

**Universidade do Minho**

Mestrado Integrado em Engenharia Informática

**Unidade Curricular de**

**Bases de Dados NOSQL**

Ano Letivo de 2019/2020

**Migração *Sakila***

**Ana Ribeiro - A82474**

**Jéssica Lemos - A82061**

**Pedro Freitas - A80975**

Janeiro, 2020

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

**Migração *Sakila***

**Ana Ribeiro - A82474**

**Jéssica Lemos - A82061**

**Pedro Freitas - A80975**

Janeiro, 2020

**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 Resumo i](#_Toc29573421)

[2 Índice ii](#_Toc29573422)

[3 Índice de Figuras iii](#_Toc29573423)

[4 Índice de Tabelas iv](#_Toc29573424)

[5 Base de Dados Orientada a Grafos - Neo4j 2](#_Toc29573425)

[5.1 Modelo 2](#_Toc29573426)

[5.2 Migração 3](#_Toc29573427)

[5.3 Queries 4](#_Toc29573428)

[5.4 Resultados 6](#_Toc29573429)

[5.5 Comparação com o Modelo Relacional 7](#_Toc29573430)

[6 Referências Bibliográficas 8](#_Toc29573431)

[9 Anexos 12](#_Toc29573432)

**Anexos**

[I. Anexo 1 – Exportação de Dados Neo4j 13](#_Toc29573433)

[II. Anexo 2 – Importação de Dados Neo4j 16](#_Toc29573434)

**Índice de Figuras**

[Figura 1 - Esquema Modelo Neo4j 3](#_Toc29572852)

[Figura 2 - Resultado da *query* Top 5 dos filmes mais alugados 6](#_Toc29572853)

[Figura 3 - Resultado da *query* Top 5 das categorias mais alugadas 7](#_Toc29572854)

**Índice de Tabelas**

**Não foi encontrada nenhuma entrada do índice de ilustrações.**

# Introdução

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

## Contextualização

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

## Apresentação do Caso de Estudo

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

## Motivação e Objectivos

<<Esta secção acolherá os diversos motivos, acompanhados por uma breve descrição, que conduziram à proposta e ao desenvolvimento do trabalho, assim como a apresentação detalhada dos diversos objectivos a alcançar com a sua realização.>>

## Estrutura do Relatório

<<Após a leitura da introdução de um relatório é "simpático" apresentar uma breve descrição daquilo que se vai encontrar nos demais capítulos do relatório.>>

# Base de Dados Orientada a Grafos - Neo4j

## Modelo

De modo a obtermos o modelo apropriado para o *Neo4j* da Base de Dados *Sakila* tornou-se necessário realizarmos algumas alterações no modelo da mesma, entre as quais:

* **Retirar a Identificação –** No Modelo Relacional todas as entidades possuem uma chave estrangeira, que se caracterizam por um ID. Uma vez que no *Neo4j*, por ser uma base de dados orientada a grafos, não são necessários estes mesmos identificadores para criarmos as entidades e os correspondentes relacionamentos decidimos remover os mesmos. Contudo, constatamos que a *Store* não possuía mais nenhum elemento que a caracteriza-se pelo que esta é a única entidade com respetivo identificador.
* **Retirar tabelas desnecessárias –** Ao longo deste processo tornou-se percetível que existiam algumas tabelas que não eram necessárias serem convertidas, sendo estas as correspondentes aos relacionamentos muitos para muitos, ou seja, as tabelas*film\_category*, *film\_actor* e *inventory*. No entanto, nesta última existia um indentificador como chave estrangeira, que permitia obtermos o *stock* de filmes existente em cada *store*. Para armazenar este valor optamos por criar uma propriedade no relacionamento entre a *Store* e o *Film* com o *stock* correspondente àquela loja. Por último, optamos por não converter a tabela *film\_text*, tendo em conta que consideramos que esta não acrescentava informação útil nem necessária às seguintes fases do projeto.

Terminada a fase de tomada de decisão, podemos então esboçar o esquema que pretendemos para o nosso modelo, que é apresentado na Figura 1.

