

**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.

**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 5](#__RefHeading___Toc535645391)

[3. Modelação Conceptual 6](#__RefHeading___Toc535645395)

[3.1. Abordagem à modelação 6](#__RefHeading___Toc1661_1061193116)

[3.2. Entidades do sistema 6](#__RefHeading___Toc1663_1061193116)

[3.3. Relacionamentos do sistema 7](#__RefHeading___Toc1665_1061193116)

[3.4. Atributos do sistema 7](#__RefHeading___Toc1667_1061193116)

[3.5. Detalhe e generalização de entidades 8](#__RefHeading___Toc1669_1061193116)

[3.6. Diagrama ER 8](#__RefHeading___Toc1671_1061193116)

[3.7. Validação do modelo com o cliente 8](#__RefHeading___Toc76_512549187)

[4. Modelação Lógica 9](#__RefHeading___Toc78_512549187)

[4.1. Construção e validação do modelo 9](#__RefHeading___Toc80_512549187)

[4.2. Desenho do modelo lógico 10](#__RefHeading___Toc82_512549187)

[4.3. Validação do modelo através da normalização 10](#__RefHeading___Toc84_512549187)

[4.4. Validação do modelo com interrogações 11](#__RefHeading___Toc86_512549187)

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

[4.6. Revisão do modelo lógico com o cliente 12](#__RefHeading___Toc90_512549187)

[5. Implementação Física 13](#__RefHeading___Toc936_512549187)

[5.1 Seleção do sistema de gestão de bases de dados 13](#__RefHeading___Toc938_512549187)

[5.2. Tradução do esquema lógico para o SGBD 13](#__RefHeading___Toc940_512549187)

[5.3.Tradução das interrogações para SQL 15](#__RefHeading___Toc942_512549187)

[5.4. Tradução de transações em SQL 15](#__RefHeading___Toc944_512549187)

[5.5. Escolha, definição e caracterização de índices em SQL 16](#__RefHeading___Toc946_512549187)

[5.6. Estimativa do espaço ocupado em disco e taxa de crescimento anual 16](#__RefHeading___Toc948_512549187)

[5.7. Definição e caracterização de vistas de utilização 17](#__RefHeading___Toc1192_512549187)

[5.8. Mecanismos de segurança 17](#__RefHeading___Toc950_512549187)

[5.9. Revisão do sistema implementado com o cliente 18](#__RefHeading___Toc1194_512549187)

[6. Conclusões e Trabalho Futuro 19](#__RefHeading___Toc1196_512549187)

[Anexos 22](#__RefHeading___Toc1675_1061193116)

# Índice de Figuras

Índice de Figuras

[Figura 1 - Diagrama ER 8](#FIGURA!0|sequence)

[Figura 2: Modelo lógico gerado automaticamente pelo brModelo 10](#FIGURA!1|sequence)

[Figura 3: Modelo inserido no MySQL Workbench 14](#FIGURA!2|sequence)

[Figura 4: Exemplo de queries implementadas 15](#FIGURA!3|sequence)

[Figura 5: Exemplos de transações implementadas 15](#FIGURA!4|sequence)

[Figura 6: Exemplo de vistas implementadas 17](#FIGURA!5|sequence)

[Figura 7: Permissões e utilizadores da base de dados 18](#FIGURA!6|sequence)

# Índice de Tabelas

[Tabela 1 - Entidades do modelo ER. 6](#TABELA!0|sequence)

[Tabela 2 - Relacionamentos do modelo ER. 7](#TABELA!1|sequence)

[Tabela 3 - Atributos do modelo ER. 8](#TABELA!2|sequence)

[Tabela 4 - Tamanhos das entidades em bytes 16](#TABELA!3|sequence)

# 1. Definição do Sistema

Para este projeto vamos desenvolver uma base de dados relacional com o gestor M*ySQL,* que permita sevir 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, tinhamos 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 compoentes. Fizémos 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 factores 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 Requesitos

## 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 tivémos 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, sestando 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 foramos 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 efectividade 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 facturado. 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 signifca 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 estva 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 respectivas 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)  Adress – Street  Adress – City  Adress – Number  Adress – 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 entidade 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

