Universidade do Minho
Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Lectivo de 2017/2018

Sétima Arte: gestão de sessões de cinema

Lúcia Abreu

Novembro, 2017

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Sétima Arte: gestão de sessões de cinema

Lúcia Abreu

Novembro, 2017

Resumo

Este projeto consiste na migração de dados contidos no sistema relacional (MySQL) implementado na parte I do projeto, para um sistema não relacional (MongoDB), no âmbito do tema de gestão de sessões de cinema, nomeadamente a base de dados para a reserva de bilhetes de sessões de cinema, pedida pela empresa **Braga Cinema Center (BCC).**

Ao longo deste relatório serão apresentados todos os passos que efetuados durante este processo de migração.

Inicialmente é apresentada uma breve introdução e contextualização do projeto, o caso de estudo em questão bem como a motivação do projeto e os seus principais objetivos.

De seguida é introduzido o paradigma **Não Relacional** de Sistemas de Base de Dados - o **NoSQL** - e também o **MongoDB**, o modelo de dados orientado a documentos utilizado.

Logo após, é abordado todo o processo de migração que levou à implementação da Base de Dados. Para a migração dos dados recorreu-se à ferramenta **Mongify**, contudo antes foi realizada uma análise de modo a decidir como seria o novo esquema da base de dados em NoSQL, de forma a continuar equivalente ao modelo da base de dados relacional.

Após a migração definiu-se e implementou-se algumas queries no MongoDB com a finalidade de demonstrar que se fez o povoamento da base de dados. Por fim, fez-se uma análise crítica do trabalho realizado.

A base de dados foi implementada em MySQL uma vez que é um SGBD *open-source* com bastante reconhecimento e suporte da comunidade. A base de dados implementada permite a validação dos dados e das restrições gerais identificadas nos requisitos. Desta forma garante-se consistência de dados e regras de negócio.

Área de Aplicação: Arquitetura e Migração de Sistemas de Bases de Dados Não Relacional.
Palavras-Chave: Base de Dados, Modelo Relacional, Modelo Não Relacional, MongoDB, MySQL, Mongify, Migração de Dados, Queries NoSQL.

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

A parte II do projeto realizado no âmbito da unidade curricular de Base de Dados, consiste na migração de dados contidos no sistema relacional (MySQL) implementado na parte I do projeto para um sistema não relacional (MongoDB).

A realização deste trabalho tem como objetivo introduzir e dar algumas noções sobre NoSQL, planeando, implementando e executando um projeto em Sistemas de Bases de dados (SBD) não relacionais, neste caso no MongoDB.

O trabalho está organizado em seis capítulos. Além deste capítulo inicial, o segundo capítulo introduz o paradigma Não Relacional e o modelo MongoDB, o terceiro capítulo especifica quais as decisões tomadas e o porquê destas, de modo a desenvolver um esquema para a base de dados não relacional que seja equivalente ao esquema da base de dados relacional da parte I do projeto. No quarto capítulo, explica-se como se realizou o processo de migração, que ferramentas se utilizaram e detalha-se todos os passos do processo. No quinto capítulo do trabalho definiu-se e implementou-se algumas *queries* no MongoDB de modo a demonstrar que a base de dados não relacional se encontra povoada, isto é, o processo de migração foi realizado com sucesso. Por fim, no sexto capítulo do trabalho realiza-se uma análise crítica do trabalho realizado, visando comparar sempre que possível, o modelo e a funcionalidades agora implementadas com aquelas que se implementou, aquando da realização da parte I do projeto numa base de dados relacional.

1.1. Contextualização

Cada vez mais o cinema clássico tem menos público, principalmente público jovem. De ano para ano, são cada vez menos as pessoas que se deslocam ao cinema para assistir a filmes mais clássicos, curtas-metragens e filmes locais.

Tradicionalmente as salas de cinema que oferecem este tipo de filmes não possuem um sistema de reservas *online*. E são prejudicadas por este fator, pois o público mais jovem, com um ritmo de vida cada vez mais frenético, e que passam cada vez mais tempo no local de trabalho, preferem fazer a compra/reserva dos seus bilhetes de cinema *online*. Obrigando o cliente a fazer a reserva ou compra presencialmente, após todos estes fatores, acaba por restar pouco tempo para tratar dos afazeres mais banais do dia-a-dia e da família.

Com o avanço tecnológico, hoje em dia é cada vez mais usual efetuar-se todo ou quase todo o tipo de compras online. O tempo livre durante um dia de trabalho é cada vez

menor, para que uma pessoa possa deslocar-se a uma bilheteira de cinema para comprar um bilhete. Tal não é necessário nas principais salas de cinema que projetam os típicos *blockbusters* de *Hollywood*, porém as salas de cinemas mais clássicas exigem-no. Porque não facilitar a vida das pessoas ao disponibilizar-lhes um sistema de reservas de bilhetes de cinema *online*? Assim sendo, basta ter acesso à internet e aceder ao site da empresa ou à sua aplicação, registar-se numa conta e depois efetuar a compra, de forma simples, prática e rápida. Desta forma não é necessário ter de se lidar com dinheiro.

