



**Universidade do Minho**  
Mestrado Integrado em Engenharia Informática

## **Unidade Curricular de Bases de Dados**

Ano Letivo de 2017/2018

## **MuDBa – Base de Dados Musical**

**Francisco José Moreira Oliveira, a78416**

**João Pedro de Moura Pereira e Ferreira Carvalho, a79073**

**Raul Vilas Boas, a79617**

**Vitor Emanuel Carvalho Peixoto, a79175**

Janeiro, 2018

**BD**

Data de Receção	
Responsável	
Avaliação	
Observações	

## MuDBa – Base de Dados Musical

**Francisco José Moreira Oliveira, a78416**

**João Pedro de Moura Pereira e Ferreira Carvalho, a79073**

**Raul Vilas Boas, a79617**

**Vitor Emanuel Carvalho Peixoto, a79175**

Janeiro, 2018

## Resumo

Para este trabalho foi-nos pedida a implementação de uma base de dados não relacional orientada por grafos a partir da base de dados relacional desenvolvida na primeira parte deste trabalho.

Nesta fase será apenas abordada uma ou outra questão inicial acerca do projeto, sendo que iremos concentrar grande parte do relatório na transformação da base de dados relacional para não relacional e na validação do modelo de dados.

No fim elaboramos uma breve conclusão acerca do trabalho realizado e daquilo que poderá ser o futuro desta base de dados.

**Área de Aplicação:** Catálogo de músicas online (“IMDb” da Música).

**Palavras-Chave:** Sistema de base de dados, Base de dados, Base de dados não relacional, NoSQL, Música, Artista, Álbum, Gravação, Género, Banda, Grafo, NÓ, Relacionamento, Requisito, Modelação Conceitual, Interrogação, Neo4j, Cypher.

# Índice

1.	Definição do sistema	1
1.1.	Contexto de aplicação do sistema	1
1.2.	Fundamentação da implementação da base de dados	1
2.	Modelação da base de dados	2
2.1.	Modelo conceitual	2
2.2.	Modelo lógico	3
2.3.	Modelação não relacional	3
3.	Implementação da base de dados	4
3.1.	Seleção do sistema	4
3.2.	Construção do sistema	4
3.2.1.	Nós	4
3.2.2.	Relacionamentos	9
3.2.3.	Grafo final do <i>Neo4j</i>	11
3.3.	Seleção e definição de índices e restrições	11
3.4.	Tradução das interrogações do utilizador para <i>Cypher</i>	13
4.	Conclusões e trabalho futuro	16

# Índice de Figuras

<b>Figura 1</b> – Diagrama ER.	2
<b>Figura 2</b> – Modelo Lógico.	3
<b>Figura 3</b> – Nós dos artistas.	5
<b>Figura 4</b> – Nós das gravações.	6
<b>Figura 5</b> – Nós das faixas.	7
<b>Figura 6</b> – Nós das localidades.	7
<b>Figura 7</b> – Nós dos géneros musicais.	8
<b>Figura 8</b> – Nós dos membros.	9
<b>Figura 9</b> – Esquema do <i>Neo4j</i> .	11
<b>Figura 10</b> – Resultado da <i>Query</i> extra em <i>Cypher</i> .	13
<b>Figura 11</b> – Resultado da <i>Query</i> extra em <i>SQL</i> .	13
<b>Figura 12</b> – Resultado da 1 <sup>a</sup> <i>Query</i> em <i>Cypher</i> .	13
<b>Figura 13</b> – Resultado da 1 <sup>a</sup> <i>Query</i> em <i>SQL</i> .	14
<b>Figura 14</b> – Pequeno excerto do resultado da 2 <sup>a</sup> <i>Query</i> em <i>Cypher</i> .	14
<b>Figura 15</b> – Pequeno excerto do resultado da 2 <sup>a</sup> <i>Query</i> em <i>SQL</i> .	14
<b>Figura 16</b> – Pequeno excerto do resultado da 3 <sup>a</sup> <i>Query</i> em <i>Cypher</i> .	15
<b>Figura 17</b> – Pequeno excerto do resultado da 3 <sup>a</sup> <i>Query</i> em <i>SQL</i> .	15

# 1. Definição do Sistema

## 1.1. Contexto de aplicação do sistema

Num mundo onde a digitalização da informação é um tópico que tem apresentado um crescimento exponencial nos últimos anos, a necessidade de ter toda a informação em todo o lado é cada vez mais imperativa.

Na sociedade de hoje, a Internet é utilizada maioritariamente por jovens, que são um público bastante exigente em termos de informação disponível na Internet.

Este público dedica bastantes horas do seu dia a ouvir música em todo o tipo diferente de plataformas, em qualquer sítio, a qualquer hora. Assim torna-se evidente que há uma necessidade deste público ter à sua disposição toda a informação que achem relevante acerca de uma música, artista ou álbum.

Visto que a quantidade de dados do mundo musical está a crescer a um ritmo exponencial torna-se crucial o sistema que armazene as informações estar adaptado à enorme quantidade de dados. Uma base de dados relacional acaba por se tornar lenta com grandes volumes de dados, logo torna-se imperativo a sua migração para uma base de dados não-relacional, desenhadas para alojar grandes quantidades de dados e responderem às interrogações a uma velocidade decente.

## 1.2. Fundamentação da implementação da base de dados

A plataforma terá disponível diversas informações relativas ao mundo da música, incluindo os álbuns, as faixas, os artistas, incluindo bandas e os seus integrantes e diversas outras informações.

