



Universidade do Minho

Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Lectivo de 2018/2019

FNAC

João Teixeira A85504,

José Ferreira A83683,

Maria Silva A83840,

Miguel Solino A86435

Janeiro, 2020

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

FNAC

João Teixeira A85504,
José Ferreira A83683,
Maria Silva A83840,
Miguel Solino A86435

Janeiro, 2020

Resumo

Este relatório foi desenvolvido no âmbito do desenvolvimento de uma Base de Dados para a empresa FNAC, nomeadamente na área da venda de produtos online. Este software tem como objetivo melhorar a qualidade dos serviços prestados pela empresa.

Ao longo deste relatório são apresentadas todas as etapas tomadas para o desenvolvimento desta base de dados.

Em primeiro lugar foi realizado um levantamento de requisitos da base de dados. Para tal, foi estudado todas as funcionalidades que era necessário implementar.

Em seguida, recorrendo aos requisitos obtidos, foi iniciada a Modelação conceptual do sistema de base de dados. Após verificar que todos os requisitos eram suportados, este modelo foi implementado fisicamente após ser convertido num modelo lógico. O modelo físico foi implementado com recurso a MySQL e MySQL Workbench.

A fim do produto final ser mais flexível, mais eficiente e a visualização dos dados contidos na base de dados ser mais visualmente apelativa, a base de dados gerada em MySQL foi convertida para Neo4j.

Concluindo este processo, após a aprovação do cliente e a implementação da base de dados, foi dado como concluído o projeto.

Área de Aplicação: Desenho e arquitetura de Sistemas de Bases de Dados responsável pela gestão de vendas online.

Palavras-Chave: Base de dados relacional, Base de dados não relacional, Análise de Requisitos, Modelo Conceptual, Modelo Lógico, Modelo Físico, MySQL Workbench, MySQL, SQL, NoSQL, Neo4j

Índice

Resumo	i
Índice	ii
Índice de Figuras	v
Índice de Tabelas	viii
1. Introdução	1
1.1. Contextualização	1
1.2. Apresentação do Caso de Estudo	1
1.3. Motivação e Objetivos	2
1.4. Estrutura do Relatório	2
2. Levantamento e análise de requisitos	3
2.1. Método de levantamento e de análise de requisitos adotados	3
2.2. Especificação de Requisitos	3
2.2.1 Requisitos de descrição	3
2.2.2 Requisitos de exploração	4
2.2.3 Requisitos de controlo	4
2.3. Análise geral dos requisitos	5
2.4. Caracterização dos Perfis de Utilização	5
3. Modelação conceptual	6
3.1. Apresentação da abordagem de modelação realizada	6
3.2. Identificação e caracterização das entidades	6
3.2.1 Dicionário de dados das entidades	7
3.3. Identificação e caracterização dos relacionamentos	8
Cliente e Compra	8
Compra e Artigo	9
Artigo e Stock	10
Artigo e Filme	11
Artigo e Livro	12
Artigo e Jogo	12
Artigo e Musica	13
3.3.1 Dicionário de relacionamento	14
3.4. Identificação e Associação de Atributos aos Tipos de Entidades e de Relacionamentos	15

3.5. Determinação das chaves candidatas, chaves primárias e chaves alternadas	18
3.6. Detalhe ou generalização de entidades	19
3.7. Apresentação e explicação do diagrama ER	19
3.8. Validação do modelo de dados com o utilizador	20
4. Modelo lógico	21
4.1. Construção e validação do modelo de dados lógico	21
4.1.1 Entidades fortes	21
4.1.2 Relacionamentos de um para muitos (1:N)	22
4.1.3 Relacionamentos de muitos para muitos (N:M)	23
4.1.4 Relacionamentos hierárquico	23
4.2. Desenho do modelo lógico	24
4.3. Dependências Funcionais	24
4.4. Validação do modelo com interrogações do utilizador	25
4.5. Validação do modelo com as transações estabelecidas	26
4.6. Revisão do modelo lógico com o utilizador	29
5. Implementação Física	30
5.1. Seleção do sistema de gestão de bases de dados	30
5.2. Tradução do esquema lógico para o sistema de gestão de bases de dados escolhido em SQL	30
5.2.1 Desenho das relações base	30
5.2.2 Desenho das restrições	35
5.3. Tradução das interrogações do utilizador para SQL	40
5.4. Estimativa do espaço em disco da base de dados e taxa de crescimento anual	41
5.5. Definição e caracterização das vistas de utilização em SQL	44
5.6. Revisão do sistema implementado com o utilizador	46
6. Projeto de um Sistema de Base de Dados não Relacional	47
6.1. Utilização de um sistema NoSQL	47
6.1.1 Neo4j	47
6.2. Objetivos da Base de Dados	48
6.3. Identificação das Queries realizadas sobre o sistema	48
6.4. Estrutura base para o sistema de dados NoSQL	49
6.4.1 Objetos de dados no sistema NoSQL	49
6.5. Migração de Dados	50
6.5.1 Exportação dos Dados	50
6.5.2 Criação dos Nodos	51
6.5.3 Criação dos Relacionamentos	53
6.6. Resolução das Queries Propostas	56
6.7. Base de Dados em Neo4j	61

7. Conclusões e Trabalho Futuro	63
8. Referências	64
9. Lista de Siglas e Acrónimos	65
10. Anexos	66
10.1. Anexo 1 – Script Completo de Criação	66
10.2. Anexo 2 – Script Parcial do Povoamento	70
10.3. Anexo 3 – Queries em SQL	76
10.4. Anexo 4 – Exportação de Dados	82

Índice de Figuras

Figura 1 - Relacionamento Cliente e Compra	9
Figura 2 - Relacionamento Compra e Artigo	10
Figura 3 - Relacionamento Artigo e Stock	11
Figura 4 - Relacionamento Artigo e Filme	11
Figura 5 - Relacionamento Artigo e Livro	12
Figura 6 - Relacionamento Artigo e Jogo	13
Figura 7 - Relacionamento Artigo e Musica	14
Figura 8 - Modelo Conceptual dos dados	19
Figura 9 - Tabelas alteradas pela transação Registrar Cliente	26
Figura 10 - Tabelas alteradas pela transação Registrar Artigo	27
Figura 11 - Tabelas alteradas pela transação Alterar preço de artigo	28
Figura 12 - Tabelas alteradas pela transação Alterar Stock	28
Figura 13 - Tabelas alteradas pela transação Registrar Compra	29
Figura 14 - Relação Cliente	31
Figura 15 - Relação Compra	31
Figura 16 - Relação Artigo	32
Figura 17 - Relação Stock	32
Figura 18 - Relação Filme	33
Figura 19 - Relação Livro	33
Figura 20 - Relação Musica	34
Figura 21 - Relação Jogo	34
Figura 22 - Relação Compra_de_X_Artigos	35
Figura 23 - Criação da Tabela Cliente	35
Figura 24 - Criação da Tabela Compra	36
Figura 25 - Criação da Tabela Artigo	36
Figura 26 - Criação da Tabela Stock	37
Figura 27 - Criação da Tabela Filme	37
Figura 28 - Criação da Tabela Livro	38
Figura 29 - Criação da Tabela Jogo	38
Figura 30 - Criação da Tabela Musica	39
Figura 31 - Criação da Tabela Compra_de_X_Artigos	39

Figura 32 - Resolução do requisito de exploração 14	40
Figura 33 - Resolução do requisito de exploração 20	40
Figura 34 - Resolução do requisito de exploração 21	41
Figura 35 - Resolução do requisito de exploração 22	41
Figura 36 - View Filmes	45
Figura 37 - View Jogos	45
Figura 38 - View Livros	45
Figura 39 - View Musica	45
Figura 40 - View Top 3 Artigos mais vendidos	46
Figura 41 - View Vendas Autor	46
Figura 42 - Estrutura adotada para o sistema NoSQL	49
Figura 43 - Script MySQL para exportação	50
Figura 44 - Botão Exportação/Importação	50
Figura 45 - Campo de inserção do nome	50
Figura 46 - Formato dos ficheiros de exportação	51
Figura 47 - Ficheiros Exportados	51
Figura 48 - Criação do Nodo Cliente	51
Figura 49 - Criação do Nodo Compra	52
Figura 50 - Criação do Nodo Loja	52
Figura 51 - Criação do Nodo Filme	52
Figura 52 - Criação do Nodo Livro	52
Figura 53 - Criação do Nodo Jogo	53
Figura 54 - Criação do Nodo Musica	53
Figura 55 - Criação do Relacionamento Cliente - Compra	53
Figura 56 - Criação do Relacionamento Compra - Filme	54
Figura 57 - Criação do Relacionamento Compra - Livro	54
Figura 58 - Criação do Relacionamento Compra - Jogo	54
Figura 59 - Criação do Relacionamento Compra - Musica	54
Figura 60 - Criação do Relacionamento Filme - Loja	55
Figura 61 - Criação do Relacionamento Livro - Loja	55
Figura 62 - Criação do Relacionamento Jogo - Loja	55
Figura 63 - Criação do Relacionamento Musica - Loja	56
Figura 64 - Requisito de exploração 11 em Cypher	56
Figura 65 - Requisito de exploração 12 em Cypher	56
Figura 66 - Requisito de exploração 13 em Cypher	57
Figura 67 - Requisito de exploração 14 em Cypher	57
Figura 68 - Requisito de exploração 15 em Cypher	57
Figura 69 - Requisito de exploração 16 em Cypher	57
Figura 70 - Requisito de exploração 17 em Cypher	58
Figura 71 - Requisito de exploração 18 em Cypher	58

Figura 72 - Requisito de exploração 19 em Cypher	59
Figura 73 - Requisito de exploração 20 em Cypher	59
Figura 74 - Requisito de exploração 21 em Cypher	59
Figura 75 - Requisito de exploração 22 em Cypher	60
Figura 76 - Requisito de exploração 23 em Cypher	60
Figura 77 - Requisito de exploração 24 em Cypher	60
Figura 78 - Requisito de exploração 25 em Cypher	60
Figura 79 - Requisito de exploração 26 em Cypher	61
Figura 80 - Número de dados no grafo	61
Figura 81 - Grafo da base de dados	62

Índice de Tabelas

Tabela 1 - Dicionário de dados das entidades	8
Tabela 2 - Dicionário de relacionamento	14
Tabela 3 - Dicionário de dados dos atributos das entidades e relacionamentos	17
Tabela 4 - Representação da entidade Cliente	21
Tabela 5 - Representação da entidade Compra	22
Tabela 6 - Representação da entidade Artigo	22
Tabela 7 - Representação da entidade Stock	22
Tabela 8 - Modelo Lógico	24
Tabela 9 - Espaço ocupado no disco por cada tipo de dados	42
Tabela 10 - Espaço ocupado no disco por Cliente	42
Tabela 11 - Espaço ocupado no disco por Compra	42
Tabela 12 - Espaço ocupado no disco por Artigo	42
Tabela 13 - Espaço ocupado no disco por Stock	43
Tabela 14 - Espaço ocupado no disco por Filme	43
Tabela 15 - Espaço ocupado no disco por Jogo	43
Tabela 16 - Espaço ocupado no disco por Livro	43
Tabela 17 - Espaço ocupado no disco por Compra_de_X_Artigos	44
Tabela 18 - Espaço ocupado no disco pela base de dados	44

1. Introdução

1.1. Contextualização

A FNAC portuguesa recebe diariamente críticas e reclamações do Marketplace que têm disponível na loja online devido a alguns produtos não chegarem ou o tempo de entrega ser muito extenso. Com o intuito de melhorar a qualidade do serviço, a parte administrativa que trata da loja online decidiu reunir-se afim de encontrar uma solução para este problema.

Como o problema se focava no Marketplace, a solução que decidiram foi separar as duas lojas. Para aplicar esta solução, seria necessário refazer a base de dados para conseguir uma melhor gestão dos produtos disponíveis fisicamente nas lojas e separar os produtos do Marketplace. Focando-se, assim, numa nova base de dados que apenas contém os produtos fora do Marketplace.

1.2. Apresentação do Caso de Estudo

A FNAC é uma empresa multinacional com um volume de vendas astronómico.

Como tal, seria de esperar que os seus serviços online estivessem ao nível de tantas outras empresas que populam a internet com entregas que cumprem os tempos indicados e que de facto entregam o produto comprado.

No entanto, com a introdução do Marketplace, a FNAC começou a receber cada vez mais ondas de clientes dessatisfeitos com os atrasos e a má qualidade de serviço o que levou a uma fuga dos clientes habituais para outros serviços.

Para que a FNAC volte a estar na crista da onda a nível da qualidade e versatilidade dos serviços é necessária uma separação do Marketplace da sua loja normal.

Assim, tendo em conta o apresentado anteriormente, é necessário desenvolver uma nova base de dados que suporte todo o sistema.

1.3. Motivação e Objetivos

Visto que a qualidade do serviço tem vindo a decair ao longo do tempo, a FNAC decidiu reestruturar o funcionamento da base de dados.

O objetivo deste projeto é criar uma base de dados que seja mais eficiente e que esteja melhor adaptada aos requisitos a fim de melhorar a qualidade de serviço proporcionado pela loja online FNAC.

1.4. Estrutura do Relatório

Primeiramente, iremos explicar como procedemos para realizar o levantamento de requisitos no terreno, assim como as conclusões retiradas desse mesmo levantamento. Para tal, apresentamos e analisamos os vários tipos de requisitos obtidos ao longo das várias entrevistas realizadas.

Em seguida, procedemos a uma modelação conceptual do modelo de base de dados que segue o conjunto de requisitos definidos no passo anterior.

Finalmente, iremos proceder a propor duas implementações físicas da base de dados. Sendo que a primeira é uma base de dados relacional e a segunda é uma base de dados não relacional.

2. Levantamento e análise de requisitos

2.1. Método de levantamento e de análise de requisitos adotados

Sendo que precisávamos de informações para conhecermos os requisitos e os objetivos necessários para a nova base de dados reunimo-nos com o representante da parte administrativa que estava encarregue da nova loja. Para além desta reunião, foram agendadas várias reuniões para acompanhamento do projeto e esclarecer dúvidas que surgiram com o desenvolvimento do projeto e não foram abordadas na primeira reunião.

2.2. Especificação de Requisitos

Tendo em foco o que foi decidido na primeira reunião, foram identificados os seguintes requisitos:

2.2.1 Requisitos de descrição

1. Os clientes devem ser registados sendo-lhes atribuído um ID sequencial e data de subscrição pela aplicação. Deve ser fornecido um nome, data de nascimento, distrito e telemóvel.
2. As compras devem ser registadas com um ID de compra, atribuído pela aplicação, com o ID do cliente (sendo que cada compra só pode ser registada com um cliente), montante, loja (de onde o stock foi diminuído) e data da compra.
3. As compras existentes estão associadas aos ID's de artigos que foram comprados e a quantidade dos mesmos.
4. Os artigos devem ser registados sendo-lhes atribuído um ID sequencial pela aplicação, uma classificação nula inicial, título, tipo, preço, ano de lançamento e descrição.
5. Os artigos existentes podem ser de quatro tipos diferentes: Filme, Livro, Jogo e Música.

6. Os artigos que são filmes têm registado em adição a sua duração, realizador e género.
7. Os artigos que são livros têm registado em adição o número de páginas, o seu autor, género e editora.
8. Os artigos que são jogos têm registado em adição a plataforma, número de jogadores máximo, género, idade mínima e a empresa publicador.
9. Os artigos que são músicas têm registado em adição o formato, o artista e o género musical.
10. Os stocks dos artigos devem ter a si relacionado o ID do artigo, a quantidade disponível, a loja e distrito.

2.2.2 Requisitos de exploração

11. Obter o número de clientes nas lojas todas
12. Obter o número de clientes num determinado distrito
13. Consultar os artigos por tipo
14. Consultar os artigos que não tem stock
15. Obter o top 3 dos clientes com mais artigos comprados
16. Obter o top 3 dos clientes que mais dinheiro gastaram
17. Obter o top 3 artigos mais vendidos
18. Obter o top 3 de filmes mais vendidos
19. Obter o top 3 de livros mais vendidos
20. Obter o top 3 de músicas mais vendidas
21. Consultar livros de um autor específico
22. Verificar se o cliente tem idade mínima para o jogo que quer comprar
23. Verificar que quantidade e montante um autor específico vendeu
24. Consultar quanto foi vendido em uma loja especifica num ano específico
25. Consultar quanto foi vendido em cada mês num ano específico
26. Verificar a existência de stock de um certo artigo no distrito de um certo cliente
27. Fazer análise das vendas de um ano (loja que lucrou mais, qual o lucro e a média entre lojas desse ano)

2.2.3 Requisitos de controlo

28. O administrador tem permissões para consultar, inserir e gerir os artigos e os seus respetivos stocks, consultar os clientes e consultar as compras.
29. Os clientes têm permissões para ver os produtos à venda, consultar o seu histórico de compras e consultar e alterar os seus dados pessoais.

2.3. Análise geral dos requisitos

Com os requisitos já definidos e elaborados, decidimos reunir novamente com o representante para podermos apresentar e explicar cada um deles. Nesta reunião foi então aprovado e permitiu-nos prosseguir com o trabalho, começando então por construir o Modelo Conceptual mais adequado para os requisitos apresentados.

2.4. Caraterização dos Perfis de Utilização

Tendo em conta os requisitos que apresentamos, é possível identificar dois perfis de utilização: o de cliente e o de administrador.

Um cliente pode apenas ver os artigos em stock, comprar artigos e ver/alterar os seus dados pessoais no seu perfil.

O administrador pode inserir e gerir artigos, atualizar stocks, consultar os clientes registados e os respetivos históricos de compras.

3. Modelação conceptual

3.1. Apresentação da abordagem de modelação realizada

Para realizar este projeto optamos pela utilização de uma vista de análise e desenvolvimento. Esta escolha foi tomada devido à baixa complexidade do projeto em questão que leva a que mais do que uma vista seja desnecessária.

3.2. Identificação e caracterização das entidades

Para iniciar a construção do Modelo conceptual primeiro teremos que identificar as entidades do problema, tendo em atenção para não existir uma seleção errada das mesmas, pois existem objetos com grande importância para o modelo, mas que não são considerados entidades.

Com isso, foram então decididas as seguintes entidades, seguidas cada uma do seu significado.

Cliente

Este é um termo que descreve todos os clientes que estão registados na base de dados da loja. Cada um deles tem os seus atributos próprios sendo assim possível distinguir cada um deles (além do ID a que está associado).

Artigo

Este é o termo geral que descreve todos os artigos registados na base de dados. Também tem atributos próprios sendo possível diferenciar cada artigo. Esta entidade pode ser comprada pelos clientes e gerida pelo administrador.

Stock

Entidade que indica quais artigos existem em diferentes lojas dos vários distritos. Tem a sua importância para que se saiba onde o cliente se precisa dirigir caso queira fazer levantamento em loja física.

Compra

Termo geral para todas as compras realizadas pelos clientes. Cada compra tem todas as informações relevantes para o controlo das vendas na parte administrativa e visíveis no histórico do cliente que a realizou.

Filme

Termo que descreve o tipo filme dos artigos. Este tem duração do filme, realizador e género como caraterísticas adicionais.

Livro

Termo que descreve o tipo livro dos artigos. Este tem número de páginas, autor, género e editora como caraterísticas adicionais.

Jogo

Termo que descreve o tipo jogo dos artigos. Este tem plataforma, idade mínima, número máximo de jogadores, género e publicadora como caraterísticas adicionais.

Música

Termo que descreve o tipo música dos artigos. Este tem formato, género e artista como caraterísticas adicionais.

