MySQL - Guia Rápido

Anúncios

Página anterior Próxima página

MySQL - Introdução

O que é um banco de dados?

Um banco de dados é um aplicativo separado que armazena uma coleção de dados. Cada banco de dados tem uma ou mais APIs distintas para criar, acessar, gerenciar, pesquisar e replicar os dados que ele contém.

Outros tipos de armazenamentos de dados também podem ser usados, como arquivos no sistema de arquivos ou grandes tabelas de hash na memória, mas a busca e a gravação de dados não seriam tão rápidas e fáceis com esse tipo de sistema.

Atualmente, usamos sistemas de gerenciamento de banco de dados relacionais (RDBMS) para armazenar e gerenciar grandes volumes de dados. Isso é chamado de banco de dados relacional porque todos os dados são armazenados em tabelas diferentes e as relações são estabelecidas usando chaves primárias ou outras chaves conhecidas como **Chaves Estrangeiras** .

Um **Sistema de Gerenciamento Relacional de DataBase (RDBMS)** é um software que

- Permite que você implemente um banco de dados com tabelas, colunas e índices.
- Garante a integridade referencial entre linhas de várias tabelas.
- Atualiza os índices automaticamente.
- Interpreta uma consulta SQL e combina informações de várias tabelas.

Terminologia RDBMS

Antes de prosseguirmos para explicar o sistema de banco de dados MySQL, vamos revisar algumas definições relacionadas ao banco de dados.

- **Banco de dados** - Um banco de dados é uma coleção de tabelas, com dados relacionados.

Tabela - Uma tabela é uma matriz com dados. Uma tabela em um banco de dados parece uma simples planilha.

Coluna - uma coluna (elemento de dados) contém dados de um mesmo tipo, por exemplo, o código postal da coluna.

Linha - Uma linha (= tupla, entrada ou registro) é um grupo de dados relacionados, por exemplo, os dados de uma assinatura.

Redundância - Armazenando dados duas vezes, de forma redundante, para tornar o sistema mais rápido.

Chave Primária - Uma chave primária é exclusiva. Um valor de chave não pode ocorrer duas vezes em uma tabela. Com uma chave, você pode encontrar apenas uma linha.

Chave estrangeira - Uma chave estrangeira é o pino de vinculação entre duas tabelas.

Chave composta - Uma chave composta (chave composta) é uma chave que consiste em várias colunas, porque uma coluna não é suficientemente exclusiva.

Índice - Um índice em um banco de dados se assemelha a um índice na parte de trás de um livro.

Integridade referencial - **Integridade** referencial garante que um valor de chave estrangeira sempre aponte para uma linha existente.

Banco de dados MySQL

O MySQL é um RDBMS rápido e fácil de usar sendo usado por muitas pequenas e grandes empresas. O MySQL é desenvolvido, comercializado e suportado pela MySQL AB, que é uma empresa sueca. O MySQL está se tornando tão popular por causa de muitas boas razões -

O MySQL é lançado sob uma licença de código aberto. Então você não tem nada a pagar para usá-lo.

O MySQL é um programa muito poderoso por si só. Ele lida com um grande subconjunto da funcionalidade dos pacotes de banco de dados mais caros e poderosos.

O MySQL usa uma forma padrão da conhecida linguagem de dados SQL.

O MySQL funciona em muitos sistemas operacionais e com muitas linguagens, incluindo PHP, PERL, C, C ++, JAVA, etc.

O MySQL funciona muito rapidamente e funciona bem mesmo com grandes conjuntos de dados.

O MySQL é muito amigável ao PHP, a linguagem mais apreciada para o desenvolvimento web.

O MySQL suporta grandes bancos de dados, até 50 milhões de linhas ou mais em uma tabela. O limite de tamanho de arquivo padrão para uma tabela é de 4 GB, mas você pode aumentá-lo (se o sistema operacional permitir) para um limite teórico de 8 milhões de terabytes (TB).

O MySQL é personalizável. A licença GPL de código aberto permite que os programadores modifiquem o software MySQL para ajustar-se a seus próprios ambientes específicos.

Antes de você começar

Antes de começar este tutorial, você deve ter um conhecimento básico das informações abordadas em nossos tutoriais de PHP e HTML.

Este tutorial se concentra fortemente no uso do MySQL em um ambiente PHP. Muitos exemplos dados neste tutorial serão úteis para programadores PHP.

Recomendamos que você verifique nosso tutorial PHP para sua referência.

MySQL - Instalação

Todos os downloads para o MySQL estão localizados em Downloads MySQL . Escolha o número da versão do **MySQL Community Server** que é necessário junto com a plataforma na qual você irá executá-lo.

Instalando o MySQL no Linux / UNIX

A maneira recomendada de instalar o MySQL em um sistema Linux é via RPM. A MySQL AB disponibiliza os seguintes RPMs para download em seu site -

MySQL - O servidor de banco de dados MySQL gerencia os bancos de dados e tabelas, controla o acesso do usuário e processa as consultas SQL.

MySQL-client - programas clientes MySQL, que permitem conectar-se e interagir com o servidor.

MySQL-devel - Bibliotecas e arquivos de cabeçalho que são úteis ao compilar outros programas que usam o MySQL.

MySQL-shared - Bibliotecas compartilhadas para o cliente MySQL.

MySQL-bench - Ferramentas de teste de desempenho e benchmark para o servidor de banco de dados MySQL.

Os RPMs do MySQL listados aqui são todos construídos em um **sistema SuSE Linux** , mas eles geralmente funcionam em outras variantes do Linux sem dificuldades.

Agora, você precisará seguir os passos abaixo, para prosseguir com a instalação -

Faça o login no sistema usando o usuário **root** .

Alterne para o diretório que contém os RPMs.

Instale o servidor de banco de dados MySQL executando o seguinte comando. Lembre-se de substituir o nome do arquivo em itálico pelo nome do arquivo do seu RPM.

```
[root@host]# rpm -i MySQL-5.0.9-0.i386.rpm
```

O comando acima se encarrega de instalar o servidor MySQL, criando um usuário do MySQL, criando a configuração necessária e iniciando o servidor MySQL automaticamente.

Você pode encontrar todos os binários relacionados ao MySQL em / usr / bin e / usr / sbin. Todas as tabelas e bancos de dados serão criados no diretório / var / lib / mysql.

A caixa de código a seguir tem uma etapa opcional, mas recomendada, para instalar os RPMs restantes da mesma maneira -

```
[root@host]# rpm -i MySQL-client-5.0.9-0.i386.rpm
[root@host]# rpm -i MySQL-devel-5.0.9-0.i386.rpm
[root@host]# rpm -i MySQL-shared-5.0.9-0.i386.rpm
[root@host]# rpm -i MySQL-bench-5.0.9-0.i386.rpm
```

Instalando o MySQL no Windows

A instalação padrão em qualquer versão do Windows agora é muito mais fácil do que costumava ser, já que o MySQL agora vem bem empacotado com um instalador. Simplesmente baixe o pacote do instalador, descompacte-o em qualquer lugar e execute o arquivo setup.exe.

O instalador padrão setup.exe irá orientá-lo através do processo trivial e, por padrão, instalará tudo em C: \ mysql.

Teste o servidor ativando-o a partir do prompt de comando na primeira vez. Vá para a localização do **servidor mysqld** que é provavelmente C: \ mysql \ bin e digite -

```
mysqld.exe --console
```

NOTA - Se você está no NT, então você terá que usar o mysqld-nt.exe ao invés do mysqld.exe

Se tudo correr bem, você verá algumas mensagens sobre inicialização e **InnoDB** . Caso contrário, você pode ter um problema de permissão. Certifique-se de que o diretório que contém seus dados esteja acessível para qualquer usuário (provavelmente MySQL) em que os processos do banco de dados sejam executados.

O MySQL não se adicionará ao menu iniciar, e também não há uma maneira GUI particularmente agradável de parar o servidor. Portanto, se você tende a iniciar o servidor clicando duas vezes no executável mysqld, lembre-se de interromper o processo manualmente usando mysqladmin, Task List, Task Manager ou outros meios específicos do Windows.

Verificando a instalação do MySQL

Depois que o MySQL foi instalado com sucesso, as tabelas base foram inicializadas e o servidor foi iniciado: você pode verificar se tudo está funcionando como deveria, através de alguns testes simples.

Use o utilitário mysqladmin para obter o status do servidor

Use **mysqladmin** binary para verificar a versão do servidor. Este binário estaria disponível em / usr / bin no linux e em C: \ mysql \ bin no windows.

```
[root@host]# mysqladmin --version
```

Isso produzirá o seguinte resultado no Linux. Pode variar dependendo da sua instalação -

```
mysqladmin  Ver 8.23 Distrib 5.0.9-0, for redhat-linux-gnu on i386
```

Se você não receber essa mensagem, pode haver algum problema em sua instalação e você precisará de ajuda para consertá-la.

Execute comandos SQL simples usando o cliente MySQL

Você pode se conectar ao seu servidor MySQL através do cliente MySQL e usando o comando **mysql** . Neste momento, você não precisa fornecer nenhuma senha, pois, por padrão, ela será definida como em branco.

Você pode apenas usar o seguinte comando -

```
[root@host]# mysql
```

Deve ser recompensado com um prompt mysql>. Agora, você está conectado ao servidor MySQL e pode executar todos os comandos SQL no prompt mysql> da seguinte maneira -

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
|   mysql  |
|   test   |
+----------+
2 rows in set (0.13 sec)
```

Etapas de pós-instalação

O MySQL vem com uma senha em branco para o usuário root do MySQL. Assim que você instalou com sucesso o banco de dados e o cliente, é necessário definir uma senha de root conforme determinado no bloco de código a seguir -

```
[root@host]# mysqladmin -u root password "new_password";
```

Agora para fazer uma conexão com seu servidor MySQL, você teria que usar o seguinte comando -

```
[root@host]# mysql -u root -p
Enter password:*******
```

Os usuários do UNIX também desejarão colocar seu diretório do MySQL em seu PATH, para que você não tenha que continuar digitando o caminho completo toda vez que quiser usar o cliente de linha de comando.

Para bash, seria algo como -

```
export PATH = $PATH:/usr/bin:/usr/sbin
```

Executando o MySQL no Boot Time

Se você deseja executar o servidor MySQL no momento da inicialização, certifique-se de ter a seguinte entrada no arquivo /etc/rc.local.

```
/etc/init.d/mysqld start
```

Além disso, você deve ter o binário mysqld no diretório /etc/init.d/.

MySQL - Administração

Executando e desligando o servidor MySQL

Primeiro verifique se o seu servidor MySQL está em execução ou não. Você pode usar o seguinte comando para verificar isso -

```
ps -ef | grep mysqld
```

Se o seu MySql estiver rodando, você verá o processo **mysqld** listado no seu resultado. Se o servidor não estiver em execução, você poderá iniciá-lo usando o seguinte comando -

```
root@host# cd /usr/bin
./safe_mysqld &
```

Agora, se você quer desligar um servidor MySQL já em execução, então você pode fazê-lo usando o seguinte comando -

```
root@host# cd /usr/bin
./mysqladmin -u root -p shutdown
Enter password: ******
```

Configurando uma conta de usuário do MySQL

Para adicionar um novo usuário ao MySQL, você só precisa adicionar uma nova entrada à tabela **user** no banco de dados **mysql** .

O programa a seguir é um exemplo de adição de um novo usuário **convidado** com os privilégios SELECT, INSERT e UPDATE com a senha **guest123;** a consulta SQL é -

```
root@host# mysql -u root -p
Enter password:*******
mysql> use mysql;
Database changed

mysql> INSERT INTO user
   (host, user, password,
   select_priv, insert_priv, update_priv)
   VALUES ('localhost', 'guest',
   PASSWORD('guest123'), 'Y', 'Y', 'Y');
Query OK, 1 row affected (0.20 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT host, user, password FROM user WHERE user = 'guest';
+-----------+---------+------------------+
|    host   |   user  |     password     |
+-----------+---------+------------------+
| localhost |  guest  | 6f8c114b58f2ce9e |
+-----------+---------+------------------+
1 row in set (0.00 sec)
```

Ao adicionar um novo usuário, lembre-se de criptografar a nova senha usando a função PASSWORD () fornecida pelo MySQL. Como você pode ver no exemplo acima, a senha mypass é criptografada para 6f8c114b58f2ce9e.

Observe a declaração FLUSH PRIVILEGES. Isso informa ao servidor para recarregar as tabelas de permissões. Se você não usá-lo, então você não será capaz de se conectar ao MySQL usando a nova conta de usuário, pelo menos até que o servidor seja reinicializado.

Você também pode especificar outros privilégios para um novo usuário definindo os valores das colunas a seguir na tabela do usuário para 'Y' ao executar a consulta INSERT ou atualizá-los posteriormente usando a consulta UPDATE.

- Select_priv
- Insert_priv
- Update_priv
- Delete_priv
- Create_priv
- Drop_priv
- Reload_priv
- Shutdown_priv
- Process_priv
- File_priv
- Grant_priv

References_priv

Index_priv

Alter_priv

Outra maneira de adicionar conta de usuário é usando o comando GRANT SQL. O exemplo a seguir adicionará o usuário **zara** com a senha **zara123** para um banco de dados específico, chamado **TUTORIALS** .

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use mysql;
Database changed

mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
   -> ON TUTORIALS.*
   -> TO 'zara'@'localhost'
   -> IDENTIFIED BY 'zara123';
```

Isso também criará uma entrada na tabela de banco de dados MySQL chamada como **usuário** .

NOTA - O MySQL não finaliza um comando até que você dê um ponto e vírgula (;) no final do comando SQL.

A configuração do arquivo /etc/my.cnf

Na maioria dos casos, você não deve tocar neste arquivo. Por padrão, ele terá as seguintes entradas -

```
[mysqld]
datadir = /var/lib/mysql
socket = /var/lib/mysql/mysql.sock

[mysql.server]
user = mysql
basedir = /var/lib

[safe_mysqld]
err-log = /var/log/mysqld.log
pid-file = /var/run/mysqld/mysqld.pid
```

Aqui, você pode especificar um diretório diferente para o log de erros, caso contrário você não deve alterar nenhuma entrada nesta tabela.

Comando Administrativo do MySQL

Aqui está a lista dos importantes comandos do MySQL, que você usará periodicamente para trabalhar com o banco de dados MySQL -

USE Databasename - Isso será usado para selecionar um banco de dados na área de trabalho do MySQL.

MOSTRAR BANCOS DE DADOS - Lista os bancos de dados que são acessíveis pelo MySQL DBMS.

SHOW TABLES - Mostra as tabelas no banco de dados depois que um banco de dados for selecionado com o comando use.

SHOW COLUMNS FROM nome da *tabela:* Mostra os atributos, tipos de atributos, informações chave, se NULL é permitido, padrões e outras informações para uma tabela.

SHOW INDEX FROM tablename - Apresenta os detalhes de todos os índices na tabela, incluindo a PRIMARY KEY.

SHOW TABLE STATUS LIKE nome da tabela \ G - Relata detalhes do desempenho e das estatísticas do MySQL DBMS.

No próximo capítulo, discutiremos a respeito de como o PHP Syntax é usado no MySQL.

MySQL - Sintaxe PHP

O MySQL funciona muito bem em combinação de várias linguagens de programação como PERL, C, C ++, JAVA e PHP. Fora desses idiomas, o PHP é o mais popular por causa de seus recursos de desenvolvimento de aplicativos da web.

Este tutorial se concentra fortemente no uso do MySQL em um ambiente PHP. Se você está interessado no MySQL com PERL, então você pode considerar ler o Tutorial PERL
.

O PHP fornece várias funções para acessar o banco de dados MySQL e manipular os registros de dados dentro do banco de dados MySQL. Você precisaria chamar as funções PHP da mesma maneira que chama qualquer outra função PHP.

