

**Um Belo PC**

**Universidade do Minho**

Mestrado Integrado em Engenharia Informática

Licenciatura em Ciências da Computação

**Unidade Curricular de**

**Bases de Dados**

Ano Lectivo de 2018/2019

**Afonso Sousa A74196**

**Gonçalo Moreira A73591**

**José Ramos A73855**

**Hak Ngin E8627**

**Pedro Terra A73182**

Novembro de 2017

**BD**

|  |  |
| --- | --- |
| Data de Recepção |  |
| Responsável |  |
| Avaliação |  |
| Observações |  |

**Afonso Sousa A74196**

**Gonçalo Moreira A73591**

**José Ramos A73855**

**Hak Ngin E8627**

**Pedro Terra A73182**

Novembro de 2017

**Um Belo PC**

# Resumo

Durante a primeira metade do semestre fomos desenvolvendo um projeto no âmbito da UC como forma de aplicar na prática os conhecimentos aprendidos nas aulas sobre o modelo relacional e o sistema de gestão de bases de dados MySQL. Foi-nos pedido para inventar um problema para o qual a solução fosse uma base de dados relacional, ficando a nosso encargo analisar o problema e determinar os requisitos da solução, modelar essa mesma solução e implementá-la.

Como problema escolhemos uma loja de componentes para computadores, para a qual será necessário ter um sistema que permita registar clientes, vendas e devoluções, consultar o catálogo. Deve ser possível fazer uma consulta eficaz do catáogo por forma a ser o mais fácil possível aos clientes encontrar os produtos que querem comprar. Com este objetivo em mente (e porque se trata de um cliente fictício) começámos por determinar requisitos para a solução, seguindo-se uma fase de modelação conceptual da solução, um modelo lógico e finalmente uma implementação física.

Toda a análise de requisitos foi feita sem qualquer contacto com um cliente real, pelo que os requisitos foram definidos conforme aquilo que nos parece ser o modelo de negócio de lojas desses produtos (PCDIGA, PcComponente, entre outras).

O esquema físico foi testado localmente com povoamentos preparados por nós para o efeito. O nosso sistema de base de dados foi feito de acordo com o com os objetivos para este trabalho e com os conteúdos lecionados nas aulas.

Ao longo deste relatório vamos tentar mostrar de que forma evoluímos os modelos ao longo do desenvolvimento.

**Área de Aplicação:** Implementação de sistemas de bases de dados.

**Palavras-Chave**: Base de Dados, Modelo Conceptual, Modelo Lógico, Modelo Físico, MySQL, JAVA, Requisitos, BrModelo.

Tabela de Conteúdos

[Resumo IV](#__RefHeading___Toc2881_2551110197)

[Índice de Figuras VII](#__RefHeading___Toc1651_1061193116)

[Índice de Tabelas VII](#__RefHeading___Toc1653_1061193116)

[1. Definição do Sistema 1](#__RefHeading___Toc535645383)

[1.1 Contexto de aplicação do sistema 1](#__RefHeading___Toc535645384)

[1.2 Fundamentação da implementação da base de dados 2](#__RefHeading___Toc535645385)

[1.3 Análise da viabilidade do processo 2](#__RefHeading___Toc535645386)

[2. Levantamento e Análise de Requesitos 3](#__RefHeading___Toc535645388)

[2.1. Método de levantamento de requisitos 3](#__RefHeading___Toc535645389)

[2.2 Requisitos levantados 3](#__RefHeading___Toc535645390)

[2.3. Análise geral dos requisitos 4](#__RefHeading___Toc535645391)

[3. Modelação Conceptual 5](#__RefHeading___Toc535645395)

[3.1. Abordagem à modelação 5](#__RefHeading___Toc1661_1061193116)

[3.2. Entidades do sistema 5](#__RefHeading___Toc1663_1061193116)

[3.3. Relacionamentos do sistema 6](#__RefHeading___Toc1665_1061193116)

[3.4. Atributos do sistema 6](#__RefHeading___Toc1667_1061193116)

[3.5. Detalhe e generalização de entidades 7](#__RefHeading___Toc1669_1061193116)

[3.6. Diagrama ER 7](#__RefHeading___Toc1671_1061193116)

[3.7. Validação do modelo com o cliente 7](#__RefHeading___Toc76_512549187)

[4. Modelação Lógica 8](#__RefHeading___Toc78_512549187)

[4.1. Construção e validação do modelo 8](#__RefHeading___Toc80_512549187)

[4.2. Desenho do modelo lógico 8](#__RefHeading___Toc82_512549187)

[4.3. Validação do modelo através da normalização 9](#__RefHeading___Toc84_512549187)

[4.4. Validação do modelo com interrogações 10](#__RefHeading___Toc86_512549187)

[4.5. Validação do modelo com transações 11](#__RefHeading___Toc88_512549187)

[4.6. Revisão do modelo lógico com o cliente 11](#__RefHeading___Toc90_512549187)

[5. Implementação Física 12](#__RefHeading___Toc936_512549187)

[5.1 Seleção do sistema de gestão de bases de dados 12](#__RefHeading___Toc938_512549187)

[5.2. Tradução do esquema lógico para o SGBD 12](#__RefHeading___Toc940_512549187)

[5.3.Tradução das interrogações para SQL 13](#__RefHeading___Toc942_512549187)

[5.4. Tradução de transações em SQL 14](#__RefHeading___Toc944_512549187)

[5.5. Escolha, definição e caracterização de índices em SQL 14](#__RefHeading___Toc946_512549187)

[5.6. Estimativa do espaço ocupado em disco e taxa de crescimento anual 14](#__RefHeading___Toc948_512549187)

[5.7. Definição e caracterização de vistas de utilização 15](#__RefHeading___Toc1192_512549187)

[5.8. Mecanismos de segurança 16](#__RefHeading___Toc950_512549187)

[5.9. Revisão do sistema implementado com o cliente 16](#__RefHeading___Toc1194_512549187)

[6. Conclusões e Trabalho Futuro 18](#__RefHeading___Toc1196_512549187)

[Anexos 21](#__RefHeading___Toc1675_1061193116)

# Índice de Figuras

Índice de Figuras

[Figura 1: Diagrama ER 7](#FIGURA!0|sequence)

[Figura 2: Modelo lógico gerado com o brModelo 8](#FIGURA!1|sequence)

[Figura 3: Modelo lógico após correções 9](#FIGURA!3|sequence)

[Figura 4: Modelo na terceira fórmula normal de Codd 10](#FIGURA!2|sequence)

[Figura 5: Modelo lógico final, usado para gerar o físico no MySQL Workbench 13](#FIGURA!4|sequence)

[Figura 6: Exemplo de queries implementadas 13](#FIGURA!5|sequence)

[Figura 7: Exemplos de transações implementadas 14](#FIGURA!6|sequence)

[Figura 8: Exemplo de vistas implementadas 16](#FIGURA!7|sequence)

[Figura 9: Permissões e utilizadores da base de dados 16](#FIGURA!8|sequence)

# Índice de Tabelas

[Tabela 1 - Entidades do modelo ER. 5](#TABELA!0|sequence)

[Tabela 2 - Relacionamentos do modelo ER. 6](#TABELA!1|sequence)

[Tabela 3 - Atributos do modelo ER. 6](#TABELA!2|sequence)

[Tabela 4 - Tamanhos das entidades em bytes 15](#TABELA!3|sequence)

# 1. Definição do Sistema

Para este projeto vamos desenvolver uma base de dados relacional com o gestor M*ySQL,* que permita servir uma loja de venda de componentes fictícia. A base de dados será feita de acordo com base num modelo entidade relacionamento também feito por nós.

## 1.1 Contexto de aplicação do sistema

Desde tenra idade que nós fantasiamos com máquinas potentes, bonitas, com teras de RAM e leitores de disquetes. Porém, nem tudo são rosas neste mundo, e nós, tal como muitos outros jovens rapazes, tínhamos muito tempo, muita vontade, mas pouco dinheiro para seguir os nossos sonhos.

Coincidência das coincidências, enquanto alunos de Bases de Dados, apareceu-nos uma proposta para ajudar uma nova startup com um modelo de negócio absolutamente revolucionário: vender componentes a clientes. O que a “Um Belo PC” precisa é de alguém que os ajude a gerir o triplo Cliente-Venda-Produto, bem como prestadores de apoio moral e umas disquetes para pendurar ao pescoço. A “Um Belo PC” tem apenas uma loja a retalho, um funcionário a tempo inteiro, alguns clientes já em carteira (e suas compras/devoluções), bem como um bom inventário de componentes.

Nós, como alunos de Bases de Dados que somos, ficámos super entusiasmados por saber que conseguimos ajudar esta futura Amazon dos componentes. Fizemos três vivas, penduramos uma disquete no funcionário e metemos mãos no papel para escrever os requisitos como bons estudantes de Bases de Dados que somos.

