

**A TRANSMISSÃO DA  
AULA COMEÇA EM**

**INSTANTES**



# Sistemas de Informação

# **Bando de Dados 1**

Prof. Dr. Ronaldo Castro de Oliveira

[ronaldo.co@ufu.br](mailto:ronaldo.co@ufu.br)

FACOM - 2022

# SQL

## DDL – Comandos para definição de dados

Prof. Ronaldo Castro de Oliveira

# Composição do SQL

---

- ▶ Linguagem de Definição dos Dados (DDL)
  - ▶ CREATE - utilizada para criação de estruturas no DDL, permite criar bancos, schemas, tabelas, bancos, restrições, ...
  - ▶ ALTER – utilizado para alteração de estruturas criadas pelo create.
  - ▶ DROP - elimina praticamente tudo aquilo criado pelo create.



# CREATE DATABASE

```
CREATE {DATABASE | SCHEMA} nome  
    [USER `username` [PASSWORD `password` ]  
    ... ;
```

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

# DROP DATABASE

```
DROP {DATABASE | SCHEMA} nome  
[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

- 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

# DDL – Comandos para definição de dados

---

## ▶ Comandos DDL

### ▶ CREATE – Cria uma definição

#### ▶ CREATE TABLE tab ...]

- Cria uma nova tabela (relação) no BD
- Nova tabela não possui dados

### ▶ ALTER – Altera uma definição

#### ▶ ALTER TABLE tab ADD ...

- Altera a estrutura de uma tabela já existente no BD

### ▶ DROP – Exclui uma definição

#### ▶ DROP TABLE tab

- Remove uma tabela e suas instâncias do BD

(**CUIDADO – APAGA A TABELA E TODOS OS DADOS**)



# DDL – Comandos para definição de dados

---

- ▶ **CREATE TABLE** – cria uma tabela, seus campos e as restrições de campo

```
CREATE TABLE <nome da tabela> (  
    <definição de coluna I>  
    ....  
    ....  
    <definição de coluna N>  
    <restrições de integridade>  
);
```

Onde <definição de coluna> pode ser  
<nome atributo> <tipo de dado> <restrições de integridade>





# 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.

# 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');.

# 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.

# 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 <code>timestamptz</code> used sometimes in PostgreSQL, which is a shorthand for <code>timestamp</code> with time zone.
interval	interval	Stores an interval of approximately $\pm 178,000,000$ 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.

# Tipos de dados

---

## ► Caracteres

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

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

# Tipos de dados

---

- ▶ Existem outros tipos de dados além dos apresentados anteriormente. Consulte o manual do PostgreSQL:
- ▶ <http://www.postgresql.org/docs/8.4/static/datatype.html>
- ▶ Livro: Beginning databases with PostgreSQL: Matthew and Stones, 2<sup>nd</sup> ed. Apress



# Identificadores

---

- ▶ Iniciam com letras (a-z) ou underscore (\_)
  - ▶ Caracteres subsequentes: letras, dígitos (0-9), \_
- ▶ Identificadores e palavras-chave não são *case-sensitive*
  - ▶ UPDATE MY\_TABLE SET A = 5;
  - ▶ uPDaTE my\_TabLE SeT a = 5;
- ▶ Convenção adotada
  - ▶ Palavras-chave em maiúscula
  - ▶ Identificadores em minúsculo
    - ▶ UPDATE my\_table SET a = 5;
- ▶ Identificadores com aspas - Ao colocar aspas em um identificador ele torna-se **CASE-SENSITIVE**
  - ▶ Aceitam quaisquer caracteres
    - ▶ UPDATE "my\_table" SET "a" = 5;



# Create Table

---

- ▶ Sintaxe completa: consultar manual PostgreSQL
- ▶ <https://www.postgresql.org/docs/9.5/static/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 ]
```



(continua no próximo slide)