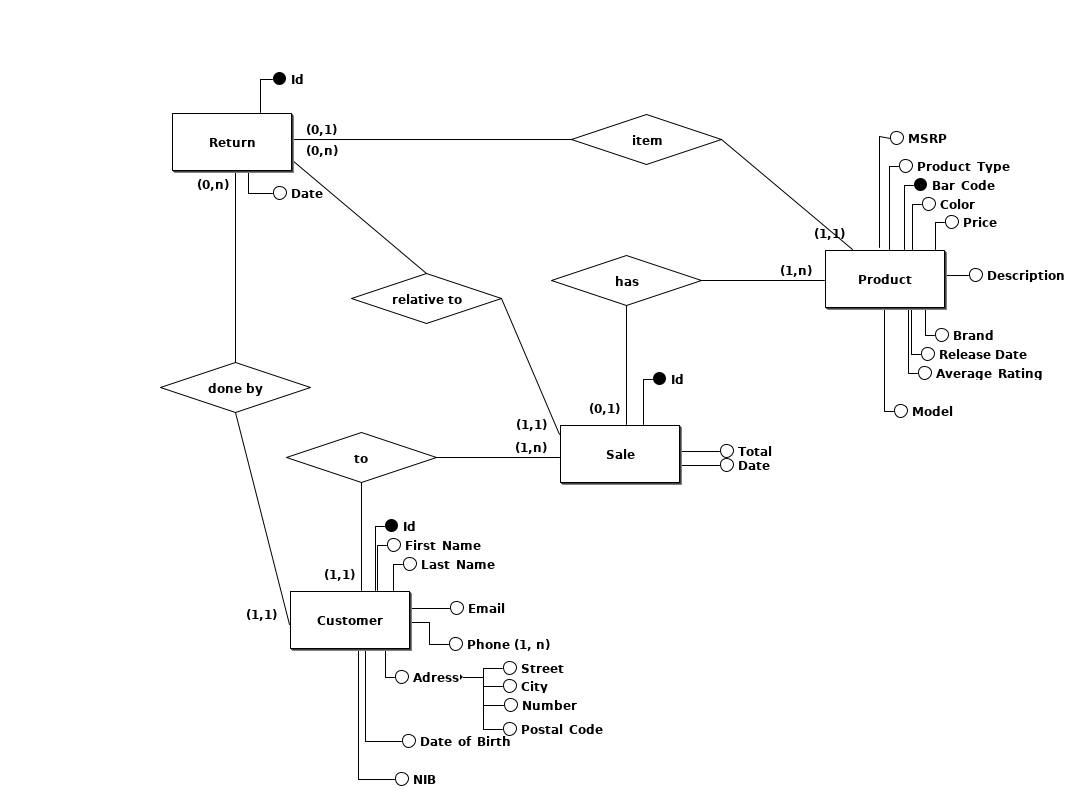


Figura 1 - Diagrama ER

## 3.7. Validação do modelo com o cliente

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

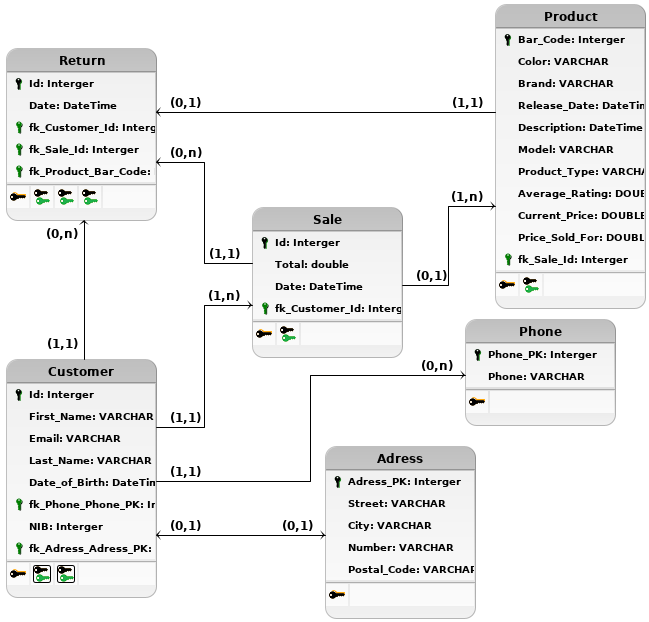
## 4.1. Construção e validação do modelo

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.

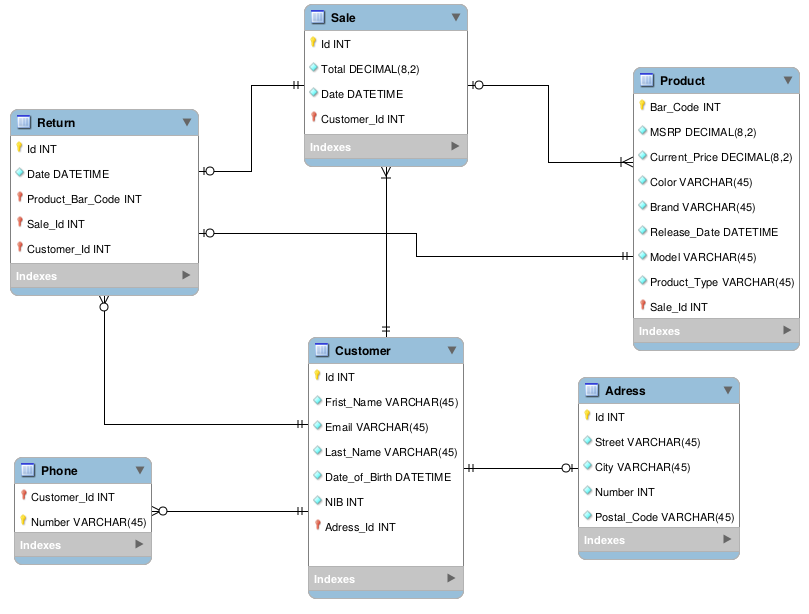
Com isto em mente, invocamos o “*Gerar Esquema Lógico*” no programa e fomos definir o modelo lógico de forma analítica (Codd, 1969) tendo o modelo automático como um primeiro esquema.