A fundamentação da implementação desta base de dados já foi abordada na primeira fase deste projeto, no entanto, torna-se importante discutir quais as vantagens de implementar novamente a base de dados, desta vez num modelo não relacional.

Para começar, uma base de dados orientada por grafos, permite desde logo uma muito maior *performance* ao lidar com ligações entre dados. Nas bases de dados relacionais, as junções irão piorar a *performance* das *queries* conforme a quantidade de dados cresce, no entanto, em grafos, a *performance* permanece relativamente constante ao longo do crescimento da base de dados [1].

Para além disso, os grafos não restringem a um determinado modelo, podendo adicionar novas entidades e relacionamentos se acharmos benéfico para o desempenho deste. Esta flexibilidade também permite acelerar o processo de modelação, visto que não temos de planear exaustivamente o modelo, tendo em vista o futuro da base de dados [1].

Em último, as bases de dados não relacionais são as ideais para armazenar grandes quantidades de dados, que será o caso deste projeto numa fase futura.

## 2. Modelação da base de dados

### 2.1. Modelo conceitual

Este foi o diagrama ER obtido através dos requisitos levantados na fase primária deste projeto.

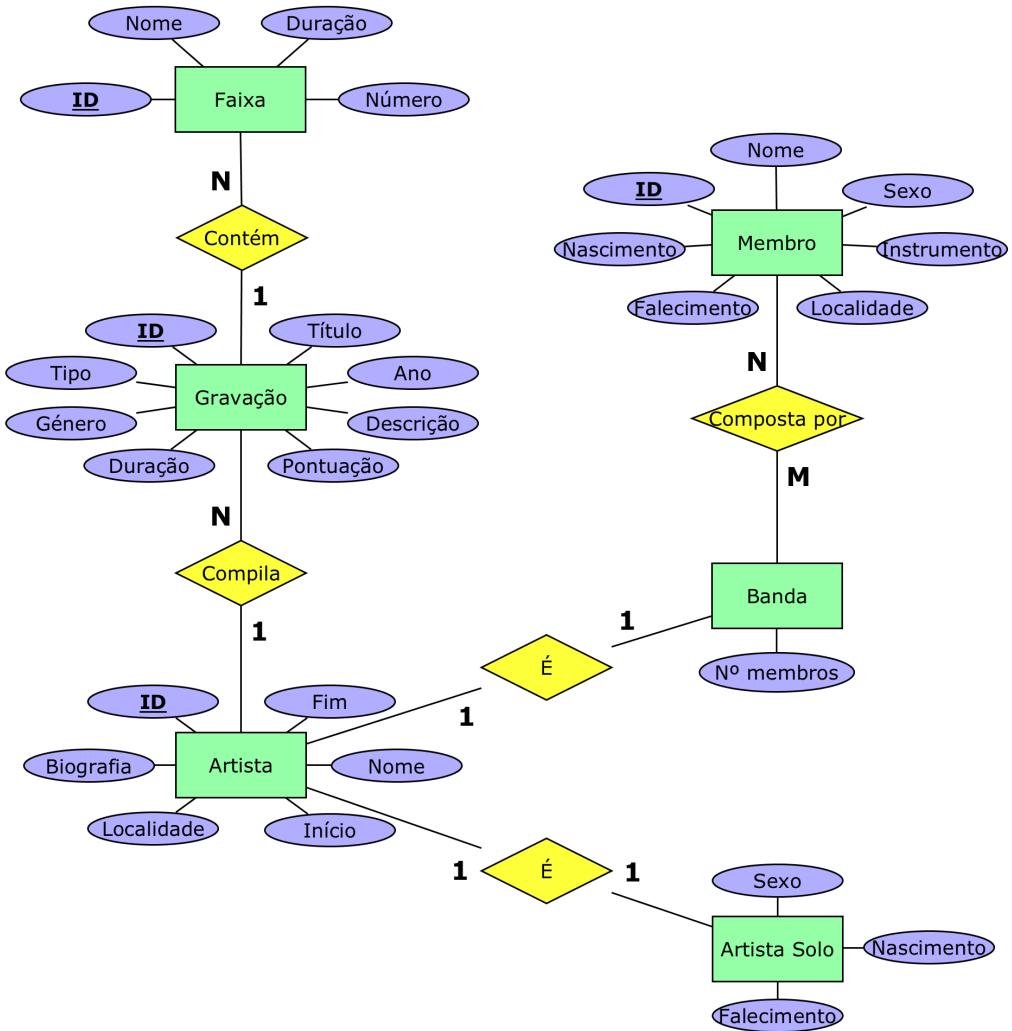


Figura 1 – Diagrama ER.

