Aula 7

Professor:

Claudio Esperança

SQL básico

Conteúdo:

- Comandos SHOW
- Removendo bancos de dados e tabelas
- Alterando tabelas
- Inserindo dados
- O comando SELECT
- A cláusula WHERE
- Removendo registros
- Atualizando tabelas

Fundação CECIERJ
Consórcio cederj

Comandos SHOW

- Usando o programa **`mysql`** é possível conhecer as diversas estruturas de bancos de dados usando comandos **`SHOW`**
  - **`show databases`**
    - Lista todos os bancos de dados ao qual o usuário tem acesso
  - **`show tables`**
    - Lista todas as tabelas do banco de dados correntemente selecionado
    - Se um banco de dados não foi mencionado como argumento do comando **`mysql`**, é preciso usar o comando use *banco*
  - **`show columns from`** *tabela*
  - **`show index from`** *tabela*
  - **`show status`**

Exemplo

```
esperanc@claudio:~$ mysql -ualuno -paluno
...
mysql> show databases;
+----------+
| Database |
+----------+
| prog2    |
+----------+
1 row in set (0.00 sec)
mysql> use prog2;
Database changed
mysql> show tables;
+-----------------+
| Tables_in_prog2 |
+-----------------+
| cliente         |
| conta           |
| possui          |
+-----------------+
3 rows in set (0.00 sec)
```

Abrir mysql

Comandos SHOW (continuação)

- **`show columns from`** *tabela*
  - Lista todas as colunas (atributos) de *tabela*
- **`show index from`** *tabela*
  - Lista todos os índices criados para *tabela*
  - Observe que se uma tabela possui uma chave primária, então o MySql automaticamente cria um índice para essa chave
- **`show status`**
  - Lista algumas informações e estatísticas de desempenho do servidor MySql

Exemplo

```
mysql> show columns from cliente;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| id       | int(11)     |      | PRI | NULL    | auto_increment |
| nome     | varchar(50) |      | MUL |         |                |
| endereco | varchar(80) | YES  |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql> show index from cliente;
+---------+------------+-------------+--------------+...
| Table   | Non_unique | Key_name    | Seq_in_index |
+---------+------------+-------------+--------------+...
| cliente |          0 | PRIMARY     |            1 |
| cliente |          1 | nomecliente |            1 |
+---------+------------+-------------+--------------+...
```

Abrir mysql

Removendo bancos de dados e tabelas

- O comando **`drop`** pode ser usado para remover tabelas ou bancos de dados inteiros
  - **`drop database`** *banco*
    - Remove o banco de dados *banco*
    - Todas as tabelas do banco de dados são removidas
  - **`drop table`** *tabela*
    - Remove *tabela* do banco de dados corrente

Exemplo

- Obs: O exemplo abaixo assume que o usuário tem privilégio para criar bancos de dados

```
mysql> create database teste;
Query OK, 1 row affected (0.00 sec)

mysql> use teste;
Database changed
mysql> create table tabela1 (a int, b char(10));
Query OK, 0 rows affected (0.02 sec)

mysql> create table tabela2 (x int, y decimal(20,3));
Query OK, 0 rows affected (0.02 sec)

mysql> drop table tabela2;
Query OK, 0 rows affected (0.00 sec)

mysql> drop database teste;
Query OK, 0 rows affected (0.00 sec)
```

Abrir mysql

Alterando tabelas

- O comando **`ALTER`** pode ser usado para alterar o esquema de uma tabela, inserindo ou removendo atributos ou índices
  - **`alter table`** *tabela* **`add column`** **`(`***atributo tipo***`)`**
    - Adiciona um novo *atributo* à *tabela*
  - **`alter table`** *tabela* **`drop column`** *atributo*
    - Remove um *atributo* da *tabela*
  - **`alter table`** *tabela* **`add index`** *índice (atributos)*
    - Adiciona à *tabela* um novo *índice* definido sobre uma lista de *atributos*
  - **`alter table`** *tabela* **`drop index`** *índice*
    - Remove um *índice* da *tabela*

Exemplo

```
mysql> create table test (a int, b char(10));
Query OK, 0 rows affected (0.08 sec)

mysql> alter table test add column (c timestamp);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> alter table test drop column b;
Query OK, 0 rows affected (0.10 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show columns from test;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| a     | int(11)      | YES  | MUL | NULL    |       |
| c     | timestamp(14)| YES  |     | NULL    |       |
+-------+--------------+------+-----+---------+-------+
```

