Sequences, Default & RESTRIÇÕES

Bruno A. N. Travençolo – FACOM

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');
```

Relembrando a sintaxe do **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] *-- aula de hoje*
  [*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)

Relembrando a sintaxe do **CREATE TABLE**

- 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) | *-- aula de hoje*
  REFERENCES reftable [(refcolumn)] [MATCH FULL | MATCH PARTIAL | MATCH SIMPLE] **[ON DELETE action] [ON UPDATE action]** } *-- aula de hoje*

[DEFERRABLE | NOT DEFERRABLE] [INITIALLY DEFERRED | INITIALLY IMMEDIATE]

(continua no próximo slide)

Relembrando a sintaxe do **CREATE TABLE**

- 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)

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 zo	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

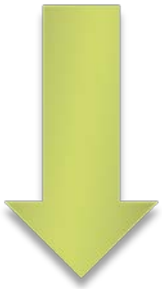
CREATE SEQUENCE *tablename_colname*_seq;

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:

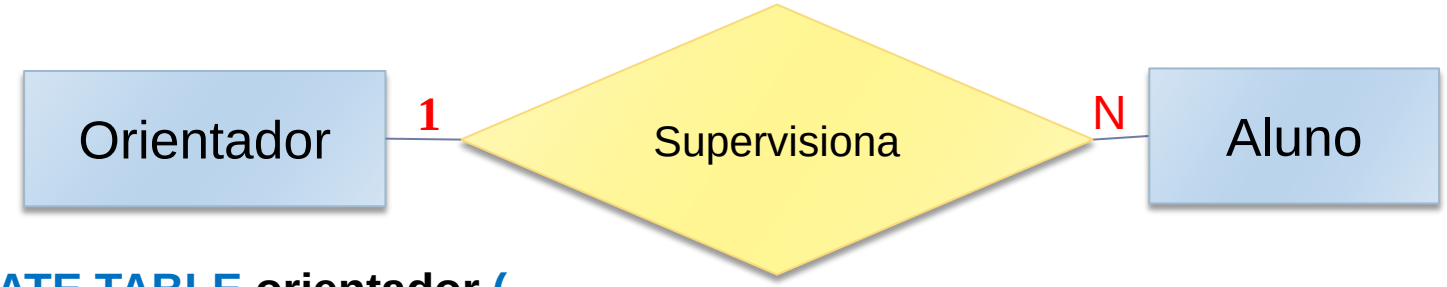
[CONSTRAINT constraint_name]
{ UNIQUE (column_name [, ...]) index_parameters |
  PRIMARY KEY (column_name [, ...]) index_parameters |
  CHECK (expression) |
  FOREIGN KEY (column_name [, ...]) REFERENCES reftable [(refcolumn [, ...])]
    [MATCH FULL | MATCH PARTIAL | MATCH SIMPLE] [ON DELETE action] [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

- Açoes: 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
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 aluno
WHERE nome = 'Carlos';
```

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

	id integer	nome character varying(255)
1	1	Prof. José
2	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 aluno
WHERE nome = 'Roberto';
```

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

	id integer	nome character varying(255)
1	1	Prof. José
2	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 = 'Prof. José';
```

	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	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
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';
```

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

	id integer	nome character varying(255)
1	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
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';
```

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

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

Chaves estrangeiras

- Açoes: 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 uma registro que será apagado ou alterado
    - ON DELETE SET NULL ; ON DELETE SET DEFAULT
    - ON UPDATE SET NULL ; ON UPDATE SET DEFAULT

Referências

- http://www.postgresql.org/docs/8.1/interactive/ddl-default.html