3.2. Fazendo Consultas

Tenha certeza que você está conectado ao servidor, como discutido na seção anterior. Isto feito, não será selecionado nenhum banco de dados para trabalhar, mas não tem problemas. Neste momento, é mais importante saber um pouco sobre como fazer consultas do que já criar tabelas, carregar dados para elas, e recuperar dados delas. Esta seção descreve os princípios básicos da entrada de comandos, usando diversas consultas você pode tentar se familiarizar com o funcionamento do **mysql**.

Aqui está um comando simples que solicita ao servidor seu número de versão e a data atual. Digite-o como visto abaixo seguindo o prompt **mysql>** e digite a tecla RETURN:

```
mysql> SELECT VERSION(), CURRENT_DATE;
+--------------+--------------+
| version()    | CURRENT_DATE |
+--------------+--------------+
| 3.22.20a-log | 1999-03-19   |
+--------------+--------------+
1 row in set (0.01 sec)
mysql>
```

Esta consulta ilustra várias coisas sobre o **mysql**:

- Um comando normalmente consiste de uma instrução SQL seguida por um ponto e vírgula. (Existem algumas exceções onde um ponto e vírgula podem ser omitidos. **QUIT** mencionado anteriormente, é um deles. Saberemos de outros mais tarde.)
- Quando você emite um comando, o **mysql** o envia para o servidor para execução e mostra os resultados, depois imprime outro prompt **mysql>** para indicar que está pronto para outro comando.
- O **mysql** mostra a saída da consulta em forma tabular (linhas e colunas). A primeira linha contém rótulos para as colunas. As linhas seguintes são o resultado da consulta. Normalmente, rótulos de colunas são os nomes das colunas que você busca das tabelas do banco de dados. Se você está recuperando o valor de uma expressão no lugar de uma coluna de tabela (como no exemplo já visto), o **mysql** rotula a coluna usando a própria expressão.
- O **mysql** mostra quantas linhas foram retornadas e quanto tempo a consulta levou para executar, o que lhe dá uma vaga idéia da performance do servidor. Estes valores são impreciso porque eles representam tempo de relógio (Não tempo de CPU ou de máquina), e porque eles são afetados pelos fatores como a carga do servidor e latência de rede. (Para resumir, a linha ``rows in set'' não é mostrada nos exemplos seguintes deste capítulo.)

Palavras Chave podem ser entradas em qualquer caso de letra. As seguintes consultas são equivalentes:

```
mysql> SELECT VERSION(), CURRENT_DATE;
mysql> select version(), current_date;
mysql> SeLeCt vErSiOn(), current_DATE;
```

Aqui está outra consulta. Ela demonstra que você pode usar o **mysql** como uma calculadora simples:

```
mysql> SELECT SIN(PI()/4), (4+1)*5;
+-------------+---------+
| SIN(PI()/4) | (4+1)*5 |
```

```
+-------------+---------+
|    0.707107 |      25 |
+-------------+---------+
```

As consultas mostradas até agora têm sido instruções relativamente pequenas, de uma linha. Você pode também entrar com múltiplas instruções em uma única linha. Basta finalizar cada uma com um ponto e vírgula:

```
mysql> SELECT VERSION(); SELECT NOW();
+--------------+
| VERSION()    |
+--------------+
| 3.22.20a-log |
+--------------+

+---------------------+
| NOW()               |
+---------------------+
| 1999-03-19 00:15:33 |
+---------------------+
```

Um comando não necessita estar todo em uma única linha, então comandos extensos que necessitam de várias linhas não são um problema. O **mysql** determina onde sua instrução termina através do ponto e vírgula terminador, e não pelo final da linha de entrada. (Em outras palavras, o **myqsl** aceita entradas de livre formato: Ele coleta linhas de entrada mas não as executa até chegar o ponto e vírgula.)

Aqui está uma instrução simples usando múltiplas linhas:

```
mysql> SELECT
    -> USER()
    -> ,
    -> CURRENT_DATE;
+--------------------+--------------+
| USER()             | CURRENT_DATE |
+--------------------+--------------+
| joesmith@localhost | 1999-03-18   |
+--------------------+--------------+
```

Neste exemplo, note como o prompt altera de **mysql>** para **->** depois de você entrar a primeira linha de uma consulta com múltiplas linhas. Isto é como o **mysql** indica que ainda não achou uma instrução completa e está esperando pelo resto. O prompt é seu amigo, porque ele fornece um retorno valioso. Se você usa este retorno, você sempre estará ciente do que o **mysql** está esperando.

