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

Unidade Curricular de Bases de Dados

Ano Letivo de 2020/2021

Livraria Online

Eduardo Silva, A89516

José Nuno Martins,A90122

Maria Sofia Jordão Marques, A87963

Pedro Nogueira Pereira, A89232

BD

Novembro, 2020

Data de Receção	
Responsável	
Avaliação	
Observações	

Livraria Online

Eduardo Silva, A89516

José Nuno Martins, A90122

Maria Sofia Jordão Marques, A87963

Pedro Nogueira Pereira, A89232

Novembro, 2020

Resumo

O projeto apresentado neste relatório consiste na implementação de um sistema de base de dados referente ao processo venda de livros de uma livraria *online*.

Numa primeira fase começamos por analisar melhor o problema para compreendermos o que estaria em causa. De seguida, levantamos os requisitos fundamentais para o sistema, identificando assim as principais entidades, os seus atributos e os respetivos relacionamentos entre estas.

Após a realização do modelo conceptual, passámos à realização do modelo lógico, representando as tabelas correspondentes a cada entidade. Seguidamente passámos à realização e implementação do modelo físico, através do *MySQL*. Apresentámos o desenho da representação física e analisámos as principais transações existentes.

Após esta análise, realizámos uma estimativa do espaço necessário em disco para a fase inicial da base de dados. Após concluída esta etapa, efetuámos o povoamento da Base de Dados. Por último, definimos as restrições e as vistas dos utilizadores, bem como as regras de acesso à Base de Dados.

Área de Aplicação: Desenho e arquitetura de Sistemas de Bases de Dados para gestão de livraria *online*.
Palavras-Chave: Bases de Dados Relacionais, Livro, Encomenda, Leitor, Entidade, Atributo, Relacionamento.

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

1.1. Contextualização

A livraria situada em Guimarães pertence à família Jordão há várias gerações. Tendo em conta a longa data de negócio existe uma basta rede de clientes habituais e, portanto, nunca houve a necessidade de utilizar grandes tecnologias para gerir o negócio.

No entanto, com o crescimento do negócio e a possível exploração de outros meios por onde o expandir, tornou-se cada vez mais complicado a sua gestão.

1.2. Motivação e Objetivos

Cada vez mais nos dias de hoje a internet passou a ser um ponto essencial na vida das pessoas, desde um meio de comunicação a um meio de compras e vendas. Hoje em dia facilmente podemos encontrar tudo na internet, e fazer tudo sem ter de sair do conforto da nossa casa.

Na atualidade, onde ficar em casa é uma necessidade, cada vez mais se encontram pequenos e grandes negócios a redirecionar o seu comércio para os meios digitais de forma a se manterem ativos num período de grandes incertezas e flutuações no mercado.

Assim sendo a Sra. Alice decidiu tomar esse passo em frente e levar a sua livraria para o meio digital e implementar assim um sistema que tornasse mais fácil a gestão do seu negócio. Neste sentido, surge a necessidade de guardar, numa base de dados, a informação relativa aos livros, bem como sobre os clientes que os compram. Para além disso, é fundamental guardar também a informação de todas as transações efetuadas.

Outra das motivações para a realização deste projeto é também de extrema importância e prende-se com a necessidade de organizar toda esta informação de forma a existir uma consulta e análise dos dados mais eficientes.

O principal objetivo na elaboração deste projeto passa por implementar um sistema de gestão de bases de dados relacional que possibilite um ambiente apropriado e eficiente, capaz de armazenar toda a informação pertencente aos diferentes livros, clientes e também das suas encomendas. Uma vez que falamos em transações *online* também é percetível a necessidade de a solução implementada ser segura, garantindo a integridade, disponibilidade e a confidencialidade da informação.

1.3. Estrutura do Relatório

Numa parte inicial do relatório vão ser explicados todos os detalhes relacionados com a análise de requisitos que permitiu ter uma melhor compreensão do problema, a fim de realizar o projeto pretendido.

Nos capítulos seguintes, procede-se à apresentação dos modelos conceptual, lógico e físico do sistema de gestão de bases de dados relacional, bem como as suas respetivas validações. Por último, este apresentará ainda uma secção dedicada à análise crítica do projeto sumarizando desta forma todos os esforços efetuados para alcançar o objetivo proposto.

2. Análise de Requisitos

A análise de requisitos, embora difícil de concretizar com precisão, é uma das fases mais importantes para a implementação do sistema de gestão de bases de dados relacional.

2.1. Método de levantamento de requisitos

No intuito de identificar os requisitos necessários para a elaboração do projeto, procuramos exemplos de livrarias *online*, como por exemplo a *Amazon*, de forma a ter melhor noção daquilo que será necessário criar. Ao analisarmos a informação recolhida, consideramos como fundamentais os seguintes requisitos:

2.2. Requisitos

2.2.1. Requisitos de descrição

1. Todos os livros devem estar registados no sistema;
2. Os livros devem ser caracterizados por: código do livro, título, editora, ISBN, edição, ano, género, língua, quantidade em *stock*, preço;
3. Os livros são escritos por autores e pertencem a uma editora;
4. Os clientes devem ser caracterizados por: número de cliente, nome, NIF, morada, correio eletrónico;
5. Um cliente pode ser Premium e ter descontos associados à sua conta;
6. As encomendas devem ser caracterizadas por: número da encomenda, data, valor da encomenda e o seu tamanho;
7. Um cliente pode encomendar um ou vários livros;
8. O sistema devera manter o registo das vendas efetuadas.

2.2.2. Requisitos de exploração

1. Agrupar livros por autor;
2. Agrupar livros por género;
3. Agrupar livros conforme o idioma em que são escritos;
4. Agrupar livros do mesmo preço, ou gamas de preço;
5. Organizar mais informação, como por exemplo, livros de um dado autor ordenados por ano ou preço;
6. Identificar quais os livros mais vendidos;
7. Identificar quais os autores *bestsellers* num dado mês;
8. Saber qual/quais os clientes que mais compram;
9. Agrupar os clientes por áreas (morada);
10. Identificar os meses em que vendem mais;
11. Identificar qual o total faturado por mês/ano.

2.2.3. Requisitos de controlo

1. O dono da loja devera ter controlo total da base de dados;
2. Os clientes não têm acesso à base de dados.

3. Modelo Conceptual

Uma vez definido o sistema e o levantamento de requisitos, o grupo avançou para a modelação conceptual da base de dados, onde se pretende definir as várias entidades que possam existir, os seus atributos e as relações entre entidades.

3.1. Abordagem de modelação realizada

A modelação conceptual foi feita através de alguns passos:

- Identificação das entidades do sistema, dos seus atributos e definição do tipo/domínio destes;
- Identificação dos relacionamentos entre entidades;
- Determinação das chaves das entidades;
- Construção e validação do modelo conceptual.

3.2. Entidades

De forma a conseguirmos identificar as diferentes entidades que o sistema engloba, foi necessário determinar quais os objetos que se enquadram nesta definição. Para isso, após a leitura dos requisitos podemos chegar à conclusão de que as entidades fundamentais são: o Livro, o Autor, a Editora, o Cliente, o Premium e a Encomenda.

De seguida é apresentada uma tabela com o nome de cada entidade, bem como os seus sinónimos e as suas ocorrências.

Nome da Entidade	Descrição	Sinonimo	Ação
Livro	O objeto de venda da livraria.	Exemplar, Obra	Um livro pode ser vendido para um cliente.
Editora	A editora de um livro.	Editorial	Publica livros.
Autor	Pessoa que escreveu o livro.	Escritor	Escreve um ou vários livros.
Cliente	Leitor que pretende comprar um livro.	Leitor	Pode comprar vários livros.
Premium	Cliente com bónus.	Dourado	Descontos de compra para clientes específicos.
Encomendas	Comprovativo da compra.	Compra	Listar a compra do cliente.

Tabela 1 – Entidades do sistema

3.3. Relacionamentos

Após identificar as entidades é necessário detetar todos os relacionamentos existentes entre as mesmas. A leitura da análise de requisitos permite identificar estes relacionamentos, bem como a sua respetiva cardinalidade. Desta forma, apresentamos na seguinte tabela todos os relacionamentos entre entidades que conseguimos identificar.

Nome da Entidade	Multiplicidade	Relação	Multiplicidade	Nome da Entidade
Encomenda	1..*	possui	1..*	Livro
Autor	1	tem	1..*	Livro
Livro	1..*	pertence	1	Editora
Cliente	1	faz	1..*	Encomenda
Cliente	0..1	é	1	Premium
Cliente	0..1	Recomendado por	0..1	Cliente

Tabela 2 – Relacionamentos entre as entidades

3.4. Associação entre atributos com as entidades

Entidade	Atributos	Tipo e Tamanho	Nulo	Tipo
Livro	codLivro	Valor inteiro positivo	Não	Simples
	ISBN	45 Caracteres variáveis	Não	Simples
	Título	45 Caracteres variáveis	Não	Simples
	Preço	Valor decimal positivo	Não	Simples
	Ano	Valor inteiro positivo	Sim	Simples
	Língua	45 Caracteres variáveis	Não	Simples
	Género	45 Caracteres variáveis	Sim	Simples
	Quantidade	Valor inteiro positivo	Não	Simples
	Edição	45 Caracteres variáveis	Sim	Simples

Tabela 3 - Atributos da entidade Livro

Entidade	Atributos	Tipo e Tamanho	Nulo	Tipo
Cliente	NIF	45 Caracteres variáveis	Não	Simples
	Nome	45 Caracteres variáveis	Sim	Simples
	"e-mail"	45 Caracteres variáveis	Sim	Simples
	nrCliente	Valor inteiro positivo	Não	Simples
	Morada			
	Rua	45 Caracteres variáveis	Não	Simples
	Concelho	45 Caracteres variáveis	Não	Simples
	CodigoPostal	45 Caracteres variáveis	Não	Simples
	QuantidadeComprada	Valor inteiro positivo	Não	Simples
	TotalGasto	Valor decimal(5,2)	Não	Simples

Tabela 4 - Atributos da entidade Cliente

Entidade	Atributos	Tipo e Tamanho	Nulo	Tipo
Autor	idAutor	45 Caracteres variáveis	Não	Simples
	Nome	45 Caracteres variáveis	Sim	Simples
	Idade	Valor inteiro positivo	Sim	Derivado
	DataNascimento	Data	Sim	Simples

Tabela 5 - Atributos da entidade Autor

Entidade	Atributos	Tipo e Tamanho	Nulo	Tipo
Editora	Nome	45 Caracteres variáveis	Não	Simples
	Contactos			
	Email	45 Caracteres variáveis	Não	Simples
	Telefone	Valor inteiro positivo	Não	Multivalorado
	Site	45 Caracteres variáveis	Sim	Simples

Tabela 6 - Atributos da entidade Editora

Entidade	Atributos	Tipo e Tamanho	Nulo	Tipo
Premium	Desconto	Valor Decimal Positivo	Não	Simples

Tabela 7 - Atributos da entidade Premium

Entidade	Atributos	Tipo e Tamanho	Nulo	Tipo
Encomenda	Data	Data	Não	Simples
	ValorTotal	Valor Decimal Positivo	Não	Derivado
	NrEncomenda	Valor Positivo inteiro	Não	Simples
	TamanhoEncomenda	Valor Positivo Inteiro	Não	Simples

Tabela 8 - Atributos da entidade Encomenda

3.5. Detalhe ou generalização de entidades

Seguidamente, são descritos os atributos referentes às diferentes entidades.

Livro:

- codLivro: Número identificador de cada livro. É um número inteiro positivo;
- ISBN: *International Standard Book Number*. É um número inteiro positivo;
- TÍtulo: TÍtulo do Livro. É uma *string* com 45 caracteres variáveis;
- Preço: Preço do livro, representado por um valor decimal positivo;
- Ano: Ano em que o livro foi escrito. É um número inteiro positivo;
- Língua: Idioma em que o livro está escrito. É uma *string* com 45 caracteres variáveis;
- Género: Categoria de literatura em que o livro se insere. É uma *string* com 45 caracteres variáveis;
- Quantidade: Quantidade em *stock* do livro. É um número inteiro positivo;
- Edição: Número de edição do livro. É uma *string* com 45 caracteres variáveis.

