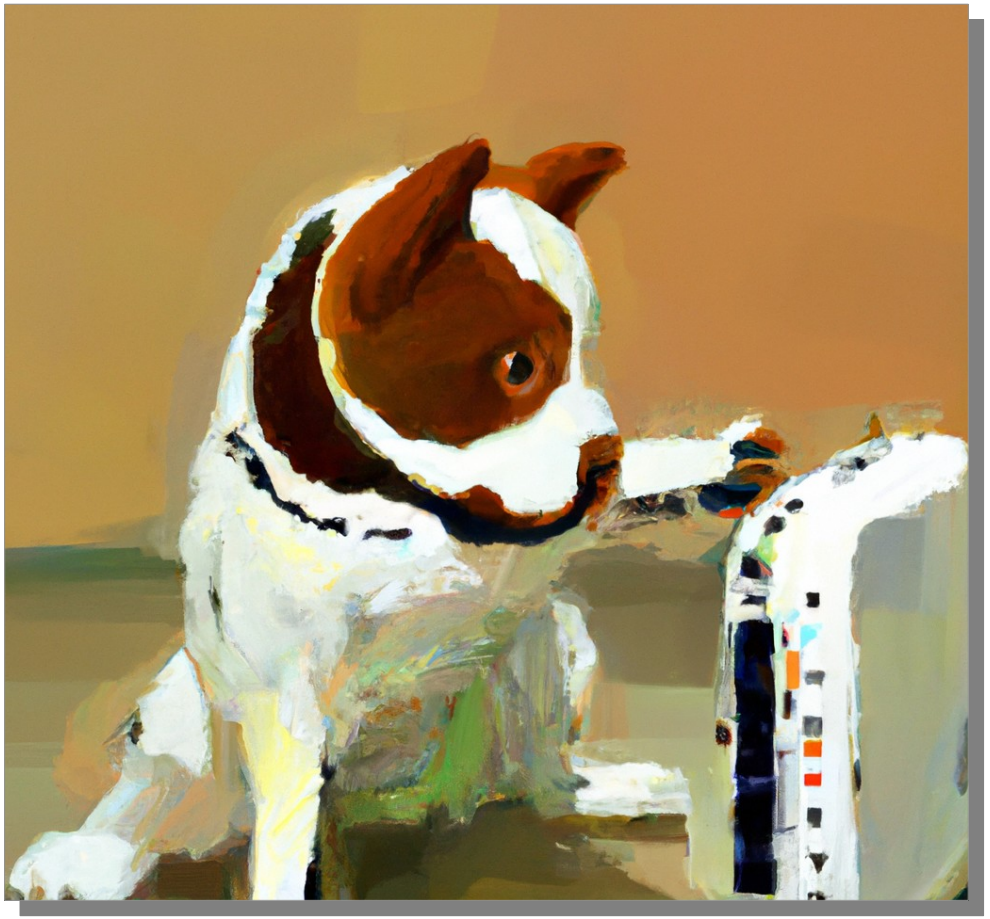
Universidade de Brasília
Departamento de Ciência da Computação

Bancos de Dados

CIC0097



Prof. Pedro Garcia Freitas

https://pedrogarcia.gitlab.io/

pedro.garcia@unb.br

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciências da Computação

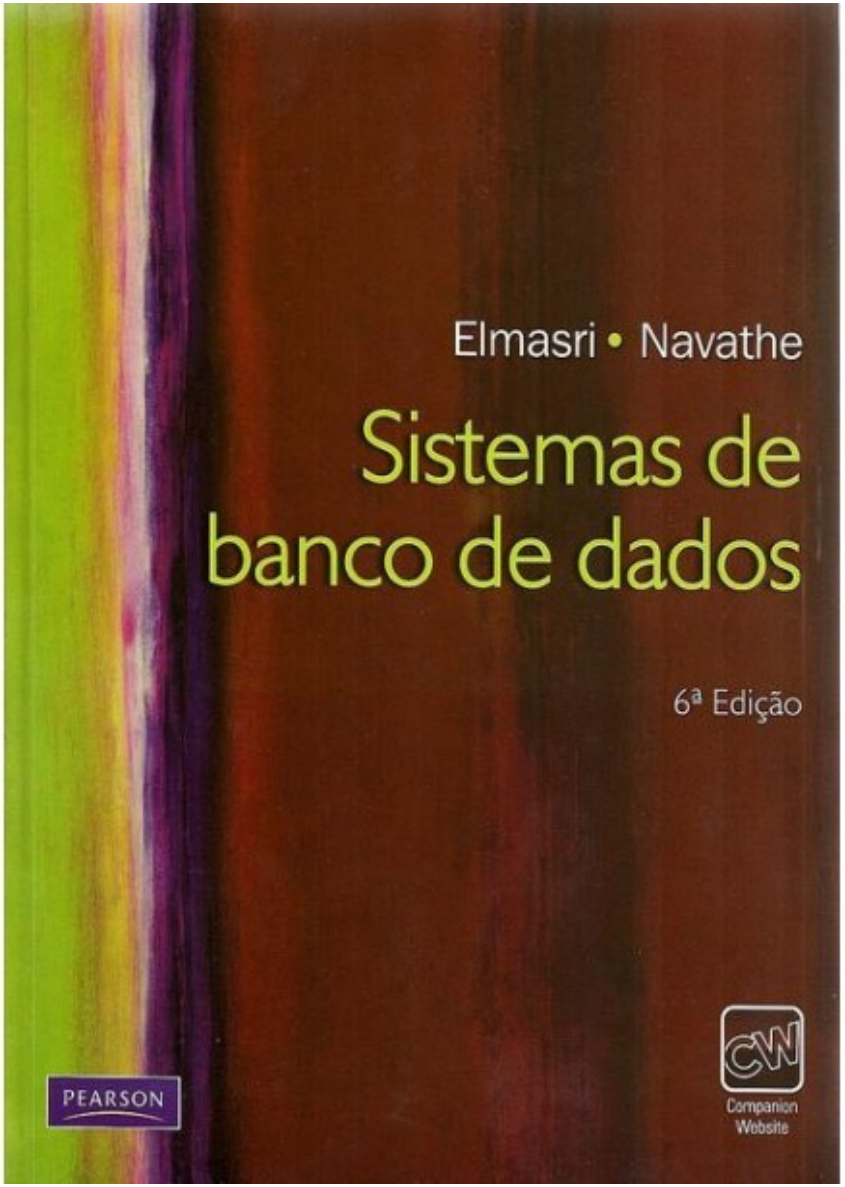
Este conjunto de slides não deve ser utilizado ou republicado sem a expressa permissão do autor.

