Ciência da Computação

**GBC043 Sistemas de Banco de Dados**

# SQL
# Linguagem de Definição de Dados

Profa. Maria Camila Nardini Barioni

camila.barioni@ufu.br

Bloco B - sala 1B137

1° semestre de 2024

# SQL DDL

- CREATE SCHEMA
  - cria um esquema de BD relacional
- DROP SCHEMA
  - remove um esquema de BD relacional

# CREATE SCHEMA

CREATE SCHEMA schema_name [ AUTHORIZATION user_name ] [ schema_element [ ... ] ]

- Cria um esquema de BD relacional
  - agrupa as tabelas e outros comandos que pertencem à mesma aplicação
  - identifica o proprietário do esquema
- Característica
  - o esquema inicial não possui tabelas/dados

# DROP SCHEMA

DROP SCHEMA [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT ]

- Remove um esquema de BD relacional
  - tabelas/dados
  - índices
  - arquivos de log

  } quaisquer elementos associados

- Usuários autorizados
  - proprietário do banco de dados
  - DBA ou usuário com privilégio de *root*

# DROP SCHEMA

- 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

# Por que é útil organizar o BD em esquemas?

- Permite o acesso do usuário aos objetos de qualquer esquema do banco de dados que ele está conectado (diferente do caso de diferentes banco de dados)
- Permite organizar os objetos do banco de dados em grupos lógicos, facilitando o gerenciamento
- Permite vários usuários utilizarem um mesmo banco de dados sem que um interfira no outro.

# Como definir um dado esquema como sendo o esquema corrente?

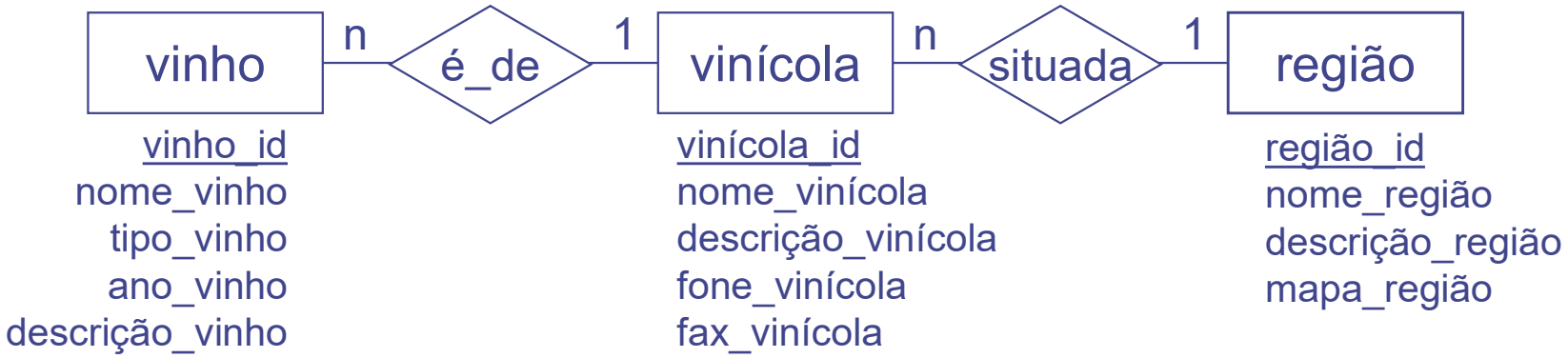
- O SGBD determina qual tabela deve ser utilizada de acordo com o *search path*, que é uma lista de esquemas que devem ser verificados
- Para colocar um novo esquema no caminho (*path*), deve-se usar:

SET search_path TO <esquema>

# SQL DDL

- CREATE TABLE
  - cria uma nova tabela (relação) no BD
  - a nova tabela não possui dados
- DROP TABLE
  - remove uma tabela (relação) e todas as suas instâncias do BD
- ALTER TABLE
  - altera a estrutura de uma tabela (relação) já existente no BD