3.2.1 Dicionário de dados das entidades

Entidade	Descrição	Alcunha	Ocorrência
Cliente	Entidade que compra os artigos da loja	-	O cliente é muito fundamental sendo que é quem faz uso da loja.
Artigo	Entidade que é comprada pelos clientes	Produto	O artigo pode se dizer que é a base do funcionamento da loja.
Stock	Entidade que indica todas as informações relativamente ao stock dos artigos nas lojas	-	O Stock indica se certo artigo pode ou não ser vendido.
Compra	Entidade que contém todos as informações relativamente a uma compra de um ou vários artigos	Venda	As compras são fundamentais para indicar como está a correr o funcionamento das lojas
Filme	Entidade que contém	-	O filme indica se certo

	as informações adicionais de um artigo do tipo Filme		artigo é do tipo filme e fornece as informações relativamente ao conteúdo
Livro	Entidade que contém as informações adicionais de um artigo do tipo Livro	-	O livro indica se certo artigo é do tipo filme e fornece as informações relativamente ao conteúdo
Jogo	Entidade que contém as informações adicionais de um artigo do tipo Jogo	-	O jogo indica se certo artigo é do tipo filme e fornece as informações relativamente ao conteúdo
Música	Entidade que contém as informações adicionais de um artigo do tipo Música	-	A música indica se certo artigo é do tipo filme e fornece as informações relativamente ao conteúdo

Tabela 1 - Dicionário de dados das entidades

3.3. Identificação e caracterização dos relacionamentos

Nesta secção tivemos como foco a identificação dos relacionamentos que existe entre as entidades definidas anteriormente, sendo apresentado também uma pequena informação adicional para cada.

Cliente e Compra

O relacionamento “efetua” entre as entidades *Cliente* e *Compra* caracteriza-se por uma cardinalidade de 1 para N, devido a um cliente poder efetuar várias compras mas uma compra só corresponder a um único cliente.

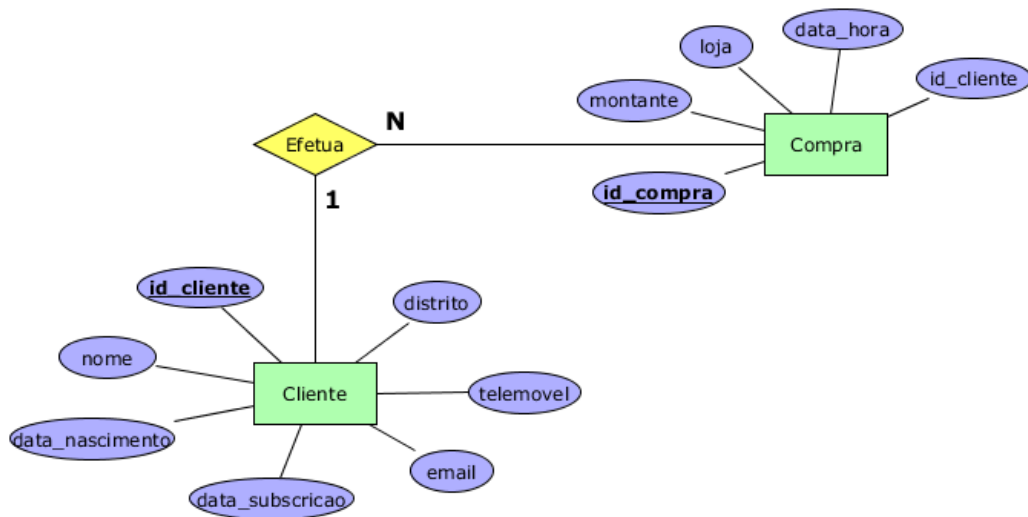


Figura 1 - Relacionamento Cliente e Compra

Compra e Artigo

O relacionamento “contém” entre as entidades *Compra* e *Artigo* caracteriza-se por uma cardinalidade de N par na, pois uma compra pode conter vários artigos e os artigos podem ter várias compras feitas.

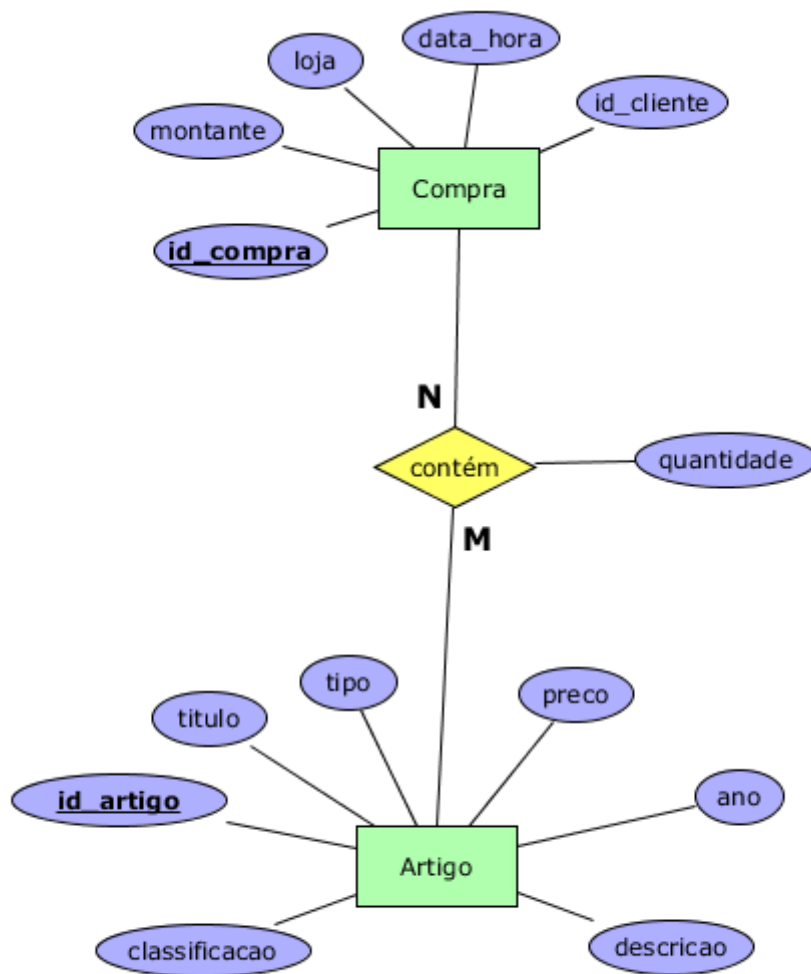


Figura 2 - Relacionamento Compra e Artigo

Artigo e Stock

O relacionamento “tem” entre as entidades *Artigo* e *Stock* caracteriza-se por uma cardinalidade de 1 para N, uma vez que um artigo tem vários stocks em lojas diferentes, mas um stock pode só corresponder a um artigo.

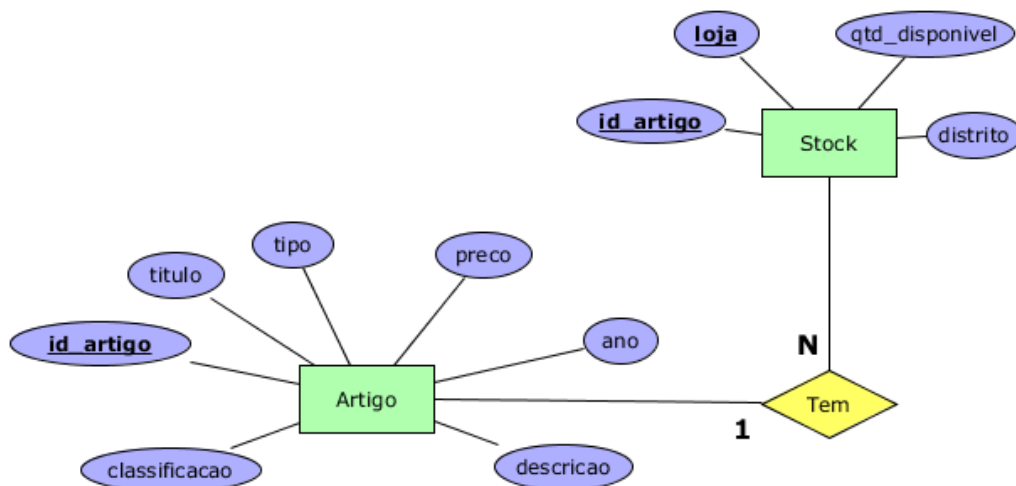


Figura 3 - Relacionamento Artigo e Stock

Artigo e Filme

O relacionamento “pode ser” entre as entidades *Artigo* e *Filme* caracteriza-se por uma cardinalidade de 1 para 1, dado que um artigo pode ser um *Filme* e um *Filme* tem de ser um *Artigo*.

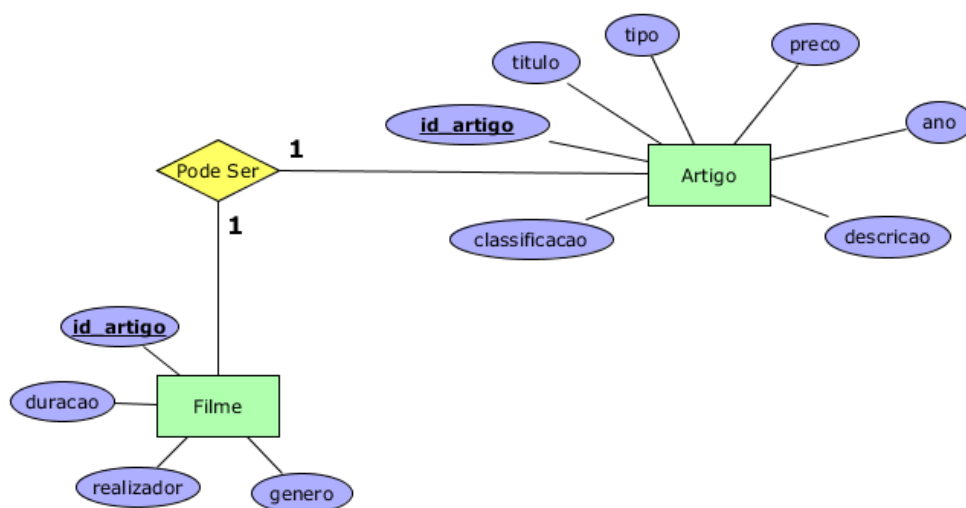


Figura 4 - Relacionamento Artigo e Filme

Artigo e Livro

O relacionamento “pode ser” entre as entidades *Artigo* e *Livro* caracteriza-se por uma cardinalidade de 1 para 1, dado que um artigo pode ser um *Livro* e um *Livro* tem de ser um *Artigo*.

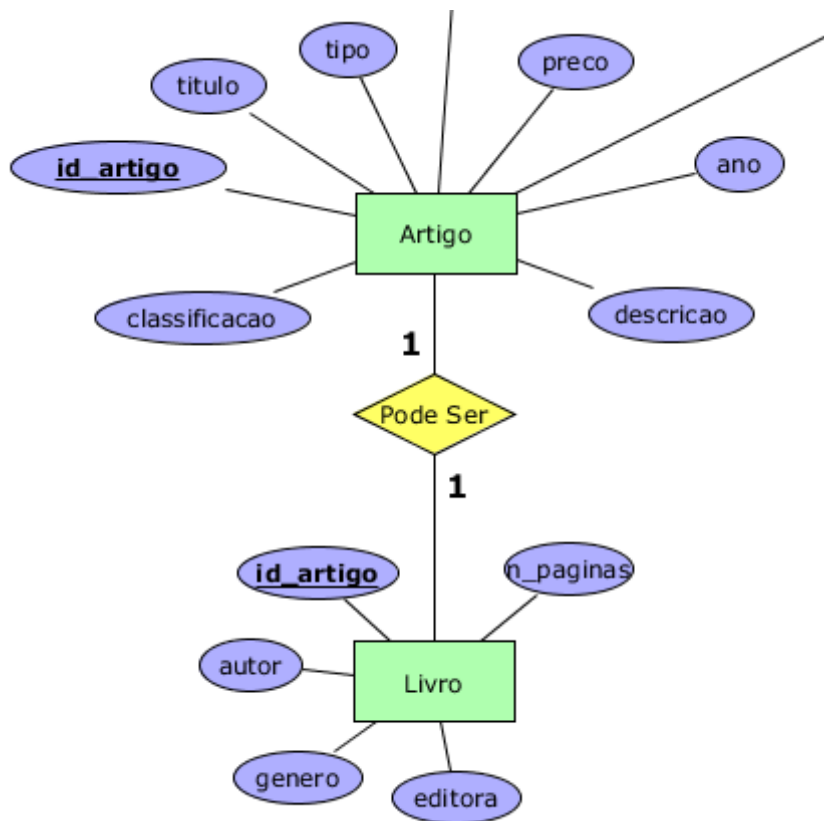


Figura 5 - Relacionamento Artigo e Livro

Artigo e Jogo

O relacionamento “pode ser” entre as entidades *Artigo* e *Jogo* caracteriza-se por uma cardinalidade de 1 para 1, dado que um artigo pode ser um *Jogo* e um *Jogo* tem de ser um *Artigo*.

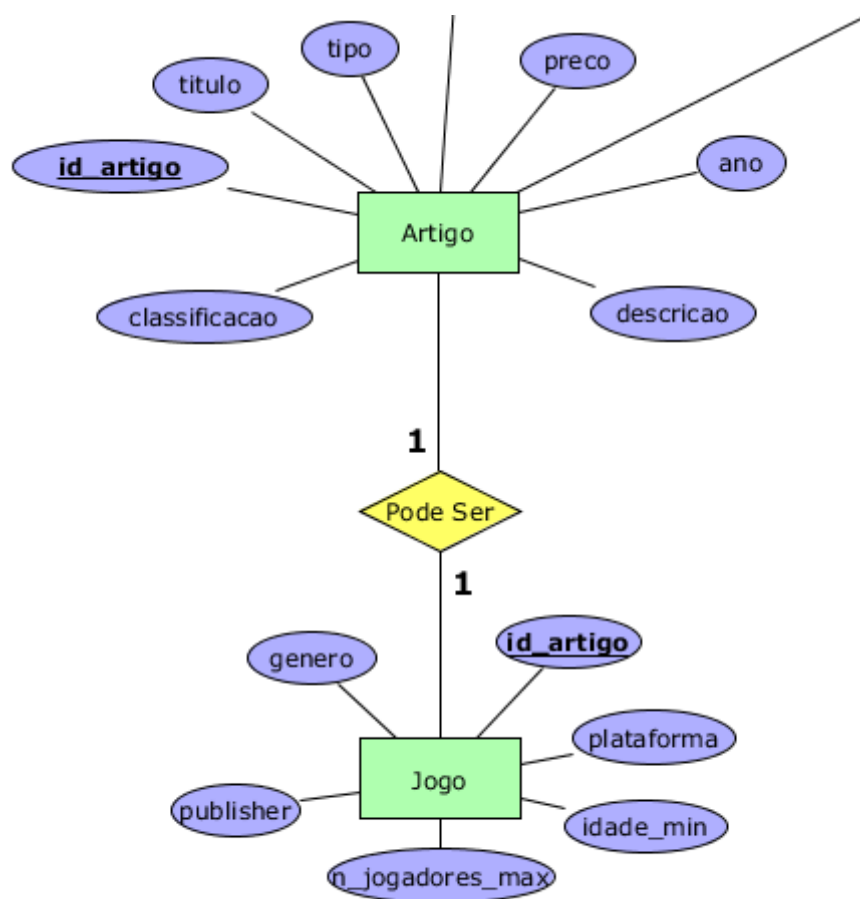


Figura 6 - Relacionamento Artigo e Jogo

Artigo e Musica

O relacionamento “pode ser” entre as entidades *Artigo* e *Musica* caracteriza-se por uma cardinalidade de 1 para 1, dado que um artigo pode ser uma *Musica* e uma *Musica* tem de ser um *Artigo*.

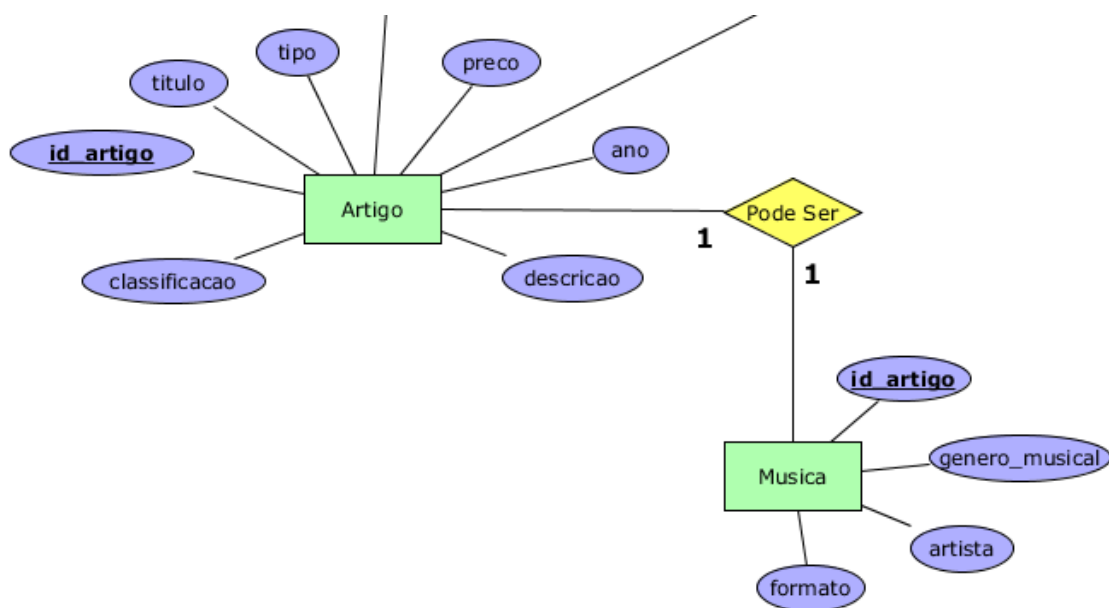


Figura 7 - Relacionamento Artigo e Musica

3.3.1 Dicionário de relacionamento

Entidade	Multiplicidade	Relacionamento	Multiplicidade	Entidade
Cliente	1...1	Efetua	0..N	Compra
Compra	1...N	contém	1..N	Artigo
Artigo	1..1	tem	1..N	Stock
Artigo	1..1	Pode ser	1..1	Filme
Artigo	1..1	Pode ser	1..1	Livro
Artigo	1..1	Pode ser	1..1	Jogo
Artigo	1..1	Pode ser	1..1	Música

Tabela 2 - Dicionário de relacionamento

3.4. Identificação e Associação de Atributos aos Tipos de Entidades e de Relacionamentos

Entidade/Relacionamento	Atributos	Descrição	Tipo do atributo	Tipo de dados e tamanho
Cliente (Entidade)	Id_cliente	Identificador do cliente	Chave primária	INT
	Nome	Nome do cliente	Simples	VARCHAR(45)
	Data de nascimento	Data de nascimento do cliente	Simples	DATE
	Data de subscrição	Data de subscrição do cliente	Simples	DATE
	Distrito	Distrito a que o cliente pertence	Simples	VARCHAR(45)
	Email	Email do cliente	Simples	VARCHAR(45)
	Telemóvel	Contacto telefónico do cliente	Simples	VARCHAR(45)
Compra (entidade)	Id_compra	Identificador da compra	Chave primária	INT
	Montante	Montante total da compra	Simples	DECIMAL(6,2)
	Loja	Loja à qual foi retirado stock dos artigos comprados	Simples	VARCHAR(45)
	Data/hora	Data e hora da realização da compra	Simples	DATETIME
	Id_cliente	Identificador	Chave	INT

		do cliente que realizou a compra	Estrangeira	
Artigo (Entidade)	Id_artigo	Identificador do artigo	Chave primária	INT
	Título	Título/Nome do artigo	Simples	VARCHAR(100)
	Tipo	Tipo do artigo	Simples	VARCHAR(10)
	Preço	Preço do artigo	Simples	DECIMAL(4,2)
	Ano	Ano em que o artigo foi lançado	Simples	INT
	Classificação	Classificação que os clientes deram ao artigo	Simples	INT
Stock (Entidade)	Id_artigo	Identificador do artigo	Chave primária	INT
	Loja	Loja a que corresponde o stock existente	Chave primária	VARCHAR(45)
	Quantidade disponível	Quantidade de artigo que existe na loja	Simples	INT
	Distrito	Distrito a que pertence a loja	Simples	VARCHAR(25)
Filme (Entidade)	Id_artigo	Identificador do artigo	Chave primária	INT
	Duração	Duração do filme	Simples	INT
	Realizador	Realizador do filme	Simples	VARCHAR(45)
	Género	Género do filme	Simples	VARCHAR(25)
Livro (Entidade)	Id_artigo	Identificador do artigo	Chave primária	INT

	Autor	Autor do livro	Simples	VARCHAR(45)
	Género	Género de livro	Simples	VARCHAR(25)
	Editora	Editora	Simples	VARCHAR(45)
	Número de páginas	Número de páginas	Simples	INT
Jogo (Entidade)	Id_artigo	Identificador do artigo	Chave primária	INT
	Plataforma	Plataforma em que se pode jogar o jogo	Simples	VARCHAR(15)
	Idade mínima	Idade mínima permitida	Simples	INT
	Número de jogadores máximo	Número máximo de jogadores possíveis	Simples	INT
	Publicadora	Empresa publicadora	Simples	VARCHAR(45)
	Género	Género de jogo	Simples	VARCHAR(45)
Música (Entidade)	Id_artigo	Identificador do artigo		INT
	Género	Género de música	Simples	VARCHAR(25)
	Artista	Artista criador	Simples	VARCHAR(45)
	Formato	Formato	Simples	VARCHAR(25)
Contém (Relacionamento)	Quantidade	Quantidade de artigos comprados	Simples	INT

Tabela 3 - Dicionário de dados dos atributos das entidades e relacionamentos

3.5. Determinação das chaves candidatas, chaves primárias e chaves alternadas

Nesta secção vamos determinar algo essencial para identificar as entidades. Para isso vamos verificar quais dos atributos de cada entidade poderão ser chaves primárias. Uma chave primária corresponde a um atributo que seja único para que seja possível a identificação da entidade.

