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

Unidade Curricular de Bases de Dados

Ano Letivo de 2018/2019

Cinema Casa das Artes

Raquel Dias (a32954),
Ricardo Pereira (a77045),
Ana Guimarães (a79987),
Luís Abreu (a82888)

Janeiro, 2019

BD

Data de Receção	
Responsável	
Avaliação	
Observações	

Cinema Casa das Artes

Raquel Dias (a32954),
Ricardo Pereira (a77045),
Ana Guimarães (a79987),
Luís Abreu (a82888)

Janeiro, 2019

Resumo

O trabalho realizado consistiu na elaboração de um Sistema de Gestão de Base de Dados (SGBD) para um Cinema que sentiu necessidade de implementar um sistema que permitisse gerir melhor a nova estrutura, de dimensão maior que a anterior.

Inicialmente, após análise do contexto, foi efetuado o levantamento e análise de requisitos, para avaliação das necessidades do cliente.

Na posse dos requisitos, passou-se à realização do modelo conceptual, onde foram identificadas e caracterizadas as entidades, os seus atributos e os relacionamentos entre elas. Definiram-se ainda as chaves de cada uma das entidades e efetuou-se uma validação final do modelo de forma a confirmar a resposta aos requisitos solicitados.

Posteriormente procedeu-se à modelação Lógica do mesmo, onde foram definidas as tabelas/relações que constariam da base de dados e os relacionamentos existentes entre elas. Foram então realizadas várias validações através da normalização, interrogações com o utilizador e com as transações estabelecidas, e feitas revisões do modelo lógico com o utilizador e as devidas correções.

A fase seguinte consistiu na conversão do Modelo Lógico em Modelo Físico, recorrendo ao SGBD *MySQL*, o qual utiliza, como interface, a linguagem SQL (*Structured Query Language* ou, traduzido para português, Linguagem de Consulta Estruturada). Nesta fase foram ainda traduzidas as relações e transações com os utilizadores, determinado o espaço de armazenamento necessário em disco e efetuado o povoamento da BD. Por fim foram caracterizados os mecanismos de segurança e revisto o sistema implementado com o utilizador.

No final, foi conseguida uma base de dados bem estruturada, manipulável e segura que vai de encontro às necessidades solicitadas pelo cliente.

Área de Aplicação: Desenho e arquitetura de um Sistema de Bases de Dados para gestão de sessões e venda de bilhetes num cinema.

Palavras-Chave: Cinema, Cliente, Sessão, Filme, Sala, Lugar e Funcionário, Base de Dados, Entidade, Atributo, Relacionamento.

Índice

Anexos

Índice de Figuras

Índice de Tabelas

1. Introdução

1.1. Contextualização

Arcos de Valdevez é um concelho do Norte de Portugal localizado no distrito de Viana do Castelo. Este concelho tem verificado nos últimos anos um grande crescimento turístico.

A Casa das Artes fica no centro histórico deste concelho, sendo uma das grandes atrações do mesmo. Esta é composta por uma biblioteca municipal, que disponibiliza espaço para o estudo ou leitura, um museu com exposição de obras de vários artistas, um auditório destinado à realização de diversos espetáculos (como por exemplo peças de teatro) e uma sala de cinema.

O cinema existente no concelho resumia-se a esta sala existente na Casa das Artes, onde eram realizadas sessões apenas aos sábados e aos domingos. Neste formato, os clientes apenas poderiam comprar o seu bilhete para a sessão no próprio dia, não sendo atribuído nenhum lugar específico a cada cliente. O bilhete era comprado ao funcionário que estava de serviço na receção/entrada do cinema, sendo apenas entregue ao cliente um *ticket* para poder entrar na sala.

Com o aumento da popularidade do cinema e também de audiências, começou a tornar-se difícil para os clientes conseguirem adquirir bilhetes. Tendo de comprar o bilhete no próprio dia, a maior parte dos clientes acabava por adquiri-lo apenas alguns minutos antes do início da sessão, o que cursava com uma longa fila e, por sua vez, atrasos na entrada para a sessão. Aliado a esta situação estava também o facto de apenas haver uma sala com cerca de 160 lugares.

Devido a este crescimento acentuado e à grande adesão ao “Cinema da Casa das Artes”, o responsável pela mesma decidiu efetuar uma reestruturação.

Esta reestruturação do formato do “Cinema Casa das Artes” consistiu na criação de mais salas e na extensão do seu horário de funcionamento, de forma a haver sessões durante toda a semana. Esta reestruturação irá permitir uma maior diversidade de sessões e conteúdos, satisfazendo o aumento da procura, alargado as propostas a um público mais diversificado e melhorando o serviço ao cliente.

1.2. Apresentação do Caso de Estudo

O “Cinema Casa das Artes” é o cinema local da Casa das Artes de Arcos de Valdevez. Após a restruturação, o seu funcionamento passou a ser idêntico ao de cinemas como “Cinema NOS” ou “Cineplace”.

Desta forma, o “Cinema Casa das Artes” tem, para cada uma das suas salas, sessões atribuídas com os respetivos filmes, horários e datas, sendo estas atualizadas semanalmente.

Para a venda ou reserva de bilhetes existe um balcão com um ou mais funcionários. O cliente pode reservar ou comprar o bilhete para uma determinada sessão junto de um destes funcionários, tendo apenas de referir qual a sessão a que pretende assistir. De seguida, o funcionário apresenta a lista de lugares disponíveis para a sessão e o cliente tem a opção de escolher o lugar que deseja, desde que este esteja disponível. Depois de efetuado o pagamento por parte do cliente, este recebe o respetivo bilhete para assistir à sessão pretendida.

Em cada sessão, as salas são abertas 20 minutos antes do início da mesma, para que todos os clientes possam entrar atempadamente e se sentarem nos seus respetivos lugares. Para dar entrada nas sessões os clientes apenas têm que mostrar os seus bilhetes à entrada das salas.

1.3. Motivação

Com este novo formato do “Cinema Casa das Artes” são realizadas várias sessões diariamente, o que resulta num elevado número de reservas de bilhetes, tornando-se assim necessária uma gestão de reservas e informação atualizada sobre os lugares que vão sendo reservados pelos clientes e os que ainda se encontram disponíveis.

Desta restruturação, surge então a necessidade de ser desenvolvida uma Base de Dados (BD) que permita armazenar toda a informação relativa ao funcionamento do cinema, nomeadamente, informação sobre as salas de cinema, sessões que vão ser realizadas, bem como informação sobre os clientes e funcionários. Para além desta, é ainda fundamental armazenar a informação de todas as reservas e compras de bilhetes efetuadas.

Uma outra motivação, de extrema importância, para a realização deste projeto é a necessidade de organizar toda esta informação de forma a permitir uma consulta e análise dos dados mais eficiente.

1.4. Objetivos

O objetivo deste projeto, consiste em implementar um Sistema de Base de Dados (SBD) consistente, que proporcione um ambiente adequado, eficiente e capaz de armazenar toda a informação relativa às salas, aos clientes e às suas reservas.

É de elevada importância que a base de dados desenvolvida seja compreensível, ou seja, que seja constituída por uma estrutura clara que permita uma consulta de dados fácil, flexível e rápida. A solução a implementar deverá ser também segura, garantindo a integridade, disponibilidade e a confidencialidade da informação.

Com a implementação deste SBD será possível pôr em prática todos os objetivos propostos.

1.5. Análise da Viabilidade do Projeto

Para que o SBD cumpra os requisitos identificados nos pontos anteriores, optou-se pela implementação de um Sistema de Base de Dados Relacional (SBDR). A escolha desta resulta no grande número de vantagens que este apresenta, das quais se destacam:

Simplicidade – O SBDR estrutura os dados de forma a evitar a complexidade e organiza intuitivamente os dados em tabelas. Isto permite que os dados sejam organizados naturalmente dentro do modelo, simplificando o desenvolvimento e uso do SBDR.

Facilidade na apresentação dos dados - Um SBDR possibilita a consulta de qualquer tabela e combinação com outras, usando funções de junção para incluir dados contidos noutras tabelas e que possam ser relevantes nos resultados. Permite também apresentar dados filtrados por qualquer requisito (presente quer em linhas ou colunas), mostrando apenas a informação necessária. Além disso, podem-se escolher as colunas a incluir nos resultados, de modo a serem exibidos apenas os dados relevantes.

Integridade dos dados - A integridade dos dados é uma característica essencial do SBDR, sendo garantida pelos tipos e integridade referencial entre tabelas (através das chaves estrangeiras), impedindo que os registos se tornem incompletos e garantindo que os dados sejam fiáveis, precisos e consistentes.

Flexibilidade - O SBDR é extensível e fornece uma estrutura flexível que permite dar resposta a mudanças de requisitos e quantidades crescentes de dados. Permite a fácil implementação de alterações na estrutura de base de dados sem afetar os dados ou a relação entre eles.

Normalização - A normalização garante que o projeto do SBDR esteja livre de anomalias/redundâncias que possam afetar a integridade e a precisão dos dados. A normalização fornece-nos confiança de que o SBDR é robusto e confiável.

1.6. Estrutura do Relatório

Este relatório encontra-se dividido essencialmente em quatro partes que representam fases distintas do projeto.

Na primeira parte (capítulo 2) são expostos todos os detalhes relacionados com a análise de requisitos efetuada, que permitiu ter uma compreensão do problema a abordar mais abrangente e detalhada, com o objetivo de desenvolver o projeto solicitado.

No capítulo 3 encontra-se todo o processo relacionado com a elaboração do modelo conceptual, como a identificação de entidades, atributos e relacionamentos, determinação do domínio dos atributos e das chaves primárias e alternativas, passando ainda pela verificação de redundâncias no modelo.

O capítulo seguinte (capítulo 4) apresenta a modelação lógica e todo o processo utilizado na conceção e validação do mesmo.

Na última parte (capítulo 5) é apresentada a implementação física do modelo, incluindo a tradução de interrogações e transações estabelecidas para SQL, a apresentação de um povoamento inicial para o SBD e as respetivas restrições do sistema.

2. Levantamento e Análise de Requisitos

2.1. Método de levantamento e de análise de requisitos adotado

A análise de requisitos, embora nem sempre fácil de concretizar com precisão, é uma das fases mais importantes para a implementação de um SBD.

Para se obter um bom levantamento de requisitos, e assim ter uma melhor perceção do problema em análise, foram recolhidas, durante vários dias, informações sobre o funcionamento do cinema.

Ao analisar a informação reunida considerou-se, como fundamentais, os requisitos apresentados de seguida, sendo os mesmos divididos em três grupos: requisitos de descrição, de exploração e de controlo.

2.2. Requisitos levantados

2.2.1 Requisitos de descrição

A BD desenvolvida deverá guardar a informação relativa a Clientes, Salas, Sessões, Filmes, Bilhetes e Funcionários.

O principal interveniente do sistema é o Cliente, visto que é este quem faz as reservas e assiste às sessões. O cliente compra os bilhetes junto do funcionário, escolhendo a sessão a que pretende assistir e o lugar que deseja, desde que este esteja disponível. A informação que caracteriza o Cliente é o seu nome, o número de identificação fiscal (NIF) e o tipo de cliente, ou seja, se é ou não estudante.

O Funcionário deve ser identificado por um número de identificação e pelo seu nome. Esta informação é necessária de forma a que fique registado na BD o funcionário que efetuou a emissão de determinado bilhete.

O Bilhete é também uma entidade essencial para o sistema, uma vez que é obrigatória a sua apresentação à entrada para assistir a respetiva sessão. Este é identificado por um

número de identificação e guarda a data em que foi emitido, identifica o lugar que o cliente escolheu no momento da sua reserva, o preço, visto que este varia conforme o tipo de cliente, e a sala em que a respetiva sessão vai ser exibida.

Relativamente às Salas, cada uma delas deve ser identificada através de um número único e guardar os seus lugares, visto que existem salas com diferentes capacidades. Por sua vez, estes lugares são identificados pelo seu id, número da cadeira e a fila.

Cada Sessão é também identificada por um número de identificação, a data em que foi ou será realizada, a hora do início, o seu preço e a sua duração. A cada Sessão deve ainda estar associado um filme que será exibido na mesma. As Sessões estão também associadas à Sala em que vão ser realizadas.

Por fim, em relação ao Filme, deverá ser armazenado o seu número de identificação, o nome, a descrição e a classificação atribuída pelo IMDb.

2.2.2 Requisitos de exploração

Na fase de exploração da base de dados deverá ser possível obter toda a informação necessária ao funcionamento normal do cinema. Assim, deve ser possível obter os seguintes elementos:

- Lista de clientes ordenada pelo número de bilhetes comprados;
- Lista de filmes que se encontram em exibição numa determinada data;
- Lista das sessões existentes para um determinado filme, numa determinada data;
- Lista de lugares disponíveis para uma determinada sessão;
- O número da sala de uma determinada sessão;
- A hora de uma determinada sessão;
- O número de bilhetes que ainda restam para uma sessão;
- Lista de salas disponíveis numa determinada data e intervalo de hora;
- Lista de funcionários que venderam mais bilhetes;
- Nome do funcionário que registou um determinado bilhete;
- Quantos bilhetes um determinado tipo de cliente comprou num determinado intervalo de datas;
- Lista das sessões com mais bilhetes vendidos num determinado intervalo de datas;
- Número de bilhetes que foram vendidos num determinado dia ou intervalo de datas;
- Lista dos 3 filmes em exibição com melhor classificação, ordenados de forma ascendente;
- Lista dos 10 filmes mais vistos no cinema.

2.2.3 Requisitos de controlo

A BD terá um perfil de administrador (*admin)*, para o responsável pela “Casa das Artes” e, por sua vez, pelo cinema. Este vai ter acesso a toda a informação e operações sobre a base de dados. Poderá inserir Salas, Sessões, Filmes, Clientes e Funcionários, colocando a sua informação e fazendo as associações estabelecidas na descrição.

Será também estabelecido um perfil de *funcionário*, sendo que este poderá consultar todas as informações da base de dados, podendo manipular apenas a informação correspondente aos bilhetes e aos clientes.

2.3. Análise geral dos requisitos

O Cliente é um dos principais intervenientes do sistema de base dados em estudo. Este, para poder comprar bilhete, tem de estar registado no sistema, ficando identificado pelo seu número de identificação fiscal (NIF).

O cliente poderá adquirir um ou mais bilhete na bilheteira, perante o funcionário que regista esses bilhetes. Um funcionário pode registar vários bilhetes.

O cliente poderá ainda solicitar algumas informações sobre as sessões disponíveis, horários e filmes.

Cada bilhete está associado a uma sessão, reservando um lugar na sala onde a sessão decorrerá.

A sessão tem um filme e uma sala associados, sendo que o filme poderá ser reproduzido em várias sessões.

Uma sala pode ter inúmeras sessões ao longo do dia, mas nunca em simultâneo, e salas diferentes podem ter sessões à mesma hora.

3. Modelação Conceptual

3.1. Apresentação da abordagem de modelação realizada

Para a elaboração do modelo conceptual foi necessário, numa primeira abordagem, identificar as entidades envolvidas e o relacionamento entre elas, seguida pela associação de atributos às entidades.

De seguida foi determinado o domínio dos atributos e as chaves candidatas, primárias e alternativas.

O passo seguinte consistiu na normalização do modelo, normalização esta que permite verificar a eventual existência de redundâncias e que o mesmo apoia as transações requeridas.

Numa última fase, num contexto real, seria efetuada a revisão do modelo com o utilizador, de forma a garantir que a modelação realizada representasse e manipulasse a informação pretendida para o fim a que efetivamente se destina.

3.2. Identificação e caracterização das entidades

De forma a melhor identificar as diferentes entidades envolvidas no sistema, foi necessário determinar quais os objetos que se enquadram nesta definição. Assim, após a análise de requisitos, conclui-se que as entidades fundamentais presentes neste sistema são: o cliente, o bilhete, o funcionário, a sala, a sessão e o filme.

De seguida é apresentada uma tabela com a descrição de cada uma destas entidades, bem como os seus sinónimos e as suas ocorrências.

Nome da Entidade	Descrição	Sinónimos	Ocorrências
Cliente	Compra/Reserva bilhete para uma determinada sessão, para assistir um determinado filme	Estudante, Não Estudante	Pode reservar e assistir ao número de sessões que pretender.
Bilhete	Comprovativo que a reserva foi concluída.	*Ticket*, Passe, Comprovativo, Ingresso.	É registado pelo funcionário e utilizado pelo cliente na entrada da sala, de modo a ser autorizada a sua entrada para a sessão.
Funcionário	Empregado que atende o cliente e regista os bilhetes.	Empregado, Trabalhador.	Regista as reservas para as sessões, emitindo, para cada uma, o respetivo bilhete.
Sala	Local de exibição das sessões.	Auditório, Recinto, Espaço.	Realiza várias sessões.
Sessão	Intervalo de tempo durante o qual um filme é exibido.	Apresentação, Exibição.	Existem várias sessões por filme, a mesma sessão pode ser reservada por vários clientes.
Filme	Conteúdo a ser exibido na sessão.	Vídeo, obra cinematográfica	Ocorre um filme por sessão, podendo o mesmo filme ser exibido em várias sessões.

Tabela 1 - Entidades e Caracterizações

3.3. Identificação e caracterização dos relacionamentos

Identificadas as entidades que deverão estar presentes no SBD é necessário definir todos os relacionamentos existentes entre as mesmas. A leitura da análise de requisitos permite identificar os relacionamentos que as entidades estabelecem entre si, bem como a sua respetiva cardinalidade.

A tabela apresentada de seguida estabelece todos os relacionamentos entre entidades que foram identificados.

Nome da Entidade	Multiplicidade	Relacionamento	Multiplicidade	Nome da Entidade
Sala	1	Realiza	N	Sessão
Sessão	N	Exibe	1	Filme
Sessão	1	Tem associado	N	Bilhete
Bilhete	N	É registado por	1	Funcionário
Cliente	1	Compra	N	Bilhete

Tabela 2 – Relacionamentos

3.4. Identificação e caracterização das Associações dos Atributos com as Entidades e Relacionamentos

Após identificadas as entidades e os relacionamentos entre elas, segue-se a associação de atributos com entidades e relacionamentos. A análise de requisitos permite identificar possíveis caraterísticas ou propriedades de uma entidade que resultem em atributos da entidade, dependendo da importância ou relevância que estes possam ter na BD.

3.4.1 Atributos simples/compostos

Como já foi referido anteriormente, após a análise dos requisitos, foi possível identificar os atributos relativos a cada entidade. Esta análise permitiu concluir que todos os atributos do sistema definido são atributos simples, à exceção de um, o atributo lugar relativo à entidade sala. Este atributo é composto, uma vez que este é caracterizado pelo seu número de identificação, número da cadeira e fila.

3.4.2 Atributos derivados

Após a análise dos requisitos e identificação dos atributos, conclui-se que não são necessários atributos derivados no sistema.

3.4.3 Atributos multivalor

Efetuada a análise de requisitos, é possível concluir que o atributo lugar, já anteriormente referido como atributo composto, é também um atributo multivalor, uma vez que existem vários lugares para cada sala.

3.4.4 Atributos de relacionamento

Da mesma análise referida nos pontos anteriores, conclui-se que não existem atributos de relacionamento no sistema.

3.4.5 Associação entre Atributos e Entidades

Após a identificação dos atributos torna-se necessário detalhá-los.

De seguida, é apresentada uma tabela que apresenta a associação dos atributos com as suas respetivas entidades, bem como uma breve descrição sobre cada atributo, o seu tipo e tamanho, a sua nulidade, e se é multivalor, derivado ou composto.

Nome da Entidade	Atributos	Descrição	Tipo e Tamanho	Nulo	Multivalor	Derivado	Composto
Cliente	NIF	Número de Identificação Fiscal do cliente	Inteiro positivo	Não	Não	Não	Não
	Nome	Nome do cliente	45 Caracteres Variáveis	Sim	Não	Não	Não
	Tipo	Se é estudante ou não estudante.	1 Caracter (E/N)	Não	Não	Não	Não
Funcionário	idFuncionário	Número que identifica o Funcionário	Inteiro Positivo	Não	Não	Não	Não
	Nome	Nome do Funcionário	45 Caracteres Variáveis	Não	Não	Não	Não
Bilhete	idBilhete	Número que identifica o Bilhete	Inteiro Positivo	Não	Não	Não	Não
	Data_Emissão	Data em que foi emitido o Bilhete	Data	Não	Não	Não	Não
	Lugar	Número do lugar associado ao Bilhete	Inteiro Positivo	Não	Não	Não	Não
	Preço	Preço do Bilhete	Decimal Positivo	Não	Não	Não	Não
Sessão	idSessão	Número que identifica a Sessão	Inteiro Positivo	Não	Não	Não	Não
	Preço_Base	Preço da Sessão	Decimal Positivo	Não	Não	Não	Não
	Data	Data da Sessão	Data	Não	Não	Não	Não
	Hora_Início	Hora de início da Sessão	Hora	Não	Não	Não	Não
	Duração	Duração da Sessão	Tempo	Não	Não	Não	Não

Filme	idFilme		Número que identifica o Filme	Inteiro Positivo	Não	Não	Não	Não
	Nome		Nome do Filme	45 Caracteres Variáveis	Não	Não	Não	Não
	Descrição		Descrição do Filme	Texto	Sim	Não	Não	Não
	Classificação		Classificação do Filme	Decimal Positivo [0...10]	Sim	Não	Não	Não
Sala	Número_Sala		Número que identifica a Sala	Inteiro Positivo	Não	Não	Não	Não
	Lugar				Não	Sim	Não	Sim
	Lugar	idLugar	Número que identifica o Lugar	Inteiro Positivo	Não	Não	Não	Não
		Número_Cadeira	Número da cadeira	Inteiro Positivo	Não	Não	Não	Não
		Fila	Letra que identifica a fila	1 Caracter	Não	Não	Não	Não

Tabela 3 - Associação entre Atributos e Entidades

3.5. Determinação do domínio dos atributos

Seguidamente, são descritos os domínios dos atributos referentes às diferentes entidades. O domínio consiste num conjunto de valores que pertencem a um determinado tipo, e que pode ser atribuído a cada atributo.

