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

Unidade Curricular de Bases de Dados

Ano Lectivo de 2017/2018

GestHotel

A70565 Bruno Arieira

A73974 Daniel Vieira

A74264 Rafael Silva

A74216 Rodrigo Ferreira

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

GestHotel

A70565 Bruno Arieira

A73974 Daniel Vieira

A74264 Rafael Silva

A74216 Rodrigo Ferreira

Novembro, 2016

Conteúdo

Índice de Figuras

Índice de Tabelas

1. Definição do sistema

O projeto consiste na implementação de uma base de dados para a gestão de reservas de quartos de hotéis da plataforma GestHotel. O funcionamento desta plataforma será semelhante a outras já existentes, como o Booking ou o Trivago, e tem como objetivo ajudar o cliente/utilizador na pesquisa de um hotel. Para isso basta, resumidamente, que o utilizador insira a localidade para onde se pretende dirigir, a data de check in e check out, e o número e capacidade de quartos que pretende reservar, assim como o tipo de reserva pretendida. Após a inserção destes dados, serão demonstradas todas as alternativas disponíveis que obedecem aos mesmos, tendo o utilizador que fazer o registo na plataforma antes de efetuar a reserva.

Em relação à base de dados propriamente dita, o primeiro passo é fazer o levantamento de requisitos, seguido do modelo conceptual que corresponda aos mesmos na perfeição (usando diagramas ER). Para que tal seja possível, é essencial uma identificação inequívoca de todas as entidades, relacionamentos, atributos e chaves. Após a definição do modelo concetual, é necessário validá-lo segundo os requisitos e as transações.

Depois disso, passaremos ao modelo lógico, onde passaremos a trabalhar essencialmente com tabelas e linguagem SQL.

Por último, será necessário fazer a transação para o modelo físico, onde terão que ser feitas a definição das relações base, a análise das transações e a escolha de índices para garantir a eficiência das queries.

Durante o relatório a seguir descrito, tentaremos sempre documentar com tabelas, imagens ou explicações por nós redigidas todas as conclusões que retirarmos, de forma a uma exposição clara e inequívoca das nossas ideias.

Área de Aplicação: Desenho e arquitetura de Sistemas de Bases de Dados.
Palavras-Chave: Levantamento de Requisitos, Modelo Conceptual, Modelo Lógico, Modelo Físico, Diagramas ER, linguagem SQL, Gestão de Índices, Queries.

1.1. Contexto da aplicação do sistema

O caso de estudo apresentado serve para que seja possível uma redução de tempo utilizado pelo na procura de um hotel que corresponda às suas pretensões. Assim, o utilizador não necessita de procurar um a um quais os hotéis que mais lhe agradam. Basta inserir na plataforma a cidade em que pretende ficar hospedado e a data de estadia que pretende, e as opções que correspondam a estes requisitos serão apresentadas pela plataforma. Sendo assim, este sistema insere-se num contexto de facilidade na procura de hotéis, facilitando o trabalho ao utilizador consoante as suas necessidades e pretensões.

1.2. Fundamentação da implementação da base de dados

O objetivo principal deste caso de estudo é criar um sistema de Base de Dados que respeite os requisitos das atividades da plataforma GestHotel, de forma a satisfazer os seus utilizadores. Para alcançar este propósito foi seguido um modelo de desenvolvimento de base de dados por etapas: começou-se pela recolha das informações relevantes para a base de dados a desenvolver, seguindo-se uma análise das informações recolhidas em plataformas já existentes com objetivos semelhantes (como por exemplo o Booking ou o Trivago), entrevistas aos clientes para perceber na íntegra quais as suas pretensões (semelhanças e diferenças entre a plataforma GestHotel e as já existentes, que foram anteriormente referidas), elaboração dos modelos conceptual, lógico e físico em conformidade com os requisitos, seguidos de uma validação dos mesmos. A documentação de todo o processo encontra-se agregada no presente relatório.

1.3. Análise da viabilidade do processo

Os sistemas de informação estão a crescer em dimensão e existe uma necessidade premente de uma resposta que para além de eficiente, seja eficaz. O caso de estudo não só permite perceber o funcionamento de um sistema de reservas de quartos de hotéis, numa primeira abordagem, como entender o funcionamento da plataforma.

Além do mais, o caso de estudo pode ser reaproveitado para outros casos de estudo, intimamente ligados com o setor de atividade da empresa, como por exemplo sistemas de reservas de mesas na restauração, ou até outro tipo de sistemas, como viagens de comboio, avião ou camioneta, dado que existe uma convergência entre a descrição de meios a ser reservados – quartos, lugares ou mesas – e, de igual modo necessário gerir reservas desses meios. O caso de estudo é bastante genérico, o que permite uma extensão, após nova análise dos requisitos ao modelo de dados de reservas.

Finalmente, o caso de estudo é uma oportunidade para solidificar de forma estruturada conhecimentos fundamentais na conceção, desenho e implementação de uma base de dados relacional.

1.4. Estrutura do Relatório

Este documento encontra-se divido em cinco capítulos: o primeiro é o capítulo introdutório, que trata da definição do sistema; o segundo é referente ao levantamento e análise de requisitos, em que se refere o método utilizado e é realizada a descrição dos mesmos; o terceiro trata da modelação conceptual, o quarto é referente à modelação lógica, o quinto diz respeito ao modelo físico e, por fim, o sexto e último capítulo, que diz respeito à descrição das conclusões e análise sobre o trabalho futuro.

No primeiro capítulo, que diz respeito à definição do sistema, é realizada uma breve descrição (resumo) do sistema, a fundamentação da implementação da base de dados (motivações e objetivos) e uma descrição da estrutura do presente relatório.

No segundo capítulo, referente ao levantamento e análise de requisitos, é apresentado o método por nós adotado para esse levantamento, seguido de uma descrição dos mesmos.

O capítulo seguinte, que diz respeito à modelação conceptual, trata da identificação e caracterização das entidades, relacionamentos, atributos e chaves, tendo presente a apresentação do diagrama ER correspondente ao nosso sistema.

No quarto capítulo é apresentado o modelo de dados lógico e a validação do mesmo.

No capítulo referente à modelação física, é revista a implementação do sistema no SBD (sistema de bases de dados) elegido, justificada a sua utilização e descrito o processo de implementação e criação da base de dados.

2. Levantamento e análise de requisitos

Neste capítulo é abordado o método de levantamento e da análise dos requisitos adotado e, depois disso, é feita a descrição e enumeração dos requisitos adotados para o sistema.

2.1. Método de levantamento e da análise de requisitos adotado

Para o levantamento e análise dos requisitos, decidimos optar por efetuar entrevistas aos clientes, ou seja, aos utilizadores para os quais a base de dados é efetuada.

Decidimos adotar este método porque pensamos ser o mais adequado para a elaboração do sistema em questão, pois facilita bastante a perceção do que é pretendido e permite alterações consoante o desejo dos clientes.

2.2. Requisitos levantados

Se estiver à procura de um hotel numa cidade que vá visitar, o que faz? Insere a palavra “hotel” no Google e vai procurar um a um todos os hotéis e comparar os preços para uma compra? Não acha isso trabalho a mais? E que tal uma plataforma que faça o trabalho por si, que procure todos os hotéis existentes numa cidade à sua escolha e mostre os resultados dessa pesquisa.

É com base nesta breve explicação que iremos fazer a nossa base de dados. A nossa plataforma, chamada “GestHotel”, é um sistema de gestão de bases de dados que vai relacionar o hotel com o destino em questão, alem dos quartos para que seja mais rápida a visualização dos diferentes tipos de quarto para fazer a reserva que pretende.

Para isso precisamos evidentemente de um utilizador, utilizador este que vai desfrutar da plataforma visto que é ele que vai usufruir do quarto que pretende reservar.

O utilizador irá colocar os dados que pretende, isto é, dando a informação necessária, irá introduzir a cidade que pretende que se faça a procura do hotel bem a capacidade do quarto, alem das datas de check-in e check-out. Com todos esses dados, o Sistema de Gestão de Bases de Dados GuestHotel irá selecionar os hotéis que sejam compatíveis com que o utilizador pretende.

Caracterizando a nosso sistema, decidimos que:

- ➔ O Hotel tenha as suas caraterísticas como nome, cidade, telefone e classificação, alem do quarto.
- ➔ No mesmo Hotel haja vários quartos.
- ➔ Escolhendo o quarto, o utilizador só tem que reservar.
- ➔ Uma reserva pode ser feita para um ou vários quartos.
- ➔ Um quarto terá implícito sobre ele, uma e uma só reserva.
- ➔ Uma cidade poderá ter vários hotéis, onde todos estes terão vários quartos.
- ➔ Um hotel só poderá estar numa cidade.
- ➔ Na reserva terá o Nome do cliente, Nome do hotel, número de quartos reservados e preço total, tal como data de reserva, check-in, check-out e o tipo de reserva feita (consoante a escolha das 3 opções).
- ➔ A reserva é feita por uma pessoa, no entanto essa mesma reserva poderá ter vários quartos reservados como já falamos anteriormente.

Para que o utilizador, conceba a sua reserva com sucesso, é necessário que realize todos os seguintes passos:

1. Pesquisar hotel segundo: cidade pretendida, data de check in, data de check out e capacidade do quarto.
2. Apresentar hotéis correspondentes aos dados inseridos pelo utilizador.
3. Utilizador tem que estar registado na plataforma para fazer a reserva, e para esse registo é necessário que o cliente forneça os seus dados pessoais: nome, número do cartão de cidadão, email, data de nascimento, morada, número de telefone, número de cartão de débito.
4. A plataforma verifica os dados do utilizador e faz a autenticação do mesmo.
5. O utilizador poderá proceder à reserva de um ou vários quartos no mesmo hotel, consoante o que pretende, sendo que em qualquer um dos casos isso corresponde a apenas uma reserva.
6. Um hotel, evidentemente, poderá ter várias reservas, enquanto que uma reserva só pode estar relacionada com um hotel.
7. Um hotel tem, como atributos, um nome, cidade, telefone e classificação e o quarto, que corresponde a um atributo multivalor.
8. Um hotel tem vários quartos.
9. Uma reserva só pode ser efetuada por um cliente, enquanto que um cliente pode efetuar várias reservas.
10. A reserva será apresentada com o nome do cliente, nome do hotel, número de quartos, preço, data de check in, data de check out, data da reserva e tipo de reserva.
11. É possível o cliente verificar o seu registo de reservas desde a sua autenticação na plataforma.
12. É possível ao utilizador consultar os hotéis de uma localidade consoante a sua classificação.
13. É possível ao utilizador consultar os hotéis de uma localidade consoante o seu preço.
14. Cada utilizador poderá escolher a reserva que melhor serve as suas necessidades, sendo que pode escolher uma entre três opções: usufruir de pequeno almoço, usufruir de todas as refeições ou não usufruir de nenhuma delas.

Depois de todos estes passos a reserva será efetuada com sucesso.

Em relação aos requisitos de controlo, apenas os gestores do sistema podem “entrar” na base de dados.

3. Modelação conceptual

3.1. Apresentação da abordagem de modelação realizada

A modelação conceptual é usada como representação de alto nível e considera exclusivamente o ponto de vista do usuário criador dos dados. Trata-se de um diagrama em blocos que demonstra todas as relações entre as entidades e os atributos. Esta modelação tem como principal objetivo dar suporte ao problema específica de um domínio.

3.2. Identificação e caracterização das entidades

- **Cliente**

  Dado o domínio do problema, a modelação da base de dados da plataforma GuestHotel, foi natural que a entidade Cliente tenha surgido com naturalidade. Analisando os requisitos, observa-se que esta entidade é essencial pois é para o cliente que a plataforma é funcional, tendo o mesmo um contacto direto com a plataforma, visto ser ele que efetua as reservas e decide acerca dos seus parâmetros.

Figura 1-Entidade Cliente

- **Reserva**

  Neste caso, estamos também a lidar com uma entidade fundamental e imprescindível para a base de dados que queremos implementar. Estabelece um contacto direto tanto com o cliente como com o hotel, e é através dela que o cliente se assegura que

conseguiu encontrar o que pretendia (o que, no fundo, é o principal objetivo da plataforma).

Figura 2-Entidade Reserva

- **Hotel**

  Logicamente que, na modelação de uma base de dados que tem como objetivo a reserva de um determinado hotel, esta entidade não poderia deixar de existir. Através da observação e análise dos requisitos, verificamos que grande parte dos mesmos refletem ações relacionadas com esta entidade (assim como com o atributo multivalorado "quartos", que iremos descrever mais à frente).

Figura 3-Entidade Hotel

Entidade	Definição
Reserva	Identifica a entidade propriamente dita sendo que tem vários atributos que a caraterizam.
Cliente	Identifica o cliente que pretende fazer a reserva, isto é, pretende reservar o(s) quarto(s) que pretende. É identificado com todos os seus dados pessoais.
Hotel	Esta entidade contem todos os quartos que a plataforma dispõe para fornecer ao cliente. Sendo que evidentemente há vários hotéis, em que cada um tem quartos diferentes evidentemente.

Tabela 1- Representação das Entidades

3.3. Identificação e caracterização dos relacionamentos

- **Cliente – Reserva**

Relacionamento: Cliente faz reserva.

Descrição: identifica a ação de um cliente fazer uma reserva.

Cardinalidade: Cliente(1); Reserva(N) - Um cliente pode fazer várias reservas, mas uma reserva só pode ser feita por um cliente.

Obrigatoriedade: Um cliente faz obrigatoriamente uma ou mais reservas e uma reserva é obrigatoriamente feita por um cliente.

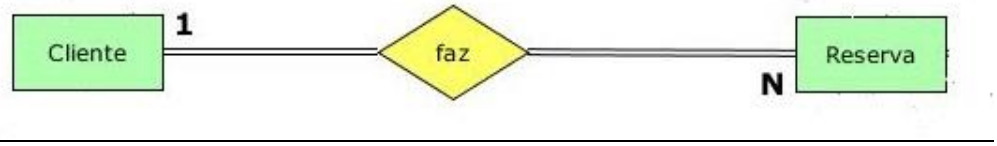


Figura 4- Relacionamento Cliente-Reserva

- **Reserva – Hotel**

Relacionamento: Reserva tem hotel.

Descrição: identifica a existência de um hotel numa reserva.

Cardinalidade: Reserva(N); Hotel(1) – Uma reserva tem apenas um hotel e um hotel pode ter várias reservas.

Obrigatoriedade: Uma reserva tem obrigatoriamente um hotel e um hotel tem obrigatoriamente reserva associada.

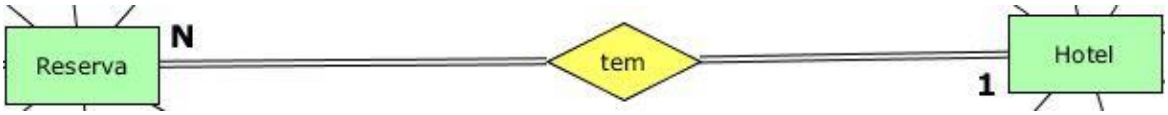


Figura 5- Relacionamento Reserva-Hotel

Entidade	Multiplicidade	Relacionamento	Multiplicidade	Entidade
Cliente	1	faz	N	*Reserva*
Reserva	1	possui	1	*Cliente*
Reserva	1	tem	1	*Hotel*
Hotel	1	tem	N	*Reserva*

Tabela 2- Representação dos Relacionamentos

3.4. Identificação e caracterização dos atributos

- **Atributos de Cliente**

  Excluindo a sua chave primária, idCliente, a entidade Cliente tem os seguintes atributos, divididos nos seus variados tipos:

  Atributos simples- Email, Contacto_Telefónico, NºCartao_Debito.