As funções do PHP para uso com o MySQL possuem o seguinte formato geral -

```
mysql_function(value,value,...);
```

A segunda parte do nome da função é específica da função, geralmente uma palavra que descreve o que a função faz. A seguir estão duas das funções, que usaremos em nosso tutorial -

```
mysqli_connect($connect);
mysqli_query($connect,"SQL statement");
```

O exemplo a seguir mostra uma sintaxe genérica do PHP para chamar qualquer função do MySQL.

```
<html>
   <head>
      <title>PHP with MySQL</title>
   </head>

   <body>
      <?php
         $retval = mysql_function(value, [value,...]);
         if( !$retval ) {
            die ( "Error: a related error message" );
         }
         // Otherwise MySQL  or PHP Statements
```

```
      ?>
   </body>
</html>
```

A partir do próximo capítulo, veremos toda a importante funcionalidade do MySQL junto com o PHP.

MySQL - Conexão

Conexão MySQL Usando MySQL Binary

Você pode estabelecer o banco de dados MySQL usando o binário **mysql** no prompt de comando.

Exemplo

Aqui está um exemplo simples para se conectar ao servidor MySQL a partir do prompt de comando -

```
[root@host]# mysql -u root -p
Enter password:******
```

Isto lhe dará o prompt de comando mysql> onde você poderá executar qualquer comando SQL. A seguir, o resultado do comando acima -

O bloco de código a seguir mostra o resultado do código acima -

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2854760 to server version: 5.0.9

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

No exemplo acima, usamos **root** como usuário, mas você também pode usar qualquer outro usuário. Qualquer usuário poderá executar todas as operações SQL, que são permitidas a esse usuário.

Você pode se desconectar do banco de dados MySQL a qualquer momento usando o comando **exit** no prompt mysql>.

```
mysql> exit
Bye
```

Conexão MySQL Usando Script PHP

O PHP fornece a função **mysql_connect ()** para abrir uma conexão com o banco de dados. Essa função usa cinco parâmetros e retorna um identificador de link do MySQL em sucesso ou FALSE em falha.

Sintaxe

```
connection mysql_connect(server,user,passwd,new_link,client_flag);
```

Sr.	Parâmetro e Descrição

Não.	
1	**servidor** Opcional - O nome do host que está executando o servidor de banco de dados. Se não for especificado, o valor padrão será **localhost: 3306** .
2	**do utilizador** Opcional - O nome de usuário que acessa o banco de dados. Se não for especificado, o padrão será o nome do usuário que possui o processo do servidor.
3	**passwd** Opcional - A senha do usuário que acessa o banco de dados. Se não for especificado, o padrão será uma senha vazia.
4	**new_link** Opcional - Se uma segunda chamada for feita para mysql_connect () com os mesmos argumentos, nenhuma nova conexão será estabelecida; em vez disso, o identificador da conexão já aberta será retornado.
5	**client_flags** Opcional - Uma combinação das seguintes constantes - MYSQL_CLIENT_SSL - Use criptografia SSL. MYSQL_CLIENT_COMPRESS - Use o protocolo de compressão. MYSQL_CLIENT_IGNORE_SPACE - Permitir espaço após os nomes das funções. MYSQL_CLIENT_INTERACTIVE - Permitir segundos de inatividade do tempo limite interativo antes de fechar a conexão.

Você pode se desconectar do banco de dados MySQL a qualquer momento usando outra função PHP **mysql_close ()** . Esta função usa um único parâmetro, que é uma conexão retornada pela função **mysql_connect ()** .

Sintaxe

```
bool mysql_close ( resource $link_identifier );
```

Se um recurso não for especificado, o último banco de dados aberto será fechado. Esta função retorna true se fechar a conexão com sucesso, caso contrário, retorna false.

Exemplo

Tente o seguinte exemplo para se conectar a um servidor MySQL -

```
<html>
   <head>
      <title>Connecting MySQL Server</title>
   </head>
   <body>
      <?php
         $dbhost = 'localhost:3306';
         $dbuser = 'guest';
         $dbpass = 'guest123';
         $conn = mysql_connect($dbhost, $dbuser, $dbpass);

         if(! $conn ) {
            die('Could not connect: ' . mysql_error());
         }
         echo 'Connected successfully';
         mysql_close($conn);
      ?>
   </body>
</html>
```

MySQL - Criar banco de dados

Criar banco de dados usando mysqladmin

Você precisaria de privilégios especiais para criar ou excluir um banco de dados MySQL. Então, supondo que você tenha acesso ao usuário root, você pode criar qualquer banco de dados usando o binário mysql **mysqladmin** .

Exemplo

Aqui está um exemplo simples para criar um banco de dados chamado **TUTORIALS** -

```
[root@host]# mysqladmin -u root -p create TUTORIALS
Enter password:******
```

Isso criará um banco de dados MySQL chamado TUTORIALS.

Criar um banco de dados usando o script PHP

O PHP usa a função **mysql_query** para criar ou excluir um banco de dados MySQL. Essa função usa dois parâmetros e retorna TRUE em caso de sucesso ou FALSE em falha.

Sintaxe

```
bool mysql_query( sql, connection );
```

Sr. Não.	Parâmetro e Descrição
1	**sql** Obrigatório - consulta SQL para criar ou excluir um banco de dados MySQL

2	**conexão** Opcional - se não for especificado, a última conexão aberta por mysql_connect será usada.

Exemplo

O exemplo a seguir para criar um banco de dados -

```
<html>
   <head>
      <title>Creating MySQL Database</title>
   </head>

   <body>
      <?php
         $dbhost = 'localhost:3036';
         $dbuser = 'root';
         $dbpass = 'rootpassword';
         $conn = mysql_connect($dbhost, $dbuser, $dbpass);

         if(! $conn ) {
            die('Could not connect: ' . mysql_error());
         }
         echo 'Connected successfully<br />';
         $sql = 'CREATE DATABASE TUTORIALS';
         $retval = mysql_query( $sql, $conn );

         if(! $retval ) {
            die('Could not create database: ' . mysql_error());
         }
         echo "Database TUTORIALS created successfully\n";
         mysql_close($conn);
      ?>
   </body>
</html>
```

Drop MySQL Database

Solte um banco de dados usando mysqladmin

Você precisaria de privilégios especiais para criar ou excluir um banco de dados MySQL. Então, supondo que você tenha acesso ao usuário root, você pode criar qualquer banco de dados usando o binário mysql **mysqladmin** .

Tenha cuidado ao excluir qualquer banco de dados, pois você perderá todos os dados disponíveis em seu banco de dados.

Aqui está um exemplo para excluir um banco de dados (TUTORIALS) criado no capítulo anterior -

```
[root@host]# mysqladmin -u root -p drop TUTORIALS
Enter password:******
```

Isso lhe dará um aviso e confirmará se você realmente deseja excluir esse banco de dados ou não.

```
Dropping the database is potentially a very bad thing to do.
Any data stored in the database will be destroyed.

Do you really want to drop the 'TUTORIALS' database [y/N] y
Database "TUTORIALS" dropped
```

Drop Database usando script PHP

O PHP usa a função **mysql_query** para criar ou excluir um banco de dados MySQL. Essa função usa dois parâmetros e retorna TRUE em caso de sucesso ou FALSE em falha.

Sintaxe

```
bool mysql_query( sql, connection );
```

Sr. Não	Parâmetro e Descrição
1	**sql** Obrigatório - consulta SQL para criar ou excluir um banco de dados MySQL
2	**conexão** Opcional - se não for especificado, a última conexão aberta por mysql_connect será usada.

Exemplo

Tente o exemplo a seguir para excluir um banco de dados -

```
<html>
   <head>
      <title>Deleting MySQL Database</title>
   </head>

   <body>
      <?php
         $dbhost = 'localhost:3036';
         $dbuser = 'root';
         $dbpass = 'rootpassword';
         $conn = mysql_connect($dbhost, $dbuser, $dbpass);

         if(! $conn ) {
            die('Could not connect: ' . mysql_error());
         }
         echo 'Connected successfully<br />';
         $sql = 'DROP DATABASE TUTORIALS';
         $retval = mysql_query( $sql, $conn );

         if(! $retval ) {
            die('Could not delete database: ' . mysql_error());
         }
         echo "Database TUTORIALS deleted successfully\n";
```

```
         mysql_close($conn);
      ?>
   </body>
</html>
```

AVISO - Ao excluir um banco de dados usando o script PHP, ele não solicita nenhuma confirmação. Portanto, tenha cuidado ao excluir um banco de dados MySQL.

Selecionando o banco de dados MySQL

Uma vez conectado ao servidor MySQL, é necessário selecionar um banco de dados para trabalhar. Isso ocorre porque pode haver mais de um banco de dados disponível com o servidor MySQL.

Selecionando o banco de dados MySQL no prompt de comando

É muito simples selecionar um banco de dados no prompt mysql>. Você pode usar o comando SQL **uso** para selecionar um banco de dados.

Exemplo

Aqui está um exemplo para selecionar um banco de dados chamado **TUTORIALS** -

```
[root@host]# mysql -u root -p
Enter password:******
mysql> use TUTORIALS;
Database changed
mysql>
```

Agora, você selecionou o banco de dados TUTORIALS e todas as operações subseqüentes serão executadas no banco de dados TUTORIALS.

OBSERVAÇÃO : todos os nomes de bancos de dados, nomes de tabelas, nomes de campos de tabela fazem distinção entre maiúsculas e minúsculas. Então você teria que usar os nomes próprios ao dar qualquer comando SQL.

Selecionando um banco de dados MySQL usando PHP Script

O PHP fornece a função **mysql_select_db** para selecionar um banco de dados. Retorna TRUE no sucesso ou FALSE na falha.

Sintaxe

```
bool mysql_select_db( db_name, connection );
```

Sr. Não.	Parâmetro e Descrição
1	

	nome_bd Requerido - Nome do banco de dados MySQL a ser selecionado
2	**conexão** Opcional - se não for especificado, a última conexão aberta por mysql_connect será usada.

Exemplo

Aqui está um exemplo mostrando como selecionar um banco de dados.

```
<html>
   <head>
      <title>Selecting MySQL Database</title>
   </head>

   <body>
      <?php
         $dbhost = 'localhost:3036';
         $dbuser = 'guest';
         $dbpass = 'guest123';
         $conn = mysql_connect($dbhost, $dbuser, $dbpass);

         if(! $conn ) {
            die('Could not connect: ' . mysql_error());
         }
         echo 'Connected successfully';
         mysql_select_db( 'TUTORIALS' );

         mysql_close($conn);
      ?>
   </body>
</html>
```

MySQL - Tipos de Dados

A definição adequada dos campos em uma tabela é importante para a otimização geral do seu banco de dados. Você deve usar apenas o tipo e o tamanho do campo que realmente precisa usar. Por exemplo, não defina um campo de 10 caracteres de largura, se você sabe que vai usar apenas 2 caracteres. Esses tipos de campos (ou colunas) também são referidos como tipos de dados, após o **tipo de dados que** você armazenará nesses campos.

O MySQL usa muitos tipos de dados diferentes divididos em três categorias -

- Numérico
- Data e hora
- Tipos de Cadeia.

Vamos agora discuti-los em detalhes.

Tipos de dados numéricos

O MySQL usa todos os tipos de dados numéricos ANSI SQL padrão, portanto, se você estiver vindo para o MySQL de um sistema de banco de dados diferente, essas definições parecerão familiares para você.

A lista a seguir mostra os tipos de dados numéricos comuns e suas descrições -

INT - Um inteiro de tamanho normal que pode ser assinado ou não assinado. Se assinado, o intervalo permitido é de -2147483648 a 2147483647. Se não assinado, o intervalo permitido é de 0 a 4294967295. Você pode especificar uma largura de até 11 dígitos.

TINYINT - Um inteiro muito pequeno que pode ser assinado ou não assinado. Se assinado, o intervalo permitido é de -128 a 127. Se não assinado, o intervalo permitido é de 0 a 255. Você pode especificar uma largura de até 4 dígitos.

SMALLINT - Um inteiro pequeno que pode ser assinado ou não assinado. Se assinado, o intervalo permitido é de -32768 a 32767. Se não assinado, o intervalo permitido é de 0 a 65535. Você pode especificar uma largura de até 5 dígitos.

MEDIUMINT - Um inteiro de tamanho médio que pode ser assinado ou não assinado. Se assinado, o intervalo permitido é de -8388608 a 8388607. Se não assinado, o intervalo permitido é de 0 a 16777215. Você pode especificar uma largura de até 9 dígitos.

BIGINT - Um número inteiro grande que pode ser assinado ou não assinado. Se assinado, o intervalo permitido é de -9223372036854775808 a 9223372036854775807. Se não for assinado, o intervalo permitido é de 0 a 18446744073709551615. Você pode especificar uma largura de até 20 dígitos.

FLOAT (M, D) - Um número de ponto flutuante que não pode ser assinado. Você pode definir o comprimento de exibição (M) e o número de decimais (D). Isso não é necessário e será padronizado para 10,2, onde 2 é o número de decimais e 10 é o número total de dígitos (incluindo decimais). Precisão decimal pode ir para 24 lugares para um FLOAT.

DOUBLE (M, D) - Um número de ponto flutuante de precisão dupla que não pode ser assinado. Você pode definir o comprimento de exibição (M) e o número de decimais (D). Isso não é obrigatório e será padronizado para 16,4, onde 4 é o número de decimais. A precisão decimal pode ir para 53 lugares para um DOUBLE. REAL é um sinônimo para DOUBLE.

DECIMAL (M, D) - Um número de ponto flutuante descompactado que não pode ser assinado. Nos decimais descompactados, cada decimal corresponde a um byte. É necessário definir o comprimento de exibição (M) e o número de decimais (D). NUMERIC é um sinônimo para DECIMAL.

Tipos de data e hora

Os tipos de dados de data e hora do MySQL são os seguintes -

DATE - Uma data no formato AAAA-MM-DD, entre 1000-01-01 e 9999-12-31. Por exemplo, 30 de Dezembro th de 1973 seria armazenado como 1973/12/30.

DATETIME - Uma combinação de data e hora no formato AAAA-MM-DD HH: MM: SS, entre 1000-01-01 00:00:00 e 9999-12-31 23:59:59. Por exemplo, 3:30 da tarde de 30 de dezembro th de 1973 seria armazenado como 1973/12/30 15:30:00.

TIMESTAMP - Um timestamp entre meia-noite, 01 de janeiro st, 1970 e em algum momento de 2037. Isto parece o formato DATETIME anterior, só que sem os hífens entre números; 3:30 da tarde de 30 de dezembro th de 1973 seria armazenado como 19731230153000 (YYYYMMDDHHMMSS).

TIME - Armazena a hora em um formato HH: MM: SS.

ANO (M) - Armazena um ano em um formato de 2 ou 4 dígitos. Se o comprimento for especificado como 2 (por exemplo, YEAR (2)), YEAR pode estar entre 1970 e 2069 (70 a 69). Se o comprimento for especificado como 4, YEAR poderá ser 1901 a 2155. O tamanho padrão é 4.

Tipos de String

Embora os tipos numéricos e de data sejam divertidos, a maioria dos dados que você armazenará estará em um formato de string. Esta lista descreve os tipos de dados de string comuns no MySQL.

CHAR (M) - Uma cadeia de comprimento fixo entre 1 e 255 caracteres de comprimento (por exemplo CHAR (5)), preenchida à direita com espaços no comprimento especificado quando armazenada. Definir um comprimento não é obrigatório, mas o padrão é 1.

VARCHAR (M) - Uma cadeia de comprimento variável entre 1 e 255 caracteres de comprimento. Por exemplo, VARCHAR (25). Você deve definir um comprimento ao criar um campo VARCHAR.

BLOB ou TEXTO - Um campo com um comprimento máximo de 65535 caracteres. Os BLOBs são "Binary Large Objects" e são usados para armazenar grandes quantidades de dados binários, como imagens ou outros tipos de arquivos. Campos definidos como TEXT também contêm grandes quantidades de dados. A diferença entre os dois é que as classificações e comparações nos dados armazenados diferenciam **maiúsculas e minúsculas** nos BLOBs e **não**

diferenciam **maiúsculas de minúsculas** nos campos TEXTO. Você não especifica um tamanho com BLOB ou TEXT.

TINYBLOB ou TINYTEXT - Uma coluna BLOB ou TEXT com um tamanho máximo de 255 caracteres. Você não especifica um tamanho com TINYBLOB ou TINYTEXT.