## 4.2. Desenho do modelo lógico

Por uma questão de legibilidade decidimos alterar alguns detalhes ao modelo automático como o nome da chave primária na tabela *Adress* de *Adress\_PK* para *Id* e a posição da chave estrangeira entre *Phone* e *Customer.* No final obtemos o seguinte modelo, já copiado para a ferramenta *MySQL Workbench*:

Figura 2: Modelo lógico gerado automaticamente pelo brModelo

## 4.3. Validação do modelo através da normalização

  
Figura 3: Modelo copiado para a ferramenta MySQL Workbench

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.

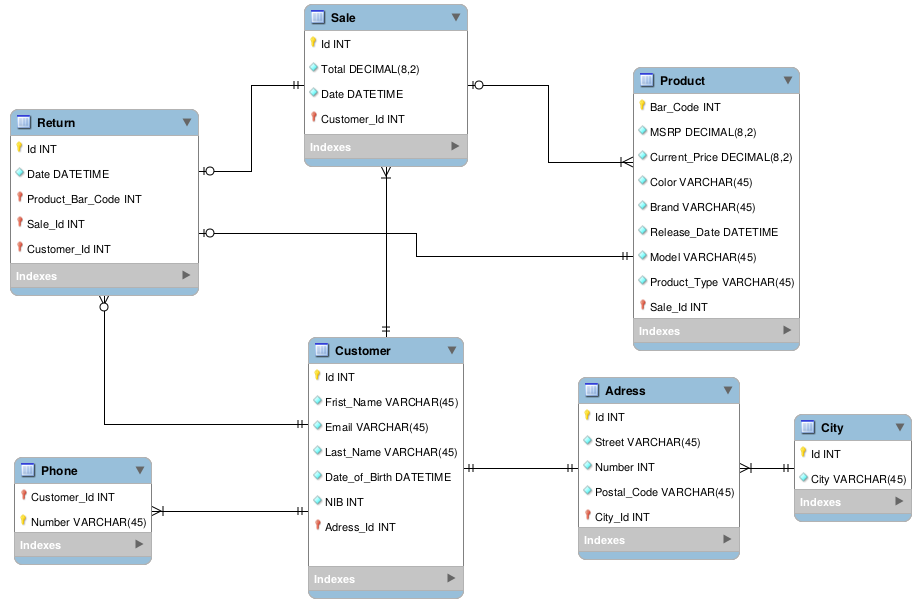
Por termos um atributo composto na entidade ‘Customer’ chamado ‘Adress’, foi criada uma nova tabela para albregar o atributos *Street, City, Number* e *Postal Code,* tornando todos os domínios em domínios simples. Como o nosso modelo satisfaz a primeira fórmula normal, não vão ser feitas alterações.

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 *Adress* 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 Adress* continuará com os restantes atributos.

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

## 4.4. Validação do modelo com interrogações

  
Figura 4: Modelo na terceira fórmula normal

Quando contactado, o Chico forneceu várias interrogações que ele quer que o sistema disponibilize, entre elas: todos os carros de um tipo de combustível (exemplo Gasolina), todos os carros de uma marca que tenham potência alta (exemplo Volkswagen com mais de 200 cavalos), e o nome dos clientes que compraram carros premium (acima de 30 000€). Também forneceu outras mais avançadas mas que só poderemos apresentar mais à frente por usarem parâmetros que não podem ser descritos via álgebra relacional (exemplo concatenação do nome próprio e apelido).