As chaves candidatas são os atributos que possam servir como chave primária. Dessas candidatas é escolhida uma primária e as restantes são classificadas como chaves alternadas. Para o nosso Modelo Conceptual estão abaixo apresentadas essas mesmas chaves.

Cliente

Chaves candidatas: Id_cliente

Chave primária: Id_cliente

Compra

Chaves candidatas: Id_compra

Chave primária: Id_compra

Artigo

Chaves candidatas: Id_artigo

Chave primária: Id_artigo

Stock

Chaves candidatas: Id_artigo, loja

Chave primária: Id_artigo

Chave alternada: Loja

Filme

Chaves candidatas: Id_artigo

Chave primária: Id_artigo

Livro

Chaves candidatas: Id_artigo

Chave primária: Id_artigo

Jogo

Chaves candidatas: Id_artigo

Chave primária: Id_artigo

Música

Chaves candidatas: Id_artigo

Chave primária: Id_artigo

3.6. Detalhe ou generalização de entidades

No nosso projeto não foi utilizada generalização nem especialização de entidades no Modelo Conceptual.

3.7. Apresentação e explicação do diagrama ER

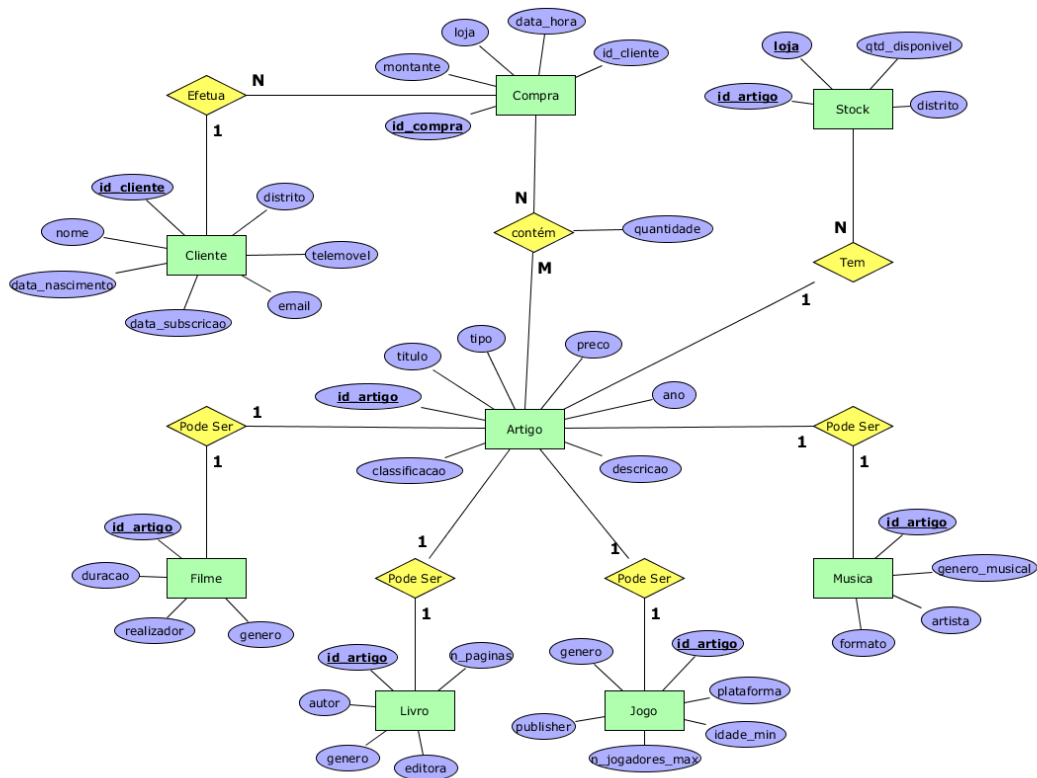


Figura 8 - Modelo Conceptual dos dados

3.8. Validação do modelo de dados com o utilizador

Terminado o Modelo Conceptual, agora é necessário validar. Para isso simulámos algumas perguntas que o utilizador possa fazer de forma a verificar se através do modelo conseguiríamos ou não responder.

1. Quais artigos existem de X tipo? **(RE 14)**

Dado um tipo específico, precisamos da entidade *Artigo* para obter informações como o título, preço, entre outras, desse tipo de artigo. Foi escolhido ordenar por preço crescente, e classificação decrescente, para mostrar uma melhor pesquisa aos utilizadores.

2. Está disponível X artigo em alguma loja? **(RE 20)**

Através da entidade *Stock* obtemos a quantidade disponível do artigo especificado em cada loja.

3. Quais livros de X autor estão disponíveis? **(RE 21)**

Recorremos às entidades *Artigo* e *Livro*. Através do relacionamento entre estes, conseguimos obter todas as informações que precisamos de mostrar ao utilizador na sua pesquisa, como o título, preço, género, entre outras. Seleccionamos todos os livros que tenham como autor o especificado no input.

4. Quais jogos existem adequados para X idade? **(RE 22)**

Damos uso à entidade *Cliente*, assim como à view *jogos_titulo_asc* (que nos mostra todas as informações sobre os jogos e as suas características de artigo). Através do *Cliente*, descobrimos a sua idade, e verificamos quais os jogos apresentados em *jogos_titulo_asc* que tenham idade mínima permitida superior à idade do cliente dado.

4. Modelo lógico

Neste capítulo, após terminarmos a conceptualização do problema, irá ser construído o Modelo Lógico. Este ajudará na derivação dos relacionamentos sendo fundamental para o desenvolvimento da nossa solução para gestão da base de dados. Após tudo isso, este modelo irá suportar o problema que nos foi proposto.

4.1. Construção e validação do modelo de dados lógico

Como foi dito anteriormente, para iniciarmos a construção do Modelo Lógico, é preciso derivar e criar os relacionamentos das entidades e os seus atributos, tendo sido estes definidos no Modelo Conceptual.

4.1.1 Entidades fortes

As entidades fortes são aquelas que tem uma chave primária própria não sendo esta de outra entidade, ou seja, uma entidade que não depende de outra para ser identificada. Cada entidade forte está representada por uma tabela no Modelo Lógico, tendo cada atributo direito a uma coluna.

Cliente (id_cliente, nome, data_nascimento, data_subscricao, email, telemovel, distrito)

Chave primária: id_cliente

Id_cliente	nome	Data_nascimento	Data_subscricao	email	telemovel	distrito
1	Abimael Cardoso	1968-04-06	2014-03-11	Abimael.cardoso@exemplo.com	978801951	Aveiro
(...)	(...)	(...)	(...)	(...)	(...)	(...)

Tabela 4 - Representação da entidade Cliente

Compra (id_compra, montante, loja, data_hora)

Chave primária: id_compra

Id_compra	montante	loja	Data_hora
1	0	Évora Plaza	2013-05-16 19:53:0

(...)	(...)	(...)	(...)
-------	-------	-------	-------

Tabela 5 - Representação da entidade Compra

Artigo (id_artigo, título, tipo, preco, ano, classificacao)

Chave primária: id_artigo

Id_artigo	título	tipo	preco	ano	classificacao
1	To Kill a Mockingbird	Livro	10.8	000	4.6
(...)	(...)	(...)	(...)	(...)	(...)

Tabela 6 - Representação da entidade Artigo

Stock (id_artigo, loja, qtd_disponivel, distrito)

Chave primária: loja

Id_artigo	Loja	Qtd_disponivel	distrito
1	Aeroporto	10	Lisboa
(...)	(...)	(...)	(...)

Tabela 7 - Representação da entidade Stock

4.1.2 Relacionamentos de um para muitos (1:N)

Neste tipo de relacionamento, é comum a entidade que possui multiplicidade N ter como atributo a chave primária da outra entidade, chamando-se assim chave estrangeira.

Neste caso, a chave estrangeira da *Compra* é a chave primária do *Cliente*, sabendo-se assim que uma certa compra foi realizada por um certo cliente.

Cliente (id_cliente, nome, data_nascimento, data_subscricao, email, telemóvel, distrito)

Chave Primária: id_cliente

Compra (id_compra, montante, loja, data_hora)

Chave Primária: id_compra

Chave Estrangeira: id_cliente

Tal como no caso anterior, a entidade que tem como chave estrangeira a principal do *Artigo* é o *Stock*, sendo possível então dizer que um certo *Stock* corresponde a um certo *Artigo*.

Artigo (id_artigo, título, tipo, preco, ano, classificacao)

Chave primária: id_artigo

Stock (id_artigo, loja, qtd_disponivel, distrito)

Chave primária: loja

Chave Estrangeira: id_artigo

4.1.3 Relacionamentos de muitos para muitos (N:M)

Neste tipo de relacionamento, é gerado um novo relacionamento, em que existem duas chaves primárias, sendo elas cada uma das chaves primárias das entidades que formaram este relacionamento.

Compra_de_X_Artigos (quantidade)

Chave primária: id_compra, id_artigo

4.1.4 Relacionamentos hierárquico

Neste tipo de relacionamento, existe uma entidade que é como se fosse uma superclasse que dá origem a subclasses, tendo cada uma das subclasses como chave primária a da superclasse. Nos relacionamentos a seguir, a superclasse é a entidade *Artigo* que dá a chave primária id_artigo às subclasses *Filme*, *Livro*, *Jogo* e *Musica*.

Artigo (id_artigo, título, tipo, preco, ano, classificacao)

Chave primária: id_artigo

Filme (id_artigo, duração, realizador, genero)

Chave primária: id_artigo

Livro (id_artigo, aturo, género, editora, n_paginas)

Chave primária: id_artigo

Jogo (id_artigo, plataforma, idade_min, publisher, n_jogadores_max, genero)

Chave primária: id_artigo

Música (id_artigo, género_musical, artista, formato)

Chave primária: id_artigo

4.2. Desenho do modelo lógico

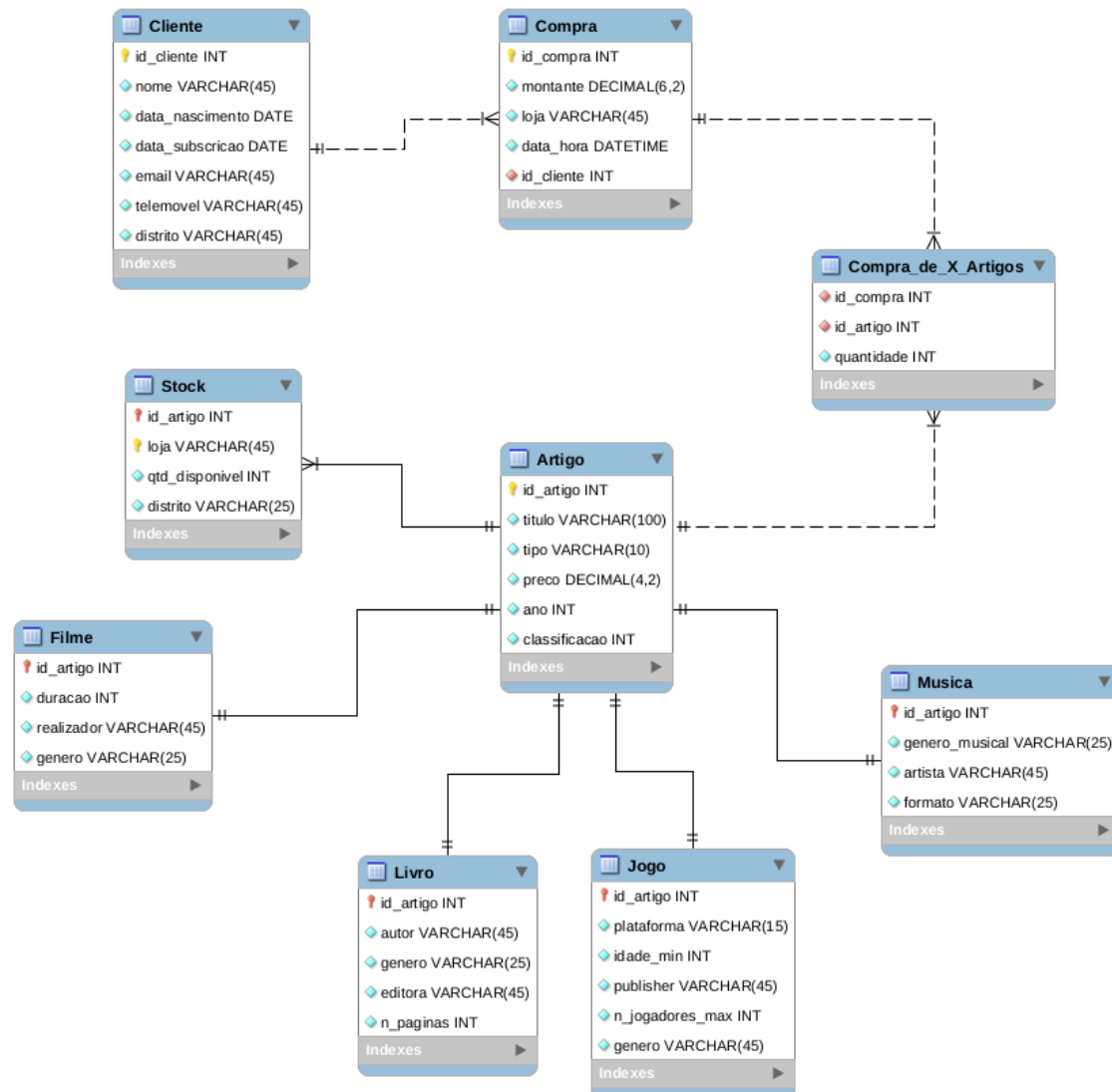


Tabela 8 - Modelo Lógico

4.3. Dependências Funcionais

Uma vez que um dos principais objetivos na construção do modelo relacional de uma base de dados é ter a certeza que os atributos são agrupados em relações com número mínimo possível e de forma consistente, é necessário validar para que isso seja assegurado, tendo sempre o foco em ser possível suportar as exigências que foram propostas.

Em baixo são apresentadas as dependências funcionais entre os atributos

Dependências Funcionais de Cliente

Id_cliente – nome, data_nascimento, data_subscrição. Email, telemóvel, distrito

Dependências Funcionais de Compra

Id_compra, montante, loja, data_hota

Dependências Funcionais de Artigo

Id_artigo – titulo, tipo, preco, ano, classificação

Dependências Funcionais de Stock

Loja – id_artigo, qtd_disponivel, distrito

Dependências Funcionais de Filme

Id_artigo – duração, realizador, género

Dependências Funcionais de Livro

Id_artigo – autor, género, editora, n_paginas

Dependências Funcionais de Jogo

Id_artigo – plataforma, idade_min, publisher, n_jogadores_max, género

Dependências Funcionais de Musica

Id_artigo – género_musical, artista, formato

Dependências Funcionais de Compra_de_X_Artigos

Id_compra, id_artigo - quantidade

4.4. Validação do modelo com interrogações do utilizador

1. Consultar os artigos por tipo (RE 14)

ρ (Titulo, preco, publicado, classificação) σ tipo = tipo_artigo \wedge inf < preco < sup (π titulo, preco, ano, classificação (Artigo))

2. Consultar a disponibilidade de um artigo em todas as lojas (RE 20)

σ id_artigo = id (Stock)

3. Consultar livros de um autor específico (RE 21)

ρ (titulo, preco, publicado, classificacao, genero, editora, N_paginas) σ l.autor (π a.titulo, a.preco, a.ano, a.classificacao, l.genero, l.editora, l.n_paginas ((ρ l(Livro)) \bowtie a.id_artigo = l.id_artigo (ρ a(Artigo))))

4. Verificar que jogos o cliente tem idade mínima para comprar (RE 22)

σ c.id_cliente = id ^ idade (c.data_nascimento) \geq j.'Idade Mínima'((ρ j(jogos_titulo_asc)) \bowtie (ρ c (Cliente)))

4.5. Validação do modelo com as transações estabelecidas

- **Registar Cliente**

Para se efetuar o registo de um cliente basta elaborar uma nova linha na tabela *Cliente*.

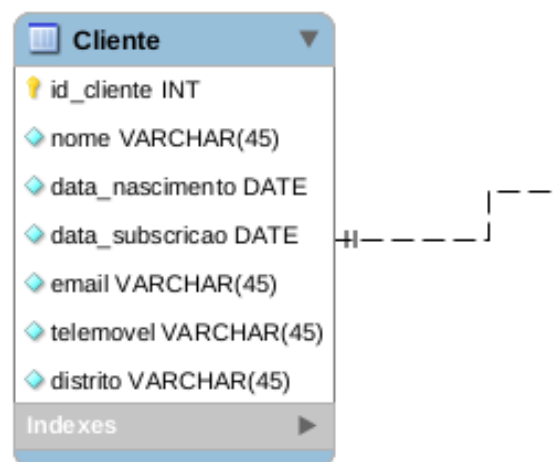


Figura 9 - Tabelas alteradas pela transação Registar Cliente

- **Registar Artigo**

Para se adicionar um novo artigo é necessário fazer um novo registo na tabela *Artigo*, seguido de outro novo registo na tabela *Stock*. Após isso é inserir na tabela respetiva ao tipo as informações adicionais relativamente ao artigo.