# Exemplo




- região (região_id, nome_região, mapa_região, descrição_região)
- vinícola (vinícola_id, nome_vinícola, descrição_vinícola, fone_vinícola, fax_vinícola, região_id)
- vinho (vinho_id, nome_vinho, tipo_vinho, ano_vinho, descrição_vinho, vinícola_id)

vinícola (vinícola_id, nome_vinícola, descrição_vinícola, fone_vinícola, fax_vinícola, região_id)

```
CREATE TABLE vinícola (
   vinícola_id          smallint        NOT NULL    ,
   nome_vinícola        varchar(100)    NOT NULL    ,
   descrição_vinícola   xml                         ,
   fone_vinícola        varchar(15)                 ,
   fax_vinícola         varchar(15)                 ,
   região_id            smallint        NOT NULL    ,
   CONSTRAINT vinipk PRIMARY KEY (vinícola_id),
   CONSTRAINT vinifk FOREIGN KEY (região_id)
         REFERENCES região(região_id)
);
```

# CREATE TABLE

CREATE TABLE nome_tabela ( $A_1$ $D_1$ $R_1$,
$A_2$ $D_2$ $R_2$,
...
$A_n$ $D_n$ $R_n$ ) ;

- Cria uma nova tabela (relação)
- Cria os atributos da nova tabela, com
  - nome do atributo: $A_i$ ($1 \leq i \leq n$)
  - tipo de dado (domínio do atributo): $D_i$
  - restrições que atuam no atributo: $R_i$

# Exemplos de Tipos de Dados

- Lógico

**Table B-1.** *PostgreSQL Logical Data Type*

| SQL Name | PostgreSQL Alternative Name | Notes |
|---|---|---|
| `boolean` | `bool` | Holds a truth value. Will accept values such as `TRUE`, `'t'`, `'true'`, `'y'`, `'yes'`, and `'1'` as `true`. Uses 1 byte of storage, and can store `NULL`, unlike a few proprietary databases. |

- Fonte: Beginning databases with PostgreSQL: Matthew and Stones, 2nd ed. Apress

# Exemplos de Tipos de Dados

- Números exatos

**Table B-2.** *postgresql Exact Number Types*

| SQL Name | PostgreSQL Alternative Name | Notes |
|---|---|---|
| `smallint` | `int2` | A signed 2-byte integer that can store –32768 to +32767. |
| `integer, int` | `int4` | A signed 4-byte integer that can store –2147483648 to +2147483647. |
| `bigint` | `int8` | A signed 8-byte integer, giving approximately 18 digits of precision. |
| `bit` | `bit` | Stores a single bit, 0 or 1. To insert into a table, use syntax such as `INSERT INTO ... VALUES(B'1');`. |
| `bit varying` | `varbit(n)` | Stores a string of bits. To insert into a table, use syntax such as `INSERT INTO ... VALUES(B'011101');` . |

- Fonte: Beginning databases with PostgreSQL: Matthew and Stones, 2nd ed. Apress

# Exemplos de Tipos de Dados

- Números aproximados

**Table B-3.** *PostgreSQL Approximate Number Types*

| SQL Name | PostgreSQL Alternative Name | Notes |
|---|---|---|
| `numeric (precision, scale)` | | Stores an exact number to the precision specified. The user guide states there is no limit to the precision that may be specified. |
| `real` | `float4` | A 4-byte, single-precision, floating-point number. |
| `double precision` | `float8` | An 8-byte, double-precision, floating-point number. |
| `money` | | Equivalent to `numeric(9,2)`, storing 4 bytes of data. Its use is discouraged, as it is deprecated and support may be dropped in the future. |

- Fonte: Beginning databases with PostgreSQL: Matthew and Stones, 2nd ed. Apress

# Exemplos de Tipos de Dados

- Dados temporais

**Table B-4.** *PostgreSQL Types for Date and Time*

| SQL Name | PostgreSQL Alternative Name | Notes |
|---|---|---|
| `timestamp` | `datetime` | Stores dates and times from 4713 BC to 1465001 AD, with a resolution of 1 microsecond. You may also see `timestamptz` used sometimes in PostgreSQL, which is a shorthand for timestamp with time zone. |
| `interval` | `interval` | Stores an interval of approximately +/– 178,000,000 years, with a resolution of 1 microsecond. |
| `date` | `date` | Stores dates from 4713 BC to 32767 AD, with a resolution of 1 day. |
| `time` | `time` | Stores a time of day, from 0 to 23:59:59.99, with a resolution of 1 microsecond. |

