Bases de Dados
2023/2024

Licenciatura em Engenharia Informática

TECNOLOGIA SETÚBAL
ESCOLA SUPERIOR POLITÉCNICO SETÚBAL

Enunciado de Projeto (2ª Fase)

Leia o enunciado com atenção!
Quaisquer dúvidas sobre o mesmo deverão ser colocadas no respetivo fórum no Moodle.

1 Introdução

A 2ª Fase do projeto incidirá sobre a codificação em SQL do modelo de base de dados relacional especificado. O projeto incide na implementação da base de dados relacional usando o SGBD MySQL, em conformidade com uma abordagem OLTP. Esta fase contempla uma componente de pesquisa (i.e., instruções em SQL) para listar informação sobre os dados armazenados na base de dados. Desenvolver interfaces para a camada de Apresentação (ver Figura 1) está fora do âmbito da unidade curricular (UC) de Bases de Dados. Matéria que será âmbito de estudo em outras UCs do curso da LEI.

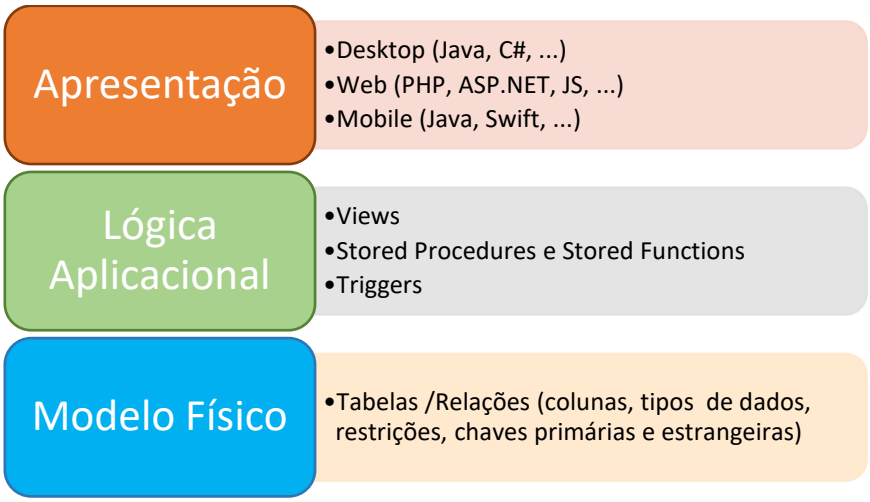

Figura 1. Arquitetura do problema – modelo das três camadas.

O objetivo é desenvolver o conjunto de comandos SQL para aceder e processar dados armazenados na base de dados OLTP. Para além da definição e codificação de consultas (*queries*), pretende-se que sejam desenvolvidos procedimentos e mecanismos capazes de:

- Manter a integridade referencial da base de dados,
- Dar suporte a funcionalidades básicas no acesso aos dados (e.g., através de uma API ao nível dos dados).

Por exemplo, para criar um novo registo de utilizador na base de dados, em vez de invocar diretamente o comando INSERT INTO (DML), instrução mais difícil de programar e com risco de *SQL-Injection*; deveria ser criada a possibilidade de invocar um **procedimento** desenvolvido para o efeito (e.g., **sp_insert_user**(...)) mitigando risco quer no acesso a dados confidenciais ou rico que possa comprometer a qualidade dos dados. O mesmo princípio aplica-se a consultas disponibilizadas através de ***Views***.

Dada a multiplicidade de variantes temáticas escolhidas pelos grupos, os requisitos a desenvolver são aqui especificados de uma forma genérica e deverão ser adaptados pelos elementos da equipa de projeto, de acordo com o domínio do problema que foi específico/modelado na 1ª Fase. Ter em atenção que um problema específico poderá levantar a necessidade de implementar um ou mais procedimentos complementares de modo a completar a lógica aplicacional a desenvolver nesta fase.

1.1 Objetivos específicos

- **Rever** e **melhorar** o modelo da base de dados no seguimento da discussão do trabalho (1ª Fase);
- **Criar** a base de dados relacional (tipologia OLTP) e proceder ao carregamento inicial de dados (i.e., dados de teste para validar as regras de negócio/requisitos especificados, pela equipa de projeto, no âmbito do domínio do problema proposto);
- **Definir** e **implementar** consultas (*Queries*), *Views, Functions, Stored Procedures* e *Triggers;*
- **Disponibilizar** um *script* para testar as funcionalidades implementadas.

