[MAC0313]
Introdução aos Sistemas de Bancos de Dados
Aula 13
Structured Query Language (SQL)
e Comandos para a Criação de Esquemas

Kelly Rosa Braghetto

DCC–IME–USP

1º de outubro de 2014

SQL – Declaração de tabela (= relação armazenada)

- É feita por meio do comando `CREATE TABLE`
- Define:
  - O nome da tabela
  - O nome e o domínio das suas colunas (atributos)
  - Quaisquer tipos de restrições sobre colunas
  - Opcionalmente, restrições de chaves primária, secundária e estrangeiras

Relações criadas por meio do comando `CREATE TABLE` são criadas e armazenadas como um arquivo pelo SGBD, ou seja, elas possuem existência física no(s) disco(s)

SQL – Declarações simples de tabelas

CREATE TABLE nome_tabela
[({nome_coluna tipo_dados [restricao_coluna])]
| restricao_tabela
| [, ...]}
]

Exemplo:

```
CREATE TABLE FUNCIONARIO (
    Nome             VARCHAR(50),
    Cpf              CHAR(11),
    Sexo             CHAR(1),
    Data_nascimento  DATE
);
```

SQL – Tipos de dados de colunas

Números inteiros

- **INTEGER** ou **INT**
  No PostgreSQL, ocupa 4 bytes e representa a seguinte faixa de valores: [-2.147.483.648, +2.147.483.647]
- **SMALLINT** – ocupa geralmente a metade da quantidade de bytes usada por um INTEGER
  No PostgreSQL, ocupa 2 bytes e representa a seguinte faixa de valores: [-32.768, +32767]

SQL – Tipos de dados de colunas

Números reais

- **FLOAT** ou **REAL**
  No PostgreSQL, ocupa 4 bytes e tem precisão de 6 dígitos decimais
- **DOUBLE PRECISION**
  No PostgreSQL, ocupa 8 bytes e tem precisão de 15 dígitos decimais

SQL – Tipos de dados de colunas

Números reais

- **DECIMAL**(*i*,*j*) ou **DEC**(*i*,*j*) ou **NUMERIC**(*i*,*j*) – onde *i* é a *precisão* e indica o total de dígitos decimais, e *j* é a *escala* e indica o número de dígitos após o ponto decimal.

  Exemplo: uma coluna do tipo `DECIMAL(6,4)` poderia armazenar um valor como `0123.45` .

  $\Rightarrow$ O valor padrão para *j* é 0, mas para *i* depende do SGBD
  No PostgreSQL, é possível representar números de até 131.072 dígitos antes da vírgula e até 16.383 dígitos depois da vírgula.

SQL – Tipos de dados de colunas

Caracteres

- **CHARACTER(*n*)** ou **CHAR(*n*)** – onde *n* é o número de caracteres, que define o **tamanho fixo** da cadeia
- **CHAR VARYING** ou **VARCHAR(*n*)** – onde *n* é o número **máximo** de caracteres da cadeia
- **CHARACTER LARGE OBJECT** ou **CLOB** – para grandes cadeias de caracteres de tamanho variável (como documentos). (No PostgreSQL, esse tipo de dado chama-se **TEXT**)

Na SQL padrão, o tamanho padrão para *n* é 1. Mas no PostgreSQL, se não especificarmos o valor de *n* para um atributo do tipo VARCHAR, então ele terá um tamanho ilimitado.

SQL – Tipos de dados de colunas

Caracteres – considerações importantes:

- Cadeias de caracteres literais devem ser delimitadas por aspas simples (apóstrofos), como em `'MAC0313'`.
- Caracteres em SQL são ***case sensitive***.
  Portanto, `'MAC0313'` $\neq$ `'mac0313'`.

Obs.: Palavras reservadas da SQL são *case insensitive*, ou seja, podemos usar `create` ou `CREATE` de forma indistinta.

SQL – Tipos de dados de colunas

Caracteres – considerações importantes:

- Se um atributo é declarado como CHAR(10), em toda tupla o valor para esse atributo será uma cadeia de 10 caracteres. Portanto, se atribuirmos o literal `'MAC0313'`, o valor que será armazenado será o `'MAC0313 '` (ou seja, serão acrescidos 3 espaços em branco no final da cadeia de caracteres).
- Geralmente, os espaços em branco no final da cadeia são desconsiderados quando duas colunas do tipo CHAR(n) são comparadas, ou quando uma coluna desse tipo é convertida para um outro tipo de cadeia de caracteres.
- Já na comparação de colunas do tipo VARCHAR(n), os espaços em branco no final da cadeia são sim considerados.