  Atributos compostos- Morada (constituído pelos atributos simples Rua, Localidade e Codigo_Postal), Cartao_Cidadao (constituído pelos atributos simples Nome e Numero).

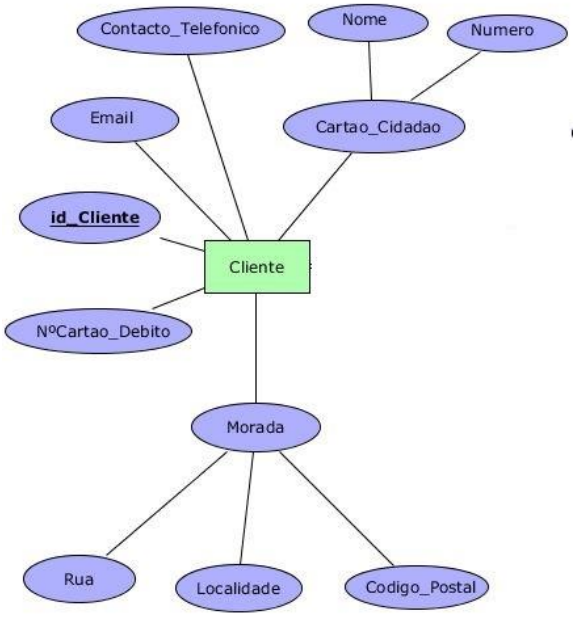


Figura 6- Atributos da Entidade Cliente

- **Atributos de Reserva**

  Excluindo a sua chave primária, idReserva, a entidade Reserva tem os seguintes atributos, divididos nos seus variados tipos:

  Atributos simples- Data_Reserva, Preco, Quantidade_Quartos, Data_Check_In e Data_Check_Out.

  Atributos multivalorados- Tipo_Reserva (é um atributo multivalorado porque as reservas podem diferir em algumas características: podem incluir as refeições todas do dia, incluir apenas pequeno almoço ou não incluir nenhuma refeição. Para além disso, a reserva pode corresponder ou não a uma suite.)

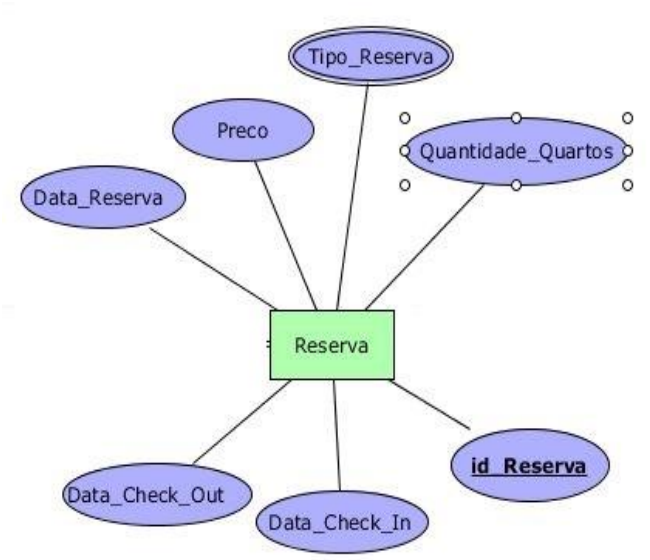


Figura 7- Atributos da Entidade Reserva

- **Atributos de Hotel**

  Excluindo a sua chave primária, id_Hotel, a entidade Hotel tem os seguintes atributos, divididos nos seus variados tipos:

  Atributos simples- Nome_Hotel, Cidade, Contacto, Classificacao, Endereco

  Atributos multivalorados- Quarto (é um atributo multivalorado porque existem vários quartos num hotel que podem diferir em determinadas características, tal como a capacidade de cada um).

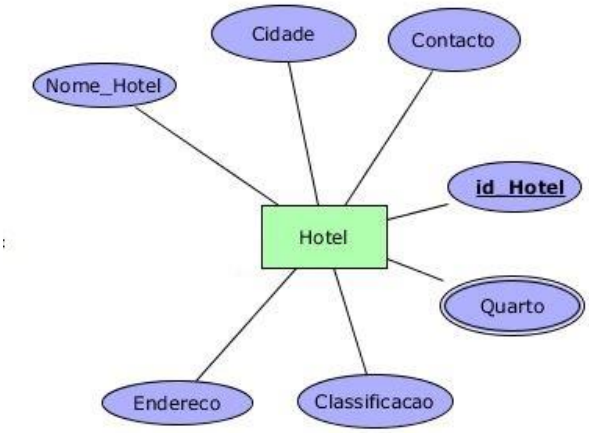


Figura 8- Atributos da Entidade Hotel

Morada

Atributo	Descrição	Tipo e Tamanho	Nulo	Multivalor	Derivado	Composto
Rua	Rua do Cliente	String com no máximo 45 de comprimento	Não	Não	Não	Não
Localidade	Localidade da rua em que vive o cliente	String com no máximo 45 de comprimento	Não	Não	Não	Não
Codigo-Postal	Codigo Postal do cliente	String com no máximo 20 de comprimento	Não	Não	Não	Não

Tabela 3- Atributo Composto Morada

Cartao_Cidadao

Atributo	Descrição	Tipo e Tamanho	Nulo	Multivalor	Derivado	Composto
Nome	Nome do Cliente	String com no máximo 45 de comprimento	Não	Não	Não	Não
Número	Número do cartão de cidadão	Inteiro com 8 bytes	Não	Não	Não	Não

Tabela 4- Atributo Composto Cartao_Cidadao

- **Dicionário de Dados**

Entidade	Atributos	Descrição	Tipo e Tamanho	Chave Primária	Chave Estrangeira	Nulo	Multivalor	Derivado	Composto
Cliente	Id_cliente	Identifica um determinado cliente	Inteiro positivo	Sim	Não	Não	Não	Não	Não
	Email	Email do cliente	String com no máximo 45 carateres	Não	Não	Não	Não	Não	Não
	NºCartao_Debito	Número de cartão de débito	Inteiro com 8 bytes	Não	Não	Não	Não	Não	Não
	Contacto_Telefónico	Contacto do cliente	Inteiro com no 9 carateres	Não	Não	Não	Não	Não	Não
	Morada	Morada do Cliente		Não	Não	Não	Não	Não	Sim
	Cartao_Cidadao	Dados Pessoais do Cliente		Não	Não	Não	Não	Não	Sim
Hotel	Id_Hotel	Identifica o hotel no qual a reserva foi feita	Inteiro positivo	Sim	Não	Não	Não	Não	Não
	Nome_Hotel	Nome do Hotel	String com no máximo 45 de carateres	Não	Não	Não	Não	Não	Não
	Cidade	Nome da cidade no qual o hotel se encontra	String com no máximo 45 de carateres	Não	Não	Não	Não	Não	Não
	Contacto	Contacto do Hotel	String com no máximo 20 de carateres	Não	Não	Não	Não	Não	Não
	Classificacao	Nota de 0 a 5	Float com 1 casa decimal sendo no máximo 2 carateres	Não	Não	Não	Não	Não	Não
	Endereco	Morada do Hotel	String com no máximo 100 de carateres	Não	Não	Não	Não	Não	Não
	Quarto	Quarto(s) do Hotel		Não	Não	Não	Sim	Não	Não

Reserva	Id_Reserva	Identifica a reserva	Inteiro positivo	Sim	Não	Não	Não	Não	Não
	Data_Check_In	Data do Check-In	DATE	Não	Não	Não	Não	Não	Não
	Data_Check_Out	Data do Check-Out	DATE	Não	Não	Não	Não	Não	Não
	Data_Reserva	Data de quando a reservafoi efetuada	DATETIME	Não	Não	Não	Não	Não	Não
	Preco	Preco da reserva efetuada(tudo incluído)	Float com 10 números sendo 2 para os decimais	Não	Não	Não	Não	Não	Não
	Quantidade_Quartos	Quantidados dos quartos que foram reservados	Inteiro com 4 números.	Não	Não	Não	Não	Não	Não

Tabela 5- Dicionário de Dados

3.5. Identificação das chaves

- **Cliente**

  Chaves candidatas: ContactoTelefonico, Email, NºCartaoDebito, idCliente, CartaoCidadao

  Chave primária escolhida: idCliente

  Chaves secundárias: ContactoTelefonico, Email, NºCartaoDebito, CartaoCidadao

- **Reserva**

  Chaves candidata: idReserva

  Chave primária escolhida: idReserva

  Chaves secundárias:

- **Hotel**

  Chaves candidata: Contacto, idHotel

  Chave primária escolhida: idHotel

  Chaves secundárias: Contacto

3.6. Apresentação do diagrama ER

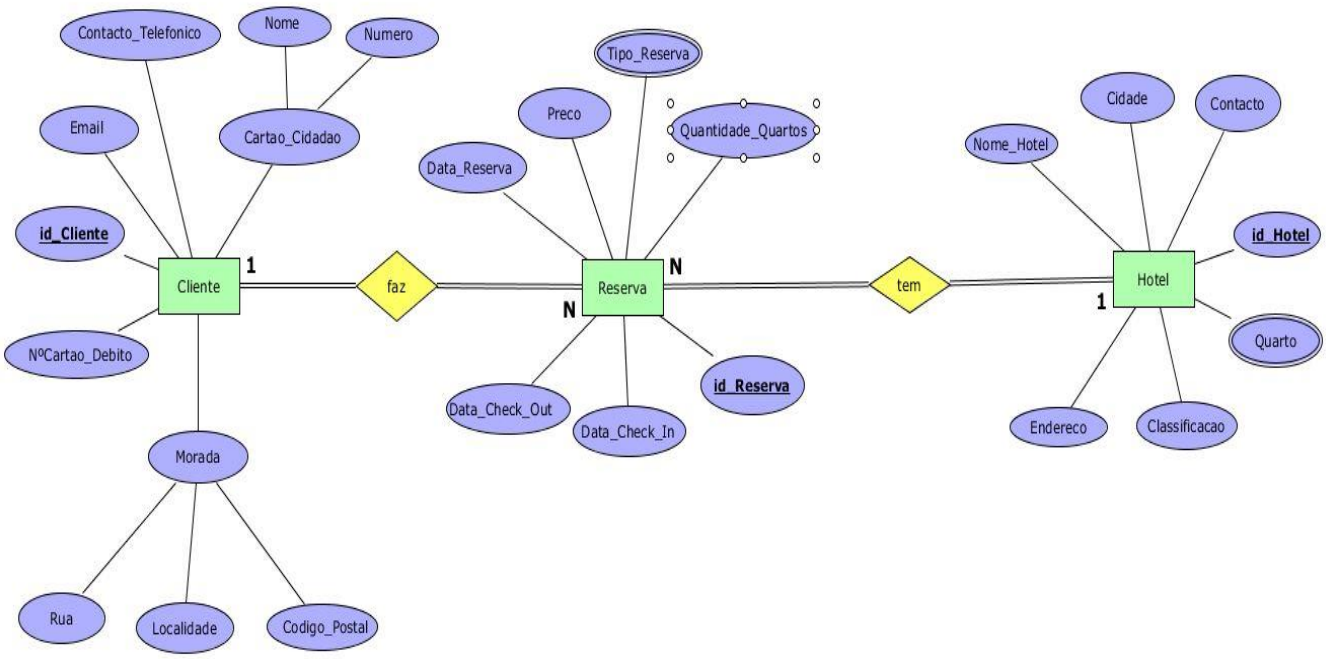


Figura 9- Diagrama ER

3.7. Validação do modelo de dados com o utilizador

Com o desenvolvimento e conclusão deste diagrama, possuímos o necessário e suficiente para prosseguir com o nosso sistema de gestão de base dados com sucesso, tendo sempre como principal objetivo, que um dado utilizador possa aceder á plataforma e tenha as funcionalidades todas a que tem direito.

Tivemos especial cuidado de analisar caso a caso, isto é, fizemos os testes necessários, para termos a certeza que podíamos prosseguir para o modelo lógico.

4. Modelação lógica

4.1. Ilustração do modelo lógico

Para esta etapa do processo de construção de uma base de dados apresentamos o respetivo modelo lógico. Para a representação deste modelo recorremos à utilização da ferramenta do MySQL Workbench.

Na tradução do modelo conceptual para o lógico deixamos de nos referir a entidades e relacionamentos passando a apresentar o mesmo conceito através de tabelas.

Nas secções que se seguem será tudo explicado de forma mais detalhada, incluindo todas as decisões tomadas na transformação do projeto aqui retratado desde o tratamento de entidades fortes ou fracas, relacionamentos de um para um, um para muitos, muitos para muitos, atributos multivalor, etc.

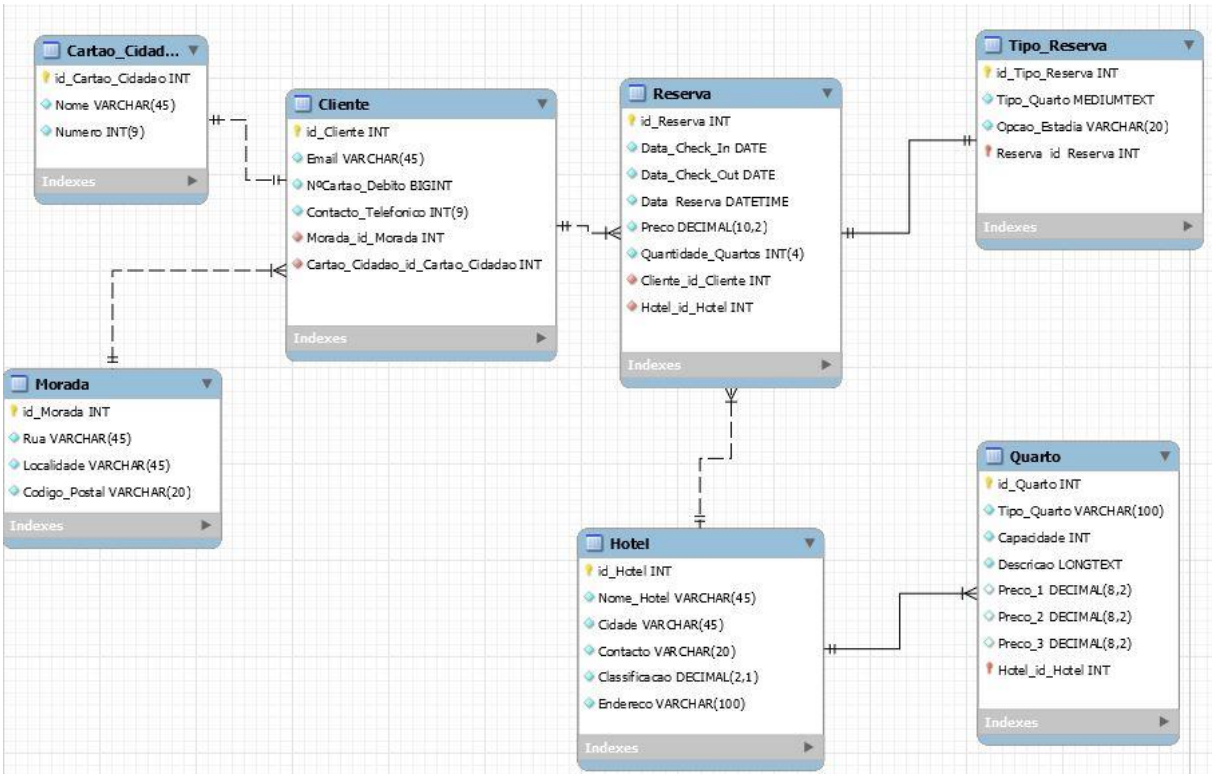


Figura 10- Modelo Lógico

4.2. Descrição das entidades e atributos multivalor

- **Cliente**

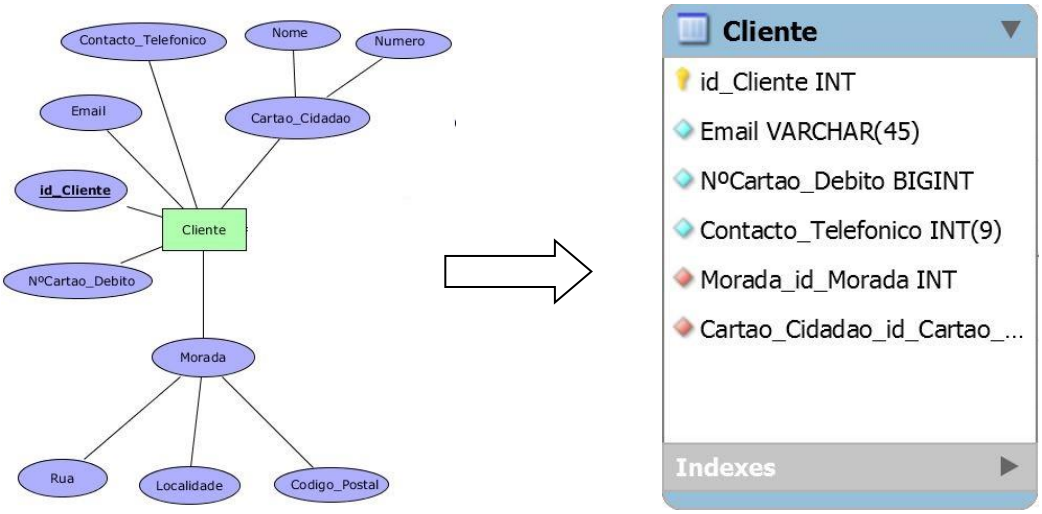


Figura 11- Conversão da Entidade Cliente

Cliente = {id_Cliente, Email, NºCartao_Debito, Contacto_Telefonico, {Rua, Localidade, Codigo_postal}, {Nome,Numero}}
Chave primária: id_Cliente
Atributos compostos: Cartao_Cidadao, Morada

