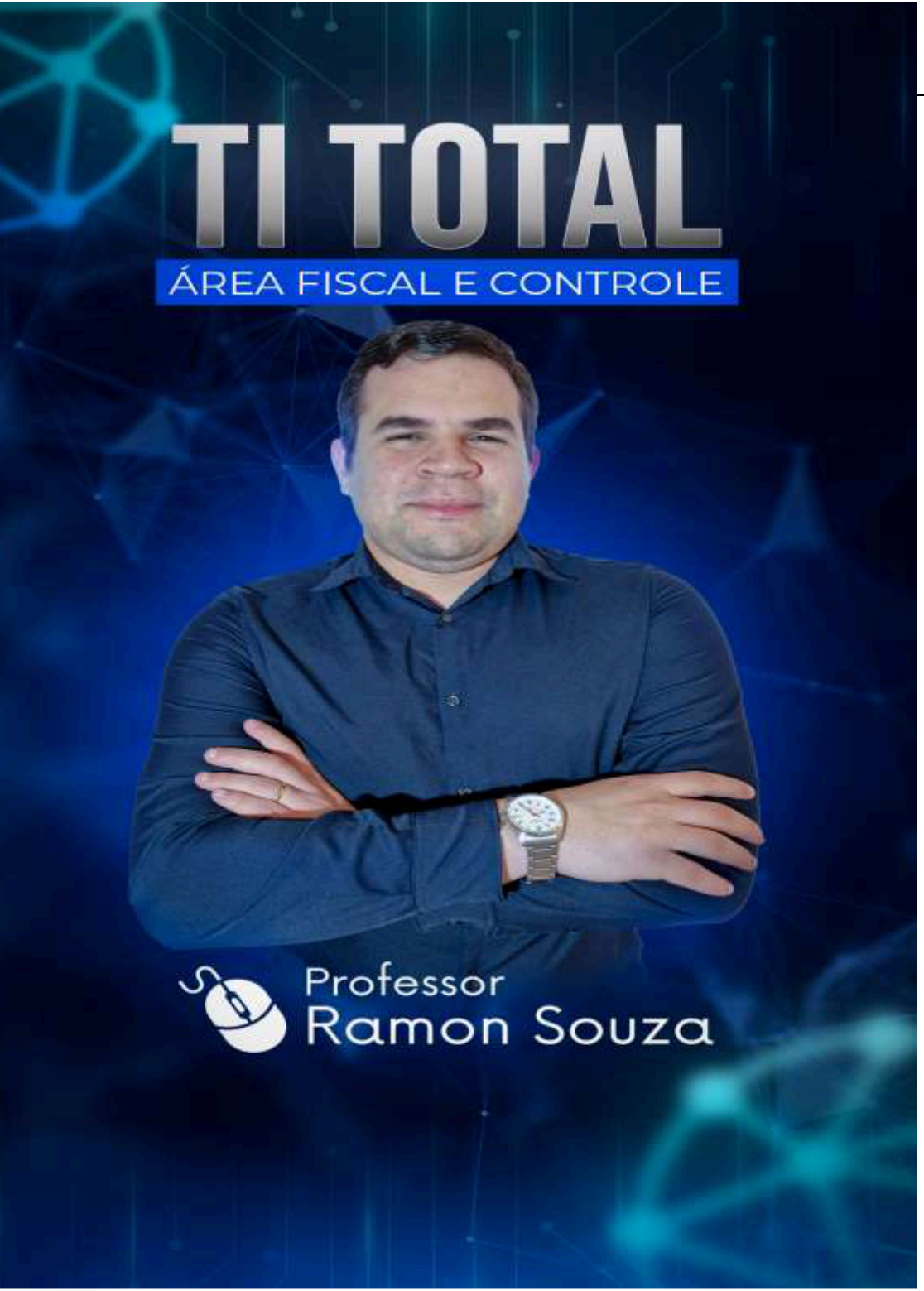
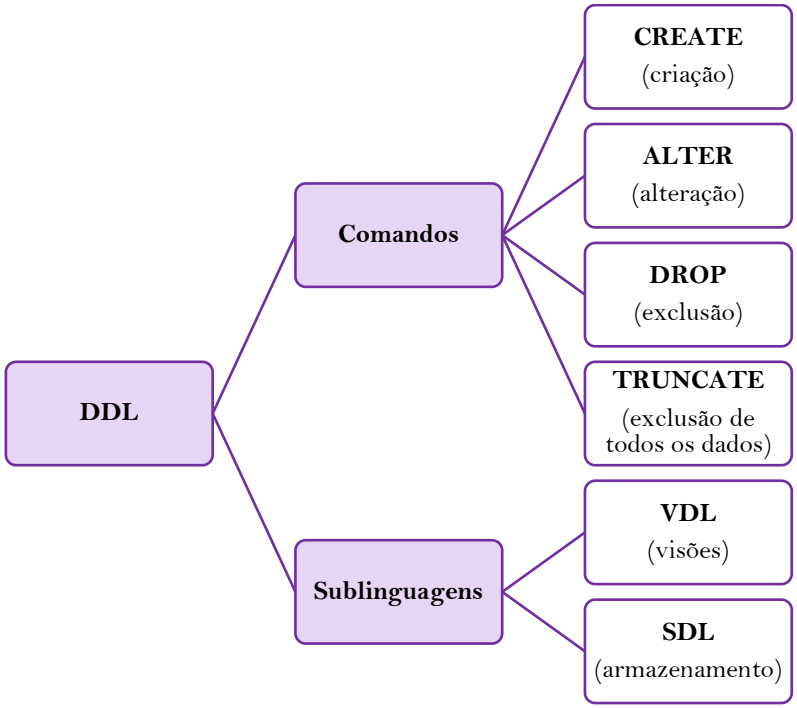