SQL – Tipos de dados de colunas

Datas e horários

- **DATE** – exemplo: `'2004-10-23'` (formato que é sempre válido: YYYY-MM-DD)
- **TIME** – exemplo: `'22:45:17'` (formato HH:MM:SS)
- **TIMESTAMP** – incluem os campos DATE e TIME e mais posições para frações decimais de segundos. Exemplo: `'2014-08-20 15:43:34.827022'`

Os tipos TIME e TIMESTAMP podem ter também um qualificador WITH TIME ZONE.
Ex. de valor para um atributo do tipo TIMESTAMP WITH TIME ZONE: `'2014-08-20 15:43:34.827022-03'`

SQL – Tipos de dados de colunas

Datas e horários (continuação)

- **INTERVAL** – especifica um valor usado para incrementar ou decrementar o valor absoluto de uma data, hora ou *timestamp*. Um intervalo é qualificado para ser YEAR-MONTH, DAY-TIME ou uma mistura dos dois. Exemplos:
  - INTERVAL '1-2' – intervalo de 1 ano e 2 meses
  - INTERVAL '3 4:05:06' – intervalo de 3 dias, 4 horas, 5 minutos e 6 segundos
  - INTERVAL '1-2 3 4:05:06' – os dois acima juntos

Podemos considerar que os tipos para data e hora em SQL são essencialmente cadeias de caracteres com um formato especial.

No PostgreSQL, a função `now()` fornece a data e hora atual do sistema.

SQL – Tipos de dados de colunas

Booleano

- **BOOLEAN** – admite os valores **TRUE**, **FALSE** ou **UNKNOWN**

Observações:

- O valor UNKNOWN pode resultar de operações de comparação envolvendo o valor NULL; veremos detalhes disso em aulas futuras.

SQL – Tipos de dados de colunas

Cadeia de bits

- **BIT(*n*)** – cadeia de bits de tamanho fixo *n*
- **BIT VARYING(*n*)** – cadeia de bits com tamanho máximo *n*
- **BINARY LARGE OBJECT** ou **BLOB** – para grandes cadeias de bits de tamanho variável (como imagens)

No PostgreSQL:

- O único tipo de dados para bits implementado é o **BYTEA**, que equivale ao BLOB do SQL padrão.

SQL – Tipos de dados no PostgreSQL

Para mais informações sobre os tipos de dados no PostgreSQL 9.3:
`http://www.postgresql.org/docs/9.3/static/datatype.html`

SQL – Declaração simples de tabela

Exemplo:

```
CREATE TABLE FUNCIONARIO (
    Nome                VARCHAR(50),
    Cpf                 CHAR(11),
    Casado              BOOLEAN,
    Sexo                CHAR(1),
    Data_nascimento     DATE,
    Salario             DECIMAL(10,2),
);
```

SQL – Criação de domínios

- É possível declarar um novo domínio e usar o seu nome como especificação para um atributo.
- Para criar um domínio, usamos o comando `CREATE DOMAIN`

Exemplo:

```
CREATE DOMAIN TIPO_CPF AS CHAR(11);

CREATE TABLE FUNCIONARIO (
    Nome              VARCHAR(50),
    Cpf               TIPO_CPF,
    Casado            BOOLEAN,
    Sexo              CHAR(1),
    Data_nascimento   DATE,
    Salario           DECIMAL(10,2),
);
```

SQL – Restrições e valores padrão para colunas

Podem ser:

- **para coluna** (quando aparecem na frente da definição do domínio da coluna)
- **para tabela** (quando aparecem depois da definição das colunas)

Exemplo:

```
CREATE TABLE DEPARTAMENTO (
   Dnumero              INT NOT NULL CHECK(Dnumero > 0 AND
                                           Dnumero <= 20),
   Dnome                VARCHAR(30) NOT NULL,
   Dt_criacao           DATE,
   Cpf_gerente          TIPO_CPF DEFAULT '12345678901',
   Dt_inicio_gerente    DATE,
   CHECK (Dt_criacao <= Dt_inicio_gerente)
 );
```