Entidade: Sala

- Número_Sala: Número que identifica a sala. É um número inteiro positivo.
- Lugar: Identificação dos lugares da sala, caracterizado por:
    - idLugar: Número inteiro positivo que identifica um lugar;
    - Número_Cadeira: Número inteiro positivo que identifica o número da cadeira;
    - Fila: Representado por um único caracter entre A-Z que identifica a fila.

Entidade: Sessão

- idSessão: Número que identifica a sessão. É um número inteiro positivo.
- Preço_Base: Preço da sessão, representado por um decimal positivo;
- Data: Data em que se realiza a sessão. Corresponde ao ano, mês e dia da sessão;
- Hora_Início: Hora de início da sessão. Corresponde à hora, minuto e segundo marcados para o início da sessão.
- Duração: Duração da sessão. Corresponde aos minutos que a sessão vai durar.

Entidade: Filme

- idFilme: Número que identifica o filme. É um número inteiro positivo;
- Nome: Nome do filme. É uma string com 45 caracteres variáveis;
- Descrição: Texto correspondente à descrição do filme;
- Classificação: Classificação do filme representada por um decimal positivo entre 0.0 e 10.0.

Entidade: Bilhete

- idBilhete: Número que identifica o bilhete. É um número inteiro positivo;
- Data_Emissão: Data em que é emitido o bilhete, representado pelo ano, mês e dia;
- Lugar: Número que identifica o lugar associado ao bilhete reservado. É representado por um número inteiro positivo;
- Preço: Preço do bilhete, de acordo com o tipo de cliente que o adquire, representado por um decimal positivo;

Entidade: Funcionário

- idFuncionário: Número que identifica o funcionário. É um número inteiro positivo;
- Nome: Nome do funcionário. É uma string com 45 caracteres variáveis;

Entidade: Cliente

- NIF: Número de identificação fiscal do cliente, identifica o cliente. É um número inteiro positivo;
- Nome: Nome do cliente. É uma string com 45 caracteres variáveis;
- Tipo: Representado por um único caracter, E (estudante) ou N (não estudante), identifica o tipo de cliente.

3.6. Determinação de chaves primárias, candidatas e alternativas

A identificação de chaves primárias é um processo bastante importante, uma vez que estas permitem identificar exclusivamente cada ocorrência das entidades, garantindo a integridade dos dados.

A primeira fase na determinação de chaves primárias é definir aquelas que poderão ser consideradas como chaves candidatas. Estas devem permitir identificar de forma unívoca qualquer ocorrência dessa entidade. Das chaves candidatas disponíveis, é escolhida aquela que melhor se adequar para chave primária.

Para a entidade cliente o único atributo que se poderá candidatar a chave primária é o "NIF", pois trata-se de um identificador numérico único que garante que a chave não se repete para diferentes ocorrências da entidade. Todos os restantes atributos seriam uma má escolha, uma vez que que o "Nome" do cliente e o "Tipo" de cliente poderão ter valores iguais para diferentes clientes, logo não satisfazem os requisitos para se tornarem chaves candidatas.

Para a entidade Funcionário selecionou-se o atributo "idFuncionário" como chave primária da entidade, por ser um atributo numérico bastante simples e que identifica unicamente cada ocorrência desta entidade. O nome do funcionário, à semelhança do que acontece com a entidade Cliente, não pode ser considerado como chave candidata, uma vez que não cumpre os requisitos necessários.

Para a entidade Bilhete, o atributo que melhor se encaixa no perfil de chave primária é o "idBilhete". O atributo Lugar poderia também, à partida, ser um atributo considerado apto a tornar-se uma chave candidata, uma vez que se poderia pensar que o número do lugar não se repete. No entanto, para diferentes sessões podem existir números repetidos do número de lugar, já que as mesmas salas com os mesmos lugares efetuam sessões diferentes. Por esse motivo e porque os restantes atributos também não satisfazem os requisitos para se tornarem chaves candidatas, determinou-se que o "idBilhete" seria a chave primária desta entidade.

Relativamente à entidade Sessão, o atributo mais adequado para se tornar a chave primária é o "idSessão" uma vez que é um atributo bastante simples e que, por identificar univocamente cada sessão, cumpre os requisitos necessários. Os atributos "Data" e "Hora_Início" não poderão sequer ser consideradas chaves candidatas, visto que pode existir mais do que uma sessão com a mesma data e hora de início, o que, não identifica unicamente cada sessão.

Para a entidade Sala, o atributo que foi identificado como chave primária foi o "Número_Sala", uma vez que cada sala tem um número associado que a identifica inequivocamente. Como já referido anteriormente neste relatório, a entidade Sala contém um atributo composto e multivalor, e que, também este precisa ser identificado por uma chave primária. Ao analisar os atributos do lugar que poderiam ser candidatos a chave primária, ponderou-se a possibilidade do "Número_Cadeira" poder ser candidato a chave primária, no entanto, existem várias cadeiras com o mesmo número, para diferentes filas. Assim, foi

considerado que o “idLugar”, que corresponde a um número inteiro positivo, é o atributo que melhor identifica cada lugar, garantindo a sua unicidade.

Para a entidade Filme, o atributo que melhor satisfaz o requisito de chave primária é o “idFilme” por ser um atributo numérico bastante simples, que identifica unicamente cada ocorrência da entidade Filme. O “nome do filme”, tal como acontece no caso do Cliente e do Funcionário, também não pode ser considerado como chave primária, pelas razões já atrás identificadas.

3.7. Verificação de redundâncias no modelo

Definidas as chaves para cada entidade, passou-se para a verificação de ocorrência de redundâncias no modelo que, sendo identificadas, deverão ser removidas.

Para esta verificação foram seguidos os seguintes passos:

1º - Voltar a examinar todos os relacionamentos 1:1;

2º - Remover relacionamentos com redundância;

3º - Considerar a dimensão do tempo nos relacionamentos.

O primeiro passo tem como objetivo proteger o modelo dos casos em que se identificam duas entidades que representem o mesmo objeto, ou seja, quando existem duas entidades que são sinónimos. Neste caso estas devem ser convertidas numa só entidade. Após rever o modelo conceptual efetuado verificou-se que este não apresenta nenhum relacionamento 1:1, logo não existe redundância a este nível.

O segundo passo, consiste em verificar se a mesma informação pode ser obtida por mais do que um relacionamento, havendo assim redundância. Depois de analisado o modelo conceptual verificou-se que não existem redundâncias nos relacionamentos.

O último passo consiste em verificar se o relacionamento entre duas entidades pode variar ao longo do tem, podendo comprometer a redundância do modelo em função do tempo. Analisando o modelo conceptual definido não foram detetadas situações que possam comprometer a redundância do modelo em função do tempo.

3.8. Detalhe ou generalização de entidades

Após a revisão de todo o modelo conceptual, e com a verificação do modelo para a ocorrência de redundâncias da secção anterior, conclui-se que não é necessário generalizar nem detalhar nenhuma entidade.

3.9. Apresentação e explicação do diagrama ER

Com o objetivo de representar conceptualmente as entidades e os relacionamentos entre elas, na base de dados, foi elaborado um diagrama ER, apresentado abaixo na figura 1.

Nas secções anteriores já foram sendo explicadas as considerações efetuadas que convergiram na obtenção deste diagrama. Desta forma, resta apenas fazer um resumo visual do modelo conceptual, de modo a resumir e apreender toda a informação.

No modelo final apresentado, pode-se observar o relacionamento entre Sala e Sessão, Filme e Sessão, Sessão e Bilhete, Cliente e Bilhete, e, por fim, Funcionário e Bilhete. Todos os relacionamentos são 1:N, indicando que:

- Uma Sala realiza várias Sessões;
- Um Filme é exibido em várias Sessões;
- Uma Sessão tem associados vários Bilhetes;
- Um Cliente compra vários Bilhetes;
- Um Funcionário regista vários Bilhetes.

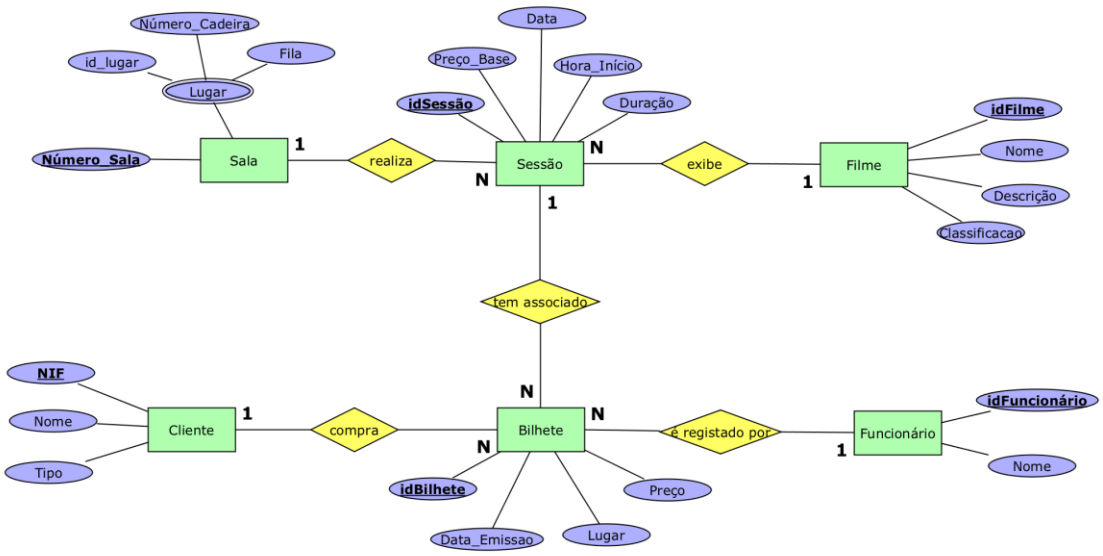

Figura 1 - Diagrama ER

3.10. Validação do modelo de dados com o utilizador

Nesta fase do projeto, já se garante um modelo conceptual que representa a informação requerida pelo utilizador. Agora torna-se necessário verificar o modelo relativamente às transações e requisitos do utilizador, ou seja, verificar se o modelo atual consegue responder às interrogações exigidas pelo utilizador. Em caso negativo, significa que houve algum erro na modelação, pelo que terá de se reanalisar alguns dos passos da modelação.

Da análise de requisitos, foram retiradas interrogações que foram implementadas. Abaixo, seguem-se estas, junto com uma explicação que garante a respetiva resposta às interrogações.

- Quais os clientes que compraram mais bilhetes?

A entidade Cliente está associada á entidade Bilhete, estando em cada bilhete identificado o cliente pelo seu NIF, logo, acedendo aos bilhetes, é possível apresentar uma lista ordenada de clientes, por ordem decrescente de número de bilhetes comprados, contando o número de ocorrências de cada cliente nos bilhetes.

- Quais são os filmes que se encontram em exibição numa determinada data?

A entidade filme está associada à entidade sessão, sendo que esta última contém como atributo a data da mesma. Acedendo à lista de sessões e selecionando uma determinada data, é possível obter os filmes em exibição na data solicitada, executando com sucesso a interrogação.

- Que sessões existem para um determinado filme, numa determinada data?

Acedendo à lista de sessões, é possível selecionar um filme numa determinada data, uma vez que a entidade filme se encontra relacionada com a entidade sessão, obtendo-se assim a lista de todas as sessões existentes para esse filme. Assim, esta interrogação é respondida com sucesso.

- Quais são os lugares disponíveis para uma sessão?

A sessão herda da entidade sala o atributo número de sala como chave estrangeira. Através desta consegue-se saber qual a sala em que determinada sessão será realizada.

Conhecendo o id da Sessão é também possível determinar os bilhetes que já foram vendidos para essa sessão, aos quais estão associados os respetivos lugares já vendidos.

Conhecendo a sala e os lugares vendidos (atributo de bilhete), excluímos da lista de lugares da sala os lugares vendidos e obtemos, com sucesso à interrogação efetuada, as listas dos lugares por vender.

- Qual o número da sala de uma determinada sessão?

A sessão herda do atributo sala o id desta como chave estrangeira. Através deste consegue-se saber qual o número da sala.

- Qual a hora de uma determinada sessão?

Consultando a relação sessão, e uma vez que a hora de início é um atributo da mesma, é possível saber qual a hora de determinada sessão.

- Quantos bilhetes ainda restam para uma sessão?

Sabendo quais os lugares da sala que ainda estão disponíveis bastam contar esses lugares e sabemos quantos bilhetes ainda restam.

- Quais foram os funcionários que mais bilhetes venderam?

O bilhete herda do funcionário o atributo idFuncionário como chave estrangeira. Através desta conseguimos saber quantos bilhetes cada funcionário vendeu, e organizá-los numa lista ordenada por número de bilhetes que cada funcionário vendeu, de forma decrescente.

- Quem foi o funcionário que registou um determinado bilhete?

O bilhete herda do funcionário o atributo idFuncionário como chave estrangeira. Através desta conseguimos determinar quem foi o funcionário que emitiu/registou determinado bilhete.

- Quantos bilhetes um determinado tipo de cliente comprou num determinado intervalo de datas?

Acedendo à relação bilhete, e uma vez que este herda como chave estrangeira o NIF do cliente, é possível identificar os bilhetes que foram vendidos a um determinado tipo de cliente. A partir daqui é possível selecionar os bilhetes num determinado intervalo de datas. Fazendo a contagem dos mesmos obtemos o número de bilhetes vendidos.

Esta é uma interrogação que poderá ser interessante para efeitos estatísticos com o objetivo de otimizar o planeamento das sessões, indo de encontro às preferências dos utilizadores.

- Quais as sessões com mais bilhetes vendidos num determinado intervalo de datas?

Esta é uma questão bastante importante para efeitos estatísticos, visto que com esta informação é possível otimizar o planeamento das sessões. Conhecendo as sessões mais frequentadas é possível otimizar os horários das sessões de forma a que a maior oferta ocorra nos horários de maior procura.

Uma vez que os bilhetes herdam como chave estrageira o id da Sessão é possível determinar quantos bilhetes foram vendidos para uma determinada sessão. Ordenando os resultados por ordem decrescente, é possível perceber quais as sessões para as quais se venderam mais.

- Número de bilhetes que foram vendidos num determinado dia ou intervalo de datas?

A entidade Bilhete contém um atributo que guarda a respetiva data em que foi emitido. Selecionando uma determinada data ou intervalo de datas, faz-se a contagem de bilhetes emitidos na mesma.

- Quais são os 3 filmes em exibição com melhor classificação?

A entidade filme tem a classificação do mesmo como atributo. Assim é possível responder com sucesso a esta interrogação, selecionando da lista de filmes apenas os 3 com classificação mais alta, ordenando-os por ordem ascendente.

- Quais foram os 10 filmes mais vistos no cinema?

Conforme já foi verificado numa interrogação anterior, é possível saber, quantos bilhetes foram vendidos para cada uma delas. Uma vez que cada sessão está relacionada com um filme é possível saber quais os filmes que foram vistos em cada sessão e determinar qual os que forma mais vistos, dando resposta a esta interrogação.

Após esta análise conclui-se que o modelo apresentado responde a todas as interrogações necessárias faces aos requisitos necessários para o utilizador.

4. Modelação Lógica

4.1. Construção e validação do modelo de dados lógico

Nesta fase será apresentado o modelo lógico obtido com base no modelo conceptual anteriormente estabelecido. Para tal, derivam-se os relacionamentos para o modelo lógico criando tabelas ou relações a fim de se representar as entidades, os atributos e os relacionamentos que foram identificados.

4.1.1 Entidades Fortes

Na modelação conceptual realizada estão presentes seis entidades: Sala, Sessão, Filme, Bilhete, Cliente e Funcionário. Por serem independentes umas das outras, são também consideradas entidades fortes. Assim, para cada uma delas, será criada uma tabela, no modelo lógico, com os seus atributos.

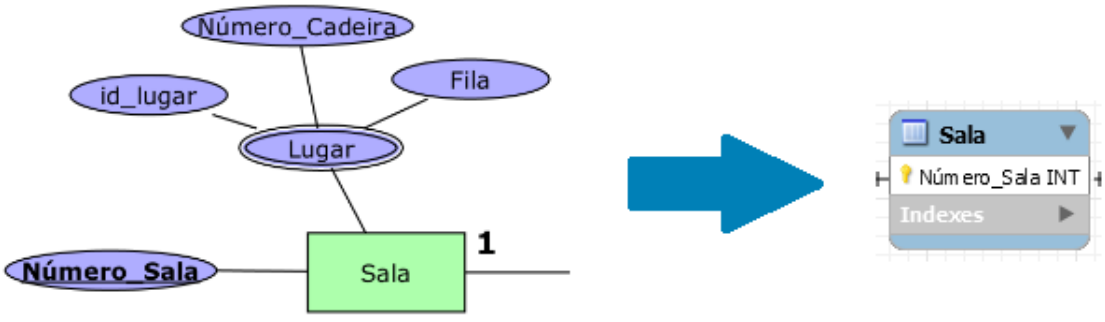

Figura 2 - Conversão do Modelo Conceptual para Lógico (Entidade Sala)

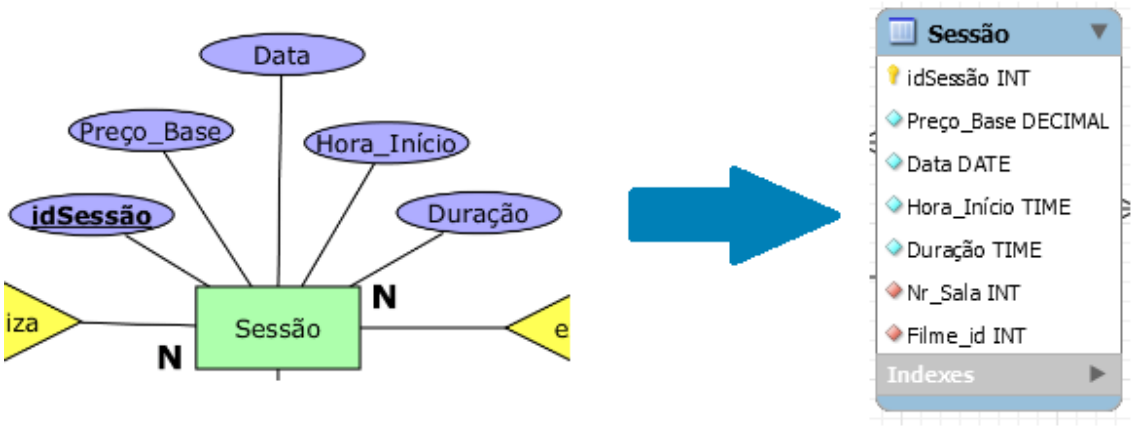

Figura 3 - Conversão do Modelo Conceptual para Lógico (Entidade Sessão)

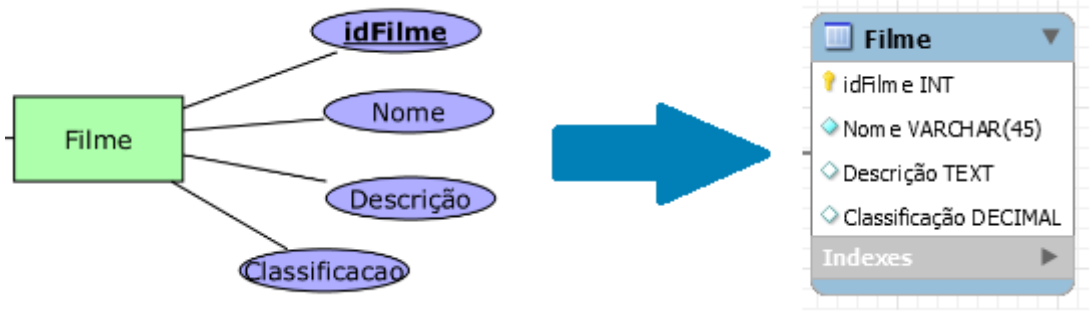

Figura 4 - Conversão do Modelo Conceptual para Lógico (Entidade Filme)