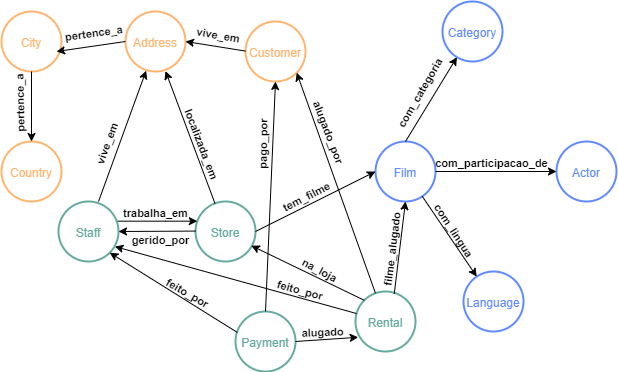


Figura 1 - Esquema Modelo Neo4j

## Migração

Nesta fase passamos para a exportação dos dados da *Sakila* para ficheiros *CSV* de modo a importá-los posteriormente para o *Neo4j*. Tendo em conta as decisões indicadas anteriormente criamos um ficheiro *sql*, que se encontra em anexo, para a extração destes mesmos dados. Basicamente o processo que adotamos para a migração foi o seguinte:

* **Exportação –** nesta fase recorrendo ao *sql* extraímos todos os dados pretendidos tendo em conta as decisões tomadas. Tal como foi indicado anteriormente, decidimos remover os identificadores uma vez que não eram necessários, contudo devido à existência de algumas entidades, como é o caso do *Payment* e da *Rental*, que não tinham nenhum atributo que as pudessem identificar houve a necessidade de migrar estas com os seus IDs para a posterior criação dos relacionamentos. Para além disso, uma vez que constatamos que existia um ator repetido optamos também por migrar o seu ID. A título exemplificativo apresentamos de seguida um excerto referente à exportação do *Country* e da *City.*

**select \* from country**

**into outfile '/var/lib/mysql-files/country.csv'**

**FIELDS TERMINATED BY ','**

**LINES TERMINATED BY '\n';**

**select city, country from city**

**inner join country on city.country\_id = country.country\_id**

**into outfile '/var/lib/mysql-files/city.csv'**

**FIELDS TERMINATED BY ','**

**LINES TERMINATED BY '\n';**

* **Importação –** realizada a exportação podemos agora importar os dados para *Neo4j*. Nesta fase, tivemos o cuidado de eliminar os identificadores das entidades *Payment*, *Rental* e *Actor,* que foram migrados tal com explicado anteriormente, após a criação dos relacionamentos. De seguida, é exemplificado a importação para os mesmos casos, em que para o *Country* são criados os nós e de seguida um índice no nome do mesmo. Enquanto que para a *City* para além da criação dos nós é criado o relacionamento entre o anterior e esta, sendo por fim criado o índice no nome da mesma.

