

SQL

DDL – Comandos para definição de dados

Prof. Bruno Travençolo

SQL

Structured Query Language

- ◆ Desenvolvida e implementada pelo laboratório de pesquisa da IBM em San Jose – início da década de 70
- ◆ Inicialmente chamada de SEQUEL (*Structured English QUery Language*)
- ◆ Criada como interface entre usuários e o primeiro SGBDR – SYSTEM R



SQL

Structured Query Language

- ▶ Uma das mais importantes linguagens relacionais
- ▶ Exemplos de SGBD que utilizam SQL

- Oracle
- Informix
- Ingress
- MS SQL Server
- Interbase/Firebird
- Sybase
- DB2
- MySQL
- PostgreSQL



SQL

Structured Query Language

- ▶ Atrativo: pequena quantidade de comandos para realizar todas as operações necessárias para definição e manipulação de relações
 - ▶ Simplicidade
 - ▶ Grande poder de consulta
- ▶ Padrão facilita migração



SQL

Structured Query Language

- ▶ O padrão SQL

- ▶ *American National Standard Institute (ANSI) e International Organization for Standardization (ISO)*

- ▶ Versão mais recente:

- ▶ SQL 2011

- ▶ Versões anteriores

- ▶ SQL 3 → SQL 99

- ▶ SQL 2 → SQL 92

- ▶ SQL 1 → SQL 86



Composição do SQL

- ▶ **Linguagem de Definição dos Dados (DDL)**
 - ▶ comandos para a definição, a modificação e a remoção de relações, além da criação e da remoção de índices
- ▶ **Linguagem Interativa de Manipulação dos Dados (DML)**
 - ▶ comandos para a consulta, a inserção, a remoção e a modificação de tuplas no banco de dados



Composição do SQL

- ▶ Linguagem de Manipulação dos Dados Embutida
 - ▶ pode ser utilizada a partir de linguagens de programação de propósito geral
- ▶ Definição de visões
 - ▶ SQL DDL inclui comandos para a criação e a remoção de visões
- ▶ Restrições de integridade
 - ▶ SQL DDL possui comandos para a especificação de restrições de integridade



Composição do SQL

- ▶ **Autorização**

- ▶ SQL DDL inclui comandos para a especificação de direitos de acesso a relações e visões

- ▶ **Gerenciamento de transações**

- ▶ introduz comandos para a especificação do início e do fim das transações

- ▶ **Recuperação de falhas**

- ▶ introduz comandos para utilização do arquivo de *log*



Composição do SQL

- ▶ Linguagem de Definição dos Dados (DDL)
 - ▶ CREATE
 - ▶ ALTER
 - ▶ DROP



DDL – Comandos para definição de dados

- ▶ Comandos DDL
 - ▶ **CREATE** – Cria uma definição
 - ▶ **CREATE TABLE** tab ...
 - ▶ **ALTER** – Altera uma definição
 - ▶ **ALTER TABLE** tab ADD ...
 - ▶ **DROP** – Exclui uma definição
 - ▶ **DROP TABLE** tab



DDL – Comandos para definição de dados

► CREATE TABLE

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),  
    UNIQUE(DNOME),  
    CONSTRAINT dptogerfk FOREIGN KEY (GERSSN)  
REFERENCES EMPREGADO(SSN)  
);
```



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
 - ▶ Aceitam quaisquer caracteres
 - ▶ UPDATE "my_table" SET "a" = 5;



Identificadores

- ▶ Ao colocar aspas em um identificador ele torna-se *case-sensitive*
- ▶ Identificadores sem aspas são sempre transformados em minúsculo (embora o padrão SQL defina que se transforme em maiúscula)
- ▶ Se você criar um esquema ou tabela usando a interface gráfica do pgAdmin e, caso o identificador deste objeto não seja composto por letras minúsculas, o objeto será identificado somente por meio de aspas.
 - ▶ Faça o teste, criando esquemas e tabelas por meio da interface gráfica e utilizando letras maiúsculas.
- ▶ Mais informações e referência:
 - ▶ <http://www.postgresql.org/docs/8.4/static/sql-syntax-lexical.html>



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 1>  
    ....  
    ....  
    <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
<code>numeric (precision, scale)</code>		Stores an exact number to the precision specified. The user guide states there is no limit to the precision that may be specified.
<code>real</code>	<code>float4</code>	A 4-byte, single-precision, floating-point number.
<code>double precision</code>	<code>float8</code>	An 8-byte, double-precision, floating-point number.
<code>money</code>		Equivalent to <code>numeric(9,2)</code> , 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, 2nd ed. Apress



DDL – Comandos para definição de dados

► CREATE TABLE

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),  
    UNIQUE(DNOME),  
    CONSTRAINT dptogerfk FOREIGN KEY (GERSSN)  
REFERENCES EMPREGADO(SSN)  
);
```



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

- ▶ 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**]



S

R(25)

(30);

ON UPDATE CASCADE ON DELETE SET 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

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);



Referências

- ▶ Slides adaptados da aula da Profa. Josiane M. Bueno (*in memoriam*)/ e Prof. Humberto Luiz Razante

