



MINISTÉRIO DAS TELECOMUNICAÇÕES, TECNOLOGIAS
DE INFORMAÇÃO E COMUNICAÇÃO SOCIAL
MINISTÉRIO DA EDUCAÇÃO

TLP – UNIDADE VII: LINGUAGENS DE BASE DE DADOS

**PROF. PAULO TUMBA /
PROF. LUSSATI NARCISO
2022-2023**



INSTITUTO DE TELECOMUNICAÇÕES



ÍNDICE

1. Revisão de Conceitos;
2. Modelo Relacional;
3. Linguagem SQL;
4. DDL;
5. DML
6. Consultas



ÍNDICE

7. Exercícios;



Objectivo – CONHECER O MODELO RELACIONAL, DESMISTIFICAR A LINGUAGEM SQL E AS SUAS SUBDIVISÕES (DDL E DML)



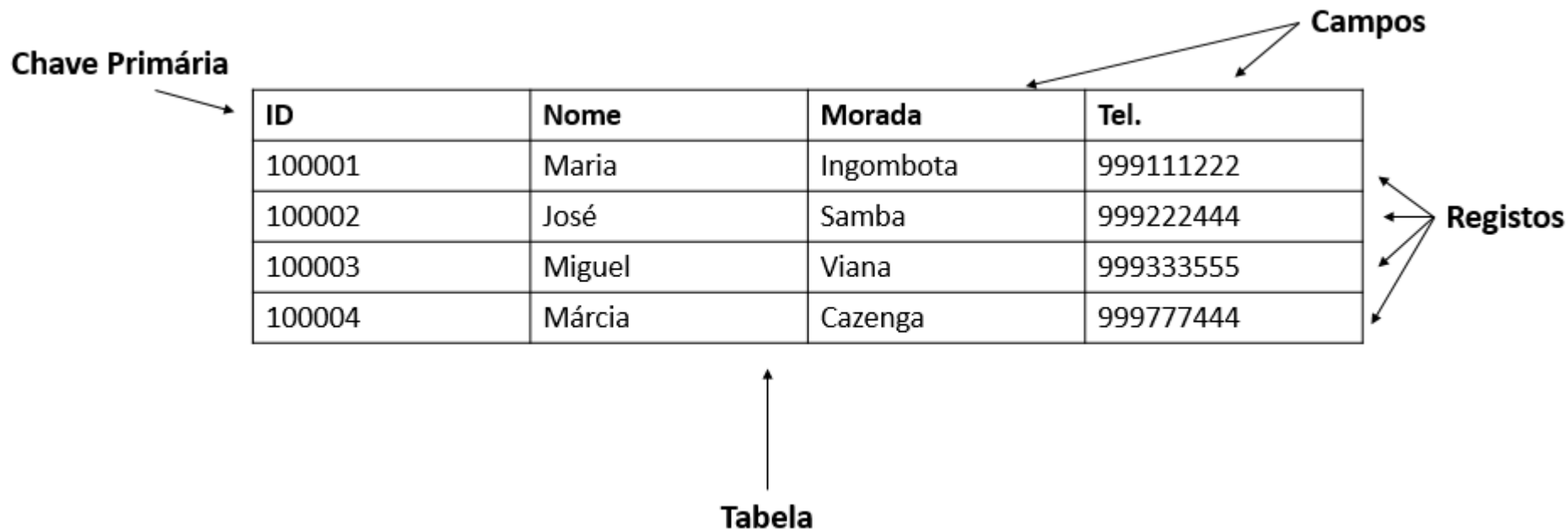
Revisão de Conceitos

- Base de Dados: conjunto de dados estruturados, armazenados de maneira persistente e organizados de acordo com determinado contexto. (Damas,2005)
- Modelo de Dados: forma lógica de representação dos dados. Estrutura de uma
- base de dados pela forma como será implementado pelo SGBD. Ou seja, modelo
- de implementação dos dados.
 - Hierárquico;
 - Em Rede
 - Relacional
 - Orientado à Objectos



Modelo Relacional

- Permite a utilização de uma estrutura de dados, a tabela, que implementa o conceito matemático de relação.
- Ex. Tabela cliente.





