PROGRAMAÇÃO EM SQL E ADMINSTRAÇÃO DE BDs
UFCD(s) 10788 / 3933

GUIA DE LABORATÓRIO 1.1 (Versão ß)
DEFINIÇÃO DO MODELO DE DADOS

OBJECTIVOS

- Criar e definir tabelas (e respectivos campos) no MySQL e inserir dados nas tabelas criadas
- Iniciar a implementação da base de dados `ENSINO` que será usada como exemplo neste e nos próximos laboratórios

INSTRUÇÕES

1a Parte – Criação das tabelas

1.1 Eis a descrição do modelo de dados que queremos implementar:

> *Uma instuição de ensino pretende criar um sistema de informação para apoio às actividades lectivas. Este SI assenta numa BD relacional que deverá ser implementada em MySQL.*
>
> *O SI permite que os estudantes se inscrevam em uma ou mais acções de cursos. Um estudante é identificado pelo identificador de estudante, e possui nome, endereço, entre outros atributos. Cada estudante inscreve-se então numa acção de um curso. Para cada inscrição, além das chaves primária e estrangeiras apropriadas, é necessário armazenar a data de inscrição e a classificação final do estudante. Para cada acção, guarda-se o número da acção, as datas inicial e final, bem como o nome do coordenador da acção que, para simplificar, é apenas o nome de um individuo. Para cada curso armazena-se o identificador de curso, o nome do curso, a duração e o tipo de curso. Para já, o tipo de curso é apenas uma descrição genérica - eg, "Ciências da Vida", "Ciências Económicas", etc.*

Desenhe o modelo de dados para esta descrição.

2. Inicie o MySQL Workbench e crie uma BD com o nome `ENSINO`. Os comandos que se seguem devem ser executados nesta BD.

```
DROP DATABASE IF EXISTS ENSINO;
CREATE DATABASE ENSINO;

USE ENSINO;
```

Esta BD será utilizada nos laboratórios e exemplos seguintes e, por isso, deverá guardá-la num local bem definido e ao qual tenha acesso quando for necessário.
*O comando **`CREATE DATABASE`** é um dos vários comando DDL (Data Definition Language) da linguagem SQL, comandos utilizádos para definir o modelo de dados e criar objectos na BD.*
*Por razões de conveniência, estes comandos **`CREATE`** devem ser precididos por um comando **`DROP`** com a cláusula **`CREATE IF EXISTS`** . Assim, é mais fácil e rápido re-criar um objecto.*

*O comando **`USE`** é específico do MySQL (ou seja, não faz parte da linguagem SQL) e serve para informar uma ferramenta cliente da BD (como o MySQL Workbench, por exemplo) em que BD os comandos seguintes devem ser dados.*

3. Introduza o seguinte comando para criar a tabela **Estudante**

```
DROP TABLE IF EXISTS Estudante;
CREATE TABLE Estudante (
       ID               INTEGER PRIMARY KEY AUTO_INCREMENT,
       NomeEstudante    VARCHAR(20),
       Endereco         VARCHAR(50),
       Cidade           VARCHAR(50),
       CodigoPostal     INTEGER,
       DataNascimento   DATE
);
```

*A primeira "propriedade" após o nome de cada campo é o **tipo de dados** do campo. Um tipo de dados define um universo de valores admissíveis para um campo. Os tipos em cima utilizados foram:*
*- **INTEGER** ou **INT**: valores inteiros de - a + infinito*
*- **VARCHAR**: sequência de caracteres de tamanho variável, tendo no máximo 20 caracteres*
*- **DATE**: sequência de caracteres que permitem definir uma data válida. Esta data deve estar no formato 'AAAA-MM-DD'.*

Para mais informação, consultar:
https://www.w3schools.com/mysql/mysql_create_table.asp
https://dev.mysql.com/doc/refman/8.0/en/create-table.html