This set of slides should not be used or republished without the author's express permission.

Módulo 13

Linguagem de Consulta Estruturada Parte 2: Introdução ao SQL, DDL, DML

CIC0097/2023.1
T1/T2



Esta aula se baseia no Capítulo 4 (SQL básica) do Elmasri e Navathe (6ª Edição).

1. Objetivos

Esta aula apresenta os conceitos iniciais sobre linguagem de definição de dados do SQL. Serão apresentados os comandos **`create`**, **`alter`**, **`drop`**, **`insert`**, **`delete`** e **`update`**.

2. SQL

- Linguagem de banco de dados relacionais com recursos de **definição** de dados, **consulta** de dados e **manipulação** de dados.
- Permite especificar restrições que devem ser impostas aos dados possibilitanto a implementação da integridade e segurança da informação armazenada.

2. SQL

- SQL é uma linguagem padrão estabelecida pelo International Standards Organization (ISO).
  - "**ISO/IEC 9075** "Information technology - Database languages - SQL"
- Alguns produtos comerciais estendem a sintaxe padrão do SQL, mas geralmente incluem apenas pequenas alterações no ISO/IEC 9075.

2. SQL

- (A padronização da) SQL pode ser considerada um dos principais motivos para o sucesso dos bancos de dados relacionais comerciais.
- Como ela se tornou um padrão para esse tipo de bancos de dados, ela permitiu a interoperacionalidade entre diferentes SGBDs relacionais.

2. SQL

- A interoperabilidade entre diferentes SGBDs favorece a migração e desacoplamento dos dados com uma dada solução.
- Isso garante que, mesmo que se um usuário estiver insatisfeito com um produto de SGBD relacional em particular, a conversão para outro produto de SGBD relacional não é tão cara ou demorada, pois os dois sistemas seguem os mesmos padrões de linguagem.

2. SQL

- Outra vantagem da padronização é que os usuários podem escrever comandos em um programa de aplicação de banco de dados que pode acessar dados armazenados em dois ou mais SGBDs relacionais sem ter de mudar a sublinguagem de banco de dados (SQL).

2. SQL

- SQL: Structured Query Language (Linguagem de Consulta Estruturada).
- Foi criada e implementada na IBM Research como a interface para um sistema de banco de dados relacional experimental, chamado SYSTEM R .
  - “R” foi a linguagem criada por Edgar F. Codd, matemático criador da álgebra relacional.

2. SQL

- SQL é uma linguagem de banco de dados abrangente: tem instruções para definição de dados, consultas e atualizações. Logo, ela é uma linguagem completa, atuando como *Data Query Language* (DQL), *Data Manipulation Language* (DML), *Data Definition Language* (DDL), and *Data Control Language* (DCL).

3. Linguagens do Banco de Dados

- O sistema precisa oferecer linguagens e interfaces apropriadas para cada categoria de usuário.
- Em muitos SGBDs, a chamada linguagem de definição de dados (DDL - *Data Definition Language*), é usada pelos projetistas de banco de dados para definir os esquemas.

3. Linguagens do Banco de Dados

- O SGBD terá um compilador da DDL cuja função é processar instruções da DDL a fim de identificar as descrições dos construtores de esquema e armazenar a descrição de esquema no catálogo do SGBD.
- Nos SGBDs, que mantêm uma separação clara entre os níveis conceitual e interno, a DDL é usada para especificar apenas o esquema conceitual.