Trabalhando com uma BD

- As bases de dados têm que primeiramente ser criadas, i.e criar toda a estrutura para armazenar os dados.
- As operações (acções) que poderão ser realizadas sobre classificadas em:
 - Operações de consulta (leitura)
 - Operações de actualização (escrita)
 - Inserção de novos dados (insert)
 - Remoção de dados (delete)
 - Modificação de dados existentes (update)



Linguagem SQL

- Linguagem de Consulta estruturada SQL(Structured Query Language), é linguagem padrão para se trabalhar com bases de dados relacionais.
- Tem como origem a linguagem SEQUEL desenvolvida pela IBM no início da década de 70 dentro do projecto System R.
- Objecto de um esforço de padronização levado a cabo pela ANSI(American National Standards Institute) e ISO (International Standards Organization) .
 - SQL-86
 - SQL-92
 - SQL:1999
 - SQL:2003
 - SQL:2006
 - SQL:2008
 - SQL:2011
 - SQL:2016

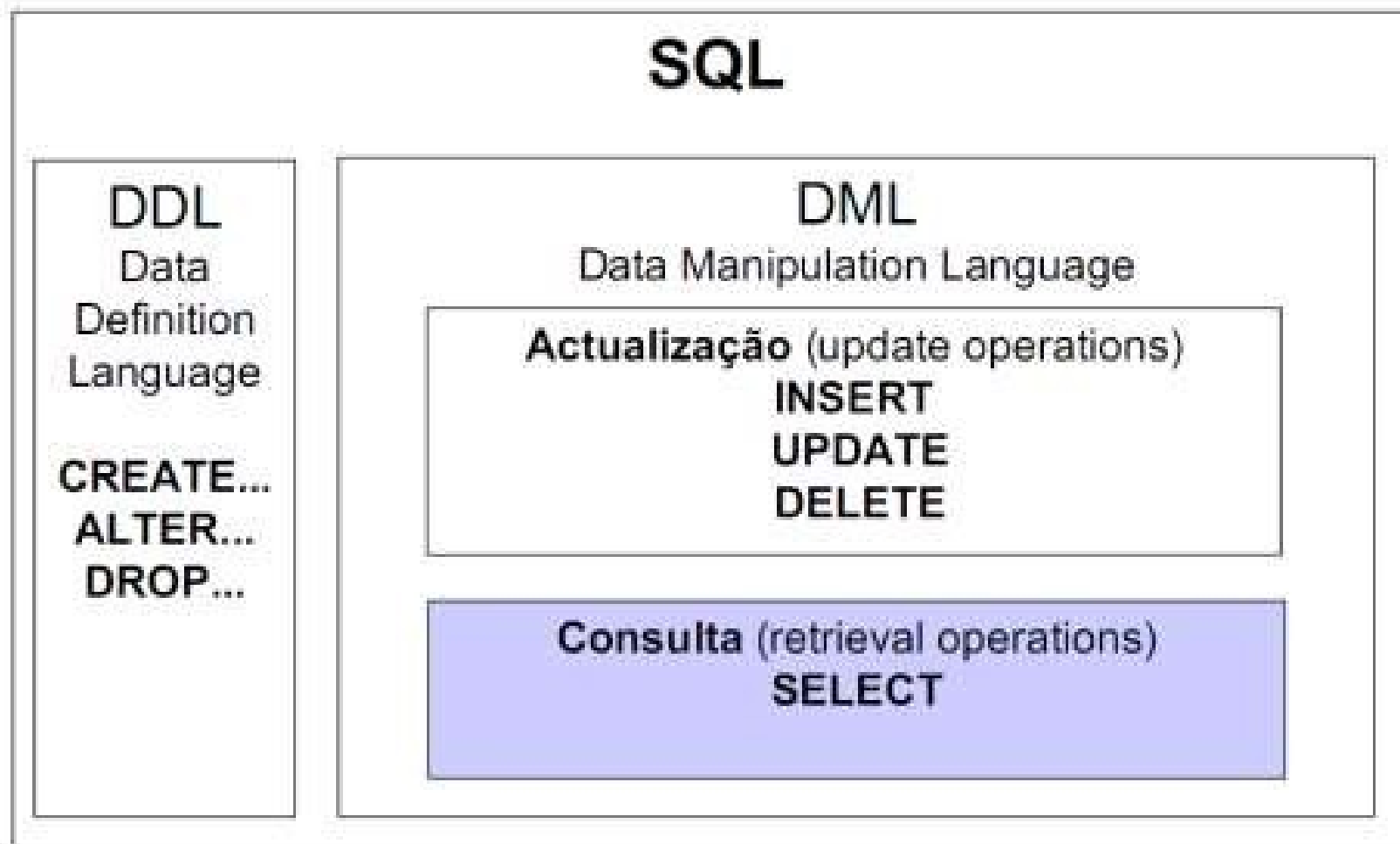


Linguagem SQL

- Linguagem de definição de dados (Data Definition Language - DDL) – utilizada para a definição de esquemas. Especifica os tipos de dados, as estruturas bem como as relações entre os dados (criar tabelas, restrições de integridade).
- Linguagem de Manipulação de Dados (Data Manipulation Language – DML) – permite acessar e/ou manipular os dados (inserir, actualizar, apagar).
- Uma parte da DML é chamada de Linguagem de Consulta (Query Language) – permite consultar dados armazenados na BD.
- Linguagem de Controlo de Dados (Data Control Language - DCL) – permite controlar as permissões de acesso a base de dados, por grupos de utilizadores.
- Nota: Normalmente a DCL é considerado como tarefas de administração de BD.



Linguagem SQL





Data Definition Language – DDL