Tecnologia da Informação

RESUMO

SQL (DDL)

INTRODUÇÃO À DDL



TRABALHANDO COM BANCO DE DADOS

A instrução **CREATE DATABASE** é usada para **criar um banco de dados**.

```
CREATE DATABASE nome_do_banco;
```

A instrução **SHOW DATABASES** **lista os bancos** de dados existentes.

```
SHOW DATABASES;
```

A instrução **DROP DATABASE** é usada para **deletar um banco** de dados existente.

```
DROP DATABASE nome_do_banco;
```

TRABALHANDO COM TABELAS

A instrução **CREATE TABLE** é usada para **criar uma nova tabela** no banco de dados. A sintaxe básica dessa instrução é:

```
CREATE TABLE nome_da_tabela (
  coluna1 tipo_de_dado,
  coluna2 tipo_de_dado,
  ....
);
```

A instrução **ALTER TABLE** é usada para **adicionar, deletar ou modificar colunas em uma tabela existente**. Essa instrução também pode ser utilizada para adicionar ou deletar restrições a esta tabela.

Para **adicionar uma coluna**, usamos a cláusula **ADD**:

ALTER TABLE nome_da_tabela

ADD nome_da_coluna tipo_de_dado;

Para **modificar uma coluna**, usamos a cláusula **ALTER COLUMN** (SQL Server/Access) **ou MODIFY COLUMN** (MySQL/Oracle até antes do 10G) **ou MODIFY** (Oracle 10G e superiores):

ALTER TABLE nome_da_tabela

ALTER COLUMN nome_da_coluna tipo_de_dado;

OU

ALTER TABLE nome_da_tabela

MODIFY COLUMN nome_da_coluna tipo_de_dado;

OU

ALTER TABLE nome_da_tabela

MODIFY nome_da_coluna tipo_de_dado;

Para **deletar uma coluna**, usamos a cláusula **DROP COLUMN**:

ALTER TABLE nome_da_tabela

DROP COLUMN nome_da_coluna;

A instrução **DROP TABLE** é usada para **deletar uma tabela existente**.

DROP TABLE nome_da_tabela;

Essa instrução irá deletar todos os dados da tabela, bem como a própria tabela.

Contudo, você pode desejar **excluir apenas os dados da tabela, sem excluir a estrutura dessa tabela**. Para isso, poderá usar o comando **TRUNCATE**:

TRUNCATE TABLE nome_da_tabela;

RESTRIÇÕES

As seguintes restrições são comumente usadas no SQL:

- **NOT NULL**: Garante que uma coluna não pode ter um valor NULL.
- **UNIQUE**: Garante que todos os valores em uma coluna sejam diferentes.
- **PRIMARY KEY**: Uma combinação de NOT NULL e UNIQUE. Identifica exclusivamente cada linha em uma tabela.
- **FOREIGN KEY**: Identifica exclusivamente uma linha / registro em outra tabela.
- **CHECK**: Garante que todos os valores em uma coluna satisfaçam uma condição específica.
- **DEFAULT**: Define um valor padrão para uma coluna quando nenhum valor é especificado.
- **INDEX**: Usado para criar e recuperar dados do banco de dados muito rapidamente.