- **Reserva**

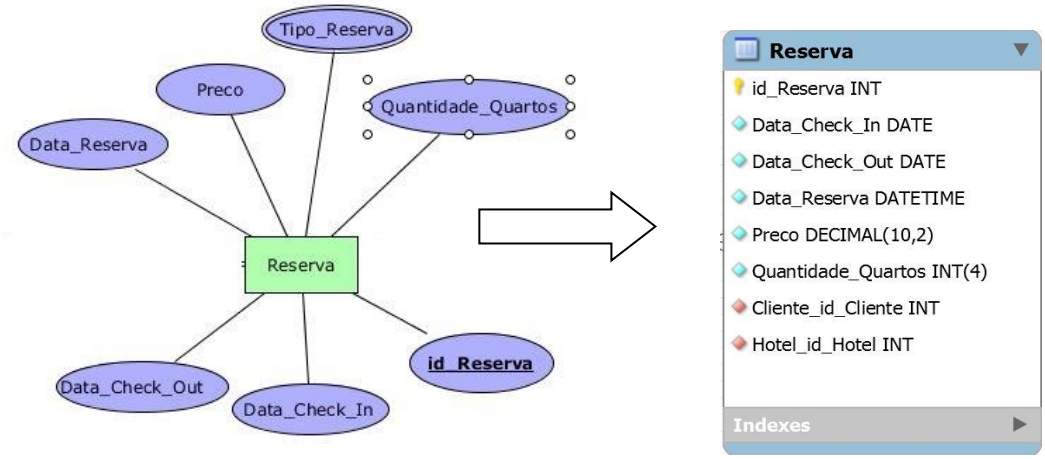


Figura 12- Conversão da Entidade Reserva

Reserva = {id_Reserva, Preco, Quantidade_Quartos Data_Reserva, Data_Check_In, Data_Check_Out, Tipo_Reserva}
Chave primária: id_Reserva
Atributos multivalor: Tipo_Reserva

- **Hotel**

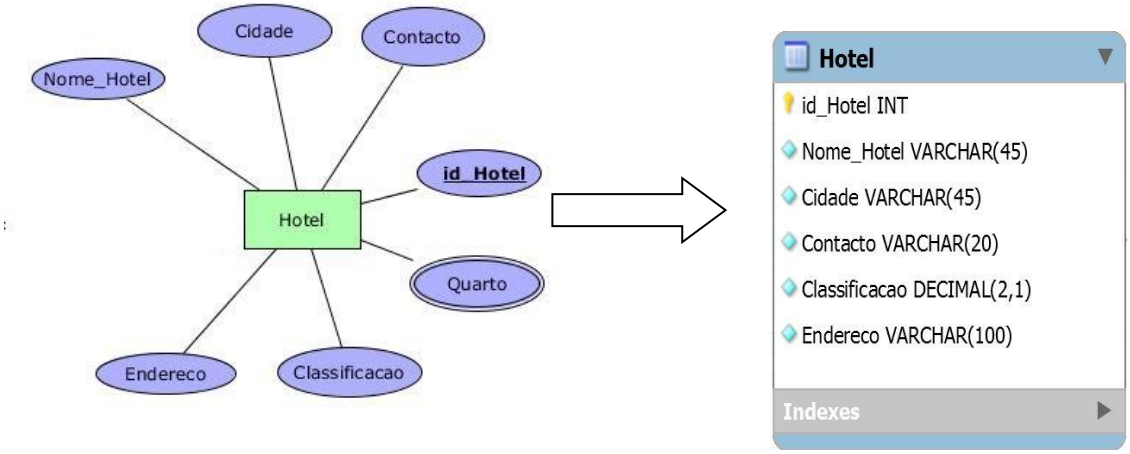


Figura 13- Conversão da Entidade Hotel

Hotel = {id_Hotel, Cidade, Contacto, Nome_Hotel, Quarto, Classificacao, Endereco}
Chave primária: id_Hotel
Atributos multivalor: Quarto

- **Cartao_Cidadao**

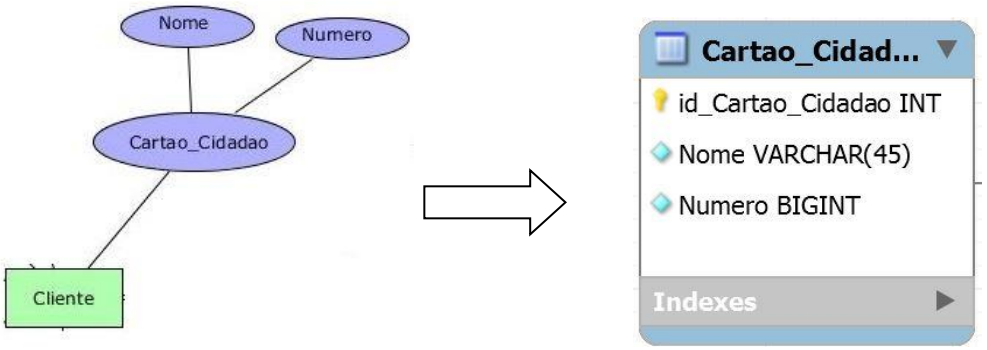


Figura 14-Conversão do atributo composto Cartao_Cidadao

- **Morada**

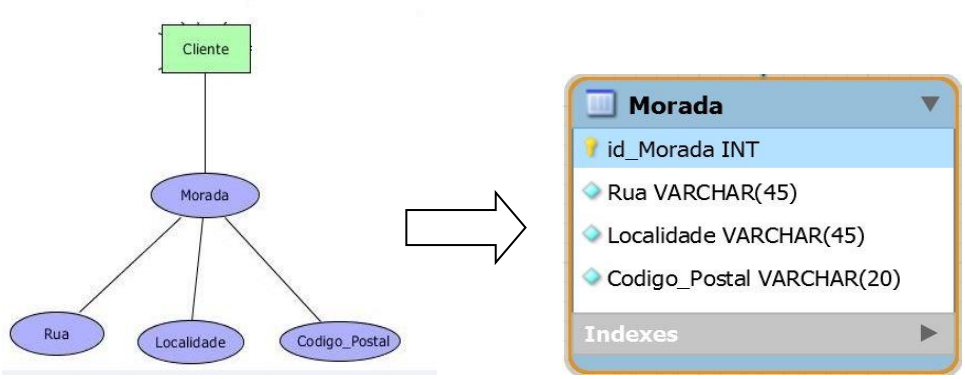


Figura 15- Conversão do atributo composto Morada

- **Tipo_Reserva**

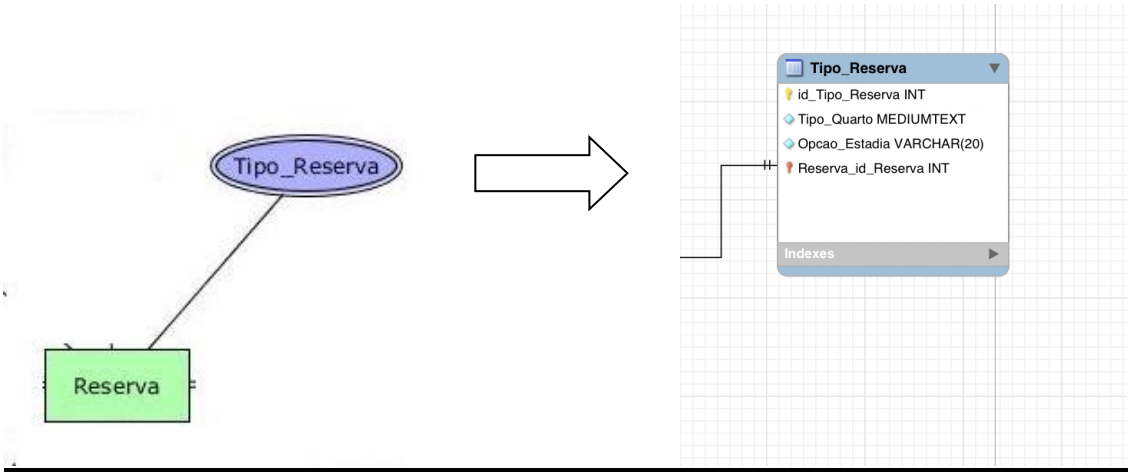


Figura 16- Conversão do atributo multivalor Tipo_Reserva

- **Quarto**

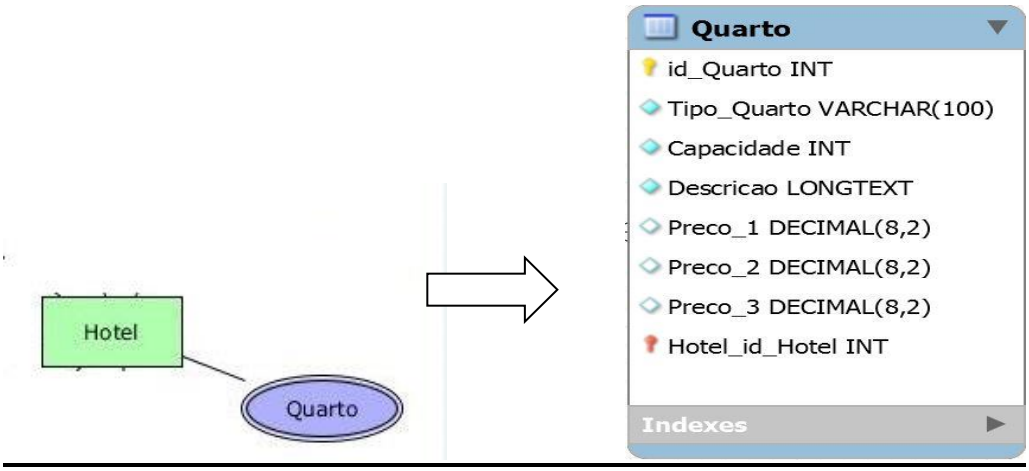


Figura 17- Conversão do atributo multivalor Quarto

4.3. Descrição dos relacionamentos

Cartão_Cidadao__Cliente

Chave principal: id_CartaoCidadao

Chave Estrangeira: Cartao_Cidadao_id_Cartao_Cidadao

A chave estrangeira deste relacionamento vai ser Carta_Cidadao_id_Cartao_Cidadao que vai estar evidentemente na tabela Cliente que vai ficar relacionado com a chave principal da outra tabela na qual se relaciona, chave esta que se chama id_cartaoCidadao.

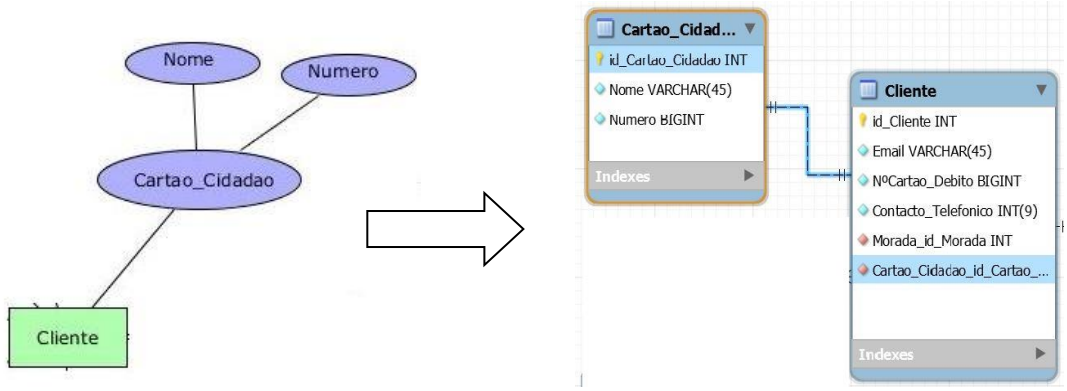


Figura 18- Conversão do relacionamento Cliente-Cartao_Cidadao

Morada__Cliente

Chave principal: id_Morada

Chave Estrangeira: Morada_id_Morada

A chave estrangeira deste relacionamento vai ser Morada_id_Morada que vai estar evidentemente na tabela Cliente que vai ficar relacionado com a chave principal da outra tabela na qual se relaciona,chave esta que se chama id_Morada.

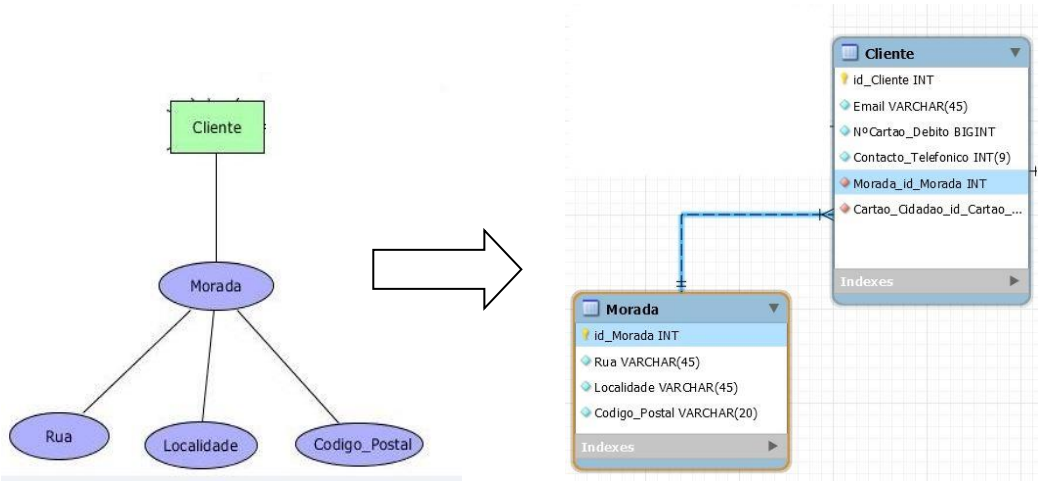


Figura 19 - Conversão do relacionamento Cliente-Morada

Cliente_Reserva

Chave principal: id_Cliente

Chave Estrangeira: Cliente_id_Cliente

A chave estrangeira deste relacionamento vai ser Cliente_id_Cliente que vai estar evidentemente na tabela Cliente que vai ficar relacionado com a chave principal da outra tabela na qual se relaciona,chave esta que se chama id_Cliente.