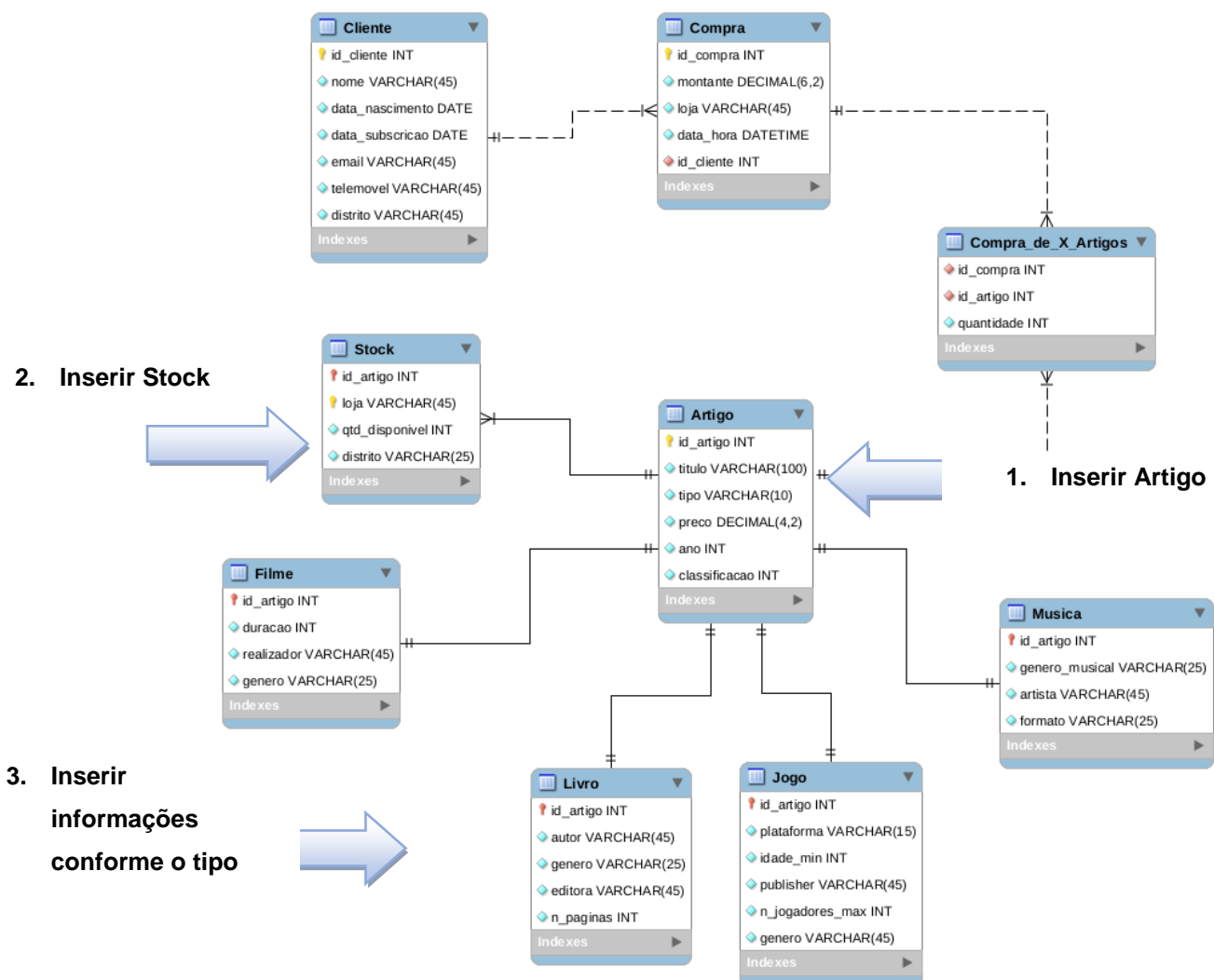


Figura 10 - Tabelas alteradas pela transação Registrar Artigo

- **Alterar preço de Artigo**

Para se alterar o preço de um artigo teremos que alterar o mesmo na tabela à qual está associado esse artigo, sendo essa tabela *Artigo*.

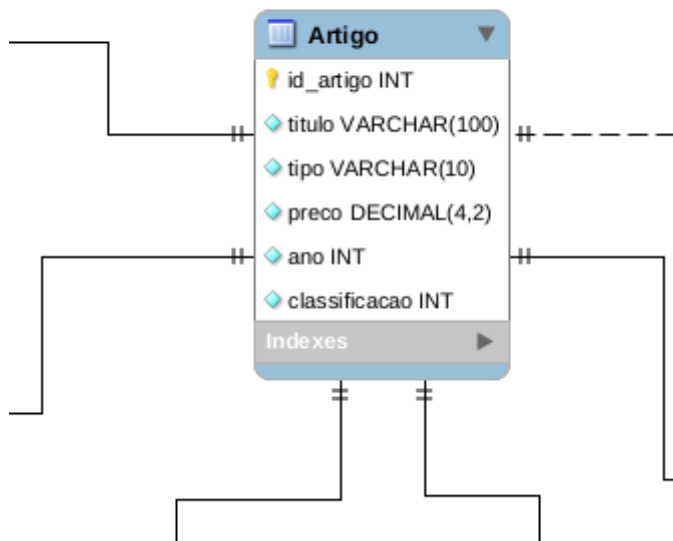


Figura 11 - Tabelas alteradas pela transação Alterar preço de artigo

- **Alterar Stock**

Para se alterar o stock de um artigo basta alterar a **qtd_disponivel** na tabela *Stock* associada ao mesmo. É de referir que caso o artigo não exista, então a tabela *Stock* desse mesmo também não irá existir, sendo necessário registar o artigo primeiro para se conseguir alterar o stock.

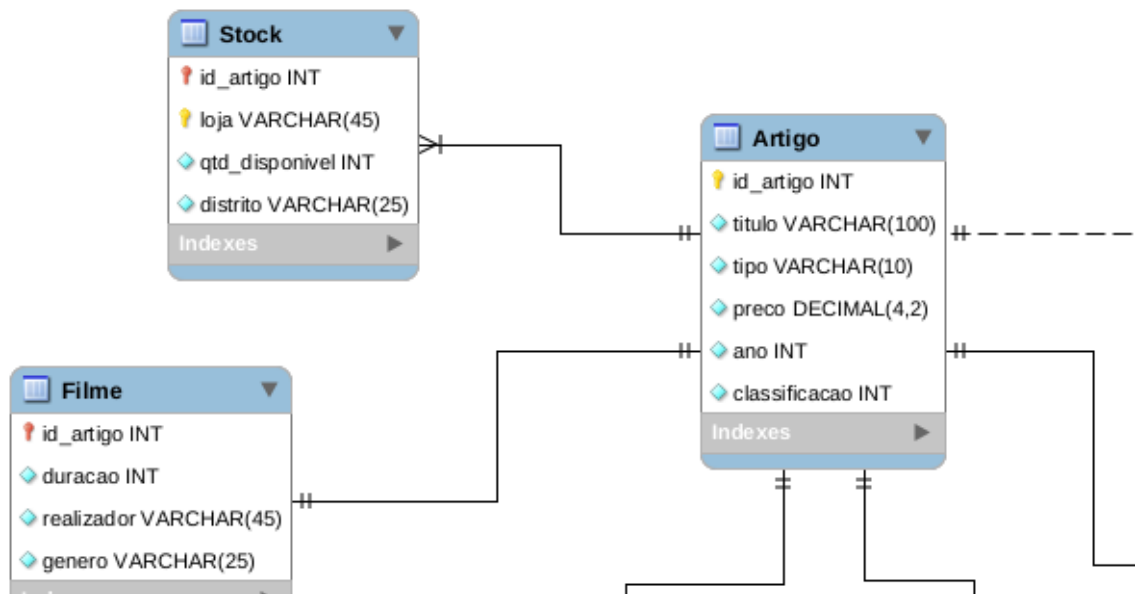


Figura 12 - Tabelas alteradas pela transação Alterar Stock

- **Registar Compra**

Para existir um registo de uma compra é necessário primeiro haver uma verificação se os artigos que o cliente deseja comprar têm stock disponível. Após essa verificação é então feito um novo registo na tabela *Compra* onde o montante ainda não está definido. Logo de seguida, são registadas várias linhas na tabela *Compra_de_X_Artigos* com as quantidades e os ids respetivos. Finalmente é registado o montante total na compra através de um *trigger* automático e são atualizados os stocks dos artigos comprados.

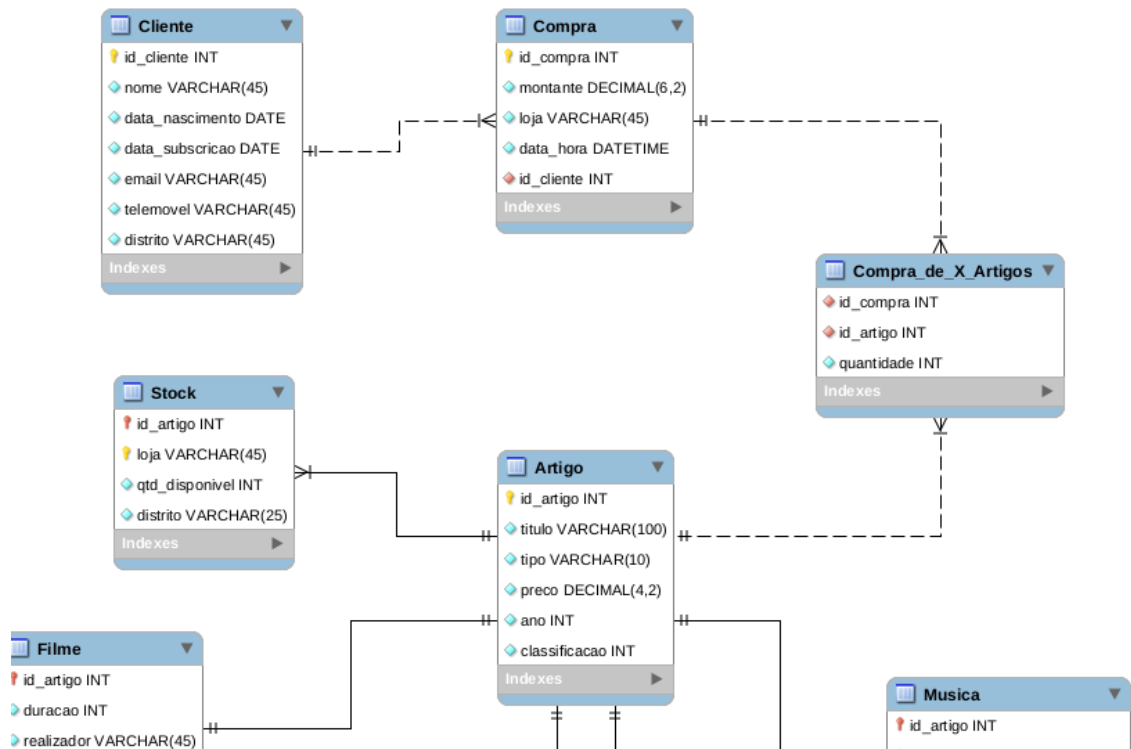


Figura 13 - Tabelas alteradas pela transação Registrar Compra

4.6. Revisão do modelo lógico com o utilizador

Com tudo o que foi resolvido voltamos a reunir com o administrador para apresentar o que foi feito, revendo cada pormenor para ter a certeza que estava tudo bem resolvido e processado. Começamos então a trabalhar no Modelo Físico pois obtivemos aprovação do administrador.

5. Implementação Física

5.1. Seleção do sistema de gestão de bases de dados

Para o nosso projeto decidimos escolher um sistema de gestão que fosse gratuito, fácil de utilizar e principalmente que tivesse um bom desempenho. Esta base de dados ser de tamanho reduzido também facilitou a escolha não sendo necessário uma ferramenta tão complexa. Tendo em conta todas as necessidades enumeradas, consideramos então a melhor escolha o MySQL, que cumpre todos os nossos requisitos.

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

Nesta etapa, tendo o Modelo Lógico já terminado, passamos à elaboração do Modelo Físico. Para tal recorreremos à ferramenta *Forward Engineer* do *MySQL Workbench*, que gera o Modelo Físico através do Modelo Lógico, facilitando o processo de sustentar as relações base e restrições anteriormente definidas.

5.2.1 Desenho das relações base

- **Relação Cliente**

Cliente - Table									
Table	Columns	Indexes	Foreign Keys	Triggers	Partitioning	Options	Inserts	Privileges	
Column Name		Datatype		PK	NN				
id_cliente		INT		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				
nome		VARCHAR(45)		<input type="checkbox"/>	<input checked="" type="checkbox"/>				
data_nascimento		DATE		<input type="checkbox"/>	<input checked="" type="checkbox"/>				
data_subscricao		DATE		<input type="checkbox"/>	<input checked="" type="checkbox"/>				
email		VARCHAR(45)		<input type="checkbox"/>	<input checked="" type="checkbox"/>				
telemovel		VARCHAR(45)		<input type="checkbox"/>	<input checked="" type="checkbox"/>				
distrito		VARCHAR(45)		<input type="checkbox"/>	<input checked="" type="checkbox"/>				

Figura 14 - Relação Cliente

Domínio id_cliente	Inteiro
Domínio nome	String tamanho variável
Domínio data_nascimento	Data
Domínio data_subscricao	Data
Domínio email	String tamanho variável
Domínio telemóvel	String tamanho variável
Domínio distrito	String tamanho variável

PRIMARY KEY (id_cliente);

- **Relação Compra**

Compra - Table									
Table	Columns	Indexes	Foreign Keys	Triggers	Partitioning	Options	Inserts	Privileges	
Column Name		Datatype		PK	NN				
id_compra		INT		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				
montante		DECIMAL(6,2)		<input type="checkbox"/>	<input checked="" type="checkbox"/>				
loja		VARCHAR(45)		<input type="checkbox"/>	<input checked="" type="checkbox"/>				
data_hora		DATETIME		<input type="checkbox"/>	<input checked="" type="checkbox"/>				
id_cliente		INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>				

Figura 15 - Relação Compra

Domínio id_cliente	Inteiro
Domínio nome	String tamanho variável
Domínio data_nascimento	Data
Domínio data_subscricao	Data

Domínio email	String tamanho variável
Domínio telemóvel	String tamanho variável
Domínio distrito	String tamanho variável

- **Relação Artigo**

Artigo - Table									
Table	Columns	Indexes	Foreign Keys	Triggers	Partitioning	Options	Inserts	Privileges	
Column Name						Datatype		PK	NN
🔑 id_artigo						INT		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
💎 titulo						VARCHAR(100)		<input type="checkbox"/>	<input checked="" type="checkbox"/>
💎 tipo						VARCHAR(10)		<input type="checkbox"/>	<input checked="" type="checkbox"/>
💎 preco						DECIMAL(4,2)		<input type="checkbox"/>	<input checked="" type="checkbox"/>
💎 ano						INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>
💎 classificacao						INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>

Figura 16 - Relação Artigo

Domínio id_artigo	Inteiro
Domínio titulo	String tamanho variável
Domínio tipo	String tamanho variável
Domínio preco	Decimal
Domínio ano	Inteiro
Domínio classificação	Inteiro

- **Relação Stock**

Stock - Table									
Table	Columns	Indexes	Foreign Keys	Triggers	Partitioning	Options	Inserts	Privileges	
Column Name						Datatype		PK	NN
🔑 id_artigo						INT		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
🔑 loja						VARCHAR(45)		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
💎 qtd_disponivel						INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>
💎 distrito						VARCHAR(25)		<input type="checkbox"/>	<input checked="" type="checkbox"/>

Figura 17 - Relação Stock

Domínio id_artigo	Inteiro
Domínio loja	String tamanho variável
Domínio qtd_disponivel	Inteiro

Domínio distrito

String tamanho variável

- **Relação Filme**

Filme - Table									
Table	Columns	Indexes	Foreign Keys	Triggers	Partitioning	Options	Inserts	Privileges	
Column Name						Datatype		PK	NN
id_artigo						INT		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
duracao						INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>
realizador						VARCHAR(45)		<input type="checkbox"/>	<input checked="" type="checkbox"/>
genero						VARCHAR(25)		<input type="checkbox"/>	<input checked="" type="checkbox"/>

Figura 18 - Relação Filme

Domínio id_artigo

Inteiro

Domínio duracao

Inteiro

Domínio realizador

String tamanho variável

Domínio genero

String tamanho variável

- **Relação Livro**






Livro - Table										
Table	Columns	Indexes	Foreign Keys	Triggers	Partitioning	Options	Inserts	Privileges		
Column Name					Datatype		PK	NN	UQ	
 id_artigo						INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 autor						VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 genero						VARCHAR(25)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 editora						VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 n_paginas						INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

Figura 19 - Relação Livro

Domínio id_artigo

Inteiro

Domínio autor

String tamanho variável

Domínio genero

String tamanho variável

Domínio editora

String tamanho variável

Domínio n_paginas

Inteiro

- **Relação Musica**

Musica - Table								
Table	Columns	Indexes	Foreign Keys	Triggers	Partitioning	Options	Inserts	Privileges
Column Name		Datatype		PK	NN			
id_artigo		INT		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
genero_musical		VARCHAR(25)		<input type="checkbox"/>	<input checked="" type="checkbox"/>			
artista		VARCHAR(45)		<input type="checkbox"/>	<input checked="" type="checkbox"/>			
formato		VARCHAR(25)		<input type="checkbox"/>	<input checked="" type="checkbox"/>			

Figura 20 - Relação Musica

Domínio id_artigo	Inteiro
Domínio genero_musical	String tamanho variável
Domínio artista	String tamanho variável
Domínio formato	String tamanho variável

- **Relação Jogo**

Jogo - Table								
Table	Columns	Indexes	Foreign Keys	Triggers	Partitioning	Options	Inserts	Privileges
Column Name		Datatype		PK	NN			
id_artigo		INT		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
plataforma		VARCHAR(15)		<input type="checkbox"/>	<input checked="" type="checkbox"/>			
idade_min		INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>			
publisher		VARCHAR(45)		<input type="checkbox"/>	<input checked="" type="checkbox"/>			
n_jogadores_max		INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>			
genero		VARCHAR(45)		<input type="checkbox"/>	<input checked="" type="checkbox"/>			

Figura 21 - Relação Jogo

Domínio id_artigo	Inteiro
Domínio plataforma	String tamanho variável
Domínio idade_min	Inteiro
Domínio publisher	String tamanho variável
Domínio n_jogadores_max	Inteiro
Domínio genero	String tamanho variável

- **Relação Compra_de_X_Artigos**

Compra_de_X_Artigos - Table									
Table	Columns	Indexes	Foreign Keys	Triggers	Partitioning	Options	Inserts	Privileges	
Column Name						Datatype		PK	NN
id_compra						INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>
id_artigo						INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>
quantidade						INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>

Figura 22 - Relação Compra_de_X_Artigos

Domínio id_compra Inteiro

Domínio id_artigo Inteiro

Domínio quantidade Inteiro

5.2.2 Desenho das restrições

Abaixo estão apresentados os scripts de criação.