**load csv from "file:///country.csv" as line**

**create (co :Country {country : line[1]});**

**create index on :Country(country);**

**load csv from "file:///city.csv" as line**

**match (co: Country)**

**where co.country = line[1]**

**create (ci: City {city: line[0]}),**

**(ci)-[r:pertence\_a]->(co);**

**create index on :City(city);**

## Queries

1. **Top 5 dos filmes mais alugados**

**match(f:Film)<-[a:filme\_alugado]-(r:Rental)**

**return f, count(a) as num\_films**

**order by num\_films desc**

**limit 5**

1. **Top 3 das lojas que mais faturam**

**match(s:Store)<-[:trabalha\_em]-(st:Staff)<-[:feito\_por]-(p:Payment)**

**return s, sum(p.amount) as num\_stores**

**order by num\_stores desc**

**limit 3**

1. **Top 3 dos alugueres de filmes mais longos**

**match(r:Rental)-[:filme\_alugado]->(f:Film)**

**where exists(r.return\_date)**

**return f.title, avg(duration.between(datetime(r.rental\_date), datetime(r.return\_date))) as dur\_avg**

**order by dur\_avg desc**

**limit 3**

1. **Top 5 das categorias mais alugadas**

**match(r:Rental)-[:filme\_alugado]->(f:Film)-[:com\_categoria]->(c:Category)**

**return count(f) as num\_films,c**

**order by num\_films desc**

**limit 5**

1. **Top 3 dos atores que participaram em mais filmes**

**match(f:Film)-[:com\_participacao\_de]->(a:Actor)**

**return count(f) as num\_films,a**

**order by num\_films desc**

**limit 3**

1. **Top 5 dos distritos com mais clientes**

**MATCH (c:Customer)-[r:vive\_em]->(a:Address)**

**return count(c) as num\_customer, a.district**

**order by num\_customer desc**

**limit 5**

1. **Top 5 dos funcionários que alugaram mais filmes**

**match(r:Rental)<-[:alugado]-(p:Payment)-[:feito\_por]->(s:Staff)**

**return count(r) as num\_films, s**

**order by num\_films desc**

**limit 5**

## Resultados

Através das *queries* definidas pelo grupo tornou-se assim possível obter várias informações relevantes da Base de Dados *Sakila*. De seguida são apresentados os resultados obtidos em algumas das apresentadas anteriormente.

1. **Top 5 dos filmes mais alugados**

Através desta *query* é possível obter os 5 filmes mais alugados, sendo possível observar neste caso em específico o nome dos filmes bem como o número de vezes que foi alugado. Nesta optamos por devolver os filmes, ou seja, todas as informações referentes aos mesmos, contudo no resultado seguinte apenas apresentamos o título, dado que se tornaria muita informação para ser apresentada.

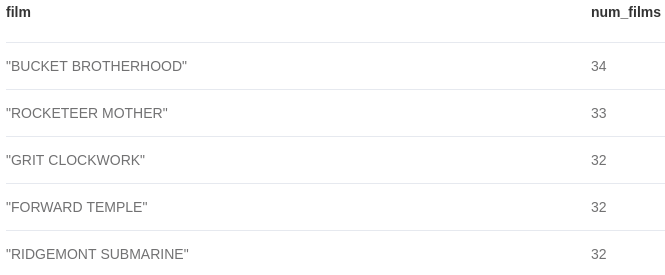


Figura 2 - Resultado da *query* Top 5 dos filmes mais alugados

1. **Top 5 das categorias mais alugadas**

Nesta podemos constatar as 5 categorias mais alugadas, ou seja, as categorias dos filmes que foram mais vezes alugados.

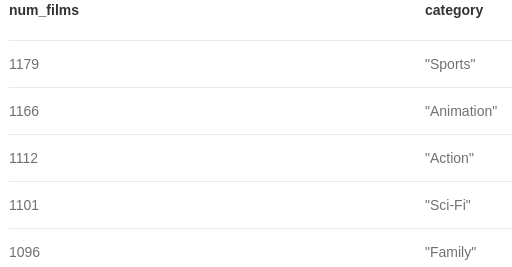


Figura 3 - Resultado da *query* Top 5 das categorias mais alugadas

1. **Top 3 dos atores que participaram em mais filmes**

Podemos ainda obter os atores que participaram mais vezes em filmes. Note-se que neste exemplo apenas apresentamos o primeiro nome e último nome dos atores, uma vez que estes possuem várias informações, contudo optamos por devolver os atores na *query* desenvolvida.



Figura 4 – Resultado da *query* Top 3 dos atores que participaram em mais filmes

## Comparação com o Modelo Relacional

Como era expectável a migração da Base de Dados Relacional *Sakila* para *Neo4j* permitiu a implementação de um modelo bastante mais simples, em parte porque os relacionamentos de muitos para muitos podem ser tratados uma forma eficiente, e ainda foi possível eliminar as chaves estrangeiras. Para além disso, a implementação das *queries* é bastante mais percetível e rápida sendo que a performance é melhorada uma vez que em *MySQL* as mesmas teriam uma resolução mais complexa com vários *JOINs.*

# Conclusões e Trabalho Futuro

Referências

1. Ian Robinson, Jim Webber and Emil Eifrem 2015. Graph Databases New Opportunities for Connected Data. Second Edition / O’Reilly Media.