## 1.2 Fundamentação da implementação da base de dados

A “Um Belo PC” quer fornecer a melhor experiência possível aos seus clientes no que toca a consultar o inventário de componentes. Por causa disso será necessário ter um bom sistema de procura implementado, podendo um produto ser procurado por um ou mais fatores tais como código de barras, marca, modelo, popularidade, preço e classificação dos consumidores.

De forma a manter a loja operacional, o nosso cliente precisa de ter atenção a três fatores importantes para o negócio: o inventário de componentes, os clientes e as transações feitas com a loja. Por imposição do fornecedor, sempre que um cliente não fica satisfeito e decide devolver o produto, esse produto tem de ser entregue ao fornecedor e não pode voltar a ser colocado para venda.

Com este sistema a loja espera também conseguir cumprir as suas obrigações no que toca a pagar impostos, estar de acordo com o regulamento geral de proteção de dados e livrar-se do combustível caderno de vendas, ainda em papel.

## 1.3 Análise da viabilidade do processo

Após uma análise às necessidades da loja, chegamos à conclusão que a melhor solução será implementar uma base de dados de suporte ao negócio. Isto permitirá consultas de inventário extraordinariamente rápidas quando comparadas com a procura num catálogo em papel, e gerir clientes, vendas e devoluções.

Finalmente, uma base de dados permite à “Um Belo PC” livrar-se do risco que é o formato papel logo assim que esteja implementada porque a base de dados pode ser carregada com os dados já existentes.

# 2. Levantamento e Análise de Requisitos

## 2.1. Método de levantamento de requisitos

Tal como em qualquer contrato, falar com o cliente é a melhor forma de saber o que ele pretende. Para a “Um Belo PC”, a nossa principal fonte de informação sobre o negócio é o seu funcionário pois é a única pessoas que conhece em detalhe toda a operação.

Para recolhermos informação realizámos entrevistas por forma a conhecer melhor o negócio. Mesmo com estas entrevistas, não nos foi possível identificar todos os requisitos necessários, pelo que tivemos de contactar com o funcionário várias vezes ao longo do desenvolvimento.

Durante esses contactos, a nossa atenção estava centrada em identificar os nomes mencionados, as suas características e a forma como estas relacionam entre si.

## 2.2 Requisitos levantados

Como resultado das entrevistas e conversas que tivemos com a “Um Belo PC”, foram levantados os seguintes requisitos para o sistema a desenvolver:

### 2.2.1. Requisitos de descrição

1. Cada Cliente tem um com um identificador único, um nome próprio, um apelido, um número de identificação bancária, uma data de nascimento, uma e uma só morada permanente (rua, cidade, número de casa/apartamento e código postal), um email e números de telefones;
2. Cada Produto tem um com um código de barras, uma marca, modelo, tipo de produto e descrição, uma cor, uma data de lançamento, um preço recomendado pelo fabricante, um preço atual e uma média das avaliações dos clientes;
3. Cada Venda é identificada por um id único, sendo cada uma feita a um e um só cliente, relativa a um conjunto não vazio de produtos, guardando a data em que a venda foi feita e o preço total cobrado ao cliente, que pode ser inferior à soma dos preços individuais;
4. Cada Devolução é identificada por um id único, estando associada a uma e uma só venda, relativo a um e um só produto, guardando a data em que a devolução foi feita;
5. Um cliente tem que estar associado a pelo menos uma venda, podendo ou não estar associado a uma devolução;
6. Um produto pode estar associado a, no máximo, uma e uma só venda.
7. Um produto pode estar associado a, no máximo, uma e uma só devolução.

### 2.2.2. Requisitos de exploração

O sistema deverá permitir:

1. saber quanto foi o lucro gerado desde o início do ano;
2. saber quais foram os produtos mais vendidos;
3. saber qual o cliente que mais lucro gerou à loja;
4. registar clientes, devoluções, produtos e vendas;
5. alterar o preço cobrado aos clientes via promoções;
6. consultar o catálogo de produtos para venda, com possibilidade de filtrar via qualquer combinação entre marca, modelo, cor, tipo de produto, classificação, popularidade, data de lançamento e preço;
7. saber os códigos de barra dos produtos comprados por um cliente que foram submetidos para reparação;
8. saber quais os produtos (marca, modelo e data de lançamento) menos fiáveis.

### 2.2.3. Requisitos de controlo

1. O sistema deverá permitir aos clientes consultar o catálogo de produtos para venda, com possibilidade de filtrar via qualquer combinação entre marca, modelo, cor, tipo de produto, classificação, popularidade, data de lançamento e preço atual;
2. O sistema deverá permitir aos funcionários alterar o preço atual de um dado produto;
3. O sistema deverá permitir aos funcionários consultar e registar vendas, devoluções, produtos e clientes;

## 2.3. Análise geral dos requisitos

A partir das entrevistas e conversas realizadas percebemos que a principal preocupação da “Um Belo PC” é que o sistema seja versátil na consulta do catálogo de produtos por forma a causar o mínimo de dificuldade aos clientes nas consultas aos produtos. Pretendem também tirar proveito da base de dados como forma de melhorar o negócio e a efetividade das campanhas publicitárias, transformando as heurísticas em estratégias fundamentadas. Querem saber coisas como quais as faixas etárias mais frequentes, quais são os modelos mais procurados e quanto foi faturado. Com esse propósito será necessário processar os dados existentes na base de dados.

O nosso cliente quer também manter o controlo sobre os dados, tal como tinha em papel. Isto significa diferenciação no acesso a certas informações entre clientes e funcionários.

# 3. Modelação Conceptual

## 3.1. Abordagem à modelação

Inicialmente construímos um modelo a partir dos requisitos definidos para este trabalho, mas conforme a matéria da UC estava a ser lecionada, foram necessárias adições e alterações aos requisitos que mudaram o consequente modelo conceptual.

Como exemplo podemos mencionar as cardinalidades entre as entidades do modelo, pois quando foram lecionadas as aulas sobre chaves estrangeiras e valores nulos percebemos que o nosso stand teria relações com cardinalidades possivelmente nulas. Isto pode ser visto no sexto requisito de descrição apresentado, em que colocamos *“Um carro pode estar associado a um conjunto de alugueres”,* indicando que a cardinalidade poderia tomar o valor zero.

Chegámos também a ter dois modelos conceptuais distintos na forma, mas que geravam um modelo lógico semelhante. Nessa situação fomos verificar os nossos modelos junto dos docentes para despistar erros e saber qual dos dois estava correto.

De resto a nossa abordagem foi idêntica à feita nas aulas práticas, determinar entidades, as relações entre elas e respetivas cardinalidades, representar tudo isso com um modelo Chen (Collony, T. e Begg, C., 2015, pp509) acabando por passar o modelo de papel para o brModelo.

## 3.2. Entidades do sistema

|  |  |  |
| --- | --- | --- |
| Entidade | Descrição | Ocorrência |
| Customer | Toda a informação pessoal relativa a um cliente da loja | Cada cliente está associado a pelo menos uma venda |
| Product | Toda a informação estática relativa a um produto | Cada produto pode estar associado no máximo a uma venda e uma devolução |
| Return | Todos os dados pessoais de um cliente da loja. | Cada cliente teve que estar associado em pelo menos uma venda ou um aluguer em algum ano |
| Sale | Toda a informação relativa a uma venda | Cada venda ocorre sempre associada a um cliente e pelo menos um produto |

Tabela 1 - Entidades do modelo ER.

## 3.3. Relacionamentos do sistema

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entidade | Cardinalidade | Relacionamento | Cardinalidade | Entidade |
| Sale | 1..n | to | 1..1 | Customer |
| Return | 0..1 | done by | 1..1 | Customer |
| Return | 0..n | relative to | 1..1 | Sale |
| Return | 0..1 | item | 1..1 | Product |
| Sale | 0..1 | has | 1..n | Product |

Tabela 2 - Relacionamentos do modelo ER.

