

**Universidade do Minho**

Mestrado Integrado em Engenharia Informática

**Unidade Curricular de**

**Bases de Dados NOSQL**

Ano Letivo de 2018/2019

**Migração de uma BD**

**<<Marco Barbosa, Manuel Moreno, Ricardo Certo, Sara Pereira >>**

<<Janeiro, 2019>>

|  |  |
| --- | --- |
| Data de Recepção |  |
| Responsável |  |
| Avaliação |  |
| Observações |  |

**<<Migração de uma BD>>**

**>>**

**<<Marco Barbosa, Manuel Moreno, Ricardo Certo, Sara Pereira >>**

<<Janeiro, 2019>>

<</opcional Dedicatória>>

# Resumo

<<O resumo tem como objectivo descrever de forma sucinta o trabalho realizado. Deverá conter uma pequena introdução, seguida por uma breve descrição do trabalho realizado e terminando com uma indicação sumária do seu estado final. Não deverá exceder as 400 palavras.>>

**Área de Aplicação:** <<Identificação da Área de trabalho. Por exemplo: Desenho e arquitectura de Sistemas de Bases de Dados.>>

**Palavras-Chave:** <<Conjunto de palavras-chave que permitirão referenciar domínios de conhecimento, tecnologias, estratégias, etc., directa ou indirectamente referidos no relatório. Por exemplo: Bases de Dados Relacionais, Gestão de Índices, JAVA, Protocolos de Comunicação.>>

# Índice

[1. Introdução 1](#_Toc535645383)

[1.1. Contextualização 1](#_Toc535645384)

[1.2. Apresentação do Caso de Estudo 1](#_Toc535645385)

[1.3. Motivação e Objectivos 1](#_Toc535645386)

[1.4. Estrutura do Relatório 1](#_Toc535645387)

[2. Sugestões para Escrita do Relatório 2](#_Toc535645388)

[2.1. Sugestões Gerais 2](#_Toc535645389)

[2.2. Termos Estrangeiros 2](#_Toc535645390)

[2.3. Tabelas e Figuras 2](#_Toc535645391)

[2.4. Siglas e Acrónimos 3](#_Toc535645392)

[2.5. Referências Bibliográficas 3](#_Toc535645393)

[2.6. Tipo de Ficheiro 3](#_Toc535645394)

[3. Conclusões e Trabalho Futuro 4](#_Toc535645395)

[Bibliografia 5](#_Toc535645396)

[Referências WWW 6](#_Toc535645397)

[Lista de Siglas e Acrónimos 7](#_Toc535645398)

**Anexos**

[I. Anexo 1 9](#_Toc535644882)

# Índice de Figuras

[Figura 1 - Ilustração de inserção de uma figura e legenda. 3](#_Toc535433491)

# Índice de Tabelas

[Tabela 1 - Ilustração de inserção de uma tabela e sua legenda. 3](#_Toc535433540)

1. Introdução

<<Este primeiro capítulo deverá ter obrigatoriamente as subsecções abaixo apresentadas.>>

* 1. Contextualização

<<Nesta secção deverá ser apresentado o contexto no qual se desenvolve o caso de estudo seleccionado.>>

* 1. Apresentação do Caso de Estudo

<<Esta secção acolherá uma descrição concisa do caso de estudo seleccionado.>>

* 1. Motivação e Objectivos

Na realização deste projeto foi-nos proposta a migração da BD sakila do MySQL para uma BD implementada em ORACLE, para uma uma em MongoDB e para uma em NEO4J. Este trabalho representa alguma complexidade a nível de migração das BD pois requer que possamos um conhecimento posterior sobre base de dados relacionais e não relacionais.

Com o desenvolvimento deste projeto pretendemos aprender a identificar quais sistemas de base de dados devemos utilizar dependendo da informação que lá pretendemos guardar. Pois, todos estes SGBD possuem as suas vantagens e as suas desvantagens que irão ser mencionadas numa fase mais adiante deste relatório.