# Create Table - *column\_constraint*

---

## ► Especificando a restrição **em frente à coluna**

where *column\_constraint* is:

```
[ CONSTRAINT constraint_name ]  
{ NOT NULL |  
  NULL |  
  CHECK ( expression ) [ NO INHERIT ] |  
  DEFAULT default_expr |  
  UNIQUE index_parameters |  
  PRIMARY KEY index_parameters |  
  REFERENCES reftable [ ( refcolumn ) ]  
    [MATCH FULL|MATCH PARTIAL|MATCH SIMPLE ]  
    [ ON DELETE action ] [ ON UPDATE action ]  
}  
[DEFERRABLE | NOT DEFERRABLE][INITIALLY DEFERRED | INITIALLY IMMEDIATE]
```

---



(continua no próximo slide)

# DDL – Comandos para definição de dados

---

## ► Exemplo: CREATE TABLE

*/\* Cria tabela departamento \*/*

```
CREATE TABLE Departamento (  
    DNOME VARCHAR(15) NOT NULL,  
    DNUMERO INT NOT NULL,  
    GERSSN CHAR(9) NOT NULL,  
    GERDATAINICIO DATE,  
    CONSTRAINT dptopk PRIMARY KEY (DNUMERO),  
    UNIQUE(DNOME),  
    CONSTRAINT dptogerfk FOREIGN KEY (GERSSN)  
        REFERENCES EMPREGADO(SSN)  
);
```



# DDL – Comandos para definição de dados

---

- ▶ Restrição de chave primária (PRIMARY KEY) na coluna
- ▶ Restrição de chave estrangeira (FOREIGN KEY) na coluna
  - Observe que a palavra chave REFERENCES é usada
- ▶ Restrição de unicidade (UNIQUE) coluna

## Exemplo:

*/\* Cria tabela departamento \*/*

```
CREATE TABLE Departamento (  
    DNOME VARCHAR(15) NOT NULL UNIQUE,  
    DNUMERO INT NOT NULL PRIMARY KEY ,  
    GERSSN CHAR(9) NOT NULL REFERENCES EMPREGADO(SSN),  
    GERDATAINICIO DATE  
);
```



# DDL – Comandos para definição de dados

---

- ▶ Adicionando um nome à restrição

## Exemplo:

*/\* Cria tabela departamento \*/*

```
CREATE TABLE Departamento (  
    DNOME VARCHAR(15) NOT NULL CONSTRAINT uqdnome UNIQUE,  
    DNUMERO INT NOT NULL CONSTRAINT dptopk PRIMARY KEY ,  
    GERSSN CHAR(9) NOT NULL  
        CONSTRAINT gerssnfk REFERENCES EMPREGADO(SSN),  
    GERDATAINICIO DATE,  
);
```



# Create Table - *table\_constraint*

---

- ▶ Especificando a restrição na **tabela**
  - ▶ Observe a mudança na sintaxe de algumas restrições (de chave primária, chave estrangeira)

and *table\_constraint* is:

```
[ CONSTRAINT constraint_name ]
{ CHECK ( expression ) [ NO INHERIT ] |
  UNIQUE ( column_name [, ... ] ) index_parameters |
  PRIMARY KEY ( column_name [, ... ] ) index_parameters |
  FOREIGN KEY ( column_name [, ... ] )
    REFERENCES reftable [ ( refcolumn [, ... ] ) ]
    [MATCH FULL| MATCH PARTIAL | MATCH SIMPLE ]
    [ ON DELETE action ] [ ON UPDATE action ]
}
[DEFERRABLE|NOT DEFERRABLE][INITIALLY DEFERRED|INITIALLY IMMEDIATE]
```



(continua no próximo slide)

# DDL – Comandos para definição de dados

---

- ▶ Restrição de chave primária (PRIMARY KEY) na tabela
- ▶ Restrição de chave estrangeira (FOREIGN KEY) na tabela
- ▶ Restrição de unicidade (UNIQUE) na tabela

## Exemplo:

*/\* Cria tabela departamento \*/*

```
CREATE TABLE Departamento (  
    DNOME VARCHAR(15) NOT NULL,  
    DNUMERO INT NOT NULL,  
    GERSSN CHAR(9) NOT NULL,  
    GERDATAINICIO DATE,  
    PRIMARY KEY (DNUMERO),  
    UNIQUE(DNOME),  
    FOREIGN KEY (GERSSN) REFERENCES EMPREGADO(SSN)  
);
```



