Treinamento SQL

Um pouco de prática – Linguagem SQL

Prof. Dr. Francisco Isidro Massetto
francisco.isidro@techschool.com.br

Agenda

- Sintaxe SQL para criação de tabelas e definição de restrições
  - Criação
  - Modificação
- Tipos de dados
  - Numéricos
  - Alfanuméricos
  - outros
- Manipulação de registros
  - Inserção
  - Remoção
  - Atualização

Antes de mais nada...

- Do ponto de vista do SGBD
  - Uma base de dados é uma coleção
    - Tabelas
    - Usuários
    - Restrições
    - Procedimentos
    - Triggers
- Neste treinamento: foco maior em MySQL para exercitar os conceitos

Criando uma base

> create database minhaBase;

Query ok, 1 row(s) affected

- >
  - prompt de comando do gerenciador de bancos de dados
- create database
  - Instrução de criação da base
- minhaBase
  - Nome da base de dados
- Query ok, 1 row(s) affected
  - Resultado da operação, indicando que a operação foi um sucesso

Exibindo bases existentes

mysql> show databases;

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| acervo             |
| login              |
| mysql              |
| pessoadb           |
| pessoas            |
| scjp               |
| sispe              |
| sistemalogin       |
| soa                |
| struts             |
| techschool_cursos  |
| test               |
+--------------------+
13 rows in set (0.14 sec)
```

Removendo a base e usando-a

mysql> drop database minhaBase;

Query ok, 0 row(s) affected

- O fato do resultado indicar 0, significa que não há mais bases com nome “minhaBase” nos metadados do mysql

mysql>use minhaBase;

Database changed

- Permite iniciar operações na base de dados selecionada
- Não é possível criar qualquer tabela ou realizar qualquer tarefa de manipulação de dados sem selecionar uma base previamente

Criando uma tabela

```
CREATE TABLE nomeDaTabela
(
    campo1 tipoDeDado1 restriçõesDoCampo1,
    campo2 tipoDeDado2 restriçõesDoCampo2,
    ...

    CONSTRAINT nome TIPO
);
```

Tipos de dados

- Alfanuméricos – cadeias de caracteres
- Tamanho máximo para cadeias: 255 caracteres
  - Char
    - Tamanho fixo
  - Varchar
    - Tamanho variável
- Mais do que 255 caracteres
  - TynyText - 255
  - Text – 65.535
  - MediumText – 16Mega
  - LongText – 4 Giga

Tipos de Dados

- Numéricos Inteiros
  - Tynnyint
    - -128 a 127 ou 0 a 255 (unsigned)
  - Smallint
    - -32.768 a 32.767 ou 0 a 65535 (unsigned)
  - Mediumint
    - -8Mega a 8Mega ou 0 a 16Mega (unsigned)
  - Int
    - -2Giga a 2Giga ou 0 a 4Giga (unsigned)
  - BigInt
    - -8 Exa a 8 Exa ou 0 a 16 Hexa (unsigned) 1000^6

Tipos de Dados

- Numéricos com precisão decimal
  - Float (p, e)
    - Permite especificar a quantidade de dígitos decimais e inteiros
    - Também permite valores unsigned
    - -3,402823466 E+38 a -1,175494351 E-38 e de
    - 1,175494351 E-38 a 3,402823466 E+38
  - Double (p,e)
    - Idem ao float, porém com maior precisão

Tipos de Dados

- Temporais
  - Date
    - AAAA-MM-DD
      - de 1000-01-01 a 9999-12-31
  - DateTime
    - AAAA-MM-DD HH:MI:SS
      - de 1000-01-01 00:00:00 a 9999-12-31:23:59:59
  - TimeStamp
    - AAAA-MM-DD HH:MI:SS
      - de 1970-01-01 00:00:00 a 2037-12-31:23:59:59
  - Year
    - AAAA (de 1901 a 2155)
  - Time
    - HH:MI:SS (-838:59:59 a 838:59:59)

Exemplo prático