SQL – Restrições e valores padrão para colunas

Restrição contra valores nulos – cláusula **NOT NULL**

- Define que uma coluna não pode receber o valor NULL
- É especificada de forma implícita para colunas que fazem parte da chave primária da tabela

Valor padrão – cláusula **DEFAULT**

- Define um valor que será atribuído à coluna em uma nova tupla sempre que o valor para essa coluna não for fornecido
- Se uma coluna não possuir a restrição de NOT NULL e nenhum valor padrão for definido para ela, então o valor NULL será usado como padrão

SQL – Restrições para colunas

Restrição de verificação – cláusula **CHECK**

- Restringe os valores que uma coluna pode assumir

Exemplo:

```
CREATE TABLE FUNCIONARIO (
    Nome            VARCHAR(50) NOT NULL,
    Cpf             TIPO_CPF    NOT NULL,
    Salario         DECIMAL(10,2) NOT NULL
                     CHECK (Salario > 650 AND Salario < 50000),
    Idade           INT CHECK (Idade >= 18 AND Idade <= 120),
    Casado          BOOLEAN NOT NULL DEFAULT FALSE,
    Cpf_supervisor  TIPO_CPF DEFAULT '12345678901'
 );
```

SQL – Restrições de chave

Restrição de chave primária – cláusula **PRIMARY KEY**

- Especifica uma ou mais colunas que compõem a chave primária da tabela
- Se a chave primária tiver uma única coluna, a cláusula pode aparecer como uma *restrição para coluna* na definição da tabela
- Para chaves com uma ou mais colunas, usa-se uma cláusula de *restrição para tabela*

Lembrete: sobre uma chave primária, sempre é imposta uma restrição de NOT NULL.

SQL – Restrições de chave

Exemplo 1: Chave primária como restrição para coluna

```
CREATE TABLE DEPARTAMENTO (
    Dnumero     INT PRIMARY KEY,
    Dnome       VARCHAR(30) NOT NULL
 );
```

Exemplo 2: Chave primária como restrição para tabela

```
CREATE TABLE LOCALIZACAO_DEP (
    Dnumero     INT NOT NULL,
    Dlocal      VARCHAR(15) NOT NULL,
    PRIMARY KEY(Dnumero,Dlocal),
 );
```

SQL – Restrições de chave

Restrição de chave secundária – cláusula **UNIQUE**

- Especifica uma ou mais colunas que compõem uma chave secundária (= alternativa) da tabela
- Se a chave secundária tiver uma única coluna, a cláusula pode aparecer como uma *restrição para coluna* na definição da tabela
- Para chaves secundárias com uma ou mais colunas, usa-se uma cláusula de *restrição para tabela*
- Assim como ocorre com a chave primária, não pode haver tuplas com valores repetidos para a chave secundária
- Diferentemente do que ocorre com a chave primária, uma coluna da chave secundária pode receber valores NULL

SQL – Restrições de chave

Exemplo 1: Chave secundária como restrição para coluna

```
CREATE TABLE DEPARTAMENTO (
    Dnumero     INT PRIMARY KEY,
    Dnome       VARCHAR(30) NOT NULL UNIQUE
 );
```

Exemplo 2: Chave secundária como restrição para tabela

```
CREATE TABLE DEPARTAMENTO (
    Dnumero     INT,
    Dnome       VARCHAR(30) NOT NULL,
    PRIMARY KEY(DNumero),
    UNIQUE(Dnome)
 );
```

SQL – Restrições de integridade referencial

Cláusula **FOREIGN KEY** – especifica uma chave estrangeira

Exemplo: chaves estrangeiras como restrições para colunas

```
CREATE TABLE FUNCIONARIO (
   Nome          VARCHAR(50) NOT NULL,
   Cpf           CHAR(11) PRIMARY KEY,
   Salario       DECIMAL(10,2)            );

CREATE TABLE DEPARTAMENTO (
   Dnumero         INT PRIMARY KEY,
   Dnome           VARCHAR(30) NOT NULL UNIQUE,
   Cpf_gerente     CHAR(11) NOT NULL DEFAULT '12345678901'
                            REFERENCES FUNCIONARIO(Cpf) );

CREATE TABLE LOCALIZACAO_DEP (
   Dnumero     INT NOT NULL REFERENCES DEPARTAMENTO(Dnumero),
   Dlocal      VARCHAR(15) NOT NULL,
   PRIMARY KEY(Dnumero,Dlocal)  );
```

