DDL
Linguagem de definição de dados

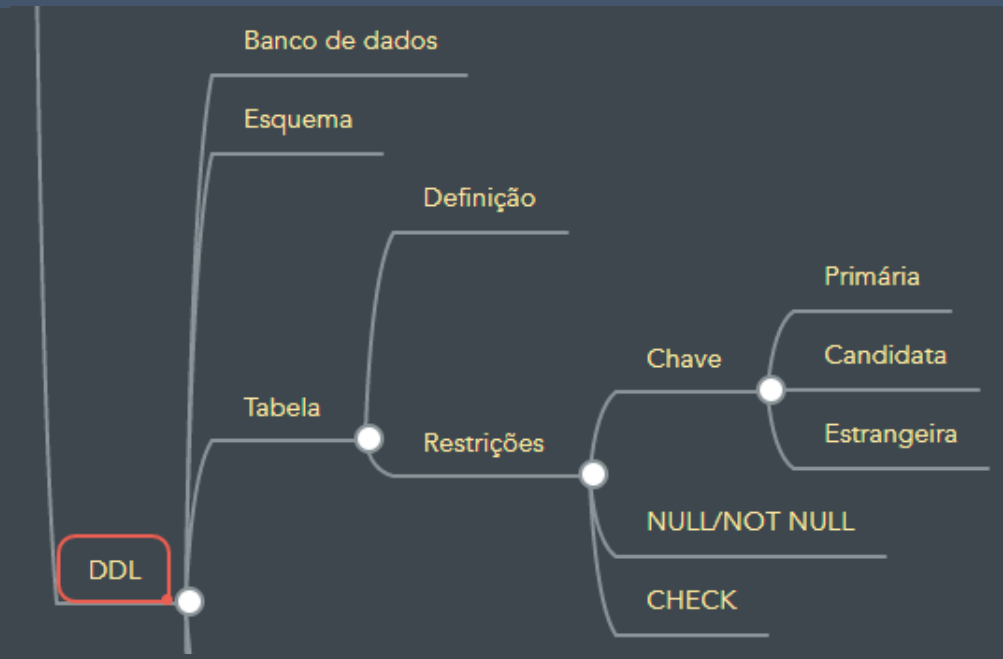
BANCO DE DADOS

VANESSA BORGES

vanessa.a.borges@ufms.br

Abrangência da DDL

Abrangência da DDL



Definição de banco de dados
CREATE DATABASE

- **Especifica um novo banco de dados relacional**

Conjunto de esquemas S = {R1, R2,..., Rn}
Conjunto de restrições de integridade RI

- **Sintaxe:**

```
CREATE DATABASE <nome>
[WITH OWNER = <dono_do_banco_de_dados> ]
[ENCODING = <codificação>];
```

Normalmente, o criador se torna o dono do novo banco de dados

- **Exemplo:**

```
CREATE DATABASE empresadb;
```

Definição de banco de dados
DROP DATABASE

- Remover a definição de um banco de dado

  DROP DATABASE [**IF EXISTS**] <nome_do_database>

  [**CASCADE** | **RESTRICT**];

  - Pode ser executado somente pelo dono do banco de dados
  - É necessário que não existam usuários conectados no banco de dados no momento que o comando é executado

  - IF EXISTS
    - Não retorna erro se o banco de dados não existir
  - CASCADE
    - Remove um banco de dados, incluindo todos os seus esquemas, tabelas e outros elementos
  - RESTRICT
    - Remove um banco de dados somente se não existirem elementos definidos

Elementos do Esquema

- O conceito de um esquema SQL foi incorporado a partir do SQL2 com o objetivo de **agrupar tabelas e outros elementos referentes ao banco de dados de uma aplicação**
  - Um banco de dados pode conter várias aplicações sobre ele. Contudo, é possível criar um esquema para cada aplicação sobre o mesmo banco de dados
  - **Os principais elementos de um esquema:**
    - Tabela
    - Restrições
    - Trigger
    - Visões
    - Procedimentos
    - Funções
    - Índices

Esquema e Conceitos de Catálogo em SQL

- **Integração entre os esquemas:**
  - Restrições de **integridade referencial** podem ser definidas sobre duas tabelas **apenas se elas estão contidas em esquemas do mesmo catálogo**.
  - **Esquemas pertencentes ao mesmo catálogo** podem compartilhar elementos como **tabela**, **funções**, e **definições de domínio** dentre outros.

