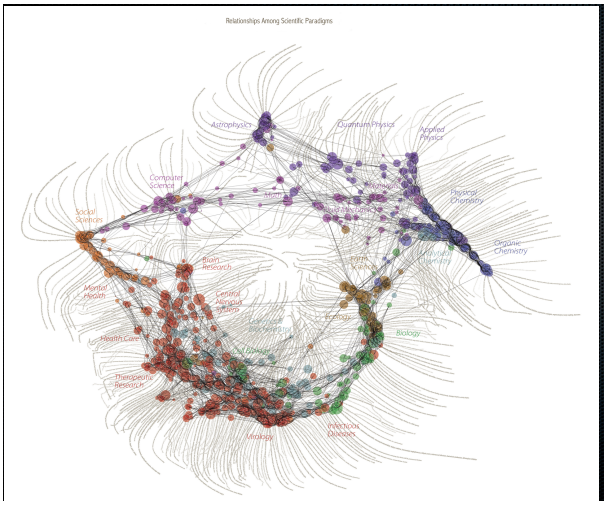

- bull


- franklin


- robertson


- horton


- hall-janko



Neo4j
The World's Leading Graph Database

- grafo kollegger

(todas as possibilidades de grafo)

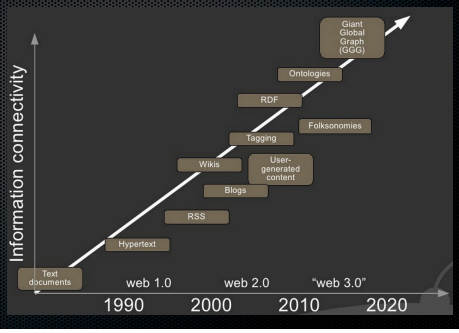
Um graph database:

- Otimizado para as conexões entre os registros
- Extremamente rápido nas consultas através dos registros
- Um banco de dados: transacional com as operações usuais
  - "Um banco de dados relacional pode dizer a você a média de idade de todos que estão nesta reunião... mas um graph database vai te dizer quem tem mais chances de comprar uma cerveja (por exemplo)."
  - "Você conhece um relacional...
  - "Então agora considere apenas as relações!"

As 4 tendências...

As 4 tendências:

- Tendência 1: o tamanho do conjunto de dados (crescimento vertiginoso)
- Tendência 2: conectividade



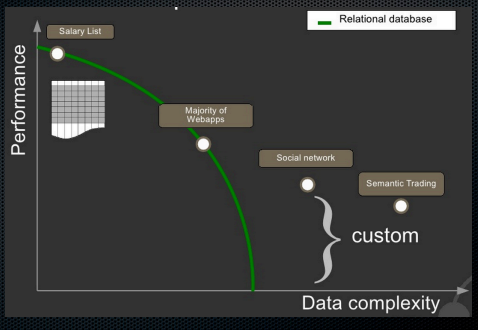
As 4 tendências:

- Tendência 3: semi-estrutura
  - Individualização do conteúdo
    - Na listas de salários dos anos 70, todos os elementos tinham exatamente uma ocupação
    - Na listas de salários dos anos 2000, nós precisamos de 5 colunas de ocupação! Ou 8? Ou 15?
  - A acelerada tendência pela descentralização da geração de conteúdo é a marca da era da participação ("web 2.0")

As 4 tendências:

Neo4j
The World's Leading Graph Database

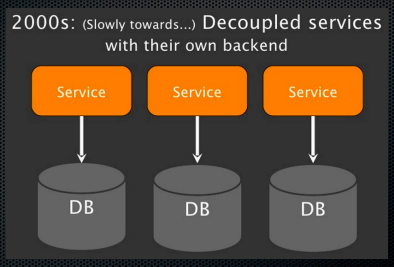
- De um lado: o desempenho dos RDBMS



As 4 tendências:

Neo4j
The World's Leading Graph Database

- Tendência 4: arquitetura



Overview NOSQL:

Neo4j
The World's Leading Graph Database

- Tendência 1: Tamanho
- Tendência 2: Conectividade
- Tendência 3: Semi-estrutura
- Tendência 4: Arquitetura

Overview NOSQL:

Neo4j
The World's Leading Graph Database

- Primeiro de tudo: o nome
  - NoSQL não é “Never SQL”
  - NoSQL não é “No To SQL”
  - NoSQL é simplesmente “Not Only SQL!”