Se você decidir que não deseja executar um comando que está no meio do processo de entrada, cancele-o digitando **\c**:

```
mysql> SELECT
    -> USER()
    -> \c
mysql>
```

Note o prompt aqui também. Ele troca para o `mysql>` depois de você digitar `\c`, fornecendo retorno para indicar que o `mysql` está pronto para um novo comando.

A seguinte tabela mostra cada dos prompts que você pode ver e resume o que ele significa sobre o estado em que o `mysql` se encontra:

Prompt	Significado
`mysql>`	Pronto para novo comando.
`->`	Esperando pela próxima linha de comando com múltiplas linhas.
`'>`	Esperando pela próxima linha, coletando uma string que comece com uma aspas simples (‘`'`’).
`">`	Esperando pela próxima linha, coletando uma string que comece com aspas duplas (‘`"`’).
`` `> ```	Esperando pela próxima linha, coletando uma string que comece com crase (‘``` ` ```’).

É muito comum instruções multi-linhas ocorrerem por acidente quando você pretende publicar um comando em uma única linha, mas esquece o ponto e vírgula terminador. Neste caso,o `mysql` espera por mais entrada:

```
mysql> SELECT USER()
    ->
```

Se isto ocorrer com você (acha que entrou uma instrução mas a única resposta é um prompt `->`), o mais provável é que o `mysql` está esperando pelo ponto e vírgula. Se você não perceber o que o prompt está lhe dizendo, você pode parar por um tempo antes de entender o que precisa fazer. Entre com um ponto e vírgula para completar a instrução, e o `mysql` irá executá-la:

```
mysql> SELECT USER()
    -> ;
+--------------------+
| USER()             |
+--------------------+
| joesmith@localhost |
+--------------------+
```

O prompt `'>` e `">` ocorrem durante a coleta de strings. No MySQL, você pode escrever strings utilizando os caracteres ‘`'`’ ou ‘`"`’ (por exemplo, `'hello'` ou `"goodbye"`), e o `mysql` permite a entrada de strings que consomem múltiplas linhas. Quando você ver um prompt `'>` ou `">`, significa que você digitou uma linha contendo uma string que começa com um caracter de aspas ‘`'`’ ou ‘`"`’ mas ainda não entrou com a aspas que termina a string. Isto é bom se você realmente está entrando com uma string com múltiplas linhas, mas qual é a probalidade disto acontecer ? Não muita. Geralmente, os prompts `'>` e `">` indicam que você, por algum descuido, esqueceu algum caracter de aspas. Por exemplo:

```
mysql> SELECT * FROM minha_tabela WHERE nome = "Smith AND idade < 30;
    ">
```

Se você entrar esta sentença `SELECT`, apertar ENTER e esperar pelo resultado, nada irá acontecer. Em vez de se perguntar o porquê desta query demorar tanto tempo, perceba a pista fornecida pelo prompt `">`. Ele lhe diz que o `mysql` espera pelo resto de uma string não terminada. (Você ve o erro na declaração? Falta a segunda aspas na string `"Smith`.)

O que fazer neste ponto ? A coisa mais simples é cancelar o comando. Entretanto, você não pode simplesmente digitar `\c` neste caso, porque o `mysql` o intrerpreta como parte da string que está coletando! Digite o caracter de aspas para fechar (então o `mysql` sabe que você fechou a string), então digite `\c`:

```
mysql> SELECT * FROM minha_tabela WHERE nome = "Smith AND idade < 30;
    "> "\c
mysql>
```

O prompt volta para **mysql>**, indicando que o **mysql** está pronto para um novo comando.

O prompt **`>** é similar aos prompts **'>** e **">**, mas indica que você começou mas não completou um identificados citado com o sinal de crase.

É importante saber o que os prompts **'>**, **">** e **`>** significam, porque se você entrar sem querer com uma string sem terminação, quaisquer linhas seguintes que forem digitadas serão ignoradas pelo **mysql** --- incluindo uma linha contendo **QUIT**! Isto pode ser um pouco confuso, especialmente se você não sabe que você precisa fornecer as aspas finais antes poder cancelar o comando atual.