SQL – Restrições de integridade referencial

Cláusula **FOREIGN KEY** – especifica uma chave estrangeira

Exemplo: chaves estrangeiras como restrições para tabelas

```
CREATE TABLE FUNCIONARIO (
   Nome          VARCHAR(50) NOT NULL,
   Cpf           CHAR(11) PRIMARY KEY,
   Salario       DECIMAL(10,2)            );

CREATE TABLE DEPARTAMENTO (
   Dnumero         INT PRIMARY KEY,
   Dnome           VARCHAR(30) NOT NULL UNIQUE,
   Cpf_gerente     CHAR(11) NOT NULL DEFAULT '12345678901',
   FOREIGN KEY(Cpf_gerente) REFERENCES FUNCIONARIO(Cpf) );

CREATE TABLE LOCALIZACAO_DEP (
   Dnumero     INT NOT NULL,
   Dlocal      VARCHAR(15) NOT NULL,
   PRIMARY KEY(Dnumero,Dlocal),
   FOREIGN KEY(Dnumero) REFERENCES DEPARTAMENTO(Dnumero) );
```

SQL – Restrições de integridade referencial

Ações de propagação para a cláusula **FOREIGN KEY**

- Há diferentes opções de ações para o tratamento das violações de integridade referencial causadas por operações de inserção, remoção e alteração:
  - opção **RESTRICT** (padrão) – rejeita a operação de atualização que causará uma violação
  - opção **SET NULL** – atribuirá NULL à chave estrangeira que ficar sem sua "referência"
  - opção **SET DEFAULT** – atribuirá um valor padrão à chave estrangeira que ficar sem sua "referência"
  - opção **CASCADE** – propaga a alteração feita na chave referenciada para as linhas que a referenciam
- As ações acima devem ser qualificadas com as cláusulas **ON DELETE** ou **ON UPDATE**

SQL – Restrições de integridade referencial

Exemplo: Chaves estrangeiras como restrições para colunas e **ações de propagação**

```
CREATE TABLE FUNCIONARIO (
   Nome          VARCHAR(50) NOT NULL,
   Cpf           CHAR(11) PRIMARY KEY,
   Salario       DECIMAL(10,2)            );

CREATE TABLE DEPARTAMENTO (
   Dnumero         INT PRIMARY KEY,
   Dnome           VARCHAR(30) NOT NULL UNIQUE,
   Cpf_gerente     CHAR(11) NOT NULL DEFAULT '12345678901'
                       REFERENCES FUNCIONARIO(Cpf)
                       ON DELETE SET DEFAULT  ON UPDATE CASCADE );

CREATE TABLE LOCALIZACAO_DEP (
   Dnumero     INT NOT NULL REFERENCES DEPARTAMENTO(Dnumero)
                            ON DELETE CASCADE  ON UPDATE CASCADE,
   Dlocal      VARCHAR(15) NOT NULL,
   PRIMARY KEY(Dnumero,Dlocal)  );
```

SQL – Restrições de integridade referencial

Exemplo: Chaves estrangeiras como restrições para tabela e **ações de propagação**

```
CREATE TABLE FUNCIONARIO (
   Nome          VARCHAR(50) NOT NULL,
   Cpf           CHAR(11) PRIMARY KEY,
   Salario       DECIMAL(10,2)            );

CREATE TABLE DEPARTAMENTO (
   Dnumero         INT PRIMARY KEY,
   Dnome           VARCHAR(30) NOT NULL UNIQUE,
   Cpf_gerente     CHAR(11) NOT NULL DEFAULT '12345678901',
   FOREIGN KEY(Cpf_gerente) REFERENCES FUNCIONARIO(Cpf)
                            ON DELETE SET DEFAULT  ON UPDATE CASCADE );

CREATE TABLE LOCALIZACAO_DEP (
   Dnumero     INT NOT NULL,
   Dlocal      VARCHAR(15) NOT NULL,
   PRIMARY KEY(Dnumero,Dlocal),
   FOREIGN KEY(Dnumero) REFERENCES DEPARTAMENTO(Dnumero)
                        ON DELETE CASCADE  ON UPDATE CASCADE        );
```

SQL – Nomeando restrições

Com a cláusula **CONSTRAINT**, é possível atribuir nomes às restrições.

