DDL (Data Definition Language): É o conjunto de comandos SQL responsável pela definição dos dados, ou seja, pela criação de base de dados, esquemas, tabelas, campos, tipos de dados, restrições etc. E utilizados pelos projetistas de base de dados para a definição das mesmas.

1- Criar Base de dados

Sintaxe:
CREATE [TEMPORARY] DATABASE [IF NOT EXISTS] <nome_da_BD>

Exemplo: Criar a base de dados DBDevMedia

a. CREATE DATABASE DBDevMedia

b. CREATE DATABASE IF NOT EXISTS DBDevMedia

   Caso não se tenha a certeza se já existe uma base de dados com o mesmo nome, evitando um erro, caso a db exista.

TEMPORARY: Indica que a tabela criada será temporária, ou seja, ela expira assim que a sessão no MySQL terminar. Deve usar-se sempre que se estiver a fazer testes.

1.1 Utilizar/selecionar a base de dados para manipulação
Sintaxe:
USE <nome_da_BD>

Exemplo: USE DBDevMedia

1.2 Visualizar uma lista com todas as bases de dados existentes no servidor
Sintaxe:
SHOW DATABASES

1.3 Remover as bases de dados existentes no servidor (atenção, a ação é irreversível!)

Sintaxe:
DROP <nome_da_BD>

Exemplo: DROP DBDevMedia

2- Criar Tabelas

Sintaxe:

```
CREATE TABLE <nome_tabela> (
<campo1> <tipo> [tamanho] [<restricao>],
<campo2> <tipo> [tamanho] [<restricao>],
…
<campoN> <tipo> [tamanho] [<restricao>])
```

- Os elementos entre parêntesis retos são opcionais.

Exemplo: Criar a tabela Empregado com os campos Id, Nome, Dat_Nasc e Salario, com o tipo, inteiro, texto, data e real, respetivamente.

```
CREATE TABLE Empregados ( Id integer,
                          Nome varchar (50),
                          Dat_Nasc datetime,
                          Salario float (4))
ENGINE=InnoDB                                    ←
DEFAULT CHARSET=latin1        Outras opções possíveis após o fecho do parêntesis
```

1.1 Inserção de valores por defeito em campos da tabela

Poderão ser indicadas as caraterísticas próprias de cada um dos campos, que valores a admitir e, caso não seja introduzido um valor, qual o valor por defeito, etc.

No exemplo anterior não foi indicada qualquer restrição nos campos, pelo que as colunas podem admitir valores nulos (isto é, admitem como valor por defeito NULL).

2.1 Criação de uma tabela a partir de outra tabela usando a cláusula SELECT

Sintaxe:

```
CREATE TABLE tabela_nova AS SELECT <campo1>, … ,<campoN>
FROM tabela_original
```

Exemplo:

```
CREATE TABLE Pessoa_aux1 AS
SELECT *
FROM Pessoa
WHERE Id<=35
```

NOTA Na criação da tabela poderás especificar se um campo pode ou não admitir valores nulos através das cláusulas NULL ou NOT NULL, imediatamente a seguir ao tipo de dados.

2.2 Restrições (CONSTRAINTS)

Constraints são regras a que os valores de um ou mais campos devem obedecer.

Todas as linhas que se pretendem inserir numa tabela terão que obedecer ao conjunto das restrições definidas para os campos (colunas) dessa tabela.

A utilização de restrição é a única garantia quer temos que os dados existentes nos campos estão de acordo com as regras especificadas no desenho da base de dados.

A utilização de restrições garante-nos a correção dos dados.

2.2.1 *CONSTRAINT* NOT NULL

Impede a introdução de valores nulos no campo onde é especificado.

Exemplo:

```
CREATE TABLE teste ( Id integer          NOT NULL,
                     Nome Char (30) NOT NULL)
```

2.2.2 *CONSTRAINT* UNIQUE

Permite indicar que os valores do campo não se podem repetir. Numa tabela pode existir tantas colunas UNIQUE quantas forem necessárias.

Exemplo:

```
CREATE TABLE teste ( Id       integer     NOT NULL,
                     Nome   char (15),
                     Apelido char(15),
                     CC        integer,
                     UNIQUE (nome, apelido),
                     UNIQUE (CC) )
```

Ou escrito da seguinte forma:

```
CREATE TABLE teste ( Id       integer          NOT NULL,
                     Nome   char (15)        UNIQUE,
                     Apelido char(15)        UNIQUE,
                     CC       integer          UNIQUE)
```

Está garantido que não existirão na tabela duas pessoas com o mesmo nome e apelido e com o mesmo n.º de CC

2.2.3 *CONSTRAINT* CHECK

Restringe os valores inserir nas tabelas de acordo com determinados critérios a definir.