*Uma tabela cria-se com o comando SQL **CREATE TABLE** que necessita do nome da tabela seguido duma lista de campos entre parênteses e separados por vírgulas. Cada campo consiste de um nome, tipo e propriedades.*
*Neste exemplo, criámos a tabela `Estudante` cuja chave-primária é o campo `ID`. É um campo inteiro, incrementado automaticamente conforme indica a propriedade **AUTO_INCREMENT**; ou seja, o seu valor é preenchido pelo MySQL sempre que uma nova linha (ie, um novo estudante) for inserida. O seu valor será igual ao do código do estudante anterior incrementado em uma unidade.*
Uma tabela corresponde a uma definição de campos/colunas e de propriedades globais à tabela. Neste exemplo, apenas definimos campos. Mas, a título de exemplo, poderíamos ter definido a chave-primária da seguinte forma:

```
CREATE TABLE Estudante(
    ID INTEGER AUTO_INCREMENT,
    ...etc...
    CodigoPostal INTEGER,
    PRIMARY KEY(ID)
);
```

4. Introduza o seguinte comando para criar a tabela **Curso**:

```
DROP TABLE IF EXISTS Curso;
CREATE TABLE Curso (
       ID           INTEGER PRIMARY KEY AUTO_INCREMENT,
       Nome         VARCHAR(20),
       Duracao      INTEGER,
       Tipo         VARCHAR(40)
);
```

5. Agora, introduza e execute os comando em baixo, e confirme a estrutura de ambas as tabelas:

```
DESCRIBE Estudante;
DESCRIBE Curso;
```

Não precisa de guardar num script estes comandos e os que se seguem no ponto seguinte. Servem só para testar e acompanhar o modelo de dados que está a desenvolver.

6. Confirme a mesma informação executando os seguintes dois comandos:

```
SELECT COLUMN_NAME, COLUMN_TYPE, IS_NULLABLE, COLUMN_KEY, COLUMN_DEFAULT, EXTRA
FROM   information_schema.columns
WHERE  table_name = 'Estudante';
```

```
SELECT COLUMN_NAME, COLUMN_TYPE, IS_NULLABLE, COLUMN_KEY, COLUMN_DEFAULT, EXTRA
FROM   information_schema.columns
WHERE  table_name = 'Curso';
```

7. Vamos agora criar a tabela Accao:

NOTA: 1) todos os identificadores criados nestes laboratórios não terão nem acentos nem 'ç'; 2) neste e nos restantes laboratórios utilizar-se-á a grafia do antigo acordo ortográfico.

```
DROP TABLE IF EXISTS Accao;
CREATE TABLE Accao (
    ID          INTEGER PRIMARY KEY AUTO_INCREMENT,
    Numero      INTEGER,
    DataInicial DATE,
    DataFinal   DATE,
    Coordenador VARCHAR(20),
    IDCurso     INTEGER NOT NULL,
    FOREIGN KEY (IDCurso) REFERENCES Curso(ID)
);
```

Uma chave-estrangeira permite que uma tabela estabeleça um relacionamento com outra tabela. É criada a partir de uma ligação de uma ou mais colunas a um número equivalente de colunas noutra tabela. Esta ligação permite ao MySQL garantir que a informação entre as duas tabelas é consistente. Esta garantia de consistência é designada de ***integridade referencial***.
A sintaxe para criar uma chave estrangeira é a seguinte:

```
[CONSTRAINT nome_chave_estr]
FOREIGN KEY [index_name] (column_name, ...)
REFERENCES parent_table(colunm_name,...)
[ON DELETE reference_option]
[ON UPDATE reference_option]
```

Desde o MySQL 8.0.16, `nome_chave_estr` indica o nome da chave-estrangeira e `index_name` é ignorado. Este nome pode depois ser utilizado para remover a chave-estrangeira ou alterar as suas propriedades. Falaremos sobre `ON DELETE` e `ON UPDATE` no próximo laboratório.

8. E, finalmente, a tabela Inscricao:

```
CREATE TABLE Inscricao (
    DataInscricao       DATE,
    ClassificacaoFinal  DECIMAL
);
```

NOTA 1: Propositadamente omitimos chaves-estrangeiras desta tabela para as tabelas **Estudante** e **Accao**. Ambas serão definidas no próximo laboratório.

NOTA 2: Doravante, por uma questão de brevidade, iremos omitir o comando DROP antes do respectivo comando CREATE. Deve, no entanto, acrescentá-lo aos seus scripts à medida que for criando objectos na MySQL.