MEDIUMBLOB ou MEDIUMTEXT - Uma coluna BLOB ou TEXT com um tamanho máximo de 16777215 caracteres. Você não especifica um tamanho com MEDIUMBLOB ou MEDIUMTEXT.

LONGBLOB ou LONGTEXT - Uma coluna BLOB ou TEXT com um comprimento máximo de 4294967295 caracteres. Você não especifica um tamanho com LONGBLOB ou LONGTEXT.

ENUM - Uma enumeração, que é um termo chique para lista. Ao definir um ENUM, você está criando uma lista de itens a partir dos quais o valor deve ser selecionado (ou pode ser NULL). Por exemplo, se você quiser que seu campo contenha "A" ou "B" ou "C", você definiria seu ENUM como ENUM ('A', 'B', 'C') e somente esses valores (ou NULL) poderia povoar esse campo.

No próximo capítulo, discutiremos como criar tabelas no MySQL.

Crie tabelas do MySQL

Para começar, o comando de criação de tabela requer os seguintes detalhes -

Nome da tabela

Nome dos campos

Definições para cada campo

Sintaxe

Aqui está uma sintaxe SQL genérica para criar uma tabela MySQL -

```
CREATE TABLE table_name (column_name column_type);
```

Agora, vamos criar a seguinte tabela no banco de dados **TUTORIALS** .

```
create table tutorials_tbl(
   tutorial_id INT NOT NULL AUTO_INCREMENT,
   tutorial_title VARCHAR(100) NOT NULL,
   tutorial_author VARCHAR(40) NOT NULL,
   submission_date DATE,
   PRIMARY KEY ( tutorial_id )
);
```

Aqui, alguns itens precisam de explicação

Atributo de campo **NOT NULL** está sendo usado porque não queremos que este campo seja NULL. Então, se um usuário tentar criar um registro com um valor NULL, o MySQL irá gerar um erro.

Atributo de campo **AUTO_INCREMENT** diz ao MySQL para ir em frente e adicionar o próximo número disponível ao campo id.

A **palavra-** chave **PRIMARY KEY** é usada para definir uma coluna como uma chave primária. Você pode usar várias colunas separadas por uma vírgula para definir uma chave primária.

Criando tabelas no prompt de comando

É fácil criar uma tabela MySQL a partir do prompt mysql>. Você usará o comando SQL **CREATE TABLE** para criar uma tabela.

Exemplo

Aqui está um exemplo, que irá criar **tutorials_tbl** -

```
root@host# mysql -u root -p
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> CREATE TABLE tutorials_tbl(
   -> tutorial_id INT NOT NULL AUTO_INCREMENT,
   -> tutorial_title VARCHAR(100) NOT NULL,
   -> tutorial_author VARCHAR(40) NOT NULL,
   -> submission_date DATE,
   -> PRIMARY KEY ( tutorial_id )
   -> );
Query OK, 0 rows affected (0.16 sec)
mysql>
```

OBSERVAÇÃO - O MySQL não finaliza um comando até que você forneça um ponto-e-vírgula (;) no final do comando SQL.

Criando Tabelas Usando Script PHP

Para criar uma nova tabela em qualquer banco de dados existente, você precisaria usar a função PHP **mysql_query ()** . Você passará seu segundo argumento com um comando SQL apropriado para criar uma tabela.

Exemplo

O programa a seguir é um exemplo para criar uma tabela usando script PHP -

```
<html>
   <head>
      <title>Creating MySQL Tables</title>
   </head>

   <body>
      <?php
         $dbhost = 'localhost:3036';
         $dbuser = 'root';
         $dbpass = 'rootpassword';
         $conn = mysql_connect($dbhost, $dbuser, $dbpass);

         if(! $conn ) {
            die('Could not connect: ' . mysql_error());
```

```
         }
         echo 'Connected successfully<br />';
         $sql = "CREATE TABLE tutorials_tbl( ".
            "tutorial_id INT NOT NULL AUTO_INCREMENT, ".
            "tutorial_title VARCHAR(100) NOT NULL, ".
            "tutorial_author VARCHAR(40) NOT NULL, ".
            "submission_date DATE, ".
            "PRIMARY KEY ( tutorial_id )); ";
         mysql_select_db( 'TUTORIALS' );
         $retval = mysql_query( $sql, $conn );

         if(! $retval ) {
            die('Could not create table: ' . mysql_error());
         }
         echo "Table created successfully\n";
         mysql_close($conn);
      ?>
   </body>
</html>
```

Drop MySQL Tables

É muito fácil descartar uma tabela existente do MySQL, mas você precisa ter muito cuidado ao excluir qualquer tabela existente, pois os dados perdidos não serão recuperados após a exclusão de uma tabela.

Sintaxe

Aqui está uma sintaxe SQL genérica para soltar uma tabela MySQL -

```
DROP TABLE table_name ;
```

Eliminando Tabelas do Prompt de Comando

Para descartar tabelas a partir do prompt de comando, precisamos executar o comando DROP TABLE SQL no prompt mysql>.

Exemplo

O seguinte programa é um exemplo que exclui o **tutorials_tbl** -

```
root@host# mysql -u root -p
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> DROP TABLE tutorials_tbl
Query OK, 0 rows affected (0.8 sec)
mysql>
```

Descartando Tabelas Usando Script PHP

Para eliminar uma tabela existente em qualquer banco de dados, você precisaria usar a função PHP **mysql_query ()** . Você passará seu segundo argumento com um comando SQL apropriado para descartar uma tabela.

Exemplo

```
<html>
   <head>
      <title>Creating MySQL Tables</title>
   </head>

   <body>
      <?php
         $dbhost = 'localhost:3036';
         $dbuser = 'root';
         $dbpass = 'rootpassword';
         $conn = mysql_connect($dbhost, $dbuser, $dbpass);

         if(! $conn ) {
            die('Could not connect: ' . mysql_error());
         }
         echo 'Connected successfully<br />';
         $sql = "DROP TABLE tutorials_tbl";
         mysql_select_db( 'TUTORIALS' );
         $retval = mysql_query( $sql, $conn );

         if(! $retval ) {
            die('Could not delete table: ' . mysql_error());
         }
         echo "Table deleted successfully\n";
         mysql_close($conn);
      ?>
   </body>
</html>
```

MySQL - Inserir Consulta

Para inserir dados em uma tabela MySQL, você precisaria usar o comando SQL **INSERT INTO** . Você pode inserir dados na tabela MySQL usando o prompt mysql> ou usando qualquer script como o PHP.

Sintaxe

Aqui está uma sintaxe SQL genérica do comando INSERT INTO para inserir dados na tabela MySQL -

```
INSERT INTO table_name ( field1, field2,...fieldN )
   VALUES
   ( value1, value2,...valueN );
```

Para inserir tipos de dados de string, é necessário manter todos os valores em aspas duplas ou simples. Por exemplo, **"valor"** .

Inserindo Dados do Prompt de Comando

Para inserir dados a partir do prompt de comando, usaremos o comando SQL INSERT INTO para inserir dados na tabela tutorials_tbl do MySQL.

Exemplo

O exemplo a seguir criará 3 registros na tabela **tutorials_tbl** -

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed

mysql> INSERT INTO tutorials_tbl
   ->(tutorial_title, tutorial_author, submission_date)
   ->VALUES
   ->("Learn PHP", "John Poul", NOW());
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO tutorials_tbl
   ->(tutorial_title, tutorial_author, submission_date)
   ->VALUES
   ->("Learn MySQL", "Abdul S", NOW());
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO tutorials_tbl
   ->(tutorial_title, tutorial_author, submission_date)
   ->VALUES
   ->("JAVA Tutorial", "Sanjay", '2007-05-06');
Query OK, 1 row affected (0.01 sec)
mysql>
```

NOTA - Por favor, note que todos os sinais de seta (->) não fazem parte do comando SQL. Eles estão indicando uma nova linha e eles são criados automaticamente pelo prompt do MySQL enquanto pressiona a tecla enter sem fornecer um ponto e vírgula no final de cada linha do comando.

No exemplo acima, não fornecemos um tutorial_id porque, no momento da criação da tabela, fornecemos AUTO_INCREMENT para esse campo. Então, o MySQL cuida de inserir esses IDs automaticamente. Aqui, **NOW ()** é uma função do MySQL, que retorna a data e hora atuais.

Inserindo Dados Usando um Script PHP

Você pode usar o mesmo comando SQL INSERT INTO na função PHP **mysql_query ()** para inserir dados em uma tabela MySQL.

Exemplo

Este exemplo irá pegar três parâmetros do usuário e irá inseri-los na tabela MySQL -

```
<html>

   <head>
      <title>Add New Record in MySQL Database</title>
   </head>

   <body>
      <?php
         if(isset($_POST['add'])) {
            $dbhost = 'localhost:3036';
            $dbuser = 'root';
            $dbpass = 'rootpassword';
            $conn = mysql_connect($dbhost, $dbuser, $dbpass);

            if(! $conn ) {
               die('Could not connect: ' . mysql_error());
```

```
         }

         if(! get_magic_quotes_gpc() ) {
            $tutorial_title = addslashes ($_POST['tutorial_title']);
            $tutorial_author = addslashes ($_POST['tutorial_author']);
         } else {
            $tutorial_title = $_POST['tutorial_title'];
            $tutorial_author = $_POST['tutorial_author'];
         }

         $submission_date = $_POST['submission_date'];

         $sql = "INSERT INTO tutorials_tbl ".
            "(tutorial_title,tutorial_author, submission_date) "."VALUES ".
            "('$tutorial_title','$tutorial_author','$submission_date')";
            mysql_select_db('TUTORIALS');
         $retval = mysql_query( $sql, $conn );

         if(! $retval ) {
            die('Could not enter data: ' . mysql_error());
         }

         echo "Entered data successfully\n";
         mysql_close($conn);
      } else {
   ?>

   <form method = "post" action = "<?php $_PHP_SELF ?>">
      <table width = "600" border = "0" cellspacing = "1" cellpadding = "2">
         <tr>
            <td width = "250">Tutorial Title</td>
            <td>
               <input name = "tutorial_title" type = "text" id = "tutorial_title">
            </td>
         </tr>

         <tr>
            <td width = "250">Tutorial Author</td>
            <td>
               <input name = "tutorial_author" type = "text" id = "tutorial_author">
            </td>
         </tr>

         <tr>
            <td width = "250">Submission Date [   yyyy-mm-dd ]</td>
            <td>
               <input name = "submission_date" type = "text" id = "submission_date">
            </td>
         </tr>

         <tr>
            <td width = "250"> </td>
            <td> </td>
         </tr>

         <tr>
            <td width = "250"> </td>
            <td>
               <input name = "add" type = "submit" id = "add"  value = "Add Tutorial">
            </td>
         </tr>
      </table>
   </form>
<?php
   }
```

```
   ?>
   </body>
</html>
```

Ao fazer uma inserção de dados, é melhor usar a função **get_magic_quotes_gpc ()** para verificar se a configuração atual da cotação mágica está definida ou não. Se esta função retornar false, use a função **addslashes ()** para adicionar barras antes das aspas.

Você pode colocar várias validações para verificar se os dados inseridos estão corretos ou não e pode executar a ação apropriada.

MySQL - Selecione Consulta

O comando SQL **SELECT** é usado para buscar dados do banco de dados MySQL. Você pode usar este comando no prompt mysql>, bem como em qualquer script como o PHP.

Sintaxe

Aqui está a sintaxe SQL genérica do comando SELECT para buscar dados da tabela MySQL -

```
SELECT field1, field2,...fieldN
FROM table_name1, table_name2...
[WHERE Clause]
[OFFSET M ][LIMIT N]
```

Você pode usar uma ou mais tabelas separadas por vírgula para incluir várias condições usando uma cláusula WHERE, mas a cláusula WHERE é uma parte opcional do comando SELECT.

Você pode buscar um ou mais campos em um único comando SELECT.

Você pode especificar estrela (*) no lugar dos campos. Neste caso, o SELECT retornará todos os campos.

Você pode especificar qualquer condição usando a cláusula WHERE.

Você pode especificar um deslocamento usando **OFFSET** de onde o SELECT começará a retornar os registros. Por padrão, o deslocamento começa em zero.

Você pode limitar o número de devoluções usando o atributo **LIMIT** .

Buscando dados de um prompt de comando

Isso usará o comando SQL SELECT para buscar dados da tabela do MySQL **tutorials_tbl** .

Exemplo

O exemplo a seguir retornará todos os registros da tabela **tutorials_tbl** -

```
root@host# mysql -u root -p password;
Enter password:*******
```

```
mysql> use TUTORIALS;
Database changed
mysql> SELECT * from tutorials_tbl
+-------------+----------------+-----------------+-----------------+
| tutorial_id | tutorial_title | tutorial_author | submission_date |
+-------------+----------------+-----------------+-----------------+
|           1 | Learn PHP      | John Poul       | 2007-05-21      |
|           2 | Learn MySQL    | Abdul S         | 2007-05-21      |
|           3 | JAVA Tutorial  | Sanjay          | 2007-05-21      |
+-------------+----------------+-----------------+-----------------+
3 rows in set (0.01 sec)

mysql>
```

Buscando dados usando um script PHP

Você pode usar o mesmo comando SQL SELECT em uma função PHP **mysql_query ()** . Esta função é usada para executar o comando SQL e, posteriormente, outra função PHP, **mysql_fetch_array (),** pode ser usada para buscar todos os dados selecionados. Essa função retorna a linha como uma matriz associativa, uma matriz numérica ou ambos. Esta função retorna FALSE se não houver mais linhas.

O programa a seguir é um exemplo simples que mostrará como buscar / exibir registros da tabela **tutorials_tbl** .

Exemplo

O bloco de código a seguir exibirá todos os registros da tabela tutorials_tbl.

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);

   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }
   $sql = 'SELECT tutorial_id, tutorial_title, tutorial_author, submission_date FROM tutorials

   mysql_select_db('TUTORIALS');
   $retval = mysql_query( $sql, $conn );

   if(! $retval ) {
      die('Could not get data: ' . mysql_error());
   }

   while($row = mysql_fetch_array($retval, MYSQL_ASSOC)) {
      echo "Tutorial ID :{$row['tutorial_id']}  <br> ".
         "Title: {$row['tutorial_title']} <br> ".
         "Author: {$row['tutorial_author']} <br> ".
         "Submission Date : {$row['submission_date']} <br> ".
         "--------------------------------<br>";
   }
   echo "Fetched data successfully\n";
   mysql_close($conn);
?>
```

O conteúdo das linhas é atribuído à variável $ row e os valores nessa linha são impressos.

OBSERVAÇÃO - Lembre-se sempre de colocar chaves quando quiser inserir um valor de matriz diretamente em uma seqüência de caracteres.

No exemplo acima, a constante **MYSQL_ASSOC** é usada como o segundo argumento para a função PHP **mysql_fetch_array ()** , para que ela retorne a linha como uma matriz associativa. Com um array associativo, você pode acessar o campo usando seu nome, em vez de usar o índice.

O PHP fornece outra função chamada **mysql_fetch_assoc ()** , que também retorna a linha como uma matriz associativa.

Exemplo

O exemplo a seguir mostra todos os registros da tabela tutorial_tbl usando a função mysql_fetch_assoc ().

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);

   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }

   $sql = 'SELECT tutorial_id, tutorial_title, tutorial_author, submission_date
      FROM tutorials_tbl';

   mysql_select_db('TUTORIALS');
   $retval = mysql_query( $sql, $conn );

   if(! $retval ) {
      die('Could not get data: ' . mysql_error());
   }

   while($row = mysql_fetch_assoc($retval)) {
      echo "Tutorial ID :{$row['tutorial_id']}  <br> ".
         "Title: {$row['tutorial_title']} <br> ".
         "Author: {$row['tutorial_author']} <br> ".
         "Submission Date : {$row['submission_date']} <br> ".
         "--------------------------------<br>";
   }
   echo "Fetched data successfully\n";
   mysql_close($conn);
?>
```

Você também pode usar a constante **MYSQL_NUM** como o segundo argumento para a função PHP mysql_fetch_array (). Isso fará com que a função retorne uma matriz com o índice numérico.

Exemplo

Experimente o seguinte exemplo para exibir todos os registros da tabela tutorials_tbl usando o argumento MYSQL_NUM.