Por exemplo:

- O conteúdo do campo *Género* só poderá conter os valores "F" ou "M";

```
...( genero char NOT NULL CHECK (genero in ('F', 'M')),..
```

- O campo *Idade* não pode conter valores negativos, …

```
...idade integer CHECK ( idade >=0),...
```

Nota: a restrição CHECK (<campo> condição) não é válida em Access (é suportada em MySql e oracle, por exemplo)

2.2.4 AUTO-INCREMENT

Utilizado para incrementar automaticamente um campo que serve de chave primária de uma tabela.

Exemplo:

```
CREATE TABLE teste ( Id       integer     NOT NULL,
                     Nome   char (15), NOT NULL AUTO_INCREMENT,
                        …
```

2.2.5 *CONSTRAINT* PRIMARY KEY

É equivalente às cláusulas NOT NULL + UNIQUE juntas, isto é, o conteúdo dos campos não pode ser nulo e não pode admitir repetições.
Numa tabela apenas pode existir uma cláusula PRIMARY KEY, enquanto podem ocorrer várias cláusulas UNIQUE. Esta cláusula indicará a chave primária da tabela (se existir).

Exemplo:

```
CREATE TABLE teste ( Id  integer  PRIMARY KEY,
                     Nome char (15)    NOT NULL,
                     Apelido char(15)  NOT NULL,
                     CC  integer       UNIQUE)
```

Se a chave primária for constituída por vários campos, então, indica-se no final os campos que constituem a mesma.

Exemplo:

```
CREATE TABLE teste ( Id  integer AUTO_INCREMENT,
                     Nome Char (15),
                     Apelido char(15)      NOT NULL,
                     CC         Char (12)  UNIQUE,
                    PRIMARY KEY ( Id , Nome)  )
```

2.2.6 *CONSTRAINT* FOREIGN KEY

Esta cláusula e a cláusula REFERENCES farão parte da instrução que indicará uma chave externa (ou estrangeira). A instrução poderá ser usada na criação (CREATE TABLE) e alteração (ALTER TABLE) de tabelas.

Sintaxe:

```
FOREIGN KEY (child_col1, child_col2, ... child_col_n)
   REFERENCES parent_table (parent_col1, parent_col2, ... parent_col_n)
   ON DELETE CASCADE
   [ ON UPDATE { NO ACTION | CASCADE | SET NULL | RESTRICT } ]
```

ATENÇÃO: Para chaves externas referentes a tabelas distintas é necessário repetir a cláusula FOREIGN KEY. Em mysql esta cláusula apenas pode ser escrita após a definição do campo, como nos exemplos apresentados na página 5).

Restrições a aplicar nas chaves externas (que garantem a integridade referencial)

NO ACTION

Tolera a falta de integridade referencial, produzindo o mesmo efeito caso não exista qualquer relação entre as chaves primária e externa;

CASCADE

Permite que sejam eliminados/alterados os registos que pertencem à tabela onde se encontra a chave externa e que têm correspondência na tabela onde se encontra a chave primária, isto é, ao apagarem-se/alterarem-se os registos da tabela principal, automaticamente ocorre o mesmo efeito nos registos das tabelas dependentes. É um efeito em cascata que pode ser útil quando se pretende apagar/alterar grandes quantidades de dados ao longo de várias tabelas;

SET NULL

Coloca o valor NULL nas tabelas dependentes quando o registo da tabela principal é eliminado/alterado;

RESTRICT

Não permite que seja eliminado/alterado um registo de uma tabela principal (a que tem a chave primária) se houver um registo numa tabela com chave externa que depende daquele.

As restrições apresentadas podem ser aplicadas em situações tanto de eliminação – **ON DELETE**, como de alteração – **ON UPDATE,** que têm carater opcional.

Exemplos:

1. Criação de uma chave externa na tabela inventario, que se refere à chave primária, constituída pelo campo `produto_id`, da tabela produtos, aquando da criação da tabela inventario.

```
CREATE TABLE inventario
( inventario_id INT PRIMARY KEY,
  produto_id INT NOT NULL,
  quantidade INT,
  min_qtd INT,
  max_qtd INT,
  FOREIGN KEY (produto_id)
  REFERENCES produtos (produto_id)
  ON DELETE SET NULL
  ON UPDATE CASCADE)
```