2a Parte – Introdução de dados

9. Vai agora introduzir um estudante na BD. Execute o seguinte comando:

```
INSERT INTO Estudante
(NomeEstudante, Endereco, Cidade, CodigoPostal, DataNascimento)
VALUES
('Antonio', 'Rua do Alecrim, n. 1', 'Albufeira', 3001, '1997-08-22');
```

*Adicionámos uma linha a uma tabela através do comando SQL **INSERT**. Dada a tabela T com os campos numéricos C1 e C2, o comando **INSERT INTO** T (C1, C2) **VALUES** (1, 2) acrescenta uma nova linha a T. Não é obrigatório indicar todas as colunas. Ex: **INSERT INTO** T (C2) **VALUES** (2). Neste caso, C1 fica com o valor por omissão **NULL** que significa "vazio". Podemos omitir a lista de colunas se inserirmos um valor para coda coluna; o primeiro exemplo desta caixa poderia ter sido escrito assim: **INSERT INTO** T **VALUES** (1, 2).*

10. De forma idêntica, introduza também os seguintes elementos:

NomeEstudante	Endereco	Cidade	CodigoPostal	DataNascimento
Beatriz	Rua do Beato, lote 2	Braga	3002	1997-02-23
Catarina	Praça da Consituição, n. 3	Coimbra	3003	1998-08-10
Diogo	Avenida Dom Afonso, lote 4	Domelas	3004	1995-02-04
Eduardo	Praça de Espanha, n. 5	Évora	3005	2002-03-05
Filipa	Travessa da Ferreirinha, 6	Faro	3006	2004-02-29

3a Parte – Visualização de dados

11. Confirme que o ID do Antonio é 1, que o da Beatriz é 2, etc., através do comando:

```
SELECT NomeEstudante, ID FROM Estudante;
```

12. Agora verifique todo o conteúdo de toda a tabela através do comando:

```
SELECT * FROM Estudante;
```

*O **SELECT** é um comando que permite pesquisar a BD. É, talvez, o comando SQL mais utilizado. Permite seleccionar algumas linhas (ou todas, como é aqui o caso) e visualizar o conteúdo de várias colunas dessas linhas. Especificamos a lista de colunas a ver entre a palavra **SELECT** e a palavra **FROM**, sendo que, as colunas devem estar separadas por vírgulas. Se utilizarmos o caracter * (asterisco) em vez da lista de colunas, o **SELECT** mostra todas as colunas. Vamos aprofundar a utilização do **SELECT** nos próximos laboratórios.*

4a parte - Modificação de dados e adição de colunas

13. Adicione a coluna Nacionalidade através do comando:

```
ALTER TABLE Estudante ADD COLUMN Nacionalidade VARCHAR(20);
```

*O **ALTER TABLE** é o comando utilizado para modificar a definição de uma tabela. O MySQL permite mudar o nome da tabela ou de uma coluna, e acrescentar uma nova coluna. Também permite a remoção de colunas e a modificação das propriedades de uma coluna.*

14. Todos os `Estudantes` são portugueses. Indique isto através do seguinte comando:

```
UPDATE Estudante SET Nacionalidade = 'Portuguesa';
```

> ***UPDATE*** *é o comando utilizado para modificar os dados de uma tabela. Neste caso, modificamos, o valor da coluna Nacionalidade para todas as linhas da tabela* ***ESTUDANTE*** *(ie, para todos os estudantes). É possível especificar um critério de selecção de colunas conforme veremos já a seguir e, em maior detalhe, num dos próximos laboratórios.*

15. Confirme que, de facto, todos os elementos possuem nacionalidade `Portuguesa`.

16. Acrescente a coluna `Apelido` que deverá ser um texto com 50 caracteres.

17. Indique que o apelido do `António` é `Américo` através do comando:

```
UPDATE Estudante SET Apelido = 'Américo' WHERE NomeEstudante = 'António';
```

18. Agora, vamos acrescentar um apelido a todos os estudantes:

Nome	Apelido
Beatriz	Bastos
Catarina	Coelho
Diogo	Diniz
Eduardo	Esteves
Filipa	Fernandes

19. Afinal o `Eduardo` vive em `Espinho`. Introduza esta alteração na BD.

20. Nos próximos laboratórios vamos recriar as tabelas `Estudante`, `Inscricao`, `Accao` e `Curso`. Assim, terminamos este laboratório removendo-as da BD (depois de todo este trabalho...):

```
DROP TABLE Estudante;
DROP TABLE Inscricao;
DROP TABLE Accao;
DROP TABLE Curso;
```