## 3.4. Atributos do sistema

|  |  |  |
| --- | --- | --- |
| Entidade | Atributo | Descrição |
| Customer | Id  First Name  Last Name  Email  Phone (1,n)  Address – Street  Address – City  Address – Number  Address – Postal Code  Date of Birth  NIB | Identificador único  Primeiro nome do cliente  Último nome do cliente  Endereço de e-mail do cliente  Números de telefone do cliente  Rua onde mora o cliente  Cidade onde mora o cliente  Número de porta do cliente  Código postal da residência do cliente  Data de nascimento do cliente  Número de identificação bancária usado para pagamento |
| Product | Bar Code  Product Type  Brand  Model  MSRP  Price  Release Date  Color  Average Rating  Description | Código de barras do produto  Tipo do produto  Marca/fabricante do produto  Modelo do produto  Preço de retalho recomendado pelo fabricante  Preço do produto  Data em que o produto foi anunciado  Cor do produto  Média dos ratings dos produtos  Descrição do produto |
| Return | Id  Date | Identificador único de devolução  Data da devolução |
| Sale | Id  Total  Date | Identificador único da venda  Montante cobrado ao cliente  Data em que a venda foi feita |

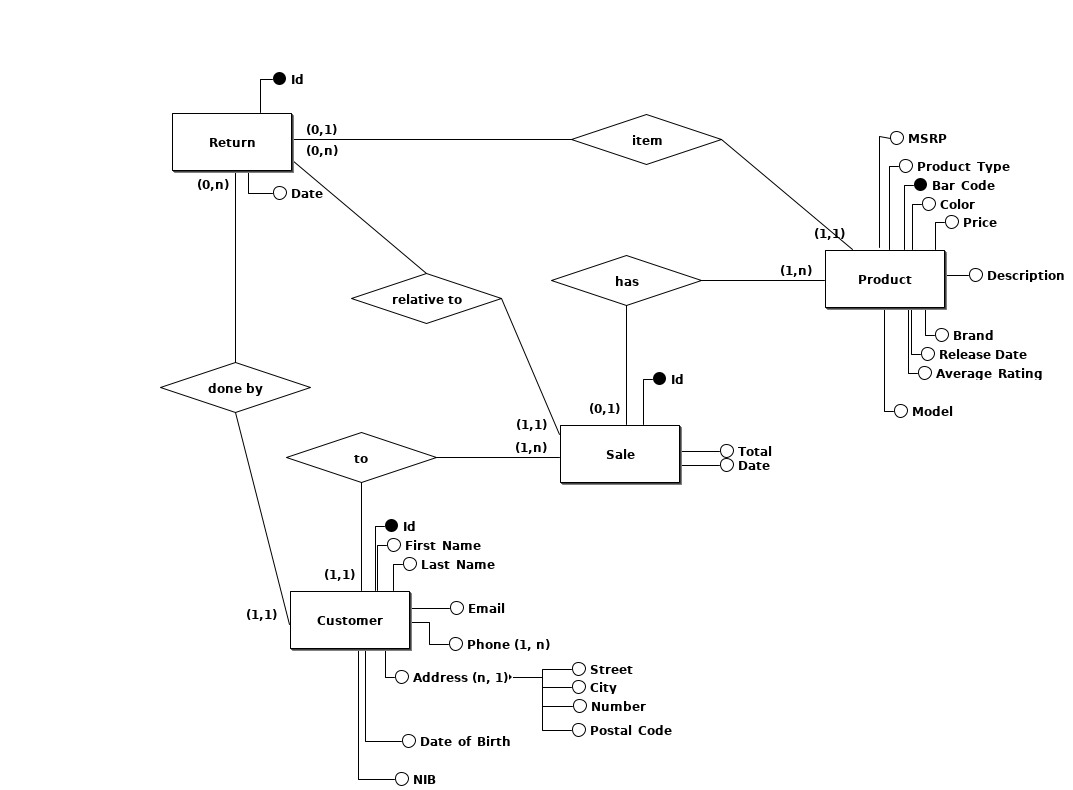
Tabela 3 - Atributos do modelo ER.

## 3.5. Detalhe e generalização de entidades

No nosso modelo não foi necessário redefinir as entidades iniciais, porque todas as entidades do modelo são atómicas e não redundantes. O modelo não contém entidades que sejam especificações umas das outras, como seria o caso se existisse um ‘VIP Customer’, que seria uma especificação da entidade Customer. À primeira vista as entidades ‘Sale’ e ‘Return’ parecem ser possíveis de generalizar para uma nova entidade ‘Transaction’. Porém, ‘Sale’ está associada a um conjunto de produtos, enquanto que ‘Return’ só se pode referir a um (há uma ‘Return’ por cada produto devolvido). Por causa de diferenças de cardinalidade como esta, não faz sentido generalizar. Todas as restantes entidades são claramente distintas.

## 3.6. Diagrama ER

## 3.7. Validação do modelo com o cliente

  
Figura 1: Diagrama ER

Estando já definido um primeiro modelo conceptual fomos apresentar a nossa proposta à loja. Após umas explicações sobre a notação do diagrama, foi-nos confirmado que o modelo corresponde ao que pretendiam. Assim sendo este modelo conceptual passou a ser usado como base para desenvolver o modelo lógico.

# 4. Modelação Lógica

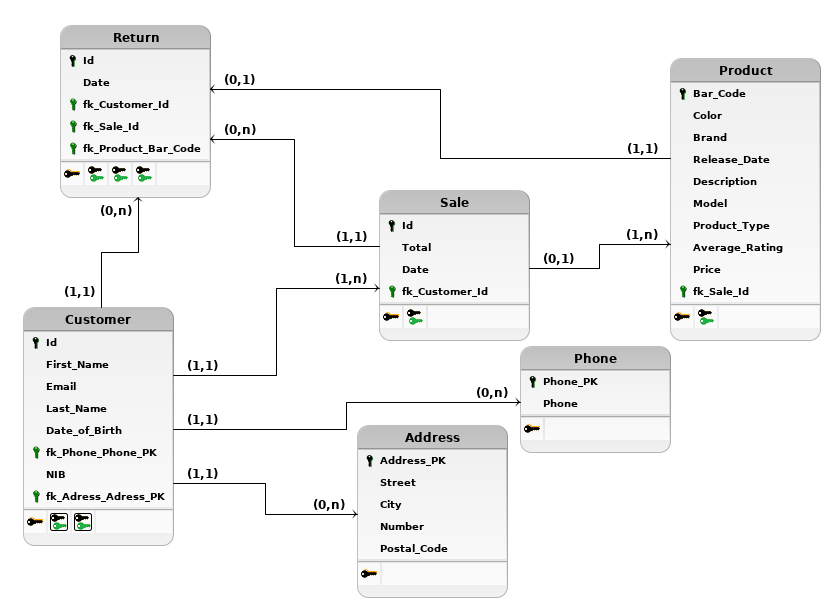
## 4.1. Construção e validação do modelo

Agora que o modelo conceptual está definido e aprovado, temos que passá-lo para uma construção lógica. Como o modelo conceptual foi inserido no programa brModelo, faz sentido utilizar a funcionalidade disponível e gerar o respetivo modelo lógico. Estando o modelo conceptual bem feito sabemos que o resultado será correto (a menos de erros no programa). Apesar de não termos encontrado qualquer tipo de documentação, este programa foi o indicado pelos docentes, dando-nos confiança no resultado.

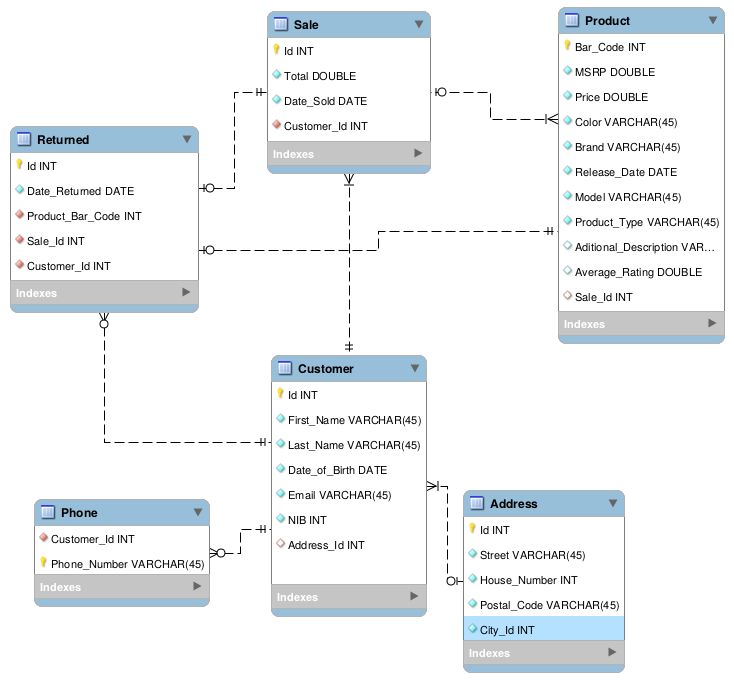
Com isto em mente, invocamos o “*Gerar Esquema Lógico*” no programa e fomos definir o modelo lógico de forma analítica (Codd, 1969) tendo o modelo automático como um primeiro esquema.