Por esse motivo, a ***Braga Cinema Center (BCC)*** decidiu investir num sistema de reservas de sessões de cinema *online*. A BCC é uma empresa que projeta filmes clássicos, curtas-metragens, filmes locais, entre outros, e constatou que a ausência de um sistema de compra e reserva de bilhetes de sessões de cinema *online* na sua empresa era uma grande lacuna, pelo que procedeu à implementação desse sistema.

Após a disponibilização da plataforma de reservas *online*, o número de reservas aumentou consideravelmente, o que leva a que a empresa tenha de garantir a estabilidade da elevada quantidade de dados.

Assim, a equipa responsável pela gestão da base de dados da BCC, que até então tinha implementado um sistema de gestão de base de dados relacional, viu-se obrigada a fazer mudanças para gerir mais eficazmente o elevado número de reservas de bilhetes de cinema com que a empresa de transportes tinha agora de lidar. Para tal, foi tomada a decisão de mudar para um sistema de base de dados não relacional, utilizando o modelo MongoDB, - que, à partida, conseguiria assegurar a estabilidade dos dados tanto na situação atual como para o futuro da empresa.

1.2. Apresentação do Caso de Estudo

A *Braga Cinema Center* (BCC)

A BCC é uma empresa local, da cidade de Braga, que projeta filmes clássicos, curtas-metragens, filmes locais e também internacionais, presentes nos maiores festivais de cinema europeus, como o Festival de Cinema de Veneza.

A fim de responder às exigências do mercado, como referido na contextualização anterior, a BCC decidiu disponibilizar aos seus clientes um sistema de reservas de bilhetes de sessões de cinema *online*.

Um cliente poderá efetuar uma reserva de um bilhete numa sessão de um filme, sendo logo paga no ato da reserva. Para tal este deverá fornecer o seu nome, *username*, data de nascimento, telefone, email e password, de forma a se poder registar numa conta que lhe irá permitir reservar e consultar as suas reservas.

As salas de cinema da empresa são de três tipos: *Basic* (capacidade máxima para 15 pessoas), *Silver* (capacidade máxima para 20 pessoas) e *Platinum* (capacidade máxima para 25 pessoas).

No processo de reserva o cliente deverá qual sessão de um filme deseja assistir, e conforme a escolha tomada consultar o(s) lugar(es) disponíveis de forma a escolher qual o(s) lugar(es) que deseja. Como o pagamento é realizado no ato da reserva, a empresa decidiu não permitir o cancelamento de reservas.

Com vista a atrair e alcançar uma vasta população alvo, a empresa dispõe de um desconto para jovens e outro para idosos. Os jovens com idade inferior a 25 anos terão um desconto de 15% nas reservas efetuadas e as pessoas com idade superior a 65 anos terão um desconto de 25%. E para atrair ainda mais público, principalmente os verdadeiros amantes de filmes clássicos, institui um desconto para membros *Premium*, que atribui um desconto de 30% em todas as sessões de cinema.

A BCC pretende analisar o comportamento dos nossos clientes e com isso realocar melhor os seus recursos, logo deverá ser possível fazer uma listagem de todas as reservas de uma determinada sessão ou de um filme.

1.3. Motivação e Objetivos

1.3.1 Motivação

Este projeto consiste na migração de dados e implementação de um sistema de base de dados NoSQL com base no projeto do trabalho I: sistema de gestão de reservas de bilhetes de cinema.

Este tipo de plataforma apresenta alguma complexidade a nível de implementação do sistema de base de dados, pois é necessário lidar com um grande número de clientes, bilhetes, lugares disponíveis de cada sessão, filmes, etc.

Com o desenvolvimento deste sistema pretende-se simplificar o processo de reserva de um bilhete de uma sessão de cinema, pois as reservas efetuam-se *online*, obtendo um sistema de gestão de bases de dados capaz de aceitar um elevado tráfego de utilizadores ao mesmo tempo que assegura a performance pretendida, sendo esta uma das principais vantagens dos sistemas NoSQL.

Pretende-se então migrar os dados para um sistema não relacional, mas continuar a manter o foco na realidade e nos problemas que surgem aos utilizadores, lacunas a preencher de modo a satisfazer o cliente e o porquê da empresa necessitar de uma Base de Dados adequada:

- Os clientes queixavam-se que não conseguiam fazer reservas com antecedência e quando iam comprar o bilhete já não havia lugares disponíveis (lugares esgotados)

- Os clientes acabavam por escolher as salas de cinema das superfícies comerciais com o seu serviço *online*, porque não tinham acesso à disponibilidade de lugares e não tinham tempo nem paciência para se deslocar aos postos de venda
- Os utilizadores manifestaram vontade de escolher o seu próprio lugar
- Disponibilizar serviços a pensar no utilizador para aumentar as vendas (descontos para membros, possibilidade de reserva)
- Possibilitar uma futura aplicação mobile onde cada utilizador possa gerir as suas reservas (aproximar o cliente da empresa)
- Desconhecimento do perfil do cliente e recolha de estatísticas/informação relevantes para otimizações futuras:
- Com quanta antecedência é feita a reserva (estudar possíveis promoções)
- Criação de sistema de pontos
- Gestão de recursos (salas, lugares) antecipada (em vez de previsão)

1.3.2 Objetivos

Nesta secção foram determinados os objetivos que a empresa BCC visa alcançar com a implementação de uma Base de dados.