- **Cliente**

```
-- Table `FNAC`.`Cliente`

CREATE TABLE IF NOT EXISTS `FNAC`.`Cliente` (
  `id_cliente` INT NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  `data_nascimento` DATE NOT NULL,
  `data_subscricao` DATE NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  `telemovel` VARCHAR(45) NOT NULL,
  `distrito` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id_cliente`))
ENGINE = InnoDB;
```

Figura 23 - Criação da Tabela Cliente

- **Compra**

```

-----
-- Table `FNAC`.`Compra`
-----

CREATE TABLE IF NOT EXISTS `FNAC`.`Compra` (
  `id_compra` INT NOT NULL,
  `montante` DECIMAL(6,2) NOT NULL,
  `loja` VARCHAR(45) NOT NULL,
  `data_hora` DATETIME NOT NULL,
  `id_cliente` INT NOT NULL,
  PRIMARY KEY (`id_compra`),
  INDEX `fk_Compra_Cliente_idx` (`id_cliente` ASC),
  CONSTRAINT `fk_Compra_Cliente`
    FOREIGN KEY (`id_cliente`)
      REFERENCES `FNAC`.`Cliente` (`id_cliente`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

Figura 24 - Criação da Tabela Compra

- **Artigo**

```

-----
-- Table `FNAC`.`Artigo`
-----

CREATE TABLE IF NOT EXISTS `FNAC`.`Artigo` (
  `id_artigo` INT NOT NULL COMMENT 'Aplicável a todos',
  `titulo` VARCHAR(100) NOT NULL COMMENT 'Aplicável a todos',
  `tipo` VARCHAR(10) NOT NULL,
  `preco` DECIMAL(4,2) NOT NULL COMMENT 'Aplicável a todos',
  `ano` INT NOT NULL COMMENT 'Aplicável a todos',
  `classificacao` INT NOT NULL COMMENT 'Classificação de 1-10\nAplicável a todos',
  PRIMARY KEY (`id_artigo`))
ENGINE = InnoDB;

```

Figura 25 - Criação da Tabela Artigo

- **Stock**

```

-----
-- Table `FNAC`.`Stock`
-----

CREATE TABLE IF NOT EXISTS `FNAC`.`Stock` (
  `id_artigo` INT NOT NULL,
  `loja` VARCHAR(45) NOT NULL,
  `qtd_disponivel` INT NOT NULL,
  `distrito` VARCHAR(25) NOT NULL,
  PRIMARY KEY (`id_artigo`, `loja`),
  CONSTRAINT `fk_Stock_Artigo1`
  FOREIGN KEY (`id_artigo`)
  REFERENCES `FNAC`.`Artigo` (`id_artigo`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

Figura 26 - Criação da Tabela Stock

- Filme

```

-----
-- Table `FNAC`.`Filme`
-----

CREATE TABLE IF NOT EXISTS `FNAC`.`Filme` (
  `id_artigo` INT NOT NULL,
  `duracao` INT NOT NULL COMMENT 'Duração de um filme em minutos',
  `realizador` VARCHAR(45) NOT NULL COMMENT 'Falta mostrar atores\nAplicável ao Filme',
  `genero` VARCHAR(25) NOT NULL COMMENT 'Género - aplicável ao livro, filme, ou jogo.',
  PRIMARY KEY (`id_artigo`),
  CONSTRAINT `fk_Filme_Artigo1`
  FOREIGN KEY (`id_artigo`)
  REFERENCES `FNAC`.`Artigo` (`id_artigo`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

Figura 27 - Criação da Tabela Filme

- Livro

```

-----
-- Table `FNAC`.`Livro`
-----
CREATE TABLE IF NOT EXISTS `FNAC`.`Livro` (
  `id_artigo` INT NOT NULL,
  `autor` VARCHAR(45) NOT NULL COMMENT 'Autor do livro',
  `genero` VARCHAR(25) NOT NULL COMMENT 'Género - aplicável ao livro, filme, ou jogo.',
  `editora` VARCHAR(45) NOT NULL COMMENT 'Editora do livro',
  `n_paginas` INT NOT NULL COMMENT 'Nº de páginas do livro',
  PRIMARY KEY (`id_artigo`),
  CONSTRAINT `fk_Livro_Artigo1`
  FOREIGN KEY (`id_artigo`)
  REFERENCES `FNAC`.`Artigo` (`id_artigo`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

Figura 28 - Criação da Tabela Livro

- **Jogo**

```

-----
-- Table `FNAC`.`Jogo`
-----
CREATE TABLE IF NOT EXISTS `FNAC`.`Jogo` (
  `id_artigo` INT NOT NULL,
  `plataforma` VARCHAR(15) NOT NULL COMMENT 'PC\nPS4\nXBOX\n...\nAplicável ao Jogo',
  `idade_min` INT NOT NULL COMMENT 'Idade mínima permitida\nAplicável a Jogo e Filme',
  `publisher` VARCHAR(45) NOT NULL COMMENT 'Aplicável ao Jogo',
  `n_jogadores_max` INT NOT NULL COMMENT 'Aplicável ao Jogo',
  `genero` VARCHAR(45) NOT NULL COMMENT 'Género - aplicável ao livro, filme, ou jogo.',
  PRIMARY KEY (`id_artigo`),
  CONSTRAINT `fk_Jogo_Artigo1`
  FOREIGN KEY (`id_artigo`)
  REFERENCES `FNAC`.`Artigo` (`id_artigo`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

Figura 29 - Criação da Tabela Jogo

- **Música**


```

-----
-- Table `FNAC`.`Musica`
-----
CREATE TABLE IF NOT EXISTS `FNAC`.`Musica` (
  `id_artigo` INT NOT NULL,
  `genero_musical` VARCHAR(25) NOT NULL COMMENT 'Género - Musica',
  `artista` VARCHAR(45) NOT NULL COMMENT 'Artista - Musica',
  `formato` VARCHAR(25) NOT NULL COMMENT 'CD\nVinil\n...\nAplicável à Música',
  PRIMARY KEY (`id_artigo`),
  CONSTRAINT `fk_Musica_Artigo1`
  FOREIGN KEY (`id_artigo`)
  REFERENCES `FNAC`.`Artigo` (`id_artigo`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

Figura 30 - Criação da Tabela Musica

- **Compra_de_X_Artigos**

```

-----
-- Table `FNAC`.`Compra_de_X_Artigos`
-----
CREATE TABLE IF NOT EXISTS `FNAC`.`Compra_de_X_Artigos` (
  `id_compra` INT NOT NULL,
  `id_artigo` INT NOT NULL,
  `quantidade` INT NOT NULL,
  INDEX `fk_Compra_has_Artigo_Artigo1_idx` (`id_artigo` ASC),
  INDEX `fk_Compra_has_Artigo_Compra1_idx` (`id_compra` ASC),
  CONSTRAINT `fk_Compra_has_Artigo_Compra1`
  FOREIGN KEY (`id_compra`)
  REFERENCES `FNAC`.`Compra` (`id_compra`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
  CONSTRAINT `fk_Compra_has_Artigo_Artigo1`
  FOREIGN KEY (`id_artigo`)
  REFERENCES `FNAC`.`Artigo` (`id_artigo`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

Figura 31 - Criação da Tabela Compra_de_X_Artigos

5.3. Tradução das interrogações do utilizador para SQL

Nesta secção são apresentadas alguns dos requisitos de exploração, sendo eles possíveis interrogações dos utilizadores.

1. Consultar os artigos por tipo (RE 14)

```
-- 14. Consultar os artigos por tipo
-- tipo específico, ordenado por preço ascendente, classificacao descendente, com
-- limites inferior e superior de preço
drop procedure if exists pesquisa_por_tipo;
DELIMITER //
create procedure pesquisa_por_tipo(in tipo_artigo varchar(45), in inf int, in sup int)
begin
select titulo as Título, preco as Preço,
ano as Publicado, classificacao as Classificação
from Artigo
where tipo = tipo_artigo and preco between inf and sup
order by preco ASC, classificacao DESC;
end //
DELIMITER ;

call pesquisa_por_tipo('Livro',10,20);
```

Figura 32 - Resolução do requisito de exploração 14

2. Consultar a disponibilidade de um artigo em todas as lojas (RE 20)

```
-- 20.Consultar a disponibilidade de um artigo em todas as lojas
drop procedure if exists verifica_stock;
DELIMITER //
create procedure verifica_stock(in id int)
begin
select * from Stock where id_artigo = id;
end //
DELIMITER ;

call verifica_stock(1);
```

Figura 33 - Resolução do requisito de exploração 20

3. Consultar livros de um autor específico (RE 21)

```
-- 21. Consultar livros de um autor específico
drop procedure if exists livros_autor;
DELIMITER //
create procedure livros_autor(in autor varchar(45))
begin
select a.titulo as Título, a.preco as Preço,
a.ano as Publicado, a.classificacao as Classificação,
l.genero as Género, l.editora as Editora, l.n_paginas as Nº_Páginas
from Livro l join Artigo a on a.id_artigo = l.id_artigo
where l.autor = autor
order by a.titulo ASC;
end //
DELIMITER ;

call livros_autor('Jane Austen');
```

Figura 34 - Resolução do requisito de exploração 21

4. Verificar que jogos o cliente tem idade mínima para comprar (RE 22)

```
-- 22. Verificar que jogos o cliente tem idade mínima para comprar
drop procedure if exists jogos_permitidos;
DELIMITER //
create procedure jogos_permitidos(in id int)
begin
select * from jogos_titulo_asc j, Cliente c
where c.id_cliente = id
and idade(c.data_nascimento) >= j.`Idade Mínima`;
end //
DELIMITER ;

call jogos_permitidos(1);
```

Figura 35 - Resolução do requisito de exploração 22

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

No desenvolvimento da base de dados é importante ter em atenção o espaço que esta vai ocupar, minimizando sempre que possível para evitar espaço extra ocupado. Abaixo está uma estimativa do que a nossa base de dados irá ocupar no futuro por casa tabela do Modelo Lógico.

Tipo de dados	Tamanho (bytes)
INT	4

VARCHAR(N)	N+1
DATE	3
DECIMAL(P,S)	S – P
DATETIME	8

Tabela 9 - Espaço ocupado no disco por cada tipo de dados

Cliente	Atributos	Tipo de dados	Espaço no disco
	Id_cliente	INT	4
	Nome	VARCHAR(45)	46
	Data_nascimento	DATE	3
	Data_subscricao	DATE	3
	Email	VARCHAR(45)	46
	Telemóvel	VARCHAR(45)	46
	Distrito	VARCHAR(45)	46
Total			194

Tabela 10 - Espaço ocupado no disco por Cliente

Compra	Atributos	Tipo de dados	Espaço no disco
	Id_compra	INT	4
	Montante	DECIMAL(6,2)	4
	loja	VARCHAR(45)	46
	Data_hora	DATETIME	8
	Id_cliente	VARCHAR(45)	46
Total			108

Tabela 11 - Espaço ocupado no disco por Compra

Artigo	Atributos	Tipo de dados	Espaço no disco
	Id_artigo	INT	4
	titulo	VARCHAR(100)	101
	Tipo	VARCHAR(10)	11
	Preco	DECIMAL(4,2)	2
	Ano	INT	4
	Classificação	INT	4
Total			126

Tabela 12 - Espaço ocupado no disco por Artigo

Stock	Atributos	Tipo de dados	Espaço no disco
	Id_artigo	INT	4
	Loja	VARCHAR(45)	46
	Qtd_disponivel	INT	4

	Distrito	VARCHAR(25)	26
Total			80

Tabela 13 - Espaço ocupado no disco por Stock

Filme	Atributos	Tipo de dados	Espaço no disco
	Id_artigo	INT	4
	Duração	INT	4
	Realizador	VARCHAR(45)	46
	Gênero	VARCHAR(25)	26
Total			80

Tabela 14 - Espaço ocupado no disco por Filme

Jogo	Atributos	Tipo de dados	Espaço no disco
	Id_artigo	INT	4
	Plataforma	VARCHAR(45)	46
	Idade_min	INT	4
	Publisher	VARCHAR(45)	46
	N_jogadores_max	INT	4
	Gênero	VARCHAR(45)	46
Total			150

Tabela 15 - Espaço ocupado no disco por Jogo

Livro	Atributos	Tipo de dados	Espaço no disco
	Id_artigo	INT	4
	Autor	VARCHAR(45)	46
	Gênero	VARCHAR(25)	26
	Editora	VARCHAR(45)	46
	N_paginas	INT	4
Total			126

Tabela 16 - Espaço ocupado no disco por Livro

Musica	Atributos	Tipo de dados	Espaço no disco
	Id_artigo	INT	4
	Genero_musical	VARCHAR(25)	26
	Artista	VARCHAR(45)	46
	Formato	VARCHAR(25)	26
Total			102

Compra_de_X_Artigos	Atributos	Tipo de dados	Espaço no disco
	Id_compra	INT	4

	Id_artigo	INT	4
	quantidade	INT	4
Total			12

Tabela 17 - Espaço ocupado no disco por Compra_de_X_Artigos

Tendo em conta o espaço ocupado de cada tabela, em seguida é mostrado o espaço que ocupará com os dados que temos atualmente:

Tabela	Espaço no disco
Cliente	$6228 * 194 = 1208232$
Compra	$2082 * 108 = 224856$
Artigo	$1552 * 126 = 195552$
Stock	$17 * 1552 * 80 = 2110720$
Filme	$250 * 80 = 20000$
Livro	$118 * 126 = 13688$
Jogo	$885 * 150 = 132750$
Musica	$269 * 102 = 27438$
Compra_de_X_Artigos	$5124 * 12 = 61488$
Total	3994724

Tabela 18 - Espaço ocupado no disco pela base de dados

Observando a tabela acima, podemos ver que a estimativa para a nossa base de dados com os dados que temos atualmente esperamos ser 3,994,724 bytes.

5.5. Definição e caracterização das vistas de utilização em SQL

Nesta secção, são apresentadas views que consideramos que serão mais úteis para os utilizadores da base de dados.

Nas figuras abaixo, estão apresentadas as views dos filmes, jogos, filmes e músicas ordenados por título de forma ascendente (alfabeticamente a começar por “a”).

```
-- pesquisa completa dos filmes
drop view if exists filmes_titulo_asc;
create view filmes_titulo_asc as
select a.id_artigo as Id, a.titulo as Título, f.realizador as Realizador,
a.preco as Preço, a.ano as Ano, a.classificacao as Classificação,
f.genero as Género, f.duracao as Duração
from Filme f join Artigo a on a.id_artigo = f.id_artigo
order by a.titulo ASC;

select * from filmes_titulo_asc;
```

Figura 36 - View Filmes

```
-- pesquisa completa dos jogos
drop view if exists jogos_titulo_asc;
create view jogos_titulo_asc as
select a.id_artigo as Id, a.titulo as Título, j.plataforma as Plataforma, a.preco as Preço,
a.ano as Ano, a.classificacao as Classificação, j.genero as Género,
j.publisher as Editor, j.idade_min as `Idade Mínima`, j.n_jogadores_max as `Nº Max Jogadores`
from Jogo j join Artigo a on a.id_artigo = j.id_artigo
order by a.titulo ASC;

select * from jogos_titulo_asc;
```

Figura 37 - View Jogos

```
-- pesquisa completa dos livros
drop view if exists livros_titulo_asc;
create view livros_titulo_asc as
select a.id_artigo as Id, a.titulo as Título, l.autor as Autor,
a.preco as Preço, a.ano as Publicado, a.classificacao as Classificação,
l.genero as Género, l.editora as Editora, l.n_paginas as `Nº Páginas`
from Livro l join Artigo a on a.id_artigo = l.id_artigo
order by a.titulo ASC;

select * from livros_titulo_asc;
```

Figura 38 - View Livros

```
-- pesquisa completa das musicas --
drop view if exists musicas_titulo_asc;
create view musicas_titulo_asc as
select a.id_artigo as Id, a.titulo as Título, m.artista as Artista,
m.formato as Formato, a.preco as Preço, a.ano as Publicado,
a.classificacao as Classificação, m.genero_musical as Género
from Musica m join Artigo a on a.id_artigo = m.id_artigo
order by a.titulo ASC;

select * from musicas_titulo_asc;
```

Figura 39 - View Musica

Seguidamente, está apresentada a view que mostra os top 3 artigos mais vendidos.

```
-- 18. Obter o top 3 artigos mais vendidos
drop view if exists top_artigos;
create view top_artigos as
select a.id_artigo, titulo, tipo, preco, sum(co.montante) as `montante total` from Artigo a
join Compra_de_X_Artigos cx on cx.id_artigo = a.id_artigo
join Compra co on co.id_compra = cx.id_compra
group by a.id_artigo
order by sum(co.montante) DESC
limit 3;

select * from top_artigos;
```

Figura 40 - View Top 3 Artigos mais vendidos

E por último, a view que mostra o quanto vendeu cada autor na secção dos livros.