## 4.2. Desenho do modelo lógico

O modelo gerado desde o conceptual contém erros que foram corrigidos manualmente, tais como a cardinalidade entre *Customer* e *Address,* e a posição da chave estrangeira entre *Phone* e *Customer.* Por uma questão de legibilidade decidimos alterar alguns detalhes ao modelo automático como o nome da chave primária na tabela *Address* de *Address\_PK* para *Id,* bem como os nomes da tabela *Return* para *Returned*, bem como pequenos ajustes nos nomes dos atributos, tal como indicado mais à frente no [ponto 5.2](#5.2. Tradução do esquema lógico para o SGBD|outline)*.* No final obtemos o seguinte modelo, já copiado para a ferramenta *MySQL Workbench*:

  
Figura 2: Modelo lógico gerado com o brModelo

## 4.3. Validação do modelo através da normalização

  
Figura 3: Modelo lógico após correções

Já com um primeiro esquema fomos refinar o resultado via normalização, decompondo as relações iniciais sem perdas e preservando as dependências do esquema.

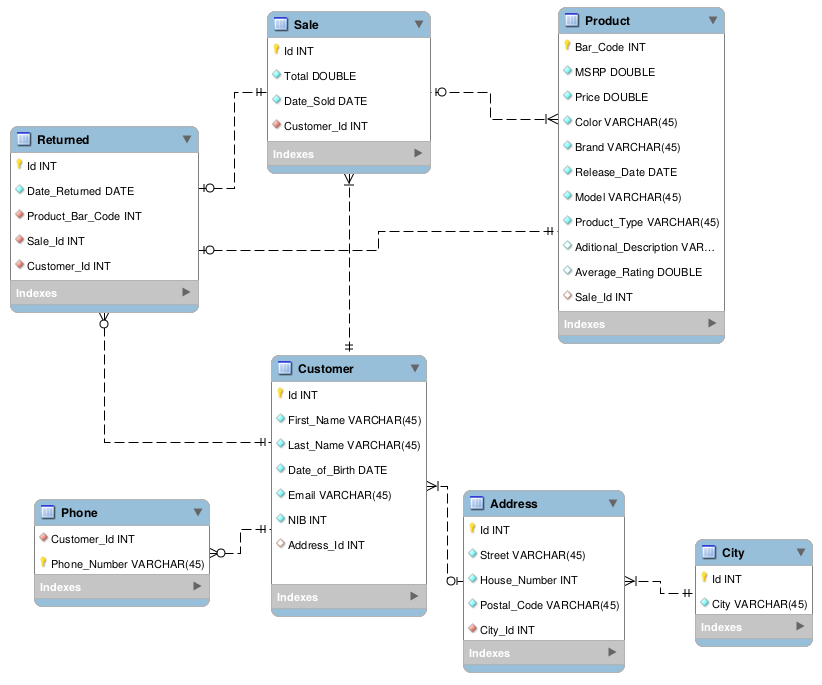
Com a aplicação da primeira fórmula normal (Codd, 1970) o nosso esquema lógico ganhou mais uma tabela ‘Phone’, relativa ao atributo multivalorado do cliente. A cardinalidade será 1..n, significando que cada cliente poderá ter várias entradas na tabela Telefone.

Por termos um atributo composto na entidade ‘Customer’ chamado ‘Address’, foi criada uma nova tabela para albergar os atributos *Street, City, Number* e *Postal Code,* tornando todos os domínios em domínios simples. Como o nosso modelo satisfaz a primeira fórmula normal, não vão ser feitas alterações.

A aplicação da segunda fórmula normal (Codd, 1971), que visa dependências parciais, não levou a quaisquer alterações porque o nosso modelo não contém nenhuma chave composta, nem relacionamentos n..n. O modelo está na segunda fórmula normal.

Pela aplicação da terceira fórmula normal (Codd, 1971) vão ser originadas uma série de tabelas novas por efeito das dependências transitivas no nosso modelo. O A tabela *Address* contém um atributo *City* que pode ser determinado a partir do atributo *Postal Code*. Desta forma, vamos ter mais uma tabelas relativa ao atributo *City. A tabela Address* continuará com os restantes atributos.

Fazendo estas alterações temos então o esquema lógico na terceira fórmula normal:

  
Figura 4: Modelo na terceira fórmula normal de Codd

## 4.4. Validação do modelo com interrogações

Quando contactado, o funcionário sugeriu algumas interrogações que ele quer que o sistema disponibilize, entre elas: determinar toda as vendas feitas a um determinado cliente, saber todos os produtos envolvidos numa determinada venda e, para quando há problemas com as devoluções, saber a informação de contacto (nome e telefone) do cliente que a fez.

Estas três interrogações podem ser feitas no nosso sistema das seguintes formas:

1. πSale\_Id, Date (σ Customer\_Id = '1' (Customer⋈Sale))
2. σ Sale\_Id= ‘1‘ (Product)
3. πNumber, Fiirst\_Name, Last\_Name(σ Phone\_Customer\_Id=Customer\_Id(Phone⋈(σ Return\_Id=’1’ (Cusotmer⋈Return)))

## 4.5. Validação do modelo com transações

Por forma a garantir manutenção de consistência da base de dados, é necessário garantir a possibilidade de fazer algumas transações, entre as quais se destacam: registar um produto, registar um cliente, registar uma venda e registar uma devolução. O nosso sistema consegue responder das seguintes maneiras:

1. Returned ← Returned - (θ Returned\_Id = input\_Returned\_Id Λ Returned\_Product\_Bar\_Code = input\_Product\_Bar\_Code (Returned))

Returned ←Returned ∪ {(input\_Id, input\_Date\_Returned, input\_Product\_Bar\_Code, input\_Sale\_Id, input\_Customer\_Id)}

1. Product ← Product – (θ Product\_Bar\_Code= input\_Bar\_Code (Product))

Product ← Product ∪{(input\_Bar\_Code, input\_MSRP, input\_Price, input\_Color, input\_Brand, input\_Release\_Date, input\_Model, input\_Product\_Type, null)}

1. Customer ← Customer – (θ Customer\_Id= input\_Id (Customer))

Customer ← Customer ∪{(input\_Id, input\_First\_Name, input\_Email, input\_Last\_Name, input\_Date\_of\_Birth, input\_NIB, null)}

1. Sale ← Sale – (θ Sale\_Id=input\_Id (Sale))

Sale ← Sale ∪{(input\_Id, input\_Total, input\_Date\_Sold, input\_Customer\_Id)}

Os nossos computadores não suportam o símbolo tradicional G caligráfico, tendo usado uma alternativa comum ϒ segundo o indicado em (3).

## 4.6. Revisão do modelo lógico com o cliente

Neste ponto do trabalho, quando o modelo lógico ficou pronto, fomos apresentá-lo ao funcionário para saber se estava satisfeito com a nossa proposta. Como as interrogações simples e transações que ele referiu são válidas no modelo, o modelo foi aceite.

# 5. Implementação Física

## 5.1 Seleção do sistema de gestão de bases de dados

Durante a nossa procura por um sistema de gestão os principais critérios de seleção foram: possibilidade de programação de baixo nível, existência de uma boa interface gráfica disponível, custo de implementação (licenças) e, sobretudo, o sistema tem de ser relacional. O nosso cliente é muito paranoico com bases de dados não-relacionais, e ficou mais que feliz por ter um sistema relacional para o seu negócio.

Para além da base de dados, estão também encomendados outros softwares, entre as quais uma aplicação JAVA para smartphones, que permita consultar o inventário disponível e comprar produtos, pelo que convém escolher um SGBD que seja comum.

A melhor opção para os nossos critérios, e a que escolhemos, foi o MySQL, pelo que toda a implementação física foi feita a pensar no seu suporte com a linguagem SQL.

## 5.2. Tradução do esquema lógico para o SGBD

Devido às limitações de sintaxe da linguagem SQL, não nos foi possível manter todos os nomes dados no modelo conceptual e no modelo lógico. Como toda a base de dados foi modelada na língua Inglesa, não há problemas com acentuação. Porém, alguns nomes coincidem com palavras-chave de SQL, pelo que fomos obrigados a fazer alterações:

* Return → Returned
* Address.Number → Address.House\_Number
* Address.Address\_id → Address.Id
* Return.Date → Returned.Date\_Returned
* Sale.Date → Sale.Date\_Sold