3. Linguagens do Banco de Dados

- Outra linguagem, a linguagem de definição de armazenamento (SDL — *Storage Definition Language*), é utilizada para especificar o esquema interno.
- Na maioria dos SGBDs relacionais, **não existe uma linguagem padrão que realiza o papel de SDL**, sendo específica da implementação interna do SGBD.

3. Linguagens do Banco de Dados

- Quando os esquemas são compilados e o banco de dados é populado, os usuários precisam de alguma forma de manipulá-lo.
- As manipulações típicas incluem, inserção, recuperação, exclusão e modificação dos dados.
  - Do inglês, ***c****reate*, ***r****ead*, ***u****pdate*, and ***d****elete* (CRUD).

3. Linguagens do Banco de Dados

- O SGBD oferece um conjunto de operações ou uma linguagem chamada linguagem de manipulação de dados (DML — Data Manipulation Language) para essas finalidades.
- No SQL, DML e DDL não são considerados linguagens distintas, mas formam uma linguagem integrada e abrangente.

3. Linguagens do Banco de Dados

- Embora Elmasri e Navathe classifique o SQL em DDL e DML, outros autores subdividem essa linguagem em mais sublinguagens:
  - **DQL: Data Query Language** (Linguagem de Consulta de Dados) para consulta de dados.
  - **DML: Data Manipulation Language** (Linguagem de Manipulação de Dados) para edição de dados.
  - **DDL: Data Definition Language** (Linguagem de Definição de Dados) para estruturação de dados.
  - **DCL: Data Control Language** (Linguagem de Controle de Dados) para administrar o banco de dados.

Fonte: https://learnsql.com/blog/what-is-dql-ddl-dml-in-sql/

3. Linguagens do Banco de Dados

Data Query Language (DQL) - A *Linguagem de Consulta de Dados* é a sublinguagem responsável por ler, ou consultar, dados de um banco de dados.

- Em SQL, isso corresponde ao comando **`SELECT`**.

3. Linguagens do Banco de Dados

Data Manipulation Language (DML) - A *Linguagem de Manipulação de Dados* é a sublinguagem responsável por adicionar, editar ou excluir dados de um banco de dados.

- Em SQL, isso corresponde aos comandos **INSERT**, **UPDATE** e **DELETE**.

3. Linguagens do Banco de Dados

Data Definition Language (DDL) - A *Linguagem de Definição de Dados* é a sublinguagem responsável por definir a forma como os dados são estruturados em um banco de dados.

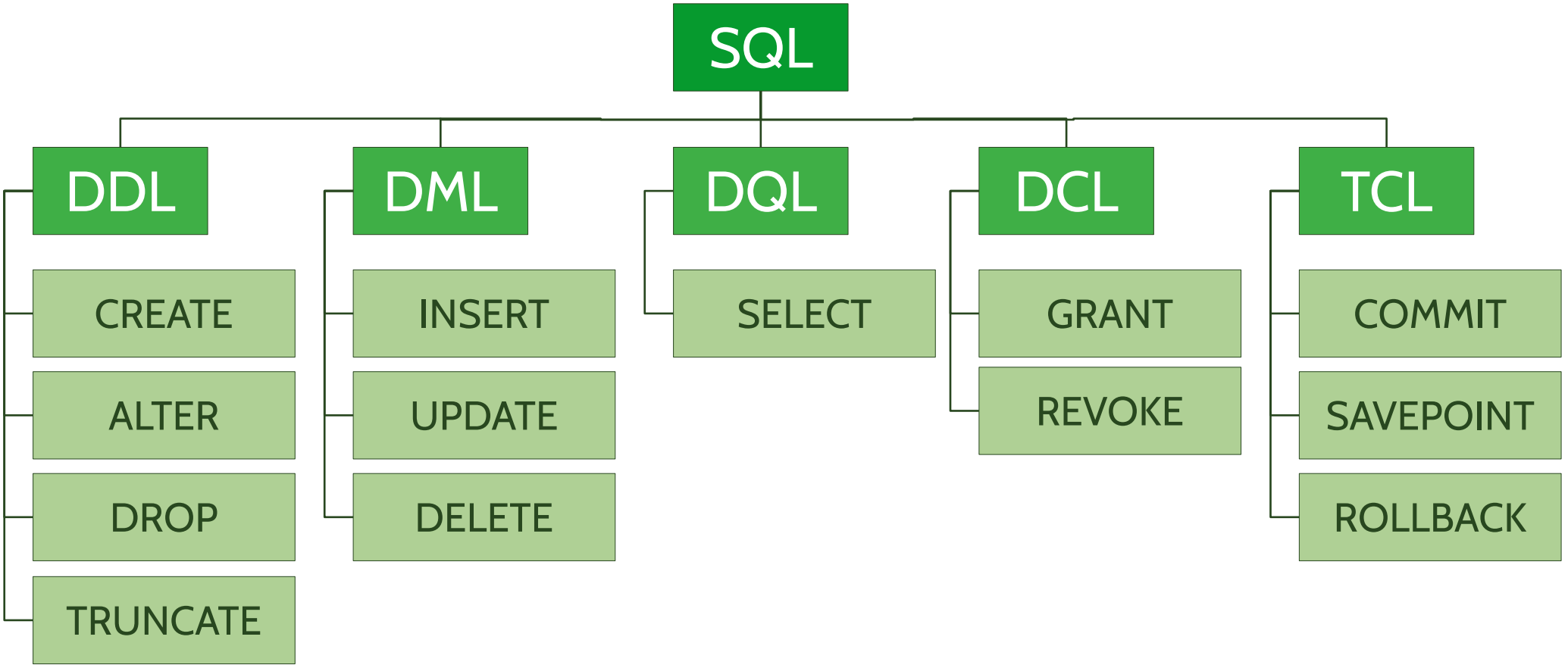
- Em SQL, isso corresponde à manipulação de tabelas por meio dos comandos **`CREATE`**, **`ALTER TABLE`** e **`DROP TABLE`**.