- Fonte: Beginning databases with PostgreSQL: Matthew and Stones, 2nd ed. Apress

# Exemplos de Tipos de Dados

- Caracteres

**Table B-5.** *PostgreSQL Character Types*

| SQL Name | PostgreSQL Alternative Name | Notes |
|---|---|---|
| `char, character` | `bpchar` | Stores a single character. |
| `char(n)` | `bpchar(n)` | Stores exactly *n* characters, which will be padded with blanks if fewer characters are actually stored. |
| `character varying(n)` | `varchar(n)` | Stores a variable number of characters, up to a maximum of *n* characters, which are not padded with blanks. This is the standard choice for character strings. |
| | `text` | A PostgreSQL-specific variant of `varchar`, which does not require you to specify an upper limit on the number of characters. |

- Fonte: Beginning databases with PostgreSQL: Matthew and Stones, 2nd ed. Apress

# Exemplos de Tipos de Dados

- Além dos tipos exemplificados, existem outros
- Para maiores detalhes consulte o manual do PostgreSQL
  - https://www.postgresql.org/docs/current/datatype.html

# Restrições de Integridade
## Atributo

- Valor nulo
  - representado por NULL
  - membro de todos os domínios
- Restrição NOT NULL
  - especificada quando NULL não é permitido
  - proíbe que o atributo receba valor nulo
- Comparações
  - usar IS NULL e IS NOT NULL

# Restrições de Integridade
## Chave

- Cláusula PRIMARY KEY
  - identifica os atributos da relação que formam a sua chave primária
    - os atributos são implicitamente NOT NULL
  - sintaxe

    CONSTRAINT nome_restrição PRIMARY KEY (atributo$_1$, atributo$_2$, ..., atributo$_X$)

- Cláusula UNIQUE
  - não permite valores duplicados para um determinado atributo

# Restrições de Integridade
## Chave

- Integridade referencial
  - dependência existente entre a chave estrangeira de uma relação e a chave primária da relação relacionada
  - problemas
    - atualização ou exclusão de elementos da chave primária sem fazer um ajuste coordenado nas chaves estrangeiras
    - inclusão ou alteração de valores não nulos na chave estrangeira que não existam na chave primária

# Restrições de Integridade
## Chave

- Cláusula FOREIGN KEY
  - características
    - elimina a possibilidade de violação da integridade referencial
    - reflete nas chaves estrangeiras todas as alterações na chave primária
  - sintaxe

    CONSTRAINT nome_restrição FOREIGN KEY (atributos)
    REFERENCES nome_relação (atributos)
    [ON UPDATE ação]
    [ON DELETE ação]

# CREATE TABLE

## Sintaxe completa

- https://www.postgresql.org/docs/current/sql-createtable.html

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } | UNLOGGED ] TABLE [ IF NOT EXISTS ] table_name ( [
  { column_name data_type [ COLLATE collation ] [ column_constraint [ ... ] ]
    | table_constraint
   | LIKE source_table [ like_option ... ] }
    [, ... ]
] )
[ INHERITS ( parent_table [, ... ] ) ]
[ WITH ( storage_parameter [= value] [, ... ] ) | WITH OIDS | WITHOUT OIDS ]
[ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ]
[ TABLESPACE tablespace_name ]

...
```

# Como ler a sintaxe

| | |
|---|---|
| UPPERCASE (maiúsculo) | Palavra-chave SQL. |
| lowercase (minúsculo) | Identificadores ou constantes SQL informadas pelo usuário |
| *itálico* | Nome de um bloco de sintaxe. Essa convenção é usada para indicar blocos longos de sintaxe que podem ser usados em mais de um local. |
| \| (barra vertical) | Separa elementos opcionais da sintaxe dentro de colchetes ou chaves. Somente um dos itens pode ser escolhido. |
| [ ] (colchetes) | Item de sintaxe opcional. Os colchetes não fazem parte do comando. |
| { } (chaves) | Item da sintaxe obrigatório. As chaves não fazem parte do comando. |
| [,...] | O item precedente pode ser repetido N vezes. A separação entre os itens é feita por uma vírgula |
| [ ...] | O item precedente pode ser repetido N vezes. A separação entre os itens é feita por um espaço em branco. |

# DROP TABLE

```
DROP TABLE nome_tabela ;
```

- Remove uma tabela existente do BD
  - dados – metadados
  - índices
  - gatilhos que referenciam a tabela
- Usuários autorizados
  - proprietário do banco de dados
  - DBA ou usuário com privilégio de *root*

# DROP TABLE