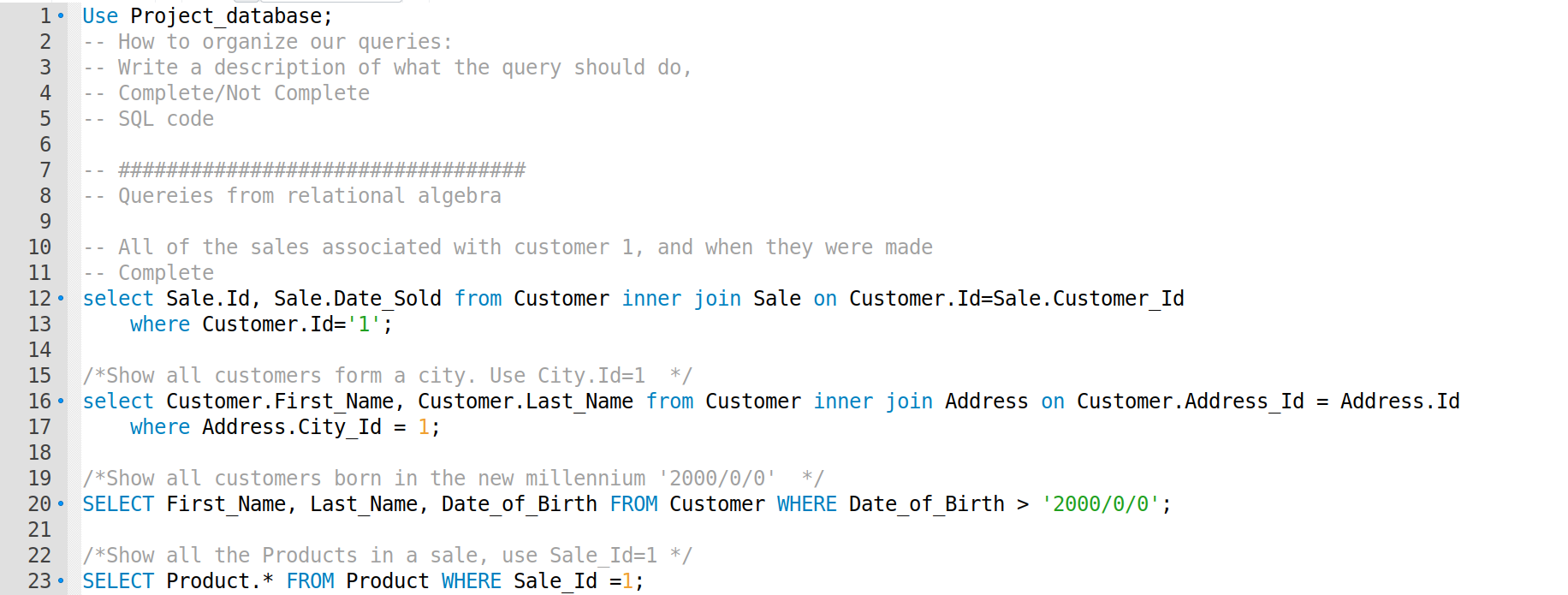
Estas alterações já foram apresentadas em figuras anteriores. No final de todas as correções, temos o seguinte modelo lógico final, a partir do qual será gerado o físico:

## 

Figura 5: Modelo lógico final, usado para gerar o físico no MySQL Workbench

## 5.3.Tradução das interrogações para SQL

Para além das interrogações mencionadas, foram desenvolvidas outras para o nosso cliente:

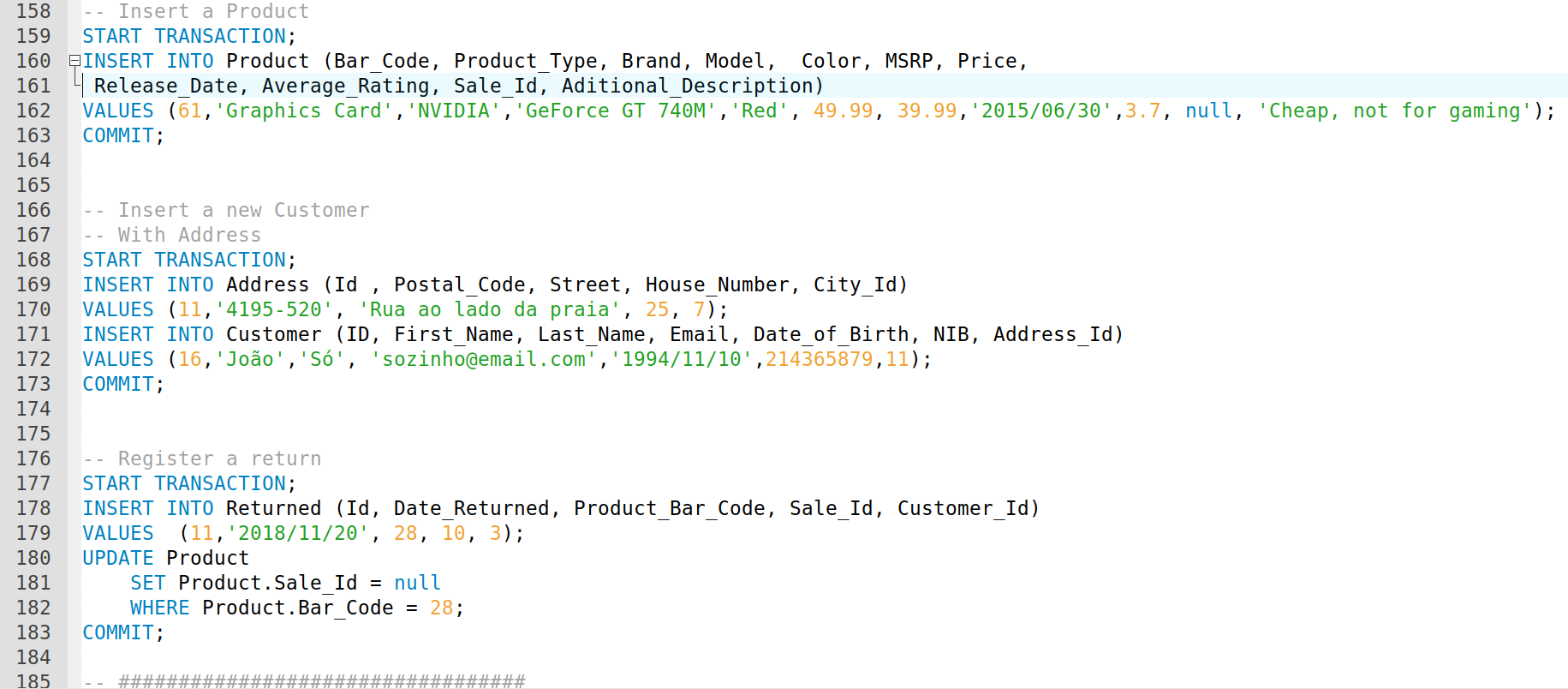
  
Figura 6: Exemplo de queries implementadas

## 5.4. Tradução de transações em SQL

Segundo o que o Chico nos disse, esta base de dados será ligada a uma aplicação em JAVA, sendo para isso preciso definir transações no nosso sistema.

As transações mais evidentes são os registos de contratos de venda e alugueres, as quais estão exemplificadas para os casos em que o cliente é novo no stand.

  
Figura 7: Exemplos de transações implementadas



## 5.5. Escolha, definição e caracterização de índices em SQL

Como a base de dados que estamos a desenvolver é muito pequena, o proveito tirado pelo uso de índices seria irrelevante, porque mesmo com alguns anos de uso, o facto de a base de dados ser limpa anualmente faz com que não se tire proveito dos índices.

## 5.6. Estimativa do espaço ocupado em disco e taxa de crescimento anual

Para estimar o espaço em disco, vamos fazer uma análise de pior caso, ou seja, assumir que todos os contratos que o stand assina são relativos a clientes novos, e que todos os atributos de comprimento variável são completamente preenchidos. Para não tomarmos valores infinitos vamos assumir apenas um número de telefone por cliente e que um carro só pode ser alugado uma vez. Neste caso, sabemos que um cliente novo, ao contar com a morada, precisa de: 12 VARCHAR(45), 6 INT e 1 DATE.

Ao registar uma venda, serão ocupados 2 VARCHAR(45), 3 INT e 1 DATE. Um aluguer precisa de 2 VARCHAR(45), 3 INT e 2 DATE. Finalmente, um carro será representado com 5 VARCHAR(45), 5 INT e 1 DATE.