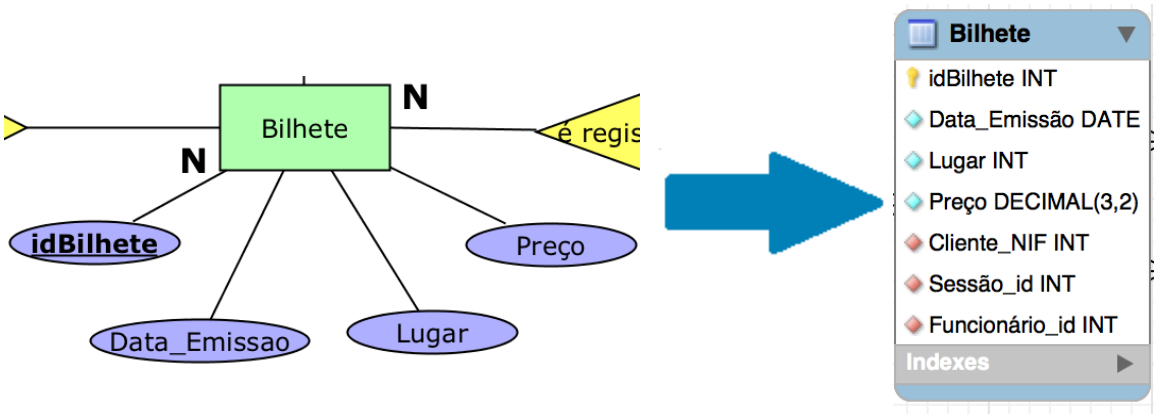

Figura 5 - Conversão do Modelo Conceptual para Lógico (Entidade Bilhete)

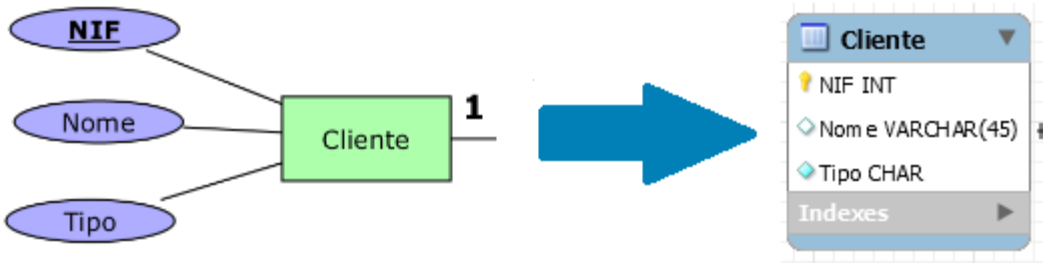

Figura 6 - Conversão do Modelo Conceptual para Lógico (Entidade Cliente)

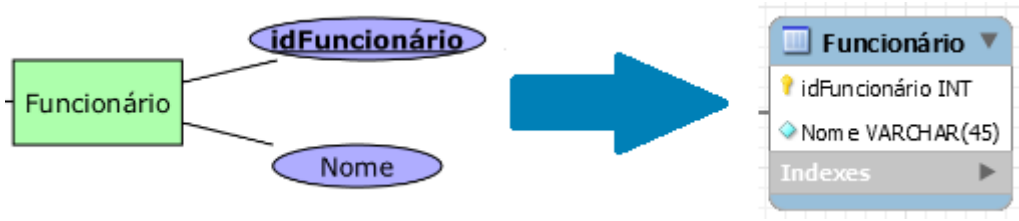

Figura 7- Conversão do Modelo Conceptual para Lógico (Entidade Funcionário)

4.1.2 Entidades Fracas

No modelo conceptual apresentado não há nenhuma ocorrência de entidades fracas.

4.1.3 Relacionamentos 1:N

No modelo conceptual apresentado todos os relacionamentos são de 1:N. Para cada um destes relacionamentos, a entidade que se encontra do lado de maior cardinalidade (N) vai possuir como chave estrangeira uma referência à entidade com menor cardinalidade (1).

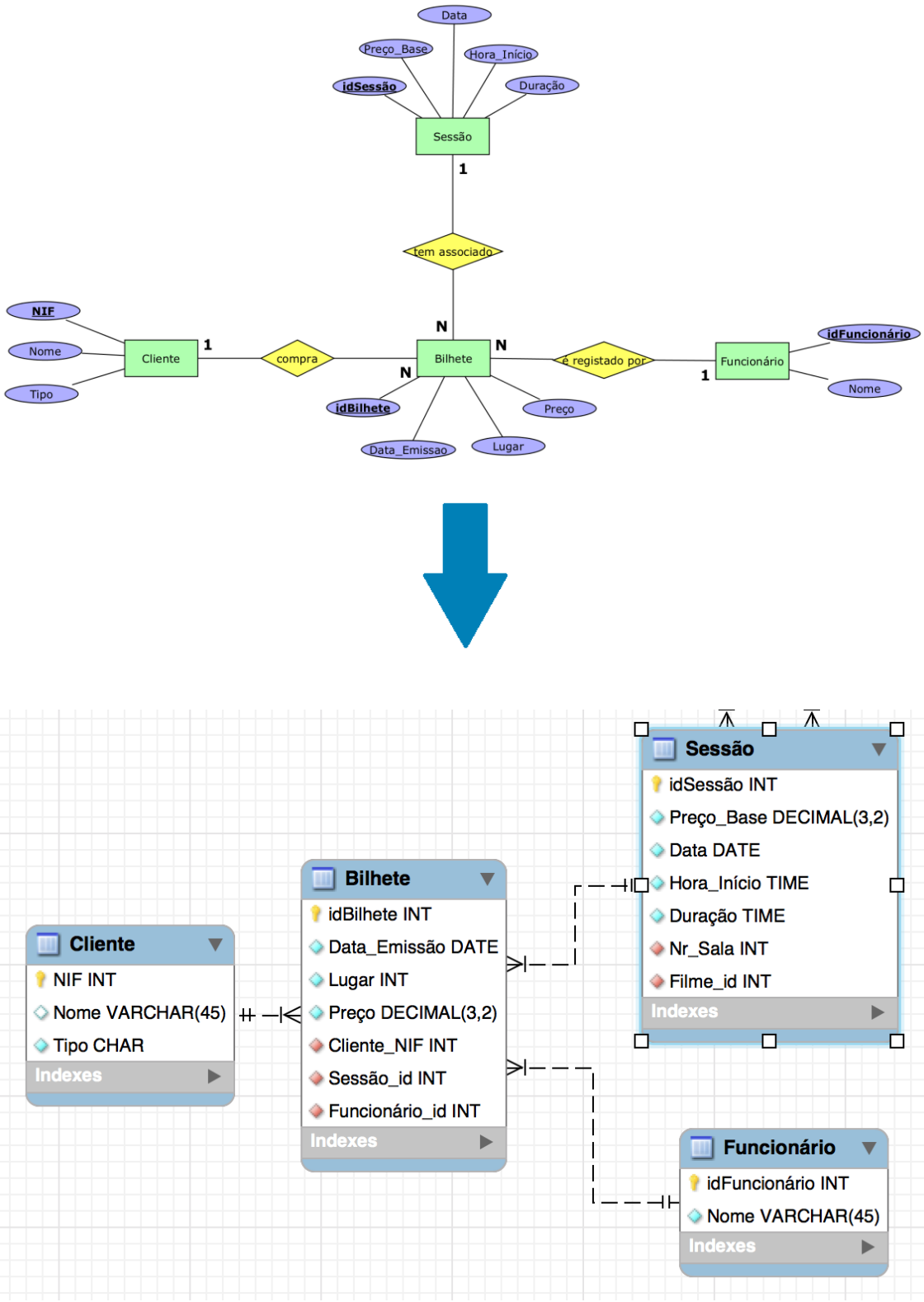

Figura 8 - Conversão do Modelo Conceptual para Lógico (relacionamentos 1:N com a entidade Bilhete)

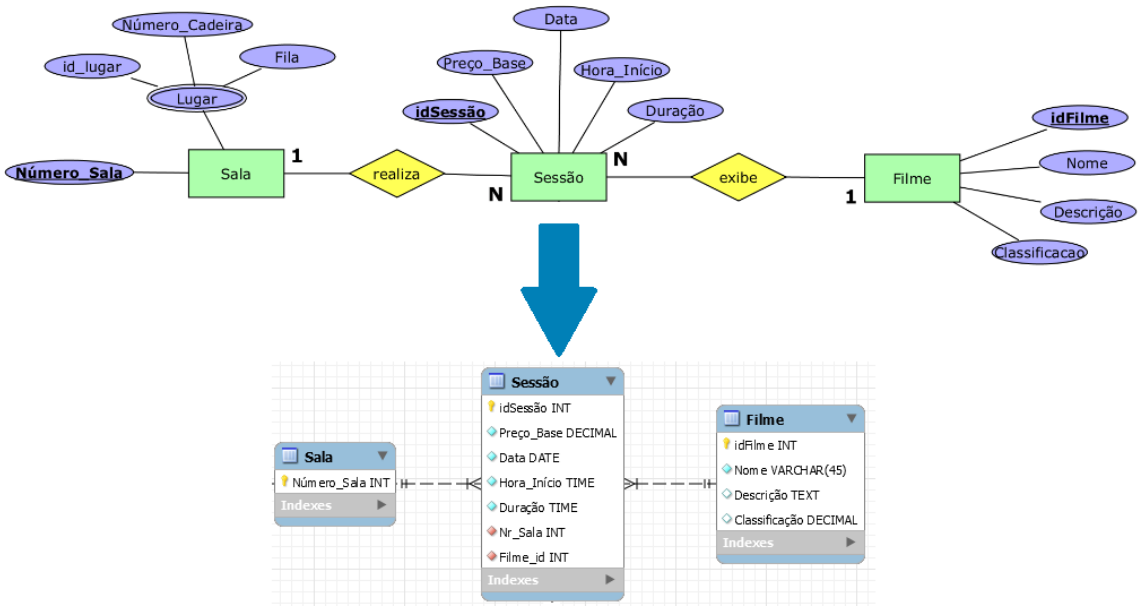

Figura 9 - Conversão do Modelo Conceptual para Lógico (relacionamentos 1:N com a entidade Sessão)

4.1.4 Relacionamentos 1:1

No modelo conceptual apresentado não existem relacionamentos 1:1.

4.1.5 Relacionamentos recursivos 1:1

No nosso modelo conceptual não existem relacionamentos recursivos 1:1.

4.1.6 Relacionamentos superclasse/subclasse

No presente modelo conceptual não existem relacionamentos superclasse/subclasse.

4.1.7 Relacionamentos N:M

No modelo conceptual não existem relacionamento N:M.

4.1.8 Relacionamentos Complexos

Neste modelo conceptual não existem relacionamentos complexos.

4.1.9 Atributos multivalor

Neste modelo conceptual existe um atributo multivalor, o lugar. Como tal, para este atributo é criada uma tabela ou relação. Nesta tabela, cujo nome é o mesmo do atributo, é colocada uma cópia da chave primária da entidade à qual pertence o atributo, que também faz parte da chave primária.

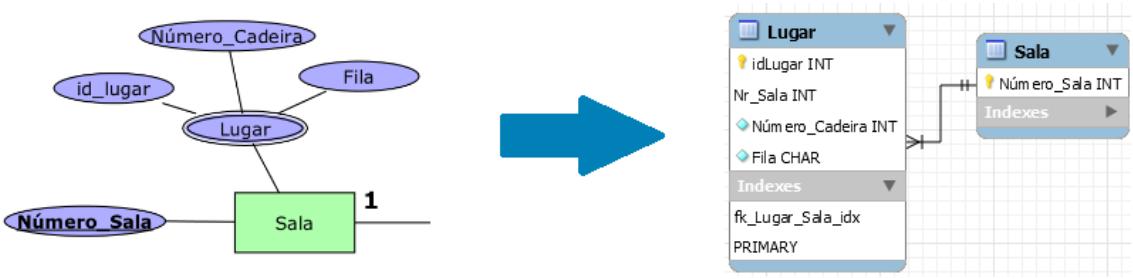

Figura 10 - Conversão do Modelo Conceptual para o Modelo Lógico (atributo multivalor Lugar)

4.2. Desenho do modelo lógico

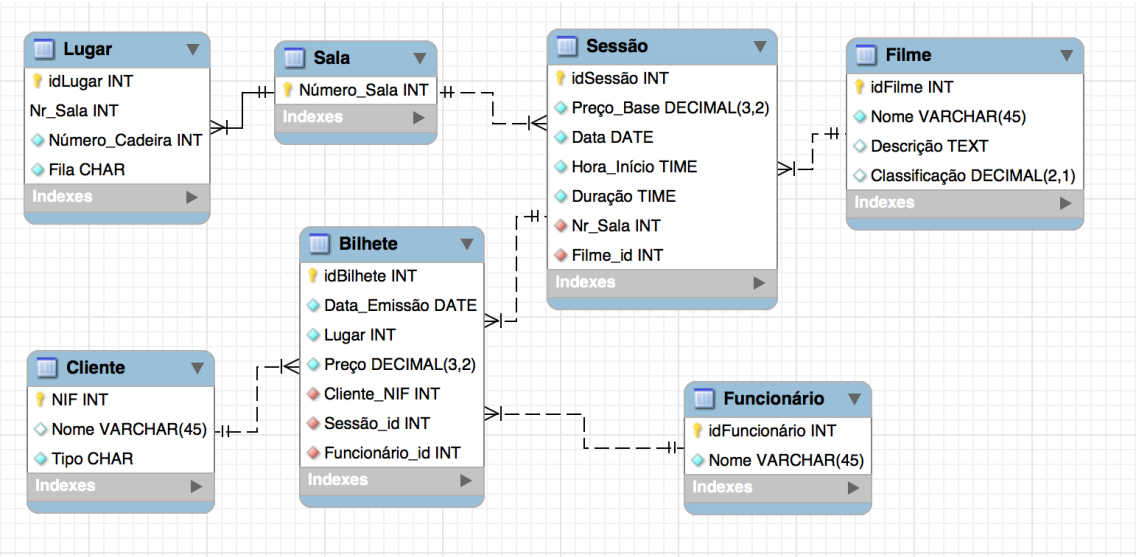

Figura 11 – Modelo Lógico

4.3. Validação do modelo através da normalização

Nesta fase será efetuada a normalização dos dados. A normalização da base de dados consiste num conjunto de regras que tem como principal objetivo reduzir redundâncias, aumentando o desempenho e a integridade dos dados.

Para efetuar a normalização deve-se examinar os atributos de uma entidade e as relações entre entidades, evitando anomalias que possam surgir na inclusão, exclusão e alteração de registos.

Para a normalização existem 5 regras designadas de formas normais. No entanto, a partir da 3° forma normal diz-se que o base de dados já se encontra normalizada. Neste projeto realizou-se a normalização dos dados até à terceira forma normal, inclusive.

Primeira Forma Normal (1FN)

Uma relação está na 1ª Forma Normal (1NF) se todos os atributos forem atómicos, ou seja, se cada linha contém exatamente um valor para cada atributo.

Analisando o modelo lógico construído pode-se concluir que todas as tabelas se encontram na 1FN, uma vez que não existem entidades com atributos repetidos e, o único atributo multivalorado que existia no modelo conceptual, deu origem a uma nova tabela.

Segunda Forma Normal (2FN)

Para que uma relação esteja na 2ª Forma Normal ela tem de estar na 1ª Forma Normal, o que já foi verificado, e todos os atributos descritores dependerem da totalidade da chave da tabela, e não apenas de parte dela (Dependências Funcionais Parciais).

Todas as tabelas com um só atributo como chave primária estão, automaticamente, na 2FN (desde que a 1FN também já tenha sido verificada). Assim só é necessária a análise para as tabelas com chaves primárias compostas.

A única tabela no modelo que contém uma chave composta é a tabela relativa ao Lugar. No entanto esta não apresenta erro de modelação, visto que um determinado lugar tem uma dependência funcional com ambos os atributos que compõe a chave, nomeadamente o idLugar e NrSala. Por exemplo, se o lugar “Fila=A”, “Número_Cadeira=1” corresponder ao “idLugar=1” e, admitindo que este lugar vai existir em todas as salas do cinema, implicaria uma violação da chave primária, visto que esta passaria a estar duplicada, o que não é permitido. Assim, neste caso em especial torna-se totalmente necessária a utilização de uma chave composta.

Após esta análise pode-se concluir que o modelo lógico, se encontra na 2FN.

Terceira Forma Normal (3FN)

Para que uma relação esteja na 3ª Forma Normal ela tem de estar na 2ª Forma Normal e não podem existir atributos descritores a dependerem funcionalmente de outros atributos descritores, as chamadas Dependências Transitivas. Assim, cada atributo deve depender apenas da chave da relação.

Nesta fase as tabelas do modelo proposto já se encontram validadas ao nível da 2FN, sendo agora necessário então verificar se a aplicação da 3FN também se verifica.

Desde já se pode afirmar que a tabela Funcionário está na 3FN, visto que tem apenas um atributo descritor.

Em relação à tabela Bilhete, ponderou-se a possibilidade de associar a sala como atributo simples. No entanto acabou por se verificar que este era funcionalmente dependente da sessão. De forma a que esta tabela certificasse a 3ª Forma Normal, optou-se por não incluir este atributo na entidade bilhete.

Em relação às restantes tabelas, não foi detetada mais nenhuma dependência transitiva, pelo que se pode afirmar que as tabelas do modelo apresentado se encontram na 3FN.

Após esta análise, e efetuadas as retificações necessárias, pode-se concluir que todas as tabelas do modelo lógico se encontram agora normalizados até à terceira forma normal da normalização.

4.4. Validação do modelo com interrogações do utilizador

Nesta fase procedeu-se à validação do modelo lógico, verificando as interrogações do utilizador que podem ser satisfeitas com base neste.

- Lista de clientes ordenada pelo número de bilhetes comprados:

$\tau_{COUNT(Cliente\_NIF)\ DESC}\ (\gamma_{NIF}\ (\pi_{\ NIF,Nome,COUNT(Cliente\_NIF)}((Cliente) \bowtie_{NIF=Cliente\_NIF} (Bilhete))))$

- Filmes que se encontram em exibição numa determinada data:

$\tau_{HoraInicio\ ASC}\ (\pi_{idFilme,Nome,Data,HoraInicio,Nr\_Sala}\ (\sigma_{Data='xdata'}((Sessão) \bowtie_{idFilme=Filme\_id} (Filme))))$

- Sessões que existem para um determinado filme, numa determinada data:

$\pi_{idSessão,Hora\_Início,Nome}(\sigma_{DATA='Xdata'\ AND\ Nome='XFilme'}((Sessão) \bowtie_{\ idFilme=Filme\_id}\ (Filme)))$

- Lugares disponíveis para uma sessão:

$(\pi_{idLugar,Numero\_Cadeira,Fila}(\sigma_{idSessao='xsessao'}((Lugar)\bowtie_{Numero\_Sala=Nr\_Sala}(Sala)$

$\bowtie_{Numero\_Sala=Nr\_Sala}(Sessao)))) -$

$(\pi_{idLugar,Numero\_Cadeira,Fila}(\sigma_{idSessao='xsessao'}((Bilhete)\bowtie_{idSessao=Sessao\_id}(Sessao)\bowtie_{Numero\_Sala=Nr\_Sala}(Sala)$

$\bowtie_{Numero\_Sala=Nr\_Sala}(Lugar)))$

- Número da sala de uma determinada sessão:

$\pi_{Número\_Sala}(\sigma_{idSessão='XSessão'}((Sessao)\bowtie_{Numero\_Sala=Nr\_Sala}(Sala)))$

- Hora de uma determinada sessão:

$\pi_{hora\_início}(\sigma_{idSessão='XSessão'}(Sessao))$

- Preço de um bilhete para um determinado tipo de cliente:

$\gamma_{Preco}(\pi_{Preco}(\sigma_{Tipo='xtipo'}((Cliente)\bowtie_{NIF=Cliente\_NIF}(Bilhete))))$

- Número de bilhetes que ainda restam para uma sessão:

$\pi_{COUNT(idLugar)}((\pi_{idLugar,Numero\_Cadeira,Fila}(\sigma_{idSessao='xsessao'}((Lugar)\bowtie_{Numero\_Sala=Nr\_Sala}(Sala)$

$\bowtie_{Numero\_Sala=Nr\_Sala}(Sessao)))) -$

$(\pi_{idLugar,Numero\_Cadeira,Fila}(\sigma_{idSessao='xsessao'}((Bilhete)\bowtie_{idSessao=Sessao\_id}(Sessao)\bowtie_{Numero\_Sala=Nr\_Sala}(Sala)$

$\bowtie_{Numero\_Sala=Nr\_Sala}(Lugar))))$

- Lista de funcionários que venderam mais bilhetes:

$\tau_{COUNT(Funcionario\_id)DESC}$

$(\gamma_{Funcionario\_id}(\pi_{Nome,COUNT(Funcionario\_id)}((Bilhete)\bowtie_{idFuncionario=Funcionario\_id}(Sessao))))$

- Funcionário que registou um determinado bilhete:

$\pi_{idFuncionário,Nome}(\sigma_{idBilhete='XBilhete'}((Bilhete)\bowtie_{Funcionário\_id=idFuncionário}(Funcionário)))$

- Número de bilhetes que um determinado tipo de cliente comprou num determinado intervalo de datas:

$\pi_{COUNT(idBilhete)}(\sigma_{Tipo='xtipo'\ AND(Data\_Emissao\ Between\ 'xdata1'\ AND\ 'xdata2')}((Bilhete)\bowtie_{NIF=Cliente\_NIF}(Cliente)))$

- Sessões com mais bilhetes vendidos num determinado intervalo de datas ou numa data especifica:

$\tau_{COUNT(Sessao\_id)DESC}(\gamma_{Sessao\_id}\ \pi(_{idSessao,Nome,Data,Hora\_inicio,Count(Sessao\_id)}(\sigma_{Between\ 'xdata1'\ AND\ 'xdata2'}((Bilhete)$

$\bowtie_{idsessao=Sessao\_id}(Sessao)))))$

- Número de bilhetes que foram vendidos num determinado dia ou intervalo de datas:

$\pi_{COUNT(idBilhete)}(\sigma_{Data\_Emissao\ BETWEEN\ 'xdata1'\ AND\ 'xdata2'}(Bilhete))$

- Os 3 filmes em exibição com melhor classificação, ordenados de forma ascendente:

$\tau_{Classificacao\ ASC,\ LIMIT\ 3}\ (\pi_{Nome,Classificacao}\ (Filme))$

- Quais foram os 10 filmes mais vistos no cinema?