- Possibilitar um serviço ao cliente, onde este possa efetuar as suas reservas e informatizar esse processo
- Permitir ao utilizador consultar, ter acesso e escolher o lugar que quer reserva
- Gestão de reserva de bilhetes eficiente e centralizada
- Obter uma forma simples de consultar todas as reservas de uma sessão de um filme
- Recolher dados que permitam traçar perfis de clientes
- Futura personalização da experiência

1.4. Estrutura do Relatório

Este relatório está dividido em seis capítulos.

O primeiro capítulo é um capítulo introdutório que faz uma contextualização e apresentação do caso de estudo, assim como aborda a motivação e objetivos para a implementação da base de dados não relacional

O segundo capítulo introduz o paradigma Não Relacional e o modelo MongoDB, o terceiro capítulo especifica quais as decisões tomadas e o porquê destas, no quarto capítulo explica-se como se realizou o processo de migração, no quinto capítulo definiu-se e implementou-se algumas *queries* no MongoDB e por fim, no sexto capítulo do trabalho realiza-se uma análise crítica do trabalho realizado.

2. Paradigma Não Relacional

2.1. Análise e Justificação da Migração para NoSQL

O objetivo com este trabalho é a migração de dados e implementação de um sistema de base de dados não relacional (NoSQL), recorrendo ao modelo **MongoDB** (NoSQL), a pedido da empresa ***Braga Cinema Center*** **(BCC)**.

Desta forma, é importante saber o que são sistemas de base de dados NoSQL e em que medida estes nos trazem vantagens em relação aos sistemas de base de dados relacionais.

Desde há vários anos que os sistemas de base de dados relacionais dominam o mercado, tornando-se ao longo do tempo o sistema padrão para a maioria dos Sistemas de Gestão de Base de Dados (SGBD). Tal deve-se ao modelo na qual este tipo de sistemas se baseiam, o **modelo ACID**, que preservam a integridade de uma dada transação através de Atomicidade, Consistência, Isolamento e Durabilidade.

Tais caraterísticas permitiram ao longo dos anos conservar os sistemas de base de dados relacionais sempre numa posição de superioridade entre os restantes. Revela-se se problemática e não suficientemente eficiente.

Como tal, e devido ao crescente aumento do volume de dados da ***Braga Cinema Center*** **(BCC)**, fruto da implementação de uma tão aguardada plataforma online, leva à necessidade de sistema que encarregue de tratar eficientemente o volume de dados cada vez maior do sistema.

As bases de dados relacionais são extremamente eficientes no armazenamento de informação **estruturada e consistente**. Contudo, há situações em que podem não ser as características mais relevantes no armazenamento de informação. Como tal, começam a surgir soluções alternativas capazes de armazenar e consultar dados **não estruturados**.

As bases de dados relacionais apresentam uma **estrutura muito específica**, que deve ser assegurada durante toda a sua utilização. Logo, a necessidade de realizar mudanças na sua estrutura pode revelar-se uma tarefa demorada, o que não é desejável.

Os sistemas de bases de dados não relacionais apresentam flexibilidade nesta matéria, uma vez que, não apresentando uma estrutura rígida, permitem que alterações na estrutura de armazenamento de dados do sistema seja uma tarefa mais rápida que nas bases de dados relacionais.

Portanto, os principais problemas encontrados com a utilização do Modelo Relacional residem na dificuldade de conciliar o tipo de modelo com o aumento da escalabilidade. Desta forma, surge assim um novo paradigma: **Não Relacional** (NoSQL – Not Only SQL), que procura de certa forma tentar solucionar diversos problemas relacionados com a escalabilidade, performance e disponibilidade dos sistemas de bases de dados não relacionais.

O surgimento dos modelos NoSQL não visa extinguir o uso dos sistemas de bases de dados relacionais, mas utilizar os não relacionais em situações onde é necessária uma maior flexibilidade na estruturação da base de dados.

As bases de dados NoSQL usam diversos modelos de dados, como por exemplo:

- Documentos
- Grafos
- Key/Value
- *Column Oriented*

Como revelado anteriormente, o modelo que será utilizado é o MongoDB. Com esta migração pretendemos obter um sistema de gestão de base de dados capaz de tratar um elevado número de utilizadores, enquanto garante a performance pretendida.

2.2. Modelo de dados em MongoDB

O modelo de dados em MongoDB é um modelo orientado a documentos, onde cada "registo" e respetivos dados são considerados um ***documento***. É um modelo *open-source*, sendo dos mais desenvolvidos e explorados no mercado na atualidade, e também um dos modelos mais usados hoje em dia, por empresas como por exemplo: Facebook, Google, ebay, cisco, entre outras.

O MongoDB armazena os dados em documentos JSON com um esquema flexível, isto é, os dados armazenados não têm de seguir um esquema rígido, podem seguir uma qualquer estrutura que se possa desejar.

Este modelo possui diversas vantagens, que são:

- **Documentos como unidades independentes**:

  O aumento do volume de dados presente nas bases de dados proporcionou a distribuição dos dados por várias máquinas (*clusters*). Com os documentos a comportarem-se como unidades independentes, é possível que estes sejam distribuídos pelos vários *clusters,* melhorando assim consideravelmente o desempenho e permitindo também maior escalabilidade.

- **Dados não estruturados são mais facilmente armazenados:**

  As inserções num sistema de base de dados orientado a documentos revelam um desempenho consideravelmente melhor que numa base de dados relacional. Isto porque, e uma vez que este modelo não apresenta um *schema*, não existe uma série de considerações a serem verificadas.