## 2.2. Modelo lógico

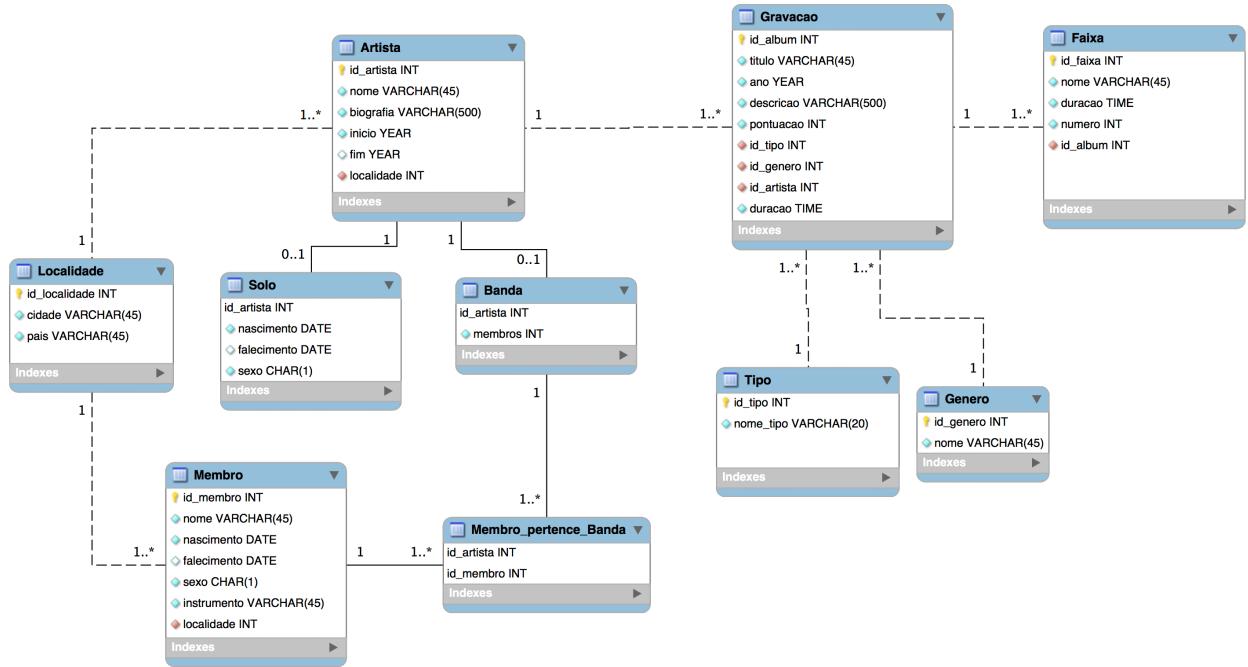


Figura 2 – Modelo Lógico.

## 2.3. Modelação não relacional

Numa base de dados não relacional, a modelação acaba por não ter um papel tão importante como numa relacional, visto que o modelo não relacional é por natureza flexível, permitindo facilmente a modificação do modelo, introduzindo novas entidades (etiquetas) e relações sem grande custo.

Assim, a modelação conceitual acaba por ser um breve esboço do que a base de dados vai armazenar, visto não ser necessário efetuar uma análise exaustiva dos requisitos e do planeamento futuro da base de dados.

Acabamos por não implementar alguns atributos definidos nestes modelos (aqueles com textos extensivos – biografia, descrição, etc.), mas também adicionar outros, que achamos interessantes (ano de entrada de um membro na banda).

Dada a flexibilidade da base de dados não relacional acabamos também por adicionar novas funcionalidades que seriam de difícil implementação no modelo relacional, como a participação de outros artistas em faixas (conhecidos como *featuring*).

## 3. Implementação da base de dados

### 3.1. Seleção do sistema

Uma base de dados orientada por grafos é por ventura uma das melhores opções no que a bases de dados não relacionais diz respeito, visto que é altamente escalável e rápida na armazenagem e leitura de grandes quantidades de dados. Dentro do seu ramo, o sistema que mais se destaca é o *Neo4j* dadas todas as suas funcionalidades e a facilidade de uso.

### 3.2. Construção do sistema

O *MySQL Workbench* permite criar rapidamente ficheiros .csv com todos os tuplos de todas as tabelas existentes na base de dados relacional. O *Neo4j* por sua vez disponibiliza comandos que permitem facilmente carregar todos os tuplos para o grafo como nós.

```
LOAD CSV WITH HEADERS FROM "file:///artista.csv" AS row
CREATE (a:Artista {id_artista:toInt(row.id_artista)})
SET a.nome = toString(row.nome),
    a.biografia = toString(row.biografia),
    a.inicio = toInt(row.inicio),
    a.fim = toInt(row.fim),
    a.localidade = toInt(row.localidade)
RETURN a;
```

No entanto isto iria levar-nos a eliminar propriedades que neste sistema não relacional não seriam necessárias, como o “*id\_artista*” e a “*localidade*”, chave primária e estrangeira respetivamente. Apesar deste ser um caminho menos trabalhoso inicialmente, iria levar posteriormente a um trabalho desnecessário.

Optamos então por desenhar todos os nós e relações de início, tendo em conta ainda assim a modelação feita na primeira fase deste trabalho. Esta opção tomada pode, no entanto, levar a erros na transição aos estivemos atentos. Por isso, numa base de dados de grande dimensão, o melhor método a ser adotado seria mesmo carregar os ficheiros .csv, apagar as propriedades desnecessárias e criar os relacionamentos entre nós do grafo.

#### 3.2.1. Nós

A criação dos nós assentou basicamente nas entidades estabelecidas nos modelos realizados na primeira fase deste projeto. No fundo toda a estrutura é baseada na que já foi desenvolvida, contando com eventuais alterações que achamos benéficas para o projeto, dadas as capacidades de uma base de dados orientada a grafos.