```
DROP TABLE nome_tabela ;
```

- Obs.: 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.

# ALTER TABLE

ALTER TABLE nome_tabela;

- Altera o esquema de uma tabela do BD
  - adiciona
  - remove
  - Altera

  } colunas ou restrições de integridade

  - https://www.postgresql.org/docs/current/sql-altertable.html

# ALTER TABLE
## mais detalhes

**ALTER TABLE <nome da tabela>**
**ADD <definição de Coluna>**
**ADD <Restrição de integridade> -- Chaves primárias, Estrangeiras**
**ALTER <definição de Coluna>**
**ALTER <definição de Coluna> DEFAULT <default-value>**
**ALTER <definição de Coluna> [ NOT ] NULL**
**DROP <definição de Coluna>**
**DROP CONSTRAINT <nome da restrição> -- Remove uma restrição**
**RENAME TO <novo nome> -- Renomeia a tabela**
**RENAME <Atributo> TO <novo atributo>**
**Onde <definição de coluna> pode ser:**
**<Nome Atributo> <Tipo de Dado> [NULL ] |**
**[ DEFAULT default-value ] -- nao vale [NOT NULL]**

# Exemplos: ALTER TABLE

```
ALTER TABLE vinicola ADD nome_fundador VARCHAR(30)

ALTER TABLE vinicola DROP descrição_vinicola

ALTER TABLE vinicola ALTER nome_fundador TYPE  VARCHAR (45)

ALTER TABLE vinicola DROP CONSTRAINT vinipk

ALTER TABLE vinicola ADD CONSTRAINT vinipk PRIMARY KEY
    (vinicola_id)
```

# SEQÜÊNCIAS, VALORES DEFAULT E RESTRIÇÕES

# Seqüência

- Facilita o processo de criação de identificadores únicos de um registro em um banco de dados
- É um contador automático que é acionado toda vez que é acessado
- O número gerado por ela pode ser usado para atualizar o campo chave em uma tabela, garantindo que não existam duas linhas com o mesmo código

# Seqüência

- Uma seqüência padrão tem as seguintes características
  - Começa sempre a partir do número 1
  - Tem ordem ascendente
  - É aumentada em 1

# Seqüência – no PostgreSQL

- Similar a propriedade de auto incremento disponível em outros SGBDs
- Resumi-se na criação de um campo do tipo `SERIAL`

```
CREATE TABLE nome_tabela (
        id SERIAL
)
```

# Seqüência – no PostgreSQL

```
CREATE TABLE nome_tabela (
        id SERIAL
)
```

- corresponde a execução dos seguintes comandos:

```
CREATE SEQUENCE nome_tabela_id_seq
    INCREMENT 1
    MINVALUE 1
    START 1;

ALTER TABLE nome_tabela ALTER COLUMN id SET DEFAULT
   NEXTVAL('nome_tabela_id_seq')
```

# Seqüência – no PostgreSQL

- Observações:
  - `nextval()` → é uma função utilizada para obter o próximo valor de uma sequência
  - se você desejar que um campo do tipo `SERIAL` seja uma chave primária é necessário especificar a restrição `PRIMARY KEY` no momento da definição do campo na criação da tabela

# Seqüência

- Sintaxe básica

```
CREATE SEQUENCE sequence_name
[INCREMENTED BY integer_value]
[START WITH integer_value]
```

# Seqüência

- Para acessar o número seqüencial gerado é necessário trabalhar com duas funções
  - `CURRVAL()`: Retorna o valor atual da seqüência
  - `NEXTVAL()`: Aumenta o valor da seqüência e retorna o próximo valor
  - Exemplos:

```
1) SELECT CURRVAL('sequencia')
2) INSERT INTO nome_tabela VALUES(
       NEXTVAL('sequencia'),
       outros_valores, ...  )
```

# Seqüência - Exemplo

```
/* cria uma sequencia iniciando em 1000 e
   incrementada em 1 */
-- DROP SEQUENCE Seq
CREATE SEQUENCE Seq
START WITH 1001
INCREMENT BY 1
-- testando a sequencia (rodar várias vezes)
SELECT NEXTVAL('Seq');
-- usando com INSERT INTO
CREATE TABLE teste(n int);

