L.EC012: Bases de Dados

Projeto de BD 21/22

[English version in pages 7-12 of this document]

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 [31 de outubro, 23h55]

Definição do tema [7 de novembro, 23h55]

Entrega I [21 de novembro, 23h55]

i. **Tarefa**: A
ii. **Entregas**: Relatório em formato pdf com: capa (1 página), índice (1 página), contexto (1 página) e diagrama UML (1 página)
iii. **Avaliação**: 25% da nota do projeto

Entrega II [12 de dezembro, 23h55]

i. **Tarefas**: B, C, D, E e F
ii. **Entregas**:
  - **Relatório em formato pdf entregue em I com as seguintes secções adicionais**: diagrama UML revisto (1 página), esquema relacional (1 página), análise dependências funcionais e formas normais (1 a 2 páginas); lista e forma de implementação das restrições (1 a 2 páginas)
  - Ficheiros: criar.sql e povoar.sql

i. **Avaliação**: 25% da nota do projeto

Entrega III [30 de janeiro, 23h55]

i. **Tarefas**: G e H
ii. **Entregas**:
  - **Relatório em formato pdf entregue em II com as seguintes secções adicionais**: listagem das 10 interrogações em linguagem natural (1 página); listagem dos 3 gatilhos em linguagem natural (1 página)
  - **Ficheiros: criar.sql, povoar.sql, intN.sql (x10), gatilhoN_adiciona.sql (x3), gatilhoN_remove.sql (x3)** ***e*** gatilhoN_verifica.sql (x3)

i. **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 2LEIC01.

Definição do tema

O tema deve ser proposto pelo grupo ao docente das aulas teórico-práticas na semana que começa a 1 de novembro. Após aprovação do tema, 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. A descrição do contexto deve incluir toda a informação que possa ser importante para a avaliação do modelo conceptual e **não deve ser** uma descrição 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. Relações que não estejam na Forma Normal Boyce-Codd nem na 3ª Forma Normal devem ser decompostas para uma destas formas normais.

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, devem implementar apenas um dos gatilhos e indicar no relatório que tipo de gatilho(s) adicional(is) seria(m) necessário(s) para impor 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(abort, ‘<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.

Avaliação da participação dos vários elementos do grupo

No final do relatório associado a cada entrega deve constar um parágrafo que avalie qualitativamente a contribuição de cada elemento do grupo para o resultado final associado a essa entrega. Este parágrafo permitirá detetar casos anómalos e agir em conformidade.

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 de um processo disciplinar a todos os elementos do grupo em questão.

DB Project 21/22

This project aims to create and query a database in a context proposed by each group. To do so, it will be necessary to create the conceptual model for the selected theme, map this model to a relational schema, implement the schema in a SQLite database, populate the database and, finally, query the database.

Deliverables and dates

Composition of the working group [October 31, 11:55 PM]

Theme Definition [November 7, 11:55 PM]

Deliverable I [November 21, 11:55 PM]

iv. **Task**: A
v. **Deliverables**: Report in pdf format with: cover (1 page), table of contents (1 page), context description (1 page) and UML diagram (1 page)
vi. **Evaluation:** 25% of the project grade

Deliverable II [December 12, 11:55 PM]

ii. **Tasks**: B, C, D, E e F
iii. **Deliveries**:
  - **Report in pdf format delivered in I with the following additional sections**: revised UML diagram (1 page), relational schema (1 page), functional dependencies analysis and normal forms (1 to 2 pages); list and implementation form of constraints (1 to 2 pages)
  - Files: create.sql and populate.sql

i. **Evaluation**: 25% of the project grade

Deliverable III [January 30, 11:55 PM]

ii. **Tasks**: G e H
iii. **Deliverables**:
  - **Report in pdf format delivered in II with the following additional sections**: listing of 10 queries in natural language (1 page); listing of 3 triggers in natural language (1 page)
  - Files: create.sql, populate.sql, intN.sql (x10), triggerN_add.sql (x3), triggerN_remove.sql (x3) and triggerN_verify.sql (x3)

i. **Evaluation**: 50% of the project grade

Composition of each group of students

The project will be done in groups of 3 students. Each group should be created in the "Groups for the project" activity in moodle. The first digit of the group identifier should be the class number. For example, group 101 should be composed by students from the class 2LEIC01.

Theme Definition

The theme should be proposed by the group to the lecturer of the theoretical-practical classes during the week beginning November 1st. After approval of the topic, the group must send, through Moodle, a description of its topic with a maximum of 100 words. This description should be similar to the statements of the exercises of the theoretical-practical classes, stating the expected entities, their attributes and how they relate to each other.

As an indication, relational schemas with 10 to 15 relations are expected. Some of these relations should have composite keys.

Tasks

I. Conceptual Model Definition

Before creating the conceptual model there should be a familiarization with the context associated with the topic of the work. In this step it is intended that students understand in detail the data that will have to be stored in the database.

A conceptual model must then be created in UML for the database to be created. In this model all necessary constraints and multiplicities of associations must be clearly indicated. If there are derived elements, these must be flagged appropriately.

The UML class diagram should be included in a report, where the context associated with the database should also be described. The context description must include all information that may be important for the evaluation of the conceptual model and *should not* be a description of the conceptual model.