##### 3.2.1.1. Artistas

Na primeira fase deste trabalho já havíamos definido os atributos para os artistas, separando os artistas que são bandas e artistas solo. Os atributos (propriedades) definidos mantiveram-se todos excetuando a

biografia do artista, que foi removida para simplificar a BD, de modo a não ser necessário escrever texto extensivamente para o povoamento inicial. As propriedades definidas para bandas foram as seguintes:

- Nome
- Início
- Fim (se for o caso)
- Número de membros

As propriedades definidas para os artistas solo foram as seguintes:

- Nome
- Início
- Fim (se for o caso)
- Nascimento
- Falecimento (se for o caso)
- Sexo

Assim, definimos os seguintes artistas:

```
CREATE (BonIver:Banda {nome:'Bon Iver', inicio:2006, membros:4}),
       (Radiohead:Banda {nome:'Radiohead', inicio:1985, membros:5}),
       (RHCP:Banda {nome:'Red Hot Chili Peppers', inicio:1983, membros:4}),
       (Coldplay:Banda {nome:'Coldplay', inicio:1996, membros:4}),
       (TheKillers:Banda {nome:'The Killers', inicio:2001, membros:4}),
       (TheChainsmokers:Banda {nome:'The Chainsmokers', inicio:2012, membros:2})

CREATE (Kanye:Solo {nome:'Kanye West', inicio:1996, nascimento:19770608, sexo:'m'}),
       (DavidBowie:Solo {nome:'David Bowie', inicio:1964, fim:2016, nascimento:19470108,
                        falecimento:20160110, sexo:'m'})
```

Após executarmos o código *Cypher* no *Neo4j browser*, obtivemos os seguintes nós de artistas:



**Figura 3 – Nós dos artistas.**

### 3.2.1.2. Gravações

Os atributos para as gravações mantiveram-se os mesmos, excetuando a descrição da gravação (para facilitar o processo de povoamento), a duração (que pode ser obtida somando as durações das faixas desse álbum, como iremos ver numa query mais à frente) e as chaves estrangeiras (serão dadas pelos relacionamentos com outros nós).

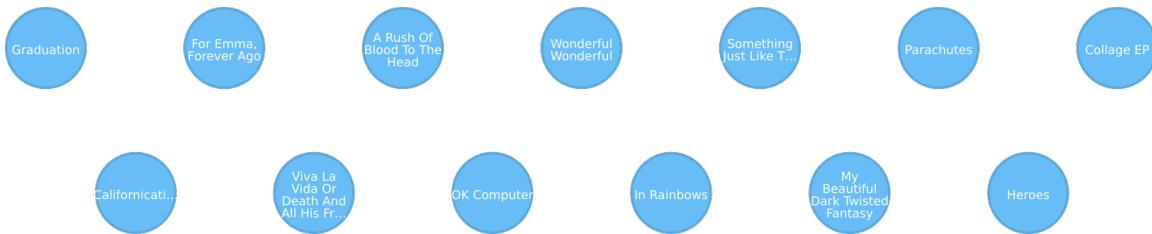
As propriedades definidas para as gravações foram as seguintes:

- Nome
- Ano
- Pontuação (rating de 0-10)

Definimos várias gravações, de vários tipos (álbuns, EP's e singles), sendo que cada artista tem pelo menos uma gravação sua na base de dados:

```
CREATE (OKComputer:Album {nome:'OK Computer', ano:1997, pontuacao:10}),  
       (ParachutesAlbum:Album {nome:'Parachutes', ano:1999, pontuacao:7}),  
       (Collage:EP {nome:'Collage EP', ano:2016, pontuacao:3}),  
       (SJLT:Single {nome:'Something Just Like This', ano:2017, pontuacao:5}),  
       (AROBTH:Album {nome:'A Rush Of Blood To The Head', ano:2002, pontuacao:9}),  
       (HeroesAlbum:Album {nome:'Heroes', ano:1977, pontuacao:8}),  
...  
...
```

Após executarmos o código Cypher no Neo4j browser, obtivemos os seguintes nós:



**Figura 4 – Nós das gravações.**

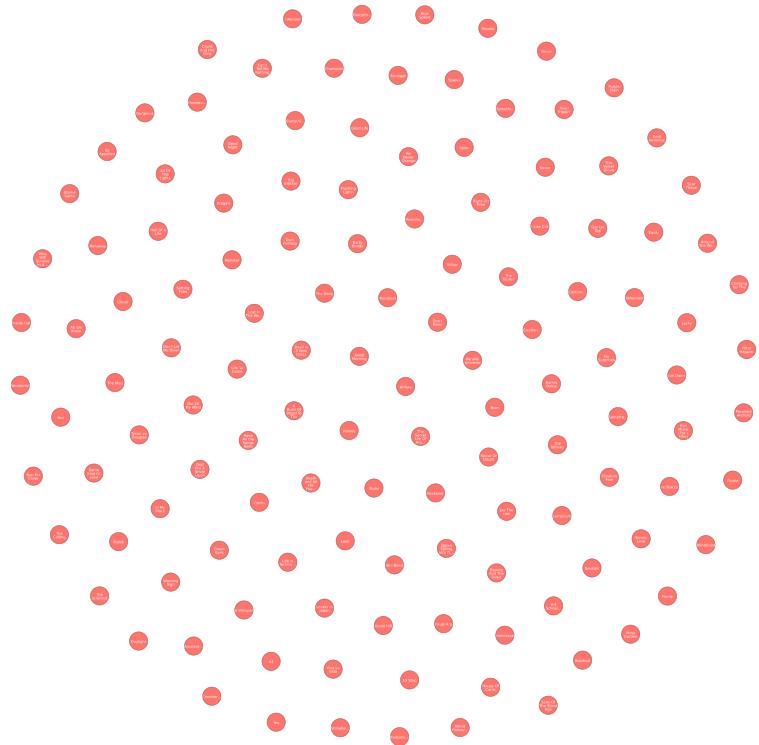
### 3.2.1.3. Faixas

As propriedades para as faixas são os atributos que foram definidos na primeira fase:

- Nome
- Duração
- Número (ordem da faixa na gravação respetiva)