NOT NULL

```
CREATE TABLE nome_da_tabela (
   coluna1 tipo_de_dado NOT NULL,
   coluna2 tipo_de_dado,
   ....
);
```

OU

```
ALTER TABLE nome_da_tabela
MODIFY coluna2 tipo_de_dado NOT NULL;
```

UNIQUE

Na cláusula CREATE TABLE:

```
CREATE TABLE nome_da_tabela (
   coluna1 tipo_de_dado UNIQUE,
...
);
```

OU

```
CREATE TABLE nome_da_tabela (
   coluna1 tipo_de_dado,
UNIQUE (coluna1)
);
```

OU

```
CREATE TABLE nome_da_tabela (
   coluna1 tipo_de_dado,
   coluna2 tipo_de_dado,
CONSTRAINT nome_da_restricao UNIQUE (coluna1, coluna2));
);
```

Na cláusula ALTER TABLE:

```
ALTER TABLE nome_da_tabela
ADD UNIQUE (coluna);
```

OU

```
ALTER TABLE nome_da_tabela
ADD CONSTRAINT nome_da_restricao UNIQUE (coluna1, coluna2));
```

Para excluir uma restrição UNIQUE:

```
ALTER TABLE nome_da_tabela
DROP INDEX nome_da_restricao;
```

OU

```
ALTER TABLE nome_da_tabela
DROP CONSTRAINT nome_da_restricao;
```

PRIMARY KEY

Na cláusula CREATE TABLE:

```
CREATE TABLE nome_da_tabela (
   coluna1 tipo_de_dado PRIMARY KEY,
...
);
```

OU

```
CREATE TABLE nome_da_tabela (
   coluna1 tipo_de_dado,
PRIMARY KEY (coluna1)
);
```

OU

```
CREATE TABLE nome_da_tabela (
   coluna1 tipo_de_dado,
   coluna2 tipo_de_dado,
CONSTRAINT nome_da_restricao PRIMARY KEY (coluna1, coluna2));
);
```

Na cláusula ALTER TABLE:

```
ALTER TABLE nome_da_tabela
ADD PRIMARY KEY (coluna);
```

OU

```
ALTER TABLE nome_da_tabela
ADD CONSTRAINT nome_da_restricao PRIMARY KEY (coluna1, coluna2));
```

Para excluir uma restrição PRIMARY KEY:

```
ALTER TABLE nome_da_tabela
DROP PRIMARY KEY;
```

OU

```
ALTER TABLE nome_da_tabela
DROP CONSTRAINT nome_da_restricao;
```

FOREIGN KEY

Na cláusula CREATE TABLE:

```
CREATE TABLE nome_da_tabela (
   coluna1 tipo_de_dado PRIMARY KEY,
   coluna2 tipo_de_dado FOREIGN KEY REFERENCES tabela_referenciada(chave),
);
```

OU

```
CREATE TABLE nome_da_tabela (
   coluna1 tipo_de_dado,
   coluna2 tipo_de_dado,
PRIMARY KEY (coluna1),
FOREIGN KEY (coluna2) REFERENCES tabela_referenciada (chave)
);
```

OU

```
CREATE TABLE nome_da_tabela (
   coluna1 tipo_de_dado,
   coluna2 tipo_de_dado,
   coluna3 tipo_de_dado,
CONSTRAINT nome_da_restricao FOREIGN KEY (coluna1, coluna2))
REFERENCES tabela_referenciada (chave1, chave2););
```

Na cláusula ALTER TABLE:

```
ALTER TABLE nome_da_tabela
ADD FOREIGN KEY(coluna) REFERENCES tabela_referenciada (chave);
```

OU

```
ALTER TABLE nome_da_tabela
ADD CONSTRAINT nome_da_restricao FOREIGN KEY(coluna1, coluna2) REFERENCES tabela_referenciada (chave1, chave2);
```

Para excluir uma restrição FOREIGN KEY:

```
ALTER TABLE nome_da_tabela
DROP FOREIGN KEY coluna;
```

OU

```
ALTER TABLE nome_da_tabela
DROP CONSTRAINT nome_da_restricao;
```

CHECK

Na cláusula CREATE TABLE:

```
CREATE TABLE nome_da_tabela (
   coluna1 tipo_de_dado,
   coluna2 tipo_de_dado CHECK (condicao),
   coluna3 tipo_de_dado,
);
```

OU

```
CREATE TABLE nome_da_tabela (
   coluna1 tipo_de_dado,
   coluna2 tipo_de_dado,
   coluna3 tipo_de_dado,
   CHECK (condicao)
);
```

OU

```
CREATE TABLE nome_da_tabela (
   coluna1 tipo_de_dado,
   coluna2 tipo_de_dado,
   coluna3 tipo_de_dado,
CONSTRAINT nome_da_restricao CHECK (condicao1 AND condicao2));
);
```

Na cláusula ALTER TABLE:

```
ALTER TABLE nome_da_tabela
ADD CHECK (condicao);
```

OU

```
ALTER TABLE nome_da_tabela
ADD CONSTRAINT nome_da_restricao CHECK (condicao1 AND condicao2));
```