Assim sendo, contruímos as novas bases de dados de acordo com as boas práticas definidas na documentação das mesmas. Podemos assim entender como funcionam os diferentes sistemas de base de dados e assim decidirmos em futuras implementações qual destes sistemas se adequa mais com o que pretendemos.

* 1. Estrutura do Relatório

Este relatório encontra-se organizado e distribuído pelos seguintes capítulos:

* 1º Capítulo: Introdução acerca da importância do uso de Base de Dados. Apresentação do caso em estudo e respetivas formas de o abordar.
* 2º Capítulo: Apresenta uma breve explicação das características de uma Base de Dados Relacional, e de seguida faz uma comparação das diferenças entre os diferentes tipos que esta pode ser.
* 3º Capítulo: Este capítulo faz referência a um tipo de base de dados não relacional, o Neo4j e compara-o com as bases de dados relacionais e enumera as diferenças entre ambos.
* 4º Capítulo: Este capítulo faz referência a um outro tipo de base de dados não relacional, o MongoDB e compara-o com as bases de dados relacionais e enumera as diferenças entre ambos.
* 5º Capítulo: No último capítulo apresentamos a conclusão acerca do trabalho realizado e descrito nos capítulos anteriores.

1. Base de Dados Relacionais
   1. SQL

O SQL é uma linguagem de programação que foi criada na década de 70 pela IBM para as Bases de Dados Relacionais, com o objetivo de implementar as regras de relacionamento das mesmas. Esta linguagem caracteriza-se por ser uma linguagem declarativa, isso significa que o programador necessita apenas de indicar qual o objetivo pretendido que seja executado pelo SGBD. Alguns dos principais comandos SQL para manipulação de dados são: INSERT (inserção), SELECT (consulta), UPDATE (atualização), DELETE (exclusão). SQL possibilita ainda a criação de relações entre tabelas e o controle do acesso aos dados.

* 1. Oracle (PL-SQL)

A oracle em 1991, começou por introduzir o PL-SQL que é uma linguagem de programação procedural. Ela surgiu com a necessidade da Oracle de ter uma linguagem interna, já que o SQL tem a função objetivo as pesquisa**(??????)**, recuperar e formatar dados de forma simples. Também possibilita o desenvolvimento de programas que são armazenados, compilados e executados dentro do próprio do próprio servidor da base de dados Oracle. É tipicamente utilizado para a criação de aplicações que requerem alto desempenho na execução das suas tarefas e quando pretendemos manipular grandes volumes de dados.

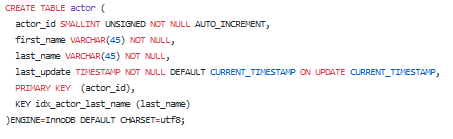
* 1. Diferenças

Uma das dúvidas mais frequentes no mundo das bases de dados relacionais é reconhecer a diferença entre SQL e PL/SQL. Essa diferença começa-se a evidenciar no nome pois PL/SQl (Procedual Language extensions to SQL) consiste em uma extensão da linguagem SQL (Structured Query Language). O PL/SQL como é executado dentro do sistema Oracle, garante assim que todos os dados manipulados não saiam do SGBD (Sistema Geral de Base de Dados) e sendo assim não precisam de recorrer a API’s externas, o que não acontece no SQL.

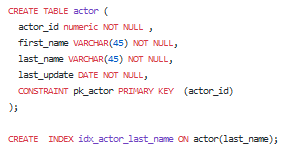
De seguida, vamos apresentar as diferenças entre a criação de uma tabela e a elaboração de uma view usando SQL e PL/SQL

**Criação de uma tabela:**

* SQL:



* PL/SQL:

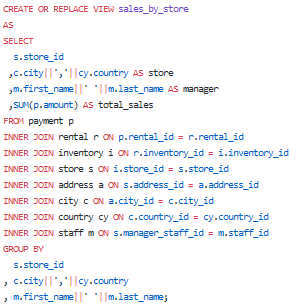


**Criação de uma view:**

* SQL:



* PL/SQL:



Através da visualização das imagens acima podemos verificar as pequenas diferenças que existem na sintaxe das duas linguagens, isso acontece devido a uma derivar da outra.