```php
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);

   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }

   $sql = 'SELECT tutorial_id, tutorial_title, tutorial_author, submission_date
      FROM tutorials_tbl';

   mysql_select_db('TUTORIALS');
   $retval = mysql_query( $sql, $conn );

   if(! $retval ) {
      die('Could not get data: ' . mysql_error());
   }

   while($row = mysql_fetch_array($retval, MYSQL_NUM)) {
      echo "Tutorial ID :{$row[0]}  <br> ".
         "Title: {$row[1]} <br> ".
         "Author: {$row[2]} <br> ".
         "Submission Date : {$row[3]} <br> ".
         "--------------------------------<br>";
   }
   echo "Fetched data successfully\n";
   mysql_close($conn);
?>
```

Todos os três exemplos acima produzirão o mesmo resultado.

Liberando Memória

É uma boa prática liberar a memória do cursor no final de cada instrução SELECT. Isso pode ser feito usando a função PHP **mysql_free_result ()** . O programa a seguir é o exemplo para mostrar como ele deve ser usado.

Exemplo

Experimente o seguinte exemplo -

```php
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);

   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }

   $sql = 'SELECT tutorial_id, tutorial_title, tutorial_author, submission_date
      FROM tutorials_tbl';

   mysql_select_db('TUTORIALS');
   $retval = mysql_query( $sql, $conn );

   if(! $retval ) {
      die('Could not get data: ' . mysql_error());
```

```
   }

   while($row = mysql_fetch_array($retval, MYSQL_NUM)) {
      echo "Tutorial ID :{$row[0]}  <br> ".
         "Title: {$row[1]} <br> ".
         "Author: {$row[2]} <br> ".
         "Submission Date : {$row[3]} <br> ".
         "--------------------------------<br>";
   }
   mysql_free_result($retval);
   echo "Fetched data successfully\n";
   mysql_close($conn);
?>
```

Ao buscar dados, você pode escrever um código tão complexo quanto quiser, mas o procedimento permanecerá o mesmo mencionado acima.

MySQL - cláusula WHERE

Vimos o comando SQL **SELECT** para buscar dados de uma tabela MySQL. Podemos usar uma cláusula condicional chamada **Cláusula WHERE** para filtrar os resultados. Usando esta cláusula WHERE, podemos especificar um critério de seleção para selecionar os registros necessários de uma tabela.

Sintaxe

O bloco de código a seguir tem uma sintaxe SQL genérica do comando SELECT com a cláusula WHERE para buscar dados da tabela MySQL -

```
SELECT field1, field2,...fieldN table_name1, table_name2...
[WHERE condition1 [AND [OR]] condition2.....
```

Você pode usar uma ou mais tabelas separadas por uma vírgula para incluir várias condições usando uma cláusula WHERE, mas a cláusula WHERE é uma parte opcional do comando SELECT.

Você pode especificar qualquer condição usando a cláusula WHERE.

Você pode especificar mais de uma condição usando os operadores **AND** ou **OR** .

Uma cláusula WHERE pode ser usada junto com o comando DELETE ou UPDATE SQL também para especificar uma condição.

A cláusula **WHERE** funciona como uma **condição if** em qualquer linguagem de programação. Essa cláusula é usada para comparar o valor fornecido com o valor do campo disponível em uma tabela MySQL. Se o valor fornecido de fora for igual ao valor do campo disponível na tabela MySQL, ele retornará essa linha.

Aqui está a lista de operadores, que podem ser usados com a cláusula **WHERE** .

Suponha que o campo A detenha 10 e o campo B detenha 20, então -

Operador	Descrição	Exemplo

=	Verifica se os valores dos dois operandos são iguais ou não, se sim, a condição se torna verdadeira.	(A = B) não é verdade.
! =	Verifica se os valores dos dois operandos são iguais ou não, se os valores não forem iguais, a condição se torna verdadeira.	(A! = B) é verdade.
>	Verifica se o valor do operando esquerdo é maior que o valor do operando direito, se sim, a condição se torna verdadeira.	(A> B) não é verdade.
<	Verifica se o valor do operando esquerdo é menor que o valor do operando direito, se sim, a condição se torna verdadeira.	(A <B) é verdade.
> =	Verifica se o valor do operando esquerdo é maior ou igual ao valor do operando direito, se sim, a condição se torna verdadeira.	(A> = B) não é verdade.
<=	Verifica se o valor do operando esquerdo é menor ou igual ao valor do operando direito, se sim, a condição se torna verdadeira.	(A <= B) é verdade.

A cláusula WHERE é muito útil quando você deseja buscar as linhas selecionadas de uma tabela, especialmente quando você usa o **MySQL Join** . As junções são discutidas em outro capítulo.

É uma prática comum procurar registros usando a **chave primária** para tornar a pesquisa mais rápida.

Se a condição especificada não corresponder a nenhum registro na tabela, a consulta não retornará nenhuma linha.

Buscando dados do prompt de comando

Isso usará o comando SQL SELECT com a cláusula WHERE para buscar os dados selecionados da tabela do MySQL - **tutorials_tbl** .

Exemplo

O exemplo a seguir retornará todos os registros da tabela **tutorials_tbl** para a qual o nome do autor é **Sanjay** .

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed
```

```
mysql> SELECT * from tutorials_tbl WHERE tutorial_author = 'Sanjay';
+-------------+----------------+-----------------+-----------------+
| tutorial_id | tutorial_title | tutorial_author | submission_date |
+-------------+----------------+-----------------+-----------------+
|      3      | JAVA Tutorial  |      Sanjay     |    2007-05-21   |
+-------------+----------------+-----------------+-----------------+
1 rows in set (0.01 sec)

mysql>
```

A menos que você execute uma comparação **LIKE** em uma string, a comparação não faz distinção entre maiúsculas e minúsculas. Você pode tornar seu caso de pesquisa sensível usando a palavra-chave **BINARY da** seguinte forma:

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> SELECT * from tutorials_tbl \
   WHERE BINARY tutorial_author = 'sanjay';
Empty set (0.02 sec)

mysql>
```

Buscando dados usando um script PHP

Você pode usar o mesmo comando SQL SELECT com WHERE CLAUSE na função PHP **mysql_query ()** . Esta função é usada para executar o comando SQL e, posteriormente, outra função PHP, **mysql_fetch_array (),** pode ser usada para buscar todos os dados selecionados. Essa função retorna uma linha como uma matriz associativa, uma matriz numérica ou ambas. Esta função retorna FALSE se não houver mais linhas.

Exemplo

O exemplo a seguir retornará todos os registros da tabela **tutorials_tbl** para a qual o nome do autor é **Sanjay** -

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);

   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }

   $sql = 'SELECT tutorial_id, tutorial_title,
      tutorial_author, submission_date
      FROM tutorials_tbl
      WHERE tutorial_author = "Sanjay"';

   mysql_select_db('TUTORIALS');
   $retval = mysql_query( $sql, $conn );

   if(! $retval ) {
      die('Could not get data: ' . mysql_error());
```

```
   }

   while($row = mysql_fetch_array($retval, MYSQL_ASSOC)) {
      echo "Tutorial ID :{$row['tutorial_id']}  <br> ".
      "Title: {$row['tutorial_title']} <br> ".
      "Author: {$row['tutorial_author']} <br> ".
      "Submission Date : {$row['submission_date']} <br> ".
      "--------------------------------<br>";
   }

   echo "Fetched data successfully\n";
   mysql_close($conn);
?>
```

MySQL - Consulta UPDATE

Pode haver um requisito em que os dados existentes em uma tabela do MySQL precisam ser modificados. Você pode fazer isso usando o comando SQL **UPDATE** . Isto irá modificar qualquer valor de campo de qualquer tabela MySQL.

Sintaxe

O bloco de código a seguir tem uma sintaxe SQL genérica do comando UPDATE para modificar os dados na tabela MySQL -

```
UPDATE table_name SET field1 = new-value1, field2 = new-value2
[WHERE Clause]
```

- Você pode atualizar um ou mais campos completamente.
- Você pode especificar qualquer condição usando a cláusula WHERE.
- Você pode atualizar os valores em uma única tabela por vez.

A cláusula WHERE é muito útil quando você deseja atualizar as linhas selecionadas em uma tabela.

Atualizando dados do prompt de comando

Isso usará o comando SQL UPDATE com a cláusula WHERE para atualizar os dados selecionados na tabela do MySQL **tutorials_tbl** .

Exemplo

O exemplo a seguir atualizará o campo **tutorial_title** para um registro com o tutorial_id como 3.

```
root@host# mysql -u root -p password;
Enter password:*******

mysql> use TUTORIALS;
Database changed

mysql> UPDATE tutorials_tbl
   -> SET tutorial_title = 'Learning JAVA'
   -> WHERE tutorial_id = 3;
Query OK, 1 row affected (0.04 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql>
```

Atualizando Dados Usando um Script PHP

Você pode usar o comando SQL UPDATE com ou sem o WHERE CLAUSE na função PHP - **mysql_query ()** . Esta função irá executar o comando SQL de maneira similar a que é executada no prompt mysql>.

Exemplo

O exemplo a seguir para atualizar o campo **tutorial_title** para um registro com tutorial_id como 3.

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);

   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }

   $sql = 'UPDATE tutorials_tbl
      SET tutorial_title="Learning JAVA"
      WHERE tutorial_id=3';

   mysql_select_db('TUTORIALS');
   $retval = mysql_query( $sql, $conn );

   if(! $retval ) {
      die('Could not update data: ' . mysql_error());
   }
   echo "Updated data successfully\n";
   mysql_close($conn);
?>
```

MySQL - consulta DELETE

Se você quiser excluir um registro de qualquer tabela MySQL, então você pode usar o comando SQL **DELETE FROM** . Você pode usar este comando no prompt mysql>, assim como em qualquer script como o PHP.

Sintaxe

O bloco de código a seguir possui uma sintaxe SQL genérica do comando DELETE para excluir dados de uma tabela MySQL.

```
DELETE FROM table_name [WHERE Clause]
```

Se a cláusula WHERE não for especificada, todos os registros serão excluídos da tabela MySQL especificada.

Você pode especificar qualquer condição usando a cláusula WHERE.

Você pode excluir registros em uma única tabela por vez.

A cláusula WHERE é muito útil quando você deseja excluir linhas selecionadas em uma tabela.

Excluindo Dados do Prompt de Comando

Isso usará o comando SQL DELETE com a cláusula WHERE para excluir dados selecionados na tabela do MySQL - **tutorials_tbl** .

Exemplo

O exemplo a seguir excluirá um registro do tutorial_tbl cujo tutorial_id é 3.

```
root@host# mysql -u root -p password;
Enter password:*******

mysql> use TUTORIALS;
Database changed

mysql> DELETE FROM tutorials_tbl WHERE tutorial_id=3;
Query OK, 1 row affected (0.23 sec)

mysql>
```

Excluindo Dados Usando um Script PHP

Você pode usar o comando SQL DELETE com ou sem o WHERE CLAUSE na função PHP - **mysql_query ()** . Esta função irá executar o comando SQL da mesma forma que é executado no prompt mysql>.

Exemplo

Tente o exemplo a seguir para excluir um registro do tutorial_tbl cujo tutorial_id é 3.

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);

   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }

   $sql = 'DELETE FROM tutorials_tbl WHERE tutorial_id = 3';

   mysql_select_db('TUTORIALS');
   $retval = mysql_query( $sql, $conn );

   if(! $retval ) {
      die('Could not delete data: ' . mysql_error());
   }
   echo "Deleted data successfully\n";
   mysql_close($conn);
?>
```

MySQL - cláusula LIKE

Vimos o comando SQL **SELECT** para buscar dados da tabela MySQL. Também podemos usar uma cláusula condicional chamada de cláusula **WHERE** para selecionar os registros necessários.

Uma cláusula WHERE com o sinal 'igual a' (=) funciona bem onde queremos fazer uma correspondência exata. Como se "tutorial_author = 'Sanjay'". Mas pode haver um requisito em que queremos filtrar todos os resultados em que o nome tutorial_author deve conter "jay". Isso pode ser tratado usando a **cláusula SQL LIKE** juntamente com a cláusula WHERE.

Se a cláusula SQL LIKE for usada junto com o caractere%, ela funcionará como um meta caractere (*) como no UNIX, enquanto lista todos os arquivos ou diretórios no prompt de comando. Sem um caractere%, a cláusula LIKE é muito **igual ao** sinal de **igual,** juntamente com a cláusula WHERE.

Sintaxe

O bloco de código a seguir tem uma sintaxe SQL genérica do comando SELECT junto com a cláusula LIKE para buscar dados de uma tabela MySQL.

```
SELECT field1, field2,...fieldN table_name1, table_name2...
WHERE field1 LIKE condition1 [AND [OR]] filed2 = 'somevalue'
```

Você pode especificar qualquer condição usando a cláusula WHERE.

Você pode usar a cláusula LIKE junto com a cláusula WHERE.

Você pode usar a cláusula LIKE no lugar dos **iguais para** assinar.

Quando o LIKE é usado junto com% sign, ele funcionará como uma pesquisa de meta-caracteres.

Você pode especificar mais de uma condição usando os operadores **AND** ou **OR** .

Uma cláusula WHERE ... LIKE pode ser usada junto com o comando DELETE ou UPDATE SQL também para especificar uma condição.

Usando a cláusula LIKE no prompt de comando

Isso usará o comando SQL SELECT com a cláusula WHERE ... LIKE para buscar os dados selecionados da tabela do MySQL - **tutorials_tbl** .

Exemplo

O exemplo a seguir retornará todos os registros da tabela **tutorials_tbl** para a qual o nome do autor termina com **jay** -

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> SELECT * from tutorials_tbl
```

```
   -> WHERE tutorial_author LIKE '%jay';
+-------------+---------------+-----------------+-----------------+
| tutorial_id | tutorial_title | tutorial_author | submission_date |
+-------------+---------------+-----------------+-----------------+
|      3      |  JAVA Tutorial |     Sanjay      |    2007-05-21   |
+-------------+---------------+-----------------+-----------------+
1 rows in set (0.01 sec)

mysql>
```

Usando a cláusula LIKE dentro do PHP Script

Você pode usar a sintaxe similar da cláusula WHERE ... LIKE na função PHP - **mysql_query ()** . Esta função é usada para executar o comando SQL e depois outra função PHP - **mysql_fetch_array ()** pode ser usado para buscar todos os dados selecionados, se a cláusula WHERE ... LIKE for usada junto com o comando SELECT.

Mas se a cláusula WHERE ... LIKE estiver sendo usada com o comando DELETE ou UPDATE, nenhuma outra chamada de função PHP será necessária.

Exemplo

Experimente o seguinte exemplo para retornar todos os registros da tabela **tutorials_tbl** para a qual o nome do autor contém **jay** -

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);

   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }
   $sql = 'SELECT tutorial_id, tutorial_title,
      tutorial_author, submission_date
      FROM tutorials_tbl
      WHERE tutorial_author LIKE "%jay%"';

   mysql_select_db('TUTORIALS');
   $retval = mysql_query( $sql, $conn );

   if(! $retval ) {
      die('Could not get data: ' . mysql_error());
   }

   while($row = mysql_fetch_array($retval, MYSQL_ASSOC)) {
      echo "Tutorial ID :{$row['tutorial_id']}  <br> ".
         "Title: {$row['tutorial_title']} <br> ".
         "Author: {$row['tutorial_author']} <br> ".
         "Submission Date : {$row['submission_date']} <br> ".
         "--------------------------------<br>";
   }
   echo "Fetched data successfully\n";
   mysql_close($conn);
?>
```

MySQL - Classificando Resultados

Vimos o comando SQL **SELECT** para buscar dados de uma tabela MySQL. Quando você seleciona linhas, o servidor MySQL está livre para devolvê-las em qualquer ordem, a menos que você instrua de outra forma dizendo como ordenar o resultado. Mas, você classifica um conjunto de resultados adicionando uma **cláusula ORDER BY** que nomeia a coluna ou colunas que você deseja classificar.