$\tau_{COUNT(idFilme)DESC,\ LIMIT\ 10}\ (\gamma_{idFilme}(\pi_{idFilme,Nome,COUNT(idFilme)}\ ((Bilhete)\ \bowtie_{idSessao=Sessao\_id}\ (Sessao)\ \bowtie_{idFilme=Filme\_id}(Filme))))$

4.5. Validação do modelo com as transações estabelecidas

Neste projeto, e dados os requisitos necessários, decidiu-se implementar cinco transações que são necessárias e essenciais na utilização do modelo proposto, e que são devidamente suportadas pelo modelo lógico proposto:

- Inserir um novo cliente: insere-se um novo cliente na tabela Cliente introduzindo o seu NIF, Nome e Tipo de Cliente;

- Inserir um novo funcionário: insere-se um novo funcionário na tabela Funcionário introduzindo o seu Nome;

- Registar um bilhete: regista-se um novo bilhete na tabela Bilhete introduzindo o NIF do Cliente, a Data de Emissão, o Lugar e a Sessão;

- Inserir um novo filme: insere-se um novo filme na tabela Filme introduzindo o seu Nome, descrição e classificação;

- Inserir uma nova sessão: insere-se uma nova sessão na tabela Sessão introduzindo o Filme, a Sala, a Data, a Hora da Inicio, a Duração e Preço.

4.6. Reavaliação do modelo lógico

Após toda esta análise, não se mostrou necessário efetuar a reavaliação do modelo lógico.

4.7. Revisão do modelo lógico com o utilizador

Para finalizar a fase de modelação do modelo lógico realizou-se a verificação do modelo com o utilizador e conclui-se que o problema em questão se encontra bem definido no modelo lógico elaborado, indo de encontro aos requisitos especificados.

5. Implementação Física

5.1. Seleção do sistema de gestão de bases de dados

O SGBD usado na implementação física da base de dados foi o MySQL. As razões para a utilização deste SGBD são diversas, como por exemplo:

- É um SGBD gratuito;
- Possui código fonte aberto;
- É fácil de manipular;
- Tem um alto desempenho;
- É multiplataforma;
- Permite a implementação de regras de segurança no servidor.

5.2. Tradução do esquema lógico para o sistema de gestão de bases de dados escolhido em SQL

Para efetuar a tradução do esquema lógico para o SGBD, foi utilizada a funcionalidade "Forward Engineer" disponibilizada pelo MySQL Workbench e que deu origem à Script de criação da BD que pode ser analisada no anexo 1.

5.3. Tradução das interrogações do utilizador para SQL

```
-- Lista de Clientes Ordenada pelo numero de bilhetes comprados
SELECT C.Nome, COUNT(B.idBilhete) FROM Cliente AS C
    INNER JOIN Bilhete AS B ON C.NIF = B.Cliente_NIF
    GROUP BY C.NIF
    ORDER BY COUNT(B.idBilhete) DESC;
```

Figura 12 – *MySQL Query* – Lista de Clientes

```
-- Lista de filmes em exibição numa determinada data
SELECT F.idFilme, F.Nome, S.Data, S.Hora_Início, S.Nr_Sala FROM Filme AS F
    INNER JOIN Sessão AS S ON F.idFilme = S.Filme_id
    WHERE S.Data = '20181121'
    ORDER BY S.Hora_Início;
```

Figura 13 – *MySQL Query* – Filmes em exibição

```
-- Lista de sessões de um determinado filme numa determinada data
DROP PROCEDURE IF EXISTS sessoesPorFilme;
DELIMITER &&
CREATE PROCEDURE sessoesPorFilme (IN nome VARCHAR(45),
                                  IN data DATE)
BEGIN
    DECLARE erro BOOL DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro=1;
    START TRANSACTION;

    SELECT S.idSessão, S.Hora_Início, F.Nome FROM Sessão AS S
        INNER JOIN Filme AS F ON S.Filme_id = F.idFilme
        WHERE F.Nome = nome AND S.Data = data;

    IF erro
        THEN ROLLBACK;
        ELSE COMMIT;
    END IF;
END &&
DELIMITER &&
```

Figura 14 – *MySQL Query* – Lista de sessões para um filme

```
-- Lista de lugares disponíveis para uma determinada sessão
DROP PROCEDURE IF EXISTS lista_lugares_disponiveis;
DELIMITER &&
CREATE PROCEDURE lista_lugares_disponiveis(IN id INT)
BEGIN
    DECLARE erro BOOL DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro=1;
    START TRANSACTION;

    SELECT ls.idLugar, ls.Número_Cadeira, ls.Fila FROM
        (SELECT l.idLugar, l.Número_Cadeira, l.Fila FROM Lugar l, Sessão s
            WHERE l.Nr_Sala = S.Nr_Sala and S.idSessão = id) AS ls
                LEFT JOIN (SELECT b.Lugar FROM Bilhete b
                                WHERE b.Sessão_id = id) AS lb
                                    ON ls.idLugar = lb.Lugar
                    WHERE lb.Lugar IS null;

    IF erro
        THEN ROLLBACK;
        ELSE COMMIT;
    END IF;
END &&
DELIMITER &&
```

Figura 15 – *MySQL Query* – Lugares disponíveis

```sql
-- Numero da sala de uma determinada sessao
SELECT S.Nr_Sala FROM Sessão AS S
    WHERE S.idSessão = 1;
```

Figura 16 – *MySQL Query* – Número da sala

```sql
-- Hora de uma sessão
SELECT S.Hora_Início FROM Sessão AS S
    WHERE S.idSessão = 1;
```

Figura 17 – *MySQL Query* – Hora da Sessão

```sql
-- O número de bilhetes que ainda restam para uma sessão
SELECT COUNT(ls.idLugar) FROM
    (SELECT l.idLugar FROM Lugar l, Sessão s
        WHERE l.Nr_Sala = S.Nr_Sala and S.idSessão = 1) AS ls
            LEFT JOIN (SELECT b.Lugar FROM Bilhete b
                        WHERE b.Sessão_id = 1) AS lb
                            ON ls.idLugar = lb.Lugar
                WHERE lb.Lugar IS null;
```

Figura 18 - *MySQL Query* – Número de bilhetes disponíveis

```sql
-- Lista de funcionarios que venderam mais bilhetes
SELECT F.idFuncionário, F.Nome, COUNT(B.Funcionário_id) FROM Funcionário AS F
    INNER JOIN Bilhete AS B ON F.idFuncionário = B.Funcionário_id
        GROUP BY F.idFuncionário
        ORDER BY COUNT(B.Funcionário_id) DESC;
```

Figura 19 – *MySQL Query* – Funcionários que venderam mais bilhetes

```sql
-- Nome do funcionário que registou um determinado bilhete
SELECT F.idFuncionário, F.Nome FROM Funcionário AS F
    INNER JOIN Bilhete AS B ON F.idFuncionário = B.Funcionário_id
        WHERE B.idBilhete = 2;
```

Figura 20 – *MySQL Query* – Funcionário que registou determinado bilhete

```sql
-- Quantos bilhetes um determinado tipo de cliente comprou num determinado intervalo de datas;
SELECT COUNT(B.Cliente_NIF) FROM Bilhete AS B
    INNER JOIN Cliente AS C ON B.Cliente_NIF = C.NIF
        WHERE C.Tipo = 'E' AND (B.Data_Emissão BETWEEN '20181120' AND '20181121');
```

Figura 21 – *MySQL Query* – Número de bilhetes que um tipo de cliente comprou

```
-- Lista de sessões com mais bilhetes vendidos num determinado intervalo de datas
SELECT S.idSessão, F.Nome, S.Data, S.Hora_Início, COUNT(B.Sessão_id) FROM Sessão AS S
    INNER JOIN Filme AS F ON S.Filme_id = F.idFilme
        INNER JOIN Bilhete AS B ON S.idSessão = B.Sessão_id
            WHERE S.Data BETWEEN '20181120' AND '20181121'
            GROUP BY S.idSessão
            ORDER BY COUNT(B.Sessão_id) DESC;
```

Figura 22 – *MySQL Query* – Sessões com mais audiências

```
-- Numero de bilhetes que foram vendidos num determinado dia
SELECT COUNT(B.Data_Emissão) FROM Bilhete AS B
    WHERE B.Data_Emissão = '20181121';
```

Figura 23 – *MySQL Query* – Número de bilhetes vendidos num dia

```
-- Lista de filmes em exibiçao com melhores classificaçoes, top3
SELECT F.idFilme, F.Nome, F.Classificação, S.Data FROM Filme AS F
    INNER JOIN Sessão AS S ON F.idFilme = S.Filme_id
    WHERE S.Data = '20181125'
    ORDER BY F.Classificação DESC
    LIMIT 3;
```

Figura 24 – *MySQL Query* – Top 3 filmes melhores classificados

```
-- Lista dos 10 filmes mais vistos no cinema.
SELECT F.idFilme, F.Nome, COUNT(B.Sessão_id) FROM Filme AS F
    INNER JOIN Sessão AS S ON F.idFilme = S.Filme_id
        INNER JOIN Bilhete AS B ON S.idSessão = B.Sessão_id
        GROUP BY F.idFilme
        ORDER BY COUNT(B.Sessão_id) DESC
        LIMIT 10;
```

Figura 25 – *MySQL Query* – 10 filmes mais vistos

5.4. Tradução das transações estabelecidas para SQL

```
DROP PROCEDURE IF EXISTS inserir_cliente;
DELIMITER &&
CREATE PROCEDURE inserir_cliente(IN nif INT,
                                 IN nome VARCHAR(45),
                                 IN tipo CHAR)
BEGIN
    DECLARE erro BOOL DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro=1;
    START TRANSACTION;

    INSERT INTO Cliente VALUES (nif, nome, tipo);

    IF erro
        THEN ROLLBACK;
        ELSE COMMIT;
    END IF;
END &&
DELIMITER &&
```

Figura 26 - Transação de inserção de um novo Cliente

```
DROP PROCEDURE IF EXISTS inserir_funcionário;
DELIMITER &&
CREATE PROCEDURE inserir_funcionário(IN id INT,
                                     IN nome VARCHAR(45))
BEGIN
    DECLARE erro BOOL;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro=1;
    START TRANSACTION;

    INSERT INTO Funcionário VALUES (id, nome);

    IF erro
        THEN ROLLBACK;
        ELSE COMMIT;
    END IF;
END &&
DELIMITER &&
```

Figura 27 - Transação de inserção de um novo Funcionário

```
DROP PROCEDURE IF EXISTS registar_bilhete;
DELIMITER &&
CREATE PROCEDURE registar_bilhete(IN data_emissao DATE,
                                  IN lugar INT,
                                  IN nif_cliente INT,
                                  IN id_sessao INT,
                                  IN id_func INT)
BEGIN
    DECLARE erro BOOL;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro=1;
    START TRANSACTION;

    INSERT INTO Bilhete VALUES ((SELECT calcula_idBilhete()), data_emissao, lugar,
                                                (SELECT calcula_preco(nif_cliente)),
                                                   nif_cliente, id_sessao, id_func);

    IF erro
        THEN ROLLBACK;
        ELSE COMMIT;
    END IF;
END &&
DELIMITER &&
```

Figura 28 - Transação de registo de um Bilhete

```
DROP PROCEDURE IF EXISTS inserir_filme;
DELIMITER &&
CREATE PROCEDURE inserir_filme(IN id INT,
                               IN nome VARCHAR(45),
                               IN descricao TEXT,
                               IN classificacao DECIMAL(2,1))
BEGIN
    DECLARE erro BOOL;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro=1;
    START TRANSACTION;

    INSERT INTO Filme VALUES (id, nome, descricao, classificacao);

    IF erro
        THEN ROLLBACK;
        ELSE COMMIT;
    END IF;
END &&
DELIMITER &&
```

Figura 29 - Transação de inserção de um novo Filme

```
DROP PROCEDURE IF EXISTS inserir_sessao;
DELIMITER &&
CREATE PROCEDURE inserir_sessao(IN data DATE,
                                IN hora TIME,
                                IN duracao TIME,
                                IN nrSala INT,
                                IN filme INT)
BEGIN
    DECLARE erro BOOL;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro=1;
    START TRANSACTION;

    INSERT INTO Sessão VALUES ((SELECT calcula_idSessão()), 4.6, data, hora,
                                                        duracao, nrSala, filme);

    IF erro
        THEN ROLLBACK;
        ELSE COMMIT;
    END IF;
END &&
DELIMITER &&
```

Figura 30 - Transação de inserção de um novo Sessão

5.5. Escolha, definição e caracterização de índices em SQL

Dada a dimensão da Base de Dados não ser significativamente grande não foi sentida a necessidade de definição de tabelas de índices.

5.6. Estimativa do espaço em disco da base de dados e taxa de crescimento anual

Para efetuar a estimativa do espaço em disco da base de dados, e necessário em primeiro lugar, calcular o espaço necessário para cada registo de cada tabela, tendo em consideração o tipo de dados de cada atributo. Multiplicando este valor pelo número de registos inseridos em cada tabela, na fase de povoamento, obtém-se o espaço necessário em disco para a base de dados.

Cliente:

Atributos	Tipo	Tamanho (*Bytes*)
NIF	INT	4
Nome	VARCHAR(45)	45
Tipo	VARCHAR(1)	1
Total por registo [1]:		**50**
Nº Registos [2]:		**194**
TOTAL ([1] x [2]):		**9700 *bytes***

Tabela 4 - Tamanho dos atributos de Cliente

Funcionário:

Atributos	Tipo	Tamanho (*Bytes*)
idFuncionário	INT	4
Nome	VARCHAR(45)	45
Total por registo [1]:		**49**
Nº Registos [2]:		**4**
TOTAL ([1] x [2]):		**196 *bytes***

Tabela 5 - Tamanho dos atributos de Funcionário

Bilhete:

Atributos	Tipo	Tamanho (*Bytes*)
idBilhete	INT	4
Sala	INT	4
Data_Emissão	DATE	3
Lugar	INT	4
Preço	DECIMAL(3,2)	2
Total por registo [1]:		**17**
Nº Registos [2]:		**207**
TOTAL ([1] x [2]):		**3519 *bytes***

Tabela 6 - Tamanho dos atributos de Bilhete

Sessão:

Atributos	Tipo	Tamanho (*Bytes*)
idSessão	INT	4
Preço_Base	DECIMAL(3,2)	2
Data	DATE	3
Hora_Início	TIME	3
Duração	TIME	3
Total por registo [1]:		**15**
Nº Registos [2]:		**46**
TOTAL ([1] x [2]):		**690 *bytes***

Tabela 7 - Tamanho dos atributos de Sessão

Filme:

Atributos	Tipo	Tamanho (*Bytes*)
idFilme	INT	4
Nome	VARCHAR(45)	45
Descrição	TEXT	42
Classificação	INT	4
Total por registo [1]:		**95**
Nº Registos [2]:		**16**
TOTAL ([1] x [2]):		**1520 *bytes***

Tabela 8 - Tamanho dos atributos de Filme

Sala:

Atributos		Tipo	Tamanho (*Bytes*)
Número_Sala		INT	4
Lugar	idLugar	INT	4
	Número_Cadeira	INT	4
	Fila	VARCHAR(1)	1
Total por registo [1]:			**13**
Nº Registos [2]:			**5**
TOTAL ([1] x [2]):			**65 *bytes***

Tabela 9 - Tamanho dos atributos de Sala

Para se obter o espaço em disco ocupado, basta agora somar o tamanho ocupado pelas várias tabelas:

Cliente: 9700 *bytes*
Funcionário: 196 *bytes*
Bilhete: 3519 *bytes*
Sessão: 690 *bytes*
Filme: 1520 *bytes*
Sala: 65 *bytes*
15690 *bytes*

Após povoamento, o SGBD ocupa, no máximo, **15690** *bytes*.

Em relação à taxa de crescimento foram efetuadas as estimativas apresentadas de seguida, com base em números do atual cinema e valores previstos com a reestruturação do mesmo.

- Número de salas: mantem-se constante, em relação ao valor inicial.

- Número de funcionários: mantêm-se aproximadamente constante ao longo do tempo. Mesmo que existam algumas oscilações, relativas a eventuais reforços em épocas mais movimentadas, este valor pode ser considerado residual.

- Número de Cliente: fez-se uma estimativa de 30 novos clientes por mês, o que, ao final do ano dá um total de 360 novos clientes.

- Número de Bilhetes: fez-se uma estimativa de 2000 bilhetes vendidos por semana, o que ao final do ano corresponde a 104.000 bilhetes.

- Número de Sessões: para as sessões estimou-se a realização de 18 sessões semanais por sala (8 sessões ao fim-de-semana e 10 distribuídas ao longo da semana). Isto, para 5 salas, corresponde a um valor de 4.680 novas sessões introduzidas ao ano.

- Número de Filmes: admitiu-se um valor de 2 estreias semanais, o que corresponde a 104 novos filmes por ano introduzidos na base de dados.

Multiplicando os números obtidos acima pelo valor unitário de cada registo das tabelas obtém-se os seguintes valores:

Entidade	Tamanho por registo (*bytes*)	Estimativa de novos registos/ano	Total
Sala	13	0	0
Funcionário	49	0	0
Cliente	50	360	18 000
Bilhete	17	104 000	1 768 000
Sessão	15	4 680	70 200
Filme	95	104	9 880
		TOTAL:	**1 866 080 *bytes* (1 822,24 *KBytes*)**

Tabela 10 – Crescimento Anual

Com base nos valores estimados, obteve-se um valor de crescimento anual na ordem dos 1800 KBytes ao ano. Com estes resultados não se verifica nenhum tipo de constrangimento em relação aos crescimentos da Base de Dados ao longo da sua utilização.

5.7. Definição e caracterização das vistas de utilização em SQL

Na base de dados desenvolvida não se mostrou necessária a definição de vistas de utilização visto que não existem no sistema dados sensíveis, como por exemplo dados pessoais confidenciais, e também porque não existe necessidade de restrição na consulta e acesso aos dados por parte dos vários utilizadores da Base de Dados.

5.8. Definição e caracterização dos mecanismos de segurança em SQL

Neste capítulo encontram-se definidas as permissões de acesso por parte de cada um dos perfis criados, neste caso, o Administrador e Funcionários.

- Criação de Utilizador do tipo Administrador:

```
CREATE USER 'admin'@'localhost';
    SET PASSWORD FOR 'admin'@'localhost' = 'adminpass';
```

Figura 31 - Criação de administrador

- Criação de Utilizadores do tipo Funcionário:

```
CREATE USER 'func1'@'localhost';
    SET PASSWORD FOR 'func1'@'localhost' = 'func1pass';

CREATE USER 'func2'@'localhost';
    SET PASSWORD FOR 'func2'@'localhost' = 'func2pass';

CREATE USER 'func3'@'localhost';
    SET PASSWORD FOR 'func3'@'localhost' = 'func3pass';

CREATE USER 'func4'@'localhost';
    SET PASSWORD FOR 'func4'@'localhost' = 'func4pass';
```

Figura 32 - Criação de Funcionários

Em relação ao Administrador foram-lhe dadas permissões para efetuar qualquer tipo operações sobre as tabelas.

```
GRANT SELECT, INSERT, DELETE, UPDATE ON Cinema.* TO 'admin'@'localhost'
    WITH GRANT OPTION;
```

Figura 33 - Permissões do administrador

No caso dos utilizadores do tipo ‘funcionário’, apenas lhes foram dadas permissões para efetuar qualquer tipo de operação sobre as tabelas ‘Cliente’ e ‘Bilhete.

