

1º Trabalho Prático

## Sistemas de informação II

Licenciatura Engenharia Informática e de Computadores

**Docente:** Afonso Remédios

**Grupo 7:**

Pedro Miranda - 43584

Ricardo Rodrigues – 43594

Índice

[Introdução 3](#_Toc92882865)

[Modelo Entidade-Associação 4](#_Toc92882866)

[Modelo Relacional 5](#_Toc92882867)

[Avaliação Experimental 6](#_Toc92882868)

[Exercício A 6](#_Toc92882869)

[Exercício B 6](#_Toc92882870)

[Exercício C 6](#_Toc92882871)

[Exercício D 6](#_Toc92882872)

[Exercício E 6](#_Toc92882873)

[Exercício F 6](#_Toc92882874)

[Exercício G 7](#_Toc92882875)

[Exercício H 7](#_Toc92882876)

[Exercício I 7](#_Toc92882877)

[Exercício J 7](#_Toc92882878)

[Exercício K 7](#_Toc92882879)

[Conclusão 8](#_Toc92882880)

[WebGrafia 9](#_Toc92882881)

# Introdução

No enunciado deste trabalho é proposta a resolução de um problema através da manipulação de uma base de dados. Para tal, será necessária a concretização dos modelos relacional e entidade-associação, bem como a criação e inserção de dados nas respetivas tabelas e a utilização do mecanismo transacional de bases de dados.

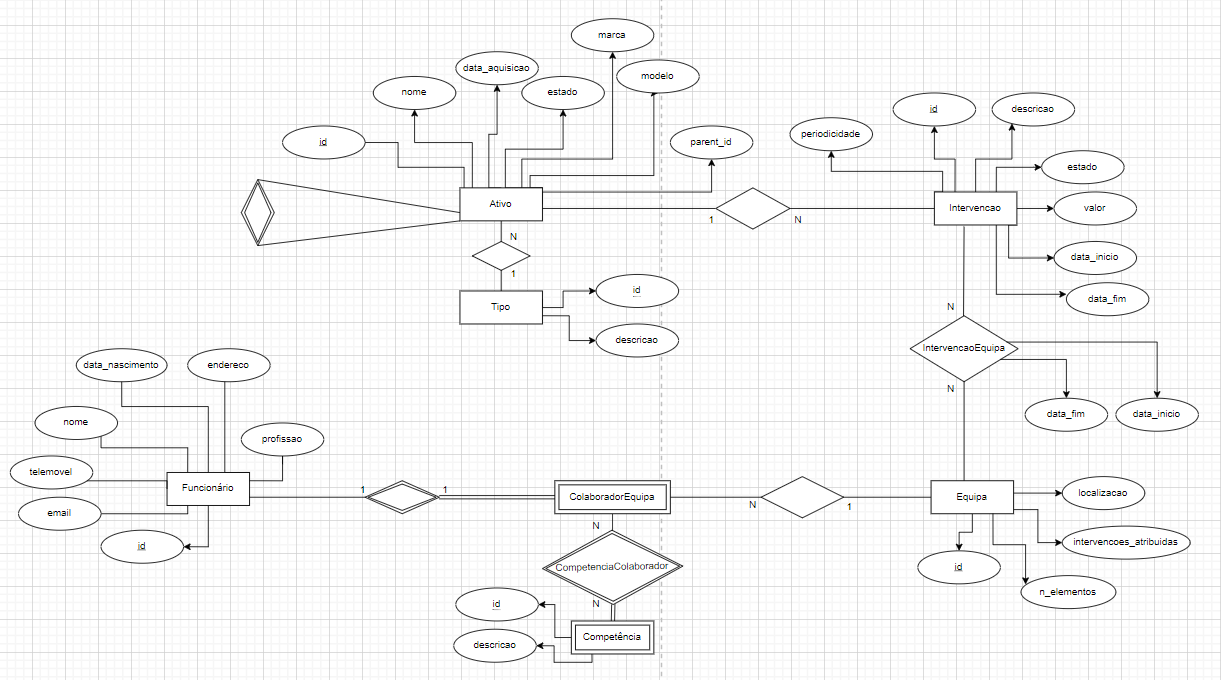
O mecanismo transacional tem um controlo que permite executar várias interações com a base de dados ao mesmo tempo, utilizando diferentes transações. Uma transação é um conjunto de instruções que irão ser usadas numa base de dados. Para que estas sejam compatíveis, de forma a poderem ser executadas em simultâneo, terão de ser atómicas. Para não corromperem a base de dados, estas também têm de ser consistentes. Isto é, isoladas uma das outras de forma a garantir resultados perseverantes, tanto no caso de um *commit* como de um *rollback*.