Assim, mostramos a seguir algumas das faixas definidas, pertencentes a gravações já criadas:

```
CREATE (Yellow:Faixa {nome:'Yellow', duracao:4.45, numero:5}),  
       (Heroes:Faixa {nome:'Heroes', duracao:6.12, numero:3}),  
       (VivaLaVida:Faixa {nome:'Viva La Vida', duracao:4.02, numero:7}),  
       (NoSurprises:Faixa {nome:'No Surprises', duracao:3.85, numero:10}),  
...  
...
```



**Figura 5 – Nós das faixas.**

### 3.2.1.4. Localidades

Optamos por manter as localidades como um tipo de dados próprio por questões de integridade dos dados. As propriedades são os mesmos atributos que foram definidos na primeira fase:

- Cidade
- País

Assim, mostramos a seguir algumas das localidades definidas:

```
CREATE (LA:Localidade {cidade:'Los Angeles', pais:'Estados Unidos da América'}),  

(Londres:Localidade {cidade:'Londres', pais:'Reino Unido'}),  

(NovaIorque:Localidade {cidade:'Nova Iorque', pais:'Estados Unidos da América'}),  

...
```



**Figura 6 – Nós das localidades.**

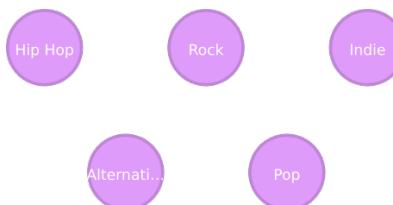
### 3.2.1.5. Géneros musicais

Optamos também por manter os géneros musicais como um tipo de dados próprio por questões de integridade dos dados. As propriedades de um género são os mesmos atributos que foram definidos no modelo relacional deste projeto:

- Nome

Assim, mostramos a seguir os géneros musicais definidos:

```
CREATE (Rock:Genero {nome:'Rock'}),  
       (Pop:Genero {nome:'Pop'}),  
       (Indie:Genero {nome:'Indie'}),  
       (HipHop:Genero {nome:'Hip Hop'}),  
       (Alt:Genero {nome:'Alternativo'})
```



**Figura 7 – Nós dos géneros musicais.**

### 3.2.1.6. Membros

Nos atributos para os membros fizemos uma pequena modificação. Removemos o atributo instrumento que passou para o seu relacionamento com a banda, e nesse mesmo relacionamento adicionamos outro atributo, a data de entrada na banda.

As propriedades definidas para os membros foram as seguintes:

- Nome
- Nascimento
- Falecimento (se for esse o caso)
- Sexo

Assim, foram definidos os seguintes membros:

```
CREATE (JustinVernon:Membro {nome:'Justin Vernon', nascimento:'19810430', sexo:'m'}),  
       (MichaelNoyce:Membro {nome:'Michael Noyce', nascimento:'19850112', sexo:'m'}),  
       (SeanCarey:Membro {nome:'Sean Carey', nascimento:'19800922', sexo:'m'}),  
       (MattMcCaughan:Membro {nome:'Matt McCaughan', nascimento:'19820801', sexo:'m'}),  
       ...
```



**Figura 8 – Nós dos membros.**

### 3.2.2. Relacionamentos

A criação dos nós assentou basicamente nas entidades estabelecidas nos modelos realizados na primeira fase deste projeto. No fundo toda a estrutura é baseada na que já foi desenvolvida, contando com eventuais alterações que achamos benéficas para o projeto, dadas as capacidades de uma base de dados orientada a grafos. Assim, e seguindo esses mesmos modelos, os relacionamentos também irão ser idênticos aos relacionamentos definidos no modelo relacional deste projeto, havendo algumas adições que serão descritas, se for o caso.

#### 3.2.2.1. Artistas que compilam gravações

```
CREATE (BonIver) -[:COMPILOU] -> (ForEmmaForeverAgo),
      (Radiohead) -[:COMPILOU] -> (OKComputer),
      (Coldplay) -[:COMPILOU] -> (ParachutesAlbum),
      (DavidBowie) -[:COMPILOU] -> (HeroesAlbum),
      ...
      
```

#### 3.2.2.2. Localidades dos artistas

```
CREATE (BonIver) -[:DE] -> (EauClaire),
      (Radiohead) -[:DE] -> (Oxford),
      (Coldplay) -[:DE] -> (Londres),
      (DavidBowie) -[:DE] -> (Londres),
      ...
      
```

### 3.2.2.3. Géneros musicais das gravações

```
CREATE (ForEmmaForeverAgo)-[:GENERO_MUSICAL]->(Indie),
       (OKComputer)-[:GENERO_MUSICAL]->(Rock),
       (ParachutesAlbum)-[:GENERO_MUSICAL]->(Indie),
       (HeroesAlbum)-[:GENERO_MUSICAL]->(Rock),
       ...
       ...
```

### 3.2.2.4. Faixas pertencentes a cada gravação

```
CREATE (ForEmmaForeverAgo)-[:CONTEM]->(Flume),
       (ForEmmaForeverAgo)-[:CONTEM]->(LumpSum),
       (ForEmmaForeverAgo)-[:CONTEM]->(SkinnyLove),
       (ForEmmaForeverAgo)-[:CONTEM]->(TheWolves),
       ...
       ...
```

### 3.2.2.5. Membros pertencentes a cada banda

O atributo do instrumento do membro migrou para o relacionamento entre o membro e a banda a que pertence neste modelo não relacional. Esta maneira é mais correta visto que um indivíduo pode tocar diferentes instrumentos nas diferentes bandas que pertence. Foi também adicionada a propriedade do ano de entrada desse membro na banda.