Cliente:

- NIF: Número de identificação fiscal. É uma *string* com 45 caracteres variáveis;
- Nome: Primeiro e último nome do cliente. É uma *string* com 45 caracteres variáveis;
- "E-mail": Endereço eletrónico associado à conta do utilizador. É uma *string* com 45 caracteres variáveis;
- NrCliente: Número identificador de cada cliente. É um número inteiro positivo;
- Morada: Morada do cliente, caracterizada por:
  - Rua: Rua onde mora o cliente. É uma *string* com 45 caracteres variáveis;
  - Concelho: Região onde mora o cliente. É uma *string* com 45 caracteres variáveis;
  - CodigoPostal: Código Postal ao qual será entregue a encomenda. É uma *string* com 45 caracteres variáveis.
- QuantidadeComprada: Quantidade de compras efetuadas pelo cliente. É um valor inteiro positivo;
- ValorGasto: Valor gasto pelo cliente. É um valor decimal positivo.

Premium

- Desconto: Taxa de desconto associada ao cliente, todos os clientes premium vão possuir do mesmo desconto, podendo ser alteradas em épocas especiais, sendo o dono da livraria o responsável por isso. Representado por um valor decimal positivo;

Autor

- idAutor: Número identificador de cada cliente Premium. É um número inteiro positivo;
- Nome: Primeiro e último nome do autor do livro. É uma *string* com 45 caracteres variáveis;
- Idade: Idade do autor. É um número inteiro positivo;
- DataNascimento: Data do nascimento do autor. É do tipo *date*;

Editora

- Nome: Nome da editora. É uma *string* com 45 caracteres variáveis;
- Contacto: Morada do cliente, caracterizada por:
    - "E-mail": Endereço eletrónico associado à editora. É uma *string* com 45 caracteres variáveis
    - Telefone: Número telefónico da editora. É um número inteiro positivo;
- "Site": *Website* da editora. É uma *string* com 45 caracteres variáveis.

Encomenda

- Data: Data em que a encomenda foi feita. É do tipo *date*;
- ValorTotal: Preço total a pagar pela encomenda. É um valor decimal positivo;
- NrEncomenda: Número identificador de cada encomenda. É um número inteiro positivo.
- TamanhoEncomenda: Quantidade de livros comprados pelo cliente numa certa encomenda. É um número inteiro positivo.

3.6. Diagrama ER

O modelo conceptual da nossa base de dados baseia-se na gestão dos recursos associados a uma livraria *online*. Depois de devidamente analisados os requisitos que nos foram apresentados conseguimos inferir seis entidades fundamentais nas quais se sustenta o nosso projeto. Tendo isto em conta, começamos por definir a entidade Livro. Esta representa um dos recursos fulcrais que o dono da livraria gostaria de guardar. Nesse sentido, seria importante associar a cada livro uma certa quantidade de características. Entre elas, temos o título e o código do livro que permite singular cada um destes, segue-se o género que serve para identificar a categoria em que o livro se integra, tal como o género, temos a língua que se refere ao idioma em que o livro se encontra escrito. O ISBN que possibilita ao proprietário identificar o livro a nível internacional, temos também preço associado ao preço de compra do mesmo. Caracterizamos ainda o ano e a edição que tanto uma como outra podem ser apresentadas na capa de um livro. Por último, a quantidade em *stock* de cada livro na livraria que tem como propósito permitir ao dono da livraria fazer a gestão dos *stocks* dos seus livros.

Um livro é escrito por um autor o qual recorreu a uma editora para o publicar, dessa forma inserimos essas duas entidades na nossa base de dados. Autor reconhecido pelo seu nome é também caracterizado pela sua data de nascimento e idade.

No entanto, sendo o nome que o identifica, não é condição suficiente para o individualizar, pois vários autores podem partilhar o mesmo nome. Assim, foi criado um id que nos permite identificar qual o autor que se trata.

Quanto à editora, seguindo a mesma lógica, tem um nome que a identifica, no entanto foi necessário criar uma chave id que a distinguisse das restantes editoras que possam ter nomes

iguais. Mesmo que pouco provável, achamos esta medida mais segura. Para além disso associamos à editora uma lista de contactos, um *email* e um *website*.

Cada livro pode ser comprado por múltiplos clientes e cada cliente pode comprar vários livros. Deste modo, introduzimos a entidade Cliente, uma vez que este representa outro recurso que o proprietário pretende gerir. Para o efeito, é relevante guardar o nome destes assim como um número que os identifica. Para além disto, é também relevante guardar o *Número de Identificação Fiscal (NIF)* do mesmo. Consideramos também importante representar o *email* do cliente e a morada para qual as encomendas posteriormente serão enviadas. Outros aspetos importantes de se guardar, segundo o proprietário, são os vários contactos que um cliente possui bem como o valor gasto do mesmo ao longo de várias compras tal como as quantidades compradas. No entanto, de forma de privilegiar bons clientes, o proprietário achou por bem arranjar uma forma de os distinguir dos restantes clientes e atribuir-lhes talões de desconto. Dessa forma, vimos a necessidade de criar uma outra entidade intimamente relacionada com cliente, a entidade Premium. Sendo então caracterizada por desconto que lhe e atribuído e o número do Cliente que lhe compete.

Ficou então por falar na última entidade forte, que é a entidade que relaciona os clientes com as compras dos livros: a Encomenda. A Encomenda é identificada por um número que permite ao cliente segui-la após ser enviada por correio. Possui um valor total, sendo este dado pela soma dos preços dos livros comprados com ou sem descontos do cliente. Para além do preço, tem também o tamanho da encomenda que reflete o número de livros comprados e por último a data em que esta foi feita.

Por motivos administrativos, vimos a necessidade de acrescentar/usar o id nas diferentes entidades para desta forma ser possível a sua individualização. Nesse sentido, este atributo funcionará como uma chave primária. Como visto no caso da entidade Livro, Cliente, Autor, Editora e Encomenda. É importante referir que a entidade Premium não possui uma chave primária uma vez que está associada à entidade Cliente. No entanto, no decorrer de todo o processo, identificamos algumas chaves candidatas. No caso da entidade Cliente poderíamos ter o nome como uma chave candidata, mas, poderíamos ter casos em que dois clientes possuam o mesmo nome pelo que este atributo não funcionaria como chave primária. Ou até no caso do NIF que pode ser partilhado por membros da mesma família. No caso da entidade Cliente o único atributo que poderia ser candidato a chave primária seria o seu número.

Apresentamos, de seguida, o desenho do diagrama E-R (Entidade-Relacionamento) de forma a representar conceptualmente as relações entre as entidades da base de dados.

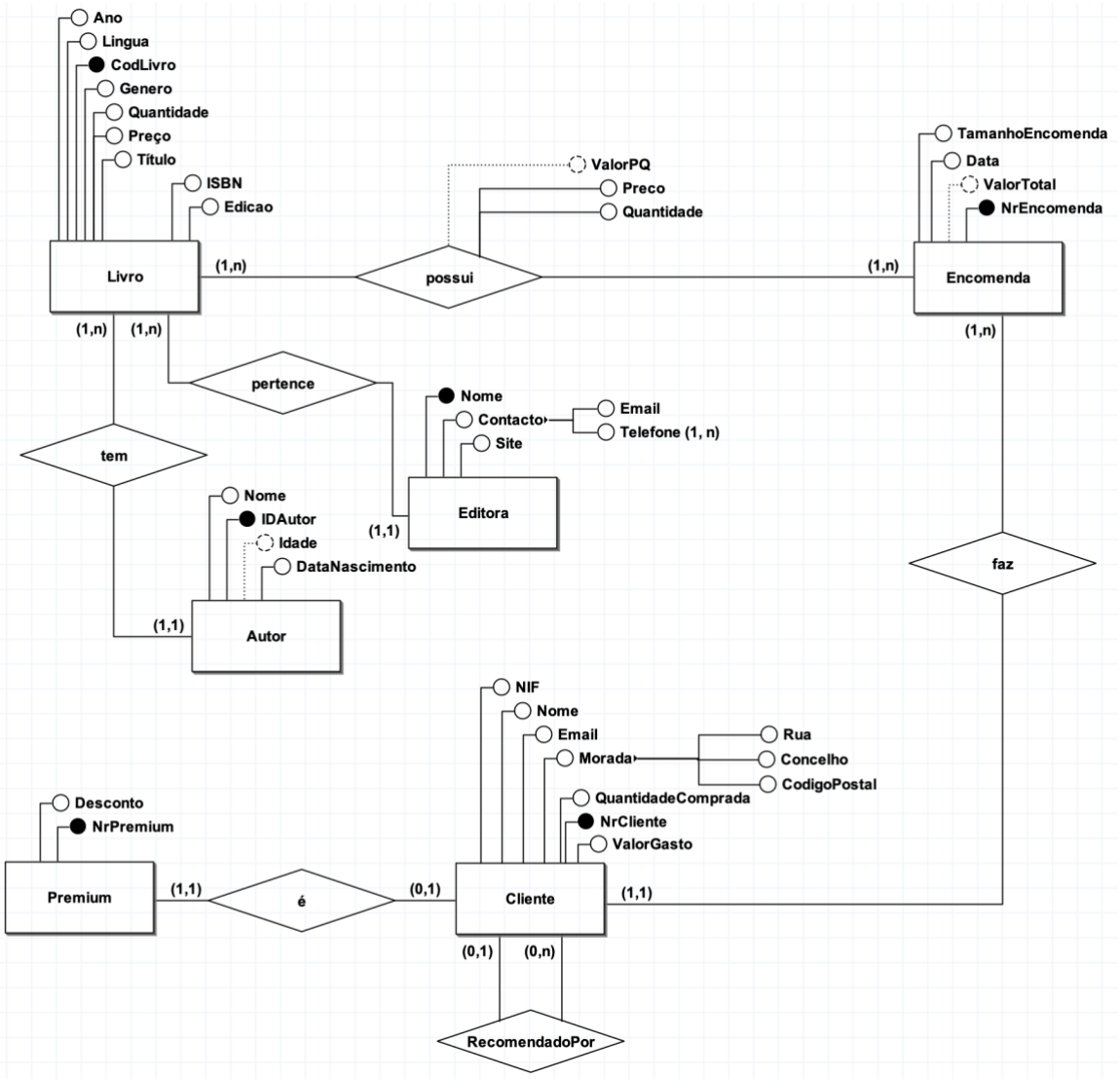


Figura 1 - Modelo conceptual do sistema de base de dados

3.7. Validação do modelo de dados produzido

Uma vez finalizado o modelo da figura acima, foi necessária a aprovação da Sra. Alice para avançar com o projeto. Para isso, foi realizada uma reunião presencial com a mesma onde foram explicados os significados de cada parte do diagrama, nomeadamente as entidades, atributos e a maneira como se relacionavam entre si. Uma vez abordada esta estrutura, foi explicado como funcionaria internamente cada funcionalidade do programa, assim como todas as potencialidades do nosso projeto. Desta forma, acreditamos que o trabalho ia de encontro ao que inicialmente foi pedido e previsto pelo grupo.

Uma vez explicado todo o sistema, a Sra. Alice mostrou-se satisfeita com o modelo e deu luz verde à continuação do seu desenvolvimento.

4. Modelo Lógico

Uma vez construído e validado o modelo conceptual, chegamos então a base sólida para continuar com o projeto. De seguida vem a modelação lógica da Base de Dados, que consiste na representação da estrutura de dados a um nível de abstração superior ao plano físico.

4.1. Construção e validação do modelo de dados lógico

De forma a construir o modelo lógico, utilizamos o conceptual para nos orientarmos, no entanto, apesar de o modelo lógico ser equivalente a este, algumas opções têm de ser tomadas de modo independente por forma a garantir consistência e coerência nos dados. Neste sentido, especificamos de seguida todo o processo.