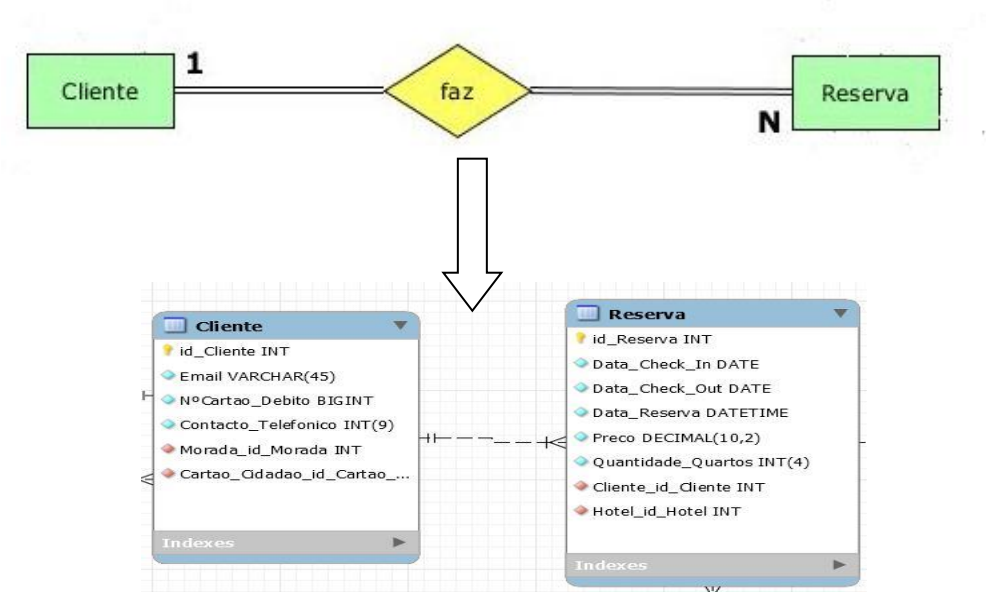


Figura 20- Conversão do relacionamento Cliente-Reserva

Hotel_Reserva

Chave principal: id_Hotel

Chave Estrangeira: Hotel_id_Hotel

A chave estrangeira deste relacionamento vai ser Hotel_id_Hotel vai estar evidentemente na tabela Cliente que vai ficar relacionado com a chave principal da outra tabela na qual se relaciona,chave esta que se chama id_Hotel.

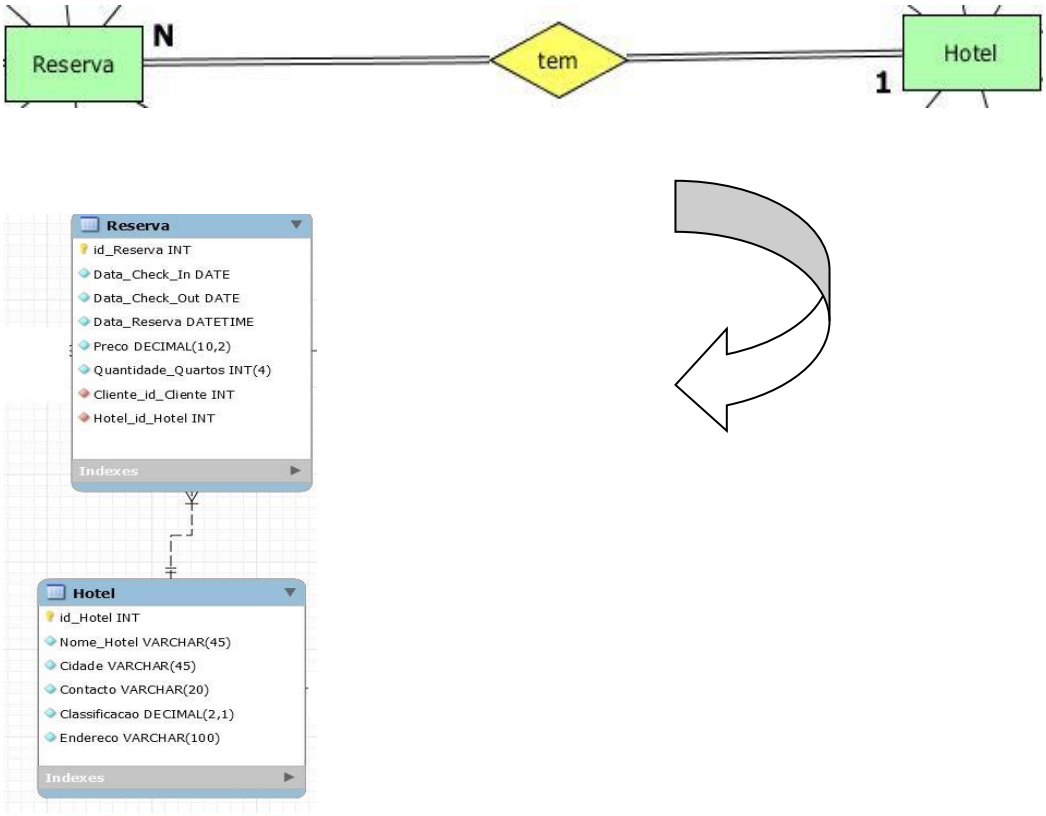


Figura 21- Conversão do relacionamento Hotel-Reserva

Reserva_Tipo_Reserva

Chave principal: id_Reserva

Chave Estrangeira: Reserva_id_Reserva

A chave estrangeira deste relacionamento vai ser Reserva_id_Reserva vai estar evidentemente na tabela Cliente que vai ficar relacionado com a chave principal da outra tabela na qual se relaciona,chave esta que se chama id_Reserva.

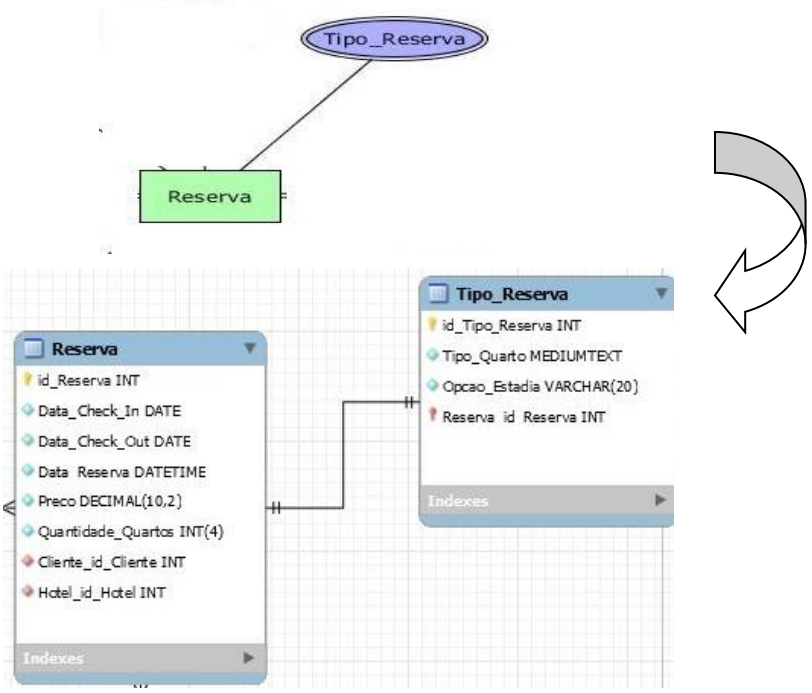


Figura 22 - Conversão do relacionamento Reserva-Tipo_Reserva

Hotel_Quarto

Chave principal: id_Hotel

Chave Estrangeira: Hotel_id_Hotel

A chave estrangeira deste relacionamento vai ser Hotel_id_Hotel vai estar evidentemente na tabela Cliente que vai ficar relacionado com a chave principal da outra tabela na qual se relaciona,chave esta que se chama id_Hotel.

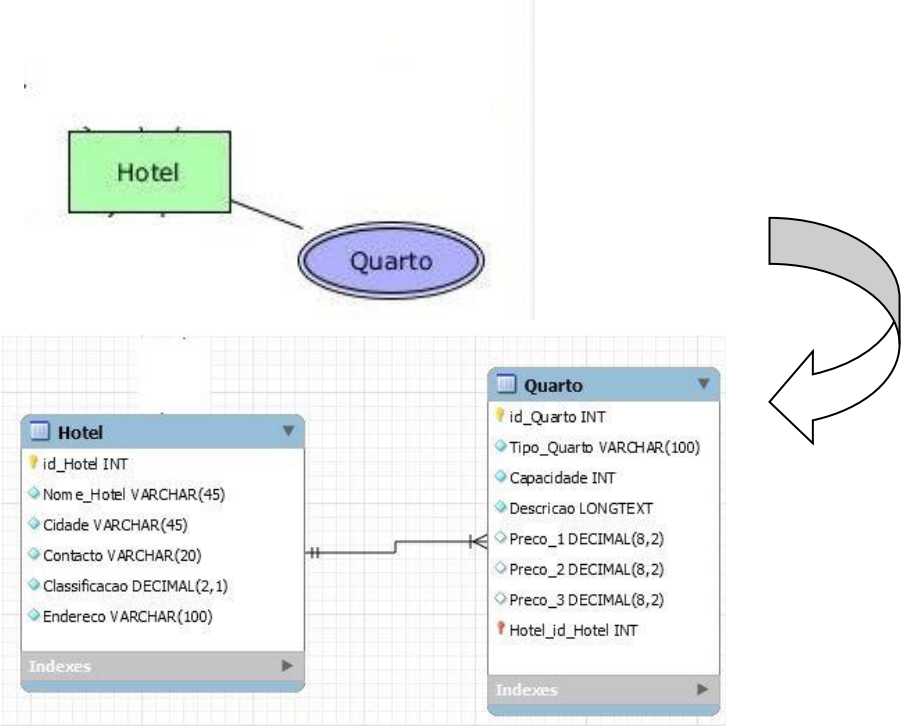


Figura 23- Conversão do relacionamento Hotel-Quarto

4.4. Validação do modelo através da normalização

Para garantir que o modelo não apresenta problemas futuros, é necessário validar as relações impostas pelo modelo lógico, recorrendo à normalização.

A normalização, neste projeto, tem como principal objetivo identificar as relações com base nas suas chaves e dependências entre os atributos e identificar relações com redundância de dados.

Aplicaremos a normalização até à Terceira Forma Normal validando o modelo lógico realizado para a plataforma de gestão de hotéis.

- **Primeira Forma Normal (1FN)**

  O objetivo desta forma de normalização, que é a primeira a ser aplicada, é identificar os possíveis elementos de informação repetidos nas tabelas.

  A estratégia de validação aplicada consiste em pesquisar entre tabelas informação que se encontre repetida e permitir um grau de facilitismo na identificação de uma chave-primária por tabela.

  Fazendo a verificação no nosso modelo, podemos afirmar que este já se encontra normalizado para a Primeira Forma Normal.

- **Segunda Forma Normal (2FN)**

  Depois de a Primeira Forma Normal ser aplicada, temos que verificar se todos os atributos que não são chaves candidatas estão dependentes de qualquer uma chave candidata, ou seja, não podemos permitir que existiam dependências parciais.

  A estratégia de validação aplicada consiste principalmente em verificar as tabelas com uma chave primária composta, pois as tabelas com uma só chave primária encontram – se já na Segunda Forma Normal.

  Sendo assim, através deste arquétipo partimos para a análise de cada atributo chave e verificamos a sua importância na tabela, identificando também os atributos que não são funcionalmente dependentes da chave-primária da tabela.

  Aplicando-a ao nosso modelo, verificamos que este já se encontra normalizado para a Segunda Forma Normal.

- **Terceira Forma Normal (3FN)**

Após a aplicação da Segunda Forma Normal, temos que verificar se todos os atributos que não são chaves candidatas serão alcançadas entre relações apenas de uma forma, ou seja, verificar as dependências transitivas.

A estratégia aplicada consiste em identificar os atributos que são funcionalmente dependentes de outros atributos que não são chaves – estes não poderão ser aceites na tabela.

Aplicando-a ao nosso modelo verificamos que este já se encontra normalizado também para a Terceira Forma Normal.

Depois da verificação do processo de normalização, concluímos que o modelo lógico realizado encontra-se normalizado até à Terceira Forma Normal, o que implica a inexistência de qualquer tipo de redundância de informação ou inconsistência na base de dados que coloque em perigo o seu funcionamento.

4.5. Validação do modelo com as transações estabelecidas

. Neste tópico iremos proceder à validação das transações no modelo lógico.
Iremos ilustrar com um exemplo sobre a nossa transação mais importante, fazer uma reserva.

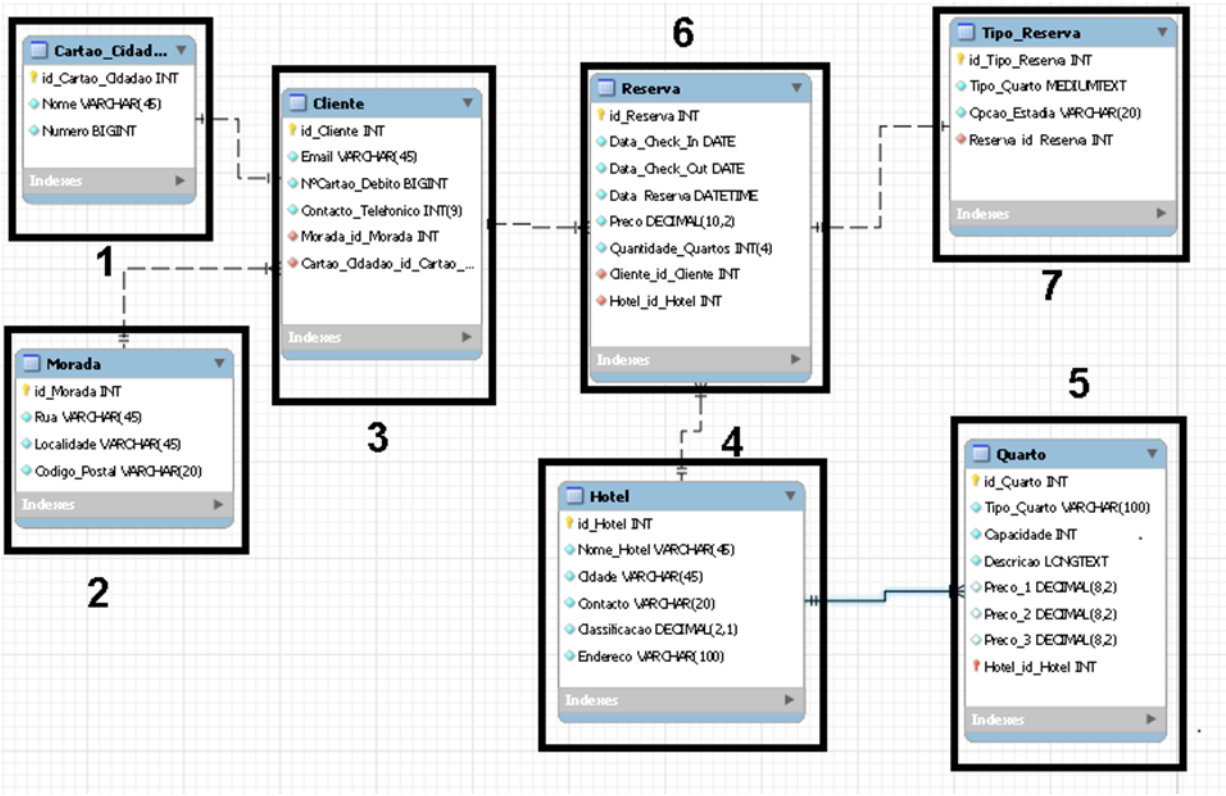


Figura 24- Exemplo de validação segundo as transações estabelecidas

Ao fazer uma reserva, o cliente tem que estar registado,logo tem que fornecer os seus dados pessoais. Os três primeiros passos a seguir são: fornecer o Cartão de Cidadão(1), a Morada(2), que irá completar o resto que falta na tabela 3 chamada Cliente(3).
Com o cliente registado o utilizador tem que escolher o Hotel(4),bem como o quarto (5).
Para finalizar irá ser a reserva propriamente dita(6) incluindo o Tipo_Reserva(7).