3. Linguagens do Banco de Dados

Data Control Language (DCL) - A *Linguagem de Controle de Dados* é a sublinguagem responsável pelas tarefas administrativas de controle do próprio banco de dados, sendo mais notável a **concessão e revogação de permissões** de banco de dados para usuários.

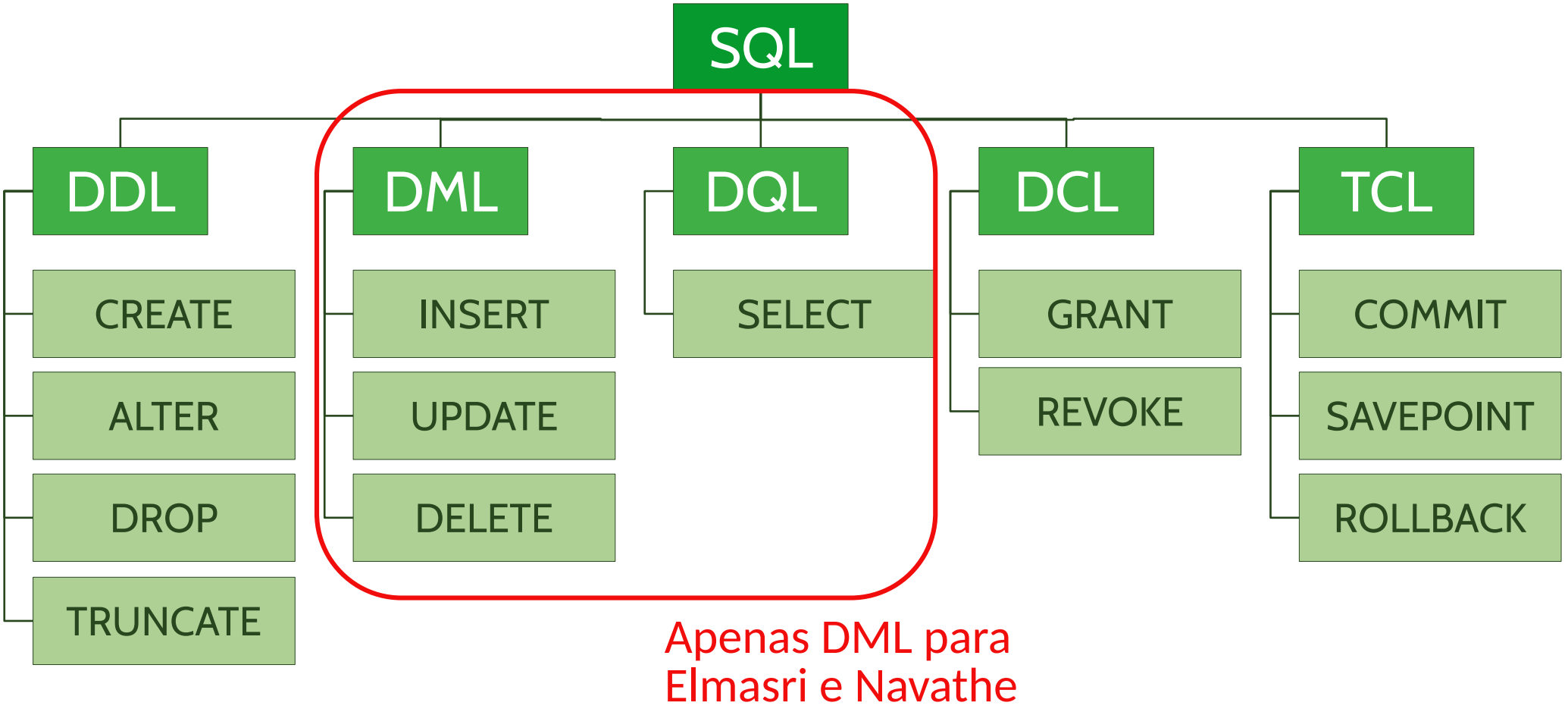
- Em SQL, isso corresponde aos comandos **GRANT**, **REVOKE** e **DENY**, entre outros.