1.2 Regras

- O trabalho deverá basear-se na submissão efetuada na 1ª fase, mantendo o tema e a constituição dos elementos da equipa de projeto;
- As equipas de projeto são incentivadas a melhorar a qualidade do trabalho desenvolvido na 1ª Fase. As melhorias incorporadas obrigatoriamente têm de ser evidenciadas e descritas de forma objetiva e verificável. O não cumprimento deste requisito terá impacto na avaliação.
- Para a 2ª Fase será disponibilizado um Template de relatório (entregável).

2 Requisitos mínimos

O projeto consiste em desenvolver uma base de dados de suporte para uma aplicação de **Gestão Integrada de Atividades Desportivas** cujo âmbito foi definido na 1ª Fase.

Requisitos obrigatórios: Os *scripts* têm de ter a estrutura listada nos tópicos abaixo, nomeadamente texto explicativo sobre o propósito do código implementado. O texto explicativo deve ser adicionado como comentários técnicos, escritos de forma sucinta e com objetividade. O não cumprimento (ou cumprimento parcial) tem impacto na avaliação.

1.1 **Criar a base de dados –** (ficheiro: **create.sql**)
- Criar a base de dados e todas as estruturas de suporte indicando os tipos de dados de cada coluna, as restrições associadas às colunas ou tabelas e reforçando a integridade referencial, sempre que for possível e aconselhável.

1.2 **Componente lógica –** (ficheiro: **logic.sql**)
- Disponibilizar, pelo menos, 5 *views*;
- Disponibilizar, pelo menos, 2 *stored functions*;
- Disponibilizar, pelo menos, 5 *stored procedures*;
- Disponibilizar, pelo menos, 2 *triggers*.

1.3 **Carregamento inicial de dados na base de dados** – (ficheiro: **populate.sql**)
- Carregar a base de dados com um mínimo de 20 registos em cada tabela. Excetuam-se tabelas onde tal seja manifestamente irrealista ou inadequado (e.g., tbl_estado_civil, tbl_genero).
- Para este requisito devem recorrer a instruções **DML** + utilização de **Stored Procedures** de INSERT e/ou UPDATE para as tabelas que foram desenvolvidas e a utilização na fase de carregamento seja pertinente.

1.4 **Consultas à base de dados –** (ficheiro: **queries.sql**)
- Implementar consultas à base de dados (ver tópico sobre consultas).

1.5 **Registo de resultados** – (ficheiro: **results.sql**)
- As tabelas relacionadas com o registo de resultados das provas/atividades desportivas deverão ser preenchidas e manipuladas através de um *script* de teste/demonstração onde seja ilustrado o funcionamento dos **stored procedures** que deverão ser implementados para tal.

1.6 **Teste de Triggers** – (ficheiro: **test_triggers.sql**) (ver tópico sobre triggers).
- Criar um *script* para testar as funcionalidades associadas aos *triggers* implementados.

1.7 **Teste de funcionalidade – (ficheiro: test.sql)**
- Criar um *script* para testar/demonstrar as restantes funcionalidades desenvolvidas.

3 Consultas à base de dados – (Queries, Views e/ou Procedures) (ficheiro: queries.sql)

As consultas deverão apresentar a informação de forma legível, isto é, em vez de mostrar uma coluna com o nome do atributo respetivo (e.g., **user_name**), deverá apresentá-lo de forma compreensível para o utilizador (exemplo: '**Nome Utilizador**').

Os comandos relativos a este ponto devem ser colocados no ficheiro **queries.sql** e, cada um deles, deve ser precedido por um comentário indicando o ponto a que corresponde. Exemplo:

```
-- Q1.1 (número identificador da consulta)
-- Listagem de participantes que competiram nas provas femininas (descrição)
SELECT * FROM Participante WHERE sexo='F' ORDER BY 1,3;
```

Devem ser implementados os comandos SQL que permitem obter os seguintes resultados:

1. Lista de participantes segundo, pelo menos, 2 critérios de consulta:
   1.1. Critério I;
   1.2. Critério II.
2. Lista de equipas e respetivos elementos segundo, pelo menos, 2 critérios de consulta:
   2.1. Critério I;
   2.2. Critério II.
3. Lista de provas do evento segundo dois critérios:
   3.1. Critério I;
   3.2. Critério II.
4. Lista de resultados/classificações em cada prova/evento segundo, pelo menos, 2 critérios:
   4.1. Critério I;
   4.2. Critério II.
5. Lista com o número médio, mínimo, máximo e desvio padrão dos participantes por prova/evento, segundo, pelo menos, 2 critérios:
   5.1. Critério I;
   5.2. Critério II.
6. Lista de resultados de cada prova/evento com ranking das equipas (e.g., classificação da equipa em cada prova/evento);
7. Lista de participantes individuais que não participaram em qualquer prova de equipas, diferenciando participantes federados e profissionais, de participantes casuais (e.g., atleta amador sem estar associado a uma entidade, clube ou federação desportiva);
8. Lista dos elementos de uma equipa com identificação do role de cada elemento na prova/evento e respetiva características de cada elemento.
9. Top 5 das provas com maior número de participantes, agrupada por ano e tendo como base os últimos três anos bem como restrições relacionadas com faixa etária e categoria dos participantes (e.g., federado, profissional, amador);
10. Consulta adicional recorrendo a, pelo menos, 3 tabelas;
11. Consulta adicional recorrendo a, pelo menos, 3 tabelas que inclua WHERE e HAVING;
12. Consulta adicional usando descrições de dados existentes num relacionamento recursivo.

13. Duas consultas adicionais utilizando dois tipos diferentes de subqueries.

4 Registo de resultados – (ficheiro: results.sql)

O registo (inserção, alteração e remoção) de resultados nas provas deve ser feito através de *stored procedures* (**sp**). Cada *stored procedure* deve ser precedido por um comentário indicando o ponto a que corresponde. Exemplo:

```
-- SP1 (número identificador do stored procedure)
-- criar registo de novo prato com origem na zona oeste
CREATE PROCEDURE sp_criar_prato(…)
```

Deverão ser implementados os seguintes *stored procedures*:

1. **sp_criar_prova** – Cria uma nova prova/evento num evento determinado, enviando todos os dados necessários à definição da mesma;
2. **sp_adicionar_participante(id_prova, …)** – Adiciona um atleta à lista de atletas/equipas que irão competir na prova/evento indicada de acordo com o tipo de prova;
3. **sp_registar_resultado(id_prova, id_participante, …)** – Regista o resultado do participante na prova/evento indicada;
4. **sp_remover_prova(id_prova, force, …)** – Remove a prova/evento identificada no parâmetro, nas seguintes circunstâncias:
    a. Caso não existam resultados associados à prova/evento (ou outros registos que sejam dependentes);
    b. Caso existam resultados associados à prova/evento e tenha sido enviado *"True"* no parâmetro **force**;
    c. Caso contrário, devolve um erro.
5. **sp_clonar_prova(id_prova, …)** – Cria uma nova prova/evento com uma cópia de todos os dados existentes na prova/evento indicada como parâmetro. A única exceção é que, à descrição da prova, deverá ser adicionada a string " --- COPIA (a preencher)".

    Exemplo: "100 metros" → "100 metros --- COPIA (a preencher)"

O mecanismo descrito por estes procedimentos pode ser alterado desde que devidamente justificado e validado antecipadamente à entrega pelo docente das práticas.

* estas *procedures* não contam para a contabilização do requisito mínimo de 5 *stored procedures.*

5 Remoção de dados

A remoção de dados deve ser encarada com a sensibilidade de manter a integridade referencial dos dados já existentes. Não devem, por isso, ser utilizadas chaves estrangeiras com a opção **ON DELETE CASCADE**. Como tal, devem ser desenvolvidos os **stored procedures** considerados apropriados para remoção de dados assegurando a integridade referencial.

* estas *procedures* não contam para a contabilização do requisito mínimo de 5 *stored procedures.*

6 Monitorização de falhas

Implemente os seguintes *triggers* que permitem fazer a monitorização de alterações nos resultados:

- **result_change** – Regista na tabela de **tbl_logs** (a criar na base de dados), pelos menos, os seguintes dados:
    - Data e hora da operação;

  - Identificação do atleta/equipa;
  - Identificação da prova/evento;
  - Resultado anterior;
  - Novo resultado;
  - Classificação.
- Outro *trigger* (nome e características a definir pelo grupo) que mantenha um log atualizado com informação sobre os resultados removidos da base de dados.