Lista de Siglas e Acrónimos

**BD** Base de Dados

**SGBD** Sistema de Gestão de Base de Dados

**SQL** Structured Query Language

**SBD** Sistema de Base de Dados

**SBDR** Sistema de Base de Dados Relacional

**NoSQL** Not only SQL

# Anexos

**1. Exportação dos Dados Neo4j**

**2. Importação dos Dados Neo4j**

1. Anexo 1 – Exportação de Dados Neo4j

select \* from country

into outfile '/var/lib/mysql-files/country.csv'

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

select city, country from city

inner join country on city.country\_id = country.country\_id

into outfile '/var/lib/mysql-files/city.csv'

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

select address, address2, district, city, postal\_code, phone, country from address

inner join city on address.city\_id = city.city\_id

inner join country on city.country\_id = country.country\_id

into outfile '/var/lib/mysql-files/address.csv'

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

select first\_name, last\_name, address, city, email, active, username, password, store\_id from staff

inner join address on address.address\_id = staff.address\_id

inner join city on city.city\_id = address.city\_id

into outfile '/var/lib/mysql-files/staff.csv'

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

select address, city, username, store.store\_id from store

inner join address on address.address\_id = store.address\_id

inner join city on city.city\_id = address.city\_id

inner join staff on staff.staff\_id = store.manager\_staff\_id

into outfile '/var/lib/mysql-files/store.csv'

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

select name from category

into outfile '/var/lib/mysql-files/category.csv'

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

select actor\_id, first\_name, last\_name from actor

into outfile '/var/lib/mysql-files/actor.csv'

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

select first\_name, last\_name, email, active, create\_date, store\_id, address, city from customer

inner join address on address.address\_id = customer.address\_id

inner join city on address.city\_id = city.city\_id

into outfile '/var/lib/mysql-files/customer.csv'

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

select rental\_id, customer.first\_name, customer.last\_name, amount, payment\_date, staff.username, payment\_id from payment

inner join customer on customer.customer\_id = payment.customer\_id

inner join staff on staff.staff\_id = payment.staff\_id

into outfile '/var/lib/mysql-files/payment.csv'

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

select name from language

into outfile '/var/lib/mysql-files/language.csv'

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

select title, description, release\_year, rental\_duration, rental\_rate, film.length, replacement\_cost, rating, special\_features, language.name from film

inner join language on language.language\_id = film.language\_id

into outfile '/var/lib/mysql-files/film.csv'

FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '\n';

select title, actor.actor\_id from film

inner join film\_actor on film.film\_id = film\_actor.film\_id

inner join actor on film\_actor.actor\_id = actor.actor\_id

into outfile '/var/lib/mysql-files/film\_actor.csv'

FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '\n';

select film.title, store\_id, film\_text.description, count(\*) from inventory

inner join film on inventory.film\_id = film.film\_id

inner join film\_text on film\_text.film\_id = inventory.film\_id

group by film.film\_id, inventory.store\_id

into outfile '/var/lib/mysql-files/inventory.csv'

FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '\n';

select title, category.name from film

inner join film\_category on film.film\_id = film\_category.film\_id

inner join category on category.category\_id = film\_category.category\_id

into outfile '/var/lib/mysql-files/film\_category.csv'

FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '\n';

select rental\_id, rental\_date, ifnull(return\_date,""), customer.first\_name, customer.last\_name, staff.username, film.title, inventory.store\_id from rental

inner join inventory on inventory.inventory\_id = rental.inventory\_id

inner join customer on customer.customer\_id = rental.customer\_id

inner join staff on staff.staff\_id = rental.staff\_id

inner join film on inventory.film\_id = film.film\_id

into outfile '/var/lib/mysql-files/rental.csv'

FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '\n';

select payment.payment\_id, rental.rental\_id from payment

inner join rental on payment.rental\_id = rental.rental\_id

into outfile '/var/lib/mysql-files/rental-payment.csv'

FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '\n';

1. Anexo 2 – Importação de Dados Neo4j

load csv from "file:///country.csv" as line

create (co :Country {country : line[1]});

create index on :Country(country);

load csv from "file:///city.csv" as line

match (co: Country)

where co.country = line[1]

create (ci: City {city: line[0]}),

(ci)-[r:pertence\_a]->(co);

create index on :City(city);

using periodic commit 100

load csv from "file:///address.csv" as line

match (ci: City {city: line[3]})-[:pertence\_a]->(co: Country {country: line[6]})

create (adr: Address { address: line[0], address2: line[1], district: line[2], code: line[4], phone: line[5]}),