```
CREATE TABLE funcionario
(
    idFunc SMALLINT UNSIGNED,
    nomeF  VARCHAR(50),
    emailF VARCHAR(50),
    sexoF  CHAR(1),
    nascF  DATE
);
```

Alguns tipos mais elaborados

- Tipo Enumerativo
  - ENUM
    - Permite armazenar rum valor dentre uma série de valores válidos para o campo
    - ENUM(‘M’, ‘F’)
    - ENUM(“seg”, “ter”, “qua”, “qui”, “sex”, “sab”, “dom”);

Algumas Restrições para Campos

- NOT NULL
  - Indica que o campo deve obrigatoriamente ter um valor
- UNSIGNED
  - Indica que o tipo numérico é sem sinal (aceita somente valores positivos)
- AUTO_INCREMENT
  - O valor numérico inteiro é acrescido sempre de 1

Algumas Restrições para Tabelas

```
CONSTRAINT nome PRMARY KEY
                (lista de campos)
```

- Define a lista de campos como chave primária identificada por um símbolo

```
CONSTRAINT nome FOREIGN KEY (campos)
          REFERENCES tabela (campos)
          ON DELETE|UPDATE
    SET NULL|CASCADE|NO ACTION
```

- Define uma chave estrangeira (identificada pelo nome) através da lista de campos que referenciam o(s) campo(s) da tabela original
  - CASCADE – alterações em cascata
  - SET NULL – permite incluir NULL no campo referenciado
  - NO ACTION – não faz nada

Criando outra tabela

```
CREATE TABLE departamento
(
   idDepto SMALLINT     UNSIGNED NOT NULL
                        AUTO_INCREMENT,
   nomeD   VARCHAR(50) NOT NULL,
   atuacao VARCHAR(30),
   CONSTRAINT pk_depto PRIMARY KEY (idDepto)
);
```

Melhorando o primeiro exemplo

```
CREATE TABLE funcionario
(
   idFunc   SMALLINT          UNSIGNED NOT NULL
                              AUTO_INCREMENT,
   nomeF    VARCHAR(50)       NOT NULL,
   emailF   VARCHAR(50)       NOT NULL,
   sexoF    ENUM('M', 'F'),
   nascF    DATE,
   idDepto SMALLINT           UNSIGNED NOT NULL,

   CONSTRAINT pk_func         PRIMARY KEY (idFunc),
   CONSTRAINT fk_func_depto   FOREIGN KEY (idDepto)
                              REFERENCES departamento
                                         (idDepto)
                                   ON DELETE CASCADE
);
```

Observações na criação de tabelas

- Deve-se entender muito bem um modelo de dados e saber como interpretá-lo
  - Uma tabela que é referenciada DEVE ser criada antes
  - Caso contrário haverá um erro indicando que a chave estrangeira referencia uma tabela que não existe
- Remoções de tabelas são atividades complexas, pois podem causar “danos” na base de dados
  - Ideal é que o modelo lógico seja definitivo quando for implementar a base de dados

Algumas considerações

- Mostrando todas as tabelas da base de dados

```
mysql> show tables;
+----------------+
| Tables_in_test |
+----------------+
| departamento   |
| funcionario    |
+----------------+
2 rows in set (0.00 sec)
```

Algumas Considerações

- Mostrando a definição de uma tabela

```
mysql> desc funcionario;
+---------+----------------------+------+-----+---------+-------+
| Field   | Type                 | Null | Key | Default | Extra |
+---------+----------------------+------+-----+---------+-------+
| idFunc  | smallint(5) unsigned | NO   | PRI | NULL    |       |
| nomeF   | varchar(50)          | NO   |     | NULL    |       |
| emailF  | varchar(50)          | NO   |     | NULL    |       |
| sexoF   | enum('M','F')        | YES  |     | NULL    |       |
| nascF   | date                 | YES  |     | NULL    |       |
| idDepto | smallint(5) unsigned | NO   | MUL | NULL    |       |
+---------+----------------------+------+-----+---------+-------+
6 rows in set (0.08 sec)
```

Inserindo Dados

- Inserindo dados na Tabela

```
INSERT INTO tabela(lista de campos)
            VALUES (lista de valores)
```

  - Lista de campos é opcional
  - Lista de valores pode seguir a lista de campos especificada
    - Matching → valores dos campos tem que combinar com os tipos declarados na descrição da tabela

O tal do NULL

- O NULL tem várias interpretações que precisam ficar muito claras para o Administrador de BD
  - Não Aplicável
  - Desconhecido
  - Conjunto Vazio
- Basicamente

Auto Incremento

- Colunas que são auto-incrementáveis precisam ter seus valores especificados como “vazios”
  - O SGBD gerencia o dado “vazio” e aplica o novo valor do campo auto-incrementável
- Instrução de inserção