* 1. Vantagens e Desvantagens

Neste subcapítulo vamos apresentar as vantagens e desvantagens de cada uma linguagem.

**SQL:**

* Vantagens:

- Linguagem padrão ANSI para a manipulação de BD

- Oferece instruções para recuperação e manipulação de dados em tabelas

- Controla transações e acessos

* Desvantagens:

- Limitação por ser uma linguagem declarativa, isto significa que não é possível criar um programa inteiro em SQL.

**PL/SQL:**

* Vantagens:

- Orientado a objetos

- Possui controlo de fluxo e tratamento de exceções

- Linguagem de programação complexa

* Desvantagens:

- Tecnologia proprietária exclusiva dos serviços da Oracle.

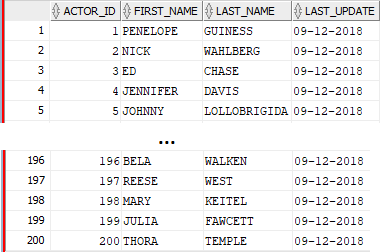
* 1. Processo de Migração

Este processo de migração foi feito de uma forma simples, uma vez que se trata de uma migração de uma BD relacional para outra Relacional. Ou seja, para a efetuarmos tivemos de consultar a sintaxe da Oracle pois apesar de serem linguagens parecidas possuem as suas diferenças. Com base nisso efetuamos a migração de tabelas, funções e views para o novo sistema. Esta nova implementação foi realizada usando SQL Developer da Oracle. No subcapítulo seguinte vamos apresentar um teste ao novo sistema de base de dados de modo a comprovar que os resultados por nós obtidos foram os pretendidos.

* 1. Teste ao novo sistema de BD

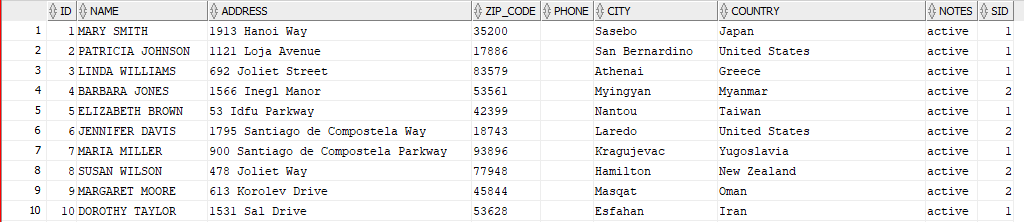
Neste subcapítulo vamos efetuar um teste ao novo sistema de BD e para isso vamos demonstrar que este sistema apresenta os resultados pretendidos. Este teste vai consistir na demonstração do conteúdo da tabela actor e a execução de algumas views.

Como podemos verificar através da imagem seguinte a tabela actor possui as mesmas 200 entradas que possuía antes da migração.

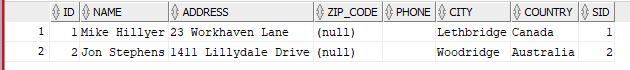


Com base na próxima imagem podemos afirmar que as views apresentadas funcionam, logo podemos afirmar que o sistema foi migrado corretamente.