Não nos foi possível encontrar documentação sobre o tamanho de DATE, por isso vamos assumir que segue o mesmo formato que o SGBD *Oracle*, em que uma data são 7 bytes (DD-MON-YY). Considerando que um caractere ocupa 1 byte, um inteiro 8 bytes (64 bits) e uma data 7 bytes, conseguimos calcular os seguintes valores:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ENTIDADE | VARCHAR(45) | INT | DATE | TOTAL BYTES |
| Aluguer | 2 | 3 | 2 | 128 |
| Carro | 5 | 5 | 1 | 272 |
| Cliente | 12 | 6 | 1 | 595 |
| Venda | 2 | 3 | 1 | 121 |
| TOTAL | 21\*45 | 17\*8 | 5\*7 | 1116 |

Tabela 4 - Tamanhos das entidades em bytes

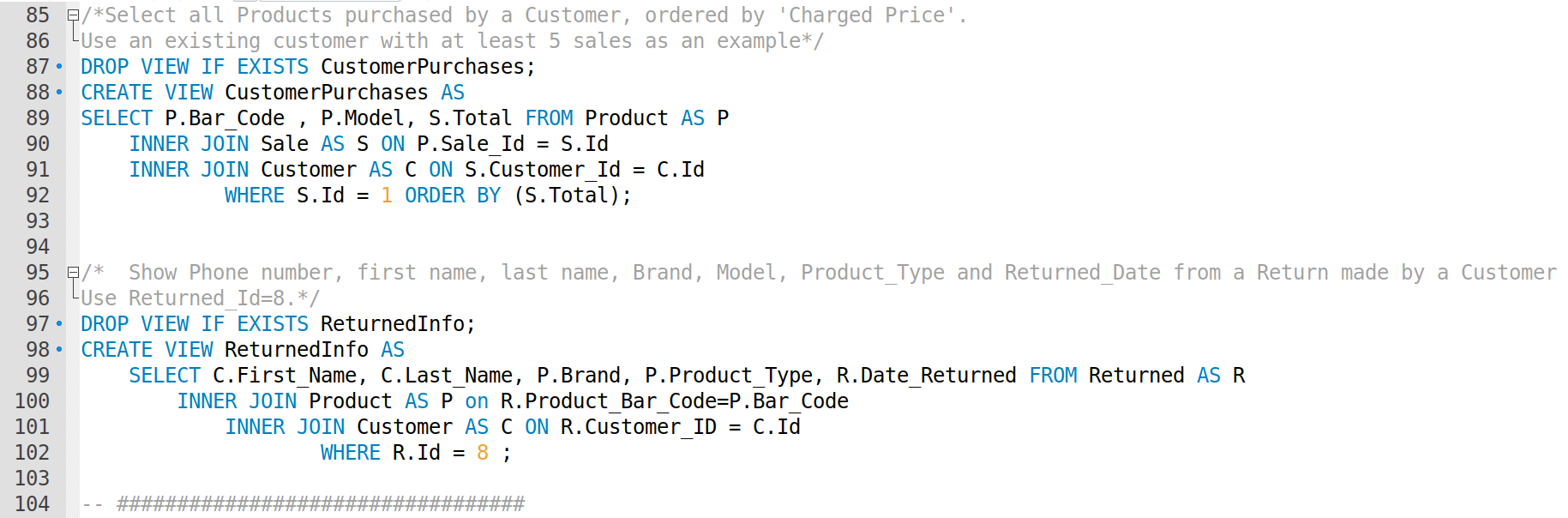
Para um cenário em que o stand recebe 12 clientes ao ano, com 6 alugueres e 6 vendas, mais 6 novos carros para inventário do stand onde já tem um stock de 10 carros, no final do ano teremos 7140 bytes para clientes, 768 bytes para alugueres, 727 bytes para venda e 4352 bytes para carros. Isto significa que, mesmo antes de limpar a base de dados no início do segundo ano, o espaço ocupado será de 12987 bytes (aprox 13 kilobytes).

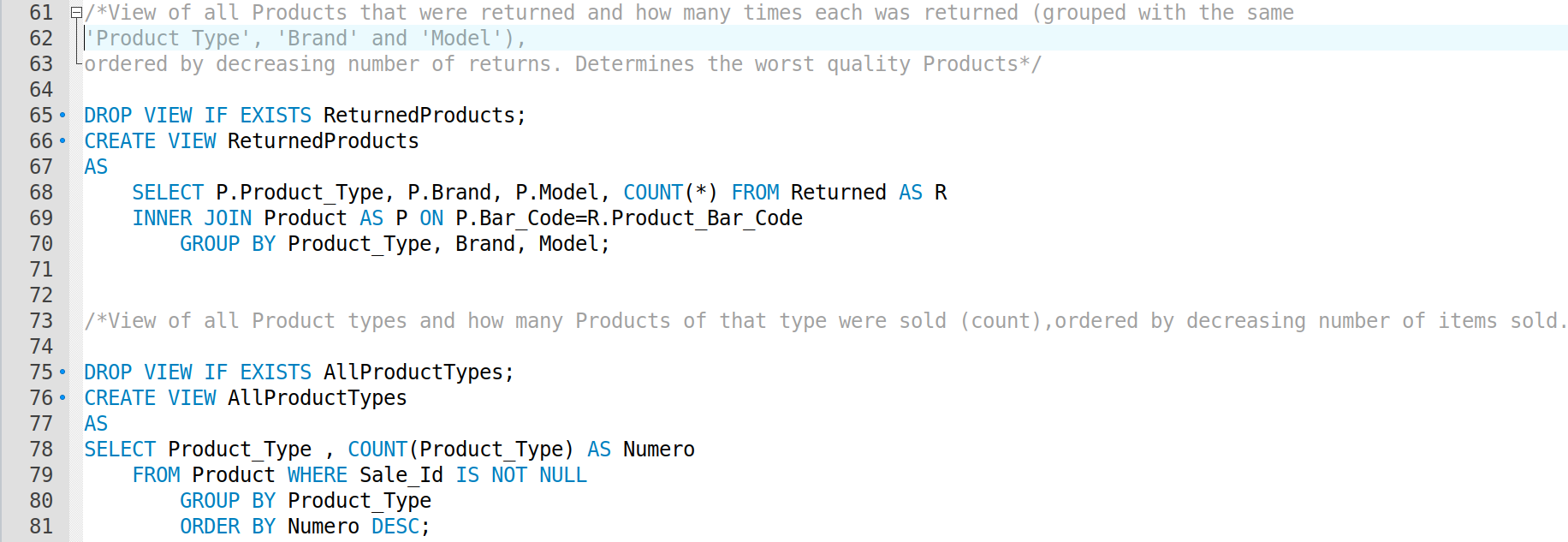
Imaginando que o stand tem um crescimento anual de 15% no volume de negócios de forma uniforme (15% mais carros, clientes, alugueres e vendas), a base de dados terá um crescimento de 2 kilobytes no segundo ano. Ao fim de uma década o sistema ocupará cerca de 53kilobytes.

Isto são valores muito pequenos para os computadores modernos, mesmo sabendo que existem cabeçalhosnos dados que estamos a ignorar como apontadores em VARCHAR(). Supondo que o Chico coloca um computador moderno dedicado à base de dados será mais provável o negócio ir à falência que a base de dados não caber em disco.

## 5.7. Definição e caracterização de vistas de utilização

Quanto a vistas, fomos implementar vistas de forma a tirar proveito delas, querendo isto dizer que as vistas foram criadas de forma a poupar trabalho a definir queries. Quando várias queries faziam um mesmo conjunto de operações, nós colocamos isso numa vista. Depois só tivemos que referenciar a vista em questão.



  
Figura 8: Exemplo de vistas implementadas

## 5.8. Mecanismos de segurança

A principal maneira de segurar um sistema é com a diferenciação entre utilizadores, em que cada um deve ter permissões para fazer o necessário e nada mais.

Ao analisar os requisitos definidos inicialmente, identifica-se facilmente dois tipos de utilizadores: funcionários e clientes.

Os funcionários deverão ter acesso a todas as tabelas na base de dados, podendo assim registar, atualizar e apagar Alugueres, Carros, Clientes e Vendas. As permissões de criação são necessárias para inserir carros novos em stock, registar contratos (e finais de Alugueres) e Clientes novos d’O Stand do Chico.

As permissões para remoção serão utilizadas no final de cada ano para limpar dados já não relevantes, como carros que já não estão em inventário e contratos do ano anterior.

Mais restrito estarão os clientes, que terão apenas acesso para consultar o inventário do stand. Não pode ser dado acesso de consulta a nenhuma outra tabela porque isso colocaria problemas de privacidade (se pudesse aceder à Cliente poderia ver a dos de outros clientes).

Todas as outras permissões não fazem sentido serem concedidas aos clientes.

## 5.9. Revisão do sistema implementado com o cliente

  
Figura 9: Permissões e utilizadores da base de dados

Quando concluímos o sistema fomos confirmar todas as nossas opções com o cliente antes de dar como concluído. Como o cliente é fictício, vamos nós fazer uma retrospetiva ao nosso trabalho.

É de notar que as estimativas de ocupação em disco não são confiáveis, as permissões dos utilizadores não estão refinadas (faria sentido, por exemplo, um cliente poder mudar os seus dados pessoais), e nem todas as interrogações e transações possivelmente relevantes para o negócio estão implementadas.

De qualquer forma, este sistema, apesar de restrito, seria possível de colocar em uso num negócio real, desde que acrescentado com interrogações relevantes e ajustadas as permissões. Como, normalmente, os utilizadores de um stand não acedem a uma base de dados via terminal, será importante definir transações adicionais para disponibilizar como API.

# 6. Conclusões e Trabalho Futuro

Desde processo de levantamento de ideias até ao estado atual de desenvolvimento do projeto tomamos decisões que tiveram mais tarde de ser revertidas. Isto aconteceu porque conforme fomos desenvolvendo e implementando a base de dados identificamos alguns problemas, duvidas ou problemas de eficiência que não levaram a adaptar para conseguirmos um modelo mais consistente.