J. Relational Schema Definition

Before the relational schema is defined, the conceptual model should be revised based on the evaluation of Deliverable I. The revised UML diagram should be added to the report that was previously delivered.

The revised conceptual model should be mapped to the relational schema, which should be added to the report in textual format using syntax:

R1 (atr1, atr2, atr3->R2)

A clear indication of the primary and foreign keys of each relation is expected.

K. Functional Dependencies Analysis and Normal forms

For each relation the associated set of functional dependencies and any violations of the Boyce-Codd Normal Form and 3rd Normal Form must be given. The non-existence of violations must be justified. Relations that are neither in the Boyce-Codd Normal Form nor in the 3rd Normal Form must be decomposed to one of these normal forms.

L. Database creation in SQLite

The next step involves creating the database in SQLite. SQLite allows you to read commands from a file. This feature should be used to (re)create the database whenever necessary.

You should create a file called create.sql that includes the SQL statements for creating all the tables mentioned in the relational schema resulting from step 4. Before the tables are created, make sure to delete previous tables with the same name. The file should look like this:

```
drop table if exists R1;

...

create table R1 ( .... );

...
```

M. Adding Constraints to the Database

When creating the database, you should include all convenient constraints for maintaining the integrity of the stored data. It is necessary to consider that the implementation of constraints in SQLite is not fully compliant with the SQL-99 (SQL2) standard.

The defined constraints must be listed, in an ordered manner and in natural language (for example: "no two students can have the same ID"), in the report. For each of the constraints you must also specify how it is implemented - key constraint (PRIMARY KEY or UNIQUE), referential integrity constraint (foreign key), CHECK constraint, NOT NULL?

Once you have identified how to implement each constraint, you need to implement them. To do this, changes must be made to the create.sql file. Constraints that require a trigger to be implemented should be left for Deliverable III of the project.

The create.sql file should be submitted in the 2nd deliverable.

N. Data Loading

After creating the database it is necessary to populate it. At this stage you must create a file called populate.sql that contains the SQL statements necessary for the introduction of data in the tables created. At the beginning of this file you must include the statement

```
PRAGMA foreign_keys = ON;
```

to ensure that the referential integrity check of the database is active.

The populate.sql file must be submitted in the 2nd deliverable.

O. Database querying

For this task a set of queries relevant to the database context must be defined. For example, a query that lists the existing countries in a library database is less relevant than a query that lists the most requested books in a given period. From this set, 10 queries should be selected that follow the criteria:

- are **different** from each other (for example, having one query that lists the name of the customers in the database and another that lists the name of the companies in the database is equivalent to having only 1 query);

- in their construction make use of the widest **diversity** of SQL operators;

- are of distinct **complexity**.

The 10 questions should be listed, in an ordered manner and in natural language, in the report.

As with the database creation, queries should first be tested interactively via the SQLite command line client.

Queries should be efficient. Whenever possible you should favor joins rather than queries with subqueries.

Each of the 10 queries should be written to its own file: int1.sql, int2.sql, ..., int10.sql. At the beginning of these files the following instructions should be included to make the result more readable:

```
.mode columns

.headers on

.nullvalue NULL
```

The file names should correspond to the order of the queries mentioned in the report.

P. Adding Triggers to the Database

Finally, 3 triggers should be defined that are useful for monitoring and maintaining the database. At least one of the triggers must implement a constraint. For each trigger 3 files must be created: triggerN_add.sql, triggerN_remove.sql and triggerN_check.sql, with N = 1, 2 or 3.

In triggerN_add.sql, you must include the SQL statement that allows you to create the trigger. If the constraint for which you are creating the trigger can be violated by more than one type of modification to the database, you must implement only one of the triggers and indicate in the report which additional type of trigger(s) would be needed to enforce the constraint. If the trigger finds that a constraint is being violated, it can modify the database to ensure that the violation is disabled or it can trigger an error. A SQLite trigger can trigger an error by:

```
SELECT raise(abort, ‘<error message>’);
```

When this instruction is executed, the action that triggered the trigger is undone and the desired error message is displayed.

In the file triggerN_remove.sql the statement that deletes the trigger from the database must be included.

In the file triggerN_verify.sql you should include the SQL statements that allow you to confirm that the trigger is well implemented. For example, if the trigger inserts a tuple into relation R2 whenever a tuple is inserted into relation R1, this file should have statements similar to:

```
SELECT * FROM TABLE R2;

INSERT INTO R1 VALUES (value1, value2, ...);

SELECT * FROM TABLE R2;
```

In the report you should briefly describe, in an ordered manner and in natural language, the 3 triggers implemented.

The file names must match the order of the interrogations mentioned in the report. In each of the files, the referential integrity check must be activated.

Evaluation of the participation of the various elements of the group

At the end of the report associated with each deliverable there should be a paragraph that qualitatively assesses the contribution of each member of the group to the final result associated with that deliverable This paragraph will allow anomalous cases to be detected and to act accordingly.

Delays

In order to be fair, late submissions will be penalized 1 value for each day of delay.

No submissions will be accepted later than 1 week from the submission date.

Authorship and Originality of the Work

All works will have their originality fully scrutinized. The authors of prevarications will be punished in case of similarities with third-party works (works not cited, other works by students of the curricular unit, etc.), that can range from the cancellation of the enrollment on the curricular unit to the initiation of disciplinary process to all elements of the group in question.