Universidade do Porto
Faculdade de Engenharia
FEUP

PROJETO DE BDAD 16/17

Este projeto tem como objetivos a criação e interrogação de uma base de dados num contexto proposto pelo grupo de trabalho. Para tal, será necessário criar o modelo conceptual para o tema definido, mapear esse modelo para um esquema relacional, implementar o esquema numa base de dados SQLite, introduzir dados e, por fim, interrogar a base de dados.

ENTREGAS

Formação do grupo de trabalho e definição do tema [26 de fevereiro, 23h55]

Entrega I [12 de março, 23h55]

i. **Tarefa**: A
ii. **Entregas**: Relatório em formato pdf com: contexto e diagrama UML
iii. **Avaliação**: 25% da nota do projeto

Entrega II [2 de abril, 23h55]

i. **Tarefas**: B, C, D, E e F
ii. **Entregas**:
   - Relatório em formato pdf com: secções da entrega I, diagrama UML revisto, esquema relacional, análise dependências funcionais e formas normais; lista e forma de implementação das restrições
   - Ficheiros: criar.sql e povoar.sql
iii. **Avaliação**: 25% da nota do projeto

Entrega III [28 de maio, 23h55]

i. **Tarefas**: G e H
ii. **Entregas**:
   - Relatório em formato pdf com: secções da entrega II, listagem das 10 interrogações em linguagem natural; listagem dos 3 gatilhos em linguagem natural
   - Ficheiros: intN.sql (x10), gatilhoN_adiciona.sql (x3), gatilhoN_remove.sql (x3) *e* gatilhoN_verifica.sql (x3)
iii. **Avaliação**: 50% da nota do projeto

FORMAÇÃO DO GRUPO DE TRABALHO

O projeto será realizado em grupos de 3 estudantes. O grupo de trabalho deve ser constituído na atividade "Grupos para o projeto" existente no moodle. O primeiro dígito do identificador do grupo deve ser o número da turma. Por exemplo, o grupo 101 deve ser formado por estudantes da turma 2MIEIC01.

DEFINIÇÃO DO TEMA

O tema deve ser proposto pelo grupo ao docente das aulas teórico-práticas na semana que começa a 20 de fevereiro. Após aprovação do tema, até 26 de fevereiro, o grupo deve enviar, via moodle, uma descrição do seu tema com o máximo de 100 palavras. Esta descrição deve ser similar aos enunciados dos exercícios das aulas teórico-práticas, enunciando as entidades previstas, os seus atributos e a forma como estas se relacionam.

A título indicativo, esperam-se esquemas relacionais com 10 a 15 relações. Algumas destas relações devem ter chaves compostas.

TAREFAS

A. Definição do Modelo Conceptual

Antes da criação do modelo conceptual deve existir uma familiarização com o contexto associado ao tema do trabalho. Nesta etapa pretende-se que os estudantes compreendam com detalhe os dados que terão de ser armazenados na base de dados.

Deve, depois, ser criado um modelo conceptual em UML para a base de dados a criar. Neste modelo devem ser claramente indicadas todas as restrições necessárias e as multiplicidades das associações. Se houver elementos derivados, estes devem ser sinalizados adequadamente.

O diagrama de classes UML deve ser incluído num relatório, onde também deve ser descrito o contexto associado à base de dados e outra informação que possa ser importante para a avaliação do modelo conceptual.

B. Definição do Esquema Relacional

Antes da definição do esquema relacional, o modelo conceptual deve ser revisto com base na avaliação da Entrega I. O diagrama UML revisto deve ser acrescentado ao relatório que foi entregue anteriormente.

O modelo conceptual revisto deve ser mapeado para o esquema relacional, que deve ser acrescentado ao relatório em formato textual utilizando a sintaxe:

R1 (atr1, atr2, atr3->R2)

Espera-se uma indicação clara das chaves, primária e estrangeira(s), de cada relação.

C. Análise de Dependências Funcionais e Formas Normais

Para cada relação deve ser indicado o conjunto de dependências funcionais associado e eventuais violações à Forma Normal Boyce-Codd e 3ª Forma Normal. A não existência de violações deve ser justificada. Esta análise deve também ser acrescentada ao relatório.

D. Criação da base de dados em SQLite

O próximo passo envolve criar a base de dados em SQLite. O SQLite permite ler comandos de um ficheiro. Esta funcionalidade deve ser usada para (re)criar a base de dados sempre que necessário.

Deve ser criado um ficheiro chamado *criar.sql* que inclua as instruções SQL para criar todas as tabelas mencionadas no esquema relacional resultante do passo 4. Antes da criação das tabelas, ser garantida a eliminação de tabelas anteriores com o mesmo nome. O ficheiro deve assemelhar-se a:

```
drop table if exists R1;
...
create table R1 ( .... );
...
```

E. Adição de restrições à base de dados

Na criação da base de dados devem ser incluídas todas as restrições convenientes para a manutenção da integridade dos dados armazenados. É necessário considerar que a implementação de restrições em SQLite não está totalmente em conformidade com o standard SQL-99 (SQL2).