Para excluir uma restrição CHECK:

```
ALTER TABLE nome_da_tabela
DROP CHECK nome_da_restricao;
```

OU

```
ALTER TABLE nome_da_tabela
DROP CONSTRAINT nome_da_restricao;
```

DEFAULT

Na cláusula CREATE TABLE:

```
CREATE TABLE nome_da_tabela (
   coluna1 tipo_de_dado,
   coluna2 tipo_de_dado DEFAULT valor,
   coluna3 tipo_de_dado,
);
```

Na cláusula ALTER TABLE:

```
ALTER TABLE nome_da_tabela
ALTER coluna SET DEFAULT valor;
```

OU

```
ALTER TABLE nome_da_tabela
ADD CONSTRAINT nome_da_restricao DEFAULT valor;
```

OU

```
ALTER TABLE nome_da_tabela
MODIFY coluna DEFAULT valor;
```

Para excluir uma restrição DEFAULT:

```
ALTER TABLE nome_da_tabela
ALTER coluna DROP DEFAULT;
```

OU

```
ALTER TABLE nome_da_tabela
ALTER COLUMN coluna DROP DEFAULT;
```

TRABALHANDO COM VISÕES

A sintaxe a seguir é utilizada para **criar uma view** em SQL:

```
CREATE VIEW [Nome da View] AS
  SELECT Coluna1, Coluna2,…
  FROM nome_da_tabela
  WHERE...;
```

Uma visão pode ser **atualizada** com o comando **CREATE OR REPLACE VIEW**:

```
CREATE OR REPLACE VIEW [Nome da View] AS
  SELECT Coluna1, Coluna2,…
  FROM nome_da_tabela
  WHERE...;
```

Uma visão é **apagada** com o comando **DROP VIEW**:

```
DROP VIEW [Nome da View];
```

TRABALHANDO COM ÍNDICES

A sintaxe a seguir é utilizada para **criar um índice** em SQL:

CREATE INDEX nome_do_indice

ON nome_da_tabela (coluna1, coluna2, ...);

Também é possível criar um índice único, isto é, em que não são permitidos valores duplicados. Para isso, utiliza-se CREATE UNIQUE INDEX:

CREATE UNIQUE INDEX nome_do_indice

ON nome_da_tabela (coluna1, coluna2, ...);

Um índice pode ser **atualizado** com o comando **ALTER INDEX**:

ALTER INDEX nome_do_indice

ON nome_da_tabela (coluna1, coluna2, ...);

Um índice pode ser **excluído** com o comando **DROP INDEX**:

DROP INDEX nome_do_indice**;**

OU

DROP INDEX nome_da_tabela.nome_do_indice;

TEMA AVANÇADO: PROCEDURES

Uma **STORED PROCEDURE** é um **código SQL preparado que você pode salvar, para que o código possa ser reutilizado repetidamente**.

Para criar uma **PROCEDURE**, basta utilizar a seguinte sintaxe:

CREATE PROCEDURE nome_da_procedure

AS

declaracoes_SQL

GO;

Após criar uma **PROCEDURE**, você pode executá-la simplemesnte executando uma chamada com a cláusula **EXEC**:

EXEC nome_da_procedure

TEMA AVANÇADO: TRIGGERS

Triggers ou gatilhos são **programas armazenados que são executados ou disparados automaticamente quando alguns eventos ocorrem**.

A sintaxe para definição de triggers varia de acordo com o SGDB. Vamos ver um exemplo de sintaxe considerando o Oracle: **PROCEDURE**

```
CREATE [OR REPLACE] TRIGGER nome_da_trigger
{BEFORE | AFTER | INSTEAD OF}
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF coluna]
ON tabela
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condicao)
DECLARE
  Declaration-statements
BEGIN
  Executable-statements
EXCEPTION
  Exception-handling-statements
END;
```

TEMA AVANÇADO: FUNCTIONS

Funções ou Functions são **rotinas que retornam valores ou tabelas**. Com elas você poderá construir visões parametrizadas ou ainda construir suas próprias funções.

A sintaxe para definição de functions varia de acordo com o SGDB. Vamos ver um exemplo de sintaxe considerando o Oracle:

```
CREATE [OR REPLACE] FUNCTION nome_da_funcao
[(nome_do_parametro [IN | OUT | IN OUT] tipo [, ...])]
RETURN return_tipo_de_dados
{IS | AS}
BEGIN
  < corpo da função >
END [nome_da_funcao];
```

PROCEDURES X TRIGGERS X FUNCTIONS

PROCEDURE	TRIGGER	FUNCTION
Código SQL preparado que você pode salvar, para que o código possa ser reutilizado repetidamente	Programas armazenados que são executados ou disparados automaticamente quando alguns eventos ocorrem.	Rotinas que retornam valores ou tabelas.