**I**nstituto **S**uperior de **E**ngenharia de **L**isboa

Licenciatura em Engenharia Informática e de Computadores

Semestre de Verão 2016/2017

**Sistemas de Informação II**

**Primeira fase do trabalho prático**

LI51D

**Trabalho elaborado por Grupo 3:**

Eduardo António N.º 40686

Hugo Carvalho N.º 36891

**7 Maio 2017**

**Índice**

[Introdução 3](#_Toc481964313)

[1 – Modelo EA 4](#_Toc481964314)

[2 – Modelo Relacional 5](#_Toc481964315)

[3 – Código T-SQL 7](#_Toc481964316)

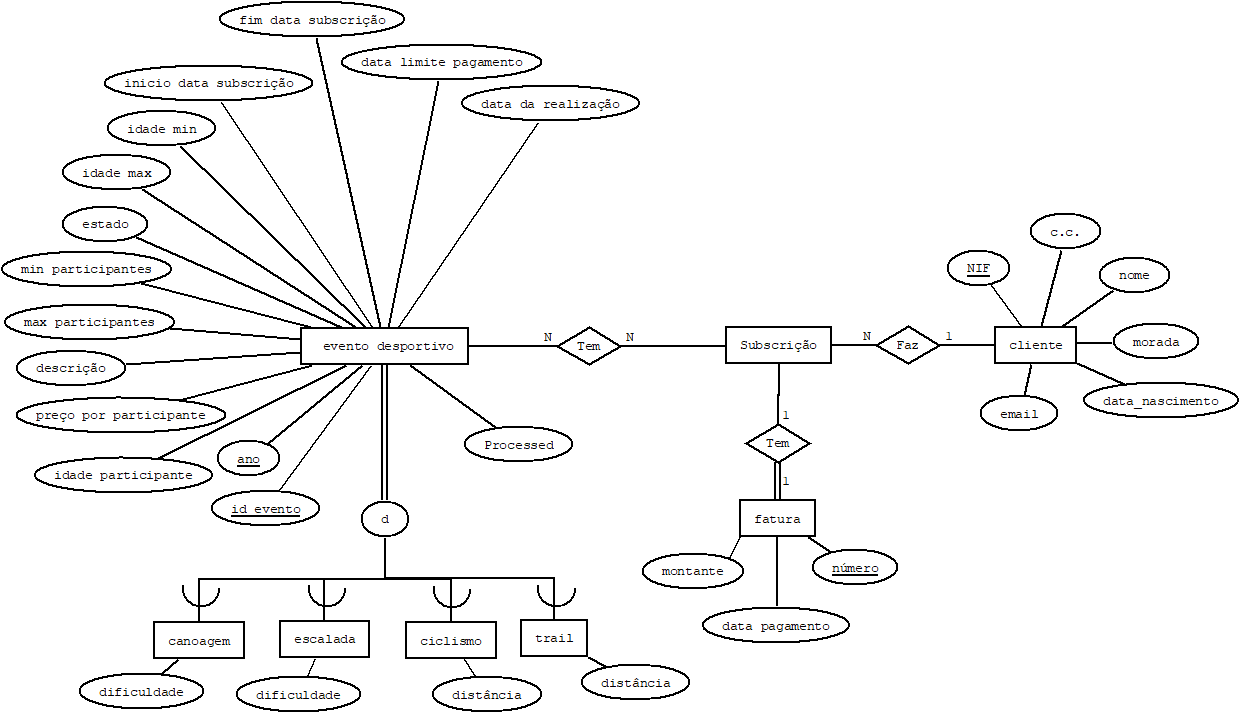
[Conclusão 11](#_Toc481964317)

Introdução

Nesta primeira fase do trabalho era pedido que se implementasse um sistema de informação para a empresa *SoAventura*.

Para isso, é necessária a elaboração de um modelo de dados conceptual e relacional – com as restrições de integridade que forem necessárias – para depois passar à criação do modelo físico usando os conhecimentos adquiridos em SI2, como por exemplo: controlo transacional, procedimentos armazenados ou funções.

1 – Modelo EA

Em baixo apresenta-se o modelo de dados conceptual elaborado de acordo com requisitos do sistema.