INSERT INTO teste VALUES (NEXTVAL('Seq'));
```

# Seqüência

- Observações importantes
  - Uma seqüência só está disponível para o esquema que a criou
  - A primeira vez que uma seqüência é acionada ela retorna o seu valor inicial

# Seqüência

- Para excluir uma seqüência
  - `DROP SEQUENCE sequence_name [ CASCADE | RESTRICT ]`
- Para alterar uma seqüência
  - `ALTER SEQUENCE sequence_name parameter_name`

# Valores Default

- Associa um valor padrão (*default) a uma coluna*
- Quando uma nova tupla é inserida e nenhum valor é passado para algumas das colunas, estas serão preenchidas com o seu respectivo valor *default*
- Se nenhum valor *default é fornecido, o sistema usa o NULL*

# Valores Default

- O valor default pode ser indicado na declaração da tabela:

```
CREATE TABLE LIVRO(
     COD_LIVRO SMALLINT NOT NULL,
     TITULO CHAR(35) NOT NULL,
     VALOR NUMERIC(7,2),
     VOLUME SMALLINT DEFAULT 1,
     COD_EDITORA SMALLINT NOT NULL,
     CONSTRAINT CHAVELIVRO PRIMARY KEY (COD_LIVRO),
     CONSTRAINT ESTRANGEIRAEDITORA FOREIGN KEY
    (COD_EDITORA)
        REFERENCES EDITORA
);
```

# Valores Default

- O valor default pode ser indicado depois na alteração da declaração de uma tabela:

```
-- Alterando campo da tabela livro do esquema Livraria
-- SET SEARCH_PATH TO LIVRARIA
ALTER TABLE LIVRO ALTER VOLUME SET DEFAULT 1;

-- Insira novas tuplas e observe os dados da tabela
INSERT INTO LIVRO (COD_LIVRO, TITULO, VALOR, COD_EDITORA)
   VALUES (32, 'LIVRO FICTÍCIO', 100, 2);

SELECT * FROM LIVRO
```

# Valores Default

- O valor default pode ser indicado depois na alteração da declaração de uma tabela:

```
-- Alterando campo da tabela livro do esquema Livraria
-- SET SEARCH_PATH TO LIVRARIA
ALTER TABLE LIVRO A

-- Insira novas tup
INSERT INTO LIVRO (
   VALUES (32, 'LIVR

SELECT * FROM LIVRO
```

Data Output | Explain | Messages | History

| | cod_livro smallint | titulo character(35) | valor numeric(7,2) | volume smallint | cod_editora smallint |
|---|---|---|---|---|---|
| 1 | 31 | MAKTUB | 24.00 | | 1 |
| 2 | 55 | BRIDA | 29.50 | | 1 |
| 3 | 63 | HISTÓRIAS PA | 35.00 | | 2 |
| 4 | 14 | DOM CASMURRO | 10.90 | | 3 |
| 5 | 13 | SQL | 10.90 | | 4 |
| 6 | 15 | O ALIENISTA | 10.90 | | 3 |
| 7 | 16 | CONTOS | 24.00 | | 3 |
| 8 | 17 | CONTOS FLUMI | 24.00 | | 3 |
| 9 | 32 | LIVRO FICTÍC | 100.00 | 1 | 2 |

# Restrições Check

- É o tipo mais genérico de restrição. Ele permite especificar que o valor em uma determinada coluna deve satisfazer uma expressão booleana:

```
-- O nome na restrição é opcional:
CREATE TABLE LIVRO(
 COD_LIVRO SMALLINT NOT NULL,
 TITULO CHAR(35) NOT NULL,
 VALOR NUMERIC(7,2) CONSTRAINT valor_positivo CHECK (VALOR > 0),
 VOLUME SMALLINT DEFAULT 1,
 COD_EDITORA SMALLINT NOT NULL,
 CONSTRAINT CHAVELIVRO PRIMARY KEY (COD_LIVRO),
 CONSTRAINT ESTRANGEIRAEDITORA FOREIGN KEY (COD_EDITORA)
                                    REFERENCES EDITORA
);
```

# Restrições Check

- Mais exemplos:

```
-- Pode-se trabalhar com mais de uma coluna:
CREATE TABLE LIVRO(
 COD_LIVRO SMALLINT NOT NULL,
 TITULO CHAR(35) NOT NULL,
 VALOR NUMERIC(7,2) CONSTRAINT valor_positivo CHECK (VALOR > 0),
 VALOR_DESCONTO NUMERIC(7,2),
 VOLUME SMALLINT DEFAULT 1,
 COD_EDITORA SMALLINT NOT NULL,
 CONSTRAINT CHAVELIVRO PRIMARY KEY (COD_LIVRO),
 CONSTRAINT ESTRANGEIRAEDITORA FOREIGN KEY (COD_EDITORA)
                                    REFERENCES EDITORA,
CONSTRAINT valor_desconto CHECK (VALOR_DESCONTO > 0),
CONSTRAINT verifica_valor CHECK (VALOR > VALOR_DESCONTO)
);
```

# Restrições Check

- Mais exemplos:

```
/* Para alterar a definição da tabela livro das aulas
   passadas */
ALTER TABLE LIVRO ADD VALOR_DESCONTO NUMERIC(7,2);

ALTER TABLE LIVRO ADD CONSTRAINT valor_positivo CHECK (VALOR >
   0););

ALTER TABLE LIVRO ADD CONSTRAINT valor_desconto CHECK
   (VALOR_DESCONTO > 0);

ALTER TABLE LIVRO ADD CONSTRAINT verifica_valor CHECK (VALOR >
   VALOR_DESCONTO);

-- Execute e observe o resultado
INSERT INTO LIVRO VALUES (33, 'LIVRO FICTÍCIO 2', 120, 2, 2, 150);
```

# Restrições Not-NULL

- Especifica que uma coluna não pode receber o valor NULL
  - Sempre especificada como uma restrição de coluna
  - Não é possível dar um nome explícito para essa restrição

```
CREATE TABLE LIVRO(
 COD_LIVRO SMALLINT NOT NULL,
 TITULO CHAR(35) NOT NULL,
 VALOR NUMERIC(7,2) NOT NULL CHECK (VALOR > 0),
 VOLUME SMALLINT DEFAULT 1,
 COD_EDITORA SMALLINT NOT NULL,
 CONSTRAINT CHAVELIVRO PRIMARY KEY (COD_LIVRO),
 CONSTRAINT ESTRANGEIRAEDITORA FOREIGN KEY (COD_EDITORA)
                                REFERENCES EDITORA);
```

# Restrições UNIQUE

- Especifica que os valores de uma ou mais colunas devem ser únicos
- Exemplo de restrição escrita como restrição de coluna:

```
CREATE TABLE LIVRO(
 COD_LIVRO SMALLINT NOT NULL,
 TITULO CHAR(35) NOT NULL UNIQUE,
 VALOR NUMERIC(7,2) NOT NULL CHECK (VALOR > 0),
 VOLUME SMALLINT DEFAULT 1,
 COD_EDITORA SMALLINT NOT NULL,
 CONSTRAINT CHAVELIVRO PRIMARY KEY (COD_LIVRO),
 CONSTRAINT ESTRANGEIRAEDITORA FOREIGN KEY (COD_EDITORA)
                                  REFERENCES EDITORA);
```

# Restrições UNIQUE

- Especifica que os valores de uma ou mais colunas devem ser únicos
- Exemplo de restrição escrita como restrição de tabela:

```
CREATE TABLE LIVRO(
 COD_LIVRO SMALLINT NOT NULL,
 TITULO CHAR(35) NOT NULL,
 VALOR NUMERIC(7,2) NOT NULL CHECK (VALOR > 0),
 VOLUME SMALLINT DEFAULT 1,
 COD_EDITORA SMALLINT NOT NULL,
 CONSTRAINT CHAVELIVRO PRIMARY KEY (COD_LIVRO),
 CONSTRAINT ESTRANGEIRAEDITORA FOREIGN KEY (COD_EDITORA)
                                    REFERENCES EDITORA,
 CONSTRAINT TITULOUNICO UNIQUE (TITULO));
```

# Restrições de Integridade
## Chave

- Cláusula PRIMARY KEY
  - identifica os atributos da relação que formam a sua chave primária
    - os atributos são implicitamente NOT NULL
  - sintaxe

    CONSTRAINT nome_restrição PRIMARY KEY (atributo$_1$, atributo$_2$, ..., atributo$_X$)

# Restrições de Integridade
## Chave

- Cláusula FOREIGN KEY
  - características
    - elimina a possibilidade de violação da integridade referencial
    - reflete nas chaves estrangeiras todas as alterações na chave primária
  - sintaxe

    CONSTRAINT nome_restrição FOREIGN KEY (atributos)
    REFERENCES nome_relação (atributos)
    [ON UPDATE ação]
    [ON DELETE ação]

# Restrições de Integridade
## Chave

- Cláusula FOREIGN KEY
  - sintaxe

    CONSTRAINT nome_restrição FOREIGN KEY (atributos)
    REFERENCES nome_relação (atributos)
    [ON UPDATE ação]
    [ON DELETE ação]

  - ações possíveis
    - RESTRICT/NO ACTION: evita a remoção/atualização da coluna que referencia
    - CASCADE: remove/atualiza também a coluna que referencia
    - SET NULL
    - SET DEFAULT

# Restrições de Integridade
## Chave