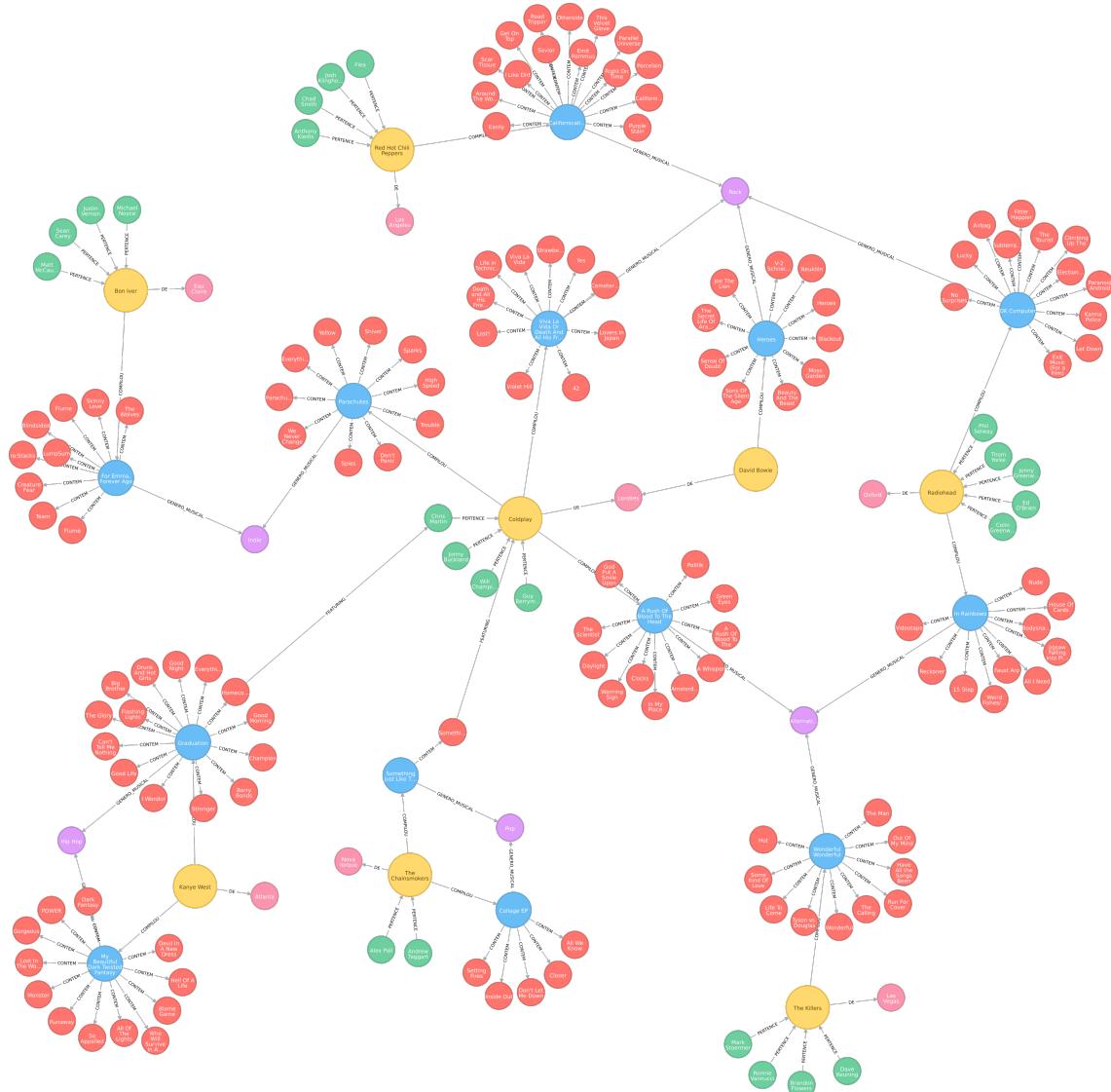
```
CREATE (JustinVernon)-[:PERTENCE {entrada:2006, funcao:'vocalista'}]->(BonIver),
       (MichaelNoyce)-[:PERTENCE {entrada:2007, funcao:'guitarrista'}]->(BonIver),
       (SeanCarey)-[:PERTENCE {entrada:2007, funcao:'teclista'}]->(BonIver),
       (MattMcCaughan)-[:PERTENCE {entrada:2007, funcao:'baterista'}]->(BonIver),
       ...
       ...
```

### 3.2.2.6. Participações em faixas (*featuring*)

Esta nova relação é um acréscimo que não existia no modelo relacional. Irá acrescentar novas informações à base de dados, aumentando a sua coerência e informação disponível.

```
CREATE (Homecoming)-[:FEATURING]->(ChrisMartin),
       (SJLT)-[:FEATURING]->(Coldplay)
```

### 3.2.3. Grafo final do Neo4j



**Figura 9** – Esquema do Neo4j.

### **3.3. Seleção e definição de índices e restrições**

Os índices desempenham uma função muito importante no desempenho das *queries* no Neo4j. Os índices permitem selecionar imediatamente um nó, sem ter de atravessar a árvore. Num grafo de pequenas dimensões a implementação de índices acaba por ser prejudicial, no entanto neste caso a longo prazo irá compensar visto que esta BD irá alojar uma grande quantidade de dados.