```
-- 23. quanto cada autor já vendeu no total (quantidade e montante total) ordenado por lucro total e qtd decrescente
drop view if exists top_vendas_autor;
create view top_vendas_autor as
select l.autor as Autor, sum(a.preco*c.quantidade) as `Montante Total`, sum(c.quantidade) as `Quantidade Total`
from Compra_de_X_Artigos c
join Artigo a on a.id_artigo = c.id_artigo
join Livro l on c.id_artigo = l.id_artigo
group by l.autor
order by `Montante Total` DESC, `Quantidade Total` DESC;
```

Figura 41 - View Vendas Autor

5.6. Revisão do sistema implementado com o utilizador

Tendo em conta tudo o que foi feito anteriormente, reunimos mais uma vez com o administrador para apresentar o que foi feito, de modo a sabermos se tudo estava de acordo como era suposto. Não só foi revisto a parte do Modelo Físico, mas tudo o que foi feito desde o início para confirmar que não havia incompatibilidades com o combinado.

6. Projeto de um Sistema de Base de Dados não Relacional

6.1. Utilização de um sistema NoSQL

Os sistemas de base de dados mais comuns no mercado são os sistemas de base de dados relacionais. Principalmente devido à eficácia no armazenamento de dados estruturados. Contudo, existem alguns problemas associados a este tipo de base de dados. Nomeadamente, falta de performance e escalabilidade.

Para mitigar estes problemas foi desenvolvido um novo tipo de base de dados, as bases de dados não relacionais. Nestas, em vez dos dados serem armazenados sobre a forma de tabelas, é utilizada uma estrutura para armazenar os dados dependente do tipo de base de dados. Existem quatro tipos distintos: chave-valor, grafos, colunas e documentos.

Tendo em conta o volume de vendas da FNAC, o sistema anteriormente apresentado seria manifestamente insuficiente para suportar o tráfego. Pelo que foi decidido em reunião converter a base de dados para uma não relacional.

6.1.1 Neo4j

O Neo4j é uma base de dados NoSQL baseada em Grafos com três componentes: os nodos, os relacionamentos entre eles e as suas propriedades.

Este tipo de base de dados são mais flexíveis e ágeis com uma melhor performance. Com o aumento do volume de dados, uma base de dados relacional torna-se mais lenta, tal não se verifica com uma base de dados não relacional baseada em grafos.

Tendo em conta estas vantagens, a administração da FNAC concordou com o desenvolvimento desta nova base de dados não relacional baseada em grafos denominada de Neo4j.

6.2. Objetivos da Base de Dados

Tendo em conta o volume de vendas que a FNAC apresenta de momento, e tendo em conta o crescimento projetado tendo em conta a melhoria dos serviços, conclui-se que seria necessário melhorar a performance da base de dados.

Deste modo, a solução encontrada para o problema foi a migração da base de dados relacional desenvolvida nos passos anteriores para uma base de dados não relacional.

6.3. Identificação das Queries realizadas sobre o sistema

Tal como foi feito para o sistema anterior, para o Neo4j também serão resolvidas as queries anteriormente definidas:

1. Obter o número de clientes nas lojas todas
2. Obter o número de clientes num determinado distrito
3. Consultar os artigos por tipo
4. Consultar os artigos que não tem stock
5. Obter o top 3 dos clientes com mais artigos comprados
6. Obter o top 3 dos clientes que mais dinheiro gastaram
7. Obter o top 3 artigos mais vendidos
8. Obter o top 3 de filmes mais vendidos
9. Obter o top 3 de livros mais vendidos
10. Obter o top 3 de músicas mais vendidas
11. Consultar livros de um autor específico
12. Verificar se o cliente tem idade mínima para o jogo que quer comprar
13. Verificar que quantidade e montante um autor específico vendeu
14. Consultar quanto foi vendido em uma loja especifica num ano específico
15. Consultar quanto foi vendido em cada mês num ano específico
16. Verificar a existência de stock de um certo artigo no distrito de um certo cliente

6.4. Estrutura base para o sistema de dados NoSQL

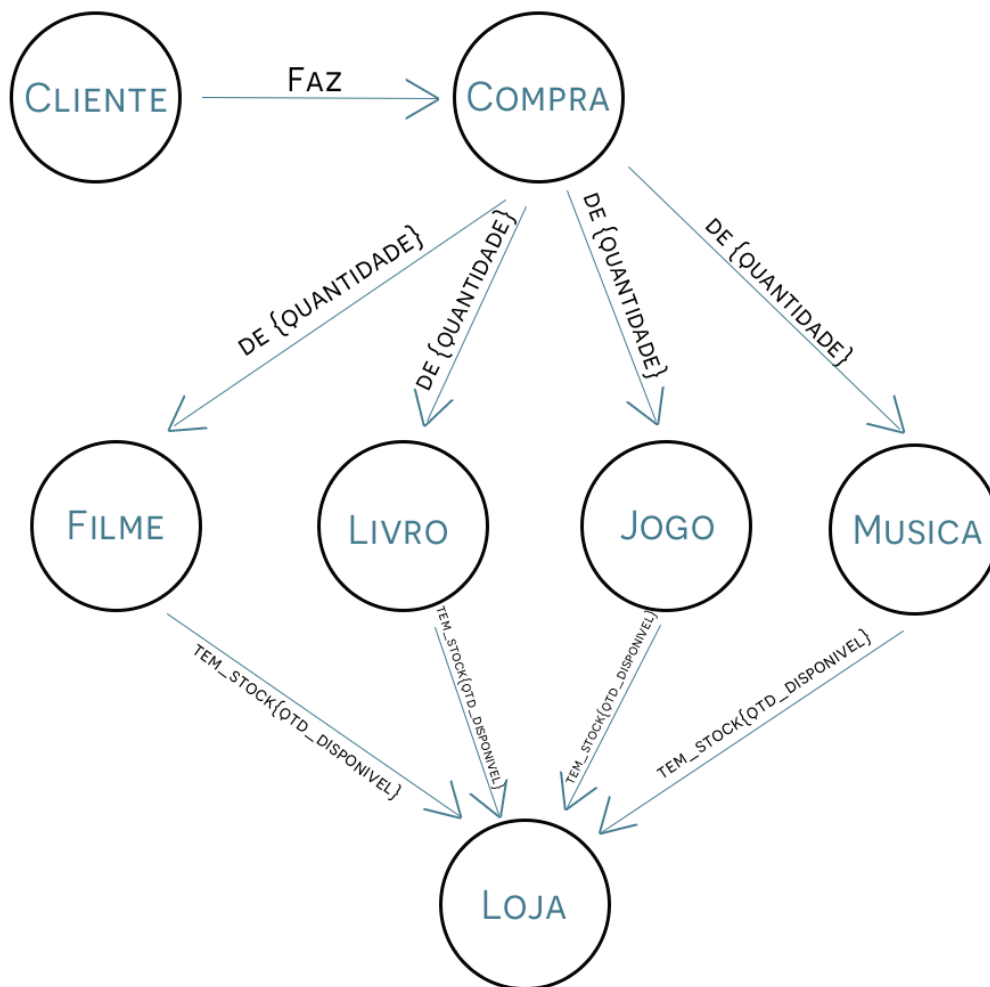


Figura 42 - Estrutura adotada para o sistema NoSQL

6.4.1 Objetos de dados no sistema NoSQL

Através do Modelo Lógico do MySQL, foi criada a estrutura representada acima. Sendo que o Neo4j trabalha com grafos, então para facilitar a passagem e representar melhor o nosso sistema de dados, a estrutura é ilustrada como um grafo em que, cada nodo corresponde a uma tabela do modelo lógico e cada ligação entre os nodos representa os relacionamentos também presentes no modelo.

6.5. Migração de Dados

6.5.1 Exportação dos Dados

Para exportação dos dados da base de dados relacional para o Neo4J utilizamos uma funcionalidade que o MySQL Workbench fornece. O processo de exportação está apresentado abaixo pelas imagens

Abaixo está representado cada um dos scripts para cada ficheiro .csv (o primeiro select é para os clientes, o segundo para os filmes, etc).

```
USE FNAC;

SELECT * FROM Cliente;

SELECT Artigo.id_artigo, titulo, tipo, preco, ano, classificacao, duracao, realizador, genero
FROM Artigo JOIN Filme ON Filme.id_artigo = Artigo.id_artigo;

SELECT Artigo.id_artigo, titulo, tipo, preco, ano, classificacao, autor, genero, editora, n_paginas
FROM Artigo JOIN Livro ON Livro.id_artigo = Artigo.id_artigo;

SELECT Artigo.id_artigo, titulo, tipo, preco, ano, classificacao, plataforma, idade_min,
publisher, n_jogadores_max, genero
FROM Artigo JOIN Jogo ON Jogo.id_artigo = Artigo.id_artigo;

SELECT Artigo.id_artigo, titulo, tipo, preco, ano, classificacao, genero_musical, artista, formato
FROM Artigo JOIN Musica ON Musica.id_artigo = Artigo.id_artigo;

SELECT * FROM Stock;

SELECT * FROM Compra;

SELECT * FROM Compra_de_X_Artigos;

SELECT DISTINCT loja, distrito FROM Stock;
```

Figura 43 - Script MySQL para exportação

Seguidamente seleciona-se a parte do script que corresponde a cada um dos ficheiros e exporta-se utilizando o “Export/Import” dando o nome e formato desejado.

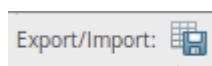


Figura 44 - Botão Exportação/Importação

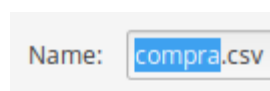


Figura 45 - Campo de inserção do nome

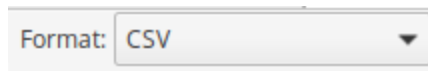


Figura 46 - Formato dos ficheiros de exportação

No final são estes os ficheiros gerados.

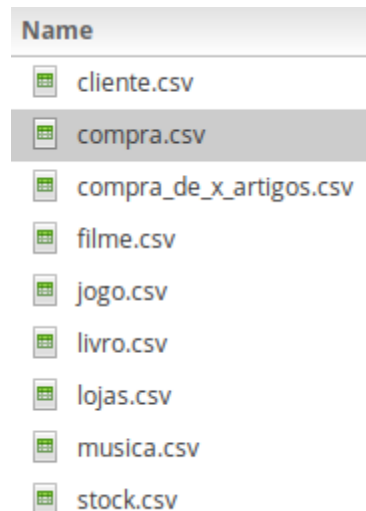


Figura 47 - Ficheiros Exportados

6.5.2 Criação dos Nodos

Efetuada a exportação e importação dos dados para o Neo4j, seguimos para a criação dos nodos. Para tal, foram criadas operações em Cypher Query Language que serão apresentadas de seguida.

Nodo Cliente

```
// Criar clientes
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:/cliente.csv" AS row
CREATE (:Cliente {id_cliente: toInteger(row.id_cliente), nome:
row.nome, data_nascimento: row.data_nascimento, data_subscricao:
row.data_subscricao, email: row.email, telemovel:
toInteger(row.telemovel), distrito: row.distrito});
```

Figura 48 - Criação do Nodo Cliente

Nodo Compra

```
// Criar compras
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:/compra.csv" AS row
CREATE (:Compra {id_compra: toInteger(row.id_compra), montante:
toFloat(row.montante), loja: row.loja, data_hora: row.data_hora,
id_cliente: toInteger(row.id_cliente)});
```

Figura 49 - Criação do Nodo Compra

Nodo Loja

```
// criar lojas
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:/lojas.csv" AS row
CREATE (:Loja {loja: row.loja, distrito: row.distrito});
```

Figura 50 - Criação do Nodo Loja

Nodo Filme

```
// Criar filmes
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:/filme.csv" AS row
CREATE (:Filme {id_artigo: toInteger(row.id_artigo), titulo:
row.titulo, preco: toFloat(row.preco), ano: toInteger(row.ano),
classificacao: toInteger(row.classificacao), duracao:
toInteger(row.duracao), realizador: row.realizador, genero:
row.genero});
```

Figura 51 - Criação do Nodo Filme

Nodo Livro

```
// Criar livros
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:/livro.csv" AS row
CREATE (:Livro {id_artigo: toInteger(row.id_artigo), titulo:
row.titulo, preco: toFloat(row.preco), ano: toInteger(row.ano),
classificacao: toInteger(row.classificacao), autor: row.autor,
genero: row.genero, editora: row.editora, n_paginas:
toInteger(row.n_paginas)});
```

Figura 52 - Criação do Nodo Livro

Nodo Jogo

```
// Criar jogos
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:/jogo.csv" AS row
CREATE (:Jogo {id_artigo: toInteger(row.id_artigo), titulo:
row.titulo, preco: toFloat(row.preco), ano:
toInteger(row.ano), classificacao: toInteger(row.classificacao),
plataforma: row.plataforma, idade_min: toInteger(row.idade_min),
publisher: row.publisher, n_jogadores_max:
toInteger(row.n_jogadores_max), genero: row.genero});
```

Figura 53 - Criação do Nodo Jogo

Nodo Musica

```
// Criar musicas
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:/musica.csv" AS row
CREATE (:Musica {id_artigo: toInteger(row.id_artigo), titulo:
row.titulo, preco: toFloat(row.preco), ano: toInteger(row.ano),
classificacao: toInteger(row.classificacao), genero_musical:
row.genero_musical, artista: row.artista, formato: row.formato});
```

Figura 54 - Criação do Nodo Musica

6.5.3 Criação dos Relacionamentos

Após a criação dos nodos é necessário criar os relacionamentos entre eles. Como foi feito na criação, também é usada a linguagem Cypher para serem elaboradas as operações.

Abaixo são apresentadas o que é usado para cada relacionamento.

Cliente – Compra

```
// cliente faz compra
MATCH (co:Compra),(cl:Cliente)
WHERE co.id_cliente = cl.id_cliente
CREATE (cl)-[f:FAZ]→(co)
RETURN cl, f, co;
```

Figura 55 - Criação do Relacionamento Cliente - Compra

Compra – Filme

```
// compra de x filmes
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:/compra_de_x_artigos.csv" AS row
MATCH (co:Compra),(f:Filme)
WHERE co.id_compra = toInteger(row.id_compra)
AND f.id_artigo = toInteger(row.id_artigo)
CREATE (co)-[de:DE {quantidade: toInteger(row.quantidade)}]→(f)
RETURN co, de, f;
```

Figura 56 - Criação do Relacionamento Compra - Filme

Compra – Livro

```
// compra de x livros
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:/compra_de_x_artigos.csv" AS row
MATCH (co:Compra),(l:Livro)
WHERE co.id_compra = toInteger(row.id_compra)
AND l.id_artigo = toInteger(row.id_artigo)
CREATE (co)-[de:DE {quantidade: toInteger(row.quantidade)}]→(l)
RETURN co, de, l;
```

Figura 57 - Criação do Relacionamento Compra - Livro

Compra – Jogo

```
// compra de x jogos
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:/compra_de_x_artigos.csv" AS row
MATCH (co:Compra),(j:Jogo)
WHERE co.id_compra = toInteger(row.id_compra)
AND j.id_artigo = toInteger(row.id_artigo)
CREATE (co)-[de:DE {quantidade: toInteger(row.quantidade)}]→(j)
RETURN co, de, j;
```

Figura 58 - Criação do Relacionamento Compra - Jogo

Compra – Musica

```
// compra de x musicas
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:/compra_de_x_artigos.csv" AS row
MATCH (co:Compra),(m:Musica)
WHERE co.id_compra = toInteger(row.id_compra)
AND m.id_artigo = toInteger(row.id_artigo)
CREATE (co)-[de:DE {quantidade: toInteger(row.quantidade)}]→(m)
RETURN co, de, m;
```

Figura 59 - Criação do Relacionamento Compra - Musica

Filme – Loja

```
// filme tem x stock na loja
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:/stock.csv" AS row
MATCH (f:Filme),(lo:Loja)
WHERE f.id_artigo = toInteger(row.id_artigo)
AND lo.loja = row.loja
CREATE (f)-[t:TEM_STOCK {qtd_disponivel:
toInteger(row.qtd_disponivel)}]→(lo)
RETURN f, t, lo;
```

Figura 60 - Criação do Relacionamento Filme - Loja

Livro – Loja

```
// livro tem x stock na loja
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:/stock.csv" AS row
MATCH (l:Livro),(lo:Loja)
WHERE l.id_artigo = toInteger(row.id_artigo)
AND lo.loja = row.loja
CREATE (l)-[t:TEM_STOCK {qtd_disponivel:
toInteger(row.qtd_disponivel)}]→(lo)
RETURN l, t, lo;
```

Figura 61 - Criação do Relacionamento Livro - Loja

Jogo – Loja

```
// jogo tem x stock na loja
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:/stock.csv" AS row
MATCH (j:Jogo),(lo:Loja)
WHERE j.id_artigo = toInteger(row.id_artigo)
AND lo.loja = row.loja
CREATE (j)-[t:TEM_STOCK {qtd_disponivel:
toInteger(row.qtd_disponivel)}]→(lo)
RETURN j, t, lo;
```

Figura 62 - Criação do Relacionamento Jogo - Loja

Musica – Loja

```
// musica tem x stock na loja
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:/stock.csv" AS row
MATCH (m:Musica),(lo:Loja)
WHERE m.id_artigo = toInteger(row.id_artigo)
AND lo.loja = row.loja
CREATE (m)-[t:TEM_STOCK {qtd_disponivel:
toInteger(row.qtd_disponivel)}]→(lo)
RETURN m, t, lo;
```

Figura 63 - Criação do Relacionamento Musica - Loja

6.6. Resolução das Queries Propostas

Nesta secção serão apresentadas as queries (Requisitos de Exploração) feitas na linguagem Cypher.

1. **Obter o número de clientes nas lojas todas**

```
// 11. Obter o número de clientes total

MATCH (c:Cliente) RETURN count(c);
```

Figura 64 - Requisito de exploração 11 em Cypher

2. **Obter o número de clientes num determinado distrito**

```
// 12. Consultar quais os clientes que vivem num dado distrito

MATCH (c:Cliente{distrito: 'Porto'}) return c;
```

Figura 65 - Requisito de exploração 12 em Cypher

3. **Consultar os artigos por tipo**

```
// 13. Obter o número de clientes que vivem em cada distrito

MATCH (c:Cliente)
RETURN c.distrito, count(c);
```

Figura 66 - Requisito de exploração 13 em Cypher

4. Consultar os artigos que não tem stock

```
// 14. Consultar os artigos por tipo

MATCH (f:Filme) RETURN f;
MATCH (l:Livro) RETURN l;
MATCH (j:Jogo) RETURN j;
MATCH (m:Musica) RETURN m;
```

Figura 67 - Requisito de exploração 14 em Cypher

5. Obter o top 3 dos clientes com mais artigos comprados

```
// 15. Consultar os artigos que não têm stock disponível

MATCH (a)-[r:TEM_STOCK{qtd_disponivel: 0}]-()
RETURN a, r, l;
```

Figura 68 - Requisito de exploração 15 em Cypher

6. Obter o top 3 dos clientes que mais dinheiro gastaram

```
// 16. Obter o top 3 dos clientes com mais artigos comprados

MATCH (cl:Cliente)-[f:FAZ]-(co:Compra)-[d:DE]-(x)
RETURN cl, sum(d.quantidade)
ORDER BY sum(d.quantidade) DESC
LIMIT 3;
```

Figura 69 - Requisito de exploração 16 em Cypher

7. Obter o top 3 artigos mais vendidos

```
// 17. Obter o top 3 dos clientes que mais dinheiro gastaram

MATCH (cl:Cliente)-[f:FAZ]-(co:Compra)
RETURN cl, sum(co.montante)
ORDER BY sum(co.montante) DESC
LIMIT 3;
```

Figura 70 - Requisito de exploração 17 em Cypher

8. **Obter o top 3 de filmes mais vendidos**

```
// 18. Obter o top 3 artigos mais vendidos

MATCH (co:Compra)-[d:DE]-(x)
RETURN x, sum(d.quantidade)
ORDER BY sum(d.quantidade) DESC
LIMIT 3;
```

Figura 71 - Requisito de exploração 18 em Cypher

9. **Obter o top 3 de livros mais vendidos**

```
// 19. Obter o top 3 artigos mais vendidos de um tipo específico

// Filme:
MATCH (co:Compra)-[d:DE]-(f:Filme)
RETURN f, sum(d.quantidade)
ORDER BY sum(d.quantidade) DESC
LIMIT 3;
// Jogo:
MATCH (co:Compra)-[d:DE]-(j:Jogo)
RETURN j, sum(d.quantidade)
ORDER BY sum(d.quantidade) DESC
LIMIT 3;
// Livro:
MATCH (co:Compra)-[d:DE]-(l:Livro)
RETURN l, sum(d.quantidade)
ORDER BY sum(d.quantidade) DESC
LIMIT 3;
// Musica:
MATCH (co:Compra)-[d:DE]-(m:Musica)
RETURN m, sum(d.quantidade)
ORDER BY sum(d.quantidade) DESC
LIMIT 3;
```

Figura 72 - Requisito de exploração 19 em Cypher

10. Obter o top 3 de músicas mais vendidas

```
// 20. Consultar a disponibilidade de um artigo em todas as lojas

MATCH (x{id_artigo: 1})-[t:TEM_STOCK]-(l:Loja)
RETURN x, t.qtd_disponivel, l;
```

Figura 73 - Requisito de exploração 20 em Cypher

11. Consultar livros de um autor específico

```
// 21. Consultar livros de um autor específico

MATCH (l:Livro{autor: 'John Green'})
RETURN l;
```

Figura 74 - Requisito de exploração 21 em Cypher

12. Verificar se o cliente tem idade mínima para o jogo que quer comprar

```
// 22. Verificar que jogos o cliente tem idade mínima para comprar

MATCH (j:Jogo),(c:Cliente{id_cliente: 1})
WHERE j.idade_min ≤
duration.between(date(c.data_nascimento),date()).years
RETURN j;
```

Figura 75 - Requisito de exploração 22 em Cypher

13. Verificar que quantidade e montante um autor específico vendeu

```
// 23. Análise das vendas de cada autor (quantidade e montante
totais)

MATCH (cl:Cliente)-[f:FAZ]-(co:Compra)-[d:DE]-(l:Livro)
RETURN l.autor, sum(d.quantidade), sum(co.montante)
ORDER BY sum(d.quantidade) DESC, sum(co.montante) DESC;
```

Figura 76 - Requisito de exploração 23 em Cypher

14. Consultar quanto foi vendido em uma loja específica num ano específico

```
// 24. Consultar quanto foi vendido numa loja específica num ano
específico

MATCH (c:Compra)
WHERE (datetime(REPLACE(c.data_hora, " ", "T"))).year = 2019
AND c.loja = "Gaia Shopping"
RETURN sum(c.montante);
```

Figura 77 - Requisito de exploração 24 em Cypher

15. Consultar quanto foi vendido em cada mês num ano específico

```
// 25. Consultar quanto foi vendido em cada mês num ano específico

MATCH (c:Compra)
WHERE (datetime(REPLACE(c.data_hora, " ", "T"))).year = 2019
RETURN (datetime(REPLACE(c.data_hora, " ", "T"))).month AS mes,
sum(c.montante)
ORDER BY mes;
```

Figura 78 - Requisito de exploração 25 em Cypher

16. Verificar a existência de stock de um certo artigo no distrito de um certo cliente

```
// 26. Verificar a existência de stock de um certo artigo no
distrito de um certo cliente