```
/* Criando as tabelas*/
CREATE TABLE orientador (
   id INT PRIMARY KEY,
   nome VARCHAR(255)
);
CREATE TABLE aluno (
   matricula INT PRIMARY KEY,
   nome VARCHAR(255),
   orientador_id INT CONSTRAINT aluno_orientador_id_fkey
        FOREIGN KEY REFERENCES orientador(id)
        ON UPDATE NO ACTION ON DELETE NO ACTION
);
```

# Restrições de Integridade
## Chave


```
-- Povoando as tabelas
INSERT INTO orientador VALUES (1,'Prof. Luiz'),
                              (2,'Profa. Maria');

INSERT INTO aluno VALUES (1,'Alessandra',NULL),
                         (2,'Walter',2),
                         (3,'Jéssica',NULL);
```


Data Output | Explain | Messages | History

| | id integer | nome character varying(255) |
|---|---|---|
| 1 | 1 | Prof. Luiz |
| 2 | 2 | Profa. Maria |

Data Output | Explain | Messages | History

| | matricula integer | nome character varying(255) | orientador_id integer |
|---|---|---|---|
| 1 | 1 | Alessandra | |
| 2 | 2 | Walter | 2 |
| 3 | 3 | Jéssica | |

```
DELETE
FROM aluno
WHERE nome = 'Alessandra';
```

Data Output | Explain | Messages | History

| | id integer | nome character varying(255) |
|---|---|---|
| 1 | 1 | Prof. Luiz |
| 2 | 2 | Profa. Maria |

Output pane

Data Output | Explain | Messages | History

| | matricula integer | nome character varying(255) | orientador_id integer |
|---|---|---|---|
| 1 | 2 | Walter | 2 |
| 2 | 3 | Jéssica | |


Data Output | Explain | Messages | History

| | id integer | nome character varying(255) |
|---|---|---|
| 1 | 1 | Prof. Luiz |
| 2 | 2 | Profa. Maria |

Output pane

Data Output | Explain | Messages | History

| | matricula integer | nome character varying(255) | orientador_id integer |
|---|---|---|---|
| 1 | 2 | Walter | 2 |
| 2 | 3 | Jéssica | |

```
DELETE
FROM aluno
WHERE nome = 'Jéssica';
```

Data Output | Explain | Messages | History

| | id integer | nome character varying(255) |
|---|---|---|
| 1 | 1 | Prof. Luiz |
| 2 | 2 | Profa. Maria |

Data Output | Explain | Messages | History

| | matricula integer | nome character varying(255) | orientador_id integer |
|---|---|---|---|
| 1 | 2 | Walter | 2 |


Data Output | Explain | Messages | History

| | id integer | nome character varying(255) |
|---|---|---|
| 1 | 1 | Prof. Luiz |
| 2 | 2 | Profa. Maria |

Data Output | Explain | Messages | History

| | matricula integer | nome character varying(255) | orientador_id integer |
|---|---|---|---|
| 1 | 2 | Walter | 2 |

```
DELETE
FROM orientador
WHERE nome = 'Prof. Luiz';
```

Output pane

Data Output | Explain | Messages | History

| | id integer | nome character varying(255) |
|---|---|---|
| 1 | 2 | Profa. Maria |

Data Output | Explain | Messages | History

| | matricula integer | nome character varying(255) | orientador_id integer |
|---|---|---|---|
| 1 | 2 | Walter | 2 |


Output pane

Data Output | Explain | Messages | History

| | id integer | nome character varying(255) |
|---|---|---|
| 1 | 2 | Profa. Maria |

Data Output | Explain | Messages | History

| | matricula integer | nome character varying(255) | orientador_id integer |
|---|---|---|---|
| 1 | 2 | Walter | 2 |