Exemplo:

```
CREATE TABLE FUNCIONARIO (
    Nome            VARCHAR(50) NOT NULL,
    Cpf             CHAR(11),
    Salario         DECIMAL(10,2),
    CONSTRAINT ChP_Func PRIMARY KEY(Cpf) );

CREATE TABLE DEPARTAMENTO (
    Dnumero           INT,
    Dnome             VARCHAR(30) NOT NULL,
    Cpf_gerente       CHAR(11) NOT NULL DEFAULT '12345678901',
    CONSTRAINT ChP_Dep PRIMARY KEY(Dnumero),
    CONSTRAINT ChS_Dep UNIQUE(Dnome),
    CONSTRAINT ChE_Ger_Dep
       FOREIGN KEY(Cpf_gerente) REFERENCES FUNCIONARIO(Cpf)
         ON DELETE SET DEFAULT  ON UPDATE CASCADE );
```

SQL – Remoção de tabelas

- A remoção de uma tabela é feita por meio do comando `DROP TABLE` .
- Depois de removida, uma tabela não faz mais parte do BD e não se pode mais acessar nenhuma de suas tuplas.

DROP TABLE nome_tabela [**CASCADE** | **RESTRICT**];

- Com a opção CASCADE – remove também os objetos que dependem da tabela removida (ex.: restrições de chave estrangeira, índices, visões).
- Com a opção RESTRICT (padrão) – impede que a tabela seja removida caso haja no BD objetos que dependam dela.

Exemplo:

```
DROP TABLE FUNCIONARIO;
```

SQL – *Esquemas*

- *Esquemas* são criados dentro de um BD
- Um *esquema* funciona como um *diretório*: ele pode ser usado para agrupar objetos do BD (tabelas, visões, tipos de dados, funções, etc.) segundo algum critério semântico de organização
  - Por exemplo, podemos pensar em agrupar as tabelas que pertencem a uma mesma aplicação ou as que se referem a um módulo particular do BD
- É possível ter dois ou mais objetos com um mesmo nome em um BD contanto que cada um esteja em um esquema diferente

SQL – Criação de *esquemas*

- É feita por meio do comando `CREATE SCHEMA`:

  CREATE SCHEMA nome_esquema;

- É possível definir/acessar objetos em um esquema qualificando o nome dos objetos. Por exemplo, os comandos

```
CREATE SCHEMA EMPRESA;
CREATE TABLE EMPRESA.FUNCIONARIO...
```

  cria um esquema chamado `EMPRESA` e depois cria a tabela `FUNCIONARIO` no esquema `EMPRESA`.

SQL – Remoção de esquemas

- **DROP SCHEMA** *nome_esquema* [**CASCADE**];
  Remove um esquema.
  Com a opção CASCADE, os objetos definidos dentro do esquema também serão removidos.
- **ALTER SCHEMA** nome **RENAME TO** novo_nome;
  ALTER SCHEMA nome **OWNER TO** novo_proprietario;
  Altera as informações de um esquema.

Mais comandos para *esquemas* específicos do PostgreSQL

- **SHOW** search_path;
  Mostra o esquema atualmente em uso.
- **SET search_path TO** *nome_esquema*;
  Define o esquema atualmente em uso.
- No PostgreSQL, o esquema padrão é o `public`

Inserção de dados – comando básico

Para testar os comandos de definição de esquemas vistos na aula, insira dados no seu BD:

INSERT INTO nome_tabela [(coluna1, coluna2, ...)]
VALUES (valor1,valor2,...);

- Insere uma linha na tabela *nome_tabela*
- Quando a ordem das colunas não é especificada no comando, os valores são atribuídos de acordo com a ordem em que as colunas foram criadas na tabela
- Se os valores passados para o comando não satisfazem as restrições definidas sobre a tabela, a linha não é inserida no BD