- **DDL** – conjunto de instruções que permitem a especificação das estruturas dos dados, relações, criação e modificação de base de dados e tabelas.
- **Comandos básicos:**
 - **CREATE** - permite a criação de um novo objecto (base de dados, tabela , tabela, índice, view, procedimentos armazenado, função,etc) de base de dados;
 - **DROP** - permite eliminar um objecto existente;
 - **ALTER** - permite alterar as características de um objecto;



Data Manipulation Language -DML

- **DML** – conjunto de instruções que permitem manipular(inserir, actualizar, apagar) os dados existentes numa base de dados.
- **Comandos básicos:**
 - **INSERT;**
 - **UPDATE;**
 - **DELETE;**
- **Linguagem de consulta** – instruções que permite consultar (visualizar) dados existentes numa tabela.
- **Comando básico:**
 - **SELECT;**



CRIAÇÃO/ELIMINAÇÃO DE BASES DE DADOS

Uma base de dados pode ser criada da seguinte forma:

Sintaxe:

– **CREATE DATABASE** Nome_da_base_de_dados;

OBS: para executar o script, posicione o cursor no final do código(;) e combine as teclas → **Ctrl+Enter**

Exemplo:

– **CREATE DATABASE** administrativo;

A base de dados pode ser eliminada com:

Sintaxe:

– **DROP DATABASE** Nome_da_base_de_dados;

Exemplo:

– **DROP DATABASE** administrativo;

Nota: execute o comando: **use Nome_da_base_de_dados;** , para especificar a base de dados que pretende manipular.



CRIAÇÃO DE TABELAS

Para criar uma tabela é necessário especificar o nome da **tabela**, os **campos** e os respectivos **domínios**.

Sintaxe :

```
CREATE TABLE r (A1 D1, A2 D2, ..., AN DN,  
                < regra de integridade1>,  
                ...,  
                < regra de integridadeN>,  
                )
```

Onde:

r – nome da tabela

A – atributo ou campo

D - Domínio



CRIAÇÃO DE TABELAS

Exemplo :