A figura acima é uma ilustração disso mesmo, encontrando-se numerada consoante a explicação acima descrita.
Assim, podemos concluir que este modelo lógico satisfaz os requisitos apresentados anteriormente.

4.6. Verificar restrições de integridade

Cliente={id_Cliente,Email,NºCartao_Debito,Contacto_Telefonico,Morada_id_Morada,Cartao_Cidadao_ id_Cartao_Cidadao**}**
Chave(s) Primária (s):id_Cliente (**Not NULL**, **Auto Increment**)
Chave(s) Estrangeiras: Morada_id_Morada, Cartao_Cidadao_id_ id_Cartao_Cidadao

Morada = {id_Morada, Rua, Localidade, Codigo_Postal }
Chave Primária: id_Morada(**Not NULL**, **Auto Increment)**
Referência desta chave primária a chave estrangeira Morada_id_Morada da tabela Cliente.

Cartao_Cidadao={ id_Cartao_Cidadao, Nome, Numero}
Chave(s) Primária(s): id_Cartao_Cidadao (**Not NULL**, **Auto Increment**)
Referência desta chave primária à chave estrangeira Cartao_Cidadao_id_ Cartao_Cidadao

Reserva={id_Reserva, Data_Check_In, Data_Check_Out, Data_Reserva, Preco, Quantidade_Quartos, Cliente_id_Cliente, Hotel_id_Hotel**}**
Chaves Primária(s): id_Reserva (**Not NULL**, **Auto Increment**)
Chave Estrangeira(s): Cliente_id_Cliente , Hotel_id_Hotel
A chave estrangeira Cliente_id_Cliente esta a relacionar-se com a chave principal da tabela Cliente,chave esta que se chama id_Cliente.

Hotel={id_Hotel, Nome_Hotel, Cidade, Contacto, Classificacao, Endereco**}**
Chaves Primária(s): id_Hotel (**Not NULL**, **Auto Increment**)
Esta chave primária refere-se à chave estrangeira Hotel_id_Hotel que se encontra na tabela Reserva.

Tipo_Reserva={id_Tipo_Reserva,Tipo_Quarto, Opcao_Estadia, Reserva_id_Reserva**}**

Chave Primária(s): id_Tipo_Reserva (**Not NULL**, **Auto Increment**)

Chave Estrangeira(s): Reserva_id_Reserva

Esta chave estrangeira esta relacionada com a chave principal da tabela Reserva id_Reserva.

Com esta demonstração concluímos que todas as restrições de integridade são verdadeiras. Assim, concluímos:

1- Nenhuma chave primária permite valores nulos.
2- Sempre que há uma chave estrangeira numa tabela, existe uma chave primária noutra tabela que lhe corresponde (que não pode ser nula).
3- O domínio e os tipos definidos nos requisitos foram corretamente restringidos no modelo lógico.
4- A multiplicidade mantém-se entre o modelo conceptual e o modelo lógico.

5. Implementação física

5.1. Seleção do sistema de gestão de bases de dados

Nesta última parte do projeto, o objetivo é fazer a conversão entre o modelo lógico e o modelo físico. Para que tal seja possível, é necessário recorrer a um sistema de gestão de bases de dados.

Utilizamos o motor de bases de dados MySQL Workbench, desenvolvido pela Oracle.

Depois disto, é necessário efetuar a construção das tabelas apresentadas anteriormente pelo modelo lógico, algumas *queries* (procedimentos, funções, entre outros) e transações que pensamos ser úteis para o funcionamento da nossa plataforma.

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

Como foi referido anteriormente, a tradução do modelo lógico para o modelo físico foi feita através do sistema de gestão de bases de dados MySQL Workbench.

Para que isto fosse possível, utilizamos um mecanismo presente no motor referido: *Forward Enginner*. Esta ferramenta permite que, a partir do modelo lógico desenvolvido, seja gerado o código SQL correspondente, respeitando os tipos de dados indicados e referências entre tabelas.

De seguida, iremos apresentar o código gerado correspondente às tabelas:


```
-- -----------------------------------------------------
-- Table `Gest_Hotel`.`Cliente`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Gest_Hotel`.`Cliente` (
  `id_Cliente` INT NOT NULL AUTO_INCREMENT,
  `Email` VARCHAR(45) NOT NULL,
  `NºCartao_Debito` BIGINT UNSIGNED NOT NULL,
  `Contacto_Telefonico` INT(9) UNSIGNED NOT NULL,
  `Morada_id_Morada` INT NOT NULL,
  `Cartao_Cidadao_id_Cartao_Cidadao` INT NOT NULL,
  PRIMARY KEY (`id_Cliente`),
  INDEX `fk_Cliente_Morada_idx` (`Morada_id_Morada` ASC),
  INDEX `fk_Cliente_Cartao_Cidadao1_idx` (`Cartao_Cidadao_id_Cartao_Cidadao` ASC),
  CONSTRAINT `fk_Cliente_Morada`
    FOREIGN KEY (`Morada_id_Morada`)
    REFERENCES `Gest_Hotel`.`Morada` (`id_Morada`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Cliente_Cartao_Cidadao1`
    FOREIGN KEY (`Cartao_Cidadao_id_Cartao_Cidadao`)
    REFERENCES `Gest_Hotel`.`Cartao_Cidadao` (`id_Cartao_Cidadao`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 25- Conversão da tabela Cliente


```
-- -----------------------------------------------------
-- Table `Gest_Hotel`.`Morada`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Gest_Hotel`.`Morada` (
  `id_Morada` INT NOT NULL AUTO_INCREMENT,
  `Rua` VARCHAR(45) NOT NULL,
  `Localidade` VARCHAR(45) NOT NULL,
  `Codigo_Postal` VARCHAR(20) NOT NULL,
  PRIMARY KEY (`id_Morada`))
ENGINE = InnoDB;
```

Figura 26- Conversão da tabela Morada

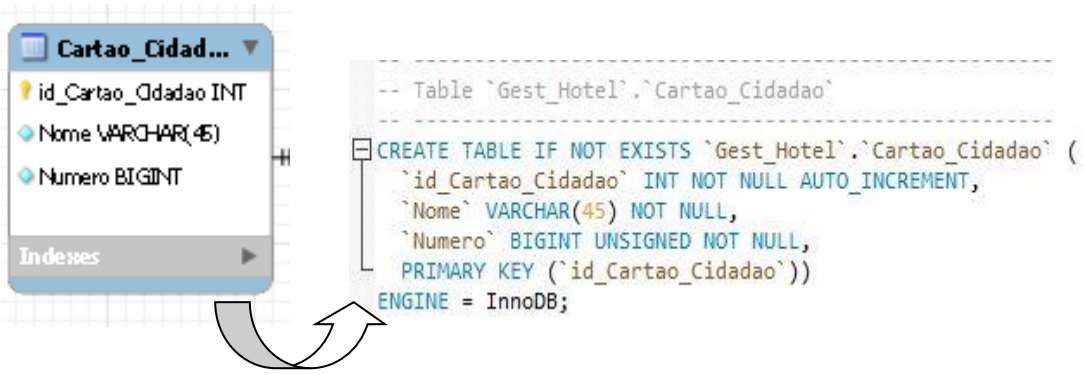


Figura 27- Conversão da tabela Cartao_Cidadao

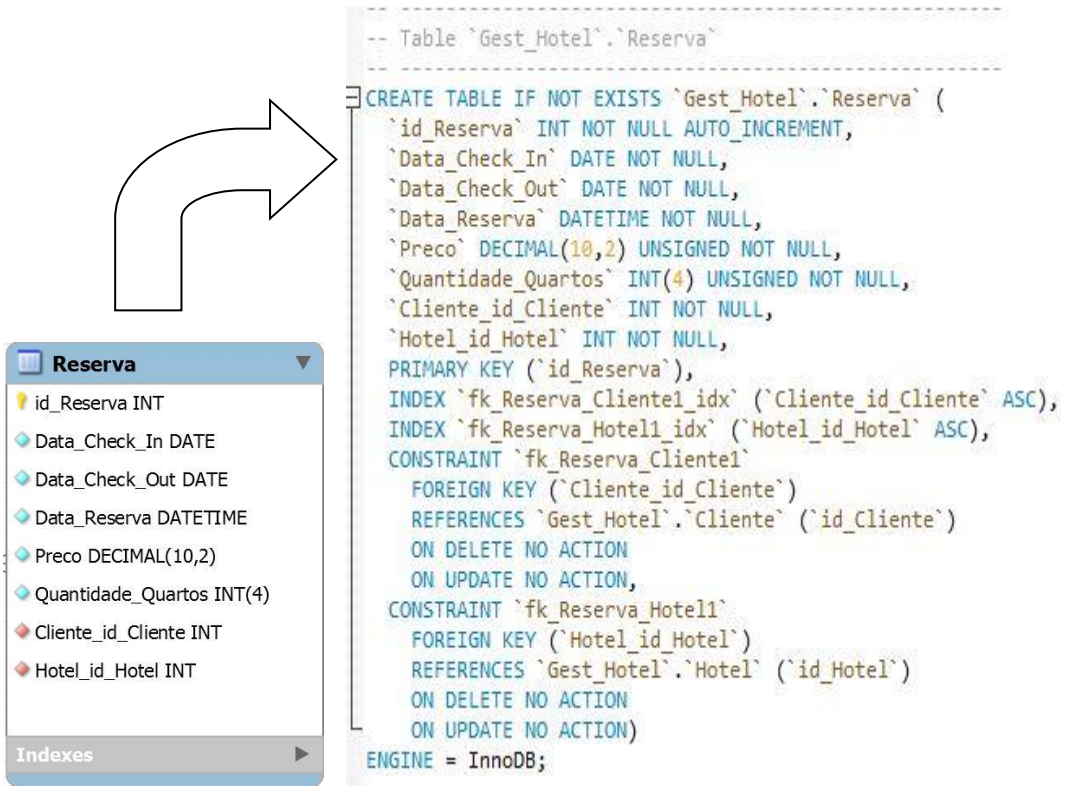


Figura 28- Conversão da tabela Reserva

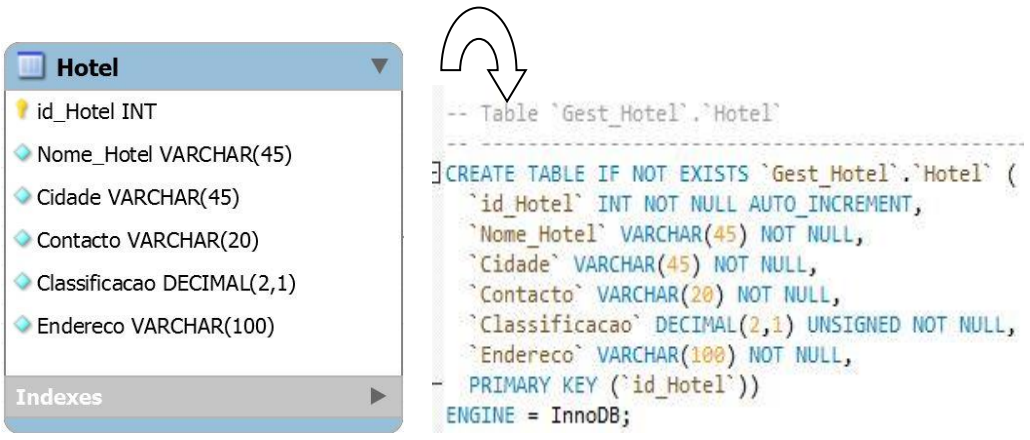
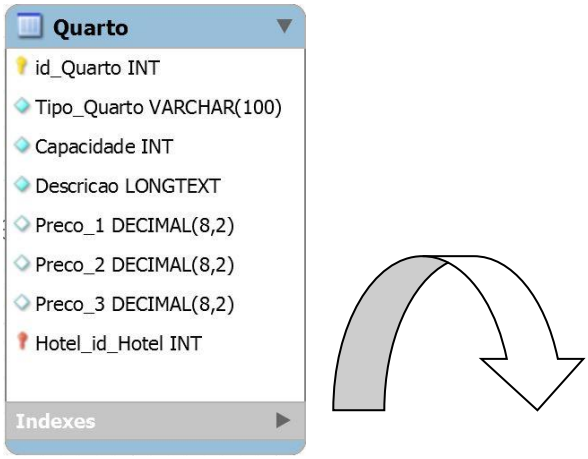


Figura 29 - Conversão da tabela Hotel




```
-- Table `Gest_Hotel`.`Quarto`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Gest_Hotel`.`Quarto` (
  `id_Quarto` INT NOT NULL AUTO_INCREMENT,
  `Tipo_Quarto` VARCHAR(100) NOT NULL,
  `Capacidade` INT UNSIGNED NOT NULL,
  `Descricao` LONGTEXT NOT NULL,
  `Preco_1` DECIMAL(8,2) UNSIGNED NULL COMMENT 'Preço só da reserva do quarto sem nada incluido.',
  `Preco_2` DECIMAL(8,2) UNSIGNED NULL COMMENT 'Preço só da reserva do quarto com pequeno-almoço incluido.',
  `Preco_3` DECIMAL(8,2) UNSIGNED NULL COMMENT 'Preço só da reserva do quarto com tudo incluido. ',
  `Hotel_id_Hotel` INT NOT NULL,
  PRIMARY KEY (`id_Quarto`, `Hotel_id_Hotel`),
  INDEX `fk_Quarto_Hotel1_idx` (`Hotel_id_Hotel` ASC),
  CONSTRAINT `fk_Quarto_Hotel1`
    FOREIGN KEY (`Hotel_id_Hotel`)
    REFERENCES `Gest_Hotel`.`Hotel` (`id_Hotel`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 30 - Conversão da tabela Quarto


```
-- -----------------------------------------------------
-- Table `Gest_Hotel`.`Tipo_Reserva`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Gest_Hotel`.`Tipo_Reserva` (
  `id_Tipo_Reserva` INT NOT NULL AUTO_INCREMENT,
  `Tipo_Quarto` MEDIUMTEXT NOT NULL,
  `Opcao_Estadia` VARCHAR(20) NOT NULL,
  `Reserva_id_Reserva` INT NOT NULL,
  PRIMARY KEY (`id_Tipo_Reserva`, `Reserva_id_Reserva`),
  INDEX `fk_Tipo_Reserva_Reserva1_idx` (`Reserva_id_Reserva` ASC),
  CONSTRAINT `fk_Tipo_Reserva_Reserva1`
    FOREIGN KEY (`Reserva_id_Reserva`)
    REFERENCES `Gest_Hotel`.`Reserva` (`id_Reserva`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 31- Conversão da tabela Tipo_Reserva

Tradução das interrogações do utilizador para SQL (alguns exemplos)

Baseado nos dados do utilizador, os procedures dão resposta às suas pertensões de uma forma mais restrita. De seguida, apresentamos alguns exemplos:

```
-- Procedure 1 (Todos dos clientes)