2 – Modelo Relacional

**Cliente(NIF, CartaoCidadao, Nome, Morada, data\_nascimento, email, existente)**

PK: {NIF}

**Fatura(Id\_Evento,NIF,ano, Data\_Pagamento, Montante, nome, morada, desc)**

PK: {Id\_Evento,NIF,ano}

FK: {Id\_Evento,NIF,ano} ref. Subscrição

**Subscrição(Id\_Evento, ano, NIF)**

PK: {Id\_Evento, ano, NIF}

FK: {Id\_Evento, ano} ref. Evento\_Desportivo

FK: {NIF} ref. Cliente

**Evento\_Desportivo(Id\_Evento, ano, idadeParticipante, preco\_por\_participante, descrição, max\_participantes, min\_participantes, estado, idade\_max, idade\_min, inicio\_data\_subscrição, fim\_data\_subscrição, data\_limite\_pagamento, data\_da\_realização, horaDaRealização)**

PK: {id\_evento, ano}

**Canoagem(id\_evento, ano, dificuldade)**

FK: {id\_evento,ano} ref. Evento

PK: {id\_evento,ano}

**Escalada(id\_evento, ano, dificuldade)**

FK: {id\_evento,ano} ref. Evento

PK: {id\_evento,ano}

**Ciclismo(id\_evento, ano, distância)**

FK: {id\_evento,ano} ref. Evento

PK: {id\_evento,ano}

**Trail(id\_evento, ano, distância)**

FK: {id\_evento,ano} ref. Evento

PK: {id\_evento,ano}

**MailsEnviados(NIF, email, msg)**

PK: {NIF}

**Restrições de Integridade:**

**RI1:** O Evento\_Desportivo.estado só pode tomar os valores seguintes: “em subscrição”, “subscrito”, “concluído” e “cancelado”.

**RI2:** Para um evento no estado de “subscrito”, uma vez terminado o prazo de pagamento, as respetivas subscrições devem ser marcadas com indicação de que o pagamento foi ou não efetuado e do montante pago.

**RI3:** O campo “dificuldade” em Canoagem e Escalada só podem ter valores entre 1 e 5.

**Ri4:** O campo “distancia” em Ciclismo e Trail tem de ter valores maior que zero.

**RI5:** O campo “existente” do Cliente só pode ser “T” ou “F”.

3 – Código T-SQL

**a) Criar o modelo físico;**

**b) Remover o modelo físico;**

* Para a criação e remoção do modelo físico foram criados os scripts SQL *CREATE.SQL* e *REMOVE.SQL*.
* As tabelas são criadas na base de dados “*SoAventura*”. O código garante a criação dessa base de dados se ainda não existir.
* De seguida, são criadas as tabelas de acordo com o modelo de dados elaborado. Tal como no ponto anterior, é primeiro verificado se as tabelas já existem ou não antes de as criar.
* Por várias vezes é usado a restrição CHECK de modo a garantir que as Restrições de Integridade são cumpridas – por exemplo, garantir que o “estado” do *Evento\_Desportivo* é apenas um dos valores que os requisitos exigem (“em subscrição”, “subscrito”, “concluído” e “cancelado”), ou que o nível de dificuldade no *Evento\_Desportivo* se situa entre 1 e 5.

**c) Inserir, remover e atualizar informação de um cliente;**

* O script C.SQL tem três procedimentos armazenados para inserir (InsertCliente), remover (DeleteCliente) e atualizar (UpdateCliente) cliente.
* Na atualização é primeiro verificado se o NIF existe e que o campo “existente” não é “F” antes de efetuar o UPDATE.
* Na remoção é lançado um erro se o NIF for NULL ou se o NIF não existir na tabela Cliente.

**d) Inserir, remover e atualizar informação de um evento;**

* O script D.SQL tem vários procedimentos armazenados para a criação, remoção e atualização tanto para os Eventos, bem como para cada tipo de evento – Canoagem, Escalada, Ciclismo e Trail.
* Na criação de um evento, o ID é incrementado do valor que já exista na tabela – por outro lado, se não existir nenhum evento o ID é colocado a zero.
* Na atualização e remoção de um evento é verificado se o ID existe.
* Já para as que dependem de Evento (Canoagem, Escalada, Ciclismo e Trail) é necessário que seja removido nessas tabelas antes de remover no Evento.