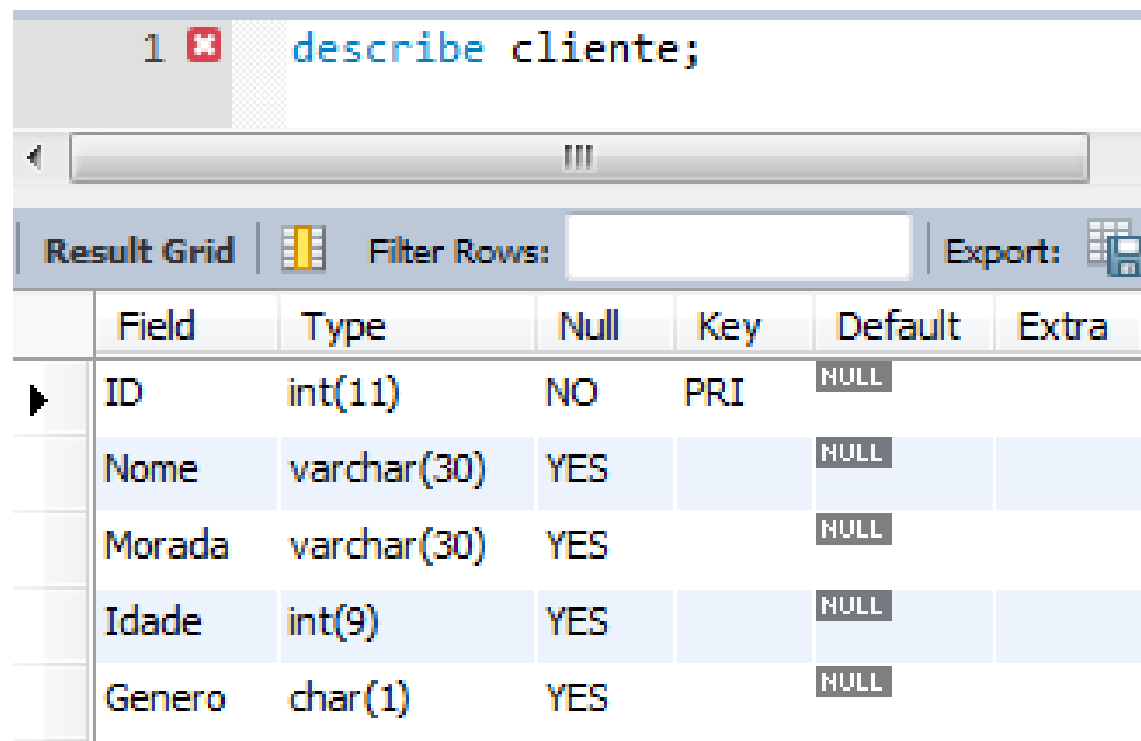
```
CREATE TABLE Cliente (  
    ID int,  
    Nome varchar(30),  
    Morada varchar(30),  
    Idade int,  
    Genero char  
)
```



EXIBIÇÃO DAS DEFINIÇÕES DE TABELAS

Sintaxe : **Describe** nome_tabela; **ou** **Desc** nome_tabela;

Exemplo: **Describe** cliente;



The screenshot shows a database management interface. At the top, a text box contains the command `describe cliente;`. Below it is a horizontal scrollbar. Under the scrollbar is a toolbar with a 'Result Grid' button, a 'Filter Rows:' input field, and an 'Export:' button with a save icon. Below the toolbar is a table displaying the structure of the 'cliente' table.

	Field	Type	Null	Key	Default	Extra
▶	ID	int(11)	NO	PRI	NULL	
	Nome	varchar(30)	YES		NULL	
	Morada	varchar(30)	YES		NULL	
	Idade	int(9)	YES		NULL	
	Genero	char(1)	YES		NULL	



ELIMINAÇÃO DE TABELAS

Sintaxe : **DROP TABLE** Nome_da_Tabela;

Exemplo: **DROP TABLE** cliente;

Nota: Este comando apaga toda a estrutura e conteúdos (dados) da tabela. Uma vez completado com sucesso não existe forma de “desfazer” esta operação. (Damas, 2005)



REGRAS DE INTERIDADE

- Permitem especificar a semântica dos dados e garantem que os dados estão de acordo com as regras especificadas no desenho da Base de dados. (Damas, 2005).
- As regras de integridade protegem a base de dados contra danos acidentais. (Silberschatz et al., 2006).