3. Linguagens do Banco de Dados

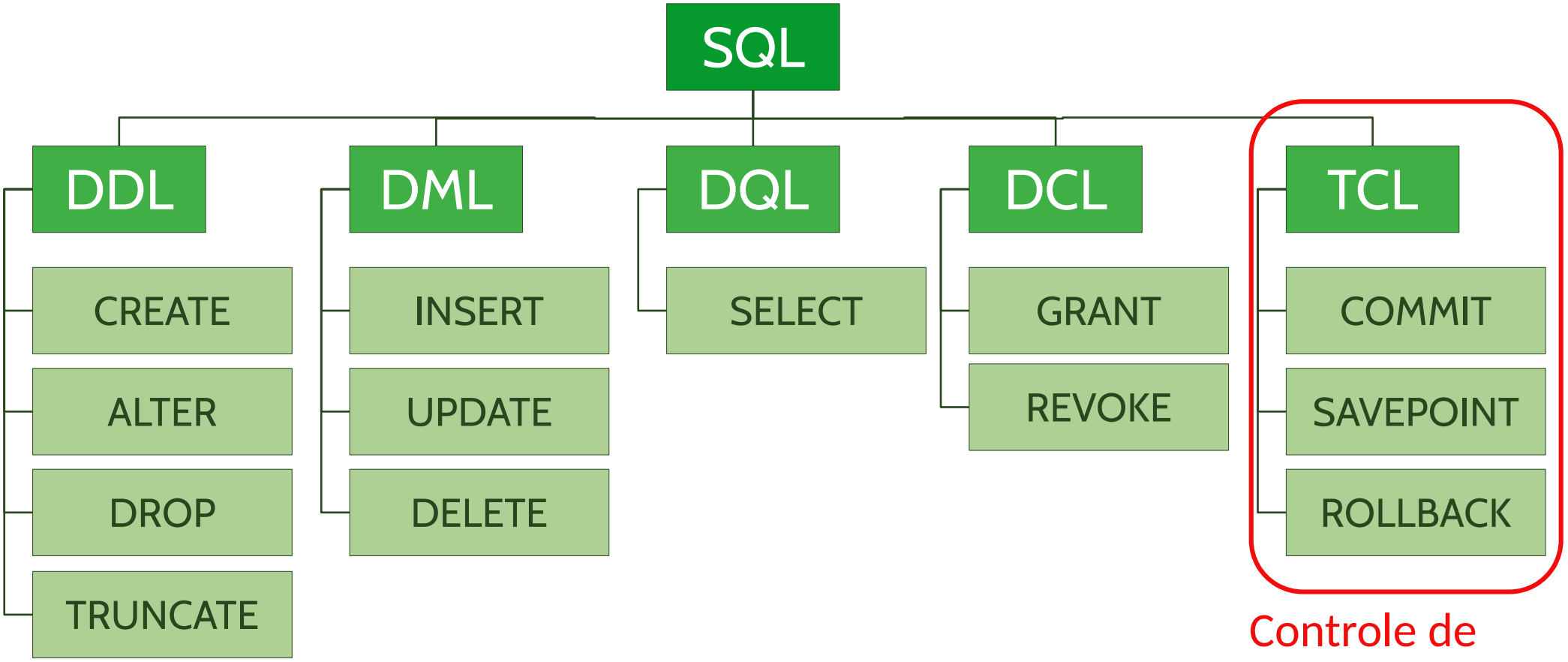


Fonte: https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/

3. Linguagens do Banco de Dados



3. Linguagens do Banco de Dados



Controle de transações.

DDL

CREATE, ALTER, DROP

4. DDL: O Comando CREATE

O comando **CREATE** permite criar objetos de banco de dados:

- **CREATE SCHEMA**
- **CREATE DOMAIN**
- **CREATE TABLE**
- **CREATE INDEX**
- **CREATE VIEW**
- **CREATE TRIGGER**
- **. . .**

4. DDL:O Comando CREATE

O comando CREATE DOMAIN permite declarar um **domínio** (tipo) especificando o seu nome para ser usado junto da criação de atributos.

- `CREATE DOMAIN custom_uint4 AS CHAR(4);`
- `CREATE DOMAIN meu_intero AS INTEGER CHECK (D_NUM > 0 AND D_NUM < 21)`

4. DDL: O Comando CREATE

- O comando **`CREATE TABLE`** é usado para especificar uma nova relação, dando-lhe um nome e especificando seus atributos e restrições iniciais.
- Os atributos são especificados primeiro, e **cada um deles** recebe um **nome, um tipo** de dado para especificar seu domínio de valores e **restrições de atributo** (e.g., **`NOT NULL`**).