Sintaxe

O bloco de código a seguir é uma sintaxe SQL genérica do comando SELECT junto com a cláusula ORDER BY para classificar os dados de uma tabela MySQL.

```
SELECT field1, field2,...fieldN table_name1, table_name2...
ORDER BY field1, [field2...] [ASC [DESC]]
```

Você pode classificar o resultado retornado em qualquer campo, se esse campo estiver sendo listado.

Você pode classificar o resultado em mais de um campo.

Você pode usar a palavra-chave ASC ou DESC para obter o resultado em ordem crescente ou decrescente. Por padrão, é a ordem crescente.

Você pode usar a cláusula WHERE ... LIKE da maneira usual para colocar uma condição.

Usando a cláusula ORDER BY no prompt de comando

Isso usará o comando SQL SELECT com a **cláusula ORDER BY** para buscar dados da tabela MySQL - **tutorials_tbl** .

Exemplo

Experimente o seguinte exemplo, que retorna o resultado em uma ordem crescente.

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> SELECT * from tutorials_tbl ORDER BY tutorial_author ASC
+-------------+----------------+-----------------+-----------------+
| tutorial_id | tutorial_title | tutorial_author | submission_date |
+-------------+----------------+-----------------+-----------------+
|      2      |  Learn MySQL   |     Abdul S     |    2007-05-24   |
|      1      |   Learn PHP    |    John Poul    |    2007-05-24   |
|      3      | JAVA Tutorial  |     Sanjay      |    2007-05-06   |
+-------------+----------------+-----------------+-----------------+
3 rows in set (0.42 sec)

mysql>
```

Verifique todos os nomes de autores listados na ordem crescente.

Usando a cláusula ORDER BY dentro de um script PHP

Você pode usar uma sintaxe similar da cláusula ORDER BY na função PHP - **mysql_query ()** . Esta função é usada para executar o comando SQL e, posteriormente, outra função PHP, **mysql_fetch_array (),** pode ser usada para buscar todos os dados selecionados.

Exemplo

Experimente o seguinte exemplo, que retorna o resultado em ordem decrescente dos autores do tutorial.

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);

   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }
   $sql = 'SELECT tutorial_id, tutorial_title,
      tutorial_author, submission_date
      FROM tutorials_tbl
      ORDER BY  tutorial_author DESC';

   mysql_select_db('TUTORIALS');
   $retval = mysql_query( $sql, $conn );

   if(! $retval ) {
      die('Could not get data: ' . mysql_error());
   }

   while($row = mysql_fetch_array($retval, MYSQL_ASSOC)) {
      echo "Tutorial ID :{$row['tutorial_id']}  <br> ".
         "Title: {$row['tutorial_title']} <br> ".
         "Author: {$row['tutorial_author']} <br> ".
         "Submission Date : {$row['submission_date']} <br> ".
         "--------------------------------<br>";
   }
   echo "Fetched data successfully\n";
   mysql_close($conn);
?>
```

Usando o MySQl Joins

Nos capítulos anteriores, estávamos obtendo dados de uma tabela de cada vez. Isso é bom o suficiente para tomadas simples, mas na maioria dos usos reais do MySQL, muitas vezes é necessário obter dados de várias tabelas em uma única consulta.

Você pode usar várias tabelas em sua única consulta SQL. O ato de ingressar no MySQL refere-se a quebrar duas ou mais tabelas em uma única tabela.

Você pode usar JOINS nas instruções SELECT, UPDATE e DELETE para unir as tabelas do MySQL. Veremos também um exemplo do LEFT JOIN, que é diferente do MySQL JOIN simples.

Usando associações no prompt de comando

Suponha que temos duas tabelas **tcount_tbl** e **tutorials_tbl** , em TUTORIAIS. Agora, dê uma olhada nos exemplos abaixo -

Exemplo

Os exemplos a seguir -

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> SELECT * FROM tcount_tbl;
+-----------------+----------------+
| tutorial_author | tutorial_count |
+-----------------+----------------+
|      mahran     |       20       |
|      mahnaz     |      NULL      |
|       Jen       |      NULL      |
|      Gill       |       20       |
|    John Poul    |        1       |
|     Sanjay      |        1       |
+-----------------+----------------+
6 rows in set (0.01 sec)
mysql> SELECT * from tutorials_tbl;
+-------------+----------------+-----------------+-----------------+
| tutorial_id | tutorial_title | tutorial_author | submission_date |
+-------------+----------------+-----------------+-----------------+
|      1      |  Learn PHP     |     John Poul   |    2007-05-24   |
|      2      |  Learn MySQL   |      Abdul S    |    2007-05-24   |
|      3      | JAVA Tutorial  |      Sanjay     |    2007-05-06   |
+-------------+----------------+-----------------+-----------------+
3 rows in set (0.00 sec)
mysql>
```

Agora podemos escrever uma consulta SQL para juntar essas duas tabelas. Esta consulta selecionará todos os autores da tabela **tutorials_tbl** e selecionará o número correspondente de tutoriais a partir do **tcount_tbl** .

```
mysql> SELECT a.tutorial_id, a.tutorial_author, b.tutorial_count
   -> FROM tutorials_tbl a, tcount_tbl b
   -> WHERE a.tutorial_author = b.tutorial_author;
+-------------+-----------------+----------------+
| tutorial_id | tutorial_author | tutorial_count |
+-------------+-----------------+----------------+
|      1      |    John Poul    |        1       |
|      3      |     Sanjay      |        1       |
+-------------+-----------------+----------------+
2 rows in set (0.01 sec)
mysql>
```

Usando Joins em um Script PHP

Você pode usar qualquer uma das consultas SQL mencionadas acima no script PHP. Você só precisa passar a consulta SQL para a função PHP **mysql_query ()** e depois buscar os resultados da maneira usual.

Exemplo

O exemplo a seguir -

```php
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);

   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }

   $sql = 'SELECT a.tutorial_id, a.tutorial_author, b.tutorial_count
      FROM tutorials_tbl a, tcount_tbl b
      WHERE a.tutorial_author = b.tutorial_author';

   mysql_select_db('TUTORIALS');
   $retval = mysql_query( $sql, $conn );

   if(! $retval ) {
      die('Could not get data: ' . mysql_error());
   }

   while($row = mysql_fetch_array($retval, MYSQL_ASSOC)) {
      echo "Author:{$row['tutorial_author']}  <br> ".
         "Count: {$row['tutorial_count']} <br> ".
         "Tutorial ID: {$row['tutorial_id']} <br> ".
         "--------------------------------<br>";
   }
   echo "Fetched data successfully\n";
   mysql_close($conn);
?>
```

MySQL LEFT JOIN

Uma junção esquerda do MySQL é diferente de uma junção simples. Um MySQL LEFT JOIN dá uma consideração extra à tabela que está à esquerda.

Se eu fizer um **LEFT JOIN** , eu obtenho todos os registros que combinam da mesma forma e IN ADDITION recebo um registro extra para cada registro não correspondido na tabela à esquerda da junção: assegurando (no meu exemplo) que todo AUTHOR recebe um menção.

Exemplo

Tente o seguinte exemplo para entender o LEFT JOIN.

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> SELECT a.tutorial_id, a.tutorial_author, b.tutorial_count
   -> FROM tutorials_tbl a LEFT JOIN tcount_tbl b
   -> ON a.tutorial_author = b.tutorial_author;
+-------------+-----------------+----------------+
| tutorial_id | tutorial_author | tutorial_count |
+-------------+-----------------+----------------+
|      1      |    John Poul    |       1        |
|      2      |     Abdul S     |      NULL      |
|      3      |     Sanjay      |       1        |
+-------------+-----------------+----------------+
3 rows in set (0.02 sec)
```

Você precisaria fazer mais prática para se familiarizar com JOINS. Este é um conceito um pouco complexo no MySQL / SQL e se tornará mais claro ao fazer exemplos reais.

Manipulando Valores NULL do MySQL

Nós vimos o comando SQL **SELECT** junto com a cláusula **WHERE** para buscar dados de uma tabela MySQL, mas quando tentamos dar uma condição, que compara o campo ou o valor da coluna para **NULL** , ele não funciona corretamente.

Para lidar com tal situação, o MySQL fornece três operadores -

IS NULL - Este operador retorna true, se o valor da coluna for NULL.

IS NOT NULL - Este operador retorna true, se o valor da coluna não for NULL.

<=> - Este operador compara valores, que (diferente do operador =) são verdadeiros mesmo para dois valores NULL.

As condições envolvendo NULL são especiais. Você não pode usar = **NULL** ou! = **NULL** para procurar valores NULL em colunas. Tais comparações sempre falham porque é impossível dizer se elas são verdadeiras ou não. Às vezes, até NULL = NULL falha.

Para procurar colunas que são ou não são NULL, use **IS NULL** ou **IS NOT NULL** .

Usando valores NULL no prompt de comando

Suponha que haja uma tabela chamada **tcount_tbl** no banco de dados TUTORIALS e contenha duas colunas denominadas **tutorial_author** e **tutorial_count** , em que um NULL tutorial_count indica que o valor é desconhecido.

Exemplo

Tente os exemplos a seguir -

```
root@host# mysql -u root -p password;
Enter password:*******

mysql> use TUTORIALS;
Database changed

mysql> create table tcount_tbl
   -> (
   -> tutorial_author varchar(40) NOT NULL,
   -> tutorial_count  INT
   -> );
Query OK, 0 rows affected (0.05 sec)

mysql> INSERT INTO tcount_tbl
   -> (tutorial_author, tutorial_count) values ('mahran', 20);

mysql> INSERT INTO tcount_tbl
   -> (tutorial_author, tutorial_count) values ('mahnaz', NULL);

mysql> INSERT INTO tcount_tbl
   -> (tutorial_author, tutorial_count) values ('Jen', NULL);

mysql> INSERT INTO tcount_tbl
```

```
   -> (tutorial_author, tutorial_count) values ('Gill', 20);

mysql> SELECT * from tcount_tbl;
+-----------------+----------------+
| tutorial_author | tutorial_count |
+-----------------+----------------+
|     mahran      |       20       |
|     mahnaz      |      NULL      |
|      Jen        |      NULL      |
|     Gill        |       20       |
+-----------------+----------------+
4 rows in set (0.00 sec)

mysql>
```

Você pode ver que = e **! =** Não funcionam com valores NULL como segue -

```
mysql> SELECT * FROM tcount_tbl WHERE tutorial_count = NULL;
Empty set (0.00 sec)

mysql> SELECT * FROM tcount_tbl WHERE tutorial_count != NULL;
Empty set (0.01 sec)
```

Para localizar os registros em que a coluna tutorial_count é ou não é NULL, as consultas devem ser gravadas conforme mostrado no programa a seguir.

```
mysql> SELECT * FROM tcount_tbl
   -> WHERE tutorial_count IS NULL;
+-----------------+----------------+
| tutorial_author | tutorial_count |
+-----------------+----------------+
|     mahnaz      |      NULL      |
|      Jen        |      NULL      |
+-----------------+----------------+
2 rows in set (0.00 sec)
mysql> SELECT * from tcount_tbl
   -> WHERE tutorial_count IS NOT NULL;
+-----------------+----------------+
| tutorial_author | tutorial_count |
+-----------------+----------------+
|     mahran      |       20       |
|     Gill        |       20       |
+-----------------+----------------+
2 rows in set (0.00 sec)
```

Manipulando Valores NULL em um Script PHP

Você pode usar a condição **if ... else** para preparar uma consulta com base no valor NULL.

Exemplo

O exemplo a seguir pega o **tutorial_count** de fora e compara-o com o valor disponível na tabela.

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);
```

```
   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }

   if( isset($tutorial_count )) {
      $sql = 'SELECT tutorial_author, tutorial_count
         FROM  tcount_tbl
         WHERE tutorial_count = $tutorial_count';
   } else {
      $sql = 'SELECT tutorial_author, tutorial_count
         FROM  tcount_tbl
         WHERE tutorial_count IS $tutorial_count';
   }

   mysql_select_db('TUTORIALS');
   $retval = mysql_query( $sql, $conn );

   if(! $retval ) {
      die('Could not get data: ' . mysql_error());
   }

   while($row = mysql_fetch_array($retval, MYSQL_ASSOC)) {
      echo "Author:{$row['tutorial_author']}  <br> ".
         "Count: {$row['tutorial_count']} <br> ".
         "--------------------------------<br>";
   }
   echo "Fetched data successfully\n";
   mysql_close($conn);
?>
```

MySQL - Regexps

Você viu o padrão do MySQL combinando com **LIKE ...%** . O MySQL suporta outro tipo de operação de correspondência de padrões com base nas expressões regulares e no operador **REGEXP** . Se você está ciente de PHP ou PERL, então é muito simples para você entender, porque essa correspondência é a mesma daqueles scripts de expressões regulares.

A seguir está a tabela de padrões, que pode ser usada junto com o operador **REGEXP** .

padronizar	O que o padrão corresponde
^	Início da corda
$	Fim da corda
.	Qualquer caractere único
[...]	Qualquer caractere listado entre os colchetes
[^ ...]	Qualquer caractere não listado entre os colchetes
p1 | p2 | p3	Alternação; corresponde a qualquer um dos padrões p1, p2 ou p3
*	Zero ou mais instâncias do elemento precedente

+	Uma ou mais instâncias do elemento precedente
{n}	n instâncias do elemento precedente
{m, n}	m a n instâncias do elemento precedente

Exemplos

Agora, com base na tabela acima, você pode configurar vários tipos de consultas SQL para atender aos seus requisitos. Aqui, estou listando alguns para sua compreensão.

Considere que temos uma tabela chamada **person_tbl** e está tendo um campo chamado **name** -

Consulta para encontrar todos os nomes que começam com **'st'** -

```
mysql> SELECT name FROM person_tbl WHERE name REGEXP '^st';
```

Consulta para encontrar todos os nomes terminados com **'ok'** -

```
mysql> SELECT name FROM person_tbl WHERE name REGEXP 'ok$';
```

Consulta para encontrar todos os nomes que contêm **'mar'** -

```
mysql> SELECT name FROM person_tbl WHERE name REGEXP 'mar';
```

Consulta para encontrar todos os nomes que começam com uma vogal e terminam com **"ok"** -