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

1. πSale\_Id, Date (σ Customer\_Id = '1' (Customer⋈Sale)) – Todas as vendas de um cliente
2. σ Sale\_Id= ‘1‘ (Product) – todos os produtos de uma venda
3. πNumber, Fiirst\_Name, Last\_Name(σ Phone\_Customer\_Id=Customer\_Id(Phone⋈(σ Return\_Id=’1’ (Cusotmer⋈Return)))– Nome e telefone do cliente que retornou um produto

## 4.5. Validação do modelo com transações

Também foram acordadas transações por serem mais seguras, entre as quais: registar uma venda e registar um aluguer. O nosso sistema consegue responder das seguintes maneiras:

1. ErroTransação ← ϒcount(θCarro\_NumeroChassis = inputNumeroChassis (Sale))

Venda ← Venda ∪ {(inputPreçoVenda, inputData, inputValorBase, inputNRIDCivil, inputNumeroChassis)}

1. Aluguer ← Aluguer - (θCarro\_NumeroChassis = nrChassis Λ Cliente\_NRIDCivil = nrIDCivil (Aluguer))

Aluguer ←Aluguer ∪ {(mensalidade, inicioAluguer, fimAluguer, quilometragemRealizada, nrIDCivil, nrChassis)}

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)}

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 Chico para saber se estava satisfeito com a nossa proposta. Como as interrogações simples e transações que ele referiu são válidas no modelo, ele aceitou-o.

# 5. Implementação Física

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

Ao procurar um gestor nós queríamos algo que fosse simples, permitisse uma programação de baixo nível, tivesse uma interface gráfica disponível, capaz de uma utilização simples e, sobretudo, que fosse relacional e não necessitasse de pagamento.

Para além disso, o Cliente tem encomendada uma aplicação JAVA para dispositivos moveis, que permita consultar o inventário de equipamentos, pedir as especificações dos produtos, satisfação dos clientes em relação a produtos adquiridos, consultar como preço recomendado e o preço em loja, efetuar retornos e/ou trocas de produtos já comprados, etc., tendo assim necessidade de uma base de dados que disponibilize uma interface.

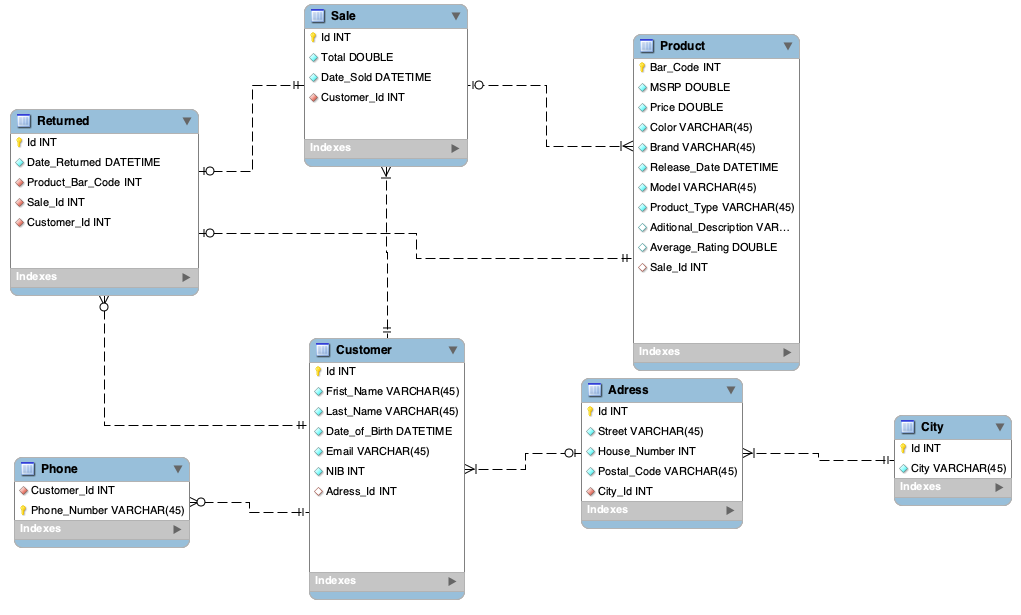
A melhor opção para os nossos critérios, e a que escolhemos, foi o MySQL, pelo que toda a implementação física foi feita a pensar no seu suporte com a linguagem SQL.

## 5.2. Tradução do esquema lógico para o SGBD

Devido às limitações de sintaxe impostas da linguagem SQL, não conseguimos manter todos os nomes dos atributos iguais aos do modelo lógico. Nomes que já predefinidos na linguagem SQL, que possuam acentuação ou espaçamento entre palavras não são suportados, obrigando-nos a converter para nomes semelhantes tais como:

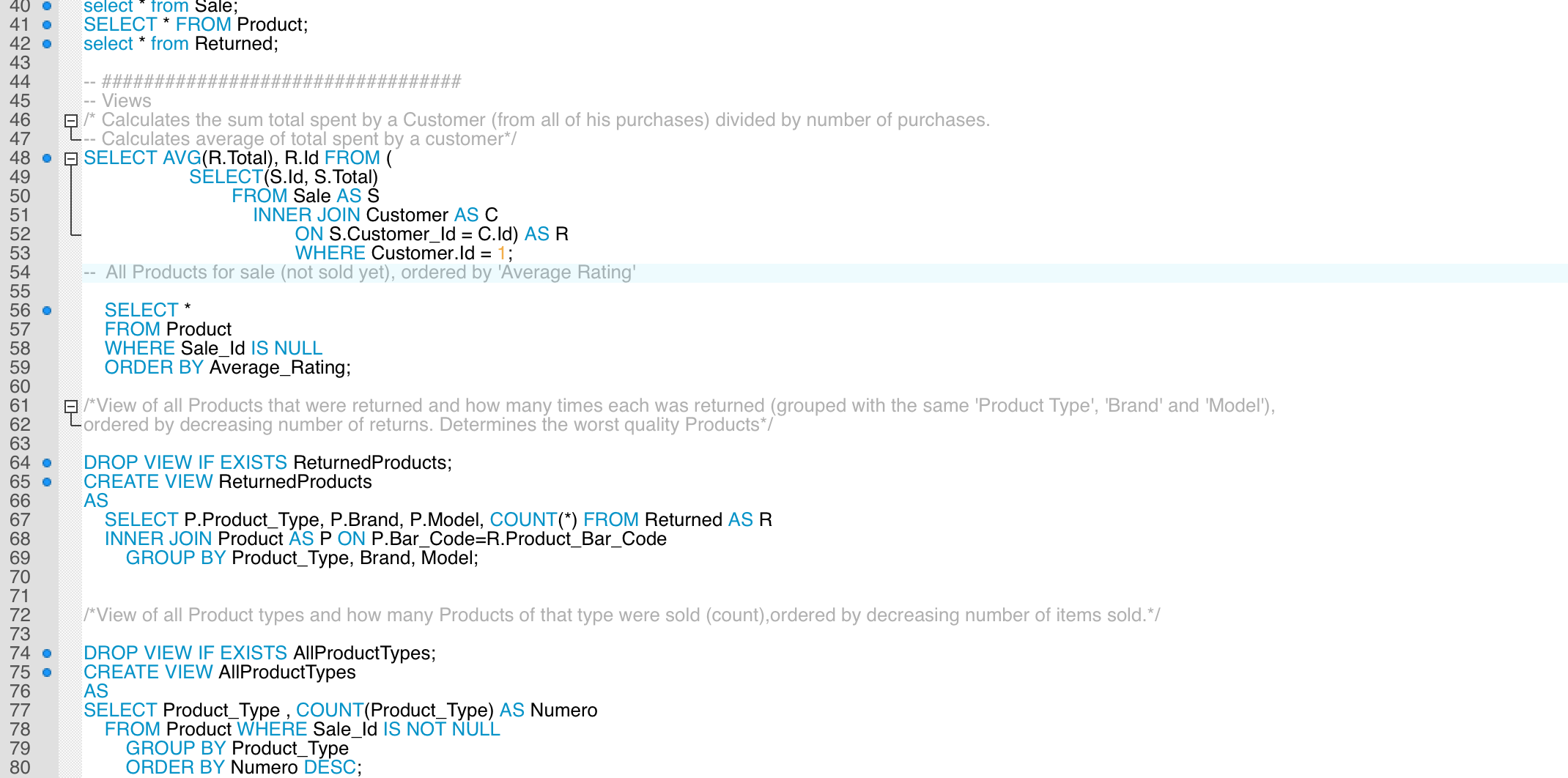
* Return → Returned
* Costumer → Customer
* Adress → Address
* Date of Birth → Date\_of\_Birth
* Bar Code→ Bar\_Code

Tendo feito estes ajustes, a implementação que se seguiu foi efetuada de forma simples e direta sendo que procuramos ao máximo das nossas capacidades manter os nomes possíveis, proceder a colocação e implementação correta das chaves e cardinalidades. Com isto feito fomos invocar o *“Forward Engineering”* do cliente *MySQL Workbench,* que nos gerou a base de dados *“*Um Belo PC*”* e a colocou num servidor local. O modelo lógico inserido no *MySQL Workbench* está na figura 4.

Figura 5: Modelo inserido no MySQL Workbench

## 5.3.Tradução das interrogações para SQL

Para além das definidas na secção 4.4, estão incluídas outras interrogações mais avançadas.

  
Figura 6: Exemplo de queries implementadas

## 5.4. Tradução de transações em SQL

Atendendo ao que o Cliente procura que este projeto de base de dados, seja integrada com uma aplicação em JAVA, tal que necessitamos definir transações para o nosso sistema e para tal focamo-nos nas mais importantes e comuns, isto é, os registos de Venda e Novo Cliente as quais estão na figura em baixo.

  
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 relativamente pequena, a utilização e o efeito pratico da implementação de índices seria mínima , uma vez que mesmo após anos de uso, o facto de a base de dados ser reacondicionada anualmente para libertar informação desnecessário (“ruido”), vai ajudar na manutenção e organização dos dados.