4. DDL: O Comando CREATE

- **As restrições** de chave, integridade de entidade e integridade referencial **podem** ser especificadas na instrução **CREATE TABLE**, **depois** que os atributos forem declarado.

```
CREATE TABLE minha_tabela(
        atributos ...
        restrições ...
);
```

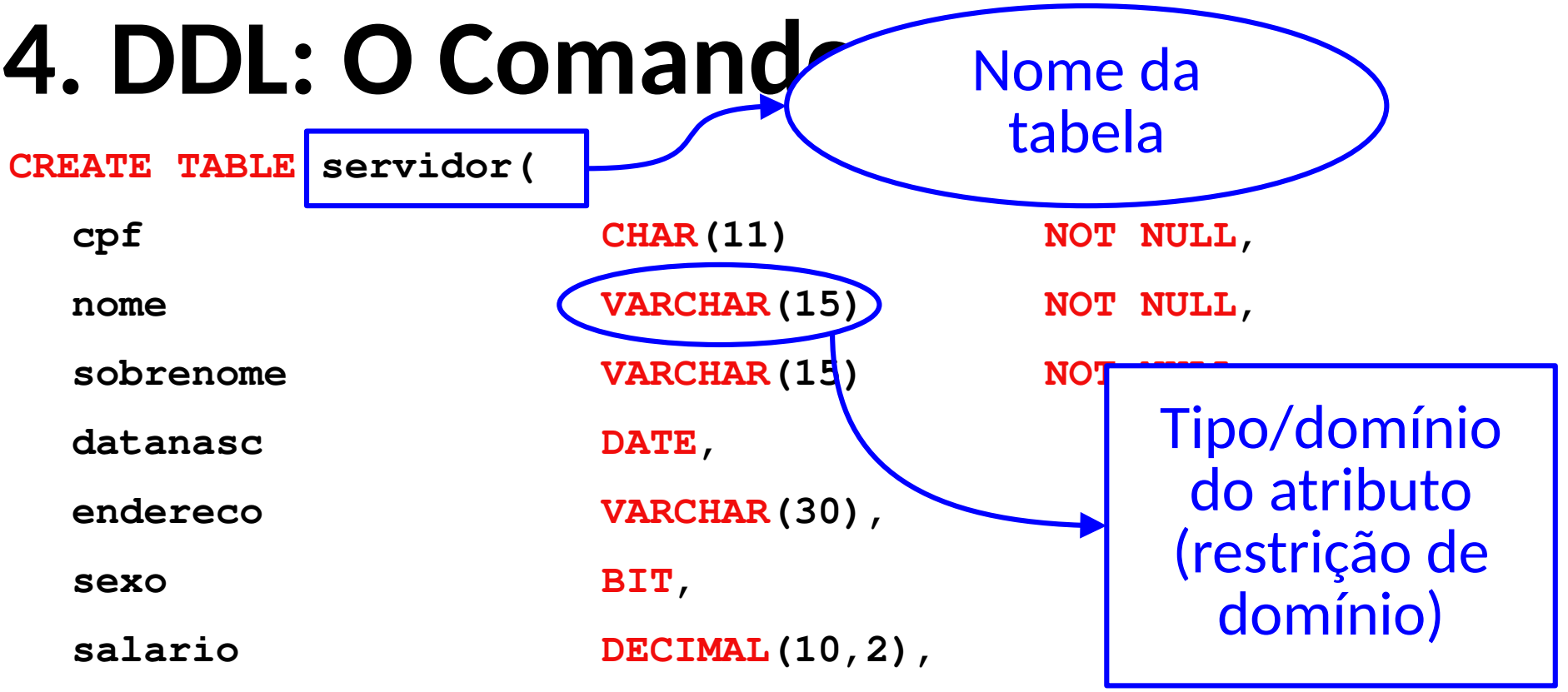
4. DDL: O Comando CREATE

```
CREATE TABLE servidor(
   cpf                  CHAR(11)          NOT NULL,
   nome                 VARCHAR(15)       NOT NULL,
   sobrenome            VARCHAR(15)       NOT NULL,
   datanasc             DATE,
   endereco             VARCHAR(30),
   sexo                 BIT,
   salario              DECIMAL(10,2),
   fk_cpf_supervisor    CHAR(11)          NOT NULL,
   fk_numero_dpto       INT,
   PRIMARY KEY(cpf),
   FOREIGN KEY (fk_cpf_supervisor) REFERENCES servidor(cpf),
   FOREIGN KEY (fk_numero_dpto) REFERENCES Departamento(numero)
);
```

4. DDL: O Comando


```
CREATE TABLE servidor(
   cpf                  CHAR(11)          NOT NULL,
   nome                 VARCHAR(15)       NOT NULL,
   sobrenome            VARCHAR(15)       NOT NULL,
   datanasc             DATE,
   endereco             VARCHAR(30),
   sexo                 BIT,
   salario              DECIMAL(10,2),
   fk_cpf_supervisor    CHAR(11)          NOT NULL,
   fk_numero_dpto       INT,
   PRIMARY KEY(cpf),
   FOREIGN KEY (fk_cpf_supervisor) REFERENCES servidor(cpf),
   FOREIGN KEY (fk_numero_dpto) REFERENCES Departamento(numero)
);
```