Em relação às restantes tabelas apenas têm permissão para consulta das mesmas.

```
GRANT SELECT, INSERT, DELETE, UPDATE ON Cinema.Bilhete TO 'func1'@'localhost',
                                                          'func2'@'localhost',
                                                          'func3'@'localhost',
                                                          'func4'@'localhost';


GRANT SELECT, INSERT, DELETE, UPDATE ON Cinema.Cliente TO 'func1'@'localhost',
                                                          'func2'@'localhost',
                                                          'func3'@'localhost',
                                                          'func4'@'localhost';


GRANT SELECT ON Cinema.Sala TO 'func1'@'localhost',
                               'func2'@'localhost',
                               'func3'@'localhost',
                               'func4'@'localhost';

REVOKE INSERT, DELETE, UPDATE ON Cinema.Sala FROM 'func1'@'localhost',
                                                  'func2'@'localhost',
                                                  'func3'@'localhost',
                                                  'func4'@'localhost';


GRANT SELECT ON Cinema.Lugar TO 'func1'@'localhost',
                                'func2'@'localhost',
                                'func3'@'localhost',
                                'func4'@'localhost';

REVOKE INSERT, DELETE, UPDATE ON Cinema.Lugar FROM 'func1'@'localhost',
                                                   'func2'@'localhost',
                                                   'func3'@'localhost',
                                                   'func4'@'localhost';


GRANT SELECT ON Cinema.Sessão TO 'func1'@'localhost',
                                 'func2'@'localhost',
                                 'func3'@'localhost',
                                 'func4'@'localhost';

REVOKE INSERT, DELETE, UPDATE ON Cinema.Sessão FROM 'func1'@'localhost',
                                                    'func2'@'localhost',
                                                    'func3'@'localhost',
                                                    'func4'@'localhost';
```

Figura 34 - Permissões do Funcionário

De forma a tornar a BD mais segura no que diz respeito à integridade de dados, foram implementados três '*triggers*', com os seguintes objetivos: [1] impedir que o cliente compre um bilhete para uma sessão que já foi realizada, [2] impedir que o cliente escolha um lugar que já esta ocupado ou que não existe e [3] impedir que, na inserção de uma sessão, seja escolhida uma sala que já esteja ocupada no mesmo dia, naquela hora.

```
DROP TRIGGER IF EXISTS dataBilhete;
DELIMITER &&
CREATE TRIGGER dataBilhete
BEFORE INSERT ON Bilhete
FOR EACH ROW
BEGIN
    DECLARE message VARCHAR(100);

    IF(SELECT ((SELECT S.Data FROM Sessão AS S WHERE S.idSessão = NEW.Sessão_id) < NEW.Data_Emissão))
    THEN SET message = 'Esta Sessão já foi realizada.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;

    END IF;
END &&
DELIMITER &&
```

Figura 35 - *Trigger* [1]

```
DROP TRIGGER IF EXISTS lugarBilhete;
DELIMITER &&
CREATE TRIGGER lugarBilhete
BEFORE INSERT ON Bilhete
FOR EACH ROW
BEGIN
    DECLARE message VARCHAR(100);

    IF((NEW.Lugar IN (SELECT Lugar FROM Bilhete WHERE Sessão_id = NEW.Sessão_id)) OR
        (NEW.Lugar > (SELECT COUNT(l.idLugar) FROM Lugar l, Sessão s
                      WHERE l.Nr_Sala = S.Nr_Sala and S.idSessão = NEW.Sessão_id)))
    THEN SET message = 'O lugar não é válido.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;

    END IF;
END &&
DELIMITER &&
```

Figura 36 - *Trigger* [2]

```
DROP TRIGGER IF EXISTS salaOcupada;
DELIMITER &&
CREATE TRIGGER salaOcupada
BEFORE INSERT ON Sessão
FOR EACH ROW
BEGIN
    DECLARE message VARCHAR(100);

    IF(NEW.Nr_Sala IN (SELECT DISTINCT(ss.Nr_Sala) FROM Sessão ss
                       WHERE (ss.Data = NEW.Data) AND (NEW.Hora_Início < (ss.Hora_Início + ss.Duração))
                           AND ((NEW.Hora_Início + NEW.Duração) > ss.Hora_Início)
                       GROUP BY ss.Nr_Sala))
    THEN SET message = 'A Sala está ocupada.';
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;

    END IF;
END &&
DELIMITER &&
```

Figura 37 - *Trigger* [3]

5.9. Revisão do sistema implementado com o utilizador

Concluído todo o processo, e após todas as verificações de funcionamento do sistema implementado, este deveria ser revisto e testado pelo utilizador de forma a que verificasse se o mesmo se encontra de acordo com o solicitado.

6. Sistema de Bases de Dados não Relacional

6.1. Justificação da utilização de um sistema NoSQL

O principal objetivo do grupo com a realização deste projeto é adquirir rotinas de planeamento e execução de sistemas de bases de dados não relacionais.

Após um aumento considerável do volume de dados no nosso SBD, a questão da eficiência começou a tornar-se muito importante. Como tal, foi necessário explorar linguagens NoSQL para tentar resolver este problema.

Uma das principais vantagens em trocar um SBD relacional por uma solução NoSQL, em particular orientado por grafos, é o seu escalonamento. Como este não possuí nenhum tipo de esquema pré-definido, o modelo possui maior flexibilidade o que favorece a inserção transparente de outros elementos.

Outro fator fundamental do sucesso desse modelo é a sua disponibilidade. A grande distribuição dos dados leva a que um maior número de solicitações aos dados seja executado pelo sistema num menor intervalo de tempo.

Optou-se por um SBD não relacional orientado por grafos, suportado pela aplicação Neo4j.

6.2. Migração de Dados

Nesta parte do relatório explicar-se-á como foi feito o processo de migração de dados do sistema de base de dados relacional apresentado no capítulo anterior para um sistema de base de dados não relacional orientado por grafos (Graphs Databases). Para esse processo, foi utilizada a ferramenta Neo4j.

6.2.1 Ficheiros CSV

Num primeiro passo, para iniciar o processo de migração de dados, exportou-se cada uma das tabelas do modelo relacional para o seu respetivo ficheiro ".csv". Este passo é exemplificado de seguida (exportação de dados da tabela funcionário).

Figura 38 - Tabela Funcionário no *MySQL*

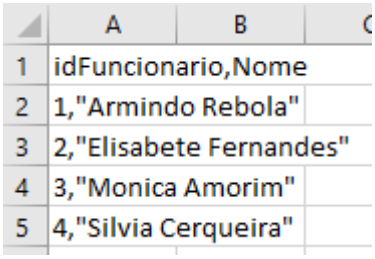

Figura 39 - Ficheiro "Funcionario.csv"

6.3. Importação de Dados

Neste tópico é explicada como foi feita a importação de dados dos ficheiros ".csv" descritos no tópico anterior.

Para importar cada uma das linhas de cada ficheiro e criar os seus respetivos nodos, foram usados comandos específicos de *cypher* que serão também demonstrados.

6.3.1 Funcionário

Como se pode verificar no modelo de dados relacional, a tabela Funcionário contém uma chave primária (idFuncionário) e um atributo (Nome). Assim, ao fazer a importação de dados e criação de nodos, são importados todos os dados do ficheiro “Funcionário.csv”.

```
LOAD CSV WITH HEADERS FROM 'file:///Funcionario.csv' AS row
CREATE (funcionario: Funcionario {idFuncionario: toInteger(row.idFuncionario)})
SET funcionario.Nome = toString(row.Nome),
RETURN funcionario;
```

Figura 40 - Import do ficheiro "Funcionario.csv"

Tal como comprovado na figura seguinte, os nodos Funcionario foram corretamente criados.

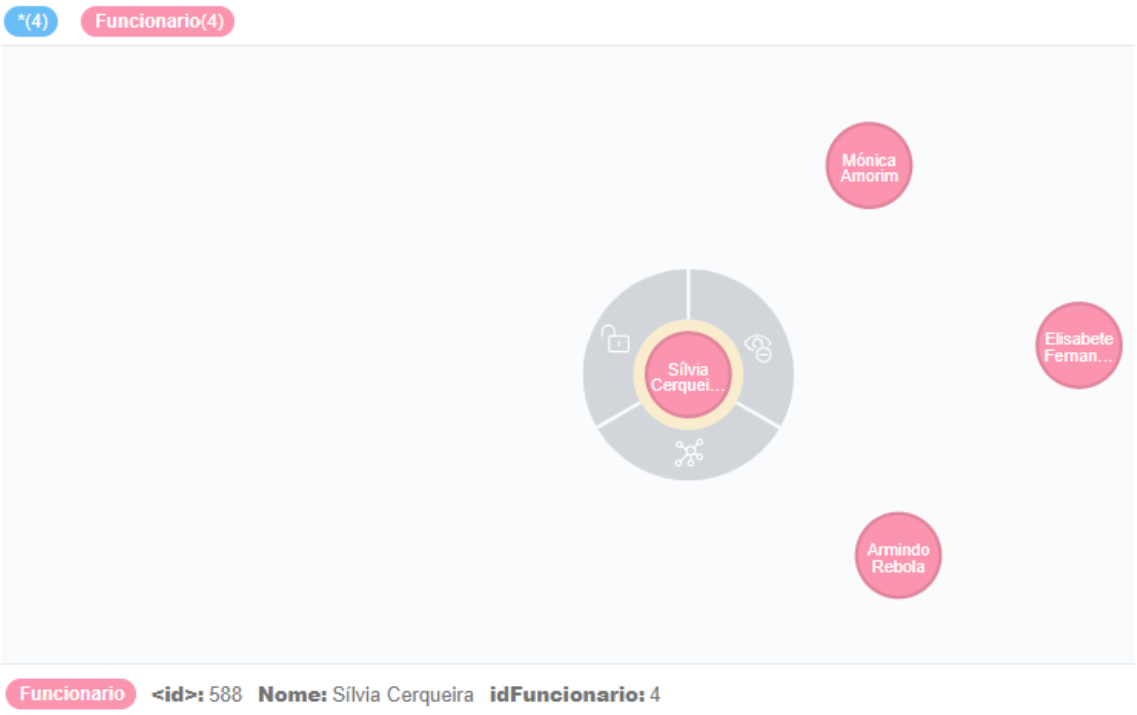

Figura 41 - Neo4j – Nodos Funcionario

6.3.2 Cliente

No modelo de dados relacional pode-se verificar que a tabela Cliente contém uma chave primária (NIF) e dois atributos (Nome, Tipo). Assim, ao fazer a importação de dados e criação de nodos, são importados todos os dados do ficheiro “Cliente.csv”.

```
LOAD CSV WITH HEADERS FROM 'file:///Cliente.csv' AS row
CREATE (cliente: Cliente {NIF: toInteger(row.NIF)})
SET cliente.Nome = toString(row.Nome),
    cliente.Tipo = toString(row.Tipo)
RETURN cliente;
```

Figura 42 - Import do ficheiro "Cliente.csv"

Tendo em conta a figura seguinte, verifica-se a correta criação dos nodos Cliente.

Figura 43 - Neo4j – Nodos Cliente

6.3.3 Filme

No modelo de dados relacional, a tabela Filme contém uma chave primária (idFilme) e três atributos (Nome, Descrição, Classificação). Assim, ao fazer a importação de dados e criação de nodos, são importados todos os dados do ficheiro “Filme.csv”.

```
LOAD CSV WITH HEADERS FROM 'file:///Filme.csv' AS row
CREATE (filme: Filme {idFilme: toInteger(row.idFilme)})
SET filme.Nome = toString(row.Nome),
    filme.Descricao = toString(row.Descricao),
    filme.Classificacao = toFloat(row.Classificacao)
RETURN filme;
```

Figura 44 - Import do ficheiro "Filme"

Os nodos Filme foram corretamente criados (figura 45).

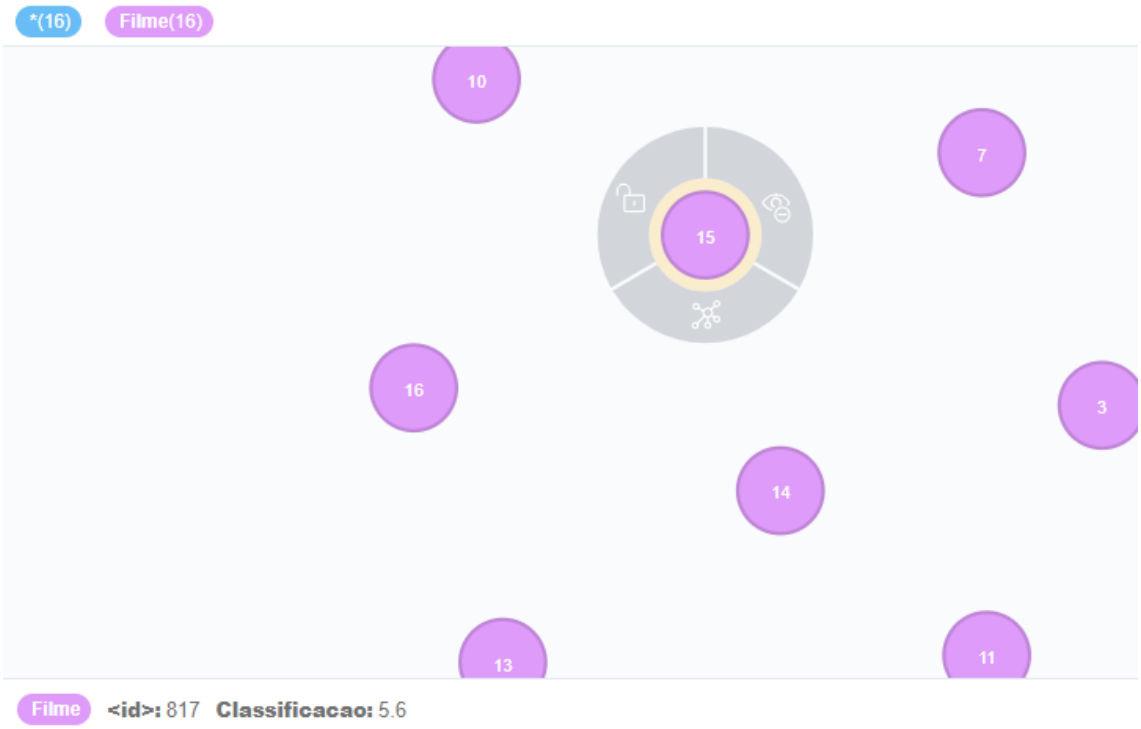

Figura 45 - Neo4j - Nodos Filme

6.3.4 Sala

A tabela Sala contém uma chave primária (Número_Sala) e não contém nenhum atributo, como verificado no modelo de dados relacional. Assim, ao fazer a importação de dados e criação de nodos, são importados todos os dados do ficheiro “Sala.csv”.

```
LOAD CSV WITH HEADERS FROM 'file:///Sala.csv' AS row
CREATE (sala: Sala {Numero_Sala: toInteger(row.Numero_Sala)})
RETURN sala;
```

Figura 46 - Import do ficheiro "Sala"

A figura seguinte comprova que os nodos Sala foram corretamente criados.

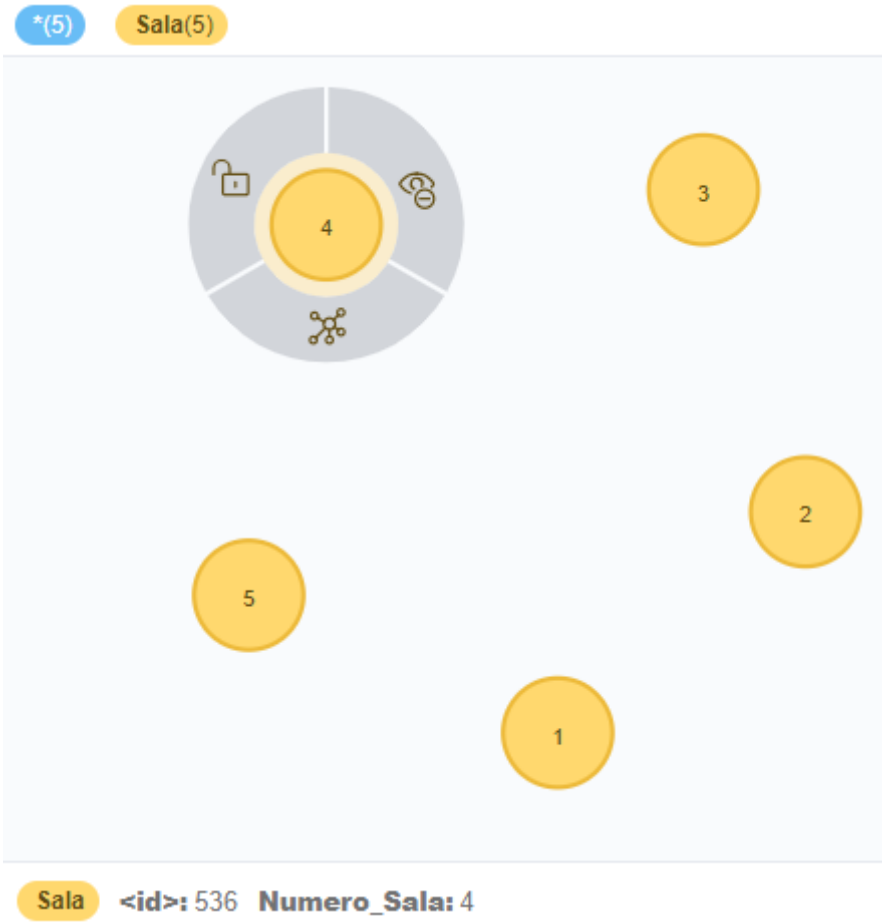

Figura 47 - Neo4j - Nodos Sala

6.3.5 Bilhete

Como se pode verificar no modelo de dados relacional, a tabela Bilhete contém uma chave primária (idBilhete), três atributos (Data_Emissão, Lugar e Preço) e ainda três chaves estrangeiras (Cliente_NIF, Sessão_id e Funcionário_id).

Uma vez que as chaves estrangeiras no modelo relacional servem para garantir a integridade referencial, e como no modelo não relacional esta é garantida pelos relacionamentos, apenas é necessário importar os dados idBilhete, Data_Emissão, Lugar e Preço do ficheiro “Bilhete.csv”.

```
LOAD CSV WITH HEADERS FROM 'file:///Bilhete.csv' AS row
CREATE (bilhete: Bilhete {idBilhete: toInteger(row.idBilhete)})
SET bilhete.Data_Emissao = toString(row.Data_Emissao),
    bilhete.Lugar = toInteger(row.Lugar),
    bilhete.Preco = toFloat(row.Preco)
RETURN bilhete;
```

Figura 48 - Import do ficheiro "Bilhete"

Os nodos Bilhete foram corretamente criados (figura 49).

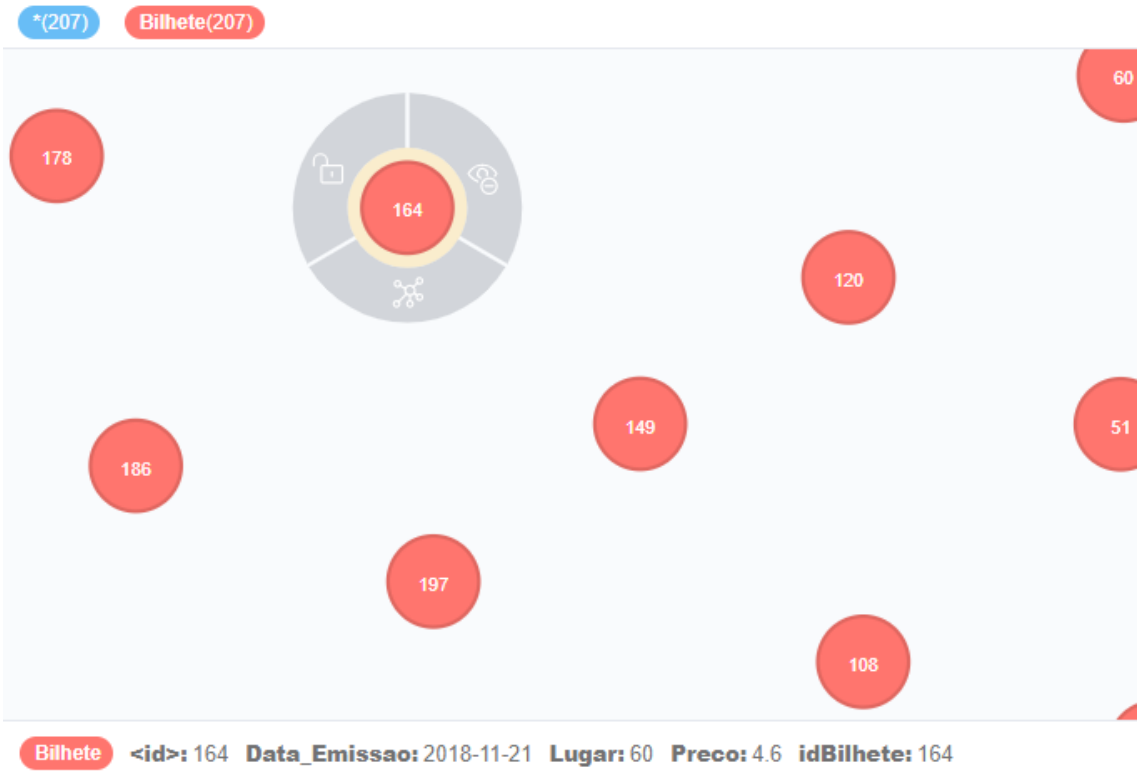

Figura 49 - Neo4j - Nodos Bilhete

6.3.6 Sessão

A tabela Sessão contém uma chave primária (idSessão), quatro atributos (Preço_Base, Data, Hora_Início e Duração) e duas chaves estrangeiras (Nr_Sala e Filme_id), como confirmado no modelo de dados relacional.