Tipos de regras de integridade:

- Domínio;
- Chave primária;
- Referencial;



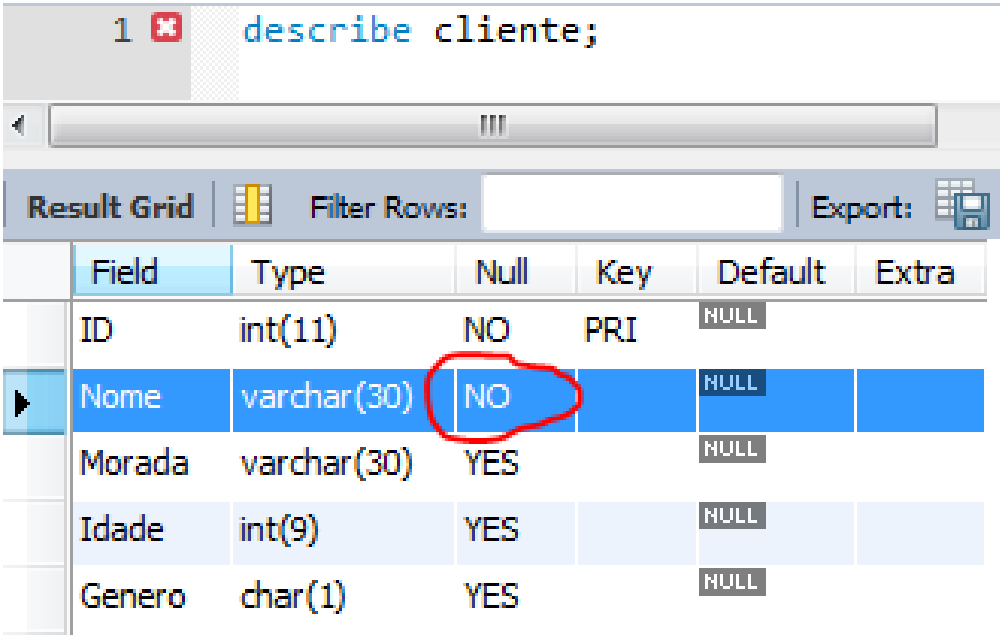
Data Definition Language – DDL

REGRAS DE INTERIDADE DE DOMÍNIO: NOT NULL

- Para que um determinado campo não aceite valores nulos, deve-se juntar ao domínio a cláusula **NOT NULL**.

EXEMPLO:

```
CREATE TABLE Cliente (  
    ID int,  
    Nome varchar(30) NOT NULL,  
    Morada varchar(30),  
    Idade int,  
    Genero char  
)
```



1 ✖ describe cliente;

Result Grid | Filter Rows: | Export:

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	HULL	
Nome	varchar(30)	NO		HULL	
Morada	varchar(30)	YES		HULL	
Idade	int(9)	YES		HULL	
Genero	char(1)	YES		HULL	



Data Definition Language – DDL

REGRAS DE INTERIDADE DE DOMÍNIO: DEFAULT

- A cláusula **Default** permite especificar um valor padrão para o campo.

EXEMPLO:

```
CREATE TABLE Cliente (  
    ID int,  
    Nome varchar(30),  
    Morada varchar(30) DEFAULT 'Luanda' ,  
    Idade int,  
    Genero char  
)
```

1 describe cliente;

Result Grid | Filter Rows: | Export:

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	
Nome	varchar(30)	NO		NULL	
Morada	varchar(30)	YES		NULL	
Idade	int(9)	YES		NULL	
Genero	char(1)	YES		NULL	



Data Definition Language – DDL

REGRAS DE INTERIDADE DE CHAVE PRIMÁRIA: PRIMARY KEY

- Usa-se a cláusula PRIMARY KEY para indicar a chave primária de uma determinada tabela.

EXEMPLO:

```
CREATE TABLE Cliente (  
    ID int PRIMARY KEY,  
    Nome varchar(30),  
    Morada varchar(30),  
    Idade int,  
    Genero char  
)
```



```
CREATE TABLE Cliente (  
    ID int,  
    Nome varchar(30),  
    Morada varchar(30),  
    Idade int,  
    Genero char,  
    PRIMARY KEY (ID)  
)
```

Problema: Como definir uma chave primária composta?

Para usar **Chaves primárias compostas** – Usa-se a segunda forma de definição, Separando os campos por vírgulas.

Nota: Adiciona-se **Auto_Increment** para gerar automaticamente a chave primária da table sempre que se inserir novo registro!