1. Cada entidade deu origem a uma relação (tabela), onde cada atributo foi convertido numa coluna da mesma. O atributo composto *Morada* da entidade *Cliente* foi decomposto nos seus diferentes valores. Na entidade *Autor* o atributo derivado *Idade* não foi introduzido uma vez que é calculado a partir do atributo *DataNascimento;*
2. Em cada relação foram identificadas as chaves primárias assim como as variáveis que não seriam nulas (NN)
3. Um atributo multivalorado seria também convertido numa relação com o identificador da outra relação a que estava associado como chave estrangeira da mesma. Neste caso, a Livraria Online não possui atributos multivalorados;
4. Foram estabelecidos relacionamentos de 1 para N, onde a entidade com multiplicidade N fica com o identificador da outra entidade como chave estrangeira. Estas chaves estrangeiras são possíveis de observar nas relações: *Cliente* para *Encomenda*, *Autor* para *Livro* e *Editora* para *Livro*;
5. De seguida foi o relacionamento 1 para 1 entre *Premium* e *Cliente*. Neste caso, qualquer relação poderia ficar com a chave estrangeira da outra. A chave estrangeira foi atribuída à entidade *Premium*.
6. Em último foi feito o relacionamento de N para N entre *Livro* e *Encomenda*, que deu origem a um nova relação *Compra* que possui como colunas, para além das chaves primárias das relações a que está ligada, o *Preco*, que corresponde ao preço individual de um *Livro*, a *Quantidade* comprada de cada *Livro* e o *PrecoPQ*, que corresponde ao produto do preço com a quantidade.

Figura 2 - Passagem da entidade Livro para modelo lógico

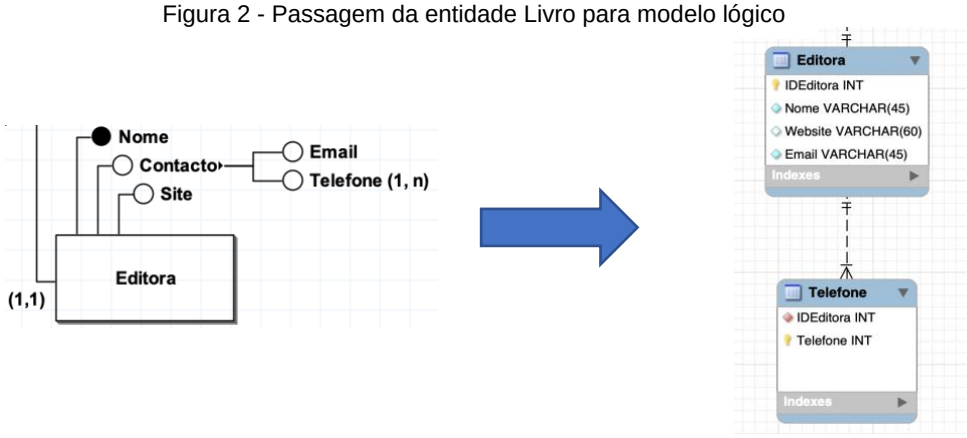


Figura 3 - Passagem da entidade Editora para modelo lógico

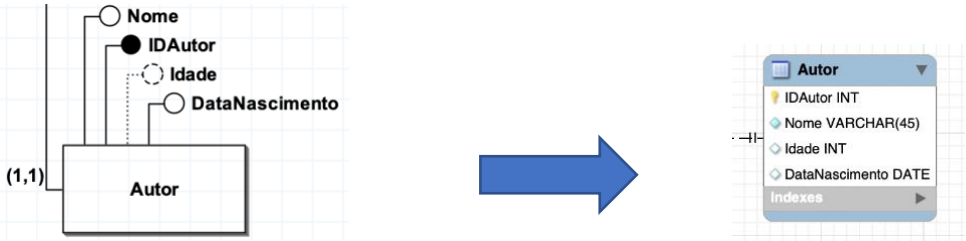


Figura 4 - Passagem da entidade Autor para o modelo lógico

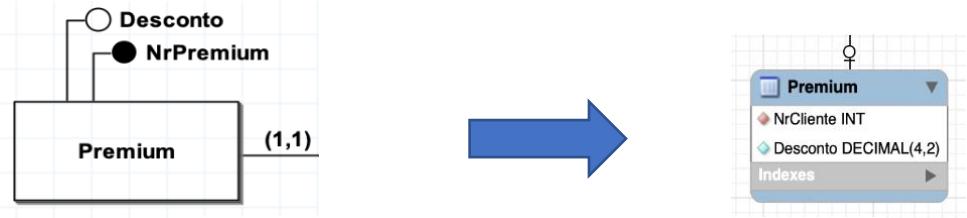


Figura 5 - Passagem da entidade Premium para o modelo lógico

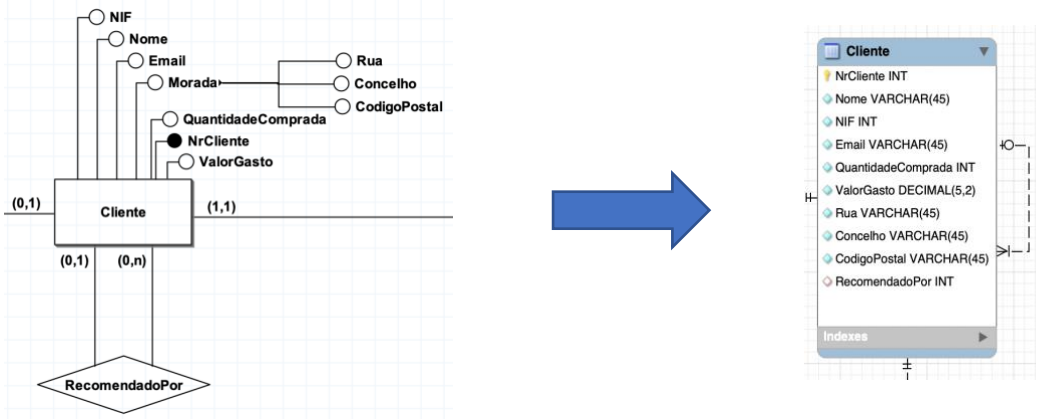


Figura 6 - Passagem da entidade Cliente para o modelo lógico

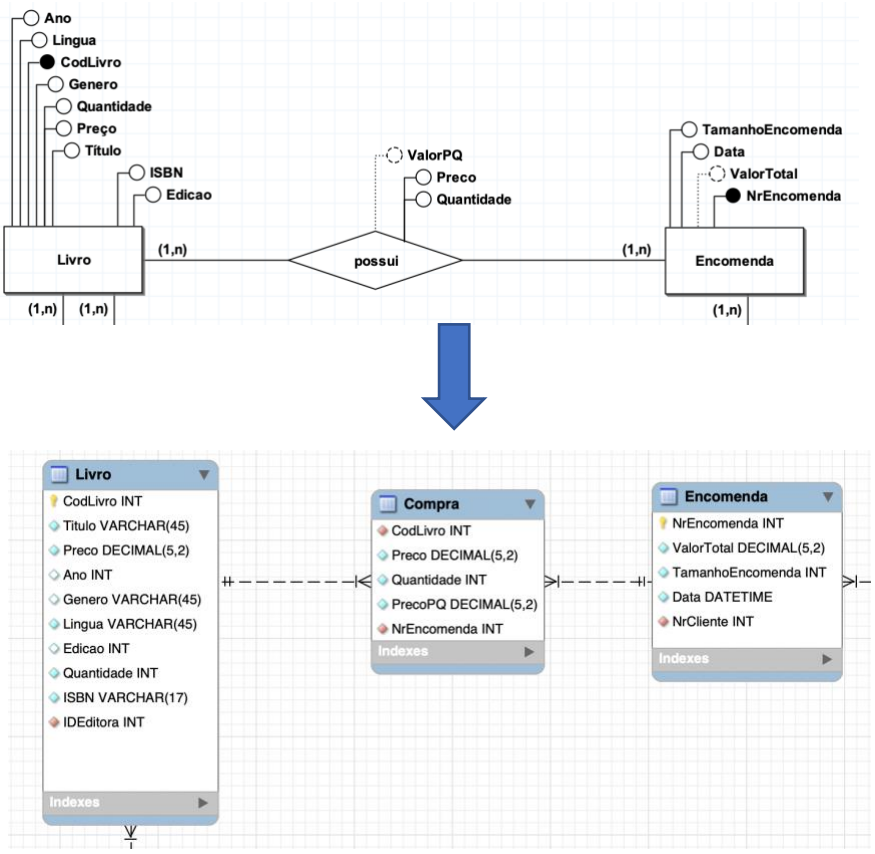


Figura 7 - Relacionamento Livro/Encomenda para o modelo lógico

4.2. Desenho do Modelo Lógico

De seguida, apresentamos uma imagem do esquema lógico elaborado.

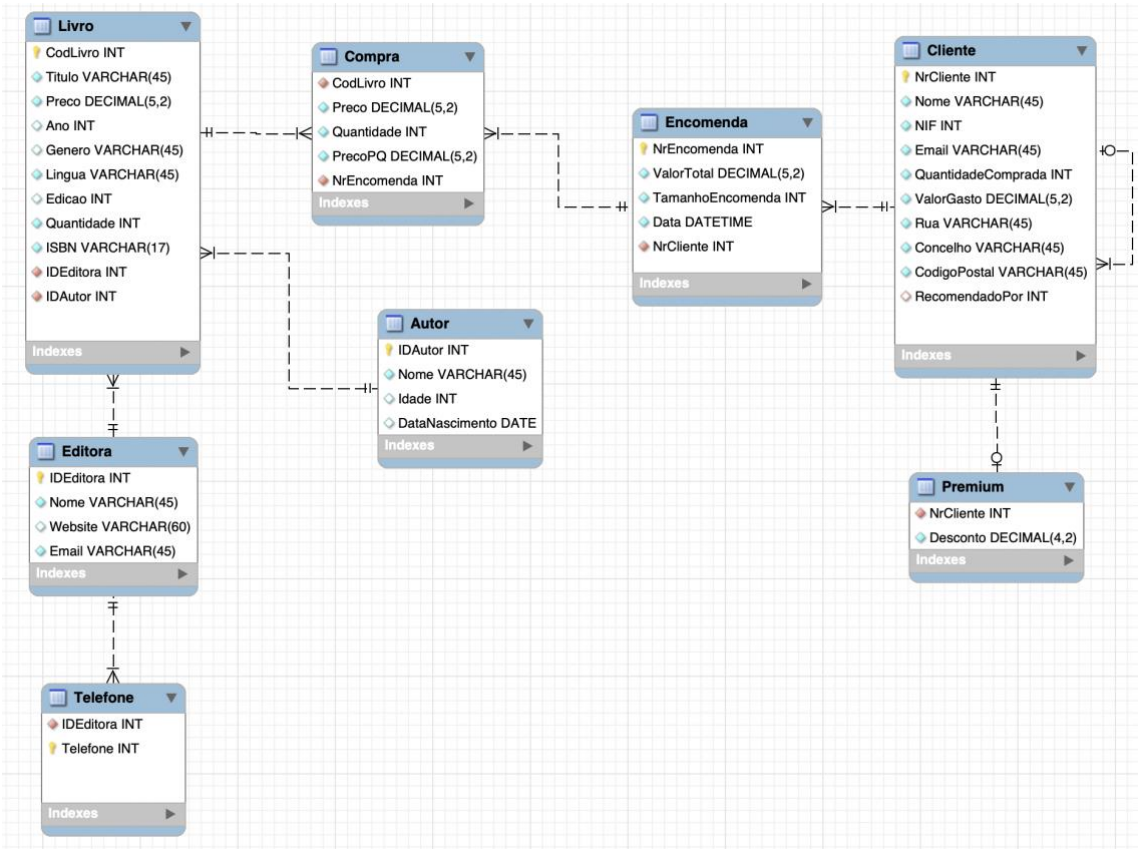


Figura 8 - Modelo lógico do sistema de base de dados

4.3. Validação do modelo através da normalização

A normalização visa organizar o modelo de maneira a reduzir a redundância de dados, incrementar a integridade e o desempenho da BD. Consiste em três regras:

4.3.1. Primeira Forma Normal

Esta primeira regra verifica-se uma vez que nenhuma tabela do nosso modelo possui colunas repetidas nem atributos multivalorados.