Tal como referido na situação do Bilhete do ponto anterior, apenas é necessário importar idSssão, Preço_Base, Data, Hora_Início e Duração do ficheiro “Sessão.csv”.

```
LOAD CSV WITH HEADERS FROM 'file:///Sessao.csv' AS row
CREATE (sessao: Sessao {idSessao: toInteger(row.idSessao)})
SET sessao.Preco_Base = toFloat(row.Preco_Base), sessao.Data = toString(row.Data),
    sessao.Hora_Inicio = toString(row.Hora_Inicio), sessao.Duracao = toString(row.Duracao)
RETURN sessao;
```

Figura 50 - Import do ficheiro "Sessao"

A criação dos nodos Sessao foi verificada.

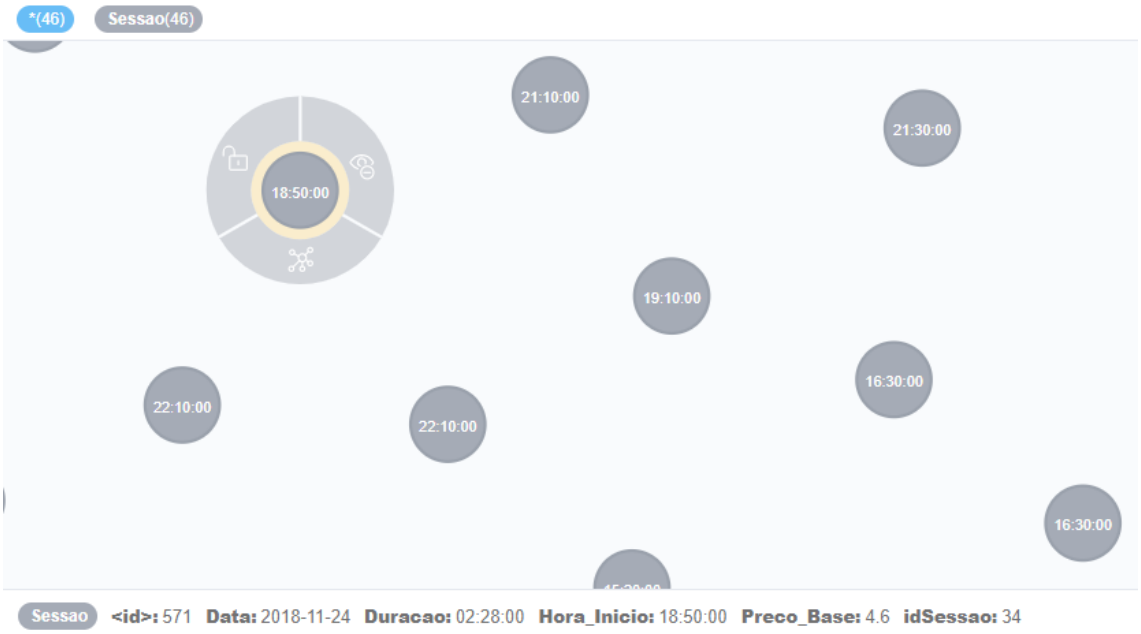

Figura 51 - Neo4j - Nodos Sessao

6.3.7 Lugar

Como evidenciado no modelo de dados relacional, a tabela Lugar contém uma chave primária composta por dois atributos (idLugar e Nr_Sala), e dois atributos (Número_Cadeira e Fila). No entanto, o atributo Nr_Sala é uma chave estrangeira.

Assim, ao contrário das situações do Bilhete e da Sessão em que não era requerida a importação da chave estrangeira, neste caso ter-se-á de a importar, pois ela compõe a chave primária, sendo fundamental para identificar cada lugar de cada sala.

Ao fazer a importação de dados e criação de nodos, são importados todos os dados do ficheiro “Lugar.csv”.

```
LOAD CSV WITH HEADERS FROM 'file:///Lugar.csv' AS row
CREATE (lugar: Lugar {idLugar: toInteger(row.idLugar)})
SET lugar.Nr_Sala = toInteger(row.Nr_Sala),
    lugar.Numero_Cadeira = toInteger(row.Numero_Cadeira),
    lugar.Fila = toString(row.Fila)
RETURN lugar;
```

Figura 52 - Import do ficheiro "Lugar"

Os nodos Lugar foram corretamente criados, como podemos comprovar na figura seguinte.

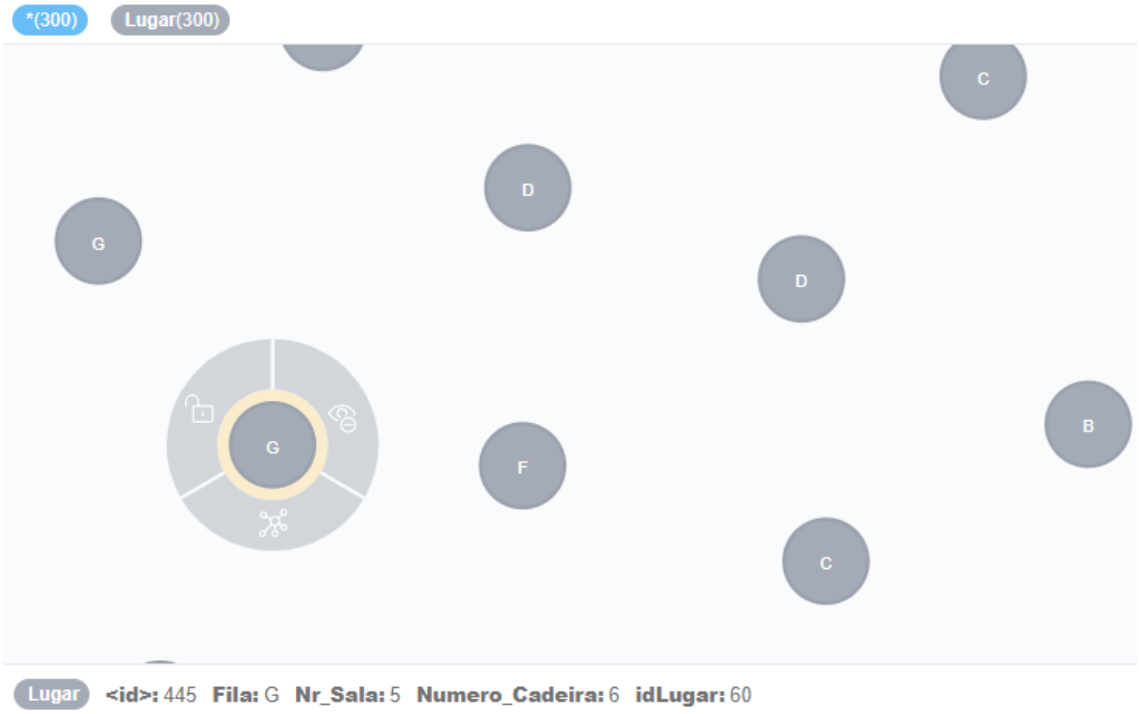

Figura 53 - Neo4j - Nodos Lugar

6.4. *Unique Constraints*

Para uma correta migração do sistema, e de modo a impedir possíveis erros futuros de inserção de dados, foram criadas algumas Unique Constraints para garantir que determinados valores só podem existir uma única vez em cada nodo.

Por exemplo, nos nodos de Funcionario foi criada uma Unique Constraint que garante que não podem existir id's de funcionarios repetidos. Tal como pode ser comprovado pela figura seguinte, a constraint foi criada corretamente.

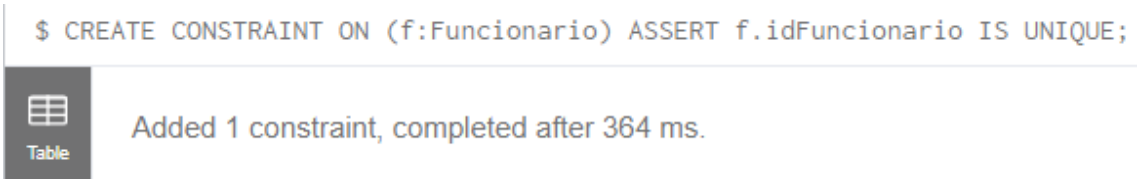

Figura 54 - *Unique Constraint* Funcionario

As restantes *constraints* criadas podem ser verificadas na figura abaixo.

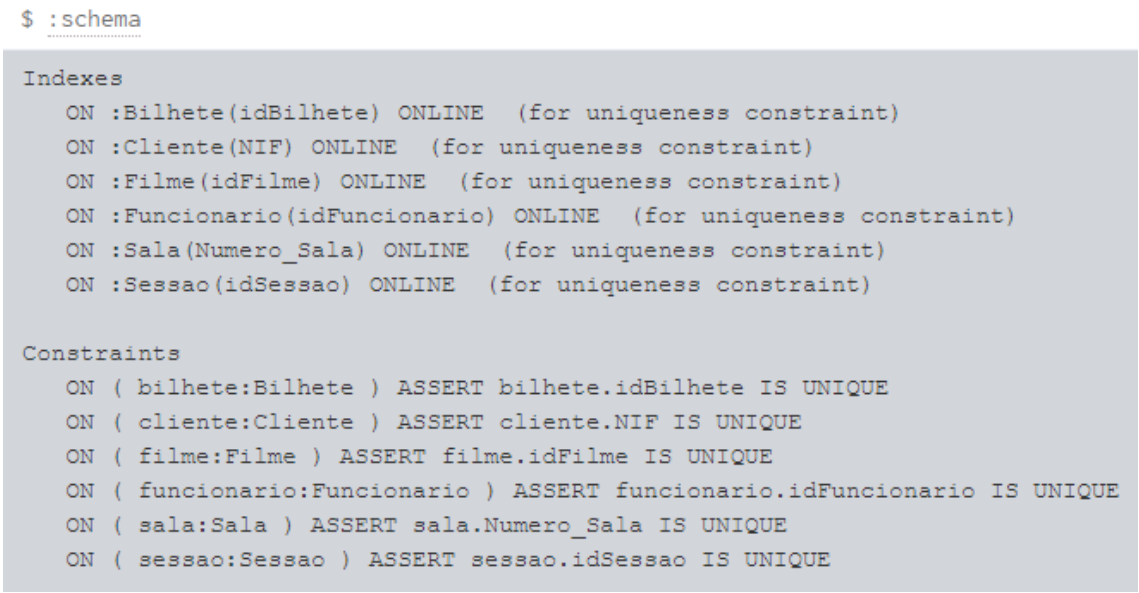

Figura 55 - Restantes *Constraints*

6.5. Relacionamentos

Em seguida, será explicado como criámos os diferentes relacionamentos existentes no sistema, com a demonstração dos respetivos comandos utilizados no *cypher*, bem como o comprovativo da correta criação dos relacionamentos.

6.5.1 Sessão Tem Bilhete

Para criar o relacionamento entre Sessão e Bilhete, é necessário procurar no ficheiro "Bilhete.csv" os números de sessões que aparecem na coluna Sessão_id (correspondente à chave estrangeira do modelo relacional).

```
LOAD CSV WITH HEADERS FROM 'file:///Bilhete.csv' AS row
MATCH (s: Sessao {idSessao: toInteger(row.Sessao_id)})
MATCH (b: Bilhete {idBilhete: toInteger(row.idBilhete)})
CREATE (s)-[:TEM]->(b);
```

Figura 56 - Relacionamento Sessão Tem Bilhete

Como podemos verificar na figura seguinte, o relacionamento foi criado corretamente.

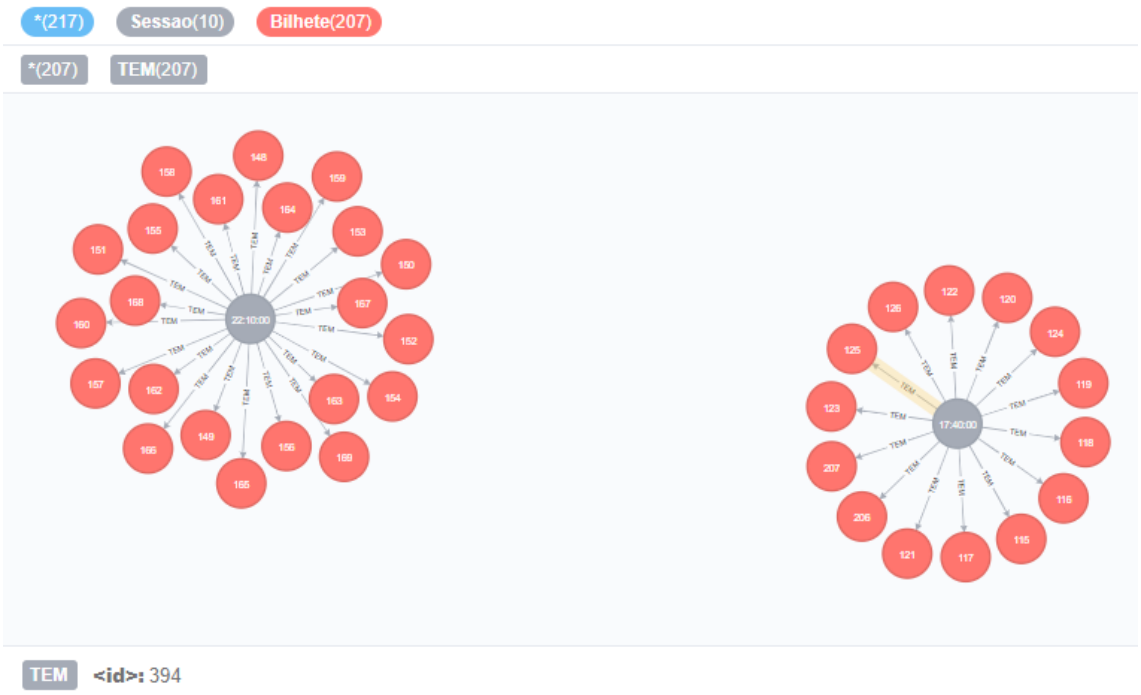

Figura 57 - Neo4j - Relacionamento Sessão Tem Bilhete

6.5.2 Sala Realiza Sessão

Para criar o relacionamento entre Sala e Sessão, é necessário procurar no ficheiro "Sessao.csv" os números das salas que aparecem na coluna Nr_Sala (correspondente à chave estrangeira do modelo relacional).

```
LOAD CSV WITH HEADERS FROM 'file:///Sessao.csv' AS row
MATCH (s: Sala {Numero_Sala: toInteger(row.Nr_Sala)})
MATCH (se: Sessao {idSessao : toInteger(row.idSessao)})
CREATE (s)-[:REALIZA_SESSAO]->(se);
```

Figura 58 - Relacionamento Sala Realiza Sessão

Como podemos verificar na figura seguinte, o relacionamento foi criado corretamente.

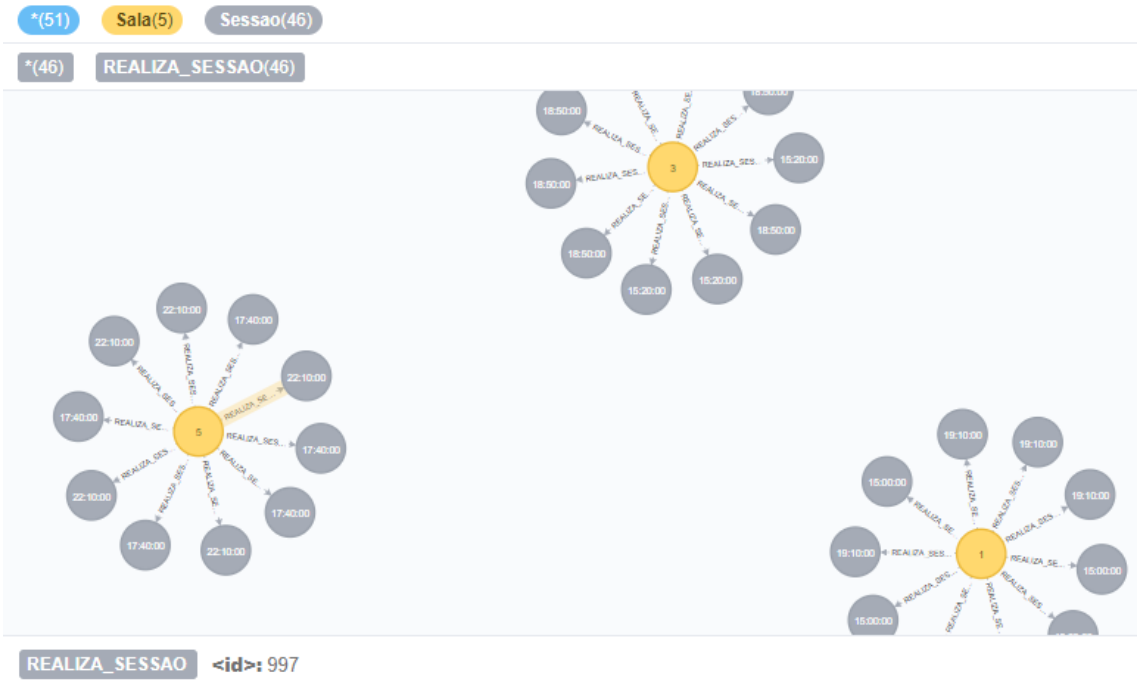

Figura 59 - Neo4j - Relacionamento Sala Realiza Sessão

6.5.3 Sessão Exibe Filme

Para criar o relacionamento entre Sessão e Filme, é necessário procurar no ficheiro “Sessao.csv” os números dos filmes que aparecem na coluna Filme_id (correspondente à chave estrangeira do modelo relacional).

```
LOAD CSV WITH HEADERS FROM 'file:///Sessao.csv' AS row
MATCH (f: Filme {idFilme: toInteger(row.Filme_id)})
MATCH (s: Sessao {idSessao : toInteger(row.idSessao)})
CREATE (s)-[:EXIBE_FILME]->(f);
```

Figura 60 - Relacionamento Sessão Exibe Filme

Como podemos verificar na figura seguinte, o relacionamento foi criado corretamente.

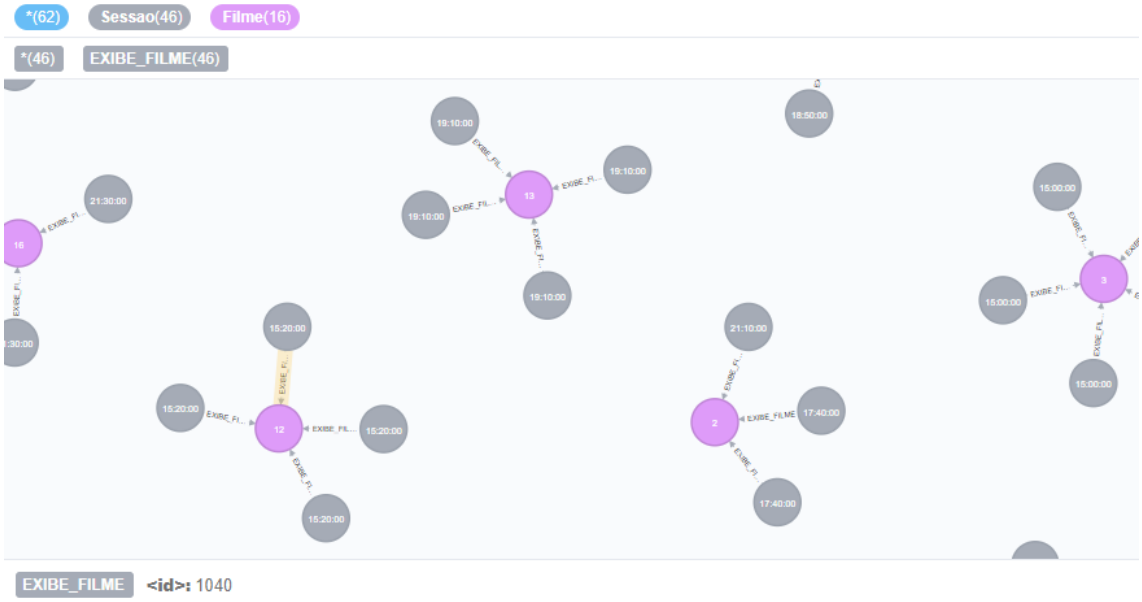

Figura 61 - Neo4j - Relacionamento Sessão Exibe Filme

6.5.4 Cliente Compra Bilhete

Para criar o relacionamento entre Cliente e Bilhete, é necessário procurar no ficheiro "Bilhete.csv" os nifs dos clientes que aparecem na coluna Cliente_NIF (correspondente à chave estrangeira do modelo relacional).

```
LOAD CSV WITH HEADERS FROM 'file:///Bilhete.csv' AS row
MATCH (c: Cliente {NIF: toInteger(row.Cliente_NIF)})
MATCH (b: Bilhete {idBilhete: toInteger(row.idBilhete)})
CREATE (c)-[:COMPRA]->(b);
```

Figura 62 - Relacionamento Cliente Compra Bilhete

Como podemos verificar na figura seguinte, o relacionamento foi criado corretamente.