(adr)-[r:pertence\_a]->(ci);

create index on :Address(address);

load csv from "file:///staff.csv" as line

match (adr: Address {address: line[2]})-[:pertence\_a]->(ci: City {city : line[3]})

create (st: Staff { first\_name: line[0], last\_name: line[1], email: line[4], active: line[5], username: line[6], password: line[7]}),

(st)-[r:vive\_em]->(adr);

load csv from "file:///store.csv" as line

match (adr: Address {address: line[0]})-[:pertence\_a]->(ci: City {city : line[1]})

match (m: Staff {username: line[2]})

create (adr)<-[:localizada\_em]-(sto: Store {store\_id: line[3]})-[:gerido\_por]->(m);

create index on :Store(store\_id);

load csv from "file:///staff.csv" as line

match (staff: Staff {username: line[6]})

match (store: Store {store\_id: line[8]})

create (staff)-[:trabalha\_em]->(store);

create index on :Staff(username);

load csv from "file:///category.csv" as line

create (:Category{name:line[0]});

create index on :Category(name);

load csv from "file:///actor.csv" as line

create (:Actor{actor\_id: line[0], first\_name: line[1], last\_name: line[2]});

create index on :Actor(actor\_id);

create index on :Actor(first\_name, last\_name);

load csv from "file:///customer.csv" as line

match (adr: Address {address: line[6]})-[:pertence\_a]->(c: City {city: line[7]})

match (st: Store {store\_id: line[5]})

create (adr)<-[:vive\_em]-(:Customer {first\_name: line[0], last\_name: line[1], email: line[2], active: line[3], create\_date: replace(line[4],' ','T')})-[:membro\_em]->(st);

load csv from "file:///payment.csv" as line

match (c: Customer {first\_name: line[1], last\_name: line[2]})

match (s: Staff {username: line[5]})

create (c)<-[:pago\_por]-(p:Payment {amount: toFloat(line[3]), payment\_date: replace(line[4],' ','T'), payment\_id: line[6]})-[:feito\_por]->(s);

create index on :Payment(payment\_id);

load csv from "file:///language.csv" as line

create (:Language {name: line[0]});

create index on :Language(name);

load csv from "file:///film.csv" as line

match (l:Language {name: line[9]})

create (:Film {title: line[0], description: line[1], release\_year: line[2], rental\_duration: line[3], rental\_rate: line[4], length: line[5], replacement\_cost: line[6], rating: line[7], special\_features: line[8]})-[:com\_lingua]->(l);

create index on :Film(title);

load csv from "file:///film\_actor.csv" as line

match (f: Film {title: line[0]})

match (a: Actor {actor\_id: line[1]})

create (f)-[:com\_participacao\_de]->(a);

match (a:Actor)

remove a.actor\_id;

load csv from "file:///inventory.csv" as line

match (f: Film {title: line[0]})

match (s: Store {store\_id: line[1]})

create (s)-[:tem\_filme {numero: line[3]}]->(f);

load csv from "file:///film\_category.csv" as line

match (f: Film {title: line[0]})

match (c: Category {name: line[1]})

create (f)-[:com\_categoria]->(c);

load csv from "file:///rental.csv" as line

match (c: Customer {first\_name: line[3], last\_name: line[4]})

match (sta: Staff {username: line[5]})

match (f: Film {title: line[6]})

match (sto: Store {store\_id: line[7]})

create (sta)<-[:feito\_por]-(r:Rental {rental\_id: line[0], rental\_date: replace(line[1],' ','T'), return\_date: replace(line[2], " ", "T")})-[:filme\_alugado]->(f),

(sto)<-[:na\_loja]-(r)-[:alugado\_por]->(c);

create index on :Rental(rental\_id);

load csv from "file:///rental-payment.csv" as line

match (p: Payment {payment\_id: line[0]})

match (r: Rental {rental\_id: line[1]})

create (p)-[:alugado]->(r);

match (p:Payment)

remove p.payment\_id;

match (r:Rental)

remove r.rental\_id;