Exemplo: Id int **auto_increment** primary key;



Data Definition Language – DDL

REGRAS DE INTERIDADE DE CHAVE PRIMÁRIA: PRIMARY KEY

```
1 describe cliente;
```

Result Grid Filter Rows: Export: W					
Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	
Nome	varchar(30)	NO	UNI	NULL	
Morada	varchar(30)	YES		Luanda	
Idade	int(9)	YES		NULL	
Genero	char(1)	YES		NULL	



REGRAS DE INTERIDADE REFERENCIAL: FOREIGN KEY

- A cláusula **REFERENCES** permite fazer a validação de chaves estrangeiras. i.e, nos atributos referenciados apenas pode-se introduzir dados que existam na tabela onde os campos são chaves primárias.

EXEMPLO:

```
CREATE TABLE Encomenda(  
    NumEcomenda int,  
    Data date,  
    IDCliente int REFERENCES Cliente(ID)  
)
```



```
CREATE TABLE Encomenda(  
    NumEcomenda int,  
    Data date,  
    IDCliente int,  
    FOREIGN KEY (IDCliente) REFERENCES Cliente(ID)  
)
```



ALTERAÇÃO DE TABELAS

- Para a alteração da estrutura (acrescentar e eliminar campos/colunas e regras de integridade) de tabelas existentes na base de dados usa-se o comando **ALTER TABLE**. Que pode ser associado às cláusulas:
 - **ADD** – adicionar;
 - **MODIFY** – modificar;
 - **DROP** – eliminar;

Nota: Atenção ao alterar a estrutura de uma tabela que já contenha dados, pois podem correr perdas.



Data Definition Language – DDL

ALTERAÇÃO DE TABELAS: ADD

- Adicionar campo Email varchar(25) a tabela Cliente.

EXEMPLO: ALTER TABLE cliente ADD Email varchar(25);

```
CREATE TABLE Cliente (  
    ID int PRIMARY KEY,  
    Nome varchar(30),  
    Morada varchar(30),  
    Idade int,  
    Genero char  
)
```



```
CREATE TABLE Cliente (  
    ID int,  
    Nome varchar(30),  
    Morada varchar(30),  
    Idade int,  
    Genero char,  
    PRIMARY KEY (ID)
```

```
1  
2 • alter table cliente add Email varchar(25);  
3 • describe cliente;
```

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	
Nome	varchar(30)	NO	UNI	NULL	
Morada	varchar(30)	YES		Luanda	
Idade	int(9)	YES		NULL	
Genero	char(1)	YES		NULL	
Email	varchar(25)	YES		NULL	

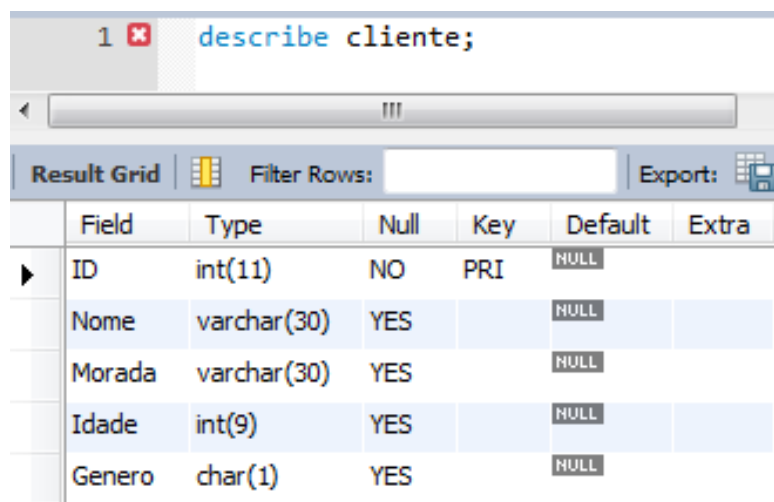


ALTERAÇÃO DE TABELAS: MODIFY

- Alterar o domínio do campo Género da tabela Cliente para Varchar (10).