Abrir mysql

Inserindo dados

- Uma vez criado o banco de dados, é hora de inserir dados nas tabelas
- O comando **`insert`** do SQL é o mais usado para isso
- Há duas variantes desse comando
  - **`insert into`** *tabela* **`values`** **`(`***valor1, valor2, ...valorN***`)`**
    - Cria novo registro na *tabela* com as colunas *1* a *N* preenchidas com os valores *valor1* a *valorN*
    - Os valores são dados na mesma ordem em que os atributos foram definidos no esquema da tabela
  - **`insert into`** *tabela* **`set`** *attrib=valor, ..., attrib=valor*
    - Cria novo registro na *tabela* onde cada atributo *atrib* citado no comando é preenchido com o valor respectivo
    - Os nomes dos atributos podem ser listados em qualquer ordem

Valores de atributos *string*

- O valor para um atributo do tipo cadeia de caracteres (**`char`** ou **`varchar`**) tem que ser escrito entre apóstrofes (') ou aspas (")
  - Caso a cadeia inclua um caractere aspas, use apóstrofes para delimitá-la e vice-versa
  - Outra opção é preceder o apóstrofe ou aspas por uma barra inclinada
  - Ex.:
    - **`"José D'Almeida"`**
    - **`'Pizza "DOC"'`**
    - **`'Pingo D\'Água'`**

Valores de atributos de tempo

- Ao especificar um valor para um atributo do tipo **`timestamp`** ou **`datetime`**, este pode ser escrito:
  - Como um número nos formatos **`AAMMDD`**, **`AAAAMMDD`**, **`AAMMDDhhmmss`** ou **`AAAAMMDDhhmmss`**
  - Como uma string nos formatos '**`AA-MM-DD`**', '**`AAAA-MM-DD`**', '**`AA-MM-DD hh:mm:ss`**' ou '**`AAAA-MM-DD hh:mm:ss`**'
  - Onde
    - **`AA`** ou **`AAAA`** = ano com 2 ou 4 dígitos
    - **`MM`** = mês (entre 01 e 12)
    - **`DD`** = dia (entre 01 e 31)
    - **`hh`** = hora (entre 00 e 23)
    - **`mm`** = minutos (entre 00 e 59)
    - **`ss`** = segundos (entre 00 e 59)

Valores de atributos de tempo (continuação)

- Lembre que atributos do tipo **`timestamp`** têm como default o instante em que a alteração do registro foi feita
- Diferentemente do tipo **`timestamp`**, o atributo **`datetime`** pode ter um default explícito
- Outros tipos de atributos de tempo são:
  - **`date`** : apenas datas (sem hora)
  - **`time`** : apenas hora (sem data)
- Para esses os formatos para especificação de valores são semelhantes ao do tipo datetime, apenas omitindo a parte das horas ou das datas

Exemplo

```
mysql> create table test (a int not null auto_increment,
    ->     b varchar(20) default '**',
    ->     c datetime default '2001-01-01 00:00:00',
    ->     primary key (a));
Query OK, 0 rows affected (0.02 sec)
mysql> insert into test values ();
Query OK, 1 row affected (0.00 sec)
mysql> insert into test set c = 951225190000;
Query OK, 1 row affected (0.00 sec)
mysql> insert into test values (null, "abc",
    ->                          "1958-02-28 23:00:00");
Query OK, 1 row affected (0.00 sec)
mysql> select * from test;
+---+------+---------------------+
| a | b    | c                   |
+---+------+---------------------+
| 1 | **   | 2001-01-01 00:00:00 |
| 2 | **   | 1995-12-25 19:00:00 |
| 3 | abc  | 1958-02-28 23:00:00 |
+---+------+---------------------+
```

Abrir mysql

Atributos `auto_increment`

- O modificador **`auto_increment`** indica que um valor único para o atributo vai ser gerado a cada inserção
  - Ao especificar o valor para o atributo deve-se usar **`null`**
  - Para saber o valor gerado automaticamente, pode-se usar a função **`last_insert_id()`** do SQL
    - Para funcionar, a função deve ser invocada imediatamente após a inserção
- Ex.:

```
mysql> insert into test values ();
Query OK, 1 row affected (0.00 sec)
mysql> select last_insert_id();
+------------------+
| last_insert_id() |
+------------------+
|                4 |
+------------------+
```