- **Consultas e transações mais simples:**

  A inexistência de relacionamentos entre documentos traduz-se na inexistência de transações e *joins* de tabelas. Assim, sempre que possível, um documento possibilita todas as informações necessárias.

Não obstante, este modelo possui também desvantagens:

- **Redundância de dados:**

  Esta é uma situação muito comum nestes tipos de modelos de bases de dados. O facto de não existirem relacionamentos entre os documentos implica que existam muitos dados repetidos.

- **Inconsistência:**

  Ao contrário dos sistemas de gestão de bases de dados relacionais, o MongoDB não garante a consistência dos dados presentes, pelo que é responsabilidade do programador da aplicação garantir que todos as inserções ou alterações nos dados os mantêm consistentes.

3. Migração dos Dados para MongoDB

O modelo de dados em MongoDB orientado a documentos foi concebido a partir do modelo conceptual e do modelo lógico relacional obtidos no trabalho anterior. Na **Figura 1** encontra-se o modelo de dados lógico relacional.

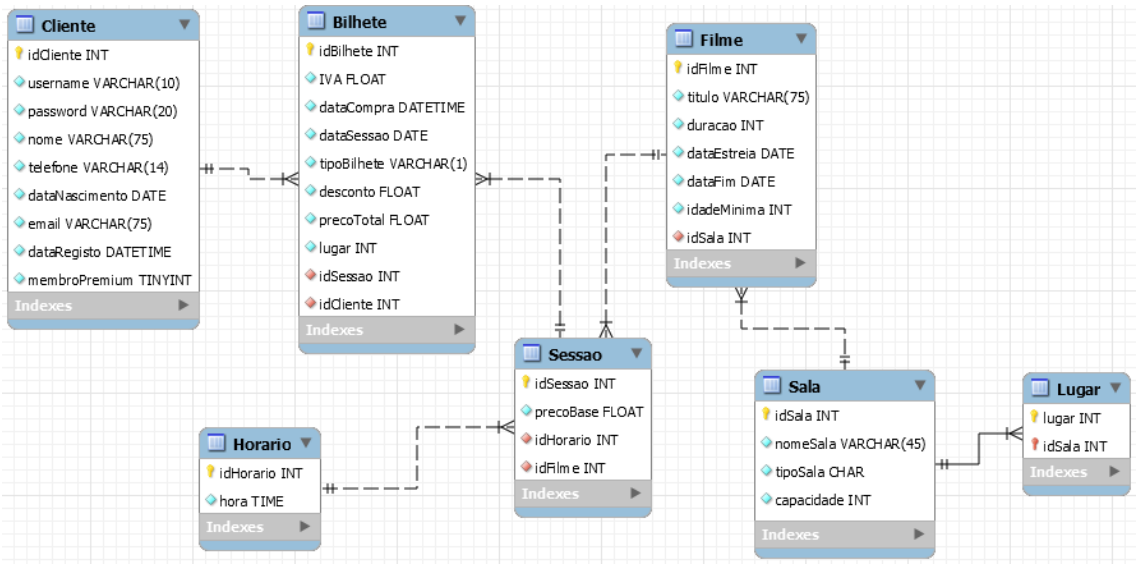

Figura 1 - Modelo de dados lógico relacional da base de dados BCC.

Para este processo, é importante referir que em MongoDB:

- **Documento**: a unidade de dados básica e corresponde a uma linha de uma tabela no modelo relacional
- **Coleção**: conjunto de documentos de um certo tipo, pode ser pensada como uma tabela no modelo relacional
- **_id**: cada documento tem uma chave especial que o identifica de forma única dentro da coleção

De seguida encontram-se definidas cada uma das etapas e as conclusões retiradas da execução de cada uma delas.

3.1. Passo 1 - Identificar as coleções

As chaves estrangeiras definidas no modelo lógico e que são atributos das tabelas, consequentemente, vão ser atributos de cada coleção o que permite referenciar a coleção correspondente.

Por exemplo, a coleção Filme tem como atributo a referência para identificar a Sala que está associada àquele filme (consequência de a tabela Filme ter como um dos atributos a chave estrangeira **idSala** para identificar a respetiva Sala).

Contudo, em MongoDB, os dados têm um esquema flexível e as coleções não forçam a estrutura de um documento. Assim sendo, no passo 2 ir-se-á analisar as relações existentes no modelo lógico de forma a verificar se alguma das coleções identificadas poderá ser embebida numa outra coleção.

Para definir o modelo em MongoDB é necessário identificar as várias coleções existentes na base de dados. De forma geral, cada tabela pode originar uma coleção e, portanto, por transformação direta originar-se-ia 5 coleções resultantes das 5 tabelas existentes no modelo lógico relacional, sendo essa uma das **possibilidades**:

- Coleção Cliente
- Coleção Bilhete
- Coleção Sessão
- Coleção Horário
- Coleção Filme
- Coleção Sala
- Coleção Lugar

Mas para definir o modelo em MongoDB é necessário também as relações.