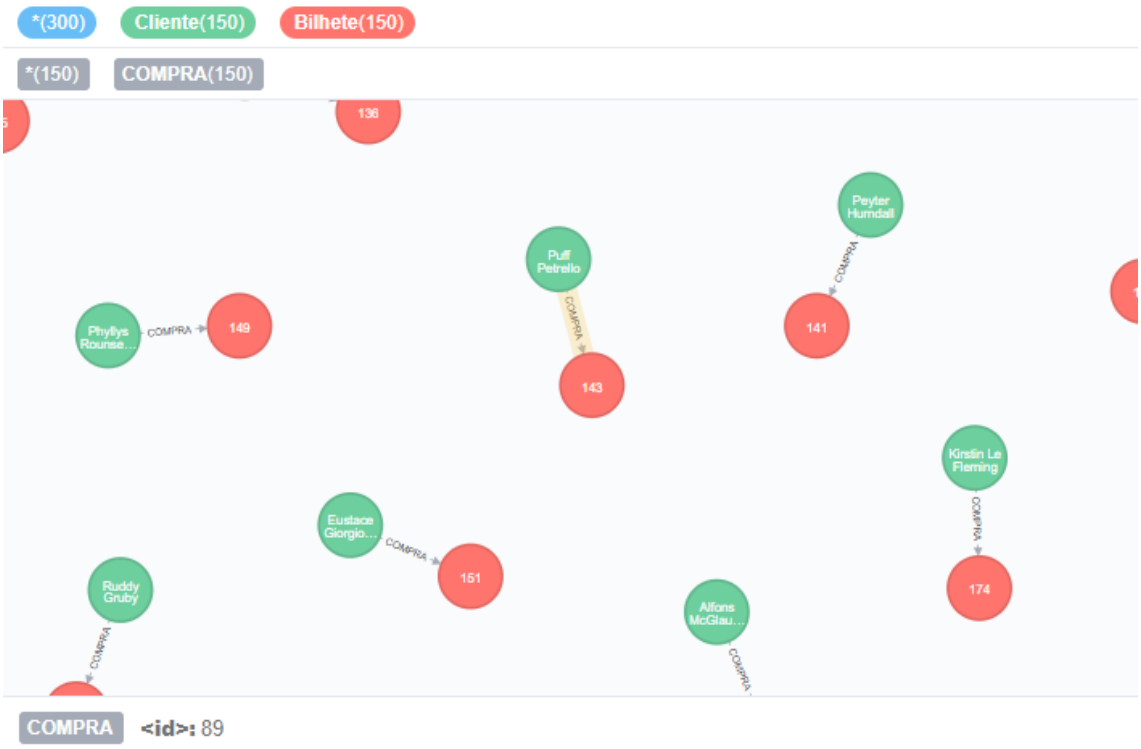

Figura 63 - Neo4j - Relacionamento Cliente Compra Bilhete

6.5.5 Funcionário Regista Bilhete

Para criar o relacionamento entre Funcionário e Bilhete, é necessário procurar no ficheiro “Bilhete.csv” os números dos funcionários que aparecem na coluna Funcionario_id (correspondente à chave estrangeira do modelo relacional).

```
LOAD CSV WITH HEADERS FROM 'file:///Bilhete.csv' AS row
MATCH (f: Funcionario {idFuncionario: toInteger(row.Funcionario_id)})
MATCH (b: Bilhete {idBilhete: toInteger(row.idBilhete)})
CREATE (f)-[:REGISTA]->(b);
```

Figura 64 - Relacionamento Funcionário Regista Bilhete

Como podemos verificar na figura seguinte, o relacionamento foi criado corretamente.

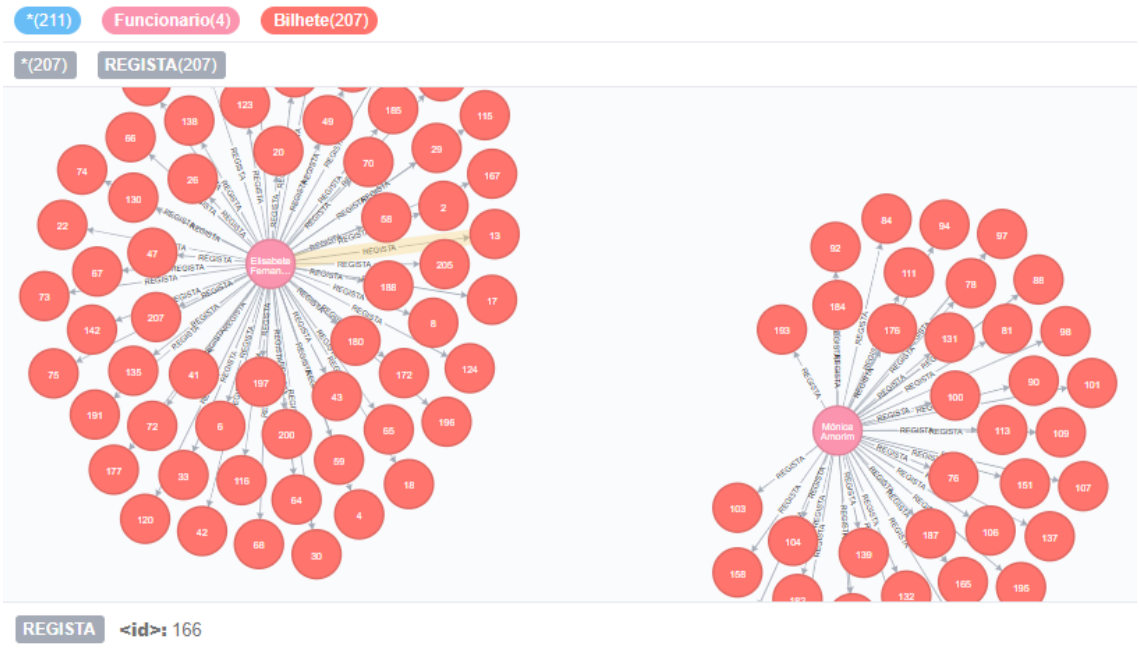

Figura 65 - Neo4j - Relacionamento Funcionário Regista Bilhete

6.5.6 Sala Tem Lugar

Para criar o relacionamento entre Sala e Lugar, ao contrário dos exemplos anteriores, não é necessário procurar no ficheiro “Lugar.csv” uma vez que o número da sala de cada lugar já foi importado anteriormente.

```
MATCH (l: Lugar),(s: Sala)
WHERE l.Nr_Sala = s.Numero_Sala
CREATE (s)-[:TEM_LUGAR]->(l);
```

Figura 66 - Relacionamento Sala Tem Lugar

Como podemos verificar na figura seguinte, o relacionamento foi criado corretamente.

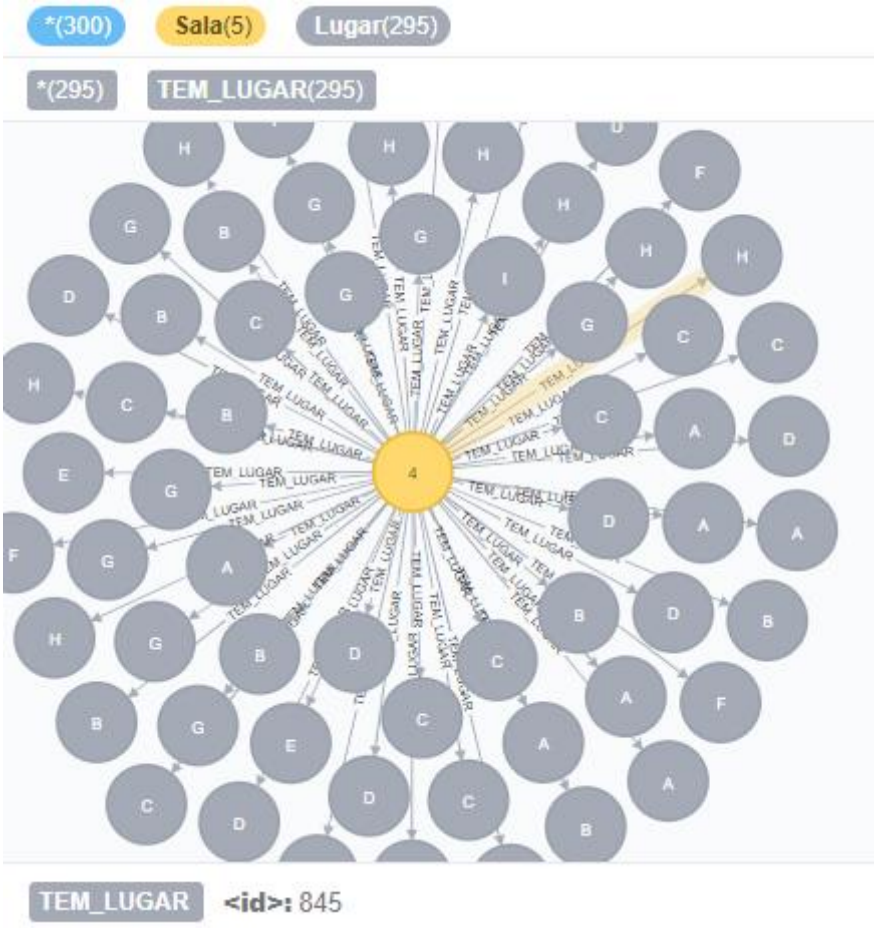

Figura 67 - Neo4j - Relacionamento Sala Tem Lugar

6.6. Modelo Neo4j

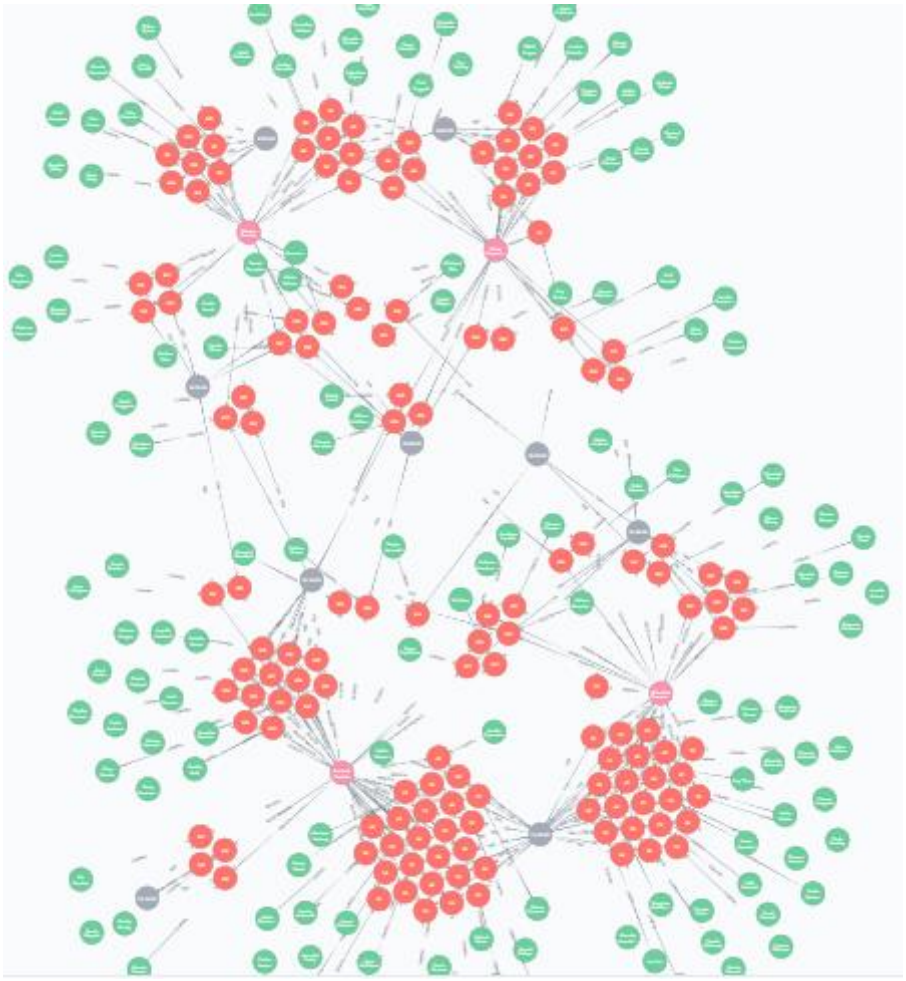

Figura 68 - Esquema Neo4j

6.7. *Queries*

De modo a demonstrar a operacionalidade do sistema implementado, serão exemplificadas de seguida um conjunto de *queries*, utilizado a linguagem de interrogação do Neo4j (*cypher*).

Por outro lado, é demonstrado também uma comparação com os resultados da mesma *query* no modelo relacional.

6.7.1 Quais os clientes que compraram mais bilhetes?

```
MATCH (c:Cliente)-[:COMPRA]->(b:Bilhete)
RETURN c.Nome, COUNT(b)
ORDER BY COUNT(b) DESC;
```

```
$ MATCH (c:Cliente)-[:COMPRA]->(b:Bilhete) RETURN c.Nome, COUNT(b) ORDER BY COUNT(b) DESC
```

Table | Text | Code

c.Nome	COUNT(b)
"Ricardo Pereira"	4
"Orlando Belo"	3
"Raquel Dias"	2
"Luís Abreu"	2
"Ottilie Ebsworth"	2
"Rita Guimarães"	2

Figura 69 - *Query* 1 Neo4j

```
-- Lista de Clientes Ordenada pelo numero de bilhetes comprados
SELECT C.Nome, COUNT(B.idBilhete) FROM Cliente AS C
    INNER JOIN Bilhete AS B ON C.NIF = B.Cliente_NIF
    GROUP BY C.NIF
    ORDER BY COUNT(B.idBilhete) DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Nome	COUNT(B.idBilhete)
Ricardo Pereira	4
Orlando Belo	3
Rita Guimarães	2
Amitie Symmers	2
Luís Abreu	2
Raquel Dias	2

Figura 70 - *Query* 1 *SQL*

6.7.2 Quais são os filmes que se encontram em exibição numa determinada data?

```
MATCH (sa:Sala)-[:REALIZA_SESSAO]->(s:Sessao)-[:EXIBE_FILME]->(f:Filme)
WHERE s.Data = '2018-11-21'
RETURN f.idFilme, f.Nome, s.Data, s.Hora_Inicio, sa.Numero_Sala
ORDER BY s.Hora_Inicio;
```

$ MATCH (sa:Sala)-[:REALIZA_SESSAO]->(s:Sessao)-[:EXIBE_FILME]->(f:Filme) WHERE s.Data = '2018-11-21' RETURN f.idFilme, f…

Table | Text | Code

f.idFilme	f.Nome	s.Data	s.Hora_Inicio	sa.Numero_Sala
3	"O Quebra-Nozes"	"2018-11-21"	"15:00:00"	1
6	"Belleville Cop - O Super Agente"	"2018-11-21"	"15:20:00"	3
9	"Viúvas"	"2018-11-21"	"16:30:00"	2
5	"A Rapariga Apanhada na Teia de Aranha"	"2018-11-21"	"17:40:00"	5
4	"Bohemian Rhapsody"	"2018-11-21"	"18:50:00"	3
1	"Coração Traiçoeiro"	"2018-11-21"	"19:10:00"	1
2	"Nasce Uma Estrela"	"2018-11-21"	"21:10:00"	2
8	"Monstros Fantásticos"	"2018-11-21"	"21:30:00"	4
7	"Overlord"	"2018-11-21"	"22:10:00"	5

Figura 71 - *Query* 2 Neo4j

```
-- Lista de filmes em exibição numa determinada data
SELECT F.idFilme, F.Nome, S.Data, S.Hora_Início, S.Nr_Sala FROM Filme AS F
    INNER JOIN Sessão AS S ON F.idFilme = S.Filme_id
    WHERE S.Data = '20181121'
    ORDER BY S.Hora_Início;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

idFilme	Nome	Data	Hora_Início	Nr_Sala
3	O Quebra-Nozes	2018-11-21	15:00:00	1
6	Belleville Cop - O Super Agente	2018-11-21	15:20:00	3
9	Viúvas	2018-11-21	16:30:00	2
5	A Rapariga Apanhada na Teia de Aranha	2018-11-21	17:40:00	5
4	Bohemian Rhapsody	2018-11-21	18:50:00	3
1	Coração Traiçoeiro	2018-11-21	19:10:00	1
2	Nasce Uma Estrela	2018-11-21	21:10:00	2
8	Monstros Fantásticos	2018-11-21	21:30:00	4
7	Overlord	2018-11-21	22:10:00	5

Figura 72 - *Query* 2 *SQL*

6.7.3 Qual o número da sala de uma determinada sessão?

```
1 MATCH (sa:Sala)-[:REALIZA_SESSAO]->(s:Sessao)
2 WHERE s.idSessao = 1
3 RETURN sa.Numero_Sala
```

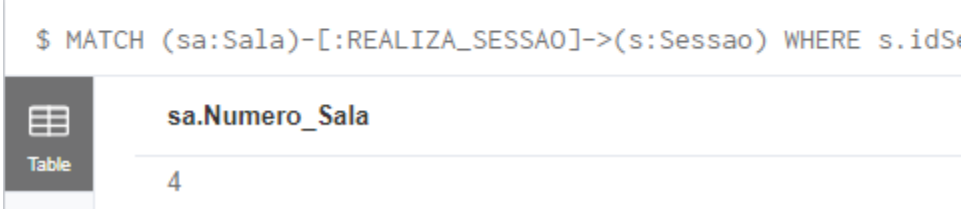

Figura 73 - *Query* 3 Neo4j

```
35  -- Numero da sala de uma determinada sessao
36  SELECT S.Nr_Sala FROM Sessão AS S
37      WHERE S.idSessão = 1;
38
```

Figura 74 - *Query* 3 *SQL*

6.7.4 Quem foi o funcionário que registou um determinado bilhete?

```
1 MATCH (f:Funcionario)-[:REGISTA]->(b:Bilhete)
2 WHERE b.idBilhete = 2
3 RETURN f.Nome
```

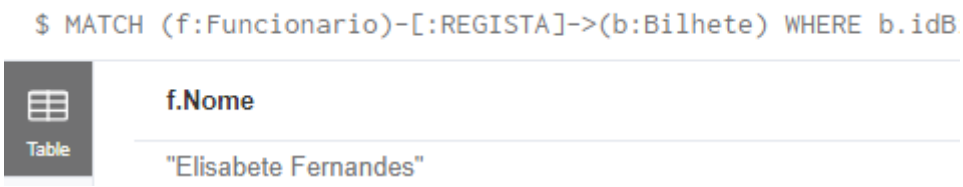

Figura 75 - *Query* 4 Neo4j

```
66  -- Nome do funcionário que registou um determinado bilhete
67  SELECT F.idFuncionário, F.Nome FROM Funcionário AS F
68      INNER JOIN Bilhete AS B ON F.idFuncionário = B.Funcionário_id
69          WHERE B.idBilhete = 2;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

idFuncionário	Nome
2	Elisabete Fernandes

Figura 76 - *Query* 4 *SQL*

6.7.5 Quantos bilhetes um determinado tipo de cliente comprou num determinado intervalo de datas?