## 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 a loja assina são relativos a novos clientes que se querem inscrever no sistema da loja. Podemos também assumir que que todos os atributos de comprimento variável são completamente preenchidos. De modo a não tomarmos valores infinitos vamos assumir apenas um número de telefone por cliente eum mesmo portudo so pode ser devolvido uma única vez por cliente.

Nestas condições, a titulo de exemplo de pio casso , sabemos que um novo Customer, necessita de : 3 VARCHAR(45), 3 INT , 1 Date e 1 DOUBLE.

Ao registar uma Sale, seá necessário prencher 2 INT , 1 Date e 1 DOUBLE.

Um Return precisa de 4 INT , 1 Date.

Um Phone será 1 VARCHAR(45) e 1 INT.

Um Address será 2 VARCHAR(45) e 3 INT.

Uma City representarse-á por 1 VARCHAR(45) e 1 INT.

Finalmente, Um Product precisa de 1 VARCHAR(100), 5 VARCHAR(45), 1 INT e 1 DATE e 3 DOUBLE.

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(100) | VARCHAR(45) | INT | | DATE | DOUBLE | TOTAL BYTES |
| Product | 1 | 5 | | 1 | 1 | 3 | 360 |
| Sale | 0 | 0 | | 2 | 1 | 1 | 27 |
| Return | 0 | 0 | | 4 | 1 | 0 | 35 |
| Customer | 0 | 3 | | 3 | 1 | 0 | 162 |
| Phone | 0 | 1 | | 1 | 0 | 0 | 53 |
| Address | 0 | 2 | | 3 | 0 | 0 | 114 |
| City | 0 | 1 | | 1 | 0 | 0 | 53 |
| TOTAL | 1\*100 | 12\*45 | | 15\*8 | 4\*3 | 4\*3 | 814 |

Tabela 4 - Tamanhos das entidades em bytes

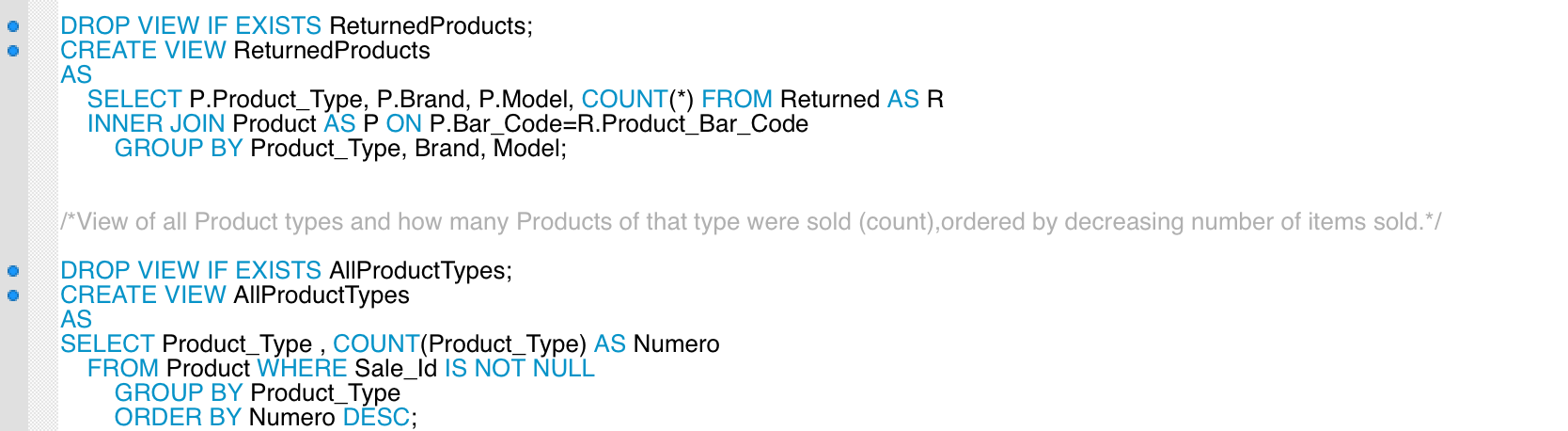
Para um cenário em que a loja recebe 25 novos clientes e 12 novos produtos ao ano , cada um a adquirir uma media de 2 produtos sendo que apenas em media 2 dos 25 novos clientes devolvem produtos para a loja, no final do ano teremos 8225 bytes para clientes(Customer + Phone + Address), 1350 bytes para vendas, 70 bytes para devoluções e 4320 bytes para produtos. Isto significa que, mesmo antes de limpar a base de dados no início do segundo ano, o espaço ocupado pelos novos dados será de 13965 bytes (aprox 14 kilobytes).