# DDL – Comandos para definição de dados

---

- ▶ Adicionando um nome à restrição

## Exemplo:

*/\* Cria tabela departamento \*/*

```
CREATE TABLE Departamento (  
    DNOME VARCHAR(15) NOT NULL,  
    DNUMERO INT NOT NULL,  
    GERSSN CHAR(9) NOT NULL,  
    GERDATAINICIO DATE,  
    CONSTRAINT dptopk PRIMARY KEY (DNUMERO),  
    CONSTRAINT uqdnome UNIQUE(DNOME),  
    CONSTRAINT dptogerfk FOREIGN KEY (GERSSN)  
        REFERENCES EMPREGADO(SSN)  
);
```



# Como ler a sintaxe

Convenção	
UPPERCASE (maiúsculo)	Palavra-chave SQL.
lowercase (minúsculo)	Identificadores ou constantes SQL informadas pelo usuário
<i>itálico</i>	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.





# DDL – Comandos para definição de dados

---

- ▶ **ALTER TABLE** – Altera as definições de campos e de restrições.

**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** ]



---

► Sintaxe ALTER TABLE

► <http://www.postgresql.org/docs/8.4/static/sql-altertable.html>

ALTER TABLE [ ONLY ] name [ \* ]

*action* [, ...]

ALTER TABLE [ ONLY ] name [ \* ]

RENAME [ COLUMN ] column TO new\_column

ALTER TABLE name

RENAME TO new\_name

ALTER TABLE name

SET SCHEMA new\_schema



---

## ► Sintaxe ALTER TABLE

where *action* is one of:

```
ADD [ COLUMN ] column type [ column_constraint [ ... ] ]
DROP [ COLUMN ] column [ RESTRICT | CASCADE ]
ALTER [ COLUMN ] column [ SET DATA ] TYPE type [ USING expression ]
ALTER [ COLUMN ] column SET DEFAULT expression
ALTER [ COLUMN ] column DROP DEFAULT
ALTER [ COLUMN ] column { SET | DROP } NOT NULL
ALTER [ COLUMN ] column SET STATISTICS integer
ALTER [ COLUMN ] column SET STORAGE { PLAIN | EXTERNAL | EXTENDED | MAIN }
ADD table_constraint
DROP CONSTRAINT constraint_name [ RESTRICT | CASCADE ]
DISABLE TRIGGER [ trigger_name | ALL | USER ]
ENABLE TRIGGER [ trigger_name | ALL | USER ]
ENABLE REPLICA TRIGGER trigger_name
ENABLE ALWAYS TRIGGER trigger_name
```



---

► (continuação)

DISABLE RULE rewrite\_rule\_name  
ENABLE RULE rewrite\_rule\_name  
ENABLE REPLICA RULE rewrite\_rule\_name  
ENABLE ALWAYS RULE rewrite\_rule\_name  
CLUSTER ON index\_name  
SET WITHOUT CLUSTER  
SET WITH OIDS  
SET WITHOUT OIDS  
SET ( storage\_parameter = value [, ... ] )  
RESET ( storage\_parameter [, ... ] )  
INHERIT parent\_table  
NO INHERIT parent\_table  
OWNER TO new\_owner  
SET TABLESPACE new\_tablespace



## DDL – Comandos para definição de dados

---

**ALTER TABLE EMPREGADO ADD COLUMN CorCabelos  
CHAR(25) DEFAULT 'Branco';**

**ALTER TABLE EMPREGADO ADD Altura INT DEFAULT NULL;**

**ALTER TABLE EMPREGADO DROP Altura;**

**ALTER TABLE EMPREGADO ALTER TYPE CorCabelos CHAR(30);**

~~**ALTER TABLE DEPARTAMENTO ADD VICESSN CHAR(9)  
FOREIGN KEY (VICESSN) REFERENCES EMPREGADO (SSN)  
ON UPDATE CASCADE ON DELETE SET NULL**~~



## DDL – Comandos para definição de dados

---

**DROP TABLE** – Exclui uma tabela existente de um banco de dados. 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.