As transações têm como objetivo preservar a integridade e a consistência de uma base de dados, sendo estas mecanismos de recuperação em caso de falhas do sistema. Os escalonamentos permitem-nos utilizar várias transações e executá-las na mesma ordem. Um conflito num escalonamento acontece quando as operações pertencem a transações diferentes. Estes conflitos geram anomalias, ocorrendo quando uma transação consegue ler dados resultantes de escritas que ainda não sofreram um *commit* por outra transação. A leitura não repetitiva (*Non Repeatable Read)*, isto é, quando uma leitura demonstra resultados diferentes, após ser executada novamente, mais tarde na mesma transação. Por último temos a “leitura fantasma” (Phantom Read), que ocorre quando em duas leituras diferentes de uma transação, há uma alteração nos registos.

De modo a evitar estas anomalias são usados níveis de isolamento. Começando pelo nível de isolamento mais “baixo”, isto é, o *Read Uncommitted,* que permite que uma transação manipule valores que não são sujeitos a *commit* por outras transações. O nível seguinte é o *Read Committed*, que deixa que uma transação leia e manipule dados que já sofreram *commit* por outras transações. De seguida temos o *Repeatable Read*, que garante que a mesma leitura de um dado se repita, originando resultados iguais para diferentes execuções na mesma transação. Para finalizar, o nível mais alto e seguro é o *Serializable* que, apesar de ser semelhante ao anterior, este garante que as linhas selecionadas por uma transação não podem ser alteradas/lidas por outra transação, até que a primeira transação seja concluída.

# Modelo Entidade-Associação

Figura 1- Modelo Entidade-Associação



# Modelo Relacional

**CE:** Chave estrangeira

**RI:** Restrições de Integridade

**Ativo** (id, parent\_id, nome, data\_aquisicao, marca, modelo, localizacao, estado, id\_tipo)

**CE:** id\_tipo referencia Tipo(id)

**RI1:** todos os campos, com exceção de marca e modelo, são obrigatórios

**RI2:** O estado só pode tomar 2 valores: ‘0’ (desativado) ou ‘1’ (operacional)

**Funcionario** (id, email, telemovel, nome, data\_nascimento, endereco, profissao)

**Tipo** (id, descricao)

**Equipa** (id, localizacao, n\_elementos, intervencoes\_atribuidas)

**RI3:** intervencoes\_atribuidas pode ter, no máximo, o valor 3

**Intervencao** (id, descricao, estado, valor, data\_inicio, data\_fim, periodicidade, ativo\_id)

**CE:** ativo\_id referencia Ativo(id)

**RI4:** descrição pode tomar como valores “avaria”, “rutura” ou “inspeção”

**RI5:** estado pode tomar valores como “por atribuir”, “em análise”, “em execução” ou “concluído”

**RI6:** data\_inicio deverá ser superior a Ativo(data\_aquisicao)

**IntervencaoEquipa** (id\_equipa, id\_intervencao, data\_inicio, data\_fim)

**CE**: id\_equipa referencia Equipa(id)

id\_intervencao referencia Intervencao(id)

**Competencia** (id, descricao)

**ColaboradorEquipa** (id, id\_equipa)

**CE**: id referencia Funcionario(id)

id\_equipa referencia Equipa(id)

**CompetenciaColaborador** (id\_competencia, id\_colaborador, id\_equipa)

**CE**: id\_competencia referencia Competencia(id)

id\_colaborador, id\_equipa referenciam ColaboradorEquipa(id, id\_equipa)

# Avaliação Experimental

A normalização foi feita até à 3ª forma normal, de forma a evitar problemas na gestão da base de dados.