EXEMPLO: **ALTER TABLE** cliente **MODIFY** Genero varchar(10);

ANTES



Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	
Nome	varchar(30)	YES		NULL	
Morada	varchar(30)	YES		NULL	
Idade	int(9)	YES		NULL	
Genero	char(1)	YES		NULL	

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	
Nome	varchar(30)	NO	UNI	NULL	
Morada	varchar(30)	YES		Luanda	
Idade	int(9)	YES		NULL	
Genero	varchar(10)	YES		NULL	
Email	varchar(25)	YES		NULL	



Data Definition Language – DDL

ALTERAÇÃO DE TABELAS: DROP

- Eliminar o campo Email da tabela Cliente;

EXEMPLO: ALTER TABLE cliente DROP Email;

ANTES

DEPOIS

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	
Nome	varchar(30)	NO	UNI	NULL	
Morada	varchar(30)	YES		Luanda	
Idade	int(9)	YES		NULL	
Genero	char(1)	YES		NULL	
Email	varchar(25)	YES		NULL	

```
5 • ALTER TABLE cliente DROP Email;  
6 • describe cliente;
```

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	
Nome	varchar(30)	NO	UNI	NULL	
Morada	varchar(30)	YES		Luanda	
Idade	int(9)	YES		NULL	
Genero	varchar(10)	YES		NULL	



INSERÇÃO DE DADOS - INSERT

- O comando INSERT permite realizar a introdução de registos numa tabela.

Sintaxe: **INSERT INTO** nome_da_tabela(campo1, ..., campoN)**VALUES**(valor1,...,valorN);

Ou **INSERT INTO** nome_da_tabela **VALUES**(valor1,...,valorN);

Nota: Quando se pretende inserir valores em todos os campos da tabela, pode-se usar o formato abreviado, mas a ordem dos valores terá que ser a mesma ordem das colunas.

EXEMPLO: **INSERT INTO** Cliente(ID, Nome, Morada, Idade, Genero)

VALUES(1, 'João', 'Prenda',38,'M');

Atenção:

- O número de campos existentes no comando INSERT tem que ser igual ao conjunto de valores na componente VALUES;
- O domínio dos dados a ser inserido tem que ser igual ao domínio definido para cada campo;
- Os valores presentes na componente VALUES devem corresponder a cada um dos campos definidos, ou então, a ordem e conjunto dos campos existentes na tabela.



VISUALIZAÇÃO DE DADOS INSERIDOS - SELECT

- Toda a consulta em uma base de dados relacional faz-se através do comando **SELECT**.
- O formato mínimo (mais básico) do comando **SELECT** exige que se indique quais colunas a selecionar e de que tabelas estas fazem parte.

Sintaxe: **SELECT** campo1, campo2, ..., campoN **FROM** tabela1, tabela2, tabelaN;

EXEMPLO: **SELECT** ID, Nome, Morada, Idade, Gen **FROM** Cliente;

Ou **SELECT** * **FROM** Cliente;

Obs: No caso de se pretender selecionar todos os campos de uma determinada tabela, usa-se o caracter asterisco *.



ATUALIZAÇÃO DE DADOS - UPDATE

- O comando **UPDATE** permite alterar os valores já existentes nos campos de uma tabela.

Sintaxe:

```
UPDATE nome_da_tabela  
SET nome_campo1 = {expressão, query},  
    ...  
    nome_campoN = {expressão, query},  
[WHERE condição]
```

Nota: O comando **UPDATE** pode conter a cláusula **WHERE**, que permite restringir o conjunto dos registos que serão afectados pelo comando.

EXEMPLO:

```
UPDATE Cliente  
SET Nome = 'Pedro'  
WHERE ID = 1
```

Atenção: Se a cláusula WHERE não for utilizada todos os registos sofrerão alterações.



ELIMINAÇÃO DE DADOS - DELETE

- O comando DELETE permite eliminar registos ou conjunto de registos de uma tabela.

Sintaxe: **DELETE FROM** nome_da_tabela [**WHERE** condição];

Nota: Tal como o comando **UPDATE**, o comando **DELETE** pode conter a cláusula **WHERE**, que permite restringir o conjunto dos registos que serão eliminados.

EXEMPLO: **DELETE FROM** Cliente **WHERE** ID = 1;

Atenção:

- Se a cláusula **WHERE** não for utilizada todos os registos da tabela serão eliminados;
- O comando **DELETE** elimina os registos de uma tabela, a tabela continua a existir;
- O comando **DELETE** elimina registos completos, não se eliminam valores de campos.



EXERCÍCIOS



INSTITUTO DE TELECOMUNICAÇÕES

MUITO OBRIGADO!