**DROP TABLE** <nome da tabela>

**Exemplo:**

*/\* Apaga tabela Departamento \*/*

**DROP TABLE** Departamento;



---

## ► Sintaxe DROP

`DROP TABLE [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT ]`



# DDL – Comandos para definição de dados

---

- ▶ **ALTER TABLE** – Altera as definições de campos e de restrições.

**ALTER TABLE** <nome da tabela>

**ADD** <definição de Coluna>

**ADD** <Restrição de integridade> -- Chaves primária, Secund. Estrang.

**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>

**RENAME** <novo nome>

**RENAME** <Atributo> **TO** <novo atributo>

Onde <definição de coluna> pode ser:

<Nome Atributo> <Tipo de Dado> [ **NULL** ] |

[ **DEFAULT** default-value ]

---






# Removendo/Adicionando uma restrição

---

**ALTER TABLE** Empregado **DROP CONSTRAINT**  
ChaveEmpregado

**ALTER TABLE** Empregado **ADD CONSTRAINT**  
ChaveEmpregado **PRIMARY KEY** (SSN);





## Sequences, Default & RESTRIÇÕES



# SEQUENCES

---

- ▶ SEQUENCES são usados para gerar automaticamente números sequenciais únicos.
  - ▶ Ex: 1,2,3,4,5,....
- ▶ Podemos usar essas sequências para atribuir valores automaticamente para atributos de tabelas
  - ▶ Ex: chave primária.
- ▶ SINTAXE

```
CREATE [ TEMPORARY | TEMP ] SEQUENCE name  
[ INCREMENT [ BY ] increment ]  
[ MINVALUE minvalue | NO MINVALUE ]  
[ MAXVALUE maxvalue | NO MAXVALUE ]  
[ START [ WITH ] start ] [ CACHE cache ]  
[ [ NO ] CYCLE ]
```

---



# SEQUENCES

*/\* 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');

-- testando o valor ATUAL DA sequencia

SELECT CURRVAL('Seq'); -- CURRENT VAL

-- usando com INSERT INTO

CREATE TABLE teste(n int);

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

INSERT INTO Aluno VALUES('JOSE DA SILVA',

NEXTVAL('Seq'), 21, 'Araguari');



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

```
CREATE TABLE Produtos (  
    id integer,  
    nome text,  
    preco numeric DEFAULT 9.99  
);
```

---



# Valores DEFAULT

---

- ▶ Expressões podem ser avaliadas

```
CREATE TABLE Produtos (  
    id integer DEFAULT  
        nextval('products_product_no_seq'),  
    nome text,  
    preco numerico DEFAULT 9.99  
);
```



# Valores DEFAULT

---

```
CREATE TABLE log (  
  t TIMESTAMP DEFAULT now(),  
  b text  
)
```

	t timestamp without time zone	b text
1	2010-06-09 22:36:16.032	texto
2	2010-06-09 22:37:55.755	texto2
3	2010-06-09 22:37:55.755	texto3
4	2010-06-09 22:38:35.243	texto4
5	2010-06-09 22:38:38.356	texto5
6	2010-06-09 22:38:39.82	texto6

```
INSERT INTO log (b) values ('texto');  
INSERT INTO log (b) values ('texto2');  
INSERT INTO log (b) values ('texto3');  
INSERT INTO log (b) values ('texto4');  
INSERT INTO log (b) values ('texto5');  
INSERT INTO log (b) values ('texto6');
```



# Valores DEFAULT

---

- ▶ Tipo **serial**

**CREATE TABLE** *tablename* ( *colname* **SERIAL** );

É equivalente a

**CREATE SEQUENCE** *tablename\_colname\_seq*;

**CREATE TABLE** *tablename* (  
    *colname* integer **DEFAULT**  
    nextval('tablename\_colname\_seq') **NOT NULL** );





# Valores Default

---

## ► Tipo **serial**

```
CREATE TABLE produtos (  
    id integer DEFAULT nextval('products_product_no_seq'),  
    nome text,  
    preco numerico DEFAULT 9.99  
);
```



```
CREATE TABLE produtos (  
    id SERIAL,  
    nome text,  
    preco numeric DEFAULT 9.99  
);
```



# Valores Default

---

- ▶ Como chamar o 'nextval' de um tipo serial?

```
CREATE TABLE produtos (  
    id SERIAL,  
    nome text,  
    preco numeric DEFAULT 9.99  
);
```