3.2. Passo 2 - Analisar as relações 1:N

Todas as relações que se encontram no modelo lógico da base de dados BCC na Figura 1 são relações de 1:N. Em MongoDB, cada uma destas relações tem de ser analisada para decidir qual deve ser a estratégia a adotar:

1. ***Documento embebido***: analisa-se se os vários N atributos de um documento podem ser embebidos nesse documento
2. ***Referência a outro documento***: analisa-se se cada um dos N documentos da coleção do lado N deverá ter a referência para o documento do lado 1

Na decisão de uma ou outra estratégia para cada uma das relações 1:N existentes devem ser tidos em conta 3 critérios:

1. Razão entre as operações leitura/escrita
2. Tipos de operações realizadas
3. Crescimento esperado para os documentos

Analisou-se as seguintes **cinco** relações 1:N presentes no modelo de dados lógico relacional da Figura 1:

- Relação Cliente – Bilhete
- Relação Sessão – Bilhete
- Relação Horário – Sessão
- Relação Filme – Sessão
- Relação Sala – Filme
- Relação Sala – Lugar

Na **Tabela 1**, apresenta-se para cada umas das relações, os critérios e a decisão entre adotar a estratégia de colocar a referência num documento para outro, ou optar por colocar o documento como embebido.

Relações 1 : N	**Razão leitura/ escrita**	**Tipo operações esperadas**	**Crescimento esperado**	**Outras**	**Decisão**
Relação Cliente - Bilhete	Espera-se mais operações de leitura ao Cliente e mais inserções de Bilhete	Queries (consultas)	Espera-se que um cliente adquira vários bilhetes, embora seja um crescimento pouco significativo	-	**Embebido** O documento Cliente terá como atributo um *array* em que cada elemento é o Bilhete
Relação Sessão - Bilhete	Espera-se mais operações de leitura a Sessão e mais inserções de Bilhete	Queries (consultas)	Espera-se que uma sessão tenha vários bilhetes, embora seja um número **fixo** (igual à capacidade da sala)	-	**Embebido** O documento Sessão terá como atributo um *array* em que cada elemento é o Bilhete
Relação Horário - Sessão	Espera-se mais operações de leitura a Horário e Sessão	Queries (consultas)	Espera-se que para cada horário se registe um elevado crescimento no número de sessões	-	**Referência** Cada documento da coleção Sessão terá a referência para o respetivo documento Horário

Relação Filme - Sessão	Espera-se mais operações de inserção de Filmes e Sessões	Queries (consultas)	Espera-se que para cada filme se registe um crescimento no número de sessões	-	**<u>Referência</u>** Cada documento da coleção Sessão terá a referência para o respetivo documento Filme
Relação Sala - Filme	Espera-se mais operações de inserção de Filmes e de leitura a Sala	Queries (consultas)	Espera-se que para cada sala se registe um crescimento no número de filmes	-	**<u>Referência</u>** Cada documento da coleção Filme terá a referência para o respetivo documento Sala
Relação Sala - Lugar	Espera-se mais operações de leitura a Sala e a Lugar	Queries (consultas)	O número de lugares de cada sala é fixo, e o número de salas tem pouca tendência a crescer	-	**<u>Embebido</u>** O documento Sala terá como atributo um *array* em que cada elemento é o Lugar

Tabela 1 - Decisão Referência Vs Documento Embebido para cada relação 1:N

Assim sendo,

- **Cliente - Bilhete: <u>Embeber</u>** em cada documento Cliente a lista dos seus bilhetes. O documento Cliente terá como atributo um *array* em que cada elemento é o Bilhete.
    - Espera-se que cada cliente faça a compra de vários bilhetes, embora o número não seja significativo, podendo os bilhetes serem embebidos na coleção Cliente.
    - A nível de *queries*, por exemplo, facilita a consulta do número de bilhetes de cada cliente, e outro tipo de consultas relativas à relação Cliente - Bilhete.

- **Sessão - Bilhete : <u>Embeber</u>** em cada documento Sessão a lista dos seus bilhetes. O documento Sessão terá como atributo um *array* em que cada elemento é o Bilhete.
    - Cada sessão apenas pode possuir um número fixo de bilhetes (a capacidade da sala), não sendo problemático então embeber os bilhetes na coleção Sessão.
    - Relativamente a *queries*, embeber os bilhetes facilita a consulta dos vários elementos e estatísticas de uma sessão (lugares reservados, total de lucro de cada sessão, etc), o que implica um elevado número de leituras aos bilhetes de cada sessão.

- **Sala - Lugar : <u>Embeber</u>** em cada documento Sala a lista dos seus lugares. Assim, cada documento de uma sala mantém no próprio documento a informação dos seus lugares

- **Restantes relações :** Cada um dos N documentos da coleção do lado N deverá ter a **<u>referência</u>** para o documento do lado 1.

    - Visto o número de consultas e inserções se revelar em menor número, optou-se por utilizar referências para os respetivos documentos.

Na **Figura 2** encontra-se o modelo de dados em MongoDB obtido.