DELIMITER $$
drop procedure if exists getA_n_clientes $$
create procedure getA_n_clientes(in quantidade int)
  begin
       select * from Cliente limit quantidade;
  end $$
DELIMITER ;
```

Figura 32 - Procedure 1

```
-- Procedure 3 (Dado um nome, anuncia o preço associado)

DELIMITER $$
DROP procedure IF EXISTS name_Preco $$
CREATE PROCEDURE name_Preco(IN Nome VARCHAR(45))
BEGIN
       if (not exists (select cc.Nome from Cartao_Cidadao as cc where (cc.Nome = Nome)))
                then select 'Nome não existe na plataforma' as msg1;
       else
              select r.Preco from Reserva as r
              inner join Cliente as c on r.Cliente_id_Cliente = c.id_Cliente
              inner join Cartao_Cidadao as cc on cc.id_Cartao_Cidadao = c.Cartao_Cidadao_id_Cartao_Cidadao and Nome=cc.Nome
              group by r.Preco;
       end if;

END$$

DELIMITER ;
```

Figura 33 - Procedure 2

```
-- Procedure 4 (Dado um determinado dia, devolve o numero de reservas efetuadas nesse dia)


DELIMITER $$

DROP procedure IF EXISTS numero_Reservas $$
CREATE PROCEDURE numero_Reservas(IN Date1 date)
BEGIN
        if( not exists(select r.Data_Reserva from Reserva as r where (date(r.Data_Reserva)=Date1)))
            then select 'Nao foram feitas reservas neste dia' as msg1;

        else
            select Date1, count(*) as Numero_Reservas from Reserva as r
            where (date(r.Data_Reserva)=Date1);
        end if;
END$$

DELIMITER ;
```

Figura 34 - Procedure 3

```
-- Procedure 5 (Dado um nome, devolve o total de numero de quartos reservados por esse cliente, independentemente do hotel)

DELIMITER $$

DROP procedure IF EXISTS quantidade_Quartos $$
CREATE PROCEDURE quantidade_Quartos(IN Nome varchar(45))
BEGIN
       if(not exists( select cc.Nome from Cartao_Cidadao as cc where (cc.Nome=Nome)))
              then select 'Nao existe esse nome' as msg1;
       else
              select count(r.Quantidade_Quartos) as Numero_de_Quartos from Reserva as r
              inner join Cliente as c on r.Cliente_id_Cliente=c.id_Cliente
              inner join Cartao_Cidadao as cc on cc.id_Cartao_Cidadao = c.Cartao_Cidadao_id_Cartao_Cidadao and (cc.Nome = Nome);
       end if;
END$$

DELIMITER ;
```

Figura 35 - Procedure 4

```
-- Procedure 6 (Calcula o quarto mais barato, com tudo incluido, de um determinado hotel)

DELIMITER $$

DROP procedure IF EXISTS maisBarato $$
CREATE PROCEDURE maisBarato(IN Nome_Do_Hotel varchar(45))
BEGIN
	if(not exists( select h.Nome_Hotel from Hotel as h where (h.Nome_Hotel = Nome_Do_Hotel)))
		then select 'Esse hotel não existe' as msg1;
	else
		select Min(q.Preco_3) as Mais_barato from Quarto as q
		inner join Hotel as h on h.id_Hotel = q.Hotel_id_Hotel and h.Nome_Hotel = Nome_Do_Hotel;
	end if;
END$$

DELIMITER ;
```

Figura 36 - Procedure 5

```
-- Procedure 7 (Mostra todos os hoteis com uma determinada classificação)

DELIMITER $$

DROP procedure IF EXISTS hoteis_classificacao $$
CREATE PROCEDURE hoteis_classificacao(IN classificacao decimal(2,1))
BEGIN
	if(not exists( select h.Classificacao from Hotel as h where (h.Classificacao=classificacao)))
		then select 'Classificação nao atribuida a nenhum hotel' as msg1;
	else
	    select h.Nome_Hotel, h.Cidade, h.Classificacao from Hotel as h
	    where h.Classificacao=classificacao
	    order by h.classificacao;
	end if;
END$$

DELIMITER ;
```

Figura 37 - Procedure 6

```
-- Procedure 8 (Dá a faturacao de todos os hoteis, num determinado dia de reserva)

DELIMITER $$

DROP procedure IF EXISTS faturacao $$
CREATE PROCEDURE faturacao(in date1 date)
BEGIN
	if(not exists (select r.Data_Reserva from Reserva as r where (date(r.Data_Reserva)=date1)))
		then select 'Nao existe faturacao para esta data' as msg1;
	else
		select h.Nome_Hotel, sum(r.Preco) as Faturacao from Reserva as r
		inner join Hotel as h on h.id_Hotel=r.Hotel_id_Hotel
		where date(r.Data_Reserva) = date1
		group by h.id_Hotel;
	end if;
END$$

DELIMITER ;
```

Figura 38 - Procedure 7

5.3. Tradução das transações estabelecidas para SQL (alguns exemplos)

As transações são essenciais porque os dados estão em constante mutação, não são sempre iguais. Assim, por exemplo, se um cliente quiser sair da plataforma, liberta-se espaço que estava ocupado desnecessariamente, e isto só é possível recorrendo às transações. De seguida apresentamos alguns exemplos:

```
-- -- -- -- -- -- -- -- -- -- -- -- Inserir Morada -- -- -- -- -- -- -- -- -- -- -- -- -- --

DELIMITER $$
drop procedure if exists insere_morada $$
create procedure insere_morada(in rua varchar(45), in localidade varchar(45), in codigo_postal varchar(45))
    begin
        start transaction;

        if(exists(select m.Rua, m.Localidade, m.Codigo_Postal from Morada as m where m.Rua=rua and m.Codigo_Postal=codigo_postal and m.Localidade=localidade))
            then
                select 'Morada Existe' as msg1;
        else

            select @id_Morada := max(id_Morada) from Morada;

            set @id_Morada = @id_Morada + 1;

            insert into Morada(id_Morada, rua, localidade, codigo_postal)
                values
                        (@id_Morada, rua, localidade, codigo_postal);
            select 'Morada Inserida';

        commit;

        end if;
    end $$
DELIMITER ;
```

Figura 39 - Transação 1

```
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- Inserir Cartao Cidadao -- -- -- -- --

DELIMITER $$
drop procedure if exists insere_cartao_cidadao $$
create procedure insere_cartao_cidadao(in nome varchar(45), in numero varchar(45))
    begin
        start transaction;

        if(exists(select cc.Numero from Cartao_Cidadao as cc where numero=cc.Numero))
            then
                select 'Cliente existe' as msg1;

        else
            select @id_Cartao_Cidadao := max(id_Cartao_Cidadao) from Cartao_Cidadao;

            set @id_Cartao_Cidadao = @id_Cartao_Cidadao + 1;

            insert into Cartao_Cidadao(id_Cartao_Cidadao, Nome, Numero)
                values
                        (@id_Cartao_Cidadao, nome, numero);
            select 'Inserido_Cartao_Cidadao';

            commit;
        end if;
    end $$
DELIMITER ;
```

Figura 40 - Transação 2

```
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- Inserir Cliente -- -- -- -- -- -- -- -- -- -- -- -- -- --

DELIMITER $$
drop procedure if exists insere_Cliente $$
create procedure insere_Cliente(in email varchar(45), in n_cartao_debito bigint, in telefone int(9), in nome varchar(45),
                                in numero varchar(45), in rua varchar(45), in localidade varchar(45), in codigo_postal varchar(45))
    begin
        start transaction;

        if(exists(select c.Email  from Cliente as c where c.Email=email))
            then
                select 'Email existente' as msg1;
        else

            call `insere_cartao_cidadao`(nome,numero);
            call `insere_morada`(rua, localidade,codigo_postal);

            select @id_Cliente := max(id_Cliente)
            from Cliente;
            set @id_Cliente = @id_Cliente  + 1;

            insert into cliente(id_Cliente, Email, NºCartao_Debito)
            values(@id_Cliente, email, n_cartao_debito);

            select 'Registo efetuado' as Msg;

            commit;

        end if;
    end $$
DELIMITER ;
```

Figura 41 - Transação 3

```
-- -- -- -- -- -- -- -- -- -- (Update Morada) -- -- -- -- -- -- -- -- -- -- -- -- -- --

DELIMITER $$
drop procedure if exists update_morada $$
create procedure update_morada(in id int , rua varchar(45), in localidade varchar(45), in codigo_postal varchar(45))
    begin
        start transaction;

        if(exists(select m.Rua, m.Localidade, m.Codigo_Postal from Morada as m where m.Rua=rua and localidade=m.Localidade and m.Codigo_Postal=codigo_postal))
            then
                select 'Morada igual a que quer mudar.' as msg1;

        else

            update Morada as m
            set m.Rua = rua
            where m.id_Morada=id;
            commit;

            update Morada as m
            set m.Localidade = localidade
            where m.id_Morada=id;
            commit;

            select 'Morada nao existe' as Msg1;
            update Morada as m
            set m.Codigo_Postal = codigo_postal
            where m.id_Morada=id;

            select 'Update da morada efetuada' as Msg;
            commit;

        end if;

    end $$
DELIMITER ;
```

Figura 42 - Transação 4

```
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- Update cartao cidadao -- -- -- -- -- -- -- -- -- -- -- -- -- --


DELIMITER $$
drop procedure if exists update_cartao_cidadao $$
create procedure update_cartao_cidadao(in id int , in nome varchar(45), in numero varchar(45))
    begin
        start transaction;

        if(exists(select cc.Numero, cc.Nome from Cartao_Cidadao as cc where cc.Numero=numero and cc.Nome=nome))
            then
                select 'Cartao de cidadao igual ao que quer mudar.' as msg1;

            else

            update Cartao_Cidadao as cc
            set cc.Nome = nome
            where cc.id_Cartao_Cidadao=id;
            commit;

            update Cartao_Cidadao as cc
            set cc.Numero = numero
            where cc.id_Cartao_Cidadao=id;
            commit;

            select 'Update do cartao de cidadao efetuada.' as Msg;
            commit;

        end if;

    end $$
DELIMITER ;
```

Figura 43 - Transação 5

```
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- Update Cliente -- -- -- -- -- -- -- -- -- -- -- -- -- --

DELIMITER $$
drop procedure if exists update_cliente $$
create procedure update_cliente(in id_C int , in id_Cc int, in id_M int, in email varchar(45), in numero_debito bigint, in telefone int(9),
                in nome varchar(45), in numero varchar(45), in rua varchar(45), in localidade varchar(45), in codigo_postal varchar(45))
    begin

        start transaction;

        if(exists(select c.Email , c.NºCartao_Debito, c.Contacto_Telefonico from Cliente as c where c.Email=email and c.NºCartao_Debito=numero_debito
            and c.Contacto_Telefonico=telefone ))
            then
                select 'Cliente igual ao que quer mudar.' as msg;

            else

            call `update_morada`(id_M,rua,localidade,codigo_postal);
            call `update_cartao_cidadao`(id_Cc,nome,numero);

            update Cliente as c
            set c.Email = email
            where c.id_Cliente=id_C;
            commit;

            update Cliente as c
            set c.NºCartao_Debito = numero_debito
            where c.id_Cliente=id_C;
            commit;

            update Cliente as c
            set c.Contacto_Telefonico = telefone
            where c.id_Cliente=id_C;
            commit;


            select 'Update do cliente efetuado com sucesso.' as Msg;
            commit;

        end if;

    end $$
DELIMITER ;
```

Figura 44 - Transação 6

```
-- -- -- -- -- -- -- -- -- -- -- Insere Tipo de Reserva -- -- -- -- -- -- -- -- -- -- -- -- -- --

DELIMITER $$
drop procedure if exists insere_tipo_reserva $$
create procedure insere_tipo_reserva(in id_TR int, in tipo_quarto mediumtext, in opcao_estadia varchar(20), in id_R int)
    begin
        start transaction;

            insert into Tipo_Reserva(id_Tipo_Reserva, Tipo_Quarto, Opcao_Estadia, Reserva_id_Reserva)
                values
                        (id_TR, tipo_quarto, opcao_estadia, id_R);
            select 'Tipo_Reserva Inserida';

        commit;

    end $$
DELIMITER ;
```

Figura 45 - Transação 7

```
-- -- -- -- -- -- -- -- -- -- -- Insere Reserva -- -- -- -- -- -- -- -- -- -- -- -- -- --

DELIMITER $$
drop procedure if exists insere_Reserva $$
create procedure insere_Reserva(in data_i date, in data_o date, in data_r datetime, in preco decimal(10,2), in quantidade_quartos int(4), in tipo_quarto mediumtext,
                            in opcao_estadia varchar(20),  in id_C int, in id_H int)
    begin
        start transaction;

            if(exists(select r.Data_Check_In, r.Data_Check_Out from Reserva as r where data_i=r.Data_Check_In and data_o=r.Data_Check_Out))
                then
                    select 'Reserva ja existe.' as msg1;

            else

            select @id_Reserva := max(id_Reserva)
            from Reserva;
            set @id_Reserva= @id_Reserva  + 1;

            insert into reserva(id_Reserva, Data_Check_In, Data_Check_Out, Data_Reserva, Preco, Quantidade_Quartos, Cliente_id_Cliente, Hotel_id_Hotel)
            values(@id_Reserva, data_i, data_o, data_r, preco, quantidade_quartos, id_C, id_H);

            call `insere_tipo_reserva`(@id_Reserva, tipo_quarto, opcao_estadia,@id_Reserva);

            select 'Registo efetuado' as Msg;

            commit;

        end if;
    end $$
DELIMITER ;
```

Figura 46 - Transação 8

```
-- Inserir Hotel

DELIMITER $$
drop procedure if exists insere_Hotel $$
create procedure insere_Hotel(in nomeHotel varchar(45), in cidade varchar(45), in contacto varchar(20), in classificacao decimal(2,1), in endereco varchar(100))
    begin
        start transaction;
        if(exists (select h.Nome_Hotel, h.Cidade, h.Contacto, h.Classificacao, h.Endereco from Hotel as h where h.Nome_Hotel=nomeHotel and h.Cidade=cidade
                    and h.Contacto=contacto or h.Classificacao=classificacao and h.Endereco=endereco))
                        then select 'Hotel existe' as msg1;
        else
            if(exists (select h.Classificacao from Hotel as h where classificacao >= 0 and classificacao <= 5))
                then
                    select @id_Hotel := max(id_Hotel) from Hotel;

                    set @id_Hotel = @id_Hotel + 1;

                    insert into Hotel(id_Hotel, Nome_Hotel, Cidade, Contacto, Classificacao, Endereco)
                        values
                            (@id_Hotel, nomeHotel, cidade, contacto, classificacao, endereco);
                    select 'Hotel Inserido';

            else
                select 'Classificacao fora dos limites' as msg2;

            commit;

            end if;
        end if;
    end $$
DELIMITER ;
```

Figura 47 - Transação 9

```
-- -- -- -- -- -- -- -- -- -- -- Remove Reserva -- -- -- -- -- -- -- -- -- -- -- -- -- --

DELIMITER $$
drop procedure if exists remove_reserva $$
create procedure remove_reserva (in id_Tr int)
    begin
        start transaction;

            if(exists(select tr.id_Tipo_Reserva from Tipo_Reserva as tr where tr.id_Tipo_Reserva=id_Tr))
                then
                    delete from Tipo_Reserva
                    where id_Tipo_Reserva = id_Tr;

                    call `remove_tipo_reserva`(id_TR);

                    select 'Reserva eliminda';

            else

select 'Reserva nao existe' as msg2;

        commit;

            end if;
    end $$
DELIMITER ;
```

Figura 48 - Transação 10

```
-- -- -- -- -- -- -- -- -- -- -- Remove Tipo de Reserva -- -- -- -- -- -- -- -- -- -- -- --