Imaginando que o stand tem um crescimento anual de 15% no volume de negócios de forma uniforme (15% mais Produtos, clientes, Vendas e Devoluções), a base de dados terá um crescimento de 2 aproximadamente kilobytes no segundo ano. Ao fim de uma década o sistema ocupará cerca de 55 kilobytes.

Isto são valores muito pequenos em comparação com o volume de informação que os discos atuais conseguem guardar, mesmo sabendo que existem cabeçalhosnos dados que estamos a ignorar como apontadores em VARCHAR().

## 5.7. Definição e caracterização de vistas de utilização

Quanto nos debroçamos sobre o tema ddecidimos elaborar vistas que fossem uteis das quais conseguíssemos tirar proveito no deenvolvimento de querries mais comoplicadas reduzindo o trabalho de pesquisa na base de dados. Assim procuramos blocos de querries mais comuns e que sabíamos que seriam necessárias para as defenições de tabelas de informação importantes, isto é, quando nos aperebiamos que várias queries faziam um mesmo conjunto de operações, colocamos isso numa vista. Depois só tivémos 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 diferenciação entre utilizadores e as permissões de controlo que lhe seriam disponibilizadas, isto é, devemos ceder permissões aos utilizadores unicamente para acções que façam parte das suas competências apenas, uma vez desnecessário e arriscado a um utilizador interagir com informação com a qual não deve trabalhar.

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

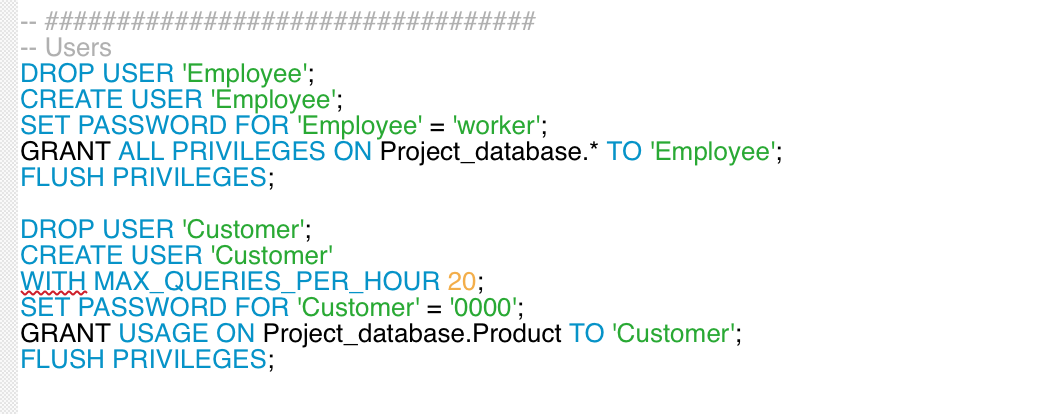
Os funcionários deverão ter acesso a todas as tabelas na base de dados, podendo assim registar, atualizar e apagar Produtos, Vendas, Clientes e Retornos. As permissões de criação são necessárias para inserir novos dados como registar vendas e Clientes novos do “Um Belo PC”.

As permissões visão a remoção de dados dados já não relevantes, como produtos que já não estão em inventário, retornos antigos ou clientes que escolham .

Mais restrito estarão os clientes, que terão apenas acesso para consultar o inventário da loja. Tal que podemos ceder acesso mais nenhuma tabela para manter a privacidade da informação que não compete aos Clientes (se pudesse aceder à Cliente poderia informação pessoal de outros clientes o que traria repercursões a loja “Um Belo PC” por não manter os direitos de privacidade do Cliente ).

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, vamosm ser nós a fazer uma retrospetiva do nosso trabalho.

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

No entanto este sistema, apesar de restrito, não deixa de ser capaz de funcionar para negócio real, sendo que necessitaria de ainda mais alguma implementação de interrogações relevantes capazes de agilizar o uso do Cliente, e um ajuste mais cuidado das permissões de modo a tornar todo o uso, quer do Cliente como Funcionario, mas simples e intuitivo. Como, normalmente, os funcionários não acedem a uma base de dados via terminal, será importante definir transações adicionais para disponibilizar como API.

# 6. Conclusões e Trabalho Futuro

Agora que a base de dados está pronta consegimos identificar algumas coisas que não correram como esperado. Antes de mais a coordenação foi razoável para um grupo que ficou definido a menos de duas semanas da entrega. As coisas tiveram que ser organizadas muito em cima da hora, com todos os elementos a trabalhar em coisas diferentes (uns em povoamentos, outros em queries, etc.), o que até correu melhor que o esperado.

Tivémos também problemas com o *forward engineering do brModelo* porque estava a gerar uma tabela Venda+Carro devido à cardinalidade entre as duas. Isto só foi resolvido ao olhar com atenção para as opções apresentadas ao gerar em vez de carregar OK em tudo.