Inserção de dados – comando básico

Exemplo:

```
INSERT INTO FUNCIONARIO(Nome, Cpf, salario)
                    VALUES ('Fernando Pessoa', '12345678901', 4532.99);
INSERT INTO FUNCIONARIO(Nome, Cpf, salario)
                    VALUES ('Clarice Lispector', '12782392989', 1238.23);
INSERT INTO FUNCIONARIO(Nome, Cpf, salario)
                    VALUES ('Carlos Drummond', '11728237928', 23919.00);

INSERT INTO DEPARTAMENTO(Dnumero, Dnome, Cpf_gerente)
                    VALUES (1, 'Contabilidade', '11728237928');
INSERT INTO DEPARTAMENTO(Dnumero, Dnome)
                    VALUES (2, 'Recursos Humanos');

INSERT INTO LOCALIZACAO_DEP VALUES (1,'Rua do Matao');
INSERT INTO LOCALIZACAO_DEP VALUES (1,'Reitoria');
```

Alteração de esquemas – modificando colunas

ALTER TABLE nome_tabela **ADD [COLUMN]** nome_coluna tipo_dado [restrições];

- Adiciona uma nova coluna em uma tabela

Exemplo:

```
ALTER TABLE Funcionario
ADD COLUMN Sexo CHAR(1) DEFAULT 'F';
```

Alteração de esquemas – modificando colunas

ALTER TABLE nome_tabela **DROP [COLUMN]** nome_coluna **[RESTRICT | CASCADE]**;

- Remove uma coluna em uma tabela. Se a opção CASCADE for usada, todas as restrições e visões que referenciam a coluna serão removidas.

Exemplo:

```
ALTER TABLE Departamento
DROP COLUMN Cpf_gerente CASCADE;
```

Alteração de esquemas – modificando a cláusula *default*

ALTER TABLE nome_tabela **ALTER [COLUMN]** nome_coluna **DROP DEFAULT**;

- Remove a cláusula *default* de uma coluna

Exemplo:

ALTER TABLE Funcionario
ALTER COLUMN Sexo DROP DEFAULT;

Alteração de esquemas – modificando a cláusula *default*

ALTER TABLE nome_tabela **ALTER [COLUMN]** nome_coluna **SET DEFAULT** valor;

- Define uma nova cláusula *default* para uma coluna

Exemplo:

ALTER TABLE Funcionario
ALTER COLUMN Sexo SET DEFAULT 'M';

Alteração de esquemas – modificando restrições nomeadas

ALTER TABLE nome_tabela **DROP CONSTRAINT** nome_restrição;

- Remove uma restrição nomeada

Exemplo:

```
CREATE TABLE Funcionario (
   Nome          VARCHAR(50) NOT NULL,
   Cpf           CHAR(11),
   Salario       DECIMAL(10,2),
   CONSTRAINT ChP_Func PRIMARY KEY(Cpf) );

...

ALTER TABLE Funcionario DROP CONSTRAINT ChP_Func;
```

Alteração de esquemas – modificando restrições nomeadas

ALTER TABLE nome_tabela **ADD CONSTRAINT**
[nome_restricao] restrição;

- Define uma nova restrição para a tabela (que pode ser nomeada ou não)

Exemplo:

```
CREATE TABLE Departamento (
   Dnumero     INT PRIMARY KEY,
   Dnome       VARCHAR(30) NOT NULL
 );

ALTER TABLE Departamento ADD CONSTRAINT ChS_Dep UNIQUE(Dnome);
```

Comandos úteis para quem usa o cliente `psql`

- Para alterar a senha de seu usuário:
  `ALTER ROLE [usuario] WITH ENCRYPTED PASSWORD '[senha]'`
  ou `\password`
- Para listar todos os BDs do servidor ao qual se está conectado:
  `\l`
- Para listar as tabelas do BD atual:
  `\dt` **ou** `SELECT * FROM pg_catalog.pg_tables`
- Para listar o esquema (= estrutura) de uma tabela:
  `\d+ nome_tabela`
- Para ver outras opções de comandos especiais do *psql*:
  `\?`

Referências Bibliográficas

- *Database Systems – The Complete Book*, Garcia-Molina, Ullman e Widom. 2002.
  Capítulo 6
- *Sistemas de Bancos de Dados* (6ª edição), Elmasri e Navathe. Pearson, 2010.
  Capítulos 4 e 5