4.3.2. Segunda Forma Normal

Como se verificou anteriormente, o modelo obedece à 1FN. Não existe nenhuma chave primária composta, pelo que não há dependências parciais. Logo, todas as relações do modelo também obedecem a 2FN.

4.3.3. Terceira Forma Normal

Como nenhum atributo nem chave primária possui uma dependência transitiva em relação às chaves candidatas da mesma tabela, isto é, todas as colunas de uma tabela dependem única e exclusivamente da respetiva chave primária, as relações obedecem também à 3FN. Caso o atributo *Idade* da entidade *Autor* tivesse sido convertido numa coluna da respetiva relação, esta obediência não se verificaria uma vez que o atributo *Idade* depende do atributo *DataNascimento*.

4.4. Validação do modelo com interrogações do utilizador

É necessário verificar que o modelo lógico assegura o funcionamento dos requisitos por exploração, isto é, interrogações que o utilizador pode fazer ao sistema.

- **Requisito 1 – Agrupar livros por autor:**

Para agrupar os livros pelo autor basta aceder ao *Autor* de cada *Livro* e agrupar conforme o autor do livro seja o mesmo;

- **Requisito 2 – Agrupar livros por género:**

Para agrupar os livros por género basta aceder ao atributo *Genero* de cada *Livro* e agrupar os livros que possuem o mesmo género.

- **Requisito 3 – Agrupar os livros conforme o idioma em que são escritos:**

Para agrupar os livros por linguagem basta aceder ao atributo *Lingua* de cada *Livro* e agrupar os livros que possuem a mesma linguagem.

- **Requisito 4 – Agrupar livros do mesmo preço ou gamas de preço:**

Para agrupar os livros por gama de preço basta aceder ao atributo *Preco* de cada *Livro* e agrupá-los consoante o seu preço.

- **Requisito 5 – Livros de um dado autor ordenados por ano ou preço:**

Para ordenar os livros de um autor por ano/preço precisamos de identificar quais os livros escritos pelo autor. Para isto, basta aceder à entidade *Autor* de cada entidade *Livro* e identificar o escrito pelo autor dado. Após isto ordenamos esses livros pelo seu atributo *Ano* ou *Preco*.

- **Requisito 6 – Identificar os livros mais vendidos:**

Para identificar os livros mais vendidos basta percorrer todas as entidades *Compra* existentes, contando e acumulando o número total de livros comprados de cada *Livro*.

- **Requisito 7 – Identificar quais os autores *bestsellers* num dados mês:**

Para identificar os autores *bestsellers* de num dados mês basta percorrer todas as entidades *Encomenda* nesse mês, acumulando o número de livros que cada autor vendeu. Temos acesso a esse número percorrendo as entidades *Compra* associada a cada *Encomenda* e identificando o Autor que escrever o *Livro* dessa compra.

- **Requisito 8 – Identificar qual/quais os clientes que mais compraram:**

Para identificar qual/quais os clientes que mais compraram, basta aceder ao atributo *QuantidadeComprada* de cada cliente e identificar aquele que mais comprou.

- **Requisito 9 – Agrupar clientes por morada:**

Para agrupar os clientes por morada basta aceder ao atributo *Morada* de cada *Cliente*.

- **Requisito 10 – Identificar os meses em que vendem mais:**

Para identificar os meses que vendem mais basta percorrer todas as entidades *Encomenda* e acumulando o atributo *ValorTotal* consoante os meses, que obtemos através do atributo *Data*.

- **Requisito 11 – Identificar qual o total faturado por mês/ano:**

Para identificar o total faturado por mês/ano basta percorrer todas as entidades *Encomenda*, acumulando o atributo *ValorTotal* e identificando o valor em cada mês/ano através do atributo *Data*.

4.5. Reavaliação do modelo lógico

Depois de concluído todo o processo que deu origem ao modelo lógico, verificamos se era necessário haver uma reavaliação deste. Para o efeito, uma nova reunião foi marcada com a Sra. Alice com o intuito de a manter a par de todo o processo e das novas atualizações da sua base de dados. O modelo lógico foi-lhe mostrado e explicado aquilo que foi concluído das validações do mesmo.

A Sra. Alice mostrou-se satisfeita com o resultado final e sugeriu que se passasse então à implementação física da base de dados para a sua Livraria *Online*.

5. Implementação Física

5.1. Seleção do sistema de gestão de bases de dados

Para o sistema de gestão de bases de dados, optamos pelo *MySQL*. Esta decisão deveu-se ao facto de este se tratar de um software *open-source* (apesar de ter sido adquirido pela Oracle) pelo que seria uma ferramenta bastante acessível. Outra das vantagens que o *MySQL* traria era o facto de existir o *MySQL Workbench*, o que nos permitia escrever as nossas instruções *SQL* tendo ao lado uma visualização do nosso modelo lógico, tornando mais fácil raciocinar sobre as interrogações que pretendíamos escrever. Por fim, sendo este apenas um trabalho académico e pelo que o *MySQL* está disponível para diferentes plataformas, facilitou na elaboração do nosso projeto, visto que diferentes membros do grupo usavam diferentes sistemas operativos (nomeadamente *Linux* e *Windows*).

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

De modo a construir o modelo físico, usamos a ferramenta de *Forward Engineering* disponibilizada pelo *MySQL*. Assim, o código necessário para esta transição foi gerado automaticamente. Explicitamos de seguida algum do código criado, o qual poderá ser consultado na íntegra em anexo.

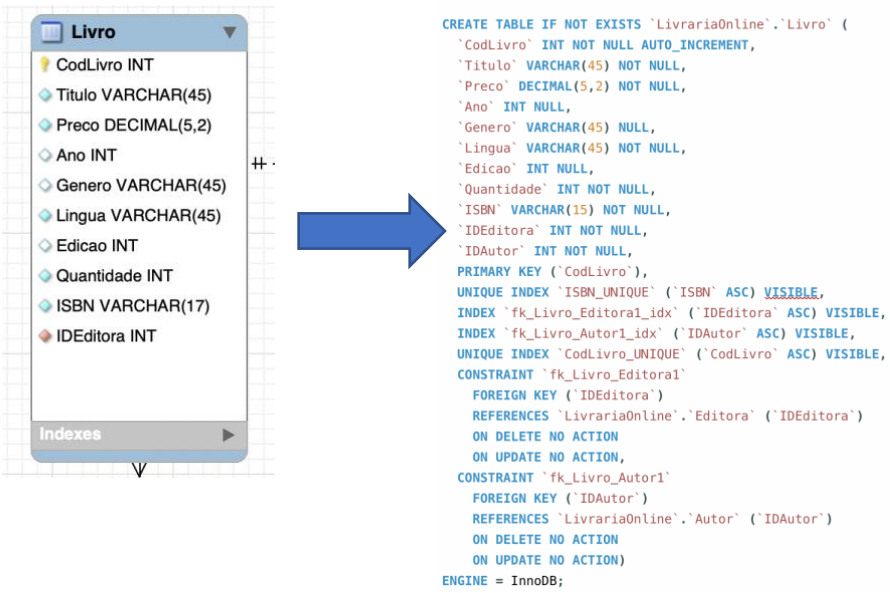
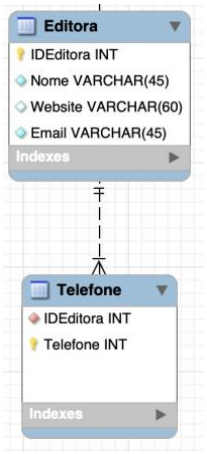


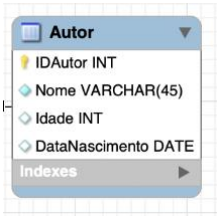
Figura 9 - Tabela Livro no modelo físico




```
-- -----------------------------------------------------
-- Table `LivrariaOnline`.`Editora`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `LivrariaOnline`.`Editora` ;

CREATE TABLE IF NOT EXISTS `LivrariaOnline`.`Editora` (
  `IDEditora` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(45) NOT NULL,
  `Website` VARCHAR(60) NULL,
  `Email` VARCHAR(45) NOT NULL,
  UNIQUE INDEX `Nome_UNIQUE` (`Nome` ASC) VISIBLE,
  PRIMARY KEY (`IDEditora`),
  UNIQUE INDEX `IDEditora_UNIQUE` (`IDEditora` ASC) VISIBLE,
  UNIQUE INDEX `Email_UNIQUE` (`Email` ASC) VISIBLE)
ENGINE = InnoDB;
```

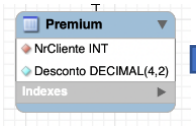
Figura 10 - Tabela Editora no modelo físico




```
-- -----------------------------------------------------
-- Table `LivrariaOnline`.`Autor`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `LivrariaOnline`.`Autor` ;

CREATE TABLE IF NOT EXISTS `LivrariaOnline`.`Autor` (
  `IDAutor` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(45) NOT NULL,
  `Idade` INT NULL,
  `DataNascimento` DATE NULL,
  PRIMARY KEY (`IDAutor`),
  UNIQUE INDEX `IDAutor_UNIQUE` (`IDAutor` ASC) VISIBLE)
ENGINE = InnoDB;
```

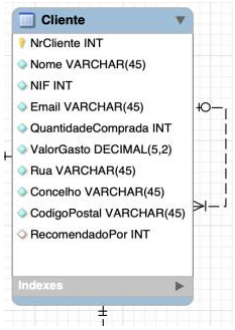
Figura 11 - Tabela Autor no modelo físico




```
-- -----------------------------------------------------
-- Table `LivrariaOnline`.`Premium`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `LivrariaOnline`.`Premium` ;

CREATE TABLE IF NOT EXISTS `LivrariaOnline`.`Premium` (
  `NrCliente` INT NOT NULL,
  `Desconto` DECIMAL(4,2) NOT NULL,
  INDEX `fk_Premium_Cliente1_idx` (`NrCliente` ASC) VISIBLE,
  CONSTRAINT `fk_Premium_Cliente1`
    FOREIGN KEY (`NrCliente`)
    REFERENCES `LivrariaOnline`.`Cliente` (`NrCliente`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 12 - Tabela Premium no modelo físico




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

CREATE TABLE IF NOT EXISTS `LivrariaOnline`.`Cliente` (
  `NrCliente` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(45) NOT NULL,
  `NIF` INT NOT NULL,
  `Email` VARCHAR(45) NOT NULL,
  `QuantidadeComprada` INT NOT NULL,
  `ValorGasto` DECIMAL(5,2) NOT NULL,
  `RecomendadePor` INT NULL,
  `Rua` VARCHAR(45) NOT NULL,
  `Concelho` VARCHAR(45) NOT NULL,
  `CodigoPostal` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`NrCliente`),
  UNIQUE INDEX `NrCliente_UNIQUE` (`NrCliente` ASC) VISIBLE,
  UNIQUE INDEX `NIF_UNIQUE` (`NIF` ASC) VISIBLE,
  INDEX `fk_Cliente_Cliente1_idx` (`RecomendadePor` ASC) VISIBLE,
  CONSTRAINT `fk_Cliente_Cliente1`
    FOREIGN KEY (`RecomendadePor`)
    REFERENCES `LivrariaOnline`.`Cliente` (`NrCliente`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 13 - Tabela Cliente no modelo físico

5.3. Tradução dos requisitos do utilizador para SQL

Depois de povoada a base de dados (ver anexo), implementamos as interrogações necessárias em *MySQL*.

- **Requisito 1 – Agrupar livros por autor:**