4. DDL: O Comando




```
CREATE TABLE servidor(
    cpf                 CHAR(11)            NOT NULL,
    nome                VARCHAR(15)         NOT NULL,
    sobrenome           VARCHAR(15)         NOT NULL,
    datanasc            DATE,
    endereco            VARCHAR(30),
    sexo                BIT,
    salario             DECIMAL(10,2),
    fk_cpf_supervisor   CHAR(11)            NOT NULL,
    fk_numero_dpto      INT,
    PRIMARY KEY(cpf),
    FOREIGN KEY (fk_cpf_supervisor) REFERENCES servidor(cpf),
    FOREIGN KEY (fk_numero_dpto) REFERENCES Departamento(numero)
);
```

4. DDL: O Comando

```
CREATE TABLE servidor(
    cpf                 CHAR(11)        NOT NULL,
    nome                VARCHAR(15)     NOT NULL,
    sobrenome           VARCHAR(15)     NOT NULL,
    datanasc            DATE,
    endereco            VARCHAR(30),
    sexo                BIT,
    salario             DECIMAL(10,2),
    fk_cpf_supervisor   CHAR(11)        NOT NULL,
    fk_numero_dpto      INT,
    PRIMARY KEY(cpf)
    FOREIGN KEY (fk_cpf_supervisor) REFERENCES servidor(cpf),
    FOREIGN KEY (fk_numero_dpto) REFERENCES Departamento(numero)
);
```

Nome da tabela

Tipo/domínio do atributo (restrição de domínio)

Restrição de chave

4. DDL: O Comando CREATE

```
CREATE TABLE servidor(
   cpf                  CHAR(11)          NOT NULL,
   nome                 VARCHAR(15)       NOT NULL,
   sobrenome            VARCHAR(15)       NOT NULL,
   datanasc             DATE,
   endereco             VARCHAR(30),
   sexo                 CHAR,
   salario              DECIMAL(10,2),
   fk_cpf_supervisor    CHAR(11)          NOT NULL,
   fk_numero_dpto       INT,
   PRIMARY KEY(cpf),
   FOREIGN KEY (fk_cpf_supervisor) REFERENCES servidor(cpf),
   FOREIGN KEY (fk_numero_dpto) REFERENCES Departamento(numero)
);
```

Chave primária

4. DDL: O Comando CREATE

```
CREATE TABLE servidor(
   cpf                  CHAR(11)          NOT NULL,
   nome                 VARCHAR(15)       NOT NULL,
   sobrenome            VARCHAR(15)       NOT NULL,
   datanasc             DATE,
   endereco             VARCHAR(30),
   sexo                 BIT,
   salario              DECIMAL(10,2),
   fk_cpf_supervisor    CHAR(11)          NOT NULL,
   fk_numero_dpto       INT,
   PRIMARY KEY(cpf),
   FOREIGN KEY (fk_cpf_supervisor) REFERENCES servidor(cpf),
   FOREIGN KEY (fk_numero_dpto) REFERENCES Departamento(numero)
);
```

Chaves estrangeiras

5. DDL: O Comando `ALTER`

- O comando **`ALTER`** é usado para alterar definições e restrições de tabelas e objetos já criados no banco de dados.

```
ALTER TABLE minha_tabela
  especificações de alterações
```

5. DDL: O Comando ALTER

```
CREATE TABLE servidor(

...                          ...                          ...
cpf                          CHAR(11)                     NOT NULL,
...
);
CREATE TABLE departamento(

numero                       INT                          NOT NULL,
...
);

ALTER TABLE servidor
   ADD CONSTRAINT id_fk_num_dpto
-  FOREIGN KEY (fk_numero_dpto) REFERENCES Departamento(numero)
```

5. DDL: O Comando ALTER

- Outros exemplos:

```
ALTER TABLE servidor
   ALTER fk_cpf_supervisor
      SET DEFAULT 12312345678
ALTER TABLE servidor
   DROP endereco CASCADE;
ALTER TABLE servidor
   DROP endereco RESTRICT;
```

5. DDL: O Comar

- Outros exemplos:

```
ALTER TABLE servid
   ALTER fk_cpf_supervi
      SET DEFAULT 123123    78

ALTER TABLE servidor
   DROP endereco CASCADE;

ALTER TABLE servidor
   DROP endereco RESTRICT;
```

A alteração (no caso, **DROP**) vai ser propagada pra todos os outros objetos que dependem deste atributo

5. DDL: O Comar

- Outros exemplos:

```
ALTER TABLE servid
   ALTER fk_cpf_sup
      SET DEFAULT 12
ALTER TABLE servid
   DROP endereco CASCADE;
ALTER TABLE servidor
   DROP endereco RESTRICT;
```

A alteração (no caso, **DROP**) não vai ser propagada, i.e., o SGBD não permitirá a remoção se houver objetos que dependem do atributo a ser removido.

6. DDL: O Comando DROP

O comando **DROP** no SQL é um comando DDL que remove permanentemente os dados (ou tabela existente) do banco de dados e libera o espaço da memória.