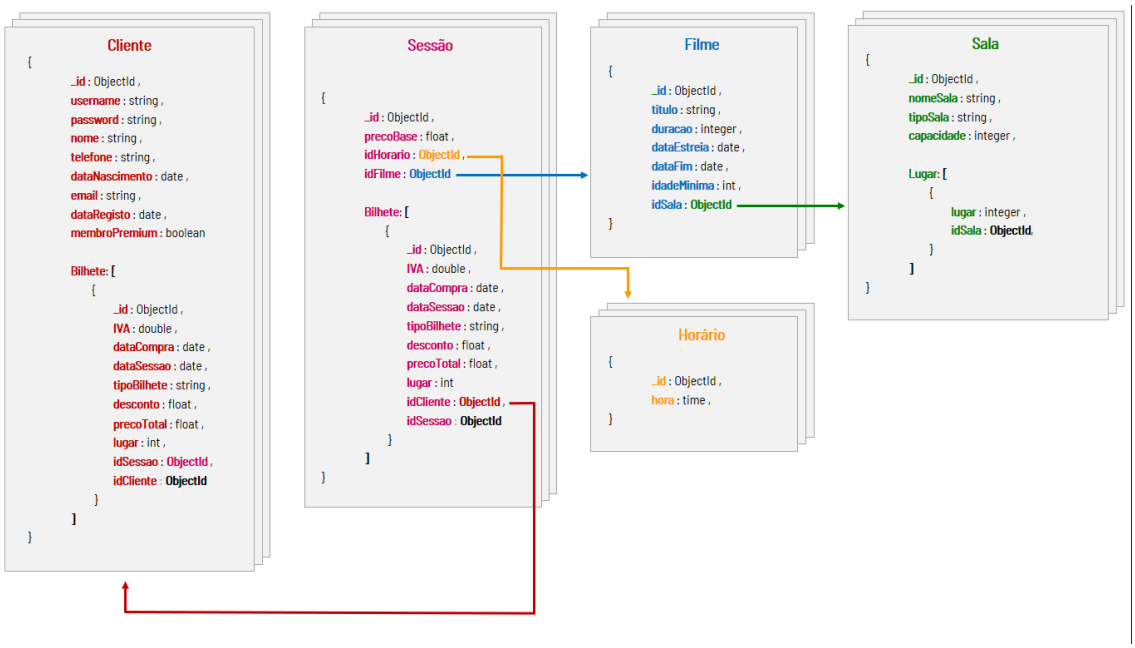

Figura 2 - Modelo de dados em MongoDB para a base de dados BCC

Para cada documento de cada coleção é gerado um _id automaticamente pelo MongoBD (ObjectID):

- permite que cada documento seja **<u>identificado de forma única</u>**
- permite ter referência de um documento para outro (conforme relações)
- corresponde à chave primária no modelo relacional

Além do _id, cada documento, possui os mesmos atributos que já a respetiva tabela detinha no modelo relacional.

4. Processo de Migração de Dados do MySQL para MongoDB

Após a definição do esquema da base de dados NoSQL, o próximo passo foi proceder à migração dos dados da base de dados relacional para o novo sistema não relacional.

Depois de alguma pesquisa, conclui-se que havia duas alternativas para realizar a migração de dados:

1. criar um programa/script numa linguagem de programação à escolha (por exemplo Java), que usasse clientes de MySQL e MongoDB para ler os dados do MySQL e que escrever no formato especificado na secção anterior, no MongoDB;

2. usar uma ferramenta *open-source* chamada **mongify,** que foi desenhada para fazer migrações de dados de sistemas relacionais para o MongoDB.

A primeira opção é mais flexível, porque ter-se-ia o controlo total sobre como seria feita a migração. No entanto, após estudar com algum cuidado o **mongify**, verificou-se que tinha suporte para tudo o que se desejava fazer, quer em termos de tipos de dados suportados, quer em termos de como seria feita a migração de trabalhos: por referência (como já é normal em base de dados relacionais) ou embebendo uma tabela dentro de outra (opção mais usada em base de dados de documentos como o MongoDB).

4.1. Migração com Mongify

De forma a realizar a migração de dados do MySQL para o MongoDB são necessários dois ficheiros:

1. Um ficheiro de configuração das bases de dados, para o **mongify** se conseguir ligar a ambas;
2. Um ficheiro de tradução, onde se especifica todas as ações a tomar na migração dos dados em si.

O ficheiro de configuração das bases de dados permite ao **mongify** verificar se se consegue ligar às bases de dados MySQL e MongoDB. A este ficheiro deu-se o nome ***database.config***:

```
sql_connection do
  adapter   "mysql"
  host      "localhost"
  username  "root"
  password  ""
  database  "bccDB"
  encoding  "utf8"
end

mongodb_connection do
  host      "localhost"
  database  "bccDB"
end
```