View Costumers List:



View Staff List:



1. Base de Dados não Relacionais – Neo4j
   1. Neo4j

Neo4j trata-se de uma base de dados não relacional que é representada em grafos, constituídos por nós e arestas que ligam os mesmos, designadas por relacionamentos. A linguagem utilizada para a implementação das queries semânticas é *Cypher.*

Profissionalmente, estes tipos de bases de dados preenchem certos requisitos como o facto de conseguir lidar com o crescimento do número de relações, mantendo a sua performance constante. Ao mesmo tempo, apresentam flexibilidade na medida em que não é necessário fazer adaptações na BD existente, apenas adicionar informação, sendo o programador quem dita as regras e não o modelo RDBMS

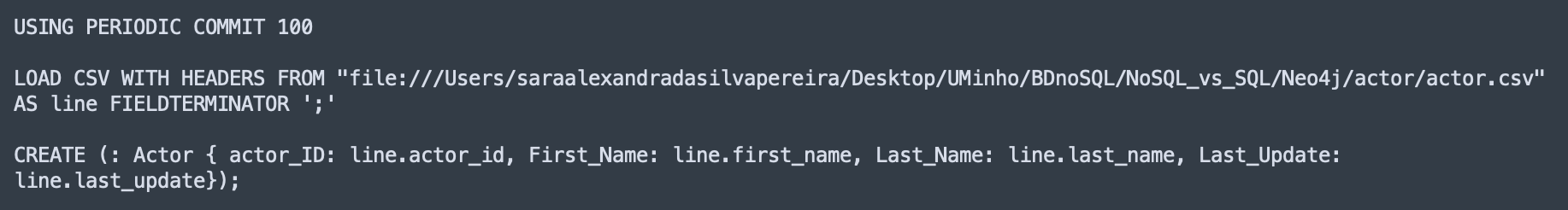
* 1. Processo de migração

Como sabemos, um grafo é composto por dois elementos: um nó e uma relação. Cada nó representa uma entidade e as relações caraterizam a associação entre diferentes nós. Sendo assim, o processo de migração da base de dados fornecida (*Sakilla*) foi feito em cinco passos:

* + Exportação das tabelas de entidades, de *mySQL,* para ficheiros *CSV*
  + Criação de nós com os dados que pertenciam às tabelas da BD relacional através da importação dos ficheiros *CSV*
  + Criação de *constraints* de maneira a garantir que os identificadores de cada nó sejam únicos
  + Criação de índices
  + Criação de relacionamentos

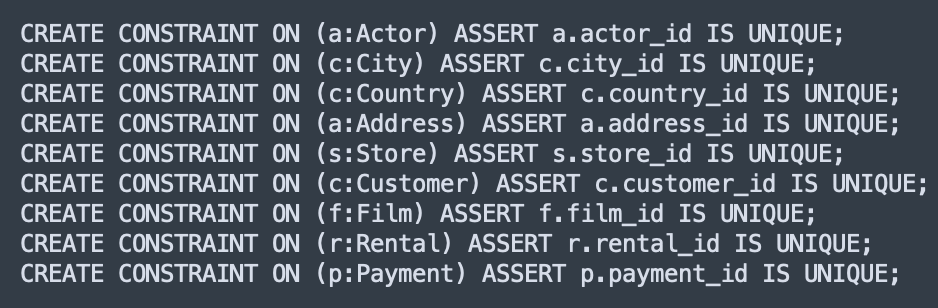
Para a realização desta migração, foi criado um script em Cypher com a seguinte estrutura:

* **Criação de nós**
  + Actor
  + Address
  + City
  + Country
  + Category
  + Customer
  + Film
  + Inventory
  + Language
  + Payment
  + Rental
  + Staff
  + Store



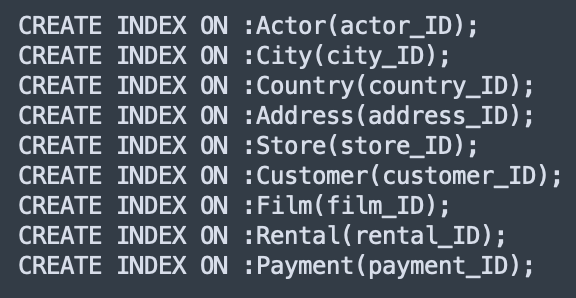
A imagem retrata um exemplo de uma query que carateriza a criação de um nó, neste caso, de um ator e respetivos atributos.