```sql
-- 1 Seleciona os livros organizados por autor
SELECT Livro.titulo, Autor.nome AS Autor
FROM Livro JOIN Autor ON Livro.IDAutor = autor.IDAutor;
```

Figura 14 - Código SQL do requisito de exploração 1

- **Requisito 2 – Agrupar livros por género:**

```sql
-- 2 Seleciona os livros organizados por genero
SELECT titulo, genero FROM Livro
ORDER BY genero ASC;
```

Figura 15 - Código SQL do requisito de exploração 2

- **Requisito 3 – Agrupar os livros conforme o idioma em que são escritos:**

```sql
-- 3 Seleciona os livros organizados por idioma
SELECT titulo, lingua AS idioma FROM Livro
ORDER BY idioma ASC;
```

Figura 16 - Código SQL do requisito de exploração 3

- **Requisito 4 – Agrupar livros do mesmo preço ou gamas de preço:**

```sql
-- 4 Seleciona os livros organizados por preco
SELECT titulo, preco FROM Livro
ORDER BY preco ASC;
```

Figura 17 - Código SQL do requisito de exploração 4

- **Requisito 5 – Livros de um dado autor ordenados por ano ou preço:**

```sql
-- 5 Seleciona os livros de cada autor organizados pelo nome do autor e posteriormente pelo preço do livro
SELECT Livro.titulo, Autor.nome AS Autor, Livro.preco
FROM Livro JOIN Autor ON Livro.IDAutor = autor.IDAutor
ORDER BY autor.nome ASC, preco ASC;
```

Figura 18 - Código SQL do requisito de exploração 5

- **Requisito 6 – Identificar os livros mais vendidos:**

```sql
-- 6 Lista de livros ordenados por mais vendidos e em caso de empate pelo preço total
SELECT Titulo, sum(compra.quantidade) AS QtdVendida, compra.Preco*sum(compra.quantidade) AS PrecoTotal
    FROM livro
    JOIN Compra ON livro.CodLivro = compra.CodLivro
    GROUP BY livro.titulo
    ORDER BY QtdVendida DESC, PrecoTotal DESC;
```

Figura 19 - Código SQL do requisito de exploração 6

- **Requisito 7 – Identificar quais os autores *bestsellers* num dados mês:**

```sql
-- 7 Autor mais vendido num mes dado
SELECT Autor.nome , Encomenda.data, sum(compra.quantidade) AS QtdVendida FROM livro
    JOIN Compra ON livro.CodLivro = compra.CodLivro
    JOIN Autor ON livro.idAutor = Autor.idAutor
    JOIN Encomenda ON compra.nrEncomenda = Encomenda.nrEncomenda
    WHERE MONTH(Encomenda.data) = 12
    GROUP BY Autor.nome
    ORDER BY QtdVendida DESC
```

Figura 20 - Código SQL do requisito de exploração 7

- **Requisito 8 – Identificar qual/quais os clientes que mais compraram:**

```sql
-- 8 Clientes que mais compraram por quantidade de livros
SELECT nome, quantidadeComprada FROM Cliente
Group by nome
Order BY quantidadeComprada DESC;

-- 8 Clientes que mais compraram por valor gasto
SELECT nome, valorGasto FROM Cliente
 Group by nome
 Order BY valorGasto DESC;
```

Figura 21- Código SQL do requisito de exploração 8

- **Requisito 9 – Agrupar clientes por morada:**

```sql
-- 9 - Ordenar Clientes por Concelho
SELECT nome, concelho, rua, codigoPostal FROM Cliente
ORDER BY concelho ASC, nome ASC;
```

Figura 22 - Código SQL do requisito de exploração 9

- **Requisito 10 – Identificar os meses em que vendem mais:**

```sql
-- 10.1 Apresentar os meses ordenados por quantidade de livros vendidos
SELECT month(Encomenda.data) AS Mes, sum(Encomenda.tamanhoEncomenda) AS QuantidadeVendida
FROM Encomenda JOIN Cliente ON encomenda.nrCliente = cliente.nrCliente
Group BY Mes
ORDER BY quantidadeVendida DESC;

-- 10.2 Apresentar os meses ordenados por ValorGasto (Faturacao)
SELECT month(Encomenda.data) AS Mes, sum(Encomenda.valorTotal) AS Faturacao
FROM Encomenda JOIN Cliente ON encomenda.nrCliente = cliente.nrCliente
GROUP BY Mes
ORDER BY Faturacao DESC;
```

Figura 23 - Código SQL do requisito de exploração 10

- **Requisito 11 – Identificar qual o total faturado por mês/ano:**

```sql
-- 11 Apresentar o total faturado por mes/ano
SELECT year(Encomenda.data) AS Ano, month(Encomenda.data) AS Mes, sum(Encomenda.valorTotal) AS Faturacao
FROM Encomenda
GROUP BY ano, mes
ORDER BY ano DESC, mes;
```

Figura 24 - Código SQL do requisito de exploração 11

Para além destes requisitos, desenvolvemos todos os mecanismos necessários para que estes fossem todos cumpridos. De facto, verificámos que algumas destas *queries* poderiam ser aliadas com outras funcionalidades que facilitarão o dia-a-dia do dono da Livraria. Assim, exemplificamos de seguida algumas delas:

```
-- Verificar se é possivel efetuar uma compra (se há stock suficiente do livro)
DELIMITER $$
CREATE TRIGGER tgVerificaStock
    BEFORE INSERT ON Compra
    FOR EACH ROW
BEGIN
    DECLARE quantity INT;
    SELECT Livro.quantidade INTO quantity FROM Livro
    WHERE Livro.codLivro = NEW.codLivro;
    IF (quantity < NEW.quantidade)
    THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Stock insuficiente';
    END IF;
END
$$
```

Figura 25 - *Trigger* que verifica se é possível efetuar uma conta

```
-- Update a quantidadeComprada e valorGasto de um cliente após efetuar uma Encomenda (depois de cada compra)
DELIMITER $$
CREATE TRIGGER tgAtualizaCliente
    AFTER UPDATE ON Encomenda
    FOR EACH ROW
BEGIN
    UPDATE cliente
        SET quantidadeComprada = quantidadeComprada + (NEW.tamanhoEncomenda - OLD.tamanhoEncomenda),
        valorGasto = valorGasto + (NEW.valorTotal - OLD.valorTotal)
        WHERE NrCliente = NEW.NrCliente;
END $$
```

Figura 26 - *Trigger* que atualiza a tabela Cliente após uma compra

```
-- Atualiza a encomenda após a inserção de 1 compra
-- (valorTotal, tamanhoEncomenda e aplica promocao para os preimium)
DELIMITER $$
CREATE TRIGGER tgAtualizaEncomenda
    AFTER INSERT ON Compra
    FOR EACH ROW
BEGIN
    DECLARE promo DECIMAL(5,2);
    DECLARE idCliente INT;
    SELECT nrCliente INTO idCLiente FROM Encomenda
    WHERE Encomenda.nrEncomenda = NEW.nrEncomenda;

    IF (idCliente IN
        (SELECT nrCliente FROM Premium))
    THEN
        SELECT Premium.desconto INTO promo FROM Premium
        WHERE Premium.nrCliente = idCliente;
        UPDATE Encomenda
            SET valorTotal = valorTotal + (NEW.precoPQ * (1 - promo)),
            tamanhoEncomenda = tamanhoEncomenda + NEW.quantidade
            WHERE nrEncomenda = NEW.nrEncomenda;

    ELSE
        UPDATE ENCOMENDA
        SET valorTotal = valorTotal + NEW.precoPQ,
            tamanhoEncomenda = tamanhoEncomenda + NEW.quantidade
            WHERE nrEncomenda = NEW.nrEncomenda;
    END IF;
END $$
```

Figura 27 - *Trigger* que atualiza uma encomenda após uma compra

```sql
-- atualizar o stock do livro após uma Compra
DELIMITER $$
CREATE TRIGGER tgAtualizaStock
    AFTER INSERT ON Compra
    FOR EACH ROW
BEGIN
    UPDATE Livro
        SET quantidade = quantidade - NEW.quantidade
        WHERE CodLivro = NEW.CodLivro;
END $$
```

Figura 28 - *Trigger* que atualiza o stock de um livro após uma compra

5.4. Tradução de outras operações do utilizador para SQL

Para além das consultas de caráter de observação, há ainda outras operações possíveis pelos vários tipos de utilizadores do sistema. Como tal, de seguida apresentam-se o código SQL que possibilita tais operações:

```sql
-- 1 DADO UM AUTOR, APRESENTA OS SEUS LIVROS (usamos o idAutor pois pode haver nomes de autores iguais)
DELIMITER $$
CREATE PROCEDURE livrosPorAutor (IN idAutor INT)
BEGIN
    SELECT Livro.codLivro, Livro.titulo
    FROM Livro JOIN Autor ON Livro.IDAutor = autor.IDAutor
    WHERE Autor.idAutor = idAutor;
END
```

Figura 29 - Código SQL para apresentar os livros de um autor

```sql
-- 2 Seleciona os livros de um dado genero
DELIMITER $$
CREATE PROCEDURE livrosPorGenero (IN genero VARCHAR(25))
BEGIN
    SELECT codLivro, titulo FROM Livro
    WHERE Livro.genero = genero;
END
```

Figura 30 - Código SQL para selecionar livros de um dado género

```sql
-- 3 Seleciona os livros de um ididoma
DELIMITER $$
CREATE PROCEDURE livrosPorIdioma (IN idioma VARCHAR(25))
BEGIN
    SELECT codLivro, titulo FROM Livro
    WHERE Livro.lingua = idioma;
```

Figura 31 - Código SQL para selecionar livros de um idioma

```sql
-- 4 AGRUPAR OS LIVROS DADA UMA GAMA DE PRECO
DELIMITER $$
CREATE PROCEDURE livrosPorGamaPreco (IN precoInicial DECIMAL, IN precoFinal DECIMAL)
BEGIN
    IF (precoInicial < 0 OR precoInicial >= precoFinal OR precoFinal < 0) THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Gama de precos invalida';
    ELSE
        SELECT codLivro, titulo, preco FROM Livro
        WHERE preco >= precoInicial AND preco <= precoFinal
        ORDER BY preco;
```

Figura 32 - Código SQL para agrupar livros por gama de preços

```sql
-- 5 AGRUPAR OS LIVROS POR ANO, DADO UM AUTOR
DELIMITER $$
CREATE PROCEDURE livroAutorAno (IN autor VARCHAR(60))
BEGIN
    SELECT Autor.nome,Livro.CodLivro, Livro.titulo, Livro.ano
    FROM Livro JOIN Autor ON Livro.idAutor = autor.idAutor
    WHERE Autor.nome = autor
    ORDER BY ano ASC;
END
```

Figura 33 - Código SQL para agrupar os livros de um autor por ano

```sql
-- 6 AGRUPAR OS LIVROS POR PRECO, DADO UM AUTOR
DELIMITER $$
CREATE PROCEDURE livroAutorPreco (IN autor VARCHAR(60))
BEGIN
    SELECT Autor.nome, Livro.CodLivro, Livro.titulo, Livro.preco
    FROM Livro JOIN Autor ON Livro.idAutor = autor.idAutor
    WHERE Autor.nome = autor
    ORDER BY preco ASC;
END
```

Figura 34 - Código SQL para agrupar os livros de um autor por preço

```sql
-- 7 (TOP 3) dos Livros mais vendidos
DELIMITER $$
CREATE PROCEDURE top3LivrosMaisVendidos ()
BEGIN
    SELECT Titulo, sum(compra.quantidade) AS QtdVendida, compra.Preco*sum(compra.quantidade) AS PrecoTotal
    FROM livro
    JOIN Compra ON livro.CodLivro = compra.CodLivro
    GROUP BY livro.titulo
    ORDER BY QtdVendida DESC, PrecoTotal DESC
    LIMIT 3;
END
```

Figura 35 - Código SQL para calcular os 3 livros mais vendidos