DELIMITER $$
drop procedure if exists remove_tipo_reserva $$
create procedure remove_tipo_reserva (in id_R int)
    begin
        start transaction;

            if(not exists(select r.id_Reserva from Reserva as r where r.id_Reserva=id_R))
                then
                    select 'Reserva nao existe.' as msg1;

            else
                delete from Reserva
                    where id_Reserva = id_R;

        commit;

            end if;
    end $$
DELIMITER ;
```

Figura 49- Transação 11

```
DELIMITER $$
CREATE TRIGGER before_cliente_update
    BEFORE UPDATE ON Cliente
    FOR EACH ROW
BEGIN
    INSERT INTO Cliente
    SET action = 'update',
     id_Cliente= OLD.id_Cliente,
        Email= OLD.Email ,
       NºCartao_Debito = OLD.NºCartao_Debito,
        Contacto_Telefonico = OLD.Contacto_Telefonico;
END$$
DELIMITER ;
```

Figura 50 - Trigger 1

5.4. Escolha, definição e caracterização de índices em SQL

Não foram criados índices adicionais para além dos que foram criados no momento da geração do código SQL utilizando o MySQL Workbench.

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

Para que seja possível calcular uma estimativa do espaço em disco da base de dados, corremos o seguinte código SQL:

```
SELECT table_name as 'Tabela', (data_length+index_length) as 'Tamanho em Bytes'
        FROM information_schema.TABLEs
        where table_schema = 'gest_hotel'
        order by (data_length + index_length) DESC;
```

Figura 51 - Requisitos do espaço de base de dados

Como resultado, obtivemos o tamanho da nossa base de dados no momento (em bytes):

	Tabela	Tamanho em Bytes
	quarto	81920
	reserva	49152
	cliente	49152
	morada	16384
	hotel	16384
	cartao_cidadao	16384

Figura 52 - Resultados do tamanho da base de dados (em bytes)

5.6. Definição e caracterização das vistas da utilização em SQL (alguns exemplos)

Um dos pontos muito importantes no desenvolvimento de um Sistema de Gestão de Base de Dados, sao as vistas de utilização em SQL. Estas podem ser consideradas como uma tabela derivadada da consulta de outras tabelas. De seguida apresentamos alguns exemplos:

```
-- Query 1 (Quantidade de reservas por cada cliente)

drop view if exists reservas_totais;
create view reservas_totais as
    select cc.Nome,c.Email, count(*) as Total_Reservas from Cartao_Cidadao as cc
        inner join Cliente as c
            on c.Cartao_Cidadao_id_Cartao_Cidadao = cc.id_Cartao_Cidadao
        inner join Reserva as r
            on c.id_Cliente=r.Cliente_id_Cliente
        group by c.id_Cliente;

-- Query 2(Quantidade quartos resevados por cliente)

drop view if exists quartos_clientes;
create view quartos_clientes as
    select cc.Nome, c.Email, sum(Quantidade_Quartos) as Total_Quartos from Cartao_Cidadao as cc
        inner join Cliente as c
            on c.Cartao_Cidadao_id_Cartao_Cidadao = cc.id_Cartao_Cidadao
        inner join Reserva as r
            on c.id_Cliente=r.Cliente_id_Cliente
        group by c.id_Cliente;

-- Query 3 (Informaçao completa de cliente)

drop view if exists cliente_morada;
create view cliente_morada as
      select cc.Nome, cc.Numero, c.Email, c.NºCartao_Debito , c.Contacto_Telefonico, m.Rua , m.Localidade, m.Codigo_Postal  from  Cartao_Cidadao as cc
          inner join Cliente as c
              on cc.id_Cartao_Cidadao=c.Cartao_Cidadao_id_Cartao_Cidadao
          inner join Morada as m
              on m.id_Morada=c.Morada_id_Morada
      group by c.id_Cliente;

-- Query 4 (Hotel por Cliente)

drop view if exists hotel_por_cliente;
create view hotel_por_cliente as
        select cc.Nome, c.Email,  h.Nome_Hotel, r.Quantidade_Quartos from Cartao_Cidadao as cc
            inner join Cliente as c
                on cc.id_Cartao_Cidadao=c.Cartao_Cidadao_id_Cartao_Cidadao
            inner join Reserva as r
                on r.Cliente_id_Cliente=c.id_Cliente
            inner join Hotel as h
                on r.Hotel_id_Hotel=h.id_Hotel
        group by r.id_Reserva;

-- Query 5 (Total gasto por cada cliente)

drop view if exists total_gasto;
create view total_gasto as
    select cc.Nome, cc.Numero, c.Email, Sum(Preco) as Gasto from Cartao_Cidadao as cc
        inner join Cliente as c
            on cc.id_Cartao_Cidadao=c.Cartao_Cidadao_id_Cartao_Cidadao
        inner join Reserva as r
            on c.id_Cliente=r.Cliente_id_Cliente
    group by c.id_Cliente;
```

```sql
-- Query 6 (Tipo de reserva por cliente)

drop view if exists tipo_reserva_cliente;
create view tipo_reserva_Cliente as
    select r.id_Reserva, cc.Nome, c.Email, h.Nome_Hotel,  tr.Tipo_Quarto, tr.Opcao_Estadia from Tipo_Reserva as tr
        inner join Reserva as r
            on tr.id_Tipo_Reserva=r.id_Reserva
        inner join Hotel as h
            on h.id_Hotel=r.Hotel_id_Hotel
        inner join Cliente as c
            on c.id_Cliente = r.Cliente_id_Cliente
        inner join Cartao_Cidadao as cc
            on cc.id_Cartao_Cidadao=c.Cartao_Cidadao_id_Cartao_Cidadao
        group by r.id_Reserva;


-- Query 7(Duraçao de cada estadia)

drop view if exists duraçao_estadia;
create view duraçao_estadia as
    select cc.Nome, c.Email, r.id_Reserva, datediff(r.Data_Check_Out,r.Data_Check_In) as Duração from Hotel as h
        inner join Reserva as r
            on r.Hotel_id_Hotel=h.id_Hotel
        inner join Cliente as c
            on c.id_Cliente = r.Cliente_id_Cliente
        inner join Cartao_Cidadao as cc
            on cc.id_Cartao_Cidadao=c.Cartao_Cidadao_id_Cartao_Cidadao
        group by r.id_Reserva;

-- Query 8 (Facturaçao por cada Hotel)

drop view if exists faturacao_hotel;
create view faturacao_hotel as
    select  h.Nome_Hotel, h.Cidade, h.Classificacao, Sum(Preco) as Faturação from Hotel as h
        inner join reserva as r
            on h.id_Hotel=r.Hotel_id_Hotel
    group by h.id_Hotel;

-- Query 9 (Hotel por cidade)

drop view if exists hotel_cidade;
create view hotel_cidade as
    select h.Nome_hotel, h.Cidade, h.Classificacao, h.Contacto, h.Endereco from Hotel as h
    group by h.id_Hotel;

-- Query 10 (Quantidade de reservas feitas para um hotel)

drop view if exists hotel_reservas;
create view hotel_reservas as
    select h.Nome_Hotel, h.Cidade, count(*) as Quantidade_de_Reservas from Hotel as h
        inner join Reserva as r
            on r.Hotel_id_Hotel=h.id_Hotel
        group by h.id_Hotel;

-- Query 11(Quantas reservas por tipo quarto e hotel)
drop view if exists reservas_tipo_quarto;
create view reservas_tipo_quarto as
    select h.Nome_Hotel, tr.Tipo_Quarto, tr.Opcao_Estadia from Tipo_Reserva as tr
        inner join reserva as r
            on r.id_Reserva=tr.Reserva_id_Reserva
        inner join Hotel as h
            on r.Hotel_id_Hotel=h.id_Hotel;

-- Query 12(Quantidade de quartos por Hotel)

drop view if exists total_quartos_hotel;
create view total_quartos_hotel as
    select h.Nome_Hotel, h.Cidade, count(*) as Total_Quartos from quarto as q
        inner join hotel as h
        where h.id_Hotel=q.Hotel_id_Hotel
    group by h.id_Hotel;
```

```sql
-- Query 13 (os 10 melhores clientes da empresa)

drop view if exists ten_best_clients;
create view ten_best_clients as
select cc.Nome, c.Email, c.NºCartao_Debito , Sum(Preco) as Money from Cartao_Cidadao as cc
    inner join Cliente as c
        on c.Cartao_Cidadao_id_Cartao_Cidadao=cc.id_Cartao_Cidadao
    inner join Reserva  as r
        on r.Cliente_id_Cliente=c.id_Cliente
    group by c.id_Cliente order by Money desc
    limit 10;

-- Query 14 (Quantidade de reservas por dia)

drop view if exists quantidade_reserva;
create view quantidade_reserva as
    select date (r.Data_Reserva), count(*) as Numero_Reservas from Reserva as r
            group by date(r.Data_Reserva);


-- Query 15 (Quantidade de reservas feitas para um hotel)

drop view if exists hotel_reservas;
create view hotel_reservas as
    select h.Nome_Hotel, h.Cidade, count(*) as Quantidade_de_Reservas from Hotel as h
        inner join Reserva as r
            on r.Hotel_id_Hotel=h.id_Hotel
        group by h.id_Hotel;

-- Query 16(Quartos por Hotel)

drop view if exists quartos_hotel;
create view quartos_hotel as
    select h.Nome_Hotel, h.Cidade, q.Tipo_Quarto, q.Capacidade, q.Descricao from Quarto as q
        inner join Hotel as h
            where h.id_Hotel=q.Hotel_id_Hotel;


-- Query 17(Capacidade de cada Hotel)

drop view if exists capacidade_hotel;
create view capacidade_hotel as
    select  h.Nome_Hotel, Sum(Capacidade) from Hotel as h
        inner join Quarto as q
            on h.id_Hotel=q.Hotel_id_Hotel
        group by h.id_Hotel;

-- Query 18(Classficaçao do hotel)

drop view if exists classificacao_hotel;
create view classificacao_hotel as
        select h.Nome_Hotel, h.Endereco, h.Cidade, h.Classificacao from Hotel as h
        order by h.Classificacao;
```

Figura 53 - Exemplos de Views

5.7. Definição e caracterização dos mecanismos de segurança em SQL (alguns exemplos)

Normalmente, no desenvolvimento de um Sistema de Gestão de Bases de Dados há restrições relativamente ás vistas de cada utilizador. Assim na nossa implementação, distinguimos os seguintes tipos de utilizadores:

- admin: utilizador que tem acesso a toda a Bases de Dados e que pode executar todas as opções que bem entender como por exemplo apagar/selecionar registos,criação de uma nova tabela bem como caso o queira em situações mais dramática,apagar toda a base de dados.
- user: utilizador que apenas tem acesso ás reservas, hotéis, assim como os quartos. Tem apenas permissão para fazer uma reserva, mas no entanto, não tem permissão para retirar.

Assim sendo, apenas o administrador tem autorização para inserir, atualizar e remover dados.

```
-- Criaçao de Utilizadores e atribuiçao de proivilegios --

create user 'admin'@'localhost'
    identified by '123';
grant all privileges on Gest_Hotel.*
    to 'admin'@'localhost' ;

create user 'user'@'localhost'
    identified by '1234';
grant select on Gest_Hotel.*
    to 'user'@'localhost';
grant insert, update on Gest_Hotel.Cliente
    to 'user'@'localhost';
grant insert, update on Gest_Hotel.Cartao_Cidadao
    to 'user'@'localhost';
grant insert, update on Gest_Hotel.Morada
    to 'user'@'localhost';
grant insert, update on Gest_Hotel.Reserva
    to 'user'@'localhost';
grant insert, update on Gest_Hotel.Tipo_Reserva
    to 'user'@'localhost';
```

Figura 54 - Segurança de dados

5.8. Povoamento da base de dados (alguns exemplos)

Para o povoamento da nossa base de dados, recolhemos informações acerca de alguns clientes, hotéis, reservas, quartos, tipo de reservas, moradas e cartões de cidadão, e usamos as transações criadas para adicionar a informação na nossa plataforma, de forma a ilustrar o seu funcionamento. De seguida, apresentamos apenas alguns exemplos do povoamento que efetuamos para cada uma das tabelas.

```
insert into Cartao_Cidadao

(id_Cartao_Cidadao, Nome, Numero)

values

(default, 'Maria Josefa dos Santos', '12345678'),
(default, 'Bruno Manuel Borlido Arieira', '67453212'),
(default, 'Carlos Daniel Vieira', '14543585'),
(default, 'Rafael Machado Silva', '17834908'),
(default, 'Rodrigo Miguel Ferreira', '82536487'),
(default, 'José António Carvalho', '98273453'),
(default, 'Clara Deolinda Silvério', '87264387'),
(default, 'Luis Miguel Bravo Ferraz', '12324239'),
(default, 'Diogo Meira Neves', '67234120'),
(default, 'Nelson Arieira Parente', '19897864'),
(default, 'Marcos Morais Luis', '65473287'),
(default, 'Cesario Miguel Perneta', '31546548'),
(default, 'Sofia Novais Bravo', '74185296'),
(default, 'Pedro Miguel Pereira', '36925847'),
(default, 'Maria Joao Soares', '85269341'),
(default, 'Rosa Margarida Leitão', '75391568'),
(default, 'Diogo Silva Meira', '32145698'),
(default, 'Tiago Fernandes Barbosa', '81392764'),
(default, 'Ana Rita Faria', '94381675'),
(default, 'Liliana Fernandes Venade', '21657843'),
(default, 'Maria Joaquina Costa', '12312331'),
(default, 'Nelson Morais Faria', '31231212'),
(default, 'Carlos Daniel Leitão', '14312315'),
```

```
insert into Cliente

(id_Cliente, Email, NºCartao_Debito, Contacto_Telefonico, Morada_id_Morada, Cartao_Cidadao_id_Cartao_Cidadao)

values

(Default, 'josefamaria@hotmail.com', '1237659871234567', '916378972', '5','1'),
(Default, 'arieirabruno@gmail.com', '9876543456273894', '912345678', '6', '2'),
(Default, 'carlitos_vieira@sapo.iol.pt', '1928347918724329','946352918', '7', '3'),
(Default, 'rafa.machados@hotmail.com', '8764524708774382', '914272387', '9', '4'),
(Default, 'ferreira_rodrigo_scp@hotmail.com', '9879384292047548', '952637812', '11', '5'),
(Default, 'zecarvalho@hotmail.com', '5468983473298499', '962719823', '10', '6'),
(Default, 'deolinda_desfado@yahoo.com', '9283748273948787', '924362374', '8', '7'),
(Default, 'loismferraz@sapo.iol.pt', '9237498579348758', '912632534', '3', '8'),
(Default, 'neves_diogo@hotmail.com', '2312848273483743', '987654657', '4', '9'),
(Default, 'nelsonparente@gmail.com', '9874324234234234', '943758654', '2', '10'),
(Default, 'marcos_rodovia@hotmail.com', '2873423199473312', '975143213', '1', '11'),
(Default, 'pernetacesario@sapo.iol.pt', '1823534621772131', '952331230', '12', '12'),
(Default, 'sophiebravo@sapo.iol.pt', '7987165423415267', '924364712', '15', '13'),
(Default, 'pmpereira@gmail.com', '1234518782398112', '987654321', '14', '14'),
(Default, 'mariajsoars14@hotmail.com', '8276418782723123', '943657232', '13', '15'),
(Default, 'guidarosa@hotmail.com', '6234623576128761', '987623834', '20', '16'),
(Default, 'meiradioguinho@gmail.com', '2187687648819890', '969234573', '18', '17'),
(Default, 'tiago_nandes@hotmail.com', '2093872875817987', '967854621', '16', '18'),
(Default, 'anarita_parente@yahoo.com', '4545645938032032', '921198374', '17', '19'),
(Default, 'lilivenade@hotmail.com', '3214553466753237', '912746823', '19', '20'),
(Default, 'quinadascostas@gmail.com', '4657654871232345', '923345211', '25', '21'),
(Default, 'nelsonfarias@yahoo.com', '6789312314231212', '924874365', '21', '22'),
(Default, 'leitao_daniel@yahoo.com', '8734655314312315', '976433423', '26', '23'),
(Default, 'arigato_marquicos@sapo.iol.pt', '8623341231439081', '976547581', '23', '24'),
```