- ▶ COMANDO INSERT INTO sem a coluna referente ao tipo SERIAL ou atribuindo o valor DEFAULT a essa coluna.
    - ▶ **INSERT INTO PRODUTOS VALUES (DEFAULT,'milho',DEFAULT)**
    - ▶ **INSERT INTO PRODUTOS VALUES (DEFAULT,'queijo')**
    - ▶ **INSERT INTO PRODUTOS(nome) VALUES ('arroz')**
    - ▶ **INSERT INTO PRODUTOS(nome,preco) VALUES ('goiabada',15.00)**
  - ▶ Obs: SERIAL possui 4 bytes; BIGSERIAL 8 bytes.
  - ▶ Valor inicial: 1
- 



# Restrição CHECK

---

- ▶ É uma restrição que verifica se os valores em uma determinada coluna satisfazem uma expressão booleana

```
CREATE TABLE produtos (  
    product_no integer,  
    nome text,  
    valor numeric CHECK (valor > 0)  
);
```



# Restrição CHECK

---

- ▶ Pode-se nomear a restrição

```
CREATE TABLE produtos (  
    product_no integer,  
    nome text,  
    valor numeric,  
    CONSTRAINT valor_positivo CHECK (valor > 0)  
);
```



# Restrição CHECK

---

- ▶ Pode-se trabalhar com mais de uma coluna

```
CREATE TABLE produtos (  
    product_no integer,  
    nome text,  
    valor numeric CHECK (valor > 0) ,  
    valor_com_desconto numeric ,  
        CHECK (valor_com_desconto > 0) ,  
        CHECK (valor > valor_com_desconto )  
);
```



# Constraints

---

## ▶ Violações de chave estrangeira

- ▶ Os SGBDs verificam automaticamente as restrições de chave estrangeira para evitar inconsistências na base
- ▶ Exemplo: tente eliminar um funcionário

-- deletando um empregado

DELETE FROM empregado

WHERE SSN= '122';

É possível?



# Create Table

---

- ▶ Sintaxe completa: consultar manual PostgreSQL
- ▶ <http://www.postgresql.org/docs/8.4/static/sql-createtable.html>

## Synopsis

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } ] TABLE table_name (  
    [ { column_name data_type [ DEFAULT default_expr ]  
        [ column_constraint [ ... ] ] -- ver próximo slide  
        | table_constraint -- ver 2 slides a frente  
        | LIKE parent_table [ { INCLUDING | EXCLUDING }  
                                { DEFAULTS | CONSTRAINTS | INDEXES } ] ... }  
    [, ... ] ] )
```

```
[ INHERITS ( parent_table [, ... ] ) ]  
[ WITH ( storage_parameter [= value] [, ... ] ) | WITH OIDS | WITHOUT OIDS ]  
[ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ]  
[ TABLESPACE tablespace ]
```

---

▶ (continua no próximo slide)

# Create Table - *column\_constraint*

---

- ▶ Sintaxe completa: consultar manual PostgreSQL
- ▶ <http://www.postgresql.org/docs/8.4/static/sql-createtable.html>

where *column\_constraint* is:

```
[ CONSTRAINT constraint_name ]  
{ NOT NULL |  
  NULL |  
  UNIQUE index_parameters |  
  PRIMARY KEY index_parameters |  
  CHECK ( expression ) |  
  REFERENCES reftable [ ( refcolumn ) ] [ MATCH FULL | MATCH PARTIAL | MATCH  
SIMPLE ]  
  [ ON DELETE action ] [ ON UPDATE action ] }  
[ DEFERRABLE | NOT DEFERRABLE ] [ INITIALLY DEFERRED | INITIALLY IMMEDIATE  
]
```

---

▶ (continua no próximo slide)



# Create Table - *table\_constraint*

---

- ▶ Sintaxe completa: consultar manual PostgreSQL
- ▶ <http://www.postgresql.org/docs/8.4/static/sql-createtable.html>

and *table\_constraint* is:

```
[ CONSTRAINT constraint_name ]  
{ UNIQUE ( column_name [, ... ] ) index_parameters |  
  PRIMARY KEY ( column_name [, ... ] ) index_parameters |  
  CHECK ( expression ) |  
  FOREIGN KEY ( column_name [, ... ] ) REFERENCES reftable [ ( refcolumn [, ... ] ) ]  
    [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ] [ ON DELETE action ] [ ON UPDATE  
action ] }  
[ DEFERRABLE | NOT DEFERRABLE ] [ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
```

*index\_parameters* in UNIQUE and PRIMARY KEY constraints are:

```
[ WITH ( storage_parameter [= value] [, ... ] ) ]  
[ USING INDEX TABLESPACE tablespace ]
```



(continua no próximo slide)

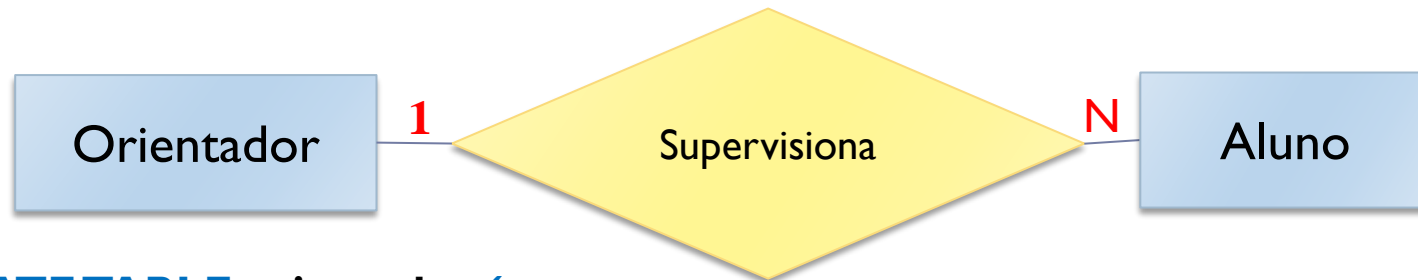
# Chaves estrangeiras

---

## ▶ Ações: ON DELETE e ON UPDATE

- ▶ **NOACTION:** A ação não é executada caso haja referência ao registro afetado
- ▶ **CASCADE:** Pode-se propagar a alteração feita em uma tabela para as outras tabelas que a referenciam.
  - ▶ ON DELETE CASCADE
  - ▶ ON UPDATE CASCADE
- ▶ **SET NULL/DEFAULT:** Pode-se atribuir NULL | DEFAULT a uma coluna que referencia um registro que será apagado ou alterado
  - ▶ ON DELETE SET NULL ; ON DELETE SET DEFAULT
  - ▶ ON UPDATE SET NULL ; ON UPDATE SET DEFAULT





```
CREATE TABLE orientador (  
  id INT PRIMARY KEY,  
  nome VARCHAR(255)  
);
```

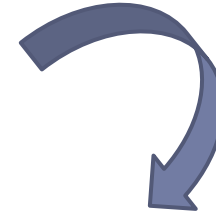
```
CREATE TABLE aluno (  
  matricula INT PRIMARY KEY,  
  nome VARCHAR(255),  
  orientador_id INT REFERENCES orientador(id)  
);
```

*/\* Povoando as tabelas\*/*

```
INSERT INTO orientador VALUES (1,'Prof. José'), (2,'Profa. Maria');  
INSERT INTO aluno VALUES (1,'Carlos',NULL), (2,'Roberto',2),  
(3,'Ailton',NULL)
```



```
CREATE TABLE aluno (  
  matricula INT PRIMARY KEY,  
  nome VARCHAR(255),  
  orientador_id INT REFERENCES orientador(id)  
);
```



