**1. Derivar relações a partir do modelo lógico**

Começa-se por derivar as relações presentes no modelo lógico de forma a apresentar as entidades, relações e atributos. Começa-se por especificar o nome da relação, seguida pela lista dos seus atributos, identificando a(s) chave(s) primária(s) e estrangeira(s). Identificando uma chave estrangeira, **(ACABAR TEXTO DEPOIS)**

Apenas temos entidades fortes, relações um-para-muitos (1:N) e atributos multi-valor.

|  |
| --- |
| **Cliente** (*ID*, Email, Nome)  **Chave Primária** *ID* |
| **Reserva** (*ID*, *Preço*, Cliente, Data)  **Chave Primária** *ID*  **Chave Estrangeira** Cliente **referencia** Cliente(*ID*)  **Derivado** *Preço* (Bilhete.*Preço* \* (1 - Bilhete.Desconto)) |
| **Bilhete** (*ID*, *Lugar*, Viagem, *Classe*, *Preço*, Reserva)  **Chave Primária** *ID*  **Chave Estrangeira** Viagem **referencia** Viagem(*ID*)  **Chave Estrangeira** Reserva **referencia** Reserva(*ID*) |
| **Viagem** (*ID*, DataPartida, Duração, *PreçoBase*, *Comboio*, Origem, Destino)  **Chave Primária** *ID*  **Chave Estrangeira** *Comboio* **referencia** *Comboio*(*ID*)  **Chave Estrangeira** Origem **referencia** *Estação*(*ID*)  **Chave Estrangeira** Destino **referencia** *Estação*(*ID*) |
| ***Comboio*** (*ID*, Capacidade, *Lugar*)  **Chave Primária** *ID* |
| ***Estação*** (*ID*, *Localidade*, *País*)  **Chave Primária** *ID* |
| ***Lugar*** (Nr, *Comboio*)  **Chave Primária** Nr  **Chave Estrangeira** *Comboio* **referencia** *Comboio*(*ID*) |

**2. Validar relações usando a normalização**

PARTE DO JOÃO (O PALHAÇO)

**3. Validar relações contra transações do utilizador**

**Transação:** adicionar um novo Cliente

Insere-se (caso não existam) um Email e um Nome na tabela Cliente. Como o *ID* (*PK*) do Cliente está na tabela Reserva como Cliente (*FK*), consegue-se associar a reserva a um cliente.

**Transação:** criar uma reserva

As tabelas Reserva e Bilhete estão associadas entre si através de, respetivamente, *ID* (*PK*) e Reserva (*FK*). Sempre que é adicionada uma entrada à tabela Reserva, é guardada a data do dia em Data. Como Reserva e Bilhete estão relacionados, *Preço* na tabela Reserva será igual à soma do valor *Preço* das entradas na tabela Bilhete com Reserva igual ao *ID* na tabela Reserva.

**Transação:** adicionar um bilhete a uma reserva

Bilhete tem *ID* (*PK*), tem *Viagem* (*FK*) que permite relacionar-se com *Viagem* através de *ID* (*PK*). Cada entrada na tabela Bilhete tem uma *Reserva* (*FK*), um *Preço* e uma *Classe*. Já que *Bilhete* e *Viagem* estão relacionados, *Preço* na tabela Bilhete será igual ao valor de *PreçoBase* da tabela *Viagem*, onde se aplica um desconto a partir de *Classe*. A tabela *Viagem* também tem uma *DataHoraPartida* que indica o dia e a hora em que se realiza a viagem, a duração (*Duração*) e o preço completo, sem descontos, da viagem (*PreçoBase*). *Comboio*, *Origem* e *Destino* são chaves estrangeiras (*FK*) desta tabela.

Viagem relaciona-se, através de Origem (*FK*) e Destino (*FK*), com o *ID* (*PK*) da tabela *Estação* que é identificada pela sua localidade (*Localidade*) e pelo seu país de origem (*País*).

Viagem também se relaciona com o *ID* (*PK*) de *Comboio*. Por sua vez, *Comboio* tem lugares que se relacionam através do seu *ID* (*PK*) e *Comboio* (*FK*) da tabela *Lugar*. Esta última tabela corresponde ao número de cada um dos lugares para um comboio.

**4. Verificar restrições de integridade**

* **Dados Necessários**

Como é necessário que todas as colunas de todas as tabelas tenham um valor associado, ativou-se a opção *NOT NULL* em cada uma.

* **Domínio das Restrições dos Atributos**

De acordo com o que foi descrito na validação do Modelo Conceptual, tem-se para cada atributo o tipo de dados apropriado segundo o seu contexto. Por exemplo, para datas foi colocado um *DATETIME* ou *DATE*, para inteiros um *INT*, para a duração da viagem um *TIME*, para preços um *DECIMAL* com duas casas decimais e tamanho máximo 8 algarismos e para as *Strings* utilizou-se um *VARCHAR* de tamanho variável.

* **Cardinalidade**

Tal como foi descrito na validação do Modelo Conceptual, traduziram-se de forma direta a cardinalidade das relações entre entidades no Modelo Conceptual para o Modelo Lógico.

* **Integridade das Entidades**

Atendendo ao que foi referido na validação do Modelo Conceptual e no ponto “Dados Necessários”, todas as colunas têm a opção *NOT NULL* ativada, incluindo as chaves primárias (*Primary Keys*).

* **Integridade referencial**

Todas as tabelas no nosso SBD estão relacionadas entre si, sendo por isso necessário que, ao fazer-se uma nova inserção numa tabela, que os dados referentes a outras tabelas já existam nessas mesmas tabelas. Por exemplo, quando se adiciona um bilhete – e o número da respetiva reserva–, é necessário que também exista na tabela *Reserva* uma entrada com *ID* igual.

Do mesmo modo, quando se alteram dados que são referenciados ou que referenciam uma tabela, é importante que haja consistência referencial.

* **Restrições gerais**

Será necessário garantir que os campos referentes aos preços (*Preço* na *Reserva*, *Preço* no *Bilhete* e *PreçoBase* na *Viagem*) são superiores a zero. A capacidade de um comboio (*Capacidade* em *Comboio*) também tem que ser superior a zero. Para que se verifique isto, ativou-se a opção *UNSIGNED DATA TYPE* em cada uma destas colunas.

Na *Viagem*, para uma mesma linha, os conteúdos de *Origem* e *Destino* têm que ser diferentes um do outro e o valor de *Duração* tem que ser superior a zero.

**5. Rever modelo lógico com o utilizador**

O objetivo principal deste modelo é permitir efetuar reserva de bilhetes. Como se pode verificar no passo 3, é possível fazer isso utilizando todos as entidades presentes neste modelo. Ou seja, o nosso modelo cumpre com aquilo que lhe é pedido. Podemos assim dizer que o nosso modelo é uma representação correta do que é pedido.

**6. Juntar modelos lógicos num modelo global (opcional)**

Como só temos um perfil de utilizador, não é necessário efetuar este passo.

**7. Verificar futuro crescimento**

O modelo lógico apresentado encontra-se preparado para um crescimento futuro, visto que não está desenhado apenas pensando nos requisitos atuais, sendo também capaz de abranger possíveis alterações futuras, sejam elas um aumento no número de clientes, a criação de novas estações e de novas viagens ou, por exemplo, uma redefinição dos preços dos bilhetes. Além disso, o sistema encontra-se normalizado e sem redundâncias o que facilitará a manutenção e desenvolvimento futuros.