```
insert into Hotel

(id_Hotel, Nome_Hotel, Cidade, Contacto, Classificacao, Endereco)

values

(default,'El Palace Barcelona' ,'Barcelona' ,'+31 23 12 34 456' , '4.8', 'Gran Via de les Corts Catalanes, 668, Eixample, 08010 Barcelona, Espanha'),
(default,'Hotel Liabeny' ,'Madrid' ,'+32 12 32 111' , '4.7', 'Salud, 3, Centro de Madrid, 28013 Madrid, Espanha'),
(default,'NH Madrid Nacional' ,'Madrid' ,'+32 45 12 342' , '4.0', 'Paseo del Prado, 48, Centro de Madrid, 28014 Madrid, Espanha'),
(default,'Jupiter Lisboa Hotel' ,'Lisboa' ,'+21 34 21 321' , '4.2', 'Avenida da Republica, 46, Avenidas Novas, 1050-195 Lisboa, Portugal'),
(default,'Sheraton Porto Hotel & Spa' ,'Porto' ,'+23 23 12 233' , '5.0', 'Rua Tenente Valadim, 146, Lordelo do Ouro e Massarelos, 4100-476 Porto, Portugal'),
(default,'Hotel The Peninsula Paris' ,'Paris' ,'+35 32 13 432' , '5.0', '19 avenue Kleber, 16º arrondissement, 75116 Paris, França'),
(default,'Park Plaza Westminster Bridge London' ,'Londres' ,'+34 21 31 312' , '4.3', 'Westminster Bridge Road, Lambeth, Londres, SE1 7UT, Reino Unido'),
(default,'Meliá Berlin' ,'Berlim' ,'+36 21 32 212' , '4.5', 'Friedrichstr. 103, Mitte, 10117 Berlim, Alemanha'),
(default,'Vincci Gala' ,'Barcelona' ,'+31 32 12 432' , '4.0', 'Ronda Sant Pere, 32, Eixample, 08010 Barcelona, Espanha'),
(default,'Hotel Princesa Lisboa Centro' ,'Lisboa' ,'+21 23 21 323' , '3.5', 'Rua Gomes Freire, 130, Arroios, 1150-180 Lisboa, Portugal'),
(default,'The Church Palace' ,'Roma' ,'+33 21 34 211' , '4.6', 'Via Aurelia 481, Aurélio, 00165 Roma, Itália'),
(default,'Hotel Rialto' ,'Veneza' ,'+30 43 65 676' , '4.8', 'Riva del Ferro, San Marco, 30124 Veneza, Itália');
```

```
insert into Morada

(id_Morada, Rua, Localidade, Codigo_Postal)

values

(default, 'Rua das Tulipas','Barcelos', '4750-159'),
(default, 'Rua dos Peões','Braga', '4750-176'),
(default, 'Rua Ramblas','Barcelona', '4750-165'),
(default, 'Rua dos Feitos', 'Viana do Castelo', '4750-132'),
(default, 'Rua do Rossio', 'Lisboa', '4750-112'),
(default, 'Rua Passeig de Gràcia' ,'Barcelona', '4750-141'),
(default, 'Rua do Apoio', 'Barcelos', '4712-141'),
(default, 'Rua do Loureiro' ,'Porto', '4212-112'),
(default, 'Avenida dos Aliados', 'Porto', '4319-134'),
(default, 'Rua de José Falcão','Porto', '4750-131'),
(default, 'Gran Via','Madrid', '4123-159'),
(default, 'Calle Léon','Madrid', '3123-112'),
(default, 'Champs Elysees','Paris', '4123-534'),
(default, 'Rue de Vaugirard','Paris', '3120-131'),
(default, 'Avenue des Ternes','Paris', '6554-154'),
(default, 'Alexanderplatz','Berlim', '4750-121'),
(default, 'Avenida Unter den Linden','Berlim', '4123-231'),
(default, 'Friedrichshain', 'Berlim', '2331-112'),
(default, 'Oxford Stree', 'Londres', '2130-122'),
(default, 'Regent Street', 'Londres', '3123-312'),
(default, 'Fleet Street', 'Londres', '3123-131'),
(default, 'Rua Arbat', 'Moscovo', '2131-139'),
(default, 'Rua das Calçadas', 'Barcelos', '4750-999'),
```

```
insert into Quarto

(id_Quarto, Tipo_Quarto, Capacidade, Descricao, Preco_1, Preco_2, Preco_3, Hotel_id_Hotel)

values

](default, 'Quarto Duplo Deluxe ', '2', '1 cama de casal grande Luxuosos e espaçosos quartos exteriores,
           decorados num estilo clássico com mobiliário elegante. 38 m² - Varanda com vista Varanda - Vista cidade -
           Televisão de ecrã plano - Ar condicionado - Banheira Casa de banho privativa - Wi-Fi gratuito - Varanda',
-          '320.00', '349.00', '389.00', '1'),
](default, 'Quarto Duplo Deluxe ', '2', '1 cama de casal grande Luxuosos e espaçosos quartos exteriores, decorados
            num estilo clássico com mobiliário elegante. 38 m² - Varanda com vista Varanda - Vista cidade -
            Televisão de ecrã plano - Ar condicionado - Banheira Casa de banho privativa - Wi-Fi gratuito -
-           Varanda', '320.00', '349.00', '389.00', '1'),
](default, 'Quarto Duplo Deluxe', '2', '1 cama de casal grande Luxuosos e espaçosos quartos exteriores, decorados
            num estilo clássico com mobiliário elegante. 38 m² - Varanda com vista Varanda - Vista cidade -
            Televisão de ecrã plano - Ar condicionado - Banheira Casa de banho privativa - Wi-Fi gratuito -
-           Varanda', '320.00', '349.00', '389.00', '1'),
](default, 'Suite Júnior','2','1 cama de casal extra-grande. Esta Suite tranquila tem vista para um típico pátio
            interior de Barcelona e apresenta área de estar e quarto de vestir separado. O quarto está equipado
            com televisão com ecrã plano HD de 94 cm e telefone IP. Também dispõe de ampla casa de banho em mármore
            com produtos de higiene pessoal Occitane en Provence e duche separado, com banheira de hidromassagem.' ,
-           '490.00', '538.10', '574.00', '1'),
](default, 'Suite Júnior','2','1 cama de casal extra-grande. Esta Suite tranquila tem vista para um típico pátio
            interior de Barcelona e apresenta área de estar e quarto de vestir separado. O quarto está equipado
            com televisão com ecrã plano HD de 94 cm e telefone IP. Também dispõe de ampla casa de banho em mármore
            com produtos de higiene pessoal Occitane en Provence e duche separado, com banheira de hidromassagem.' ,
-           '490.00', '538.10', '574.00', '1'),
](default, 'Suite Júnior Premier','4', 'Esta suite luxuosa, bem iluminada e espaçosa apresenta uma banheira romana
            decorada com mosaicos e duche com efeito de chuva, além de produtos de higiene pessoal LOccitane. Também dispõe
            de uma área de estar separada com vista da Gran Via. Alguns quartos têm lareiras originais e todos apresentam
            mobiliário em estilo clássico, 2 televisões, telefone com ligação IP e acesso gratuito à Internet', '570.00',
-           '618.00', '645.00', '1'),
```

```
insert into Reserva

(id_Reserva,  Data_Check_In,  Data_Check_Out, Data_Reserva, Preco, Quantidade_Quartos, Cliente_id_Cliente, Hotel_id_Hotel)

values

(default, '2018-05-02', '2018-05-03', '2017-11-21 11:31:00', '389.00', '1' , '1','1'),
(default, '2018-05-03', '2018-05-06', '2017-12-5 14:23:00', '1284.72', '2' , '1','2'),
(default, '2018-05-04', '2018-05-08', '2017-11-24 11:12:23', '1396.00', '1' , '1','1'),
(default, '2018-05-05', '2018-05-10', '2017-12-20 12:11:22', '626.60', '1' , '1','2'),
(default, '2018-05-06', '2018-05-07', '2017-11-15 13:41:22', '404.24', '2' , '1','3'),
(default, '2018-05-12', '2018-05-14', '2017-12-15 14:21:21', '1104.00', '2' , '1','4'),
(default, '2018-05-12', '2018-05-14', '2017-12-12 15:11:00', '2808.66', '3' , '1','5'),
(default, '2018-04-30', '2018-05-03', '2017-12-01 16:31:32', '2781.96', '1' , '1','6'),
(default, '2018-05-21', '2018-05-24', '2017-11-21 17:34:00', '1269.33', '1' , '1','7'),
(default, '2018-04-30', '2018-05-03', '2017-11-27 18:35:21', '2056.00', '2' , '1','8'),
(default, '2018-05-13', '2018-05-15', '2017-11-29 19:36:00', '380.42', '1' , '1','9'),
(default, '2018-05-01', '2018-05-03', '2017-12-20 21:37:21', '140.00', '1' , '1','10'),
(default, '2018-05-02', '2018-05-03', '2017-11-10 22:21:20', '140.77', '1' , '1','11'),
(default, '2018-05-08', '2018-05-10', '2017-11-02 21:30:00', '1579.98', '1' , '1','12'),
(default, '2018-05-04', '2018-05-07', '2017-11-12 21:12:11', '420.42', '1' , '1','2'),
(default, '2018-05-04', '2018-05-05', '2017-12-12 21:43:00', '548.94', '3' , '1','3'),
(default, '2018-04-30', '2018-05-01', '2017-11-30 21:34:10', '531.00', '3' , '1','4'),
(default, '2018-05-20', '2018-05-25', '2017-12-31 21:54:00', '2713.75', '3' , '1','5'),
(default, '2018-05-15', '2018-05-16', '2017-12-14 11:12:02', '1998.26', '2' , '1','6'),
(default, '2018-05-16', '2018-05-17', '2017-11-14 23:10:01', '837.33', '3' , '1','7'),
(default, '2018-05-17', '2018-05-19', '2017-11-06 07:31:01', '1886.42', '1' , '1','8'),
(default, '2018-05-11', '2018-05-14', '2017-11-05 11:51:10', '187.21', '1' , '1','9'),
(default, '2018-01-11', '2018-01-14', '2017-11-12 21:51:11', '1167.00', '1' , '2','1'),
(default, '2018-01-11', '2018-01-13', '2017-11-21 22:41:12', '1926.00', '2' , '3','1'),
```

```
insert into Tipo_Reserva

(id_Tipo_Reserva, Tipo_Quarto, Opcao_Estadia, Reserva_id_Reserva)

values

(default, 'Quarto Duplo Deluxe', '3','1'),
(default, 'Quartos Duplos Superior/Quartos Duplos Superior', '1/1','2'),
(default, 'Quarto Duplo Deluxe', '2','3'),
(default, 'Quarto Individual', '3','4'),
(default, 'Quarto Duplo Superior com Vista/Quarto Duplo Superior com Vista', '2/2','5'),
(default, 'Suite Junior/Suite Junior', '3/3','6'),
(default, 'Suite Executiva/Suite Executiva/Suite Executiva', '3/3/3','7'),
(default, 'Quarto Duplo Deluxe com acesso ao Spa', '1','8'),
(default, 'Suite Familiar', '2','9'),
(default, 'Suite Junior com acesso ao Salão Executivo/Suite Junior com acesso ao Salão Executivo', '1/1','10'),
(default, 'Quarto Duplo ou Twin com Cama Extra', '2','11'),
(default, 'Quarto Duplo', '3','12'),
(default, 'Quarto Triplo Clássico', '2','13'),
(default, 'Apartamento de luxo com Quarto e Vista', '3','14'),
(default, 'Oferta Romântica', '3','15'),
(default, 'Quarto Duplo Superior com Vista/Quarto Duplo Superior com Vista/Quarto Duplo Superior com Vista', '1/1/1','16'),
(default, 'Quarto Duplo Superior/Quarto Duplo Superior/Quarto Duplo Superior', '1/1/1','17'),
(default, 'Quarto Duplo ou Twin Deluxe /Quarto Deluxe Premium/Quarto Duplo ou Twin Clube com acesso ao Salão', '2/2/2','18'),
(default, 'Quarto Deluxe com acesso ao Spa e Acesso Wi-Fi Gratuito/Quarto Deluxe com acesso ao Spa e Acesso Wi-Fi Gratuito', '3/3','19'),--
(default, 'Quarto Duplo Superior com Vista Interna/Quarto Duplo Superior com Vista Interna/Quarto Duplo Superior com Vista Interna', '1/1/1','20'),
(default, 'Suite Presidencial', '3','21'),
(default, 'Quarto Duplo Superior ', '1','22'),
(default, 'Quarto Duplo Deluxe ', '3','23'),
(default, 'Suite Junior/Quarto Duplo Deluxe ', '3/3','24'),
```

Figura 55 - Exemplos do Povoamento da base de dados

5.9. Validação de transações com o utilizador

Em relação às transações definidas no desenvolvimento do modelo conceptual, conseguimos criar esses mesmo elementos durante a elaboração do modelo físico, sendo que o sistema que desenvolvemos tem a capacidade de fazer aquilo que o utilizador precisar da base de dados.

6. Conclusões e trabalho futuro

Em modo de conclusão, é possível fazer uma retrospetiva do trabalho feito desde o inicio.

Inicialmente, deparámo-nos com algumas dificuldades na escolha do tema, de modo a que este fosse original e complexo e conseguisse levar a uma melhor compreensão do Sistema de Gestão de Bases de Dados.

Inicialmente, tivemos que proceder ao levantamento de requisitos, conseguidos através de entrevistas, e validados por eventuais clientes da nossa base de dados.

No que diz respeito ao modelo conceptual, a nossa maior dificuldade foi reconhecer os atributos multivalorados, pois no início estávamos a considerar a alguns deles como entidades. Após uma discussão mútua e pesquisa, decidimos que o melhor caminho a adotar seria mesmo tomá-los como atributos multivalorados.

Como um dos nossos objetivos era que o nosso trabalho fosse original e complexo, decidimos criar 3 “tipos de preço”, variáveis consoante a opção desejada por cada cliente para a sua estadia (sem nenhuma refeição incluída (preço 1), apenas com o pequeno almoço incluído (preço 2) e com todas as refeições incluídas (preço 3)). Para além disto, também existem dois tipos de quartos na nossa base de dados, o que aumenta a complexidade e aumenta o leque de escolha.

De seguida, já com o modelo lógico concluído, fizemos o povoamento da nossa base de dados, diferentes entre si, para ilustrar de uma maneira mais eficiente o que se passa na realidade.

Depois disto, prosseguimos com a elaboração de queries. Neste capítulo também tivemos algumas dificuldades, na elaboração de algumas transações e triggers, visto que a maior parte das que propusemos inicialmente, conseguimos concluir.

Concluindo, achamos que tivemos uma prestação positiva neste projeto considerando os nossos objetivos iniciais, e ficámos com um conhecimento mais ampliado e clarificado sobre os sistemas de gestão de bases de dados.

7. Bibliografia

[1] Thomas M. Connolly and Carolyn E. Begg, *Database Systems: A practical approach to Design, Implementation and Management*, 6th Ed. New York, USA: Pearson, 2014.