Definição de esquema
CREATE SCHEMA

SCHEMA – coleção de objetos de banco de dados relacional.

- O comando utilizado para **criar** um Esquema possui a seguinte sintaxe:

```
CREATE SCHEMA [IF NOT EXISTS] <nome_do_esquema>
AUTHORIZATION <nome_grupo|nome_usuário>;
```

- O comando a seguir **cria** um esquema denominado EMPRESA cujo **dono** é o usuário ‘Jsmith’.

```
CREATE SCHEMA EMPRESA AUTHORIZATION ‘Jsmith’;
```

Definição de esquema
[ALTER | DROP] SCHEMA

- Sintaxe para alterar a definição de um esquema

ALTER SCHEMA <nome_do_esquema> **RENAME TO** <novo_nome_do_esquema>;
ALTER SCHEMA <nome_do_esquema> **OWNER TO** <novo_nome_grupo|novo_nome_usuário>;

- Sintaxe para remover a definição de um esquema

DROP SCHEMA [IF EXISTS] <nome_do_esquema> [**CASCADE** | **RESTRICT**];

  - **IF EXISTS:**
    - Não retorna um erro se o schema não existir
  - **CASCADE**
    - Remove um esquema de BD, incluindo todas as suas tabelas e os seus outros elementos
  - **RESTRICT**
    - Remove um esquema de BD somente se não existirem elementos definidos para esse esquema

Definição de tabela

- **Tabelas da base (relações da base)**
  - A relação e suas tuplas são realmente criadas e armazenadas como um arquivo pelo SGBD
- **Tabelas temporárias**
  - As tabelas temporárias são automaticamente removidas no final da sessão ou, opcionalmente, no final da transação corrente
- **Relações virtuais**
  - Criadas por meio da instrução CREATE VIEW

Definição de tabela CREATE TABLE

Comando utilizado para definir uma nova relação considerando seus atributos e restrições.

```
CREATE [TEMPORARY|TEMP] TABLE <nome da tabela>
( [..., <atributo> <tipo> [NULL| NOT NULL] [restrição do atributo],
[CONSTRAINT <nome da restrição de tabela> <tipo da restrição de tabela>, ...]);
```

Exemplo:

```
CREATE TABLE cliente (
<declaração dos atributos, tipos e restrições>
nome      VARCHAR(20),
cpf       VARCHAR(14),
… ,
<declaração das restrições de tabela>
CONSTRAINT pk_cliente PRIMARY KEY (cliid),
CONSTRAINT unique_cpf UNIQUE (cpf),
…) ;
```

Linguagens – DDL – Tipos de Dados

- **Um SGBD suporta vários tipos de dados.**
  - Tipo numérico:
    - INTEGER, INT, SMALLINT, BIGINT, FLOAT, REAL, DOUBLE, DECIMAL, NUMERIC, PRECISION
  - Cadeia de caractere:
    - CHAR, CHARACTER, VARCHAR, CHARACTER VARYING, CHARACTER LARGE OBJECT (CLOB)
  - Cadeia de bits:
    - BIT, BIT VARYING, BINARY LARGE OBJECT (BLOB)
  - Booleanos:
    - TRUE, FALSE, NULL, UNKNOWN
  - Data:
    - DATE, TIME, DATETIME
  - Outros
    - TIMESTAMP, INTERVAL, etc

Remoção de tabela
DROP TABLE

Comando utilizado para remover uma tabela e seus “objetos”

- Não pode ser excluída a tabela que possui alguma referência. Neste caso, deve-se primeiro excluir a tabela que possui algum campo que a está referenciando e depois excluir a tabela inicial.