```
DELETE
FROM orientador
WHERE nome = 'Profa. Maria';
```

Output pane

Data Output | Explain | Messages | History

| | id integer | nome character varying(255) |
|---|---|---|
| 1 | 2 | Profa. Maria |

Data Output | Explain | Messages | History

| | matricula integer | nome character varying(255) | orientador_id integer |
|---|---|---|---|
| 1 | 2 | Walter | 2 |


Output pane

Data Output | Explain | Messages | History

| | id integer | nome character varying(255) |
|---|---|---|
| 1 | 2 | Profa. Maria |

Data Output | Explain | Messages | History

| | matricula integer | nome character varying(255) | orientador_id integer |
|---|---|---|---|
| 1 | 2 | Walter | 2 |

```
DELETE
FROM orientador
WHERE nome = 'Profa. Maria';
```

ERRO: atualização ou exclusão em tabela "orientador" viola restrição de chave estrangeira "aluno_orientador_id_fkey" em "aluno"
DETAIL: Chave (id)=(2) ainda é referenciada pela tabela "aluno".

| | id integer | nome character varying(255) |
|---|---|---|
| 1 | 2 | Profa. Maria |

| | integer | character varying(255) | integer |
|---|---|---|---|
| 1 | 2 | Walter | 2 |

# Restrições de Integridade
## Chave


```
/* Criando as tabelas*/
CREATE TABLE orientador (
   id INT PRIMARY KEY,
   nome VARCHAR(255)
);
CREATE TABLE aluno (
   matricula INT PRIMARY KEY,
   nome VARCHAR(255),
   orientador_id INT CONSTRAINT aluno_orientador_id_fkey
        FOREIGN KEY REFERENCES orientador(id)
        ON UPDATE NO ACTION ON DELETE NO ACTION
);
```

# Restrições de Integridade
## Chave


```
/* Criando as tabelas*/
CREATE TABLE orientador (
   id INT PRIMARY KEY,
   nome VARCHAR(255)
);
CREATE TABLE aluno (
   matricula INT PRIMARY KEY,
   nome VARCHAR(255),
   orientador_id INT CONSTRAINT aluno_orientador_id_fkey
        FOREIGN KEY REFERENCES orientador(id)
        ON UPDATE NO ACTION ON DELETE CASCADE
);
```

# Restrições de Integridade

## Chave


```
/* Alterando definição de restrição*/
CREATE TABLE aluno (
  matricula INT PRIMARY KEY,
  nome VARCHAR(255),
  orientador_id INT CONSTRAINT aluno_orientador_id_fkey
      FOREIGN KEY REFERENCES orientador(id)
      ON UPDATE NO ACTION ON DELETE CASCADE
);
ALTER TABLE aluno DROP CONSTRAINT aluno_orientador_id_fkey

ALTER TABLE ALUNO ADD CONSTRAINT aluno_orientador_id_fkey
   FOREIGN KEY (orientador_id) REFERENCES orientador (id)
   ON UPDATE NO ACTION ON DELETE CASCADE
```

# Sintaxe SQL PostgreSQL

- Consultar material disponível em
  http://www.postgresql.org/docs/manuals/

# Manual da ferramenta pgAdim

- Consultar material disponível em
  http://www.pgadmin.org/docs/

# Leitura complementar para casa

- Capítulo 9 do livro do Elmasri – Navathe
  - Mais SQL: asserções, visões e técnicas de programação
- Capítulo 25 do livro: Deitel, H. M. Java: como programar.
- Exercícios PostgreSQL:
  - https://pgexercises.com/
- Manual do SGBD PostgreSQL
  - Explorar as outras particularidades dos comandos apresentados
  - https://www.postgresql.org/docs/

# Bibliografia


- Elmasri, Ramez; Navathe, Shamkant B. **Sistemas de banco de dados.** 4 ed. São Paulo: Addison Wesley, 2005, 724 p. Bibliografia: p. [690]-714.

- Material Didático produzido pelos professores Cristina Dutra de Aguiar Ciferri, Caetano Traina Júnior e Bruno Travençolo