A coordenação do grupo de trabalho foi razoável e após a primeira reunião de desenvolvimento o trabalho as coisas começaram a organizar-se. Optamos por uma divisão do trabalho por tópicos e pela utilização de um repositório online para a sincronização do trabalho dos elementos do grupo. O trabalho teve que ser organizado e distribuído um pouco em cima da hora, com todos os elementos a trabalhar em coisas diferentes (uns em povoamentos, outros em queries, etc.) de modo a conseguirmos cumprir a meta de entrega.

Deparamo-nos também problemas com o *forward engineering do brModelo* porque e uma pequena confusão de código com um engano da nossa parte que aconteceu pela adaptação de nomes de tabelas e atributos para não coincidirem com conceitos do MYSQLWorkbench. Isto foi resolvido quando voltamos a debruçar-nos sobre a implementação das queries e procuramos todas as incoerências e as corrigimos.

Quanto ao uso futuro deste trabalho, esta base de dados ter uso em termos de teste e ideias para algum projeto futuro, caso algum elemento do grupo decida criar uma loja online do tipo retratado no trabalho onde pode pegar na sua estrutura e com alguns ajustes desenvolver uma base de dados para um negócio de grades dimensões.

Referências

1 Connolly, T., Begg, C., 2015. Database Systems: A Practical Approach to Design, Implementation, and Management. 6th Ed. Essex: Pearson Education Limited.

2 Gouveia, F., 2014, Fundamentos de Bases de Dados, 1ª Edição, FCA.

3 ?: disponível em Center for Bioinformatics and Computational Biology, University of Maryland, através de <http://www.cbcb.umd.edu/confcour/Spring2014/CMSC424/Relational_algebra.pdf> [Acedido em 25 de Novembro de 2017]

Lista de Siglas e Acrónimos

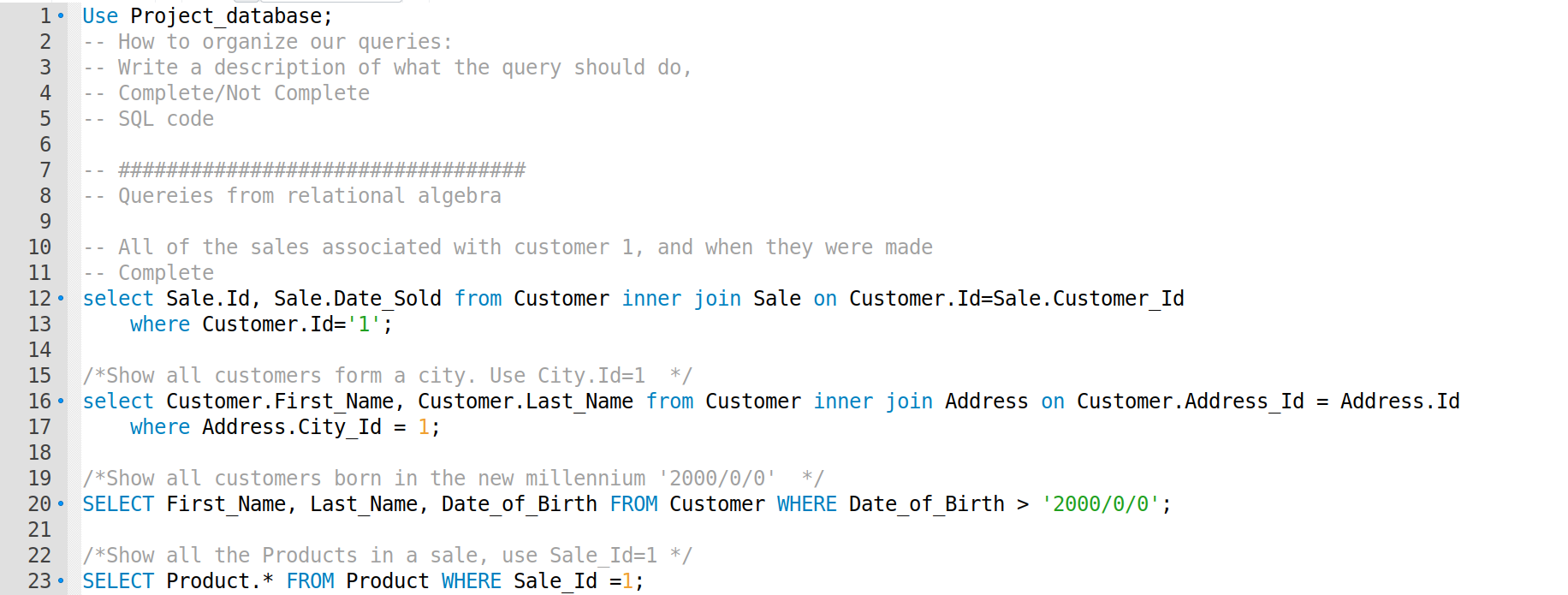
SGBD Sistema de Gestão de Bases de Dados