Overview NOSQL:

Neo4j
The World's Leading Graph Database

- 4 Categorias emergentes no NOSQL...
  - Key-value stores
    - Baseados no artigo Dynamo da Amazon
    - Modelo de dados: coleção (global) de pares de K-V (key-value / chave-valor)
    - Exemplos: Dynomite, Voldemort, Tokyo
  - ColummFamily / BigTable clones
    - Baseados no artigo BigTable da Google
    - Modelo de dados: big table, famílias de colunas
    - Exemplos: HBase, Hypertable, Cassandra

key	value
firstName	Bugs
lastName	Bunny
location	Earth

APACHE HBASE

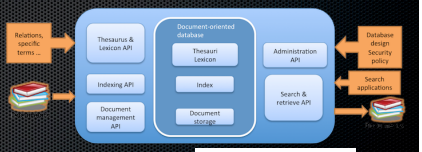
HYPERTABLE

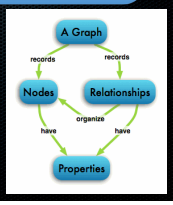
Cassandra

Overview NOSQL:

Neo4j
The World's Leading Graph Database

- 4 Categorias emergentes no NOSQL...
  - Document databases
    - Inspirado no Lotus Notes
    - Modelo de dados: coleções de coleções de K-V (key-value / chave-valor)
    - Exemplos: CouchDB, MongoDB, Riak
  - Graph databases
    - Inspirados por Euler e a teoria dos grafos
    - Modelo de dados: nós, arestas, K-V (chave-valor) em ambos
    - Exemplos: AllegroGraph, Sones, Neo4j

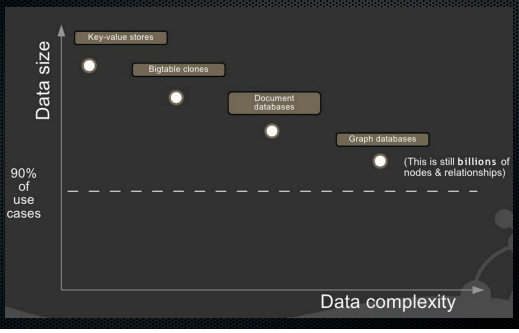




Overview NOSQL:

Neo4j
The World's Leading Graph Database

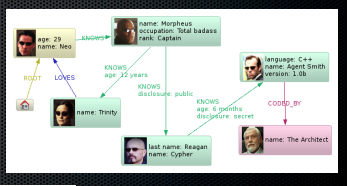
- Modelos de dados NOSQL...

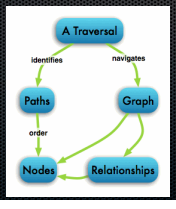


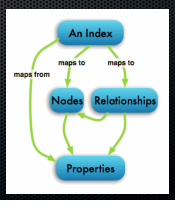
Graph DBs e Neo4j:

Neo4j
The World's Leading Graph Database

- O modelo do Graph DB: representação
  - Abstrações de núcleo:
    - Nós (nodes)
    - Relacionamentos entre os nós
    - Propriedades de ambos







Graph DBs e Neo4j:

- Código (1): construindo um node space

```
GraphDatabaseService graphDb = ... // Get factory


// Create Thomas 'Neo' Anderson
Node mrAnderson = graphDb.createNode();
mrAnderson.setProperty( "name", "Thomas Anderson" );
mrAnderson.setProperty( "age", 29 );

// Create Morpheus
Node morpheus = graphDb.createNode();
morpheus.setProperty( "name", "Morpheus" );
morpheus.setProperty( "rank", "Captain" );
morpheus.setProperty( "occupation", "Total bad ass" );

// Create a relationship representing that they know each other
mrAnderson.createRelationshipTo( morpheus, RelTypes.KNOWS );
// ...create Trinity, Cypher, Agent Smith, Architect similarly
```

Graph DBs e Neo4j:

- Código (1): construindo um node space

```
GraphDatabaseService graphDb = ... // Get factory
Transaction tx = graphDb.beginTx();

// Create Thomas 'Neo' Anderson
Node mrAnderson = graphDb.createNode();
mrAnderson.setProperty( "name", "Thomas Anderson" );
mrAnderson.setProperty( "age", 29 );

// Create Morpheus
Node morpheus = graphDb.createNode();
morpheus.setProperty( "name", "Morpheus" );
morpheus.setProperty( "rank", "Captain" );
morpheus.setProperty( "occupation", "Total bad ass" );

// Create a relationship representing that they know each other
mrAnderson.createRelationshipTo( morpheus, RelTypes.KNOWS );
// ...create Trinity, Cypher, Agent Smith, Architect similarly

tx.commit();
```

Graph DBs e Neo4j:

- Código (1b): definindo tipos de relacionamentos

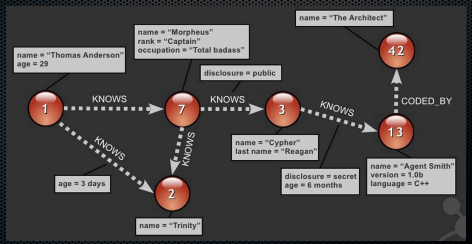
```
// In package org.neo4j.graphdb
public interface RelationshipType
{
   String name();
}

// In package org.yourdomain.yourapp
// Example on how to roll dynamic RelationshipTypes
class MyDynamicRelType implements RelationshipType
{
   private final String name;
   MyDynamicRelType( String name ){ this.name = name; }
   public String name() { return this.name; }
}

// Example on how to kick it, static-RelationshipType-like
enum MyStaticRelTypes implements RelationshipType
{
   KNOWS,
   WORKS_FOR,
}
```

Graph DBs e Neo4j:

- O modelo do Graph DB: traversal
  - Traverser framework para percorrer em alta performance através do node space



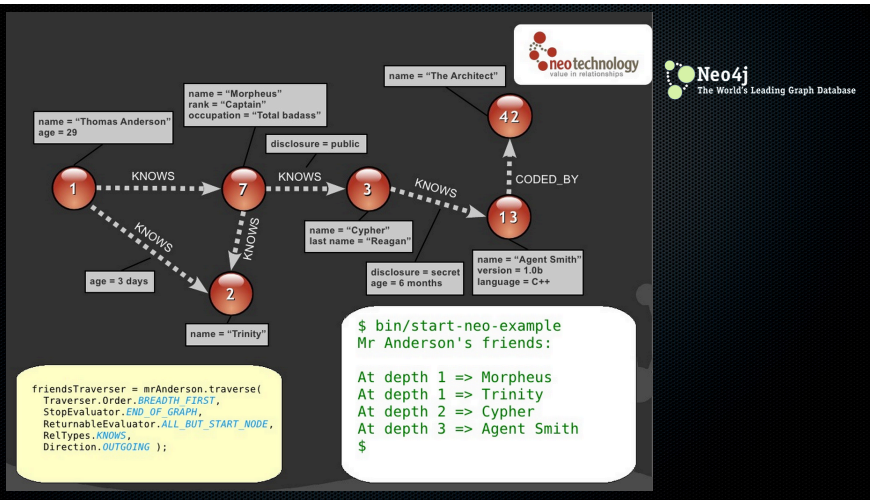
Graph DBs e Neo4j:

Neo4j
The World's Leading Graph Database

- Código (2): percorrendo um node space

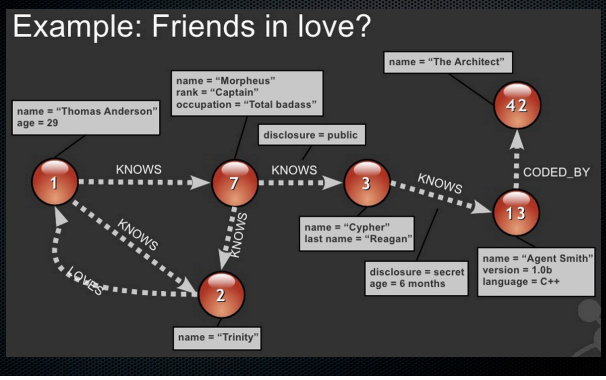
```
// Instantiate a traverser that returns Mr Anderson's friends
Traverser friendsTraverser = mrAnderson.traverse(
    Traverser.Order.BREADTH_FIRST,
    StopEvaluator.END_OF_GRAPH,
    ReturnableEvaluator.ALL_BUT_START_NODE,
    RelTypes.KNOWS,
    Direction.OUTGOING );

// Traverse the node space and print out the result
System.out.println( "Mr Anderson's friends:" );
for ( Node friend : friendsTraverser )
{
    System.out.printf( "At depth %d => %s%n",
        friendsTraverser.currentPosition().getDepth(),
        friend.getProperty( "name" ) );
}
```



Graph DBs e Neo4j:

Neo4j
The World's Leading Graph Database



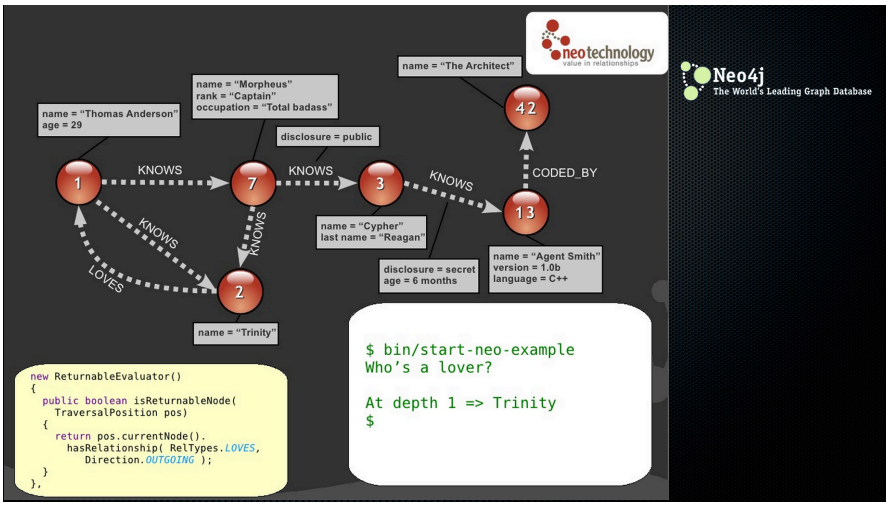
Graph DBs e Neo4j:

Neo4j
The World's Leading Graph Database

- Código (3a): percorrendo de forma personalizada

```
// Create a traverser that returns all "friends in love"
Traverser loveTraverser = mrAnderson.traverse(
    Traverser.Order.BREADTH_FIRST,
    StopEvaluator.END_OF_GRAPH,
    new ReturnableEvaluator()
    {
        public boolean isReturnableNode( TraversalPosition pos )
        {
            return pos.currentNode().hasRelationship(
                RelTypes.LOVES, Direction.OUTGOING );
        }
    },
    RelTypes.KNOWS,
    Direction.OUTGOING );
```

```
// Traverse the node space and print out the result
System.out.println( "Who's a lover?" );
for ( Node person : loveTraverser )
{
    System.out.printf( "At depth %d => %s%n",
        loveTraverser.currentPosition().getDepth(),
        person.getProperty( "name" ) );
}
```