MATCH (c:Cliente{id_cliente:1}),(x{id_artigo: 1})-[t:TEM_STOCK]-
(l:Loja)
WHERE c.distrito = l.distrito
RETURN x, t.qtd_disponivel, l;
```

Figura 79 - Requisito de exploração 26 em Cypher

6.7. Base de Dados em Neo4j

Uma funcionalidade muito útil do Neo4j é a representação da base de dados em grafos, sendo possível visualizar os relacionamentos entre cada nodo e ter uma melhor percepção de como estão ligados entre eles. Além disso também mostra a quantidade de cada. Abaixo está a representação da nossa base de dados através dessa funcionalidade.



Figura 80 - Número de dados no grafo

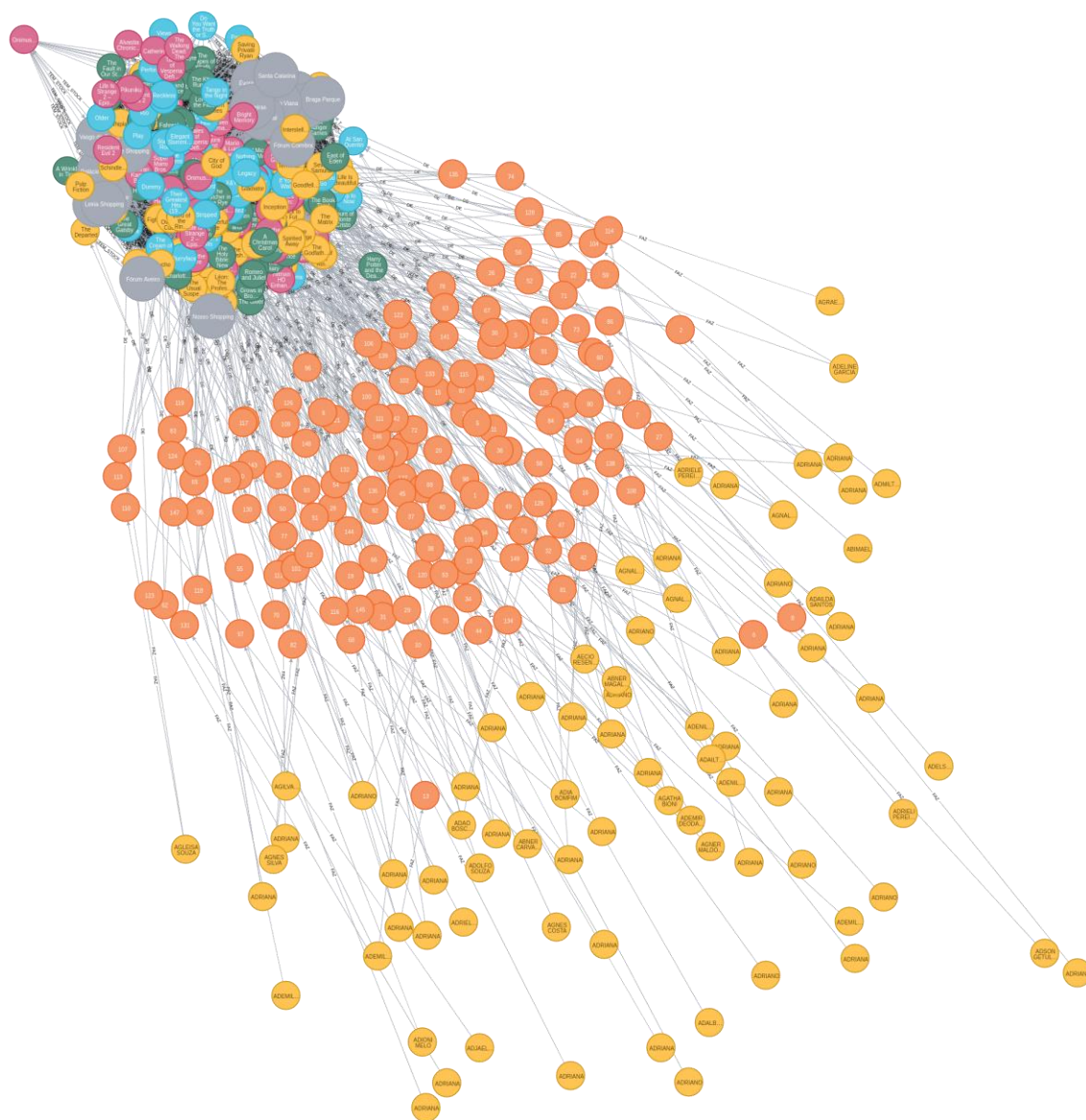


Figura 81 - Grafo da base de dados

7. Conclusões e Trabalho Futuro

Este projeto consiste em elaborar um sistema de base de dados para o site FNAC. Para tal foi efetuado um levantamento de requisitos, seguido da construção de um modelo Conceptual, modelo lógico e implementação do modelo físico. O desenvolvimento desta base de dados foi feito em paralelo com a administração da FNAC de forma a que o resultado final se molde o melhor possível à realidade.

A nossa maior dificuldade no trabalho foi a conversão de um modelo relacional para um modelo não relacional.

A realização deste projeto permitiu-nos seguir de próximo todos os passos de desenvolvimento de uma base de dados e aprofundar os conhecimentos da matéria lecionada.

Apesar dos problemas encontrados no desenvolvimento do projeto, conseguimos cumprir todos os objetivos propostos.

8. Referências

1. Thomas Connolly, Carolyn Begg 2015. Database Systems A Pratical Approach to Design, Implementation and Management. Sixth Edition / Global Edition ed. Harlow: Pearson Education Limited.
2. Feliz Gouveia 2014. Fundamentos de Bases de Dados, FCA.
3. Ian Robinson, Jim Webber and Emil Eifrem 2015. Graph Databases New Opportunities for Connected Data. Second Edition / O'Reilly Media.
4. <https://www.imdb.com/>

9. Lista de Siglas e Acrónimos

BD	Base de Dados
SGBD	Sistema de Gestão de Base de Dados
SQL	Structured Query Language
SBD	Sistema de Base de Dados
SBDR	Sistema de Base de Dados Relacional
NoSQL	Not only SQL

10. Anexos

1. Script Completo de Criação
2. Script Parcial do Povoamento
3. Queries em SQL
4. Exportação de Dados

10.1. Anexo 1 – Script Completo de Criação

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_T
ABLES,NO_ZERO_IN_DATE,
NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----
-- Schema FNAC
-- -----

-- -----
-- Schema FNAC
-- -----

CREATE SCHEMA IF NOT EXISTS `FNAC` ;
USE `FNAC` ;

-- -----
-- Table `FNAC`.`Cliente`
-- -----

CREATE TABLE IF NOT EXISTS `FNAC`.`Cliente` (
  `id_cliente` INT NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  `data_nascimento` DATE NOT NULL,
  `data_subscricao` DATE NOT NULL,
  `email` VARCHAR(45) NOT NULL,
```

```

    `telemovel` VARCHAR(45) NOT NULL,
    `distrito` VARCHAR(45) NOT NULL,
    PRIMARY KEY (`id_cliente`))
ENGINE = InnoDB;

```

```

-- -----
-- Table `FNAC`.`Compra`
-- -----

```

```

CREATE TABLE IF NOT EXISTS `FNAC`.`Compra` (
  `id_compra` INT NOT NULL,
  `montante` DECIMAL(6,2) NOT NULL,
  `loja` VARCHAR(45) NOT NULL,
  `data_hora` DATETIME NOT NULL,
  `id_cliente` INT NOT NULL,
  PRIMARY KEY (`id_compra`),
  INDEX `fk_Compra_Cliente_idx` (`id_cliente` ASC),
  CONSTRAINT `fk_Compra_Cliente`
    FOREIGN KEY (`id_cliente`)
    REFERENCES `FNAC`.`Cliente` (`id_cliente`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-- -----
-- Table `FNAC`.`Artigo`
-- -----

```

```

CREATE TABLE IF NOT EXISTS `FNAC`.`Artigo` (
  `id_artigo` INT NOT NULL COMMENT 'Aplicável a todos',
  `titulo` VARCHAR(100) NOT NULL COMMENT 'Aplicável a todos',
  `tipo` VARCHAR(10) NOT NULL,
  `preco` DECIMAL(4,2) NOT NULL COMMENT 'Aplicável a todos',
  `ano` INT NOT NULL COMMENT 'Aplicável a todos',
  `classificacao` INT NOT NULL COMMENT 'Classificação de 1-10\nAplicável a todos',
  PRIMARY KEY (`id_artigo`))
ENGINE = InnoDB;

```

```

-- -----
-- Table `FNAC`.`Stock`
-- -----

```

```

CREATE TABLE IF NOT EXISTS `FNAC`.`Stock` (
  `id_artigo` INT NOT NULL,
  `loja` VARCHAR(45) NOT NULL,
  `qtd_disponivel` INT NOT NULL,
  `distrito` VARCHAR(25) NOT NULL,
  PRIMARY KEY (`id_artigo`, `loja`),

```

```

CONSTRAINT `fk_Stock_Artigo1`
  FOREIGN KEY (`id_artigo`)
  REFERENCES `FNAC`.`Artigo` (`id_artigo`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table `FNAC`.`Compra_de_X_Artigos`
-----

```

```

CREATE TABLE IF NOT EXISTS `FNAC`.`Compra_de_X_Artigos` (
  `id_compra` INT NOT NULL,
  `id_artigo` INT NOT NULL,
  `quantidade` INT NOT NULL,
  INDEX `fk_Compra_has_Artigo_Artigo1_idx` (`id_artigo` ASC),
  INDEX `fk_Compra_has_Artigo_Compra1_idx` (`id_compra` ASC),
  CONSTRAINT `fk_Compra_has_Artigo_Compra1`
    FOREIGN KEY (`id_compra`)
    REFERENCES `FNAC`.`Compra` (`id_compra`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Compra_has_Artigo_Artigo1`
    FOREIGN KEY (`id_artigo`)
    REFERENCES `FNAC`.`Artigo` (`id_artigo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table `FNAC`.`Filme`
-----

```

```

CREATE TABLE IF NOT EXISTS `FNAC`.`Filme` (
  `id_artigo` INT NOT NULL,
  `duracao` INT NOT NULL COMMENT 'Duração de um filme em minutos',
  `realizador` VARCHAR(45) NOT NULL COMMENT 'Falta mostrar atores\nAplicá
vel ao Filme',
  `genero` VARCHAR(25) NOT NULL COMMENT 'Género -
aplicável ao livro, filme, ou jogo.',
  PRIMARY KEY (`id_artigo`),
  CONSTRAINT `fk_Filme_Artigo1`
    FOREIGN KEY (`id_artigo`)
    REFERENCES `FNAC`.`Artigo` (`id_artigo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-- -----
-- Table `FNAC`.`Musica`
-- -----
CREATE TABLE IF NOT EXISTS `FNAC`.`Musica` (
  `id_artigo` INT NOT NULL,
  `genero_musical` VARCHAR(25) NOT NULL COMMENT 'Género - Musica',
  `artista` VARCHAR(45) NOT NULL COMMENT 'Artista - Musica',
  `formato` VARCHAR(25) NOT NULL COMMENT 'CD\nVinil\n...\nAplicável à Música',
  PRIMARY KEY (`id_artigo`),
  CONSTRAINT `fk_Musica_Artigo1`
    FOREIGN KEY (`id_artigo`)
    REFERENCES `FNAC`.`Artigo` (`id_artigo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-- -----
-- Table `FNAC`.`Jogo`
-- -----
CREATE TABLE IF NOT EXISTS `FNAC`.`Jogo` (
  `id_artigo` INT NOT NULL,
  `plataforma` VARCHAR(15) NOT NULL COMMENT 'PC\nPS4\nXBOX\n...\nAplicável ao Jogo',
  `idade_min` INT NOT NULL COMMENT 'Idade mínima permitida\nAplicável a Jogo e Filme',
  `publisher` VARCHAR(45) NOT NULL COMMENT 'Aplicável ao Jogo',
  `n_jogadores_max` INT NOT NULL COMMENT 'Aplicável ao Jogo',
  `genero` VARCHAR(45) NOT NULL COMMENT 'Género - aplicável ao livro, filme, ou jogo.',
  PRIMARY KEY (`id_artigo`),
  CONSTRAINT `fk_Jogo_Artigo1`
    FOREIGN KEY (`id_artigo`)
    REFERENCES `FNAC`.`Artigo` (`id_artigo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-- -----
-- Table `FNAC`.`Livro`
-- -----
CREATE TABLE IF NOT EXISTS `FNAC`.`Livro` (
  `id_artigo` INT NOT NULL,
  `autor` VARCHAR(45) NOT NULL COMMENT 'Autor do livro',
  `genero` VARCHAR(25) NOT NULL COMMENT 'Género - aplicável ao livro, filme, ou jogo.',
  `editora` VARCHAR(45) NOT NULL COMMENT 'Editora do livro',

```

```

`n_paginas` INT NOT NULL COMMENT 'Nº de páginas do livro',
PRIMARY KEY (`id_artigo`),
CONSTRAINT `fk_Livro_Artigo1`
FOREIGN KEY (`id_artigo`)
REFERENCES `FNAC`.`Artigo` (`id_artigo`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```

10.2. Anexo 2 – Script Parcial do Povoamento

```

Use FNAC;
INSERT INTO FNAC.Cliente SELECT 1, 'ABIMAEI CARDOSO', '1968-04-06', '2014-03-11', 'abimael.cardoso@exemplo.com', 978801951, 'Aveiro';
INSERT INTO FNAC.Cliente SELECT 2, 'ABNER CARVALHO', '1977-03-19', '2015-11-06', 'abner.carvalho@exemplo.com', 907328148, 'Beja';
INSERT INTO FNAC.Cliente SELECT 3, 'ABNER MAGALHAES', '1968-04-03', '2015-01-01', 'abner.magalhaes@exemplo.com', 986408633, 'Braga';
INSERT INTO FNAC.Cliente SELECT 4, 'ADAILDA SANTOS', '1984-01-04', '2012-12-07', 'adailda.santos@exemplo.com', 942719137, 'Bragança';
INSERT INTO FNAC.Cliente SELECT 5, 'ADAILTON ANICETO', '1974-10-23', '2013-11-19', 'adailton.aniceto@exemplo.com', 983237013, 'Castelo Branco';
INSERT INTO FNAC.Cliente SELECT 6, 'ADALBERTO SOUSA', '1979-07-12', '2013-03-14', 'adalberto.sousa@exemplo.com', 933359662, 'Coimbra';
INSERT INTO FNAC.Cliente SELECT 7, 'ADAO BOSCOLO', '1972-09-03', '2019-02-24', 'adao.boscolo@exemplo.com', 937931456, 'Évora';
INSERT INTO FNAC.Cliente SELECT 8, 'ADELINE GARCIA', '1997-08-31', '2016-11-02', 'adeline.garcia@exemplo.com', 976119044, 'Faro';
.
.
.

```

```

Use FNAC;
INSERT INTO FNAC.Compra (id_compra, montante, loja, data_hora, id_cliente) VALUES (1,0, 'Colombo', '2019-04-18 0:41:26', 3615);
INSERT INTO FNAC.Compra (id_compra, montante, loja, data_hora, id_cliente) VALUES (2,0, 'Fórum Algarve', '2017-03-15 5:9:5', 1600);
INSERT INTO FNAC.Compra (id_compra, montante, loja, data_hora, id_cliente) VALUES (3,0, 'Fórum Aveiro', '2015-01-31 1:3:36', 1851);

```



```

INSERT INTO FNAC.Compra (id_compra, montante, loja, data_hora, id_cliente
) VALUES (4,0,'Palácio do Gelo','2015-09-12 13:53:28',1489);
INSERT INTO FNAC.Compra (id_compra, montante, loja, data_hora, id_cliente
) VALUES (5,0,'Évora Plaza','2013-10-09 15:58:30',2779);
INSERT INTO FNAC.Compra (id_compra, montante, loja, data_hora, id_cliente
) VALUES (6,0,'Aeroporto','2013-01-10 2:15:37',2166);
INSERT INTO FNAC.Compra (id_compra, montante, loja, data_hora, id_cliente
) VALUES (7,0,'Aeroporto','2014-08-31 18:36:56',5436);
INSERT INTO FNAC.Compra (id_compra, montante, loja, data_hora, id_cliente
) VALUES (8,0,'Estação Viana','2015-05-03 9:53:54',758);
.
.
.

```

Use FNAC;

```

INSERT INTO FNAC.Artigo (id_artigo,titulo,tipo,preco,ano,classificacao) V
ALUES (1273,'The Shawshank Redemption','Filme',12.9,1994,9.3);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipo,preco,ano,classificacao) V
ALUES (1274,'The Godfather','Filme',19.1,1972,9.2);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipo,preco,ano,classificacao) V
ALUES (1275,'The Godfather: Part II','Filme',19.5,1974,9);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipo,preco,ano,classificacao) V
ALUES (1276,'The Dark Knight','Filme',15.1,2008,9);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipo,preco,ano,classificacao) V
ALUES (1277,'12 Angry Men','Filme',20.9,1957,8.9);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipo,preco,ano,classificacao) V
ALUES (1278,'Schindler's List','Filme',11.6,1993,8.9);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipo,preco,ano,classificacao) V
ALUES (1279,'The Lord of the Rings: The Return of the King','Filme',20.5,
2003,8.9);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipo,preco,ano,classificacao) V
ALUES (1280,'Pulp Fiction','Filme',20.1,1994,8.9);
.
.
.

```

Use FNAC;

```

INSERT INTO FNAC.Artigo (id_artigo,titulo,tipo,preco,ano,classificacao) V
ALUES (119,'Catherine Classic','Jogo',25.4,2019,5.8);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipo,preco,ano,classificacao) V
ALUES (120,'Hitman HD Enhanced Collection','Jogo',48.3,2019,5.3);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipo,preco,ano,classificacao) V
ALUES (121,'Hitman HD Enhanced Collection','Jogo',48.3,2019,5.7);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipo,preco,ano,classificacao) V
ALUES (122,'Mario & Luigi: Bowser's Inside Story + Bowser Jr.'s Journey',
'Jogo',20.4,2019,4.5);

```

```

INSERT INTO FNAC.Artigo (id_artigo,titulo,tipopreco,ano,classificacao) V
ALUES (123,'New Super Mario Bros. U Deluxe','Jogo',47.9,2019,4.4);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipopreco,ano,classificacao) V
ALUES (124,'Tales of Vesperia: Definitive Edition','Jogo',43.5,2019,3.6);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipopreco,ano,classificacao) V
ALUES (125,'Tales of Vesperia: Definitive Edition','Jogo',43.5,2019,3.7);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipopreco,ano,classificacao) V
ALUES (126,'Tales of Vesperia: Definitive Edition','Jogo',43.5,2019,5.5);
.
.
.

```

Use FNAC;

```

INSERT INTO FNAC.Artigo (id_artigo,titulo,tipopreco,ano,classificacao) V
ALUES (1,'To Kill a Mockingbird','Livro',16.3,1997,5.3);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipopreco,ano,classificacao) V
ALUES (2,'Pride and Prejudice','Livro',16.8,2000,4.3);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipopreco,ano,classificacao) V
ALUES (3,'Anne Frank: The Diary of a Young Girl','Livro',12.6,1956,5.3);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipopreco,ano,classificacao) V
ALUES (4,'1984','Livro',14.9,1968,4.6);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipopreco,ano,classificacao) V
ALUES (5,'Harry Potter and the Sorcerer's Stone','Livro',14.7,2009,4.9);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipopreco,ano,classificacao) V
ALUES (6,'The Lord of the Rings','Livro',14,2010,4.6);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipopreco,ano,classificacao) V
ALUES (7,'The Great Gatsby','Livro',17.6,2007,5.3);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipopreco,ano,classificacao) V
ALUES (8,'Charlotte's Web','Livro',15.4,1980,4.3);
.
.
.

```

Use FNAC;