Algo relevante foi a maneira como a matéria foi sendo lecionada, querendo com isto dizer que alguns elementos tiveram as aulas práticas sobre transações no próprio dia da entrega deste relatório.

Quanto ao uso futuro deste trabalho, não nos parece que esta base de dados vá ter grande uso para além do próximo trabalho (que se baseia neste).

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

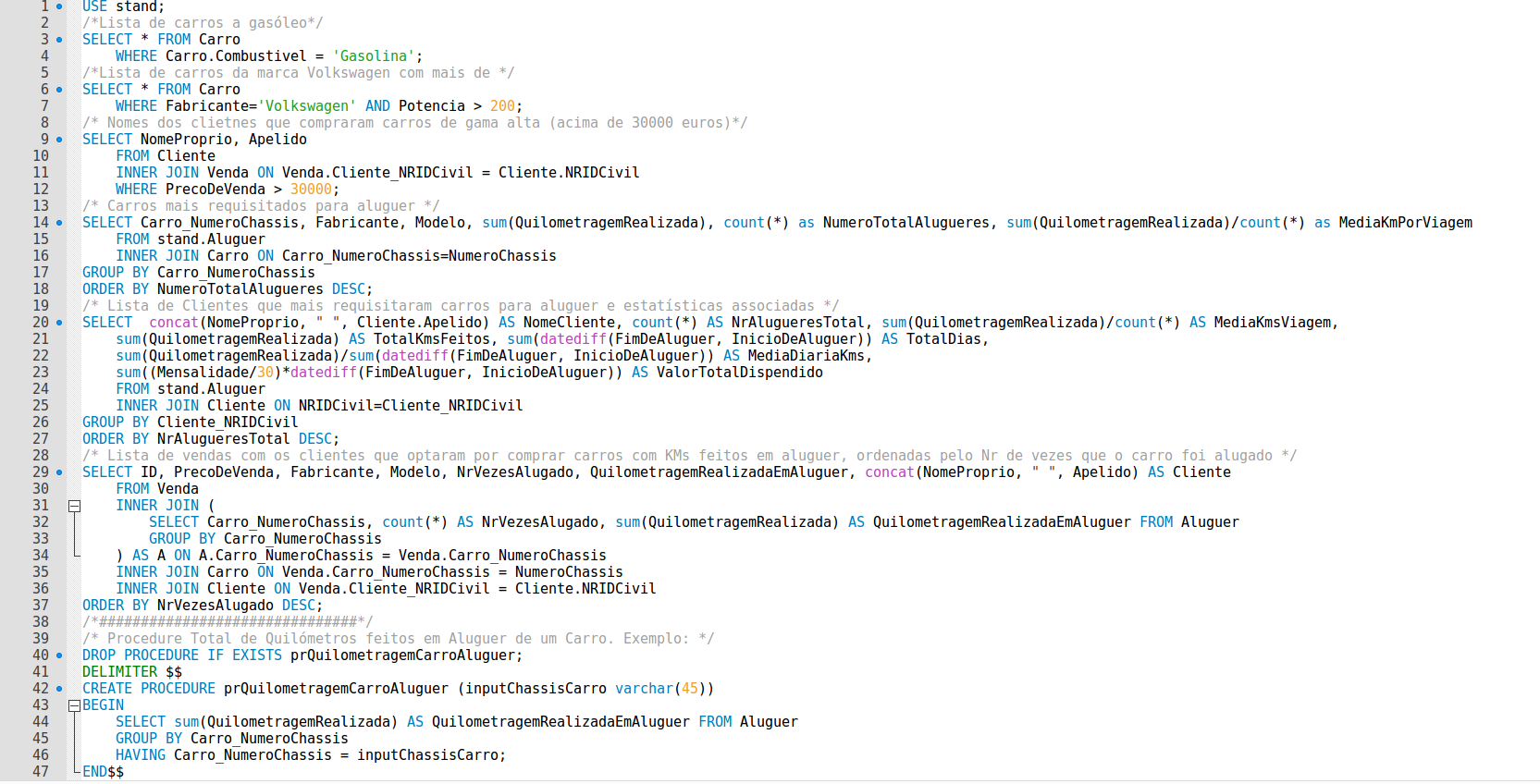
BD Base de Dados

UC Unidade Curricular

# Anexos

Ficheiro .sql com todas as scripts disponível em:

<https://mega.nz/#!PQNmgBCY!HbjnPVNEL0gFEbXocGMOgxSizGjHjP_IUajpOCHbnlg>

1. Parte 1 de 4 dos scripts
2. Parte 2 de 4 dos scripts



1. Parte 3 de 4 dos scripts
2. Parte 4 de 4 dos scripts