Abrir mysql

O comando SELECT

- É o comando genérico para consultas em SQL
- O resultado de um comando **`select`** é sempre uma tabela
- Possui várias cláusulas e pode ser bastante complexo
  - Uma abordagem mais detalhada será vista na próxima aula
- Já vimos duas utilizações simples:
  - **`select * from`** *tabela*
    - Lista todas as linhas e colunas da *tabela*
    - Para listar apenas algumas colunas pode-se usar select *atrib1, ... , atribN* from tabela
  - **`select`** *expressão*
    - Retorna uma tabela com uma linha e uma coluna contendo o resultado da avaliação da expressão
    - *Expressão* tem sintaxe semelhante à do PHP, embora as funções sejam próprias do SQL

Expressões

- Operadores mais comuns:
  - Comparação: = (igual) , != *ou* <> (diferente), >, <, >=, <=
  - Lógicos: and *ou* &&, or *ou* ||, not *ou* !, xor (ou exclusivo)
  - Aritméticos: +, -, *, /, % (resto)
  - Bit-a-bit: | , & , ^ (ou exclusivo)
- Algumas funções:
  - Matemáticas: LOG(x), EXP(x), COS(x), SIN(x), ABS(x), PI(x), POW(x,y), ATAN(y,x)
  - String: CONCAT(s1,...,sN), LENGTH(s),LOCATE(s1,s2)
  - Tempo: CURDATE(),CURTIME()

Exemplos

```
mysql> select a,b from test;
+---+------+
| a | b    |
+---+------+
| 1 | **   |
| 2 | **   |
| 3 | abc  |
+---+------+

mysql> select concat(1+3*log(2)," xyz");
+---------------------------+
| concat(1+3*log(2)," xyz") |
+---------------------------+
| 3.0794415416798 xyz       |
+---------------------------+

mysql> select 1>0 and 2>1;
+-------------+
| 1>0 and 2>1 |
+-------------+
|           1 |
+-------------+
```

Abrir mysql

A cláusula WHERE

- É usada em conjunto com o comando select e outros comandos SQL
- Permite especificar uma condição que restringe quais linhas da tabela devem ser consideradas
  - Condições são expressões booleanas envolvendo normalmente atributos de tabelas
- Assim,

`select * from` *tabela* **`where`** *condição*

retorna apenas as linhas de *tabela* que satisfazem *condição*

Exemplo

```
mysql> select * from test;
+------+------+
| a    | b    |
+------+------+
|    1 | a    |
|    2 | b    |
|    3 | c    |
+------+------+

mysql> select b from test where a > 2;
+------+
| b    |
+------+
| c    |
+------+

mysql> select a from test where b < "b" or a > 2;
+------+
| a    |
+------+
|    1 |
|    3 |
+------+
```

Abrir mysql

Removendo registros

- Para remover registros (linhas) de tabelas, usa-se o comando **`delete`**
  - Não confundir com drop que remove tabelas ou bancos de dados inteiros
- Formato:
  - **`delete from`** *tabela*
    - Remove todas as linhas da tabela
  - **`delete from`** *tabela* **`where`** *condição*
    - Remove apenas as linhas da tabela que satisfazem *condição*

Exemplo

```
mysql> select * from test;
+------+------+
| a    | b    |
+------+------+
|    1 | a    |
|    2 | b    |
|    3 | c    |
+------+------+

mysql> delete from test where b = 'b';
Query OK, 1 row affected (0.00 sec)

mysql>  select * from test;
+------+------+
| a    | b    |
+------+------+
|    1 | a    |
|    3 | c    |
+------+------+

mysql> delete from test;
Query OK, 2 rows affected (0.00 sec)

mysql> select * from test;
Empty set (0.00 sec)
```

Abrir mysql

Atualizando tabelas

- O comando update permite atualizar todos ou alguns registros de uma tabela
- Formato:
  - **`update`** *tabela* **`set`** *atrib1=expr1, ..., atribN=exprN*
    - Atualiza todas as linhas da tabela substituindo os valores dos atributos dados pelos das expressões respectivas
  - **`update`** *tabela* **`set`** *atrib1=expr1, ..., atribN=exprN* **`where`** *condição*
    - Idem acima, mas afeta apenas as linhas para as quais *condição* é verdadeira

Exemplo

```
mysql> select * from test;
+------+------+
| a    | b    |
+------+------+
|    1 | a    |
|    2 | b    |
|    3 | c    |
+------+------+

mysql> update test set a=a+1, b=upper(b);
Query OK, 3 rows affected (0.00 sec)
Rows matched: 3  Changed: 3  Warnings: 0

mysql> update test set a = 99 where b = "B";
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from test;
+------+------+
| a    | b    |
+------+------+
|    2 | A    |
|   99 | B    |
|    4 | C    |
+------+------+
```

Abrir mysql