Graph DBs e Neo4j:

Neo4j
The World's Leading Graph Database

- Bonus code: modelo de domínio
  - Como você implementa o seu modelo de domínio?
  - Use o padrão de delegação, isto é, qualquer domínio de entidade envolve um primitivo Neo4j:

```
// In package org.yourdomain.yourapp
class PersonImpl implements Person
{
   private final Node underlyingNode;
   PersonImpl( Node node ){ this.underlyingNode = node; }

   public String getName()
   {
      return (String) this.underlyingNode.getProperty( "name" );
   }
   public void setName( String name )
   {
      this.underlyingNode.setProperty( "name", name );
   }
}
```

Graph DBs e Neo4j:

Neo4j
The World's Leading Graph Database

- Domínio de camadas de frameworks
  - Qi4j (www.qi4j.org)
    - Framework para fazer DDD (Domain Driven Design) no Java5 puro
    - Define Entidades / Associações / Propriedades (parece familiar?) Nodes / Relacionamentos / Propriedades
    - Neo4j é um "EntityStore" por trás
  - Jo4neo (http://code.google.com/p/jo4neo)
    - Orientado a notação
    - Entrelaça a persistência apoiada no Neo4j dentro de um domínio de objetos em tempo de execução

Graph DBs e Neo4j:

Neo4j
The World's Leading Graph Database

- Caracterísiticas do sistema Neo4j
  - Baseado em Disco
    - Motor de armazenamento de grafo nativo com customização binária no formato em disco
  - Transacional
    - JTA/JTS, XA, 2PC, Tx recovery, deadlock detection, MVCC, etc.
  - Escalável
    - Vários bilhões de nodos/relac/prop em uma simples JVM
  - Baseado em Disco
    - 6+ anos em produção 24/7

Graph DBs e Neo4j:

- Prós e Contras comparado aos RDBMS
  - + Pode facilmente envolver schemas
  - + Pode representar informações semi-estruturadas
  - + Pode representar grafos/redes (com desempenho)
  - - Carência em ferramentas e framework de suporte
  - - Sem suporte para queries ad-hoc

Graph DBs e Neo4j:

- Linguagens de Query
  - SPARQL - "SQL for linked data"

Ex:
```
"SELECT ?person WHERE {
      ?person neo4j:KNOWS ?friend .
      ?friend neo4j:KNOWS ?foe .
      ?foe neo4j:name "Larry Ellison" .
 }"
```

  - Gremlin - "perl for graphs"

Ex: `"./outE[@label='KNOWS']/inV[@age > 30]/@name"`

    - Saiba mais em: http://try.neo4j.org

Graph DBs e Neo4j:

- O ecossistema Neo4j
  - Neo4j é um database embutido
  - Componentes do ecossistema:
    - index
    - meta-modelo
    - graph-matching
    - remote-graphdb
    - sparql-engine
    - ...
  - Veja mais em: http://components.neo4j.org

Graph DBs e Neo4j:

- Escalar - Replicação
  - Teste beta do Neo4j HA
  - Replicação master-slave (primeira configuração)
    - Estilo MySQL
    - Exceto que todas instâncias podem escrever, sincronismo entre a escrita slave & master (consistência forte)
    - Atualizações são propagadas assincronamente para outros slaves (consistência eventual)
  - É possível manipular bilhões de entidades.... mas não 100B

Graph DBs e Neo4j:

- Escalar - Particionando
  - Particionamento transparente? Previsão para o Neo4j 2.0
    - 100B? Fácil dizer... extremamente difícil fazer!
    - Fundamentais: BASE & consistência eventual