Assim, definimos um índice (a negrito) para cada etiqueta (entidade) visto que todas irão estar sujeitas a queries:

- Banda:
    - **Nome (*unique*)**
    - Início
    - Fim
    - Número de membros

- Solo:
  - **Nome (*unique*)**
  - Início
  - Fim
  - Nascimento
  - Falecimento
  - Sexo
- Gravação (álbum, *EP*, *single*):
  - **Nome**
  - Ano
  - Pontuação
- Faixa:
  - **Nome**
  - Duração
  - Número
- Localidade:
  - **Cidade (*unique*)**
  - **País (*unique*)**
- Género musical:
  - **Nome (*unique*)**
- Membro:
  - **Nome**
  - Nascimento
  - Falecimento
  - Sexo

Analisando também estas propriedades numa perspetiva de “chave primária” das etiquetas achamos conveniente definir restrições de unicidade para algumas das propriedades. Nem todas as propriedades serão únicas no entanto, visto que há propriedades que se podem repetir em vários nós (podem haver duas faixas com o mesmo título por exemplo):

```
CREATE CONSTRAINT ON (:Banda) ASSERT nome IS UNIQUE;
CREATE CONSTRAINT ON (:Solo) ASSERT nome IS UNIQUE;
CREATE CONSTRAINT ON (:Genero) ASSERT nome IS UNIQUE;
CREATE CONSTRAINT ON (:Localidade) ASSERT (cidade,pais) IS UNIQUE;
```

Infelizmente o *Neo4j* não permite a definição de duas propriedades *Unique* para a mesma etiqueta. Felizmente o *Neo4j* dispõe do comando *Node Key* (versão *Enterprise*), cujo funcionamento é semelhante:

```
CREATE CONSTRAINT ON (:Localidade) ASSERT (pais, cidade) IS NODE KEY;
```

A definição de propriedades como únicas (*IS UNIQUE*) já as define como índices, logo só resta registar as outras propriedades não únicas como índices:

```
CREATE INDEX ON :Localidade(cidade);
CREATE INDEX ON :Localidade(pais);
CREATE INDEX ON :Album(nome);
CREATE INDEX ON :EP(nome);
CREATE INDEX ON :Single(nome);
CREATE INDEX ON :Faixa(nome);
CREATE INDEX ON :Membro(nome);
```

### 3.4. Tradução das interrogações do utilizador para Cypher

Para esta etapa, usaremos as *queries* propostas pelo utilizador que foram selecionadas na primeira fase. Estas *queries* já haviam sido corrigidas recorrendo à álgebra relacional e o seu correto funcionamento comprovado em SQL. Essas *queries* serão agora expressas em Cypher e em SQL para podermos verificar se os resultados são semelhantes em ambos os modelos:

- **Query extra: qual a duração total de um determinado álbum? (seja o álbum “A Rush Of Blood To The Head”, por exemplo)**

#### CYPHER:

```
MATCH (g) - [:CONTEM] -> (f:Faixa)
WHERE g.nome = 'A Rush Of Blood To The Head'
RETURN DISTINCT SUM(f.duracao) AS Duracao
```

Duracao
54.14999999999999

**Figura 10** – Resultado da Query extra em Cypher.

#### SQL:

```
SELECT SUM(f.duracao) AS Duracao
FROM Faixa f, Gravacao g
WHERE f.id_album = g.id_album
AND g.titulo = 'A Rush Of Blood To The Head' ;
```

Duracao
54.15

**Figura 11** – Resultado da Query extra em SQL.

- **1ª Query: quais as gravações lançadas por uma banda num ano? (seja a banda “Radiohead” e o ano 1997, por exemplo)**

#### CYPHER:

```
MATCH (b:Banda) - [:COMPILOU] -> (g)
WHERE b.nome = 'Radiohead'
AND g.ano = 1997
RETURN DISTINCT b.nome AS Banda, g.nome AS Gravacao, g.pontuacao AS Pontuacao
```

Banda	Gravacao	Pontuacao
"Radiohead"	"OK Computer"	10

**Figura 12** – Resultado da 1ª Query em Cypher.

#### SQL:

```
SELECT a.nome AS Banda, g.titulo AS Gravacao, g.pontuacao AS Pontuacao
FROM Artista a, Gravacao g
WHERE a.nome = 'Radiohead'
AND g.ano = 1997;
```

Banda	Gravacao	Pontuacao
Radiohead	OK Computer	10

**Figura 13 – Resultado da 1ª Query em SQL.**

- **2ª Query: quais as músicas lançadas por todos os artistas de um determinado país (seja Reino Unido, por exemplo), por ordem alfabética?**

**CYPHER:**

```
MATCH (a) -[:COMPILOU]->(g), (g)-[:CONTEN]->(f:Faixa), (a)-[:DE]->(l:Localidade)
WHERE l.pais = 'Reino Unido'
RETURN DISTINCT a.nome AS Artista, g.nome AS Gravacao, f.nome AS Faixa
ORDER BY a.nome, g.nome, f.nome;
```

Artista	Gravacao	Faixa
"Coldplay"	"A Rush Of Blood To The Head"	"A Rush Of Blood To The Head"
"Coldplay"	"A Rush Of Blood To The Head"	"A Whisper"
"Coldplay"	"A Rush Of Blood To The Head"	"Amsterdam"
"Coldplay"	"A Rush Of Blood To The Head"	"Clocks"
"Coldplay"	"A Rush Of Blood To The Head"	"Daylight"
"Coldplay"	"A Rush Of Blood To The Head"	"God Put A Smile Upon Your Face"

**Figura 14 – Pequeno excerto do resultado da 2ª Query em Cypher.**

**SQL:**

```
SELECT a.nome AS Artista, g.titulo AS Gravacao, f.nome AS Musica
FROM Artista a
INNER JOIN Localidade AS l ON l.id_localidade = a.localidade
INNER JOIN Gravacao AS g ON g.id_artista = a.id_artista
INNER JOIN Faixa AS f ON f.id_album = g.id_album
WHERE l.pais = 'Reino Unido'
ORDER BY Artista, Gravacao, Musica;
```