```
1 MATCH (c:Cliente {Tipo: 'E'})-[:COMPRA]->(b:Bilhete)
2 WHERE b.Data_Emissao >= '2018-11-20' AND b.Data_Emissao < '2018-11-22'
3 RETURN COUNT(b)
```

```
$ MATCH (c:Cliente {Tipo: 'E'})-[:COMPRA]->(b:Bilhete) WHERE b.Data_Emissao >= '2018-11-
```

Table

COUNT(b)
59

Figura 77 - *Query* 5 Neo4j

```
79  -- Quantos bilhetes um determinado tipo de cliente comprou num determinado intervalo de datas;
80  SELECT COUNT(B.Cliente_NIF) FROM Bilhete AS B
81      INNER JOIN Cliente AS C ON B.Cliente_NIF = C.NIF
82          WHERE C.Tipo = 'E' AND (B.Data_Emissão BETWEEN '20181120' AND '20181121');
83
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

COUNT(B.Cliente_NIF)
59

Figura 78 - *Query* 5 *SQL*

6.7.6 Quais as sessões com mais bilhetes vendidos num determinado intervalo de datas?

```
MATCH (f:Filme)<-[:EXIBE_FILME]-(s:Sessao)-[:TEM]->(b:Bilhete)
WHERE s.Data >= '2018-11-20' AND s.Data <= '2018-11-21'
RETURN s.idSessao, f.Nome, s.Data, s.Hora_Inicio, COUNT(b)
ORDER BY COUNT(b) DESC
```

```
$ MATCH (f:Filme)<-[:EXIBE_FILME]-(s:Sessao)-[:TEM]->(b:Bilhete) WHERE s.Data >= '2018-11-20' AND s.Data <= '2018-11-21' …
```

Table | Text | Code

s.idSessao	f.Nome	s.Data	s.Hora_Inicio	COUNT(b)
1	"Homem de Ferro"	"2018-11-20"	"21:50:00"	75
10	"Monstros Fantásticos"	"2018-11-21"	"21:30:00"	36
9	"Overlord"	"2018-11-21"	"22:10:00"	22
2	"O Quebra-Nozes"	"2018-11-21"	"15:00:00"	19
8	"A Rapariga Apanhada na Teia de Aranha"	"2018-11-21"	"17:40:00"	14
6	"Belleville Cop - O Super Agente"	"2018-11-21"	"15:20:00"	12
7	"Bohemian Rhapsody"	"2018-11-21"	"18:50:00"	10
4	"Viúvas"	"2018-11-21"	"16:30:00"	8
3	"Coração Traiçoeiro"	"2018-11-21"	"19:10:00"	7
5	"Nasce Uma Estrela"	"2018-11-21"	"21:10:00"	4

Figura 79 - *Query* 6 Neo4j

```
85  -- Lista de sessões com mais bilhetes vendidos num determinado intervalo de datas
86  SELECT S.idSessão, F.Nome, S.Data, S.Hora_Início, COUNT(B.Sessão_id) FROM Sessão AS S
87      INNER JOIN Filme AS F ON S.Filme_id = F.idFilme
88          INNER JOIN Bilhete AS B ON S.idSessão = B.Sessão_id
89              WHERE S.Data BETWEEN '20181120' AND '20181121'
90              GROUP BY S.idSessão
91              ORDER BY COUNT(B.Sessão_id) DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

idSessão	Nome	Data	Hora_Início	COUNT(B.Sessão_id)
1	Homem de Ferro	2018-11-20	21:50:00	75
10	Monstros Fantásticos	2018-11-21	21:30:00	36
9	Overlord	2018-11-21	22:10:00	22
2	O Quebra-Nozes	2018-11-21	15:00:00	19
8	A Rapariga Apanhada na Teia de Aranha	2018-11-21	17:40:00	14
6	Belleville Cop - O Super Agente	2018-11-21	15:20:00	12
7	Bohemian Rhapsody	2018-11-21	18:50:00	10
4	Viúvas	2018-11-21	16:30:00	8
3	Coração Traiçoeiro	2018-11-21	19:10:00	7
5	Nasce Uma Estrela	2018-11-21	21:10:00	4

Figura 80 - *Query* 6 *SQL*

6.7.7 Quais são os 3 filmes em exibição com melhor classificação?

```
MATCH (s:Sessao {Data: '2018-11-25'})-[:EXIBE_FILME]->(f:Filme)
RETURN f.idFilme, f.Nome, f.Classificacao, s.Data
ORDER BY f.Classificacao DESC
LIMIT 3
```

$ MATCH (s:Sessao {Data: '2018-11-25'})-[:EXIBE_FILME]->(f:Filme) RETURN f.idFilme, f.Nome, f.Classificacao, s.Data ORDER…

Table | Text

f.idFilme	f.Nome	f.Classificacao	s.Data
16	"Vingadores: Guerra do Infinito"	8.5	"2018-11-25"
8	"Monstros Fantásticos"	7.1	"2018-11-25"
7	"Overlord"	7.1	"2018-11-25"

Figura 81 - *Query* 7 Neo4j

```
104  -- Lista de filmes em exibiçao com melhores classificaçoes, top3
105  SELECT F.idFilme, F.Nome, F.Classificação, S.Data FROM Filme AS F
106      INNER JOIN Sessão AS S ON F.idFilme = S.Filme_id
107      WHERE S.Data = '20181125'
108      ORDER BY F.Classificação DESC
109      LIMIT 3;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows

idFilme	Nome	Classificação	Data
16	Vingadores: Guerra do Infinito	8.5	2018-11-25
8	Monstros Fantásticos	7.1	2018-11-25
7	Overlord	7.1	2018-11-25

Figura 82 - *Query* 7 *SQL*

7. Conclusões e Trabalho Futuro

Com a finalização da primeira fase do projeto, e após ultrapassadas todas as etapas relativas ao mesmo, foi possível concluir que uma boa análise dos pré-requisitos é essencial, visto que esta vai influenciar a estruturação e implementação de todo o sistema de base de dados.

Depois de efetuado o levantamento e análise de requisitos, foram identificadas as diferentes entidades fundamentais, bem como os seus atributos e os relacionamentos existentes entre estas. Nesta fase inicial foi também bastante importante efetuar uma boa interpretação dos atributos, uma vez que foi necessário caracterizá-los (simples, compostos, multivalor ou derivados). Após esta caracterização dos atributos, foram determinamos os domínios destes e identificadas as chaves primárias, candidatas e alternativas correspondentes a cada entidade. Foi verificada também a eventual existência de redundâncias no modelo, em relação à qual se concluiu não existir qualquer tipo de problema. Assim, conseguiu-se elaborar um modelo conceptual para este projeto, que foi validado com as transações de utilizador. Nesta fase, surgiram algumas dúvidas que foram esclarecidas em conjunto com o professor.

Uma vez validado o modelo conceptual, prosseguiu-se com a implementação do modelo lógico. Nesta etapa foi essencial perceber como deveriam ser implementadas as diferentes entidades, os seus relacionamentos e os seus atributos. Uma das maiores preocupações que surgiram com a passagem do modelo conceptual para o modelo lógico, estava relacionada com a integridade dos dados e a normalização, uma vez que são estas que garantem a consistência dos dados e a ausência de redundância. Depois de implementado o modelo, este foi validado com as transações de utilizador.

Por fim, traduziu-se o modelo lógico para o modelo físico, onde foram analisadas e implementadas as diferentes transações de utilizador, já identificadas anteriormente no modelo lógico. Foi também importante fazer uma estimativa do espaço em disco que o sistema ocupa, para compreender se poderia existir algum problema de tamanho excessivo quer na implementação da Base de Dados, quer ao longo dos anos com a sua utilização. De seguida, definiram-se os diferentes perfis de utilizadores, assim como as diferentes regras de acesso para cada um deles. Por último, e com bastante preocupação relativamente às relações base, foram implementadas as restrições fundamentais para o sistema.

Na segunda fase do projeto foram identificados as entidades e os seus atributos do sistema relacional, de maneira a compreender o comportamento da base de dados. Assim, conseguimos interpretar melhor o problema de modo a desenvolver o novo sistema não relacional que fosse equivalente ao esquema relacional implementado anteriormente.

Todo o processo de migração dos dados do SGB relacional para o não relacional, foi explicado e definido utilizando os ficheiros ".csv" e os comandos da linguagem de interrogação do Neo4j (*cypher*). Foi nesta fase que foram evidenciadas algumas dificuldades em garantir a integridade dos dados, pois no modelo não relacional estas são garantidas pelos relacionamentos.

Por fim, foram implementadas e exemplificadas algumas queries, na linguagem especifica do Neo4j, de modo a demonstrar a execução do sistema, equiparando sempre com os resultados da mesma *query* no sistema relacional.

Com a realização deste trabalho, teve-se a perceção de que um Sistema de Base de Dados não relacional é muito eficaz e rápido na organização de dados e ainda muito flexível.

Avaliando a nossa SBD, chegamos à conclusão que um sistema relacional estaria mais adaptado devido ao grande número de transações comparativamente ao número de pesquisas que vão ser solicitadas à nossa SBD. No entanto, a realização deste projeto permitiu consolidar e aplicar os conhecimentos adquiridos na unidade curricular, bem como identificar algumas lacunas que terão de ser corrigidas futuramente.

8. Referências Bibliográficas

[1] Thomas Connolly, Carolyn Begg 2005. Database Systems: A Practical Approach to Design, Implementation, and Management, 4ª edição, Adisson Wesley.

[2] Feliz Gouveia 2014. Fundamentos de Bases de Dados, FCA

Lista de Siglas e Acrónimos

BD Base de Dados

SBD Sistema de Base de Dados

SGBD Sistema de Gestão de Base de Dados

SBDR Sistema de Base de Dados Relacional

NIF Número de Identificação Fiscal

ER Entidade-Relacionamento

id Número de Identificação

admin Administrador

Anexos

I. Anexo 1 – Script de criação da BD

```
-- -----------------------------------------------------
-- Table `Cinema`.`Cliente`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Cinema`.`Cliente` ;

CREATE TABLE IF NOT EXISTS `Cinema`.`Cliente` (
  `NIF` INT NOT NULL,
  `Nome` VARCHAR(45) NULL,
  `Tipo` CHAR NOT NULL,
  PRIMARY KEY (`NIF`))
ENGINE = InnoDB;
```

Figura 83 – Tabela Cliente

```
-- -----------------------------------------------------
-- Table `Cinema`.`Funcionário`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Cinema`.`Funcionário` ;

CREATE TABLE IF NOT EXISTS `Cinema`.`Funcionário` (
  `idFuncionário` INT NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idFuncionário`))
ENGINE = InnoDB;
```

Figura 84 – Tabela Funcionário

```
-- -----------------------------------------------------
-- Table `Cinema`.`Filme`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Cinema`.`Filme` ;

CREATE TABLE IF NOT EXISTS `Cinema`.`Filme` (
  `idFilme` INT NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  `Descrição` TEXT NULL,
  `Classificação` DECIMAL(2,1) NULL,
  PRIMARY KEY (`idFilme`))
ENGINE = InnoDB;
```

Figura 85 – Tabela Filme

```
-- -----------------------------------------------------
-- Table `Cinema`.`Sala`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Cinema`.`Sala` ;

CREATE TABLE IF NOT EXISTS `Cinema`.`Sala` (
  `Número_Sala` INT NOT NULL,
  PRIMARY KEY (`Número_Sala`))
ENGINE = InnoDB;
```

Figura 86 – Tabela Sala

```
-- -----------------------------------------------------
-- Table `Cinema`.`Lugar`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Cinema`.`Lugar` ;

CREATE TABLE IF NOT EXISTS `Cinema`.`Lugar` (
  `idLugar` INT NOT NULL,
  `Nr_Sala` INT NOT NULL,
  `Número_Cadeira` INT NOT NULL,
  `Fila` CHAR NOT NULL,
  INDEX `fk_Lugar_Sala_idx` (`Nr_Sala` ASC),
  PRIMARY KEY (`idLugar`, `Nr_Sala`),
  CONSTRAINT `fk_Lugar_Sala`
    FOREIGN KEY (`Nr_Sala`)
    REFERENCES `Cinema`.`Sala` (`Número_Sala`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 87 – Tabela Lugar

```sql
-- -----------------------------------------------------
-- Table `Cinema`.`Sessão`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Cinema`.`Sessão` ;

CREATE TABLE IF NOT EXISTS `Cinema`.`Sessão` (
  `idSessão` INT NOT NULL,
  `Preço_Base` DECIMAL(3,2) NOT NULL,
  `Data` DATE NOT NULL,
  `Hora_Início` TIME NOT NULL,
  `Duração` TIME NOT NULL,
  `Nr_Sala` INT NOT NULL,
  `Filme_id` INT NOT NULL,
  PRIMARY KEY (`idSessão`),
  INDEX `fk_Sessão_Sala1_idx` (`Nr_Sala` ASC),
  INDEX `fk_Sessão_Filme1_idx` (`Filme_id` ASC),
  CONSTRAINT `fk_Sessão_Sala1`
    FOREIGN KEY (`Nr_Sala`)
    REFERENCES `Cinema`.`Sala` (`Número_Sala`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Sessão_Filme1`
    FOREIGN KEY (`Filme_id`)
    REFERENCES `Cinema`.`Filme` (`idFilme`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 88 – Tabela Sessão

```
-- -----------------------------------------------------
-- Table `Cinema`.`Bilhete`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Cinema`.`Bilhete` ;

CREATE TABLE IF NOT EXISTS `Cinema`.`Bilhete` (
  `idBilhete` INT NOT NULL,
  `Data_Emissão` DATE NOT NULL,
  `Lugar` INT NOT NULL,
  `Preço` DECIMAL(3,2) NOT NULL,
  `Cliente_NIF` INT NOT NULL,
  `Sessão_id` INT NOT NULL,
  `Funcionário_id` INT NOT NULL,
  PRIMARY KEY (`idBilhete`),
  INDEX `fk_Bilhete_Cliente1_idx` (`Cliente_NIF` ASC),
  INDEX `fk_Bilhete_Sessão1_idx` (`Sessão_id` ASC),
  INDEX `fk_Bilhete_Funcionário1_idx` (`Funcionário_id` ASC),
  CONSTRAINT `fk_Bilhete_Cliente1`
    FOREIGN KEY (`Cliente_NIF`)
    REFERENCES `Cinema`.`Cliente` (`NIF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Bilhete_Sessão1`
    FOREIGN KEY (`Sessão_id`)
    REFERENCES `Cinema`.`Sessão` (`idSessão`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Bilhete_Funcionário1`
    FOREIGN KEY (`Funcionário_id`)
    REFERENCES `Cinema`.`Funcionário` (`idFuncionário`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 89 – Tabela Bilhete

II. Anexo 2 – Povoamento da BD

```
INSERT INTO Filme
    (idFilme, Nome, Descrição, Classificação)
    VALUES
        (1, 'Coração Traiçoeiro', 'Aos 30 anos, Lenny é um boémio irredimível que vive às custas do pai, um famoso cardiologista. David, paciente
        (2, 'Nasce Uma Estrela', 'Apesar de sobreviver à custa de um ordenado miserável como empregada de mesa, Ally nunca abandonou o sonho de
        (3, 'O Quebra-Nozes', 'A jovem Clara perde a chave mágica capaz de abrir a caixa de um presente de valor incalculável que lhe foi oferec
        (4, 'Bohemian Rhapsody', 'Um filme autobiográfico sobre os Queen, uma das maiores bandas de rock de sempre.', 8.4),
        (5, 'A Rapariga Apanhada na Teia de Aranha', 'Mikael Blomkvist, jornalista e fundador da revista "Millenium", e Lisbeth Salander, a "hac
        (6, 'Belleville Cop - O Super Agente', 'Quando Roland, um velho amigo que há anos vivia em Miami é encontrado assassinado por um grupo l
        (7, 'Overlord', 'Nos dias anteriores às operações de desembarques na Normandia, que viria a acontecer a 6 de Junho de 1944, um grupo de
        (8, 'Monstros Fantásticos', 'Depois das aventuras vividas em "Monstros Fantásticos e Onde Encontrá-los", o excêntrico magizoologista New
        (9, 'Viúvas', 'Veronica, Alice, Linda e Belle são diferentes em tudo, excepto numa coisa fundamental: herdaram dos falecidos maridos uma
        (10, 'Homem de Ferro', 'MELHOR FILME DESTA BD -- DISPENSA DESCRIÇÕES', 9.9),
        (11, 'Venom', 'Apostado em retomar a sua carreira jornalística, Eddie Brock investiga a Life Foundation, uma empresa de genética suspeit
        (12, 'Hotel Transylvania 3', 'A família do conde Drácula entende que ele deve fazer uma pausa no seu trabalho no hotel e marca-lhe uma v
        (13, 'Pedro e Inês', 'Uma conhecida história de amor contada em três versões e séculos distintos: no passado, no presente e no futuro. P
        (14, 'Johnny English Volta a Atacar', 'Esta nova aventura tem início quando um ataque cibernético revela a identidade de todos os agente
        (15, 'The Nun - A Freira Maldita', 'Roménia, 1952. Um padre veterano e uma freira novata são enviados pelo Vaticano para investigarem o
        (16, 'Vingadores: Guerra do Infinito', 'O planeta Terra enfrenta mais um inimigo: Thanos, um tirano intergaláctico que ali chega com o ol
```

Figura 90 – Povoamento da Tabela 'Filme' (pequeno extrato a título de exemplo)

```
INSERT INTO Sala
    (Número_Sala)
    VALUES
        (1), -- 63 lugares
        (2), -- 81 lugares
        (3), -- 72 lugares
        (4), -- 90 lugares
        (5); -- 81 lugares
```

Figura 91 – Povoamento da Tabela 'Sala'

```
INSERT INTO Lugar
    (idLugar, Nr_Sala, Número_Cadeira, Fila)
    VALUES
        -- Sala Nr 1
        (1, 1, 1, 'A'),
        (2, 1, 2, 'A'),
        (3, 1, 3, 'A'),
        (4, 1, 4, 'A'),
        (5, 1, 5, 'A'),
        (6, 1, 6, 'A'),
        (7, 1, 7, 'A'),
        (8, 1, 8, 'A'),
        (9, 1, 9, 'A'),
        (10, 1, 1, 'B'),
        (11, 1, 2, 'B'),
        (12, 1, 3, 'B'),
        (13, 1, 4, 'B'),
        (14, 1, 5, 'B'),
        (15, 1, 6, 'B'),
```

Figura 92 – Povoamento da Tabela 'Lugar' (pequeno extrato a título de exemplo)

```
INSERT INTO Cliente
    (NIF, Nome, Tipo) -- Nº de bilhetes associados
    VALUES
        ('429275511', 'Armstrong Strugnell', 'E'), -- 1
        ('229492717', 'Ottilie Ebsworth', 'N'), -- 2
        ('555824897', 'Sayers Nugent', 'N'), -- 1
        ('133792483', 'Sigmund Remon', 'N'), -- 1
        ('921433330', 'Ferris Fairlaw', 'N'), -- 1
        ('642852164', 'Hamnet Joly', 'E'), -- 1
        ('826042553', 'Collette Coysh', 'N'), -- 1
        ('506180454', 'Clementius Cottey', 'E'), -- 1
        ('387953652', 'Munmro Bundey', 'N'), -- 1
        ('106233117', 'Mathe Ducaen', 'N'), -- 1
        ('815905445', 'Shandee Campaigne', 'E'), -- 1
        ('269543607', 'Even Ficken', 'E'), -- 1
        ('039534568', 'Elsworth Wadsworth', 'N'), -- 1
        ('904172082', 'Edgar Sharrocks', 'N'), -- 1
        ('389461330', 'Kit Walklate', 'N'), -- 1
        ('505898510', 'Cam Sorley', 'N'), -- 1
```

Figura 93 – Povoamento da Tabela 'Cliente' (pequeno extrato a título de exemplo)

```
INSERT INTO Funcionário
    (idFuncionário, Nome)
    VALUES
        (1, 'Armindo Rebola'),
        (2, 'Elisabete Fernandes'),
        (3, 'Mónica Amorim'),
        (4, 'Sílvia Cerqueira');
```

Figura 94 – Povoamento da Tabela 'Funcionário'

```
INSERT INTO Bilhete
    (idBilhete, Data_Emissão, Lugar, Preço, Cliente_NIF, Sessão_id, Funcionário_id)
    VALUES
        -- 20/11/2018
        (1, '20181120', 1, 3.45, '100000001', 1, 1),
        (2, '20181120', 2, 3.45, '111111111', 1, 2),
        (3, '20181120', 3, 3.45, '222222222', 1, 1),
        (4, '20181120', 4, 3.45, '333333333', 1, 2),
        (5, '20181120', 5, 4.6, '444444444', 1, 1),
        (6, '20181120', 6, 3.45, '429275511', 1, 2),
        (7, '20181120', 7, 4.6, '229492717', 1, 1),
        (8, '20181120', 8, 4.6, '555824897', 1, 2),
        (9, '20181120', 9, 4.6, '133792483', 1, 1),
        (10, '20181120', 10, 4.6, '921433330', 1, 1),
        (11, '20181120', 11, 3.45, '642852164', 1, 2),
        (12, '20181120', 12, 4.6, '826042553', 1, 1),
        (13, '20181120', 13, 3.45, '506180454', 1, 2),
        (14, '20181120', 14, 4.6, '387953652', 1, 1),
        (15, '20181120', 15, 4.6, '106233117', 1, 1),
        (16, '20181120', 16, 3.45, '815905445', 1, 1),
```

Figura 95 – Povoamento da Tabela 'Bilhete' (pequeno extrato a título de exemplo)