- `DROP TABLE dependente CASCADE;`
- `DROP DATABASE database_name;`
- `DROP TABLE IF EXISTS table_name;`

DML
INSERT, UPDATE, DELETE

7. DML: O Comando `INSERT`

O comando **`INSERT`** insere uma tupla/linha em uma tabela/relação. É possível usar esse programa de duas maneiras:

- `INSERT INTO table_name (column1, column2, column3) VALUES (value1, value2, value3);`
- `INSERT INTO table_name VALUES (value1, value2, value3);`

7. DML: O Comando INSERT

Funcionário(cpf,salario,imposto,tempo_serviço)

- **INSERT INTO Funcionario (cpf, salario, imposto, tempo_servico) VALUES (01234567891, 1000, 20, 3);**
- **INSERT INTO Funcionario VALUES (01234567891, 1000, 20, 3);**

7. DML: O Comando INSERT

Funcionário(cpf,salario,imposto,tempo_serviço)

Especificar ambos nomes de atributos e valores (não depende de ordem)

```
• INSERT INTO Funcionario (cpf,
  salario, imposto, tempo_servico)
  VALUES (01234567891, 1000, 20, 3);
```

```
• INSERT INTO Funcionario
  VALUES (01234567891, 1000, 20, 3);
```

Especificar apenas os valores (depende da ordem das colunas conforme definida na criação da tabela).

7. DML: O Co

Essa sintaxe permite que nem todos atributos sejam especificados no tempo de inserção.

Funcionário(cp

Especificar ambos nomes de atributos e valores (n e de ordem)

- **INSERT INTO Funcionario (cpf, salario, imposto, tempo_servico) VALUES (01234567891, 1000, 20, 3);**

- **INSERT INTO Funcionario VALUES (01234567891, 1000, 20, 3);**

Especificar apenas os valores (depende da ordem das colunas conforme definida na criação da tabela).

7

Não existe um comando SQL padrão para trocar a ordem das colunas, mas alguns SGBDs fornecem extensões do SQL. Por exemplo, em MySQL:

```
ALTER TABLE Dependentes MODIFY COLUMN
nome VARCHAR(200) AFTER Relacionamento;
```

em)

```
VALUE                    7891, 1000, 20, 3);
```

```
• INSERT INT  Funcionario
  VALUES (01234567891, 1000, 20, 3);
```

Especificar apenas os valores (depende da ordem das colunas conforme definida na criação da tabela).

8. DML: O Comando `UPDATE`

O comando **`UPDATE`** modifica os valores dos atributos de uma ou mais tuplas de uma relação:

- ```
  UPDATE table_name
  SET column1=value1, column2=value2
  WHERE condition;
  ```
  ```

8. DML: O Comando UPDATE

Funcionário(cpf,salario,imposto,tempo_serviço)

- ```
  UPDATE Funcionario
  SET salario=salario*1.1
  WHERE salario < 2000;
  ```
  ```

8. DML: O Comando UPDATE

Funcionário(cpf,salario,imposto,tempo_serviço)

- UPDATE Funcionario
  SET salario=salario*1.1
  WHERE salario < 2000;

Novos valores

8. DML: O Comando UPDATE

Funcionário(cpf,salario,imposto,tempo_serviço)

- **UPDATE** Funcionario
  SET salario=salario*1.1
  WHERE salario < 2000;

Novos valores

Condição que indica quais tuplas devem ter seus valores modificados!

9. DML: O Comando `DELETE`

O comando **`DELETE`** remove linhas de uma tabela.

- **`DELETE FROM`** `table_name`
  `WHERE` `condition;`

9. DML: O Comando DELETE

Funcionário(cpf,salario,imposto,tempo_serviço)

- DELETE FROM Funcionário;
- DELETE FROM Funcionário
  WHERE cpf='01234567890';
- DELETE FROM Funcionário
  WHERE salario > 2000;

9. DML: O Comando DELETE

Funcionário(cpf,salario,imposto,tempo_serviço)

- `DELETE FROM Funcionário;`
- `DELETE FROM Funcionário`
  `WHERE cpf='01234567890';`
- `DELETE FROM Funcionário`
  `WHERE salario > 2000;`

Remove **todas** as linhas!

9. DML: O Comando DELETE

Funcionário(cpf,salario,imposto,tempo_serviço)

- `DELETE FROM Funcionário;`
- `DELETE FROM Funcionário WHERE cpf='01234567890';`
- `DELETE FROM Funcionário WHERE salario > 2000;`

Remove **apenas uma** tupla específica!

9. DML: O Comando DELETE

Funcionário(cpf,salario,imposto,tempo_serviço)

- DELETE FROM Funcionário;
- DELETE FROM Funcionário
  WHERE cpf='01234567890';
- DELETE FROM Funcionário
  WHERE salario > 2000;

Remove **várias** tuplas que atendem a tal condição!

Este material está disponível em: https://github.com/pedrogarciafreitas/slides-bd-2023.1

Dúvidas?