O ficheiro de tradução que permite a especificação da migração dos dados no MySQL para a base de dados MongoDB, isto é, é o ficheiro que permite fazer o mapeamento, e posterior migração, entre uma base de dados relacional e a MongoDB. A este ficheiro deu-se o nome ***translation.rb***:

```
table "Cliente" do
    column "idCliente", :key
    column "username", :string
    column "password", :string
    column "nome", :string
    column "telefone", :string
    column "dataNascimento", :date
    column "email", :string
    column "dataRegisto", :datetime
    column "membroPremium", :boolean
end

table "Bilhete",
    :embed_in => "Cliente",
    :on => "idCliente" do

    column "idBilhete", :key
    column "IVA", :float
    column "dataCompra", :datetime
    column "dataSessao", :date
    column "tipoBilhete", :string
    column "desconto", :float
    column "precoTotal", :float
    column "lugar", :integer
    column "idSessao", :integer, :references => "Sessao"
    column "idCliente", :integer, :references => "Cliente"
end

table "Sessao" do
    column "idSessao", :key
    column "precoBase", :float
    column "idHorario", :integer, :references => "Horario"
    column "idFilme", :integer, :references => "Filme"
end
```

```
table "Bilhete",
    :embed_in => "Sessao",
    :on => "idSessao" do

    column "idBilhete", :key
    column "IVA", :float
    column "dataCompra", :datetime
    column "dataSessao", :date
    column "tipoBilhete", :string
    column "desconto", :float
    column "precoTotal", :float
    column "lugar", :integer
    column "idSessao", :integer, :references => "Sessao"
    column "idCliente", :integer, :references => "Cliente"
end

table "Sala" do
    column "idSala", :key
    column "capacidade", :integer
    column "nomeSala", :string
    column "tipoSala", :string
end

table "lugar",
    :embed_in => "Sala",
    :on => "idSala" do

    column "lugar", :integer
    column "idSala", :integer, :references => "Sala"
end
```

```
table "Filme",
    column "idFilme", :key
    column "titulo", :string
    column "duracao", :integer
    column "dataEstreia", :date
    column "dataFim", :date
    column "idadeMinima", :integer
    column "idSala", :integer, :references => "Sala"
end

table "Horario",
    column "idHorario", :key
    column "hora", :time
end
```

Como se pode verificar na figura, o mapeamento é relativamente direto, pois todos os tipos de dados usados no MySQL têm correspondência direta no MongoDB. Além de especificar o nome das tabelas e as colunas, também foram inseridas “:references =>” nas colunas que eram chaves estrangeiras e que vão ser uma referência para outro documento no MongoDB. Para além disso, todas as chaves primárias têm o tipo “:key”, que permite ao **mongify** usar esse campo para obter o ObjectID usado como identificador único de documentos no MongoDB.

Também foi usado o comando ":embed_in => " para sinalizar as tabelas que vão passar a ser um documento embebido noutra coleção, em vez de serem uma tabela independente como eram no MySQL.

Depois de finalizados estes dois documentos, para realizar a migração de dados para o MongoDB apenas é necessário correr na consola o seguinte comando com os dois ficheiros anteriormente criados:

```
$ mongify process database.config translation.rb
```

5. Queries em MongoDB

Neste capítulo apresentam-se alguns exemplos de *queries* importantes que a base de dados consegue responder.

5.1. Exemplos de *Queries*

QUERY 1: Número de lugares reservados para uma dada sessão

```
> db.Sessao.find( { _id : ObjectId("5a625d9548885afdf4a608f4") }).count()
1
```

QUERY 2: Número de lugares reservados para um determinado dia de sessão

```
> db.Sessao.find( { "bilhetes.dataSessao" : ISODate("2015-05-29T23:00:00Z") } ).count()
1
```

QUERY 3: Obter as reservas/compras de um cliente

```
> db.Cliente.find( { "bilhetes.idCliente" : ObjectId("5a625c8348885afdf4a608f3") } ).pretty()
{
        "_id" : ObjectId("5a625c8348885afdf4a608f3"),
        "username" : "mariaS",
        "password" : "maria1234",
        "nome" : "Maria Santos",
        "telefone" : "+0351253193219",
        "dataNascimento" : ISODate("1983-08-21T00:00:00Z"),
        "email" : "mariasantos@gmail.com",
        "dataRegisto" : ISODate("2018-02-19T00:00:00Z"),
        "membroPremium" : false,
        "bilhetes" : [
                {
                        "_id" : ObjectId("5a62574c48885afdf4a608f5"),
                        "IVA" : 0.13,
                        "dataCompra" : ISODate("2018-02-19T00:00:00Z"),
                        "dataSessao" : ISODate("2015-05-29T23:00:00Z"),
                        "tipoBilhete" : "N",
                        "desconto" : 0,
                        "precoTotal" : 6.22,
                        "lugar" : 1,
                        "idSessao" : ObjectId("5a625d9548885afdf4a608f4"),
                        "idCliente" : ObjectId("5a625c8348885afdf4a608f3")
                }
        ]
}
>
```

QUERY 4: Obter todas as Sessões de um Filme

```
> db.Sessao.find( { "idFilme" : ObjectId("5a6266f45e6a2cb99c23443d") } ).pretty()
{
        "_id" : ObjectId("5a62613348885afdf4a608f6"),
        "precoBase" : 5.5,
        "idHorario" : ObjectId("5a6267205e6a2cb99c234443"),
        "idFilme" : ObjectId("5a6266f45e6a2cb99c23443d")
}
{
        "_id" : ObjectId("5a62613e48885afdf4a608f8"),
        "precoBase" : 7,
        "idHorario" : ObjectId("5a6267365e6a2cb99c234447"),
        "idFilme" : ObjectId("5a6266f45e6a2cb99c23443d")
}
{
        "_id" : ObjectId("5a62612c48885afdf4a608f5"),
        "precoBase" : 3.5,
        "idHorario" : ObjectId("5a6267175e6a2cb99c234441"),
        "idFilme" : ObjectId("5a6266f45e6a2cb99c23443d"),
```

QUERY 5: Obter todas os filmes de uma sala

```
db.Filme.aggregate([
    { $lookup: {
        from: "Sala",
        localField: "idSala",
        foreignField: "_id",
        as : "salas"
    }},
    { $match: {
        "salas.nomeSala" : "Woody Allen"
    }}
])
```