* **Restrições** **(Constraints)**
  + Como o Neo4j permite a criação de dois nodos iguais, foi necessário a criação de uma restrição que torna cada nodo único, não permitindo informação duplicada. No entanto, criar estas restrições torna muito pesado a nível computacional, por isso só não foi criado para as entidades que originariam quantidades pouco significativas de nós como as Category, Staff e Language.

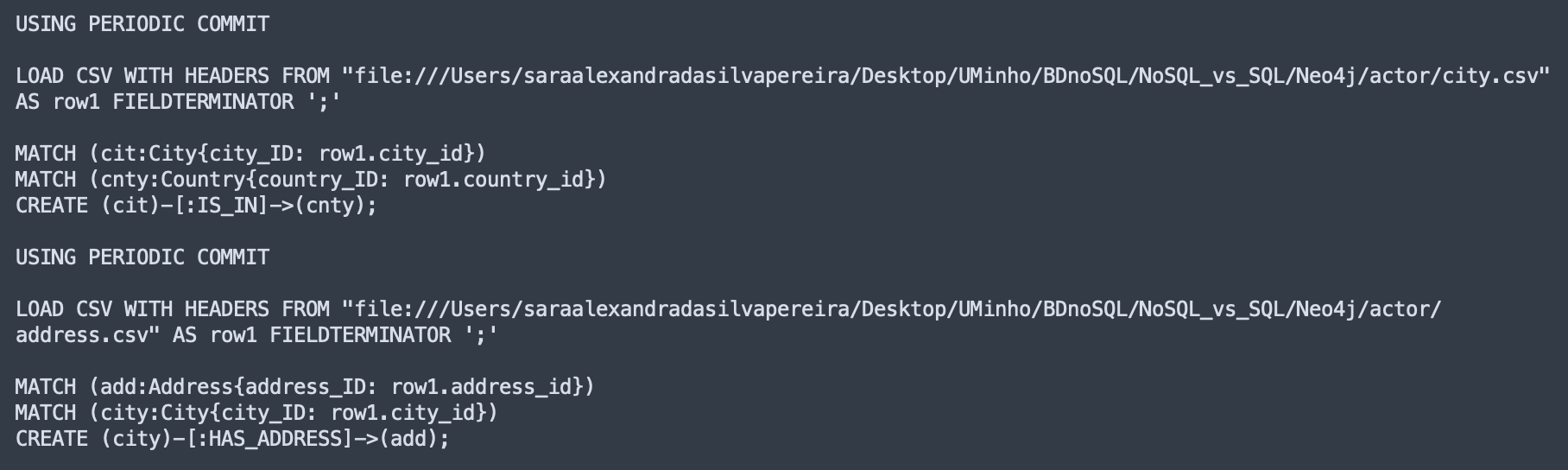


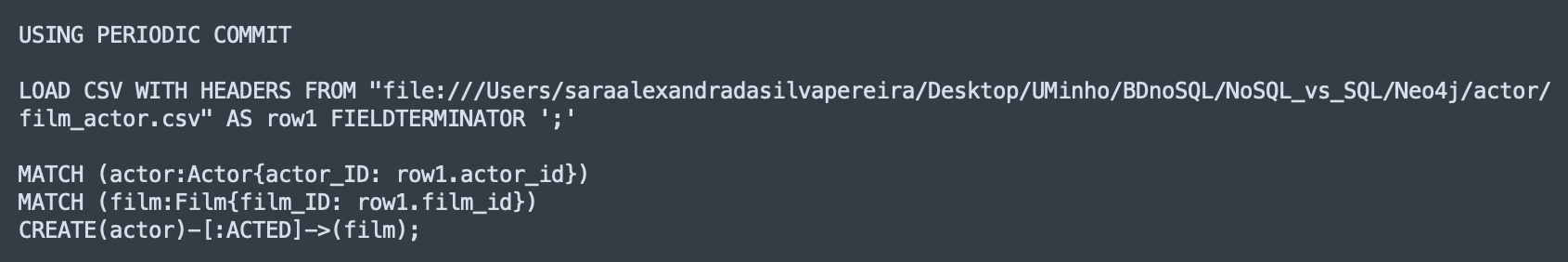
* **Índices**

Para tornar a pesquisa mais rápida foram criados vários índices correspondentes aos nós. Apesar de ser necessário mais memória, a utilização destes índices torna a pesquisa mais eficiente e mais rápida, no entanto é preciso definir muito bem quais os índices a ser criados. Por isso foi decidido só usá-los em determinados nodos, tal como na criação das constraints. Os nodos usados para a criação dos índices foram os mesmo do tópico anterior.



* **Relações**
  + Nas Bases de Dados relacionais, as referências entre tabelas são feitas a partir da repetição de chaves primárias em chaves estrangeiras noutras tabelas. Isto faz com que informação seja repetida tornando-a redundante. Estas referências em Neo4j serão as nossas relações, que consistem em ligações entre nodos.

****

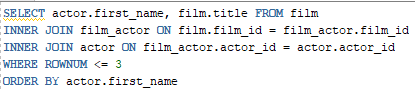