```sql
-- 8 (TOP 3) de autores bestsellers
DELIMITER $$
CREATE PROCEDURE top3AutoresBestseller (IN mes INT)
BEGIN
    SELECT Autor.nome , Encomenda.data, sum(compra.quantidade) AS QtdVendida FROM livro
    JOIN Compra ON livro.CodLivro = compra.CodLivro
    JOIN Autor ON livro.idAutor = Autor.idAutor
    JOIN Encomenda ON compra.nrEncomenda = Encomenda.nrEncomenda
    WHERE MONTH(Encomenda.data) = mes
    GROUP BY Autor.nome
    ORDER BY QtdVendida DESC
    LIMIT 3;
END
```

Figura 36 - Código SQL para calcular os 3 autores com mais livros vendidos

```sql
-- 9 (TOP 5) Melhores clientes (com maior ValorGasto)
DELIMITER $$
CREATE PROCEDURE top5MelhoresClientes ()
BEGIN
    SELECT nome, valorGasto FROM Cliente
    Group by nome
    Order BY valorGasto DESC
    LIMIT 5;
END
```

Figura 37 - Código SQL para calcular os 5 clientes que mais gastaram

```sql
-- 11 TOP 5 de meses em que se vende mais (Faturacao) num dado ano
DELIMITER $$
CREATE PROCEDURE top5melhoresMesesPorAno (IN nrAno INT)
BEGIN
    SELECT year(Encomenda.data) AS Ano, month(Encomenda.data) AS Mes, sum(Encomenda.valorTotal) AS Faturacao
    FROM Encomenda
    WHERE year(Encomenda.data) = nrAno
    GROUP BY mes
    ORDER BY Faturacao DESC;
END
```

Figura 38 - Código SQL para calcular os 5 meses com mais faturação num ano

```sql
-- Dado um nrCliente mostra os livros comprados por esse cliente
DELIMITER $$
CREATE PROCEDURE clienteLivros (IN idCliente INT)
BEGIN
    SELECT Cliente.nome, Livro.codLivro, Livro.titulo FROM Livro
    JOIN Compra ON Livro.CodLivro = Compra.CodLivro
    JOIN Encomenda ON Compra.NrEncomenda = Encomenda.NrEncomenda
    JOIN Cliente ON Encomenda.NrCliente = Cliente.NrCliente

    WHERE Cliente.nrCliente = idCliente
    ORDER BY codLivro;
```

Figura 39 - Código SQL para calcular os livros comprados por um cliente

```sql
-- TOP 10 de livros mais vendidos (por quantidade)
DELIMITER $$
CREATE PROCEDURE top10Livros ()
BEGIN
    SELECT Livro.codLivro, Livro.titulo, sum(compra.Quantidade) AS qtdVendida
    FROM Livro
    JOIN Compra ON Livro.codLivro = Compra.CodLivro
    GROUP BY livro.titulo
    ORDER BY qtdVendida DESC
    LIMIT 10;
END
```

Figura 40 - Código SQL para calcular os 10 livros mais vendidos

```sql
-- 1 Dado um codLivro retorna o seu preco
DELIMITER $$
CREATE FUNCTION getPrecoLivro (idLivro INT)
    RETURNS DECIMAL(5,2)
    DETERMINISTIC
BEGIN
    DECLARE precoLivro DECIMAL(5,2);
    SELECT preco INTO precolivro FROM Livro
        WHERE livro.codLivro = idLivro;
RETURN precoLivro;
END
```

Figura 41 - Função que dado um código de Livro retorna o preço

```sql
-- 2 Dado um NrCliente retorna se é premium ou nao
-- 1 = TRUE, 0 = FALSE
DELIMITER $$
CREATE FUNCTION isPremium (idCliente INT)
    RETURNS BOOLEAN
    DETERMINISTIC
BEGIN
    RETURN (idCliente IN (SELECT nrCliente FROM Premium));
END
```

Figura 42 - Função que dado um número de cliente indica se é premium

```sql
-- 3 dado um NrCliente retorna o valorGasto
DELIMITER $$
CREATE FUNCTION getValorGastoCliente (idCliente INT)
    RETURNS DECIMAL(5,2)
    DETERMINISTIC
BEGIN
    DECLARE faturacao DECIMAL(5,2);
    SELECT valorGasto INTO faturacao FROM Cliente
    WHERE Cliente.nrCliente = idCliente;
    RETURN faturacao;
END
```

Figura 43 - Função que dado um número de cliente retorna o valor gasto

```sql
-- 4 dado um NrCliente retorna o quantidadeComprada
DELIMITER $$
CREATE FUNCTION getQuantidadeCompradaCliente (idCliente INT)
    RETURNS INT
    DETERMINISTIC
BEGIN
    DECLARE quantidade INT;
    SELECT quantidadeComprada INTO quantidade FROM Cliente
    WHERE Cliente.nrCliente = idCliente;
    RETURN quantidade;
END $$
```

Figura 44 - Função que dado um número de cliente retorna a qd. comprada

```sql
-- 5 dado um NrCliente retorna o desconto (verifica 1º se é premium)
DELIMITER $$
CREATE FUNCTION getDesconto (idCliente INT)
    RETURNS DECIMAL(3,2)
    DETERMINISTIC
BEGIN
    DECLARE discount DECIMAL (3,2);
    SET discount = 0;
    IF (isPremium(idCliente))
    THEN
        SELECT desconto INTO discount FROM Premium
        WHERE Premium.nrCliente = idCliente;
    END IF;
    RETURN discount;
END $$
```

Figura 45 - Função que dado um número de cliente indica o seu desconto

```sql
-- 6 dado um ano retorna o total Faturado
DELIMITER $$
CREATE FUNCTION getFaturacaoAno (ano INT)
    RETURNS DECIMAL (7,2)
    DETERMINISTIC
BEGIN
    DECLARE faturacao DECIMAL (7,2);
    SELECT sum(Encomenda.valorTotal) INTO Faturacao FROM Encomenda
    WHERE year(Encomenda.data) = ano;
    RETURN faturacao;
END $$
-- DROP FUNCTION getFaturacaoAno;
    SELECT getFaturacaoAno(2020);
```

Figura 46 - Função que dado um ano retorna o total faturado

```sql
-- 7 dado um mes e um ano retorna o total faturado
DELIMITER $$
CREATE FUNCTION getFaturacaoMesAno (mes INT, ano INT)
    RETURNS DECIMAL (7,2)
    DETERMINISTIC
BEGIN
    DECLARE faturacao DECIMAL (7,2);
    SELECT sum(Encomenda.valorTotal) INTO Faturacao FROM Encomenda
    WHERE month(Encomenda.data) = mes AND year(Encomenda.data) = ano;
    RETURN faturacao;
END $$
```

Figura 47 - Função que dado um mês e um ano retorna o total faturado

```sql
-- 8 dado um codLivro retorna retorna a quatidadeVendida
DELIMITER $$
CREATE FUNCTION getQuantidadeVendidaLivro (idLivro INT)
    RETURNS INT
    DETERMINISTIC
BEGIN
    DECLARE quantidade INT;
    SELECT sum(Compra.quantidade) INTO quantidade FROM Compra
    WHERE Compra.codLivro = idLivro;
    RETURN quantidade;
END $$
```

Figura 48 - Função que dado um código de um livro retorna a qd. vendida

```sql
-- 9 dado um idAutor retorna o nº de livros vendidos
DELIMITER $$
CREATE FUNCTION getQuantidadeVendidaAutor (idAutor INT)
    RETURNS INT
    DETERMINISTIC
BEGIN
    DECLARE quantidade INT;
    SELECT sum(Compra.quantidade) INTO quantidade FROM Compra
    JOIN Livro ON Compra.codLivro = Livro.codLivro
    JOIN Autor ON Livro.idAutor = Autor.idAutor

    WHERE Autor.idAutor = idAutor;
    RETURN quantidade;
END $$
```

Figura 49 - Função que dado um ID de um autor retorna o nr. de livros vendidos

5.5. Escolha, definição e caracterização de índices em SQL

Devido à dimensão da nossa base de dados consideramos que não é necessária a criação de índices. De facto, trata-se de uma Livraria *online*, pelo que, neste momento, a quantidade de dados não é muito elevada como veremos de seguida.

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

Entidade	Atributos	Tipo	Tamanho
Livro	codLivro	Valor inteiro positivo	4 Bytes
	ISBN	17 Caracteres variáveis	17 Bytes
	Titulo	45 Caracteres variáveis	45 Bytes
	Preço	Valor decimal positivo Float(5,2)	4 Bytes
	Ano	Valor inteiro positivo	4 Bytes
	Língua	45 Caracteres variáveis	45 Bytes
	Género	45 Caracteres variáveis	45 Bytes
	Quantidade	Valor inteiro positivo	4 Bytes
	Edição	45 Caracteres variáveis	45 Bytes

Tabela 9 - Tamanho da entidade Livro

Como existem inicialmente 10 livros, a tabela *Livro* irá ocupar 10 * 213 *Bytes*, o que perfaz 2130 *Bytes*.

Entidade	Atributos	Tipo	Tamanho
Cliente	NIF	45 Caracteres variáveis	45 Bytes
	Nome	45 Caracteres variáveis	45 Bytes
	"e-mail"	45 Caracteres variáveis	45 Bytes
	nrCliente	Valor inteiro positivo	4 Bytes
	Rua	45 Caracteres variáveis	45 Bytes
	Concelho	45 Caracteres variáveis	45 Bytes
	CodigoPostal	45 Caracteres variáveis	45 Bytes
	QuantidadeComprada	Valor inteiro positivo	4 Bytes
	ValorGasto	Valor decimal positivo Float(5,2)	4 Bytes

Tabela 10 - Tamanho da entidade Cliente

Como existem inicialmente 6 clientes, a tabela *Cliente* irá ocupar 6 * 282 *Bytes*, o que perfaz 1692 *Bytes*.

Entidade	Atributos	Tipo	Tamanho
Autor	idAutor	45 Caracteres variáveis	45 Bytes
	Nome	45 Caracteres variáveis	45 Bytes
	Idade	Valor Positivo inteiro	4 Bytes
	DataNascimento	Data	3 Bytes

Tabela 11 - Tamanho da entidade Autor

Como existem inicialmente 6 autores, a tabela *Autor* irá ocupar 6 * 97 *Bytes*, o que perfaz 582 *Bytes*.

Entidade	Atributos	Tipo	Tamanho
Editora	Nome	45 Caracteres variáveis	45 Bytes
	Site	45 Caracteres variáveis	45 Bytes
	E-mail	45 Caracteres variáveis	45 Bytes

Tabela 12 - Tamanho da entidade Editora

Como existem inicialmente 5 editoras, a tabela *Editora* irá ocupar 5 * 135 *Bytes*, o que perfaz 675 *Bytes*.

Entidade	Atributos	Tipo	Tamanho
Premium	Desconto	Valor Decimal Positivo Float(5,2)	4 Bytes

Tabela 13 - Tamanho da entidade Premium

Como existem inicialmente 2 clientes premium, a tabela *Premium* irá ocupar 2 * 4 *Bytes*, o que perfaz 8 *Bytes*.

Entidade	Atributos	Tipo	Tamanho
Encomenda	Data	Data	3 Bytes
	ValorTotal Float(5,2)	Valor Decimal Positivo	4 Bytes
	NrEncomenda	Valor Positivo inteiro	4 Bytes
	TamanhoEncomenda	Valor Positivo inteiro	4 Bytes

Tabela 14 - Tamanho da entidade Encomenda

Como existem inicialmente 4 encomendas, a tabela *Encomenda* irá ocupar 4 * 16 *Bytes*, o que perfaz 64 *Bytes*.

Entidade	Atributos	Tipo	Tamanho
Telefone	nrTelefone	Valor Positivo inteiro	4 Bytes

Tabela 15 - Tamanho da entidade Telefone

Como existem inicialmente 8 números de telefone, a tabela *Telefone* irá ocupar 8 * 4 *Bytes*, o que perfaz 24 *Bytes*.

Entidade	Atributos	Tipo	Tamanho
Compra	Quantidade	Valor Positivo inteiro	4 Bytes
	Preço	Valor Decimal Positivo	4 Bytes
	PrecoQuantidade	Valor Decimal Positivo	4 Bytes

Tabela 16 - Tamanho da entidade Compra

Como existem inicialmente 8 compras realizadas, a tabela *Compra* irá ocupar 8 * 12 *Bytes*, o que perfaz 96 *Bytes*.

Desta forma, a base de dados necessita de 5271 *Bytes* ≈ 5.397504 *Kbytes*. Supondo um crescimento de 10% ao ano é possível concluir que a base de dados aumentará em aproximadamente 0.54 *Kbytes* por ano.

5.7. Definição e caracterização das vistas de utilização em SQL

Como já referimos tivemos a necessidade de criar utilizadores. Para tal, decidimos criar as vistas que achamos necessários para que estes vejam apenas os pontos de que têm acesso.