```
mysql> SELECT name FROM person_tbl WHERE name REGEXP '^[aeiou]|ok$';
```

MySQL - Transações

Uma transação é um grupo seqüencial de operações de manipulação de banco de dados, que é executado como se fosse uma única unidade de trabalho. Em outras palavras, uma transação nunca será concluída, a menos que cada operação individual dentro do grupo seja bem-sucedida. Se qualquer operação dentro da transação falhar, a transação inteira falhará.

Na prática, você agrupará muitas consultas SQL em um grupo e executará todas elas juntas como parte de uma transação.

Propriedades das Transações

As transações têm as seguintes quatro propriedades padrão, geralmente referidas pela sigla **ACID** -

> **Atomicidade** - Isso garante que todas as operações dentro da unidade de trabalho sejam concluídas com sucesso; caso contrário, a transação será interrompida no ponto de falha e as operações anteriores serão revertidas para o estado anterior.

Consistência - Isso garante que o banco de dados altere corretamente os estados em uma transação confirmada com êxito.

Isolamento - Permite que as transações operem de forma independente e transparente entre si.

Durabilidade - Isso garante que o resultado ou o efeito de uma transação confirmada persista em caso de falha do sistema.

No MySQL, as transações começam com a instrução **BEGIN WORK** e terminam com uma **instrução COMMIT** ou **ROLLBACK** . Os comandos SQL entre as instruções inicial e final formam a maior parte da transação.

COMMIT e ROLLBACK

Essas duas palavras-chave **Commit** e **Rollback** são usadas principalmente para transações do MySQL.

Quando uma transação bem-sucedida é concluída, o comando COMMIT deve ser emitido para que as alterações em todas as tabelas envolvidas entrem em vigor.

Se ocorrer uma falha, um comando ROLLBACK deve ser emitido para retornar todas as tabelas referenciadas na transação para seu estado anterior.

Você pode controlar o comportamento de uma transação definindo a variável de sessão chamada **AUTOCOMMIT** . Se AUTOCOMMIT for definido como 1 (o padrão), cada instrução SQL (dentro de uma transação ou não) será considerada uma transação completa e confirmada por padrão quando for concluída.

Quando AUTOCOMMIT é definido como 0, emitindo o comando **SET AUTOCOMMIT = 0** , a série subsequente de instruções age como uma transação e nenhuma atividade é confirmada até que uma instrução COMMIT explícita seja emitida.

Você pode executar esses comandos SQL no PHP usando a função **mysql_query ()** .

Um exemplo genérico na transação

Essa seqüência de eventos é independente da linguagem de programação usada. O caminho lógico pode ser criado no idioma que você usa para criar seu aplicativo.

Você pode executar esses comandos SQL no PHP usando a função **mysql_query ()** .

Inicie a transação emitindo o comando SQL **BEGIN WORK** .

Emita um ou mais comandos SQL como SELECT, INSERT, UPDATE ou DELETE.

Verifique se não há erro e tudo está de acordo com sua necessidade.

Se houver algum erro, emita um comando ROLLBACK; caso contrário, emita um comando COMMIT.

Tipos de tabela seguros para transações no MySQL

Você não pode usar transações diretamente, mas para certas exceções você pode. No entanto, eles não são seguros e garantidos. Se você planeja usar transações em sua programação MySQL, então você precisa criar suas tabelas de uma maneira especial. Existem muitos tipos de tabelas, que suportam transações, mas o mais popular é o **InnoDB** .

O suporte para tabelas InnoDB requer um parâmetro de compilação específico ao compilar o MySQL a partir da fonte. Se sua versão do MySQL não possui suporte a InnoDB, peça ao seu Provedor de Serviços de Internet para construir uma versão do MySQL com suporte para tipos de tabela InnoDB ou baixe e instale a **Distribuição Binária** do **MySQL-Max** para Windows ou Linux / UNIX e trabalhe com o tipo de tabela um ambiente de desenvolvimento.

Se sua instalação do MySQL suportar tabelas InnoDB, simplesmente adicione uma definição **TYPE = InnoDB** à instrução de criação de tabela.

Por exemplo, o código a seguir cria uma tabela InnoDB chamada **tcount_tbl** -

```
root@host# mysql -u root -p password;
Enter password:*******

mysql> use TUTORIALS;
Database changed

mysql> create table tcount_tbl
   -> (
   -> tutorial_author varchar(40) NOT NULL,
   -> tutorial_count  INT
   -> ) TYPE = InnoDB;
Query OK, 0 rows affected (0.05 sec)
```

Para mais detalhes sobre o InnoDB, você pode clicar no seguinte link - InnoDB

Você pode usar outros tipos de tabelas como **GEMINI** ou **BDB** , mas isso depende da sua instalação, se ele suporta ou não esses dois tipos de tabelas.

MySQL - Comando ALTER

O comando **ALTER do** MySQL é muito útil quando você deseja alterar um nome de sua tabela, qualquer campo de tabela ou se você deseja adicionar ou excluir uma coluna existente em uma tabela.

Vamos começar com a criação de uma tabela chamada **testalter_tbl** .

```
root@host# mysql -u root -p password;
Enter password:*******

mysql> use TUTORIALS;
Database changed

mysql> create table testalter_tbl
   -> (
   -> i INT,
   -> c CHAR(1)
   -> );
Query OK, 0 rows affected (0.05 sec)
```

```
mysql> SHOW COLUMNS FROM testalter_tbl;
+-------+---------+------+-----+---------+-------+
| Field |  Type   | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
|   i   | int(11) | YES  |     |   NULL  |       |
|   c   | char(1) | YES  |     |   NULL  |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Descartando, Adicionando ou Reposicionando uma Coluna

Se você quiser descartar uma coluna existente i da tabela MySQL acima, você usará a cláusula **DROP** junto com o comando **ALTER,** conforme mostrado abaixo -

```
mysql> ALTER TABLE testalter_tbl  DROP i;
```

Uma cláusula **DROP** não funcionará se a coluna for a única que resta na tabela.

Para adicionar uma coluna, use ADD e especifique a definição da coluna. A instrução a seguir restaura a coluna **i** para o testalter_tbl -

```
mysql> ALTER TABLE testalter_tbl ADD i INT;
```

Após a emissão desta declaração, o testalter conterá as mesmas duas colunas que você tinha quando criou a tabela, mas não terá a mesma estrutura. Isso ocorre porque há novas colunas adicionadas ao final da tabela por padrão. Então mesmo que originalmente **eu** tenha sido a primeira coluna no mytbl, agora é a última.

```
mysql> SHOW COLUMNS FROM testalter_tbl;
+-------+---------+------+-----+---------+-------+
| Field |  Type   | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
|   c   | char(1) | YES  |     |   NULL  |       |
|   i   | int(11) | YES  |     |   NULL  |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Para indicar que você deseja uma coluna em uma posição específica dentro da tabela, use FIRST para torná-la a primeira coluna ou **AFTER col_name** para indicar que a nova coluna deve ser colocada após o nome de coluna.

Tente as seguintes instruções **ALTER TABLE** , usando **SHOW COLUMNS** depois de cada uma para ver o efeito que cada uma tem -

```
ALTER TABLE testalter_tbl DROP i;
ALTER TABLE testalter_tbl ADD i INT FIRST;
ALTER TABLE testalter_tbl DROP i;
ALTER TABLE testalter_tbl ADD i INT AFTER c;
```

Os especificadores FIRST e AFTER funcionam apenas com a cláusula ADD. Isso significa que, se você quiser reposicionar uma coluna existente em uma tabela, primeiro você

DEVE DESCARREGAR e **ADICIONAR** na nova posição.

Alterando (Alterando) uma Definição de Coluna ou um Nome

Para alterar a definição de uma coluna, use a cláusula **MODIFY** ou **CHANGE** juntamente com o comando ALTER.

Por exemplo, para alterar a coluna **c** de CHAR (1) para CHAR (10), você pode usar o seguinte comando -

```
mysql> ALTER TABLE testalter_tbl MODIFY c CHAR(10);
```

Com **CHANGE** , a sintaxe é um pouco diferente. Após a palavra CHANGE, você nomeia a coluna que deseja alterar e, em seguida, especifica a nova definição, que inclui o novo nome.

Experimente o seguinte exemplo -

```
mysql> ALTER TABLE testalter_tbl CHANGE i j BIGINT;
```

Se você agora usa CHANGE para converter **j** de **BIGINT de** volta para **INT** sem alterar o nome da coluna, a declaração será mostrada abaixo -

```
mysql> ALTER TABLE testalter_tbl CHANGE j j INT;
```

O efeito de ALTER TABLE em atributos de valor nulo e padrão - Quando você modifica ou altera uma coluna, também é possível especificar se a coluna pode ou não conter valores NULL e qual é o valor padrão. Na verdade, se você não fizer isso, o MySQL atribuirá automaticamente valores para esses atributos.

O bloco de código a seguir é um exemplo, em que a coluna **NOT NULL** terá o valor 100 por padrão.

```
mysql> ALTER TABLE testalter_tbl
   -> MODIFY j BIGINT NOT NULL DEFAULT 100;
```

Se você não usar o comando acima, o MySQL irá preencher valores NULL em todas as colunas.

Alterando (Alterando) o Valor Padrão de uma Coluna

Você pode alterar um valor padrão para qualquer coluna usando o comando **ALTER** .

Experimente o seguinte exemplo.

```
mysql> ALTER TABLE testalter_tbl ALTER i SET DEFAULT 1000;
mysql> SHOW COLUMNS FROM testalter_tbl;
+-------+---------+------+-----+---------+-------+
| Field |  Type   | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
|   c   | char(1) | YES  |     |   NULL  |       |
```

```
|   i   | int(11) | YES  |     |   1000  |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Você pode remover a restrição padrão de qualquer coluna usando a cláusula DROP junto com o comando **ALTER** .

```
mysql> ALTER TABLE testalter_tbl ALTER i DROP DEFAULT;
mysql> SHOW COLUMNS FROM testalter_tbl;
+-------+---------+------+-----+---------+-------+
| Field |  Type   | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
|   c   | char(1) | YES  |     |   NULL  |       |
|   i   | int(11) | YES  |     |   NULL  |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Alterando (Alterando) um Tipo de Tabela

Você pode usar um tipo de tabela usando a cláusula **TYPE** junto com o comando ALTER. Experimente o seguinte exemplo para alterar o **testalter_tbl** para o tipo de tabela **MYISAM** .

Para descobrir o tipo atual de uma tabela, use a instrução **SHOW TABLE STATUS** .

```
mysql> ALTER TABLE testalter_tbl TYPE = MYISAM;
mysql>  SHOW TABLE STATUS LIKE 'testalter_tbl'\G
*************************** 1. row ***************
           Name: testalter_tbl
           Type: MyISAM
     Row_format: Fixed
           Rows: 0
 Avg_row_length: 0
    Data_length: 0
Max_data_length: 25769803775
   Index_length: 1024
      Data_free: 0
 Auto_increment: NULL
    Create_time: 2007-06-03 08:04:36
    Update_time: 2007-06-03 08:04:36
     Check_time: NULL
 Create_options:
        Comment:
1 row in set (0.00 sec)
```

Renomeando (alterando) uma tabela

Para renomear uma tabela, use a opção **RENAME** da instrução **ALTER TABLE** .

Experimente o seguinte exemplo para renomear **testalter_tbl** para **alter_tbl** .

```
mysql> ALTER TABLE testalter_tbl RENAME TO alter_tbl;
```

Você pode usar o comando ALTER para criar e soltar o comando INDEX em um arquivo MySQL. Discutiremos detalhadamente sobre esse comando no próximo capítulo.

MySQL - INDEXES

Um índice de banco de dados é uma estrutura de dados que melhora a velocidade das operações em uma tabela. Os índices podem ser criados usando uma ou mais colunas, fornecendo a base para pesquisas aleatórias rápidas e ordenação eficiente de acesso a registros.

Ao criar o índice, deve-se levar em consideração que todas as colunas serão usadas para fazer consultas SQL e criar um ou mais índices nessas colunas.

Praticamente, os índices também são um tipo de tabelas, que mantêm a chave primária ou o campo de índice e um ponteiro para cada registro na tabela real.

Os usuários não podem ver os índices, eles são usados apenas para acelerar as consultas e serão usados pelo Mecanismo de pesquisa de banco de dados para localizar registros muito rapidamente.

As instruções INSERT e UPDATE levam mais tempo em tabelas com índices, enquanto as instruções SELECT se tornam rápidas nessas tabelas. O motivo é que, ao inserir ou atualizar, um banco de dados também precisa inserir ou atualizar os valores do índice.

Índice simples e exclusivo

Você pode criar um índice exclusivo em uma tabela. Um índice exclusivo significa que duas linhas não podem ter o mesmo valor de índice. Aqui está a sintaxe para criar um índice em uma tabela.

CREATE UNIQUE INDEX **index_name** .

ON table_name (coluna1, coluna2, ...);

Você pode usar uma ou mais colunas para criar um índice.

Por exemplo, podemos criar um índice em **tutorials_tbl** usando **tutorial_author** .

CRIE UM ÍNDICE ÚNICO AUTHOR_INDEX

ON tutorials_tbl (tutorial_author)

Você pode criar um índice simples em uma tabela. Apenas omita a palavra-chave **UNIQUE** da consulta para criar um índice simples. Um índice simples permite valores duplicados em uma tabela.

Se você quiser indexar os valores em uma coluna em ordem decrescente, você pode adicionar a palavra reservada DESC após o nome da coluna.

```
mysql> CREATE UNIQUE INDEX AUTHOR_INDEX
ON tutorials_tbl (tutorial_author DESC)
```

Comando ALTER para adicionar e soltar o índice

Existem quatro tipos de instruções para adicionar índices a uma tabela -

ALTER TABLE nome_tabela ADD PRIMARY KEY (column_list) - Esta instrução adiciona uma **PRIMARY KEY** , o que significa que os valores indexados devem ser exclusivos e não podem ser NULL.

ALTER TABLE nome_tabela ADD UNIQUE index_name (column_list) - Esta instrução cria um índice para o qual os valores devem ser exclusivos (exceto para os valores NULL, que podem aparecer várias vezes).

ALTER TABLE nome_tabela ADD INDEX index_name (column_list) - Adiciona um índice comum no qual qualquer valor pode aparecer mais de uma vez.

ALTER TABLE nome_tabela ADD FULLTEXT nome_do_índice (lista_coluna) - Isso cria um índice FULLTEXT especial usado para fins de pesquisa de texto.

O bloco de código a seguir é um exemplo para adicionar índice em uma tabela existente.

```
mysql> ALTER TABLE testalter_tbl ADD INDEX (c);
```

Você pode descartar qualquer INDEX usando a cláusula **DROP** junto com o comando ALTER.

Experimente o seguinte exemplo para eliminar o índice acima criado.

```
mysql> ALTER TABLE testalter_tbl DROP INDEX (c);
```

Você pode descartar qualquer INDEX usando a cláusula DROP junto com o comando ALTER.

Comando ALTER para adicionar e soltar a PRIMARY KEY

Você também pode adicionar uma chave primária da mesma maneira. Mas certifique-se de que a chave primária funcione em colunas, que são NOT NULL.

O bloco de código a seguir é um exemplo para adicionar a chave primária em uma tabela existente. Isso fará com que uma coluna NOT NULL primeiro e, em seguida, adicione-a como uma chave primária.

```
mysql> ALTER TABLE testalter_tbl MODIFY i INT NOT NULL;
mysql> ALTER TABLE testalter_tbl ADD PRIMARY KEY (i);
```

Você pode usar o comando ALTER para soltar uma chave primária da seguinte maneira -

```
mysql> ALTER TABLE testalter_tbl DROP PRIMARY KEY;
```

Para eliminar um índice que não seja PRIMARY KEY, você deve especificar o nome do índice.

Exibindo Informações do ÍNDICE

Você pode usar o comando **SHOW INDEX** para listar todos os índices associados a uma tabela. A saída de formato vertical (especificada por \ G) geralmente é útil com essa instrução, para evitar uma longa linha de contorno -

Experimente o seguinte exemplo -

```
mysql> SHOW INDEX FROM table_name\G
........
```

MySQL - Tabelas Temporárias

As tabelas temporárias podem ser muito úteis em alguns casos para manter dados temporários. A coisa mais importante que deve ser conhecida para tabelas temporárias é que elas serão excluídas quando a sessão atual do cliente terminar.

Quais são as tabelas temporárias?

Tabelas temporárias foram adicionadas na versão 3.23 do MySQL. Se você usa uma versão mais antiga do MySQL que a 3.23, você não pode usar as tabelas temporárias, mas você pode usar as **Tabelas Heap** .

Como dito anteriormente, as tabelas temporárias durarão apenas enquanto a sessão estiver ativa. Se você executar o código em um script PHP, a tabela temporária será destruída automaticamente quando o script terminar a execução. Se você estiver conectado ao servidor de banco de dados MySQL através do programa cliente MySQL, a tabela temporária existirá até você fechar o cliente ou destruir manualmente a tabela.

Exemplo

O programa a seguir é um exemplo que mostra o uso da tabela temporária. O mesmo código pode ser usado em scripts PHP usando a função **mysql_query ()** .