- Sintaxe:

DROP TABLE **<nome da tabela>**;

- Exemplo:

```
-- Remove a tabela departamento
DROP TABLE Departamento;
```

Restrições de integridade

- **Garantem que as mudanças feitas no banco de dados por usuários autorizados não resultam em uma perda de consistência de dados**

- São definidas em expressões ou comandos que são escritos uma vez e guardados na BD e serão executados como resposta a certos eventos

- Em SQL-92 há maneira de especificar restrições de integridade como parte de um esquema de BD:
  - Restrições de chave;
  - Restrições de integridade referencial;
  - Restrições de domínio;

Restrições de chave

- Restringem os valores permitidos em determinados atributos de relações

- São especificadas como parte da instrução CREATE TABLE
  - **Chaves candidatas** são especificadas usando **UNIQUE**
    - Podem existir várias declarações UNIQUE mas só uma chave primária
  - **Chaves primárias** são especificadas usando **PRIMARY KEY**
    - Especifica um ou mais atributos que compõem a chave primária da relação

Restrições de chave UNIQUE

Atributo UNIQUE

UNIQUE (A1, A2, ... , An)

- A especificação **UNIQUE** diz que o atributo A1, ... An formam uma **chave candidata**, ou seja, duas tuplas na relação **não** podem ser iguais em todos os atributos listados
- Atributos de chave candidata **podem ser NULL**, a menos que tenham sido declarados explicitamente como NOT NULL

Restrições de chave PRIMARY KEY

- **Chaves primárias** são especificadas usando **PRIMARY KEY**
  - Especifica um ou mais atributos que compõem a chave primária da relação
  - PRIMARY KEY = UNIQUE + NOT NULL

```
CREATE TABLE departamento (
    dnome varchar(255) NOT NULL,
    dnumero integer PRIMARY KEY,
    cpf_gerente varchar(11),
    data_inicio_gerente date
);

CREATE TABLE departamento (
    dnome varchar(255) NOT NULL,
    dnumero integer,
    cpf_gerente varchar(11),
    data_inicio_gerente date,
    CONSTRAINT pkdnumero PRIMARY KEY(dnumero)
);
```

Restrições de chave
PRIMARY KEY – Chave composta

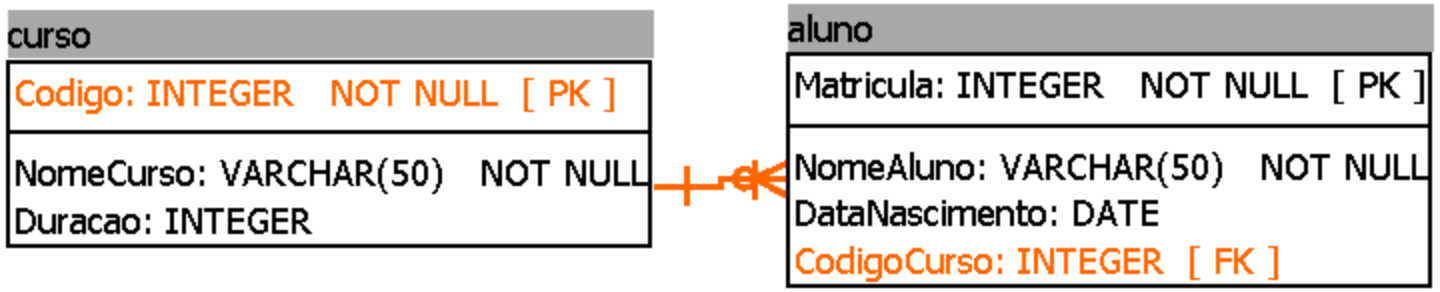
- **Chaves primárias** compostas de **mais de um atributo**

```
CREATE TABLE dependente (
    fcpf varchar(11),
    nome_dependente varchar(255),
    sexo character(1),
    datanasc date,
    parentesco varchar(255),
    PRIMARY KEY (fcpf, nome_dependente)
);
```

```
CREATE TABLE dependente (
    fcpf varchar(11) PRIMARY KEY,
    nome_dependente varchar(255) PRIMARY KEY,
    sexo character(1),
    datanasc date,
    parentesco varchar(255)
);
```

Restrição de integridade referencial

- **Integridade referencial** garante que um valor que aparece em uma relação para determinado conjunto de atributos também aparecerá para certo conjunto de atributos em outra relação.
- Para isso **chaves estrangeiras (FOREING KEY)** são especificadas como parte do comando **CREATE TABLE**