**e) Realizar a subscrição de um evento por parte de um cliente existente;**

* Ao executar o trigger SubscreverClienteEvento é garantido duas coisas necessárias antes de fazer a associação na tabela Subscrição: que o evento e cliente que se pretende subscrever existem.
* Também deverá lançar erro se:
  + O Evento já estiver concluído ou cancelado.
  + O número de subscritos for maior que o permitido (max\_participantes).
  + A idade do cliente a subscrever for maior do que o permitido (idade\_max).
  + A data for inválida.

**f) Proceder ao pagamento de uma subscrição;**

* O trigger *PagarSubscricao* garante que antes de inserir os dados na tabela Fatura, é garantido que:
  + O evento e cliente são válidos;
  + O cliente está subscrito num evento; (ver e) )
  + A data especificada não excede a data limite especificada no evento;
  + O montante a pagar é suficiente (não inferior ao *preco\_por\_participante* do Evento);
  + O pagamento não é efetuado se o estado do evento for “cancelado” ou “concluído”.
* Por fim, é inserido na tabela Fatura, em que o ID da fatura é obtido com recurso a uma seleção à tabela Fatura para obter o próximo ID.

**g) Transitar todos os eventos de estado, em função da data corrente;**

* Atualiza para o estado *“em subscrição”* os eventos em que a data de inicio de subscrição já tenha ultrapassado a data atual (*GETDATE()*).
* Atualiza para o estado *“concluído”* os eventos em que a data da realização já tenha ultrapassado a data atual (*GETDATE()*).
* Para os que estão no estado *“em subscrição”*:
  + Se a data do campo “fim\_data\_subscrição” já tiver sido ultrapassada, atualizar para o estado *“cancelado”*.
  + Se o número de participantes for superior ao número mínimo, atualizar para o estado *“subscrito”*.

**h) Enviar avisos por email a todos os clientes inscritos em eventos que se irão realizar num intervalo de tempo (em dias) indicado, a contar da data corrente;**

* Foram criados dois triggers: SendMail e EnviarMailIntervalo.
* *SendMail* insere na tabela *MailsEnviados* se o e-mail não for nulo e se o cliente for válido (ou seja, se o NIF existir na tabela Cliente).
* O objetivo do trigger *EnviarMailIntervalo* é avisar os clientes dos eventos que se irão realizar num intervalo de dias indicado. Para isso, é criado um ciclo com inicio no dia atual (GETDATE()) até *X* dias depois, o qual depois chama *SendMail* para adicionar.
* Por fim, é atualizada o valor *Processed* de modo a sinalizar que determinado evento já foi processado.

**i) Listar todos os eventos cancelados, agrupados por tipo, num dado intervalo de datas;**

* Retorna uma tabela com o número de eventos cancelados por tipo (Canoagem, Escalada, Ciclismo e Trail) em que o campo estado seja “cancelado” e que o intervalo de datas esteja no intervalo especificado.

**j) Listar todos os eventos com lugares disponíveis para um intervalo de datas especificado;**

* Retorna uma tabela com o ID dos eventos e do ano dos eventos disponíveis.
* Serão listados todos os eventos em que o número de participantes máximo seja superior aos subscritos para o intervalo de datas definido.

**k) Obter os pagamentos realizados num dado ano com um intervalo de amostragem especificado;**

* Recebe como parâmetros o ano e o montante mínimo e máximo.
* Retorna uma tabela com o Evento do ID, ano, NIF e montante da tabela Fatura com as condições especificadas anteriormente.

Conclusão

Terminada esta primeira fase do trabalho, conclui-se:

* Foi possível realizar o modelo de dados conforme os requisitos pedidos no enunciado, usando restrições de integridade que no modelo conceptual não é possível assegurar;
* Criou-se o modelo físico – código SQL tanto para a criação como para a remoção do modelo físico, tendo em conta o modelo de dados elaborado;
* Foi necessário usar técnicas de modo a garantir a boa execução do código – como por exemplo transações, procedimentos armazenados e níveis de isolamento.