Artista	Gravacao	Musica
Coldplay	A Rush Of Blood To The Head	A Rush Of Blood To The Head
Coldplay	A Rush Of Blood To The Head	A Whisper
Coldplay	A Rush Of Blood To The Head	Amsterdam
Coldplay	A Rush Of Blood To The Head	Clocks
Coldplay	A Rush Of Blood To The Head	Daylight
Coldplay	A Rush Of Blood To The Head	God Put A Smile Upon Your Face

**Figura 15 – Pequeno excerto do resultado da 2ª Query em SQL.**

- **3ª Query: quais as músicas das bandas constituídas por um determinado membro? (seja o membro “Josh Klinghoffer”, por exemplo)**

**CYPHER:**

```
MATCH (m:Membro) -[:PERTENCE]->(b:Banda), (b:Banda) -[:COMPILOU]->(g),
(g)-[:CONTEN]->(f:Faixa)
WHERE m.nome = 'Josh Klinghoffer'
RETURN DISTINCT b.nome AS Banda, g.nome AS Gravacao, f.nome AS Faixa
```

Banda	Gravacao	Faixa
"Red Hot Chili Peppers"	"Californication"	"Road Trippin!"
"Red Hot Chili Peppers"	"Californication"	"Parallel Universe"
"Red Hot Chili Peppers"	"Californication"	"Otherside"
"Red Hot Chili Peppers"	"Californication"	"Right On Time"
"Red Hot Chili Peppers"	"Californication"	"Scar Tissue"

**Figura 16 – Pequeno excerto do resultado da 3ª Query em Cypher.**

**SQL:**

```
SELECT a.nome AS Banda, g.titulo AS Gravacao, f.nome AS Faixa
FROM Faixa f
INNER JOIN Gravacao AS g ON g.id_album = f.id_album
INNER JOIN Artista AS a ON a.id_artista = g.id_artista
INNER JOIN Membro_pertence_Banda AS mpb ON mpb.id_artista = a.id_artista
INNER JOIN Membro AS m ON mpb.id_membro = m.id_membro
WHERE m.nome = 'Josh Klinghoffer';
```

Banda	Gravacao	Faixa
▶ Red Hot Chili Peppers	Californication	Around the World
Red Hot Chili Peppers	Californication	Parallel Universe
Red Hot Chili Peppers	Californication	Scar Tissue
Red Hot Chili Peppers	Californication	Otherside
Red Hot Chili Peppers	Californication	Get on Top

**Figura 17 – Pequeno excerto do resultado da 3ª Query em SQL.**

Os resultados obtidos na execução das queries foram os esperados e coincidem com os obtidos em SQL, pelo que podemos afirmar que tanto a implementação do sistema como a tradução das *queries* para Cypher estão corretas.

## 4. Conclusões e Trabalho Futuro

A realização deste trabalho foi fulcral para cimentar os conhecimentos adquiridos nas aulas acerca de bases de dados não relacionais, neste caso bases de dados orientadas a grafos, com o auxílio do *Neo4j*.

Numa fase inicial deste projeto abordamos a simples conversão de modelos através do comando “*Load CSV*” do *Neo4j*, no entanto consideramos que não iríamos tirar o melhor proveito do conceito de base de dados não relacional através de um simples “copiar-colar”. Analisamos então o que iríamos usar do modelo relacional definido na primeira fase deste projeto e definimos um conjunto de atributos (propriedades) para cada entidade (etiqueta) e abordamos a hipótese de poder criar novos tipos de dados. Concluímos que não iriam haver alterações de grande dimensão. Passamos então à criação dos nós e dos relacionamentos, efetuando eventualmente alterações que achamos oportunas fazer. Definimos índices e restrições para auxiliar na execução das *queries* e coerência de dados e por fim desenvolvemos código *Cypher* que, usando o grafo definido, fosse capaz de responder às interrogações definidas pelo utilizador na primeira parte deste projeto.

Conseguindo responder às interrogações propostas pelo utilizador, demos por concluído o desenvolvimento da base de dados, estando agora aberta à inclusão de mais dados e de mais tipos de dados, dada a flexibilidade dos modelos de dados não relacionais.

Adaptada a este novo modelo é agora possível aumentar a quantidade de dados desta base de dados, respondendo de forma rápida a qualquer interrogação proposta pelo utilizador. Este modelo apresenta ainda outras vantagens, como o seu escalonamento e flexibilidade.

Analizando o nosso desempenho nesta fase do projeto, podemos avaliar como positivo. Todos os passos tomados nesta fase foram os mais corretos na nossa opinião, sendo que todas as decisões tomadas até aqui foram debatidas e questionadas antes de serem tomadas.

Este projeto contribuiu bastante para a consolidação dos conhecimentos já adquiridos nas aulas e na aquisição de novos conhecimentos.

Numa fase futura, bases de dados não relacionais serão sem dúvida uma opção a ser considerada, tendo em conta as características da base de dados a ser desenvolvida.

## **Referências**

- [1] I. Robinson, J. Webber e E. Eifrem, Graph Databases, 2<sup>a</sup> ed., O'Reilly, 2015.
- [2] Neo4j, Inc., “Cypher Refcard,” [Online]. Available: <https://neo4j.com/docs/cypher-refcard/current/>. [Acedido em 12 1 2018].

## **Lista de Siglas e Acrónimos**

**BD**      Base de dados

**EP**      *Extended Play*