Tomando como exemplo a imagem anterior, em *SQL* sabíamos que existia um relacionamento entre as entidades City e Country devido ao facto de existir uma chave estrangeira – o identificador único de Country - na tabela da City. Assim, foi decidida a criação de uma relação entre esses nós retratando que uma cidade encontra-senum certo país. Foi utilizada a mesma ideologia na criação das restantes relações, incluindo tabelas derivadas de relacionamentos N:M. Estas tabelas seriam compostas por duas chaves primárias das que levaram à sua origem e a relação foi criada a partir da ligação entre essas duas chaves primárias. Um exemplo destas relações encontra-se no ficheiro *Film\_Category.csv,* composto apenas por duas colunas com informação relevante relativas aos identificadores dos filmes e dos identificadores das categorias dos mesmos. Sendo assim, a criação da relação entre eles é dada ****pela imagem que se segue.

* 1. Teste ao novo Sistema de BD

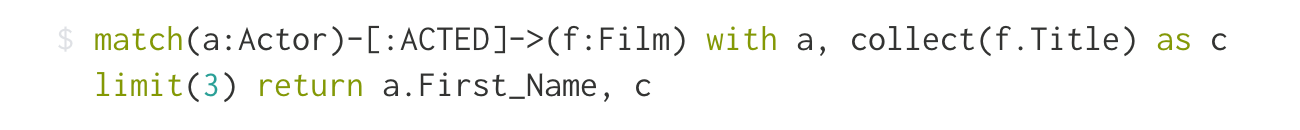
Após corrermos o nosso script *“script1.cypher”,* obtemos a nossa base de dados. Como podemos observar, cada nodo representa uma entidade em SQL possuindo relações entre si. Pela interface do programa, podemos selecionar certos nodos

Para demonstrar as diferenças a nível de queries entre Neo4j e SQL, foi decidido fazer uma demonstração com base no mesmo objetivo. Neste caso, iremos mostrar os três atores que participaram em mais filmes.