```
CREATE USER 'admin'@'localhost'
    IDENTIFIED BY 'admin';

GRANT ALL PRIVILEGES ON livrariaonline.*
    TO 'admin'@'localhost';
```

Figura 50 - Criação do utilizador

```
-- VIEW E GRANT do dono da livraria sobre o stock dos seus livros
CREATE VIEW viewDono_livros AS
    SELECT codLivro,Titulo, quantidade
        FROM Livro;
```

Figura 51 - Criação da vista do dono em relação à tabela livro

```
-- VIEW E GRANT do dono da livraria em relação aos clientes.
CREATE VIEW viewDono_Clientes AS
    SELECT NrCliente, Nome, Email, ValorGasto
        FROM Cliente;
```

Figura 52 - Criação da vista do dono em relação à tabela clientes

```
-- VIEW E GRANT do dono da livraria e das encomendas
CREATE VIEW viewDono_Encomendas AS
    SELECT NrEncomenda, TamanhoEncomenda, ValorTotal, NrCliente
        FROM Encomenda;
```

Figura 53 - Criação da vista do dono em relação à tabela encomendas

5.8. Revisão do sistema implementado

Por fim foi marcada uma última reunião com a Sra. Alice, de maneira a mostrar-lhe o produto final e deixar que este o testasse pessoalmente. Uma vez que, para além de satisfazer todos os requisitos encomendados e todas as funcionalidades operarem corretamente quando testadas, este programa tem ainda funcionalidades extras, o que deixou a Sra. Alice bastante satisfeita.

Desta forma se chegou aquilo que era o fim deste projeto para a Sra. Alice, com ambos os lados satisfeitos com o produto final.

6. Conclusões e Trabalho Futuro

Ao longo do projeto deparamo-nos com diversos desafios que pretendemos caracterizar de forma exaustiva nesta secção.

Relativamente à primeira parte do trabalho, conseguimos concluir que é crucial fazer uma boa análise de requisitos que o sistema deve respeitar, uma vez que esta vai influenciar a estruturação e implementação do sistema de base de dados. Nesse aspeto sentimos que a nossa primeira análise não estava muito longe daquilo que acabamos por implementar, e nesse aspeto não tivemos grandes dificuldades.

Após o levantamento e análise de requisitos, identificámos as diferentes entidades fundamentais, os seus atributos e os relacionamentos existentes entre estas. Foi importante também realizar uma boa interpretação dos atributos. De seguida, determinámos os domínios dos atributos e identificámos as chaves primárias, candidatas e alternativas correspondentes a cada entidade. Verificámos também se existiam redundâncias no modelo, à qual conseguimos concluir que não existia qualquer tipo de problema. Posto isto, conseguimos elaborar o modelo conceptual para o nosso trabalho.

No que diz respeito à segunda parte, no modelo lógico, não sentimos grandes adversidades uma vez que a passagem do conceptual para este fez-se à custa de certos procedimentos aprendidos nas aulas teórico-práticas. Ainda assim houve certas preocupações do grupo no processo de passagem do modelo conceptual para o modelo lógico, sendo estas relacionadas com a integridade dos dados e a normalização uma vez que são estas que garantem a consistência dos dados e a ausência de redundância.

Finalmente, traduzimos o modelo lógico para o modelo físico, onde tivemos bastante preocupação com as relações base, assim como com as restrições fundamentais para o sistema. Foi importante também fazer uma estimativa do espaço em disco que o sistema ocupa, para compreender se existia algum problema de tamanho excessivo. Por último, definimos os perfis de utilizador, assim como as diferentes regras de acesso. Foi nesta fase que surgiram mais dúvidas, uma vez que, sentimos estar menos preparados a nível prático para criar aquilo que foi definido por nós inicialmente, no entanto e graças a ajuda dos professores e material fornecido sentimos que, na sua generalidade, cumprimos nosso objetivo.

Contudo, consideramos que o trabalho tem os seus pontos fortes e menos bons. Nesse sentido, a contextualização e a fundamentação foram definidas de maneira exímia permitindo maior estabilidade em alguns aspetos. Consideramos também que relativamente ao futuro o processo é fidedigno pois apesar de se tratar de uma base de dados de pequena dimensão permite adicionar os dados necessários ao negócio para a qual foi construída. No entretanto, observamos que um trabalho como este deveria ter sido tratado com um pouco mais antecedência, de modo a, prevenir eventuais erros. Em forma de conclusão, a realização deste projeto permitiu-nos consolidar os conhecimentos adquiridos na unidade curricular, bem como identificar algumas lacunas que pretendemos corrigir futuramente.

7. Referências

Connolly, T. and Begg, C., 2015. *Database Systems*. 6th ed. Pearson.

Tou, J., 1974. *Information Systems*. New York: Plenum Press.

Belo,O.,2020. *Esquemas e Scripts SQL . Available* at: <https://elearning.uminho.pt >

Cite This For Me. 2020. *Save Time And Improve Your Marks With Citethisforme, The No. 1 Citation Tool*. *Available* at: <https://www.citethisforme.com/cite/website>

MySql. Available at: <https://www.mysql.com>

8. Lista de Siglas e Acrónimos

BD Base de Dados
ER Entidade - Relacionamento
NR Número
ID Identidade
SQL *Structured Query Language*

9. Anexos

I. Modelo Físico no *MySQL*

```
-- 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_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema LivrariaOnline
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `LivrariaOnline` ;

-- -----------------------------------------------------
-- Schema LivrariaOnline
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `LivrariaOnline` DEFAULT CHARACTER SET utf8 ;
USE `LivrariaOnline` ;

-- -----------------------------------------------------
-- Table `LivrariaOnline`.`Editora`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `LivrariaOnline`.`Editora` ;

CREATE TABLE IF NOT EXISTS `LivrariaOnline`.`Editora` (
  `IDEditora` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(45) NOT NULL,
  `Website` VARCHAR(60) NULL,
  `Email` VARCHAR(45) NOT NULL,
  UNIQUE INDEX `Nome_UNIQUE` (`Nome` ASC) VISIBLE,
  PRIMARY KEY (`IDEditora`),
  UNIQUE INDEX `IDEditora_UNIQUE` (`IDEditora` ASC) VISIBLE,
  UNIQUE INDEX `Email_UNIQUE` (`Email` ASC) VISIBLE)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `LivrariaOnline`.`Autor`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `LivrariaOnline`.`Autor` ;

CREATE TABLE IF NOT EXISTS `LivrariaOnline`.`Autor` (
  `IDAutor` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(45) NOT NULL,
  `Idade` INT NULL,
  `DataNascimento` DATE NULL,
  PRIMARY KEY (`IDAutor`),
  UNIQUE INDEX `IDAutor_UNIQUE` (`IDAutor` ASC) VISIBLE)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `LivrariaOnline`.`Livro`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `LivrariaOnline`.`Livro` ;

CREATE TABLE IF NOT EXISTS `LivrariaOnline`.`Livro` (
  `CodLivro` INT NOT NULL AUTO_INCREMENT,
  `Titulo` VARCHAR(45) NOT NULL,
  `Preco` DECIMAL(5,2) NOT NULL,
  `Ano` INT NULL,
  `Genero` VARCHAR(45) NULL,
  `Lingua` VARCHAR(45) NOT NULL,
  `Edicao` INT NULL,
  `Quantidade` INT NOT NULL,
  `ISBN` VARCHAR(15) NOT NULL,
  `IDEditora` INT NOT NULL,
  `IDAutor` INT NOT NULL,
  PRIMARY KEY (`CodLivro`),
  UNIQUE INDEX `ISBN_UNIQUE` (`ISBN` ASC) VISIBLE,
  INDEX `fk_Livro_Editora1_idx` (`IDEditora` ASC) VISIBLE,
  INDEX `fk_Livro_Autor1_idx` (`IDAutor` ASC) VISIBLE,
  UNIQUE INDEX `CodLivro_UNIQUE` (`CodLivro` ASC) VISIBLE,
  CONSTRAINT `fk_Livro_Editora1`
    FOREIGN KEY (`IDEditora`)
    REFERENCES `LivrariaOnline`.`Editora` (`IDEditora`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Livro_Autor1`
    FOREIGN KEY (`IDAutor`)
    REFERENCES `LivrariaOnline`.`Autor` (`IDAutor`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```sql
-- -----------------------------------------------------
-- Table `LivrariaOnline`.`Cliente`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `LivrariaOnline`.`Cliente` ;

CREATE TABLE IF NOT EXISTS `LivrariaOnline`.`Cliente` (
  `NrCliente` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(45) NOT NULL,
  `NIF` INT NOT NULL,
  `Email` VARCHAR(45) NOT NULL,
  `QuantidadeComprada` INT NOT NULL,
  `ValorGasto` DECIMAL(5,2) NOT NULL,
  `RecomendadePor` INT NULL,
  `Rua` VARCHAR(45) NOT NULL,
  `Concelho` VARCHAR(45) NOT NULL,
  `CodigoPostal` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`NrCliente`),
  UNIQUE INDEX `NrCliente_UNIQUE` (`NrCliente` ASC) VISIBLE,
  UNIQUE INDEX `NIF_UNIQUE` (`NIF` ASC) VISIBLE,
  INDEX `fk_Cliente_Cliente1_idx` (`RecomendadePor` ASC) VISIBLE,
  CONSTRAINT `fk_Cliente_Cliente1`
    FOREIGN KEY (`RecomendadePor`)
    REFERENCES `LivrariaOnline`.`Cliente` (`NrCliente`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `LivrariaOnline`.`Premium`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `LivrariaOnline`.`Premium` ;

CREATE TABLE IF NOT EXISTS `LivrariaOnline`.`Premium` (
  `NrCliente` INT NOT NULL,
  `Desconto` DECIMAL(4,2) NOT NULL,
  INDEX `fk_Premium_Cliente1_idx` (`NrCliente` ASC) VISIBLE,
  CONSTRAINT `fk_Premium_Cliente1`
    FOREIGN KEY (`NrCliente`)
    REFERENCES `LivrariaOnline`.`Cliente` (`NrCliente`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `LivrariaOnline`.`Telefone`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `LivrariaOnline`.`Telefone` ;

CREATE TABLE IF NOT EXISTS `LivrariaOnline`.`Telefone` (
  `IDEditora` INT NOT NULL,
  `Telefone` INT NOT NULL,
  PRIMARY KEY (`Telefone`),
  UNIQUE INDEX `Telefone_UNIQUE` (`Telefone` ASC) VISIBLE,
  INDEX `fk_Contacto_Editora1_idx` (`IDEditora` ASC) VISIBLE,
  CONSTRAINT `fk_Contacto_Editora1`
    FOREIGN KEY (`IDEditora`)
    REFERENCES `LivrariaOnline`.`Editora` (`IDEditora`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `LivrariaOnline`.`Encomenda`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `LivrariaOnline`.`Encomenda` ;