```

INSERT INTO FNAC.Artigo (id_artigo,titulo,tipopreco,ano,classificacao) V
ALUES (1004,'Legacy','Musica',3.8,2016,3.7);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipopreco,ano,classificacao) V
ALUES (1005,'Because of the Times','Musica',15.3,2007,5.7);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipopreco,ano,classificacao) V
ALUES (1006,'Do You Want the Truth or Something Beautiful?','Musica',19.4
,2009,3.6);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipopreco,ano,classificacao) V
ALUES (1007,'The Cream of Eric Clapton','Musica',6.3,1987,4.5);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipopreco,ano,classificacao) V
ALUES (1008,'Tango in the Night','Musica',15.2,1987,4.3);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tipopreco,ano,classificacao) V
ALUES (1009,'Blood Sugar Sex Magik','Musica',28.6,1991,5.3);

```

```

INSERT INTO FNAC.Artigo (id_artigo,titulo,tip,preco,ano,classificacao) V
ALUES (1010,'Performance and Cocktails','Musica',7,1999,3.1);
INSERT INTO FNAC.Artigo (id_artigo,titulo,tip,preco,ano,classificacao) V
ALUES (1011,'No Parlez','Musica',15.6,1983,3.3);

```

```

.
.
.

```

```

INSERT INTO FNAC.Stock (id_artigo,loja,qtd_disponivel,distrito) VALUES (1
,'Aeroporto',25,'Lisboa');
INSERT INTO FNAC.Stock (id_artigo,loja,qtd_disponivel,distrito) VALUES (1
,'Amoreiras',12,'Lisboa');
INSERT INTO FNAC.Stock (id_artigo,loja,qtd_disponivel,distrito) VALUES (1
,'Colombo',12,'Lisboa');
INSERT INTO FNAC.Stock (id_artigo,loja,qtd_disponivel,distrito) VALUES (1
,'Fórum Coimbra',20,'Coimbra');
INSERT INTO FNAC.Stock (id_artigo,loja,qtd_disponivel,distrito) VALUES (1
,'Fórum Aveiro',15,'Aveiro');
INSERT INTO FNAC.Stock (id_artigo,loja,qtd_disponivel,distrito) VALUES (1
,'Braga Parque',15,'Braga');
INSERT INTO FNAC.Stock (id_artigo,loja,qtd_disponivel,distrito) VALUES (1
,'Évora Plaza',1,'Évora');
INSERT INTO FNAC.Stock (id_artigo,loja,qtd_disponivel,distrito) VALUES (1
,'Fórum Algarve',20,'Faro');

```

```

.
.
.

```

Use FNAC;

```

INSERT INTO FNAC.Filme (id_artigo, duracao, realizador, genero) VALUES (1
273,142,'Frank Darabont','Drama');
INSERT INTO FNAC.Filme (id_artigo, duracao, realizador, genero) VALUES (1
274,175,'Francis Ford Coppola','Crime');
INSERT INTO FNAC.Filme (id_artigo, duracao, realizador, genero) VALUES (1
275,202,'Francis Ford Coppola','Crime');
INSERT INTO FNAC.Filme (id_artigo, duracao, realizador, genero) VALUES (1
276,152,'Christopher Nolan','Action');
INSERT INTO FNAC.Filme (id_artigo, duracao, realizador, genero) VALUES (1
277,96,'Sidney Lumet','Drama');
INSERT INTO FNAC.Filme (id_artigo, duracao, realizador, genero) VALUES (1
278,195,'Steven Spielberg','Biography');
INSERT INTO FNAC.Filme (id_artigo, duracao, realizador, genero) VALUES (1
279,201,'Peter Jackson','Adventure');
INSERT INTO FNAC.Filme (id_artigo, duracao, realizador, genero) VALUES (1
280,154,'Quentin Tarantino','Crime');

```

```

.
.

```

```

Use FNAC;
INSERT INTO FNAC.Jogo (id_artigo,plataforma,idade_min,publisher,n_jogadores_max,genero) VALUES (119,'Win',13,'Sega',3,'Puzzle-platform');
INSERT INTO FNAC.Jogo (id_artigo,plataforma,idade_min,publisher,n_jogadores_max,genero) VALUES (120,'PS4',14,'Warner Bros. Interactive Entertainment',2,'Stealth');
INSERT INTO FNAC.Jogo (id_artigo,plataforma,idade_min,publisher,n_jogadores_max,genero) VALUES (121,'XBO',14,'Warner Bros. Interactive Entertainment',2,'Stealth');
INSERT INTO FNAC.Jogo (id_artigo,plataforma,idade_min,publisher,n_jogadores_max,genero) VALUES (122,'3DS',13,'Nintendo',4,'Role-playing');
INSERT INTO FNAC.Jogo (id_artigo,plataforma,idade_min,publisher,n_jogadores_max,genero) VALUES (123,'NS',15,'Nintendo',3,'Platformer');
INSERT INTO FNAC.Jogo (id_artigo,plataforma,idade_min,publisher,n_jogadores_max,genero) VALUES (124,'Win',18,'Bandai Namco Entertainment',3,'Role-playing');
INSERT INTO FNAC.Jogo (id_artigo,plataforma,idade_min,publisher,n_jogadores_max,genero) VALUES (125,'NS',18,'Bandai Namco Entertainment',3,'Role-playing');
INSERT INTO FNAC.Jogo (id_artigo,plataforma,idade_min,publisher,n_jogadores_max,genero) VALUES (126,'PS4',18,'Bandai Namco Entertainment',3,'Role-playing');
.
.
.

```

```

Use FNAC;
INSERT INTO FNAC.Livro (id_artigo,autor,genero,editora,n_paginas) VALUES (1,'Harper Lee','Classics','Porto Editora',226);
INSERT INTO FNAC.Livro (id_artigo,autor,genero,editora,n_paginas) VALUES (2,'Jane Austen','Classics','Antígona',391);
INSERT INTO FNAC.Livro (id_artigo,autor,genero,editora,n_paginas) VALUES (3,'Anne Frank','Classics','Harper Collins',246);
INSERT INTO FNAC.Livro (id_artigo,autor,genero,editora,n_paginas) VALUES (4,'George Orwell','Dystopian','Penguin Books',376);
INSERT INTO FNAC.Livro (id_artigo,autor,genero,editora,n_paginas) VALUES (5,'J.K. Rowling','Fantasy','Cavalo de Ferro',390);
INSERT INTO FNAC.Livro (id_artigo,autor,genero,editora,n_paginas) VALUES (6,'J.R.R. Tolkien','Fantasy','Porto Editora',318);
INSERT INTO FNAC.Livro (id_artigo,autor,genero,editora,n_paginas) VALUES (7,'F. Scott Fitzgerald','Classics','Usborne Publishing',318);
INSERT INTO FNAC.Livro (id_artigo,autor,genero,editora,n_paginas) VALUES (8,'E.B. White','Classics','Editorial Presença',280);
.
.

```

```

Use FNAC;
INSERT INTO FNAC.Musica (id_artigo, genero_musical, artista, formato) VAL
UES (1004,'Indie Rock','David Bowie','MP3');
INSERT INTO FNAC.Musica (id_artigo, genero_musical, artista, formato) VAL
UES (1005,'Indie Rock','Kings of Leon','CD');
INSERT INTO FNAC.Musica (id_artigo, genero_musical, artista, formato) VAL
UES (1006,'Techno','Paloma Faith','Vinil');
INSERT INTO FNAC.Musica (id_artigo, genero_musical, artista, formato) VAL
UES (1007,'Indie Rock','Eric Clapton & Cream','MP3');
INSERT INTO FNAC.Musica (id_artigo, genero_musical, artista, formato) VAL
UES (1008,'Rock','Fleetwood Mac','CD');
INSERT INTO FNAC.Musica (id_artigo, genero_musical, artista, formato) VAL
UES (1009,'Pop','Red Hot Chili Peppers','Vinil');
INSERT INTO FNAC.Musica (id_artigo, genero_musical, artista, formato) VAL
UES (1010,'Rock','Stereophonics','MP3');
INSERT INTO FNAC.Musica (id_artigo, genero_musical, artista, formato) VAL
UES (1011,'Indie Rock','Paul Young','CD');

```

```

Use FNAC;
INSERT INTO FNAC.Compra_de_X_Artigos (id_compra, id_artigo, quantidade) V
ALUES (1,431,6);
INSERT INTO FNAC.Compra_de_X_Artigos (id_compra, id_artigo, quantidade) V
ALUES (1,973,2);
INSERT INTO FNAC.Compra_de_X_Artigos (id_compra, id_artigo, quantidade) V
ALUES (1,160,2);
INSERT INTO FNAC.Compra_de_X_Artigos (id_compra, id_artigo, quantidade) V
ALUES (2,975,1);
INSERT INTO FNAC.Compra_de_X_Artigos (id_compra, id_artigo, quantidade) V
ALUES (2,982,3);
INSERT INTO FNAC.Compra_de_X_Artigos (id_compra, id_artigo, quantidade) V
ALUES (3,250,1);
INSERT INTO FNAC.Compra_de_X_Artigos (id_compra, id_artigo, quantidade) V
ALUES (3,234,9);
INSERT INTO FNAC.Compra_de_X_Artigos (id_compra, id_artigo, quantidade) V
ALUES (3,588,2);

```

10.3. Anexo 3 – Queries em SQL

Use FNAC;

```
-- 11. Obter o número de clientes total
drop procedure if exists contagem_clientes;
DELIMITER //
create procedure contagem_clientes()
begin
    select count(id_cliente) from Cliente;
end //
DELIMITER ;
```

call contagem_clientes();

```
-- 12. Consultar quais os clientes que vivem num dado distrito
drop procedure if exists clientes_distrito;
DELIMITER //
create procedure clientes_distrito(in dist varchar(45))
begin
    select * from Cliente
    where distrito = dist;
end //
DELIMITER ;
```

call clientes_distrito('Porto');

```
-- 13. Obter o número de clientes que vivem em cada distrito
drop procedure if exists clientes_por_distrito;
DELIMITER //
create procedure clientes_por_distrito()
begin
    select distrito, count(id_cliente)
    from Cliente
    group by distrito;
end //
DELIMITER ;
```

call clientes_por_distrito();

```
-- 14. Consultar os artigos por tipo
```

```
--
```

```
    tipo especifico, ordenado por preço ascendente, classificacao descendente,
    com limites inferior e superior de preço
drop procedure if exists pesquisa_por_tipo;
DELIMITER //
```

```

create procedure pesquisa_por_tipo(in tipo_artigo varchar(45), in inf int
, in sup int)
begin
select titulo as Título, preco as Preço,
ano as Publicado, classificacao as Classificação
from Artigo
where tipo = tipo_artigo and preco between inf and sup
order by preco ASC, classificacao DESC;
end //
DELIMITER ;

```

```

call pesquisa_por_tipo('Livro',10,20);

```

-- 15. Consultar os artigos que não têm stock disponível

```

drop procedure if exists stock_indisponivel;
DELIMITER //
create procedure stock_indisponivel()
begin
select a.id_artigo, a.titulo, a.tipo, a.preco, s.loja, s.distrato fro
m Artigo a
join Stock s on a.id_artigo = s.id_artigo
where s.qtd_disponivel = 0;
end //
DELIMITER ;

```

```

call stock_indisponivel();

```

-- 16. Obter o top 3 dos clientes com mais artigos comprados

```

drop procedure if exists top_clientes_quantidade;
DELIMITER //
create procedure top_clientes_quantidade()
begin
select cl.id_cliente, cl.nome, sum(cx.quantidade) as `quantidade tota
l` from Cliente cl
join Compra co on cl.id_cliente = co.id_cliente
join Compra_de_X_Artigos cx on co.id_compra = cx.id_compra
group by cl.id_cliente
order by sum(cx.quantidade) DESC
limit 3;
end //
DELIMITER ;

```

```

call top_clientes_quantidade();

```

-- 17. Obter o top 3 dos clientes que mais dinheiro gastaram

```

drop procedure if exists top_clientes_montante;
DELIMITER //
create procedure top_clientes_montante()
begin

```

```

        select cl.id_cliente, cl.nome, sum(co.montante) as `montante total` f
rom Cliente cl
    join Compra co on cl.id_cliente = co.id_cliente
    group by cl.id_cliente
    order by sum(co.montante) DESC
    limit 3;
end //
DELIMITER ;

call top_clientes_montante();

-- 18. Obter o top 3 artigos mais vendidos
drop view if exists top_artigos;
create view top_artigos as
    select a.id_artigo, titulo, tipo, preco, sum(co.montante) as `montant
e total` from Artigo a
    join Compra_de_X_Artigos cx on cx.id_artigo = a.id_artigo
    join Compra co on co.id_compra = cx.id_compra
    group by a.id_artigo
    order by sum(co.montante) DESC
    limit 3;

select * from top_artigos;

-- 19. Obter o top 3 artigos mais vendidos de um tipo específico
drop procedure if exists top_artigos_tipo;
DELIMITER //
create procedure top_artigos_tipo(in t varchar(45))
begin
    select a.id_artigo, titulo, tipo, preco, sum(co.montante) as `montant
e total` from Artigo a
    join Compra_de_X_Artigos cx on cx.id_artigo = a.id_artigo
    join Compra co on co.id_compra = cx.id_compra
    group by a.id_artigo having a.tipo = t
    order by sum(co.montante) DESC
    limit 3;
end //
DELIMITER ;

call top_artigos_tipo('Livro');

-- 20. Consultar a disponibilidade de um artigo em todas as lojas
drop procedure if exists verifica_stock;
DELIMITER //
create procedure verifica_stock(in id int)
begin
    select * from Stock where id_artigo = id;
end //
DELIMITER ;

```



```

call verifica_stock(1);

-- 21. Consultar livros de um autor específico
drop procedure if exists livros_autor;
DELIMITER //
create procedure livros_autor(in autor varchar(45))
begin
select a.titulo as Título, a.preco as Preço,
a.ano as Publicado, a.classificacao as Classificação,
l.genero as Género, l.editora as Editora, l.n_paginas as Nº_Páginas
from Livro l join Artigo a on a.id_artigo = l.id_artigo
where l.autor = autor
order by a.titulo ASC;
end //
DELIMITER ;

call livros_autor('Jane Austen');

-- 22. Verificar que jogos o cliente tem idade mínima para comprar
drop procedure if exists jogos_permitidos;
DELIMITER //
create procedure jogos_permitidos(in id int)
begin
select * from jogos_titulo_asc j, Cliente c
where c.id_cliente = id
and idade(c.data_nascimento) >= j.`Idade Mínima`;
end //
DELIMITER ;

call jogos_permitidos(1);

-- 23. quanto cada autor já vendeu no total
-- (quantidade e montante total)
-- ordenado por lucro total e qtd decrescente

drop view if exists top_vendas_autor;
create view top_vendas_autor as
select l.autor as Autor, sum(a.preco*c.quantidade) as `Montante Total`, s
um(c.quantidade) as `Quantidade Total`
from Compra_de_X_Artigos c
join Artigo a on a.id_artigo = c.id_artigo
join Livro l on c.id_artigo = l.id_artigo
group by l.autor
order by `Montante Total` DESC, `Quantidade Total` DESC;

select * from top_vendas_autor;

```

```

-- 24. Consultar quanto foi vendido numa loja específica num dado ano
drop procedure if exists vendas_loja;
DELIMITER //
create procedure vendas_loja(in ano int)
begin
    select c.loja as Loja, sum(x.quantidade) as Quantidade, sum(c.montant
e) as `Total Faturado`
    from Compra c join Compra_de_X_Artigos x on c.id_compra = x.id_compra
    where year(c.data_hora) = ano
    group by c.loja;
end //
DELIMITER ;

call vendas_loja(2019);

-- 25. Consultar quanto foi vendido em cada mês num ano específico
-- quanto foi vendido (qtd e preco) em cada mês no ano x
drop procedure if exists vendas_mes;
DELIMITER //
create procedure vendas_mes(in ano int)
begin
    select month(c.data_hora) as Mês, sum(x.quantidade) as Quantidade, su
m(c.montante) as `Total Faturado`
    from Compra c join Compra_de_X_Artigos x on c.id_compra = x.id_compra
    where year(c.data_hora) = ano
    group by month(c.data_hora);
end //
DELIMITER ;

call vendas_mes(2019);

--
26. Verificar a existência de stock de um certo artigo no distrito de um
certo cliente
--
verifica a existência de stock de um artigo Y perto de um cliente X (no
mesmo distrito)
drop procedure if exists stock_near_me;
DELIMITER //
create procedure stock_near_me(in id_c int, in id_a int)
begin
    select s.loja as Loja, s.qtd_disponivel as `Stock`
    from Cliente c, Stock s
    where s.id_artigo = id_a
    and c.id_cliente = id_c
    and s.districto = c.districto
    and s.qtd_disponivel > 0;
end //
DELIMITER ;

```

```

call stock_near_me(11,1);

--
27. Fazer análise das vendas de cada ano (loja que lucrou mais: qual o L
ucro e a média entre lojas de cada ano)
-- por cada ano:
-- que loja lucrou mais?
-- quanto lucro?
-- qual foi a média entre lojas desse ano?
drop procedure if exists analise_anual;
DELIMITER //
create procedure analise_anual()
begin

    declare ano int;
    declare done bool;

    declare curs1 cursor for
        select year(data_hora) from Compra group by year(data_hora);
    declare continue handler for not found set done = true;

    drop table if exists `Análise Anual`;
    create table `Análise Anual` (`Ano` int, `Loja` varchar(45), `Maior L
ucro` decimal(7,2), `Média` decimal(6,2));

    open curs1;
    loop_name: loop
    fetch curs1 into ano;

    if done then
        leave loop_name;
    end if;

    insert into `Análise Anual` (Ano, Loja, `Maior Lucro`, `Média`)
        (select ano, get_loja.l, results.x, results.average
        from (select ano, max(total) as x, avg(total) as average
            from (select loja, sum(montante) as total
                from Compra
                where year(data_hora) = ano
                group by loja
            ) as linha
        ) as results
    join (select loja as l, ano, sum(montante) as total
        from Compra
        where ano = year(data_hora)
        group by year(data_hora), loja
    ) as get_loja
    on results.x = get_loja.total

```

```

);

end loop;
close curs1;

select * from `Análise Anual`;
drop table `Análise Anual`;

end //
DELIMITER ;

call analise_anual();

```

10.4. Anexo 4 – Exportação de Dados

```

USE FNAC;

SELECT * FROM Cliente;

SELECT Artigo.id_artigo, titulo, tipo, preco, ano, classificacao, duracao
, realizador, genero
FROM Artigo JOIN Filme ON Filme.id_artigo = Artigo.id_artigo;

SELECT Artigo.id_artigo, titulo, tipo, preco, ano, classificacao, autor,
genero, editora, n_paginas
FROM Artigo JOIN Livro ON Livro.id_artigo = Artigo.id_artigo;

SELECT Artigo.id_artigo, titulo, tipo, preco, ano, classificacao, platafo
rma, idade_min,
publisher, n_jogadores_max, genero
FROM Artigo JOIN Jogo ON Jogo.id_artigo = Artigo.id_artigo;

SELECT Artigo.id_artigo, titulo, tipo, preco, ano, classificacao, genero_
musical, artista, formato
FROM Artigo JOIN Musica ON Musica.id_artigo = Artigo.id_artigo;

SELECT * FROM Stock;

SELECT * FROM Compra;

SELECT * FROM Compra_de_X_Artigos;

SELECT DISTINCT loja, distrito FROM Stock;

```