```
mysql> CREATE TEMPORARY TABLE SalesSummary (
   -> product_name VARCHAR(50) NOT NULL
   -> , total_sales DECIMAL(12,2) NOT NULL DEFAULT 0.00
   -> , avg_unit_price DECIMAL(7,2) NOT NULL DEFAULT 0.00
   -> , total_units_sold INT UNSIGNED NOT NULL DEFAULT 0
);
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO SalesSummary
   -> (product_name, total_sales, avg_unit_price, total_units_sold)
   -> VALUES
   -> ('cucumber', 100.25, 90, 2);
```

```
mysql> SELECT * FROM SalesSummary;
+--------------+-------------+----------------+------------------+
| product_name | total_sales | avg_unit_price | total_units_sold |
+--------------+-------------+----------------+------------------+
|   cucumber   |   100.25    |     90.00      |         2        |
+--------------+-------------+----------------+------------------+
1 row in set (0.00 sec)
```

Quando você emite um comando **SHOW TABLES** , sua tabela temporária não será listada na lista. Agora, se você fizer logout da sessão do MySQL e, em seguida, emitir um comando **SELECT** , não encontrará dados disponíveis no banco de dados. Mesmo sua tabela temporária não existirá.

Soltando Tabelas Temporárias

Por padrão, todas as tabelas temporárias são deletadas pelo MySQL quando sua conexão com o banco de dados é terminada. Ainda assim, se você quiser excluí-los no meio, faça isso emitindo o comando **DROP TABLE** .

O programa a seguir é um exemplo sobre a eliminação de uma tabela temporária -

```
mysql> CREATE TEMPORARY TABLE SalesSummary (
   -> product_name VARCHAR(50) NOT NULL
   -> , total_sales DECIMAL(12,2) NOT NULL DEFAULT 0.00
   -> , avg_unit_price DECIMAL(7,2) NOT NULL DEFAULT 0.00
   -> , total_units_sold INT UNSIGNED NOT NULL DEFAULT 0
);
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO SalesSummary
   -> (product_name, total_sales, avg_unit_price, total_units_sold)
   -> VALUES
   -> ('cucumber', 100.25, 90, 2);

mysql> SELECT * FROM SalesSummary;
+--------------+-------------+----------------+------------------+
| product_name | total_sales | avg_unit_price | total_units_sold |
+--------------+-------------+----------------+------------------+
|   cucumber   |   100.25    |     90.00      |         2        |
+--------------+-------------+----------------+------------------+
1 row in set (0.00 sec)
mysql> DROP TABLE SalesSummary;
mysql>  SELECT * FROM SalesSummary;
ERROR 1146: Table 'TUTORIALS.SalesSummary' doesn't exist
```

MySQL - Tabelas Clônicas

Pode haver uma situação em que você precisa de uma cópia exata de uma tabela e **CREATE TABLE ... SELECT** não atende às suas finalidades porque a cópia deve incluir os mesmos índices, valores padrão e assim por diante.

Você pode lidar com essa situação seguindo as etapas abaixo -

Use SHOW CREATE TABLE para obter uma instrução CREATE TABLE que especifique a estrutura da tabela de origem, índices e todos.

Modifique a instrução para alterar o nome da tabela para a tabela clone e execute a instrução. Dessa forma, você terá a tabela de clones exata.

Opcionalmente, se você precisar copiar o conteúdo da tabela, emita também uma instrução INSERT INTO ... SELECT.

Exemplo

Experimente o exemplo a seguir para criar uma tabela de clones para **tutorials_tbl** .

Passo 1 - Obtenha a estrutura completa sobre a mesa.

```
mysql> SHOW CREATE TABLE tutorials_tbl \G;
*************************** 1. row ***************************
       Table: tutorials_tbl
Create Table: CREATE TABLE `tutorials_tbl` (
   `tutorial_id` int(11) NOT NULL auto_increment,
   `tutorial_title` varchar(100) NOT NULL default '',
   `tutorial_author` varchar(40) NOT NULL default '',
   `submission_date` date default NULL,
   PRIMARY KEY  (`tutorial_id`),
   UNIQUE KEY `AUTHOR_INDEX` (`tutorial_author`)
) TYPE = MyISAM
1 row in set (0.00 sec)

ERROR:
No query specified
```

Passo 2 - Renomeie esta tabela e crie outra tabela.

```
mysql> CREATE TABLE clone_tbl (
   -> tutorial_id int(11) NOT NULL auto_increment,
   -> tutorial_title varchar(100) NOT NULL default '',
   -> tutorial_author varchar(40) NOT NULL default '',
   -> submission_date date default NULL,
   -> PRIMARY KEY  (tutorial_id),
   -> UNIQUE KEY AUTHOR_INDEX (tutorial_author)
-> ) TYPE = MyISAM;
Query OK, 0 rows affected (1.80 sec)
```

Passo 3 - Depois de executar o passo 2, você criará uma tabela de clones no seu banco de dados. Se você deseja copiar dados da tabela antiga, você pode fazê-lo usando a instrução INSERT INTO ... SELECT.

```
mysql> INSERT INTO clone_tbl (tutorial_id,
   -> tutorial_title,
   -> tutorial_author,
   -> submission_date)

   -> SELECT tutorial_id,tutorial_title,
   -> tutorial_author,submission_date
   -> FROM tutorials_tbl;
Query OK, 3 rows affected (0.07 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

Finalmente, você terá uma tabela de clones exata como você queria ter.

MySQL - Informações do banco de dados

Obtendo e usando metadados do MySQL

Existem três tipos de informação que você gostaria de ter do MySQL.

> **Informações sobre o resultado de consultas** - isso inclui o número de registros afetados por qualquer instrução SELECT, UPDATE ou DELETE.
>
> **Informações sobre as tabelas e bancos de dados** - Isso inclui informações relativas à estrutura das tabelas e dos bancos de dados.
>
> **Informações sobre o servidor MySQL** - Isso inclui o status do servidor de banco de dados, o número da versão, etc.

É muito fácil obter todas essas informações no prompt do MySQL, mas ao usar PERL ou PHP APIs, precisamos chamar várias APIs explicitamente para obter todas essas informações.

Obtendo o número de linhas afetadas por uma consulta

Vamos ver agora como obter essa informação.

Exemplo de PERL

Em scripts DBI, a contagem de linhas afetadas é retornada pelo comando **do ()** ou pelo comando **execute ()** , dependendo de como você executa a consulta.

```
# Method 1
# execute $query using do( )
my $count = $dbh->do ($query);
# report 0 rows if an error occurred
printf "%d rows were affected\n", (defined ($count) ? $count : 0);

# Method 2
# execute query using prepare( ) plus execute( )
my $sth = $dbh->prepare ($query);
my $count = $sth->execute ( );
printf "%d rows were affected\n", (defined ($count) ? $count : 0);
```

Exemplo de PHP

No PHP, invoque a função **mysql_affected_rows ()** para descobrir quantas linhas uma consulta mudou.

```
$result_id = mysql_query ($query, $conn_id);
# report 0 rows if the query failed
$count = ($result_id ? mysql_affected_rows ($conn_id) : 0);
print ("$count rows were affected\n");
```

Listando Tabelas e Bancos de Dados

É muito fácil listar todos os bancos de dados e tabelas disponíveis em um servidor de banco de dados. Seu resultado pode ser **nulo** se você não tiver os privilégios suficientes.

Além do método que é mostrado no bloco de código a seguir, você pode usar as consultas **SHOW TABLES** ou **SHOW DATABASES** para obter a lista de tabelas ou bancos de dados no PHP ou no PERL.

Exemplo de PERL

```
# Get all the tables available in current database.
my @tables = $dbh->tables ( );

foreach $table (@tables ){
   print "Table Name $table\n";
}
```

Exemplo de PHP

```
<?php
   $con = mysql_connect("localhost", "userid", "password");

   if (!$con) {
      die('Could not connect: ' . mysql_error());
   }
   $db_list = mysql_list_dbs($con);

   while ($db = mysql_fetch_object($db_list)) {
      echo $db->Database . "<br />";
   }
   mysql_close($con);
?>
```

Obtendo metadados do servidor

Existem alguns comandos importantes no MySQL que podem ser executados no prompt do MySQL ou usando qualquer script como o PHP para obter várias informações importantes sobre o servidor de banco de dados.

Sr. Não.	Comando e Descrição
1	**SELECIONAR VERSÃO ()** String de versão do servidor
2	**SELECT DATABASE ()** Nome do banco de dados atual (vazio se nenhum)
3	**SELECIONAR USUÁRIO ()** Nome de usuário atual

4	**SHOW STATUS** Indicadores de status do servidor
5	**MOSTRAR VARIÁVEIS** Variáveis de configuração do servidor

Usando seqüências do MySQL

Uma sequência é um conjunto de inteiros 1, 2, 3, ... que são gerados em ordem em uma demanda específica. As seqüências são freqüentemente usadas nos bancos de dados, pois muitos aplicativos exigem que cada linha de uma tabela contenha um valor exclusivo, e as sequências fornecem uma maneira fácil de gerá-las.

Este capítulo descreve como usar seqüências no MySQL.

Usando a coluna AUTO_INCREMENT

A maneira mais simples no MySQL de usar Sequences é definir uma coluna como **AUTO_INCREMENT** e deixar as coisas restantes para o MySQL para tomar cuidado.

Exemplo

Experimente o seguinte exemplo. Isto irá criar uma tabela e depois disso irá inserir algumas linhas nesta tabela onde não é necessário dar ID de registro porque é auto incrementado pelo MySQL.

```
mysql> CREATE TABLE insect
   -> (
   -> id INT UNSIGNED NOT NULL AUTO_INCREMENT,
   -> PRIMARY KEY (id),
   -> name VARCHAR(30) NOT NULL, # type of insect
   -> date DATE NOT NULL, # date collected
   -> origin VARCHAR(30) NOT NULL # where collected
);
Query OK, 0 rows affected (0.02 sec)
mysql> INSERT INTO insect (id,name,date,origin) VALUES
   -> (NULL,'housefly','2001-09-10','kitchen'),
   -> (NULL,'millipede','2001-09-10','driveway'),
   -> (NULL,'grasshopper','2001-09-10','front yard');
Query OK, 3 rows affected (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0
mysql> SELECT * FROM insect ORDER BY id;
+----+-------------+------------+------------+
| id |    name     |    date    |   origin   |
+----+-------------+------------+------------+
|  1 |  housefly   | 2001-09-10 |   kitchen  |
|  2 |  millipede  | 2001-09-10 |  driveway  |
|  3 | grasshopper | 2001-09-10 | front yard |
+----+-------------+------------+------------+
3 rows in set (0.00 sec)
```

Obter valores de AUTO_INCREMENT

O **LAST_INSERT_ID ()** é uma função SQL, portanto, você pode usá-lo em qualquer cliente que entenda como emitir instruções SQL. Caso contrário, os scripts PERL e PHP fornecem funções exclusivas para recuperar o valor auto incrementado do último registro.

Exemplo de PERL

Use o atributo **mysql_insertid** para obter o valor **AUTO_INCREMENT** gerado por uma consulta. Esse atributo é acessado por meio de um identificador de banco de dados ou de um identificador de instrução, dependendo de como você emite a consulta.

O exemplo a seguir faz referência a ele por meio do identificador do banco de dados.

```
$dbh->do ("INSERT INTO insect (name,date,origin)
VALUES('moth','2001-09-14','windowsill')");
my $seq = $dbh->{mysql_insertid};
```

Exemplo de PHP

Depois de emitir uma consulta que gera um valor AUTO_INCREMENT, recupere o valor chamando o comando **mysql_insert_id ()** .

```
mysql_query ("INSERT INTO insect (name,date,origin)
VALUES('moth','2001-09-14','windowsill')", $conn_id);
$seq = mysql_insert_id ($conn_id);
```

Renumerando uma sequência existente

Pode haver um caso em que você tenha excluído muitos registros de uma tabela e deseje sequenciar novamente todos os registros. Isso pode ser feito usando um truque simples, mas você deve ter muito cuidado para fazer isso se sua tabela estiver tendo junções com a outra tabela.

Se você determinar que a resequenciação de uma coluna AUTO_INCREMENT é inevitável, a maneira de fazer isso é soltar a coluna da tabela e adicioná-la novamente.

O exemplo a seguir mostra como renumerar os **valores de id** na tabela usando essa técnica.

```
mysql> ALTER TABLE insect DROP id;
mysql> ALTER TABLE insect
   -> ADD id INT UNSIGNED NOT NULL AUTO_INCREMENT FIRST,
   -> ADD PRIMARY KEY (id);
```

Iniciando uma sequência em um valor específico

Por padrão, o MySQL irá iniciar a sequência a partir de 1, mas você pode especificar qualquer outro número também no momento da criação da tabela.

O programa a seguir é um exemplo que mostra como o MySQL irá iniciar a sequência a partir de 100.

```
mysql> CREATE TABLE insect
   -> (
   -> id INT UNSIGNED NOT NULL AUTO_INCREMENT = 100,
   -> PRIMARY KEY (id),
   -> name VARCHAR(30) NOT NULL, # type of insect
   -> date DATE NOT NULL, # date collected
   -> origin VARCHAR(30) NOT NULL # where collected
);
```

Como alternativa, você pode criar a tabela e, em seguida, definir o valor da sequência inicial com o comando **ALTER TABLE** .

```
mysql> ALTER TABLE t AUTO_INCREMENT = 100;
```

MySQL - Lidando com Duplicados

Geralmente, tabelas ou conjuntos de resultados, por vezes, contêm registros duplicados. Na maioria das vezes, é permitido, mas às vezes é necessário parar registros duplicados. É necessário identificar registros duplicados e removê-los da tabela. Este capítulo descreverá como evitar a ocorrência de registros duplicados em uma tabela e como remover os registros duplicados já existentes.

Impedindo que duplicatas ocorram em uma tabela

Você pode usar uma **PRIMARY KEY** ou um **UNIQUE** Index em uma tabela com os campos apropriados para parar registros duplicados.

Vamos dar um exemplo - A tabela a seguir não contém esse índice ou chave primária, portanto, permitiria registros duplicados para **first_name** e **last_name** .

```
CREATE TABLE person_tbl (
   first_name CHAR(20),
   last_name CHAR(20),
   sex CHAR(10)
);
```

Para evitar que vários registros com os mesmos valores de primeiro e último nome sejam criados nesta tabela, adicione uma **PRIMARY KEY** à sua definição. Quando você fizer isso, também é necessário declarar as colunas indexadas como **NOT NULL** , porque uma **PRIMARY KEY** não permite valores **NULL** -

```
CREATE TABLE person_tbl (
   first_name CHAR(20) NOT NULL,
   last_name CHAR(20) NOT NULL,
   sex CHAR(10),
   PRIMARY KEY (last_name, first_name)
);
```

A presença de um índice exclusivo em uma tabela normalmente causa um erro se você inserir um registro na tabela que duplique um registro existente na coluna ou nas colunas que definem o índice.

Use o comando **INSERT IGNORE** em vez do comando **INSERT** . Se um registro não duplicar um registro existente, o MySQL o inserirá como de costume. Se o registro é uma duplicata, a palavra-chave **IGNORE** diz ao MySQL para descartá-lo silenciosamente sem gerar um erro.

O exemplo a seguir não comete erros e, ao mesmo tempo, não insere registros duplicados também.

```
mysql> INSERT IGNORE INTO person_tbl (last_name, first_name)
   -> VALUES( 'Jay', 'Thomas');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT IGNORE INTO person_tbl (last_name, first_name)
   -> VALUES( 'Jay', 'Thomas');
Query OK, 0 rows affected (0.00 sec)
```

Use o comando **REPLACE** em vez do comando INSERT. Se o registro é novo, ele é inserido da mesma forma que com INSERT. Se for uma duplicata, o novo registro substitui o antigo.

```
mysql> REPLACE INTO person_tbl (last_name, first_name)
   -> VALUES( 'Ajay', 'Kumar');
Query OK, 1 row affected (0.00 sec)

mysql> REPLACE INTO person_tbl (last_name, first_name)
   -> VALUES( 'Ajay', 'Kumar');
Query OK, 2 rows affected (0.00 sec)
```

Os comandos INSERT IGNORE e REPLACE devem ser escolhidos de acordo com o comportamento de tratamento duplicado que você deseja efetuar. O comando INSERT IGNORE mantém o primeiro conjunto dos registros duplicados e descarta os restantes. O comando REPLACE mantém o último conjunto de duplicados e apaga todos os anteriores.