CREATE TABLE IF NOT EXISTS `LivrariaOnline`.`Encomenda` (
  `NrEncomenda` INT NOT NULL AUTO_INCREMENT,
  `ValorTotal` DECIMAL(5,2) NOT NULL,
  `TamanhoEncomenda` INT NOT NULL,
  `Data` DATETIME NOT NULL,
  `NrCliente` INT NOT NULL,
  PRIMARY KEY (`NrEncomenda`),
  UNIQUE INDEX `NrEncomenda_UNIQUE` (`NrEncomenda` ASC) VISIBLE,
  INDEX `fk_Encomenda_Cliente1_idx` (`NrCliente` ASC) VISIBLE,
  CONSTRAINT `fk_Encomenda_Cliente1`
    FOREIGN KEY (`NrCliente`)
    REFERENCES `LivrariaOnline`.`Cliente` (`NrCliente`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `LivrariaOnline`.`Compra`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `LivrariaOnline`.`Compra` ;

CREATE TABLE IF NOT EXISTS `LivrariaOnline`.`Compra` (
  `CodLivro` INT NOT NULL,
  `Preco` DECIMAL(5,2) NOT NULL,
  `Quantidade` INT NOT NULL,
  `PrecoPQ` DECIMAL(5,2) NOT NULL,
  `NrEncomenda` INT NOT NULL,
  INDEX `fk_Compra_Encomenda1_idx` (`NrEncomenda` ASC) VISIBLE,
  INDEX `fk_Compra_Livro1_idx` (`CodLivro` ASC) VISIBLE,
  CONSTRAINT `fk_Compra_Encomenda1`
    FOREIGN KEY (`NrEncomenda`)
    REFERENCES `LivrariaOnline`.`Encomenda` (`NrEncomenda`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Compra_Livro1`
    FOREIGN KEY (`CodLivro`)
    REFERENCES `LivrariaOnline`.`Livro` (`CodLivro`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```sql
USE `LivrariaOnline` ;

-- -----------------------------------------------------
-- Placeholder table for view `LivrariaOnline`.`view1`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `LivrariaOnline`.`view1` (`id` INT);

-- -----------------------------------------------------
-- View `LivrariaOnline`.`view1`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `LivrariaOnline`.`view1`;
DROP VIEW IF EXISTS `LivrariaOnline`.`view1` ;
USE `LivrariaOnline`;


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

II. Povoamento da base de dados

```sql
-- Esquema: "Mercearia"
USE LivrariaOnline;

--
-- Permissão para fazer operações de remoção de dados.
SET SQL_SAFE_UPDATES = 0;

--
-- Povoamento da tabela "Livro"
INSERT INTO Livro
    (CodLivro, Lingua, Genero, Titulo, ISBN, Edicao, Ano, Preco, Quantidade, IDEditora, IDAutor)
    VALUES
        (1, 'Português', 'História', 'Guerra Mundial I', '900-0-7334-2609-4', 1 , 2010, 014.99,3,1,1),
        (2, 'Português', 'Matemática', 'MNOL', '901-0-7334-2609-4', 1 , 2012, 030.00,4,1,3),
        (3, 'Português', 'Romance', 'Romeu e Julieta', '902-0-7334-2609-4', 1 , 1999, 023.99,3,5,1),
        (4, 'Francês', 'Francês', 'Le crepe et le baguette', '903-0-7334-2609-4', 2 , 2010, 035.99,2,4,3),
        (5, 'Português', 'Mistério', 'Código Da Vinci', '904-0-7334-2609-4', 2 , 2019, 100.00,1,2,6),
        (6, 'Inglês', 'Classico', 'How To Kill a Mockingbird', '905-0-7334-2609-4', 1 , 1989, 027.99,2,3,2),
        (7, 'Inglês', 'Matemática', 'ISD', '906-0-7334-2609-4', 1 , 2000, 057.00,10,3,4),
        (8, 'Mandarim', 'História', 'História Chinesa', '907-0-7334-2609-4', 3 , 2020, 120.00,1,5,5),
        (9, 'Português', 'Classico', '1984', '908-0-7334-2609-4', 2 , 1999, 040.00,3,2,6),
        (10, 'Inglês', 'Inglês', 'English for Beginners', '909-0-7334-2609-4', 1 , 2020, 010.00,10,3,2)
        ;
--
-- DELETE FROM Livro;
-- SELECT * FROM Livro;
-- Povoamento da tabela "Editora"
INSERT INTO Editora
    (IDEditora, Nome, Website, Email)
    VALUES
        (1,'Porto Editora', 'potoeditora.pt', 'portoeditora@gmail.com'),
        (2,'Raiz', 'raiz.pt', 'raizeditora@gmail.com'),
        (3,'Manchester Editor', 'manchestereditor.com', 'manchestereditor@gmail.com'),
        (4,'Paris Editor', null , 'pariseditor@gmail.com'),
        (5,'Arial', 'arialeditora.pt', 'arialeditora@gmail.com')
        ;
--
-- DELETE FROM Editora;
-- SELECT * FROM Editora;


INSERT INTO Telefone
    (IDEditora, Telefone)
    VALUES
        (1,  678546234),
        (1,  142356429),
        (2,  998765420),
        (3,  675008976),
        (4,  776879332),
        (5,  465423890),
        (5,  866543420),
        (5,  654545218)
        ;
--
-- DELETE FROM Telefone;
-- SELECT * FROM Telefone;
-- Povoamento da tabela "Autor"
INSERT INTO Autor
    (IDAutor, Nome, Idade, DataNascimento)
    VALUES
        (1,'José Amado', 30 , '1990/10/05'),
        (2,'Jack Miller', null , '1956/03/23'),
        (3,'Maria Blanco', 67 , '1953/04/05'),
        (4,'Harry Wilson', null , null),
        (5,'Park Lee', null , '1938/10/31'),
        (6,'Miguel Oliveira', 71 , '1949/10/31')
        ;
--
-- DELETE FROM Autor;
-- SELECT * FROM Autor;



-- Povoamento da tabela "Compra"
INSERT INTO Compra
    (Preco, Quantidade, PrecoPQ, NrEncomenda, CodLivro)
    VALUES
        (100.00, 2, 200.00, 1, 5),
        (010.00, 1, 010.00, 1, 10),
        (030.00, 2, 060.00, 2, 2),
        (014.99, 1, 014.99, 3, 1),
        (023.99, 1, 023.99, 3, 3),
        (057.00, 1, 057.00, 3, 7),
        (057.00, 1, 057.00, 4, 7),
        (010.00, 1, 010.00, 4, 9)
        ;
--
-- DELETE FROM Compra;
-- SELECT * FROM Compra;
```

```sql
-- Povoamento da tabela "Encomenda"
INSERT INTO Encomenda
    (NrEncomenda, ValorTotal, TamanhoEncomenda, Data, NrCliente)
    VALUES
        (1, 210.00, 3, '2020/10/2', 1),
        (2, 060.00, 2, '2020/9/11', 3),
        (3, 095.98, 3, '2018/10/23', 5),
        (4, 067.00, 2, '2019/9/12', 3)
        ;
--
-- DELETE FROM Encomenda;
-- SELECT * FROM Encomenda;


-- Povoamento da tabela "Cliente"
INSERT INTO Cliente
    (NrCliente, Nome, NIF, Email, QuantidadeComprada, ValorGasto, Recomendacoes, Rua, Concelho, CodigoPostal)
    VALUES
    (1, 'José Batista', 44321759, 'jose@gmail.com', 3, 210.00, null, 'Rua Oliveirinha', 'Viana do Castelo', '7255-135'),
    (2, 'Alexandre Pereira', 123785988, 'alexpereira@hotmail.com', 0, 000.00, 1, 'Rua Joao Paulo', 'Guimaraes', '4790-543'),
    (3, 'Pedro Cabral', 987012549, 'pedro99@portugalmail.pt', 4, 127.00, null, 'Rua Dória', 'Braga', '3600-123'),
    (4, 'José Silva', 455653767, 'jose123@gmail.com', 0, 000.00, 2, 'Rua Joao Paulo', 'Guimarães', '4790-534'),
    (5, 'Rita Oliveira', 978098123, 'ritaoliveira2000@hotmail.com', 3, 095.98, null, 'Avenida da Liberdade', 'Lisboa', '3550-123'),
    (6, 'Margarida Miranda', 123677854, 'guidamiranda@gmail.com', 0, 000.00, 2, 'Avenida 25 de Abril', 'Porto', '4300-53')
    ;
--
-- DELETE FROM Cliente;
-- SELECT * FROM Cliente;
-- Povoamento da Tabela "Premium"
INSERT INTO Premium
    (NrCliente, Desconto)
    VALUES
    (3,00.15),
    (5,00.15)
    ;
--
-- DELETE FROM Premium;
-- SELECT * FROM Premium;
```

III. Funcionalidades extras

```sql
-- Seleciona os livros de um autor (id = 1) dado organizados por preço
SELECT Autor.nome,Livro.titulo,Livro.preco
FROM Livro JOIN Autor ON Livro.idAutor = autor.idAutor
WHERE Livro.idAutor =1
ORDER BY preco ASC;

-- Seleciona os livros de um autor (id = 1) dado organizados por ano
SELECT Autor.nome,Livro.titulo,Livro.ano
FROM Livro JOIN Autor ON Livro.idAutor = autor.idAutor
WHERE Livro.idAutor =1
ORDER BY ano ASC;

-- Seleciona os livros de cada autor organizados por preço
SELECT Livro.titulo, Autor.nome AS Autor, Livro.preco
FROM Livro JOIN Autor ON Livro.IDAutor = autor.IDAutor
ORDER BY preco ASC;
-- Livro mais vendido (em caso de empate, desempata pelo preço)
SELECT Titulo, sum(compra.quantidade) AS QtdVendida, compra.Preco*sum(compra.quantidade) AS PrecoTotal
    FROM livro
    JOIN Compra ON livro.CodLivro = compra.CodLivro
    GROUP BY livro.titulo
    ORDER BY QtdVendida DESC, PrecoTotal DESC
    LIMIT 1;

-- Autores mais vendidos por ordem decrescente de quantidade vendida
SELECT Autor.nome, sum(compra.quantidade) AS QtdVendida FROM livro
    JOIN Compra ON livro.CodLivro = compra.CodLivro
    JOIN Autor ON livro.idAutor = Autor.idAutor
    GROUP BY autor.idAutor
    ORDER BY QtdVendida DESC;

-- Autor mais vendido
SELECT Autor.nome, sum(Compra.quantidade) AS QtdVendida FROM livro
    JOIN Compra ON livro.CodLivro = compra.CodLivro
    JOIN Autor ON livro.idAutor = Autor.idAutor
    GROUP BY autor.idAutor
    ORDER BY QtdVendida DESC
    LIMIT 1;
```

```sql
-- Ordenar os Concelhos com mais clientes
SELECT concelho, COUNT(nome) AS NrClientes FROM Cliente
GROUP BY concelho
ORDER BY nrClientes DESC;

-- Ordenar os Concelhos por quantidade de livros vendidos
SELECT concelho, sum(quantidadeComprada) FROM Cliente
GROUP BY concelho
ORDER BY quantidadeComprada DESC;

-- Ordenar os Concelhos por ValorGasto (Faturacao)
SELECT concelho, sum(valorGasto) AS Faturacao FROM Cliente
GROUP BY concelho
ORDER BY Faturacao DESC;
```

```sql
-- Apresenta o total faturado por ano
SELECT year(Encomenda.data) AS Ano, sum(Encomenda.valorTotal) AS Faturacao
FROM Encomenda
GROUP BY ano
ORDER BY ano DESC;

-- Apresenta os clientes de cada Editora ordendados por valor gasto nessa editora
SELECT Editora.nome AS Editora, Cliente.nome AS Cliente, sum(Compra.precoPQ) AS ValorGasto
From Editora JOIN Livro ON Editora.IDEditora = Livro.IDEditora
JOIN Compra ON Livro.CodLivro = Compra.CodLivro
JOIN Encomenda ON Compra.NrEncomenda = Encomenda.NrEncomenda
JOIN Cliente ON Encomenda.NrCliente = Cliente.NrCliente
Group BY Editora.nome, Cliente.nome
Order By  Editora, Cliente.valorGasto Desc;
```