## Exercício A

Para criar o modelo físico foi criado o ficheiro create\_tables.sql, que realiza diversos CREATE TABLE de forma a criar as tabelas tendo por base o modelo EA e o relacional.

## Exercício B

Para apagar o modelo físico, apenas se recorreu à criação de um ficheiro drop\_tables.sql, onde são realizados os diversos DROP TABLESdas várias tabelas.

## Exercício C

Neste exercício criámos um ficheiro populate\_tables.sql, onde foram inseridos dados nas tabelas, de forma a servirem como alvo de teste de alguns dos exercícios.

## Exercício D

No exercício D foram criadas as transações EditPerson, AddNewPerson e RemovePerson, com o objetivo de, como o próprio nome indica, editar, adicionar ou remover uma pessoa da tabela Funcionario.

## Exercício E

Atendendo ao objetivo de retornar o código de uma equipa livre, dada a descrição de uma intervenção, procedeu-se à criação de uma função que verifica quais as equipas que têm menos de três intervenções atribuídas. Neste exercício ficou em falta a verificação da data da intervenção, devendo esta retornar a equipa que teve uma intervenção atribuída há mais tempo.

## Exercício F

Neste exercício foi necessário recorrer à função do exercício E, de forma a poder obter o código de uma equipa disponível para realizar a intervenção, e a uma função auxiliar para gerar o id da intervenção. Caso a função do exercício E retorne null, o estado da intervenção é definido como “por atribuir”. Caso contrário, a intervenção é inserida na tabela Intervencao, bem como na tabela IntervencaoEquipa onde é associada à equipa que a vai realizar. É também atualizado o valor intervencoes\_atribuidas da equipa em questão.

## Exercício G

De forma a poder adicionar-se uma nova equipa, foi criado o *procedure* addNewTeam que tem o auxílio de uma função auxiliar para gerar o id da mesma. Esta é inicializada sem qualquer intervenção atribuída, mas com a localização inserida como parâmetro no *procedure.*

## Exercício H

Atendendo ao que é pedido no exercício H, foi criado um *procedure* updateTeamElements que verifica se o funcionário já pertence a uma equipa. Caso pertença, este é eliminado da base de dados. Caso contrário, este é adicionado. Em ambos os casos, o número de elementos é atualizado na tabela Equipa, assim como as respetivas competências na tabela CompetenciaColaborador.

## Exercício I

A função ListInterByYear recebe um ano e produz a listagem de intervenções que tiveram como data\_fim o ano recebido como parâmetro. Esta listagem é feita através de uma tabela com o id da intervenção e a descrição da mesma.

## Exercício J

O *procedure* UpdateInterStatus recebe o código da intervenção e o novo estado, que é inserido na mesma. Inicialmente é feita a verificação de que esta intervenção existe de facto na tabela Intervencao, caso não exista é mostrada uma mensagem que erro informando o utilizador que a intervenção não existe.

## Exercício K

A *view* ResumoInter selecionará todos os atributos de Ativo e Intervencao, onde o id do ativo é o mesmo. Através de um *trigger*, é analisado se o *update* que está a tentar ser efetuado se pode realizar, lançando um erro em caso contrário.

# Conclusão

Considerando o problema exposto, foi-nos dado o conhecimento de como criar, manipular utilizar a gestão transacional numa base de dados. Recorremos a conhecimento adquirido anteriormente relativo a sistemas de informação, bem como a escalonamentos teóricos de forma a conseguirmos prever que anomalias teríamos de evitar em cada caso. Para evitar a existência das mesmas, foi aprendida a utilidade de recorrer a níveis de isolamento e os seus impactos positivos e negativos a nível computacional, na gestão de uma base de dados.

# WebGrafia

* Relatório do trabalho
* Slides da disciplina do moodle
* [Transact-SQL Reference (Database Engine) - SQL Server | Microsoft Docs](https://docs.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-ver15)
* [Microsoft PowerPoint - bd2\_processamento-de-transacoes.ppt [Modo de Compatibilidade] (ufop.br)](http://www.decom.ufop.br/guilherme/BCC441/geral/bd2_processamento-de-transacoes.pdf)
* [bd\_transactions (up.pt)](https://www.dcc.fc.up.pt/~edrdo/aulas/bd/teoricas/bd_transactions.pdf)