BD Base de Dados

UC Unidade Curricular

# Anexos

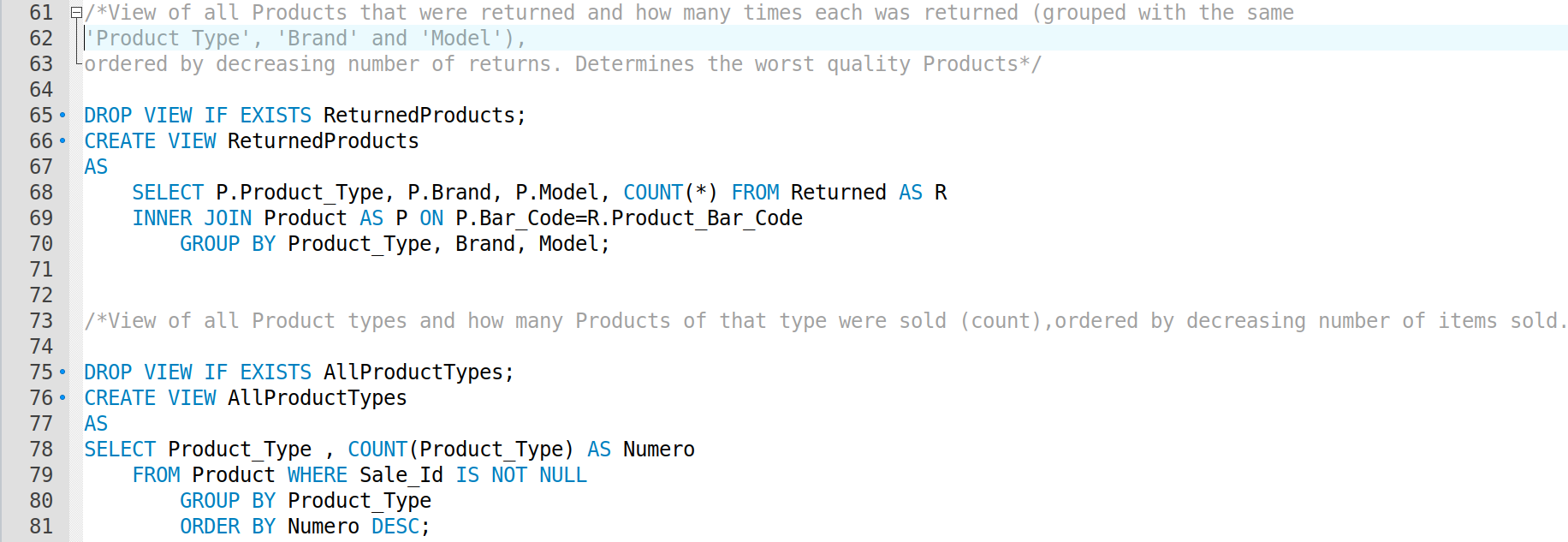
1. Parte 1 de 8 dos scripts



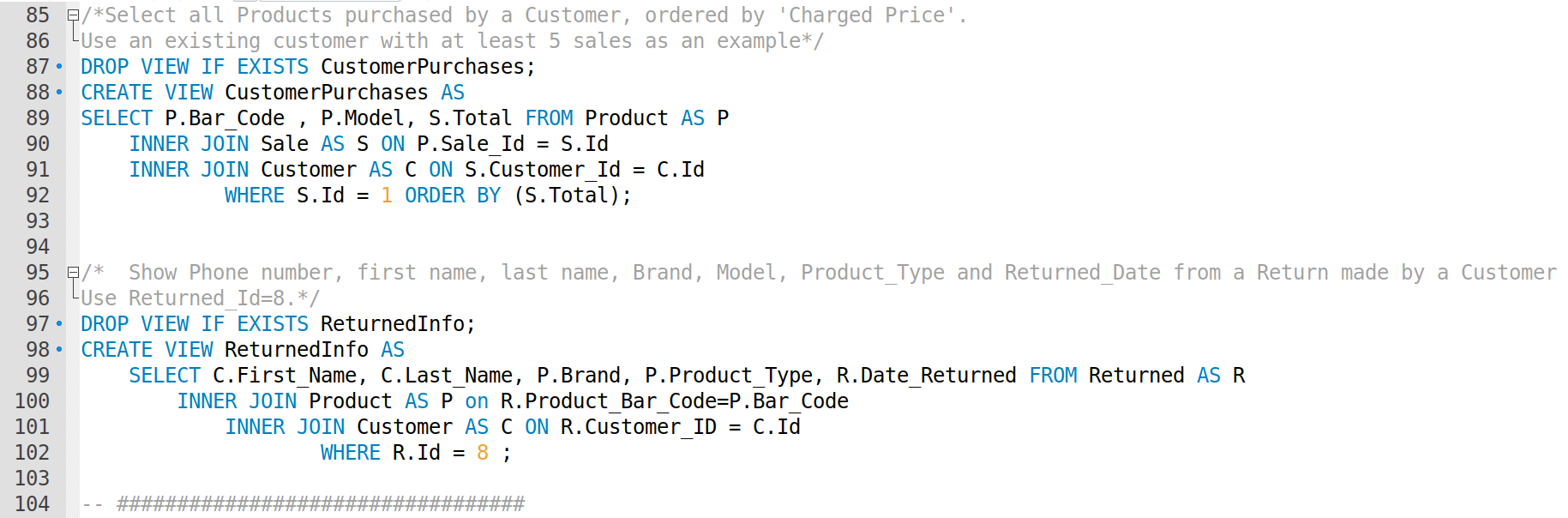
1. Parte 2 de 8 dos scripts



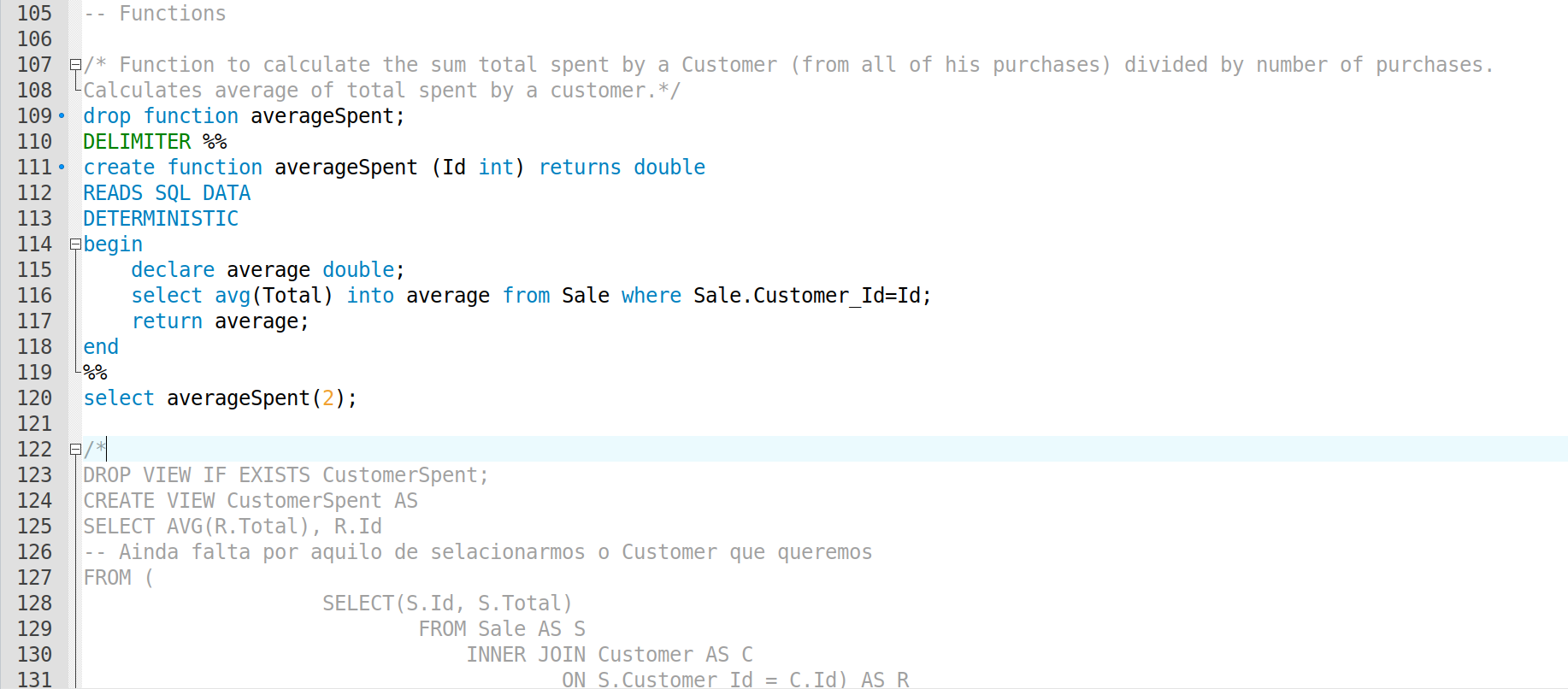
1. Parte 3 de 8 dos scripts



1. Parte 4 de 8 dos scripts



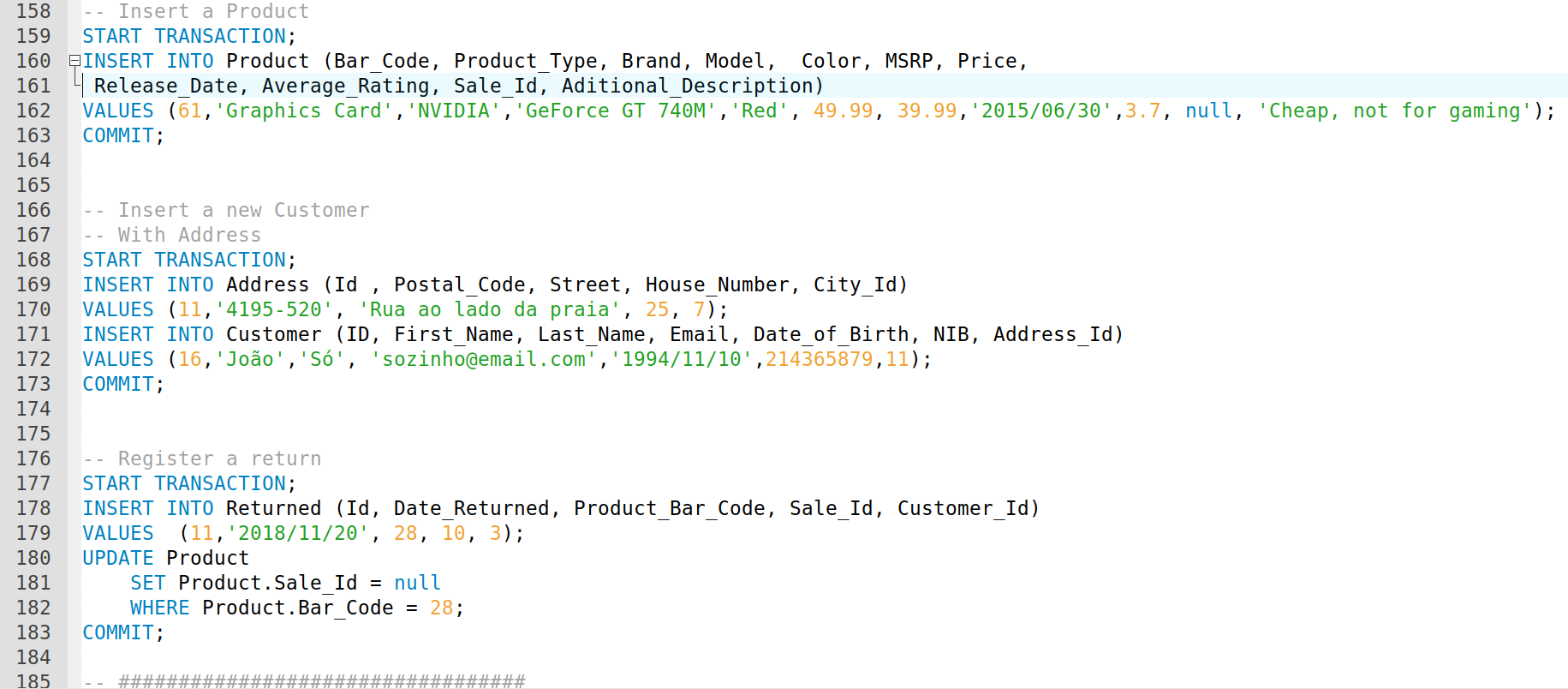
1. Parte 5 de 8 dos scripts



1. Parte 6 de 8 dos scripts



1. Parte 7 de 8 dos scripts



1. Parte 8 de 8 dos scripts