Restrição de integridade referencial
CREATE TABLE ... PRIMARY KEY ... FOREIGN KEY

Integridade referencial

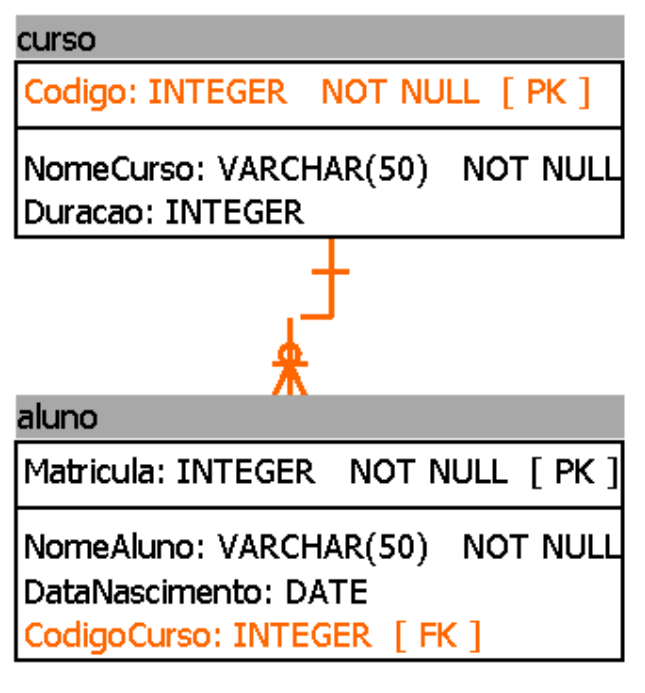
- Comando que cria uma tabela considerando suas restrições de chave estrangeira
- A integridade referencial é especificada por meio da cláusula **FOREIGN KEY**

CREATE TABLE [<nome do esquema>.]<nome da tabela>...
FOREIGN KEY (<coluna_estr>1[,...,<coluna_estr>n])
REFERENCES <tabela_ref>([<coluna_ref>[,...,<coluna_ref]])
[ON DELETE <acao_ref>] [ON UPDATE <acao_ref>]

<acao_ref>

- NO ACTION ou RESTRICTED → **impede** a ação na tabela mestre <tabela_ref>
- CASCADE → **propaga** a ação da tabela mestre
- SET NULL → valores de referencias alterados para nulo
- SET DEFAULT → valores de referencias alterados para default

Restrição de integridade referencial




```
CREATE TABLE curso (
Codigo INTEGER PRIMARY KEY,
NomeCurso VARCHAR(50) NOT NULL,
Duracao INTEGER
);

CREATE TABLE aluno (
Matricula INTEGER PRIMARY KEY,
NomeAluno VARCHAR(50) NOT NULL,
DataNascimento DATE,
CodigoCurso INTEGER,
FOREIGN KEY (CodigoCurso)
REFERENCES curso(Codigo)
ON UPDATE NO ACTION  ON DELETE NO ACTION
);
```

- Para cada aluno que for cadastrado, poderá ter um código de curso cadastrado na tabela CURSO.
- O comando REFERENCES está referenciando o campo CodigoCurso da tabela ALUNO, que é chave estrangeira, que aponta para o campo Código da tabela CURSO, que é a chave primária de CURSO.
- A partir da criação desse relacionamento, ao tentarmos cadastrar um aluno com um código de curso que não esteja cadastrado na tabela CURSO, haverá restrição.

Restrições baseadas em atributos CHECK

Atributo CHECK

- A restrição de check (verificação) especifica uma condição que precisa ser satisfeita por cada tupla da relação.
- O predicado de uma cláusula CKECK pode ser uma subconsulta SQL

```
CREATE TABLE departamento (
    dnome varchar(255) NOT NULL,
    dnumero integer PRIMARY KEY,
    cpf_gerente varchar(11),
    data_inicio_gerente date,
    CONSTRAINT chdnumero CHECK (dnumero>0 AND dnumero<20)
);
```

Verifica se o atributo dnumero é maior que zero e menor que 20