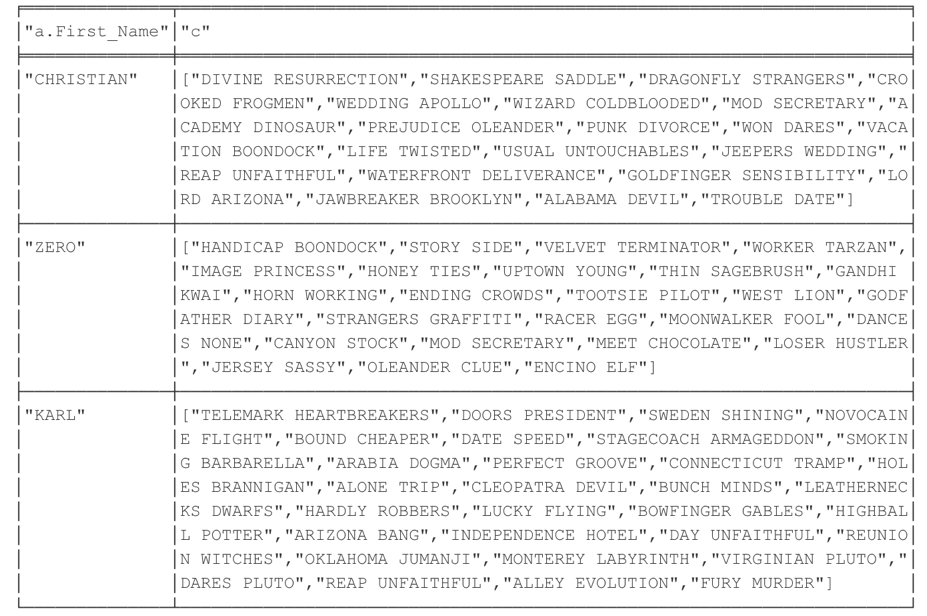
* SQL



* Neo4j

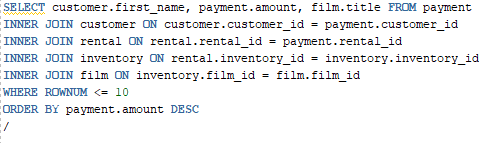


* Resultado Neo4j

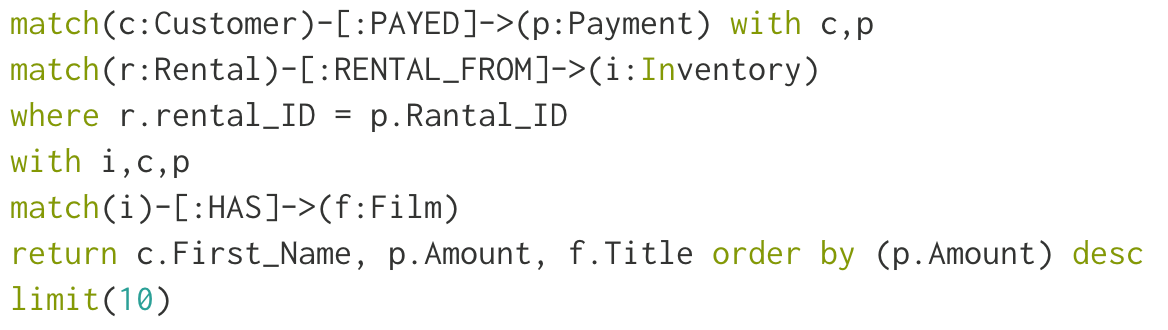


Noutro exemplo, podemos observar os Customers que compraram o Film mais caro.

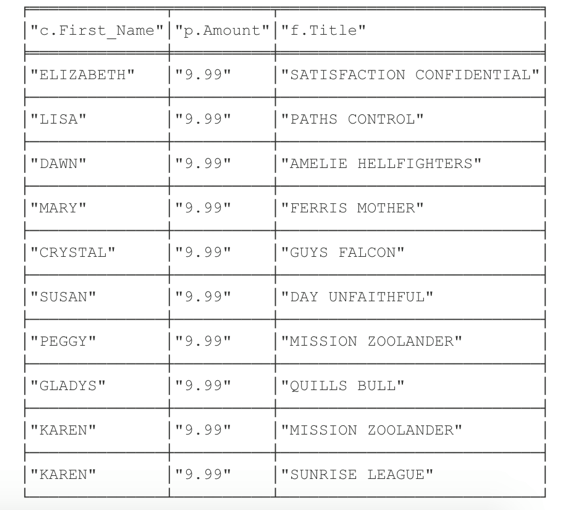
* SQL



* Neo4j



* Resultado Neo4j



1. Base de Dados não Relacionais – MongoDB

1. Conclusão
2. Referências

*... ..*