```
CREATE TABLE aluno  
(  
  matricula integer NOT NULL,  
  nome character varying(255),  
  orientador_id integer,  
  CONSTRAINT aluno_pkey PRIMARY KEY (matricula),  
  CONSTRAINT aluno_orientador_id_fkey FOREIGN KEY (orientador_id)  
    REFERENCES orientador (id) MATCH SIMPLE  
    ON UPDATE NO ACTION ON DELETE NO ACTION  
)
```



	matricula integer	nome character varying(255)	orientador_id integer
<b>1</b>	1	Carlos	
<b>2</b>	2	Roberto	2
<b>3</b>	3	Ailton	

	id integer	nome character varying(255)
<b>1</b>	1	Prof. José
<b>2</b>	2	Profa. Maria

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

	matricula integer	nome character varying(255)	orientador_id integer
<b>1</b>	2	Roberto	2
<b>2</b>	3	Ailton	

	id integer	nome character varying(255)
<b>1</b>	1	Prof. José
<b>2</b>	2	Profa. Maria



	matricula integer	nome character varying(255)	orientador_id integer
<b>1</b>	1	Carlos	
<b>2</b>	2	Roberto	2
<b>3</b>	3	Ailton	

	id integer	nome character varying(255)
<b>1</b>	1	Prof. José
<b>2</b>	2	Profa. Maria

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

	matricula integer	nome character varying(255)	orientador_id integer
<b>1</b>	1	Carlos	
<b>2</b>	3	Ailton	

	id integer	nome character varying(255)
<b>1</b>	1	Prof. José
<b>2</b>	2	Profa. Maria



	matricula integer	nome character varying(255)	orientador_id integer
<b>1</b>	1	Carlos	
<b>2</b>	2	Roberto	2
<b>3</b>	3	Ailton	

	id integer	nome character varying(255)
<b>1</b>	1	Prof. José
<b>2</b>	2	Profa. Maria

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

	matricula integer	nome character varying(255)	orientador_id integer
<b>1</b>	1	Carlos	
<b>2</b>	2	Roberto	2
<b>3</b>	3	Ailton	

	id integer	nome character varying(255)
<b>1</b>	2	Profa. Maria



	matricula integer	nome character varying(255)	orientador_id integer
1	1	Carlos	
2	2	Roberto	2
3	3	Ailton	

	id integer	nome character varying(255)
1	1	Prof. José
2	2	Profa. Maria

```
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".

1	1	Carlos	
2	2	Roberto	2
3	3	Ailton	

1	1	Prof. José
2	2	Profa. Maria



```

CREATE TABLE aluno
(
  matricula integer NOT NULL,
  nome character varying(255),
  orientador_id integer,
  CONSTRAINT aluno_pkey PRIMARY KEY (matricula),
  CONSTRAINT aluno_orientador_id_fkey FOREIGN KEY (orientador_id)
    REFERENCES orientador (id) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE NO ACTION
)

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".

1	1	Carlos	
2	2	Roberto	2
3	3	Ailton	

1	1	Prof. José
2	2	Profa. Maria

```
ALTER TABLE aluno  
  DROP CONSTRAINT aluno_orientador_id_fkey;
```

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



	matricula integer	nome character varying(255)	orientador_id integer
<b>1</b>	1	Carlos	
<b>2</b>	2	Roberto	2
<b>3</b>	3	Ailton	

	id integer	nome character varying(255)
<b>1</b>	1	Prof. José
<b>2</b>	2	Profa. Maria

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

	matricula integer	nome character varying(255)	orientador_id integer
<b>1</b>	1	Carlos	
<b>2</b>	3	Ailton	

	id integer	nome character varying(255)
<b>1</b>	1	Prof. José



```
ALTER TABLE aluno  
  DROP CONSTRAINT fk_orientador;
```

```
ALTER TABLE aluno  
ADD CONSTRAINT fk_orientador FOREIGN KEY (orientador_id)  
REFERENCES orientador(id)  
  ON UPDATE NO ACTION  
  ON DELETE SET NULL;
```



	matricula integer	nome character varying(255)	orientador_id integer
<b>1</b>	1	Carlos	
<b>2</b>	2	Roberto	2
<b>3</b>	3	Ailton	

	id integer	nome character varying(255)
<b>1</b>	1	Prof. José
<b>2</b>	2	Profa. Maria

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

	matricula integer	nome character varying(255)	orientador_id integer
<b>1</b>	1	Carlos	
<b>2</b>	3	Ailton	
<b>3</b>	2	Roberto	

	id integer	nome character varying(255)
<b>1</b>	1	Prof. José



**OBRIGADO A TODOS**

**DÚVIDAS**