QUERY 6: Obter todos os clientes que reservaram um bilhete de uma determinada sessão

```
> db.Cliente.find( { "bilhetes.idSessao" : ObjectId("5a625aaf48885afdf4a608f2") } ).pretty()
{
        "_id" : ObjectId("5a62477948885afdf4a608f0"),
        "username" : "joaoS",
        "password" : "joao1234",
        "nome" : "João Martins Silva",
        "telefone" : "+0351258843219",
        "dataNascimento" : ISODate("1997-04-11T00:00:00Z"),
        "email" : "joaomsilva@gmail.com",
        "dataRegisto" : ISODate("2013-12-10T00:00:00Z"),
        "membroPremium" : false,
        "bilhetes" : [
                {
                        "_id" : ObjectId("5a62574c48885afdf4a608f1"),
                        "IVA" : 0.13,
                        "dataCompra" : ISODate("2018-02-19T00:00:00Z"),
                        "dataSessao" : ISODate("2014-11-27T00:00:00Z"),
                        "tipoBilhete" : "P",
                        "desconto" : 0.15,
                        "precoTotal" : 6.24,
                        "lugar" : 7,
                        "idSessao" : ObjectId("5a625aaf48885afdf4a608f2"),
                        "idCliente" : ObjectId("5a62477948885afdf4a608f0")
                }
        ]
}
```

6. Conclusões

No capítulo 1 deste relatório contextualizou-se e analisou-se o caso em estudo, isto é, qual a realidade em que a **Braga Cinema Center** está inserida e como deseja migrar de uma base de dados relacional para um sistema de base de dados não relacional, de modo a conseguir lidar com o grande aumento dos dados do seu sistema, fruto da implementação do sistema de reservas *online*.

Após realizado o trabalho de modelação e implementação da base de dados BCC na forma relacional no MySQL e na forma não relacional em MongoDB, vamos analisar e comparar as duas abordagens à luz da nossa experiência com este trabalho.

O MongoDB permite que se embebeda documentos ou coleções de documentos dentro de outros documentos, o que traz um nível de flexibilidade não existente com bases de dados relacionais. É possível adicionar documentos a uma coleção, sem que todos os atributos desse documento estejam presentes nos documentos da coleção atual. No modelo relacional, para fazer o equivalente, era preciso primeiro alterar o esquema da tabela para suportar todos os atributos de todas as linhas e só depois inserir a linha nova. Adicionalmente, os novos atributos iriam necessariamente estar presente em todas as linhas da tabela, possivelmente a "null". Podemos portanto concluir que base de dados não relacionais, neste caso o MongoDB, permitem uma maior fluidez na definição do esquema da base de dados, já que não há penalização se se introduzir mais tarde no ciclo de vida da base de dados objetos com atributos novos.

A migração entre os dois sistemas de dados foi relativamente fácil, com o auxílio da ferramenta "**Mongify**". No entanto, se fosse necessário implementar um *script* para a migração de dados, para maior controlo da migração, tal não nos parece ser muito complexo de fazer: o programa usaria um cliente do MySQL para ler os dados e escrevê-los num formato de ficheiros compatível com o MongoDB como JSON ou CSV. Em alternativa, poderia ser feita a escrita dos dados diretamente no MongoDB pelo *script*.

As base de dados não relacionais, neste caso o MongoDB, geralmente não permitem implementar muita lógica na base de dados, remetendo essa responsabilidade para os clientes. Também perdemos a capacidade de ter chaves primárias compostas, já que o MongoDB cria automaticamente um ObjectID *random* para cada documento.

Quanto às *queries* à base de dados, verificamos que existem vantagens e desvantagens. *Queries* simples que apenas necessitamos de visitar informação dentro de uma dada coleção, são muito fáceis de fazer. E de facto o MongoDB sugere de forma geral que os dados estejam juntos, ou seja, que os documentos sejam auto-contidos, com outros documentos embebidos se necessário. Isto faz com que as queries a esses dados sejam fáceis de fazer e também eficientes, pois existe uma localidade dos dados, que melhora a performance da base de dados em processar a *query*. Em contraste, é normal numa base de dados relacional bem normalizada fazer *Joins* de duas ou mais tabelas, o que pode ser uma operação mais custosa.

Por outro lado, nem sempre se recomenda embeber toda a informação dentro dos respetivos documentos, quando é expectável que os dados embebidos cresçam bastante ao longo do tempo, ou quando é expectável que estes dados sejam atualizados mais tarde e haja várias cópias desses documentos embebidos noutros documentos, recomenda-se que se guarde então uma referência para esses objetos e que estes estejam centralizados numa coleção à parte.

Referências

https://docs.mongodb.com/v3.2/tutorial/
https://docs.mongodb.com/v3.2/reference/sql-comparison/
http://www.tutorialspoint.com/mongodb/
http://mongify.com/getting_started.html

Lista de Siglas e Acrónimos

Lista com todas as siglas e acrónimos utilizados durante a realização do trabalho.

BD Base de Dados
SGBD Sistema de Gestão de Base de Dados
BCC *Braga Cinema Center*
DBDL *Database Design Language*
DBMS *Database Management System*
SQL *Structured Query Language*
NoSQL *Not Only Structured Query Language*