7 Entregas – *Scripts*

Deverão ser entregues *scripts* com os seguintes propósitos:

NOTA: Ficheiros com erros de sintaxe/execução não serão considerados nem as funcionalidades que, eventualmente, pudessem facultar, caso a sua implementação fosse a correta.

create.sql	Contém a definição/implementação do modelo relacional;
logic.sql	Contém as *views, funções, stored procedures* e *triggers*;
populate.sql	Contém as instruções utilizadas para popular a base de dados (correndo os *scripts* por esta sequência, poderão fazer uso dos *stored procedures*);
queries.sql	Contém as consultas à base de dados;
results.sql	Contém os testes à gestão de informação associada aos resultados das provas;
test_triggers.sql	Contém os testes aos *triggers*;
test.sql	Contém instruções que permitam testar as funcionalidades desenvolvidas (é esperado que o *script* seja o resultado natural dos testes realizados pela equipa de projeto). Este *script* será central na discussão do projeto. Este *script* deve conter chamadas a todas as funcionalidades desenvolvidas, acompanhadas por consultas que permitam verificar a correção do funcionamento das funcionalidades.

8 Relatório

O relatório deverá conter a informação relativa à 1ª Fase, acrescido da descrição das funcionalidades desenvolvidas na 2ª Fase. Qualquer código-fonte e/ou *script* não deve ser colocado no relatório, apenas devem constar nos ficheiros em formato *.sql.

No relatório para a 2ª Fase, devem disponibilizar uma tabela com uma listagem do nome das funcionalidades implementadas e uma descrição sucinta para cada funcionalidade. Na seção Conclusões deverão reportar eventuais constrangimentos/ limitações e/ou o que não foi possível desenvolver.

Recomendamos usar uma notação padrão de forma a uniformizar a leitura das entidades informacionais (Tabelas) e respetivos atributos, designadamente:

- Nome das Tabelas, usar a notação ***PascalCase***
- Nome dos atributos das Tabelas, usar a notação ***camelCase*** precedido de um acrónimo.
- Links para convenções de escrita ***PascalCase vs camelCase***
  - Pascal case vs. camel case: What's the difference?
  - Camel Case vs. Pascal Case — Naming Conventions

Consultar o seguinte link para mais informação sobre boas práticas sobre convenções de escrita:

- MYSQL Naming Conventions, 2019

9 Datas e Entrega

A entrega deverá ser efetuada por apenas um elemento da equipa de projeto, na plataforma Moodle, no link referente à turma do docente das aulas práticas.

Todos os componentes do projeto devem ser entregues num único ficheiro no formato *.ZIP cujo nome deve seguir a seguinte nomenclatura:

nºaluno1_PrimeiroNomeUltimoNome_nºaluno2_PrimeiroNomeUltimoNome_Docente_TurnoLab.zip

Exemplo: **202201234_AntonioSantos_RuteMorais_CS_5F14h.zip**

Em que:

- Nome e apelido: sem acentos/cedilhas e com a primeira letra maiúscula;
- Números de aluno: nºaluno1 < nºaluno2;
- Docente: Iniciais do primeiro e último nome;
- TurnoLab: Dia da semana e hora de início do Laboratório (Exemplo: 2F11h).

O relatório deverá ser submetido em formato PDF juntamente com os *scripts* desenvolvidos (em ficheiros separados), num único ficheiro ZIP com a designação anteriormente indicada, até às 23h55 do **dia 14 de junho de 2024** na plataforma Moodle.

Será aplicada uma penalização de 0.1 valores por cada hora de atraso na submissão. Após as 24h de 14 de junho de 2024 é bloqueada a submissão e o projeto só poderá ser entregue em época de recurso seguindo o respetivo enunciado dessa época.

As discussões terão lugar nos laboratórios das semanas de 17/06 e 27/06 de 2024.

10 Avaliação:

- **Relatório – 10%**
- **Revisão do Modelo e melhorias (e.g., regras de negócio) face ao apresentado na 1ª Fase – 10%**
- **Implementação dos requisitos (45%)**
  - Grau de cumprimento dos requisitos mínimos;
  - Nível de desenvolvimento da lógica do modelo face aos requisitos próprios do projeto;
  - Grau de cumprimento dos requisitos específicos.
- ***Scripts* de teste e demonstração de funcionalidades – 35%**

[Fim do enunciado]