Outra maneira de aplicar a exclusividade é adicionar um índice **UNIQUE em** vez de uma PRIMARY KEY a uma tabela.

```
CREATE TABLE person_tbl (
   first_name CHAR(20) NOT NULL,
   last_name CHAR(20) NOT NULL,
   sex CHAR(10)
   UNIQUE (last_name, first_name)
);
```

Contando e identificando duplicatas

A seguir, a consulta para contar registros duplicados com first_name e last_name em uma tabela.

```
mysql> SELECT COUNT(*) as repetitions, last_name, first_name
   -> FROM person_tbl
   -> GROUP BY last_name, first_name
   -> HAVING repetitions > 1;
```

Esta consulta retornará uma lista de todos os registros duplicados na tabela person_tbl. Em geral, para identificar conjuntos de valores duplicados, siga as etapas abaixo.

Determine quais colunas contêm os valores que podem ser duplicados.

Listar essas colunas na lista de seleção de colunas, junto com o **COUNT (*)** .

Listar as colunas na cláusula **GROUP BY** também.

Adicione uma cláusula **HAVING** que elimine os valores exclusivos, exigindo que a contagem de grupos seja maior que um.

Eliminando Duplicados de um Resultado da Consulta

Você pode usar o comando **DISTINCT** junto com a instrução SELECT para descobrir registros exclusivos disponíveis em uma tabela.

```
mysql> SELECT DISTINCT last_name, first_name
   -> FROM person_tbl
   -> ORDER BY last_name;
```

Uma alternativa ao comando DISTINCT é adicionar uma cláusula GROUP BY que nomeie as colunas que você está selecionando. Isso tem o efeito de remover duplicatas e selecionar apenas as combinações exclusivas de valores nas colunas especificadas.

```
mysql> SELECT last_name, first_name
   -> FROM person_tbl
   -> GROUP BY (last_name, first_name);
```

Removendo Duplicados Usando Substituição de Tabela

Se você tiver registros duplicados em uma tabela e quiser remover todos os registros duplicados da tabela, siga o procedimento abaixo.

```
mysql> CREATE TABLE tmp SELECT last_name, first_name, sex
   -> FROM person_tbl;
   -> GROUP BY (last_name, first_name);

mysql> DROP TABLE person_tbl;
mysql> ALTER TABLE tmp RENAME TO person_tbl;
```

Uma maneira fácil de remover registros duplicados de uma tabela é adicionar um INDEX ou uma PRIMARY KEY a essa tabela. Mesmo que essa tabela já esteja disponível, você pode usar essa técnica para remover os registros duplicados e também estará seguro no futuro.

```
mysql> ALTER IGNORE TABLE person_tbl
   -> ADD PRIMARY KEY (last_name, first_name);
```

MySQL - e SQL Injection

Se você der entrada do usuário através de uma página da Web e inseri-lo em um banco de dados MySQL, há uma chance de você ter ficado aberto para um problema de segurança conhecido como **SQL Injection** . Este capítulo ensinará como evitar que isso aconteça e ajudará a proteger seus scripts e instruções MySQL.

A Injeção de SQL geralmente ocorre quando você solicita a entrada de um usuário, como seu nome e, em vez de um nome, ele fornece uma instrução do MySQL que você inadvertidamente executará em seu banco de dados.

Nunca confie nos dados fornecidos por um usuário, processe esses dados somente após a validação; Como regra, isso é feito por correspondência de padrões. No exemplo a seguir, o nome de usuário é restrito a caracteres alfanuméricos mais sublinhado e a um tamanho entre 8 e 20 caracteres - modifique essas regras conforme necessário.

```
if (preg_match("/^\w{8,20}$/", $_GET['username'], $matches)) {
   $result = mysql_query("SELECT * FROM users WHERE username = $matches[0]");
} else  {
   echo "username not accepted";
}
```

Para demonstrar esse problema, considere o seguinte trecho.

```
// supposed input
$name = "Qadir'; DELETE FROM users;";
mysql_query("SELECT * FROM users WHERE name = '{$name}'");
```

A chamada de função deve recuperar um registro da tabela users, onde a coluna name corresponde ao nome especificado pelo usuário. Em circunstâncias normais, $ name conteria apenas caracteres alfanuméricos e talvez espaços. Mas aqui, anexando uma consulta totalmente nova a **$ name** , a chamada para o banco de dados se transforma em um desastre. A consulta DELETE injetada remove todos os registros dos usuários.

Felizmente, se você usa o MySQL, a função **mysql_query ()** não permite o empilhamento de consultas ou a execução de múltiplas consultas em uma única chamada de função. Se você tentar empilhar consultas, a chamada falhará.

No entanto, outras extensões de banco de dados PHP, como **SQLite** e **PostgreSQL** , executam consultas empilhadas, executando todas as consultas fornecidas em uma cadeia de caracteres e criando um sério problema de segurança.

Impedindo a injeção de SQL

Você pode manipular todos os caracteres de escape inteligentemente em linguagens de script como PERL e PHP. A extensão MySQL para PHP fornece a função **mysql_real_escape_string ()** para escapar caracteres de entrada que são especiais para o MySQL.

```
if (get_magic_quotes_gpc()) {
   $name = stripslashes($name);
}
```

```
$name = mysql_real_escape_string($name);
mysql_query("SELECT * FROM users WHERE name = '{$name}'");
```

O Dilema LIKE

Para resolver o dilema LIKE, um mecanismo de escape personalizado deve converter caracteres% e _ fornecidos pelo usuário em literais. Use **addcslashes ()** , uma função que permite especificar um intervalo de caracteres para escapar.

```
$sub = addcslashes(mysql_real_escape_string("%something_"), "%_");
// $sub == \%something\_
mysql_query("SELECT * FROM messages WHERE subject LIKE '{$sub}%'");
```

MySQL - Exportação de Banco de Dados

A maneira mais simples de exportar os dados de uma tabela para um arquivo de texto é usando a **instrução SELECT ... INTO OUTFILE** que exporta um resultado de consulta diretamente para um arquivo no host do servidor.

Exportando dados com a instrução SELECT ... INTO OUTFILE

A sintaxe desta instrução combina um comando **SELECT** regular com o **nome** do **arquivo INTO OUTFILE** no final. O formato de saída padrão é o mesmo do comando LOAD DATA. Portanto, a instrução a seguir exporta a tabela **tutorials_tbl** para **/tmp/tutorials.txt** como um arquivo terminado por alimentação de linha e delimitado por tabulação.

```
mysql> SELECT * FROM tutorials_tbl
   -> INTO OUTFILE '/tmp/tutorials.txt';
```

Você pode alterar o formato de saída usando várias opções para indicar como citar e delimitar colunas e registros. Para exportar a tabela tutorial_tbl em um formato CSV com linhas terminadas por CRLF, use o seguinte código.

```
mysql> SELECT * FROM passwd INTO OUTFILE '/tmp/tutorials.txt'
   -> FIELDS TERMINATED BY ',' ENCLOSED BY '"'
   -> LINES TERMINATED BY '\r\n';
```

O **SELECT ... INTO OUTFILE** tem as seguintes propriedades -

O arquivo de saída é criado diretamente pelo servidor MySQL, portanto, o nome do arquivo deve indicar onde você deseja que o arquivo seja gravado no host do servidor. Não há versão LOCAL da declaração análoga à versão **LOCAL** de **LOAD DATA** .

Você deve ter o privilégio **MySQL FILE** para executar a **instrução SELECT ... INTO** .

O arquivo de saída ainda não deve existir. Isso impede que o MySQL destrua arquivos que possam ser importantes.

Você deve ter uma conta de login no host do servidor ou alguma maneira de recuperar o arquivo desse host. Caso contrário, o comando **SELECT ... INTO OUTFILE** provavelmente não terá valor para você.

No UNIX, o arquivo é criado **legível pelo mundo** e pertence ao servidor MySQL. Isso significa que, embora você possa ler o arquivo, talvez não seja possível excluí-lo.

Exportando Tabelas como Dados Brutos

O programa **mysqldump** é usado para copiar ou fazer backup de tabelas e bancos de dados. Ele pode gravar a saída da tabela como um **arquivo de dados brutos** ou como um conjunto de instruções **INSERT** que recriam os registros na tabela.

Para descarregar uma tabela como um arquivo de dados, você deve especificar uma opção **--tab** que indique o diretório, onde você deseja que o servidor MySQL **grave** o arquivo.

Por exemplo, para descarregar a tabela **tutorials_tbl** do banco de dados **TUTORIALS** para um arquivo no **diretório / tmp** , use um comando como mostrado abaixo.

```
$ mysqldump -u root -p --no-create-info \
   --tab=/tmp tutorials tutorials_tbl
password ******
```

Exportando Conteúdo da Tabela ou Definições no Formato SQL

Para exportar uma tabela no formato SQL para um arquivo, use o comando mostrado abaixo.

```
$ mysqldump -u root -p TUTORIALS tutorials_tbl > dump.txt
password ******
```

Isso criará um arquivo com o conteúdo mostrado abaixo.

```
-- MySQL dump 8.23
--
-- Host: localhost    Database: TUTORIALS
---------------------------------------------------------
-- Server version       3.23.58

--
-- Table structure for table `tutorials_tbl`
--

CREATE TABLE tutorials_tbl (
   tutorial_id int(11) NOT NULL auto_increment,
   tutorial_title varchar(100) NOT NULL default '',
```

```
   tutorial_author varchar(40) NOT NULL default '',
   submission_date date default NULL,
   PRIMARY KEY  (tutorial_id),
   UNIQUE KEY AUTHOR_INDEX (tutorial_author)
) TYPE = MyISAM;

--
-- Dumping data for table `tutorials_tbl`
--

INSERT INTO tutorials_tbl
   VALUES (1,'Learn PHP','John Poul','2007-05-24');
INSERT INTO tutorials_tbl
   VALUES (2,'Learn MySQL','Abdul S','2007-05-24');
INSERT INTO tutorials_tbl
   VALUES (3,'JAVA Tutorial','Sanjay','2007-05-06');
```

Para descarregar várias tabelas, nomeie-as todas seguidas pelo argumento do nome do banco de dados. Para despejar um banco de dados inteiro, não nomeie nenhuma tabela após o banco de dados, conforme mostrado no bloco de código a seguir.

```
$ mysqldump -u root -p TUTORIALS > database_dump.txt
password ******
```

Para fazer backup de todos os bancos de dados disponíveis em seu host, use o código a seguir.

```
$ mysqldump -u root -p --all-databases > database_dump.txt
password ******
```

A opção --all-databases está disponível na versão 3.23.12 do MySQL. Esse método pode ser usado para implementar uma estratégia de backup de banco de dados.

Copiando tabelas ou bancos de dados para outro host

Se você deseja copiar tabelas ou bancos de dados de um servidor MySQL para outro, então use o **mysqldump** com o nome do banco de dados e o nome da tabela.

Execute o seguinte comando no host de origem. Isso irá despejar o banco de dados completo no arquivo **dump.txt** .

```
$ mysqldump -u root -p database_name table_name > dump.txt
password *****
```

Você pode copiar o banco de dados completo sem usar um nome de tabela específico, conforme explicado acima.

Agora, o arquivo ftp dump.txt em outro host e use o seguinte comando. Antes de executar este comando, certifique-se de ter criado database_name no servidor de destino.

```
$ mysql -u root -p database_name < dump.txt
password *****
```

Outra maneira de conseguir isso sem usar um arquivo intermediário é enviar a saída do mysqldump diretamente pela rede para o servidor MySQL remoto. Se você puder se conectar a ambos os servidores do host em que o banco de dados de origem reside, use o seguinte comando (Certifique-se de ter acesso em ambos os servidores).

```
$ mysqldump -u root -p database_name \
   | mysql -h other-host.com database_name
```

No mysqldump, metade do comando se conecta ao servidor local e grava a saída do dump no pipe. A metade restante do comando se conecta ao servidor MySQL remoto no outro-host.com. Ele lê o pipe para entrada e envia cada instrução para o servidor other-host.com.

MySQL - Importação de Banco de Dados - Métodos de Recuperação

Existem duas maneiras simples no MySQL para carregar dados no banco de dados MySQL a partir de um arquivo previamente submetido a backup.

Importando dados com LOAD DATA

O MySQL fornece uma instrução LOAD DATA que atua como um carregador de dados em massa. Aqui está uma instrução de exemplo que lê um arquivo **dump.txt** do seu diretório atual e o carrega na tabela **mytbl** no banco de dados atual.

```
mysql> LOAD DATA LOCAL INFILE 'dump.txt' INTO TABLE mytbl;
```

Se a palavra-chave **LOCAL** não estiver presente, o MySQL procurará pelo arquivo de dados no host do servidor, usando a **observação em nome absoluto do caminho** , que especifica completamente a localização do arquivo, começando pela raiz do sistema de arquivos. O MySQL lê o arquivo a partir da localização especificada.

Por padrão, **LOAD DATA** assume que os arquivos de dados contêm linhas que são finalizadas por alimentações de linha (novas linhas) e que os valores de dados dentro de uma linha são separados por guias.

Para especificar um formato de arquivo explicitamente, use uma cláusula **FIELDS** para descrever as características dos campos dentro de uma linha e uma cláusula **LINES** para especificar a seqüência de final de linha. A seguinte instrução **LOAD DATA** especifica que o arquivo de dados contém valores separados por dois-pontos e linhas terminadas por retornos de carro e novo caractere de linha.

```
mysql> LOAD DATA LOCAL INFILE 'dump.txt' INTO TABLE mytbl
   -> FIELDS TERMINATED BY ':'
   -> LINES TERMINATED BY '\r\n';
```

O comando LOAD DATA supõe que as colunas no arquivo de dados tenham a mesma ordem que as colunas na tabela. Se isso não for verdade, você pode especificar uma lista para indicar em quais colunas da tabela as colunas do arquivo de dados devem ser carregadas. Suponha que sua tabela tenha colunas a, b e c, mas colunas sucessivas no arquivo de dados correspondem às colunas b, c e a.

Você pode carregar o arquivo conforme mostrado no bloco de código a seguir.

```
mysql> LOAD DATA LOCAL INFILE 'dump.txt'
   -> INTO TABLE mytbl (b, c, a);
```

Importando Dados com o mysqlimport

O MySQL também inclui um programa utilitário chamado **mysqlimport** que atua como um wrapper em torno de LOAD DATA, para que você possa carregar os arquivos de entrada diretamente a partir da linha de comando.

Para carregar dados do **dump.txt** no **mytbl** , use o seguinte comando no prompt do UNIX.

```
$ mysqlimport -u root -p --local database_name dump.txt
password *****
```

Se você usar **mysqlimport** , as opções da linha de comandos fornecerão os especificadores de formato. Os comandos **mysqlimport** que correspondem às duas instruções **LOAD DATA** anteriores são exibidos conforme mostrado no bloco de códigos a seguir.

```
$ mysqlimport -u root -p --local --fields-terminated-by = ":" \
   --lines-terminated-by = "\r\n"  database_name dump.txt
password *****
```

A ordem em que você especifica as opções não importa para o mysqlimport, exceto pelo fato de que todas elas devem preceder o nome do banco de dados.

A instrução **mysqlimport** usa a opção **--columns** para especificar a ordem da coluna -

```
$ mysqlimport -u root -p --local --columns=b,c,a \
   database_name dump.txt
password *****
```

Manipulando citações e caracteres especiais

A cláusula FIELDS pode especificar outras opções de formato além de **TERMINATED BY** . Por padrão, LOAD DATA assume que os valores não estão marcados e interpreta a barra invertida (\) como um caractere de escape para os caracteres especiais. Para indicar o caractere de cotação de valor explicitamente, use o comando **ENCLOSED BY** . O MySQL irá remover esse caractere das extremidades dos valores de dados durante o

processamento de entrada. Para alterar o caractere de escape padrão, use **ESCAPED BY** .

Quando você especifica ENCLOSED BY para indicar que os caracteres de aspas devem ser removidos dos valores de dados, é possível incluir o caractere de aspas literalmente nos valores de dados, duplicando-as ou precedendo-as com o caractere de escape.

Por exemplo, se os caracteres de aspas e escape forem "e \, o valor de entrada **" a "" b \ "c"** será interpretado como **"b" c** .

Para **mysqlimport** , as opções de linha de comando correspondentes para especificar os valores de quote e escape são **--fields-enclosed-by** e **--fields-escaped-by** .

Página anterior Próxima página