2 . Criação de duas chaves externas na tabela inventario, que se referem à chave primária, constituída pelos campos nome_produto e local, da tabela produtos, aquando da criação da tabela inventario.

```
CREATE TABLE inventario
( inventario_id INT PRIMARY KEY,
  nome_produto VARCHAR(50) NOT NULL,
  local VARCHAR(50) NOT NULL,
  quantidade INT,
  min_qtd INT,
  max_qtd INT,
    FOREIGN KEY (nome_produto, local)
    REFERENCES produtos (nome_produto, local))
```

3. Criação de uma chave externa na tabela inventario, que se refere à chave primária, constituída pelo campo `produto_id`, da tabela produtos, após a criação da tabela inventario.

```
ALTER TABLE inventario
ADD FOREIGN KEY (produto_id)
    REFERENCES produtos(produto_id)
```

3- Alterar Tabelas

Para alterar a estrutura de uma tabela usa-se o comando **ALTER TABLE**

A seguinte sintaxe permite adicionar um novo campo, alterar as caraterísticas de um campo já existente ou eliminar um campo.

Sintaxe:

ALTER TABLE <nome_tabela>
ADD <campo1> <tipo> [tamanho] [<restricao>] [FIRST|AFTER campoetc],
...
<campoX> <tipo> [tamanho] [<restricao>]

ALTER TABLE <nome_tabela>
MODIFY <campo> <tipo> [tamanho] [<restricao>]
(não válido em Access)

ALTER TABLE <nome_tabela>
DROP COLUMN <campo1>, ... , <campoX>

Exemplo (ADD): Adicionar à tabela Pessoa um campo de nome NIB do tipo inteiro longo.

ALTER TABLE Pessoa **ADD** NIB long

Exemplo (DROP): Para eliminar o campo anteriormente introduzido:

ALTER TABLE Pessoa **DROP COLUMN** NIB

NOTA: Não se pode eliminar um campo que seja um índice. Primeiro tem que se proceder à eliminação da estrutura de índice do campo. O comando modify não é válido em Access.

4- Apagar Tabelas

Para apagar uma tabela usa-se o comando **DROP TABLE**.
Este comando apaga a tabela especificada pelo utilizador com todos os seus dados. Depois de executado o comando não existe qualquer forma de desfazer a operação.

Exemplos:

DROP TABLE Pessoa

EXERCÍCIOS

Fig. 1

Escreve o(s) comando(s) SQL que te permita:

1. Cria uma base de dados com o nome **Empresa1** (usa o comando IF NOT EXISTS).

2. Criar:

   a) a tabela **postal**, como os campos de acordo com a fig. 1, sendo a chave primária do tipo inteiro, e de incremento automático e os restantes campos cadeias de carateres, com tamanho adequado definido por ti. Os campos são de preenchimento obrigatório.
   O motor de armazenamento deve ser InnoDB.

   b) Criar uma tabela, com o nome **comissao,** para armazenar os campos: ***CodC*** (chave primária) e ***valor*** , ambos de preenchimento obrigatório, sendo o último único. Define para o CodC o tipo inteiro e para o campo *valor* o tipo real.

   O motor de armazenamento deve ser InnoDB.

3. Criar uma tabela, com o nome **empregados**, para armazenar os campos: **Id** (chave primária), **Nome, cargo, salario, idade, CodPostal e CodC,** todos de preenchimento obrigatório. Define tipo e tamanho campos de acordo com a fig. 1.
   O campo idade deve ser maior que 18 e menor que 66.
   Define as chaves externas na tabela empregados, aplicando-lhes a restrição CASCADE para eliminação e alteração de registos.

4. Eliminar a tabela criada no exercício anterior.

5. Adicionar à tabela Empregados os campos telefone e fax, do tipo texto, com no máximo 9 carateres, que não poderá admitir repetições, nem poderá ser nulo.

6. Eliminar da tabela empregados o campo *fax*, criado no exercício anterior.

7. Inicializa o serviço de **Wamp**, e através do PHPMyAdmin elabora as seguintes operações:

   7.1. Reproduz e executa o sql relativo a todos os exercícios, exceto o 4. Deves imprimir para PDF todas as alíneas.

   7.2. Exporta o Schema para pdf.
   RELEMBRA: Selecionar a base de dados->selecionar o menu Mais->selecionar o item Desenhador

   7.3. Exporta a base de dados e envia-a para a disciplina de PSD, na classroom do Google Apps.

Bibliografia

Damas, L. (2005). SQL. Lisboa: FCA
Tavares, F. (2015). MySql. Lisboa: FCA
Praciano, E. Como criar uma tabela a partir de uma declaração SELECT.[S.I.]2014.Disponível em: https://elias.praciano.com/2014/03/mysql-como-criar-uma-tabela-a-partir-de-uma-declaracao-select/
https://www.techonthenet.com/sql_server/foreign_keys/