```
INSERT INTO departamento(codDep, nome)
  values (null, “Engenharia);
```

Inserindo datas não convencionais

- É possível manipular datas e convertê-las para o formato de dadas do MySQL
- str_to_date('data em string', 'símbolos correspondentes')
- Símbolos de data
  - %M – nome do mês (Janeiro a Dezembro)
  - %m – mês numérico (01 a 12)
  - %d – dia numérico (01 a 31)
  - %j – dia do ano (001 a 366)
  - %W – nome do dia (domingo a sábado)
  - %Y – ano com 4 dígitos
  - %y – ano com 2 dígitos
  - %H – hora (00 a 23)
  - %h – hora (00 a 12)
  - %i – minutos (00 a 59)
  - %s – segundos (00 a 59)
  - %f – microssegundos (000000 a 999999)
  - %p – AM ou PM

Alterando a Tabela

- Alterando Tabelas
  - `ALTER TABLE nome_da_tabela`
- Operações para se alterar tabelas
  - Adicionar campos
  - Remover campos
  - Modificar campos
    - Mudar nomes (CHANGE)
    - Mudar definições (MODIFY)
  - Adicionar e Remover restrições (CONSTRAINTS)
    - Chave primária e estrangeira
  - Adicionar Índices

Alterando Tabelas

- Adicionando Campos

```
ALTER TABLE tabela ADD COLUMN
                        nome tipo
  restrições;
```

- Removendo Campos

```
ALTER TABLE tabela DROP COLUMN nome
```

Alterando Tabelas

- Modificando definições de Campos

```
ALTER TABLE tabela MODIFY COLUMN
                              nome tipo restrições;
```

- Alterando um campo (inclusive nome)

```
ALTER TABLE tabela CHANGE COLUMN
  nome_antigo novo nome tipo restrições;
```

- É possível, inclusive mudar a localização do campo (antes ou depois de algum campo).
  - Em ambos os casos, pode-se incluir uma restrição <AFTER campo>

Alterando Tabelas

- Inserindo e Removendo Restrições

```
ALTER TABLE tabela ADD CONSTRAINT
nome PRIMARY KEY (campo)

ALTER TABLE tabela ADD CONSTRAINT
nome FOREIGN KEY (campo) REFERENCES
  tabela (campos)
```

Alterando Tabelas

- Inserindo Índices

```
ALTER TABLE tabela ADD INDEX
nome_indice USING [tipo] (coluna)
```

  - Tipos de índices
    - BTree
    - Hash
- Removendo Índices

```
ALTER TABLE tabela DROP INDEX nome_idx
```

Removendo Tabelas

- Remover tabelas é uma tarefa que exige cuidados
  - Relativamente frequente quando se está criando modelos de teste
  - Deve ser evitada ao máximo quando o banco de dados está em operação
- Sintaxe

```
DROP TABLE [IF EXISTS] tabela
```

Removendo Tabelas

- Algumas considerações
  - O SGBD impede de se remover uma tabela que é referenciada por outras
  - Como proceder então?
    - Primeiro são removidas as tabelas que referenciam a tabela a ser removida
    - Quando não houver nenhuma outra dependência, remove-se a tabela referenciada

Manipulando dados armazenados

- Modificando registros

```
UPDATE tabela SET campo = valor
    WHERE critério
```

  - Altera todos os registros que atendem o critério definido (opcional)
  - Para alterar um único registro?
    - Usar a Chave Primária no critério

Manipulando dados armazenados

- Apagando registros

```
DELETE FROM tabela WHERE critério
```

  - Apaga todos os registros que atendem o critério definido (opcional)
  - Para apagar um único registro?
    - Usar a Chave Primária no critério

Conferindo se os dados foram realmente armazenados

- Recuperando registros

```
SELECT campos FROM tabela WHERE
  critério
```

  - Recupera a lista de campos (* para todos os campos) segundo um determinado critério (opcional)