As restrições definidas devem ser listadas, de forma ordenada e em linguagem natural (por exemplo: "não pode haver dois estudantes com o mesmo ID"), no relatório. Para cada uma das restrições deve também ser especificada a sua forma de implementação – restrição chave (PRIMARY KEY ou UNIQUE), restrição de integridade referencial (chave estrangeira), restrição CHECK, restrição NOT NULL?

Depois de identificada a forma de implementação de cada restrição, é necessário implementá-las. Para isso devem ser feitas alterações ao ficheiro *criar.sql*. As restrições que necessitarem de um gatilho para serem implementadas, devem ser deixadas para a Entrega III do projeto.

O ficheiro *criar.sql* deve ser submetido na 2ª entrega.

F. Carregamento de dados

Após a criação da base de dados é necessário proceder ao seu povoamento. Nesta fase deve ser criar um ficheiro chamado *povoar.sql* que contenha as instruções SQL

necessárias para a introdução de dados nas tabelas criadas. No início deste ficheiro deve ser incluída a instrução

```
PRAGMA foreign_keys = ON;
```

de forma a garantir que está ativa a verificação de integridade referencial da base de dados.

O ficheiro *povoar.sql* deve ser submetido na 2ª entrega.

G. Interrogação da Base de dados

Para esta tarefa deve ser definido um conjunto de interrogações pertinentes para o contexto da base de dados. Por exemplo, uma interrogação que liste os países existentes numa base de dados de uma biblioteca é menos pertinente do que uma interrogação que lista os livros mais requisitados num dado período. Deste conjunto, devem ser selecionadas 10 interrogações que:

- sejam diferentes entre si (por exemplo, ter uma interrogação que lista o nome dos clientes na base de dados e outra que lista o nome das empresas na base de dados é equivalente a ter apenas 1 interrogação);

- na sua construção façam uso da maior diversidade de operadores SQL;

- sejam de complexidade distinta.

As 10 interrogações devem ser listadas, de forma ordenada e em linguagem natural, no relatório.

Tal como na criação da base de dados, as interrogações devem começar por ser testadas interactivamente através do cliente de linha de comando do SQLite.

As interrogações devem ser eficientes. Sempre que possível devem privilegiar as junções às sub-interrogações.

Cada uma das 10 interrogações deve ser escrita num ficheiro próprio: int1.sql, int2.sql, ..., int10.sql. No início destes ficheiros devem ser incluídas as seguintes instruções para tornar o resultado mais legível:

```
.mode columns
.headers on
.nullvalue NULL
```

Os nomes dos ficheiros devem corresponder à ordenação das interrogações mencionadas no relatório.

H. Adição de gatilhos à base de dados

Por fim, devem ser definidos 3 gatilhos que sejam úteis para a monitorização e manutenção da base de dados. Pelo menos um dos gatilhos deve implementar uma restrição. Para cada gatilho devem ser criados 3 ficheiros: *gatilhoN_adiciona.sql*, *gatilhoN_remove.sql* e *gatilhoN_verifica.sql*, com N = 1, 2 ou 3.

Em *gatilhoN_adiciona.sql*, deve ser incluída a instrução SQL que permite criar o gatilho. Caso a restrição para a qual se está a criar o gatilho possa ser violada por mais do que um tipo de modificação à base de dados, pode ser necessário criar mais do que um gatilho para garantir a restrição. Se o gatilho descobrir que uma restrição está a ser violada, pode modificar a base de dados de forma a garantir que a violação é anulada ou pode desencadear um erro. Um gatilho SQLite pode desencadear um erro através de:

SELECT raise(rollback, '<mensagem de erro>');

Quando esta instrução é executada, a ação que desencadeou o gatilho é desfeita e é apresentada a mensagem de erro pretendida.

No ficheiro *gatilhoN_remove.sql* deve ser incluída a instrução que elimina o gatilho da base de dados.

No ficheiro *gatilhoN_verifica.sql* devem ser incluídas as instruções SQL que permitem confirmar que o gatilho está bem implementado. Por exemplo, se o gatilho inserir um tuplo na relação R2 sempre que seja inserido um tuplo na relação R1, este ficheiro deverá ter instruções semelhantes a:

```
SELECT * FROM TABLE R2;

INSERT INTO R1 VALUES (valor1, valor2, ...);

SELECT * FROM TABLE R2;
```

No relatório deve descrever sucintamente, de forma ordenada e em linguagem natural, os 3 gatilhos implementados.

Os nomes dos ficheiros devem corresponder à ordenação das interrogações mencionadas no relatório. Em cada um dos ficheiros, deve ser ativada a verificação de integridade referencial.

ATRASOS

Por uma questão de justiça, entregas tardias serão penalizadas em 1 valor por cada dia de atraso.

Não serão aceites entregas após 1 semana da data de submissão.

AUTORIA E ORIGINALIDADE DO TRABALHO

Todos os trabalhos terão a sua originalidade amplamente escrutinada. Os autores de prevaricações serão punidos caso os trabalhos apresentem semelhanças com trabalhos de terceiros (trabalhos não citados, outros trabalhos de estudantes da unidade curricular, etc.), desde a anulação da inscrição à unidade curricular até à instauração dum processo disciplinar a todos os elementos do grupo em questão.