O Portal do conhecimento

http://apostilando.com

Tutorial MySQL – Apostilando.com

Table of Contents

Tutorial − MySQL

Revisão 2

Sobre este tutorial

Este documento aborda superficialmente o MySQL. Sua leitura é recomendada para leigos em servidores SQL e até mesmo em base de dados. A leitura deste documento não é recomendada para pessoas que já tenham alguma experiência no assunto, e possam ficar entediadas lendo este.

O texto aqui contido pode e deve ser distribuído livremente, contanto que sejam mantidos os nomes dos autores, e que fique documentada alguma eventual alteração.

Vale lembrar ainda, que este texto também se aplica a outros servidores de Base de Dados, como o Oracle (R), tendo em conta que a abordagem do texto é da linguagem SQL, que é um padrão.

Nota do autor

Ao escrever este manual, não pude gastar muito tempo dando atenção a todos os itens. Peço desculpas antecipadamente por alguma falha no documento, e agradeço eventuais colaborações e sugestões. Pretendo manter atualizações deste documento em minha página pessoal (http://www.linuxqos.cjb.net) portanto, as versões mais novas sempre estarão lá primeiro. Enfim, aproveitem!

Rafael V. Aroca − rafael@linuxqos.cjb.net
São Carlos, 20 de fevereiro de 2000.
Atualizada em: 18/Set/2001
SP − Brasil

Molhando seus pés

Breve introdução ao MySQL

Bancos de dados, hoje em dia, são parte fundamental da vida de quase todos seres humanos. Viver sem sistemas de bancos de dados em nosso mundo, é hoje em dia praticamente impossível. Sem nem mesmo perceber, a todo momento estamos usando um banco de dados, mesmo nas mais triviais tarefas.

Bancos, universidades e bibliotecas são três exemplos de organizações que dependem totalmente de bancos de dados. A própria internet, usa um sistema de banco de dados para controle e funcionamento dos sites, como este.

Normalmente, bases de dados com muitas informações são armazenadas em computadores de grande porte, chamados de servidores, e que permitem o uso das mesmas informações, através de uma rede, por um número (diz−se) ilimitado de usuários.

Um dos mais rápidos programas para servidores de SQL (do inglês, "Linguagem de pesquisa simples"), hoje no mercado, é o MySQL, desenvolvido pela T.c.X. DataKonsultAB. Este programa está disponível para download em sua versão original em www.mysql.com, em inglês, e também em www.mysql.com.br para sua versão brasileira, onde encontram−se projetos de tradução e documentação do MySQL em português.

Além de oferecer vários recursos não existentes em outros servidores, o MySQL tem a vantagem de ser totalmente gratuito para uso tanto comercial, quanto privado, em conformidade com a licença pública GPL.

As principais metas da equipe de desenvolvimento do MySQL é construir um servidor rápido e robusto.

Os recursos acima mencionados incluem:

− Capacidade de lidar com um número ilimitado de usuários;
− Capacidade de manipular mais de cinqüenta milhões (50.000.000) de registros;
− Execução muito rápida de comandos, provavelmente o mais rápido do mercado;
− Sistema de segurança simples e funcional.

Quem usa o MySQL:

− Silicon Graphics (www.sgi.com)

− Siemens (www.siemens.com)
− Yahoo (www.yahoo.com)
− IFX Networks (www.ifx.com.br)
− Dezenas de Web hosting e Provedores devido ao enorme sucesso que o MySQL vem fazendo

Existem sistemas rodando servidor MySQL com bases de dados de 200 Gigabytes!!! Caso você ache isto insuficiente, verifique a lista completa de usuários do MySQL no site oficial − www.mysql.com

Indo mais fundo

Visão geral

A partir deste capítulo, começaremos a usar o MySQL na prática, para tal, é preciso que você tenha acesso a algum servidor com o MySQL instalado e funcionando devidamente. A instalação do MySQL será abordada posteriormente. A forma mais comum de se utilizar o MySQL, é via telnet. Uma vez conectado ao servidor, um segundo comando permite o acesso ao servidor MySQL.

Para acessar o servidor MySQL, é preciso utilizar o comando que segue. É importante lembrar, que o MySQL tem seu próprio cadastro de nomes e senhas, sendo que você pode possuir uma conta no servidor SQL, mas não telnet, e o oposto também é válido. Você também pode usar clienter gráficos para o MySQL, como o FreeMascon e o MyNavigator.

_____

mysql −u elaine −p supersenha

Sintaxe:

mysql −u (usuário) −p (senha)

ou

mysql −h servidor −u (usuário) −−password=(senha)

−−−

Após validada a senha, e pré−supondo que o MySQL tenha sido corretamente instalado, você verá algo como:

−−−

Cliente MySQL − NBS

Bem vindo ao monitor do MySQL. Use ; após os comandos ou \g.

Sua id é 49, e você está conectado a um servidor versão 3.21.23−beta−log

Digite 'help' para ajuda.

mysql>

−−−

Uma vez no prompt do MySQL, podemos começar a utilizar os comandos do MySQL e manipular os dados e o servidor. Contudo, antes de modificar a base de dados, nós devemos escolher qual desejamos usar, da seguinte forma:

−−−

```
mysql>use teste;

Base de dados alterada.
```

−−

Troque teste, pelo nome da base de dados desejada. Você obterá uma mensagem confirmando a alteração da base de dados. Isto significa que você está conectado a ela.

Repare que o comando está seguido de ponto e vírgula, pois como em C, a maioria dos comandos do MySQL são sucedidos por ponto e vírgula.

Antes de fazer qualquer coisa, seria interessante você consultar a ajuda, que listará os comandos disponíveis neste momento da execução do MySQL.

Isto deve ser feito através do comando help.

−−

```
mysql>help

...

...
```

−−

Provavelmente, nem todas as funções serão úteis neste momento, porém é bom gastar um tempo aprendendo cada uma delas para uso futuro.

Por outro lado, funções como status, connect, clear, e quit serão usadas com uma freqüência tão grande que você deve se familiarizar com elas.

Neste ponto, você deve ter um conhecimento básico de como conectar−se ao servidor, selecionar a base de dados, e executar operações simples. No próximo capítulo, aprenderemos os conceitos e técnicas necessárias

para preparar e manter uma base de dados.

Até seu pescoço em MySQL

Tabelas e campos

Chegamos! Talvez no capítulo mais difícil deste texto. Você logo entenderá, portanto, tenha uma ótima noção de bases de dados!

Algumas pessoas podem achar fácil, mas minha experiência no assunto, diz que muitas pessoas têm dificuldades sobre os conceitos básicos de bases de dados. Portanto, só passe para o próximo tópico quando você estiver plenamente seguro que domina os conceitos de bases de dados.

Uma base de dados, nada mais é do que estruturas complexas de dados. Estes dados são gravados em forma de registros em tabelas. Parece simples, certo? Façamos então uma analogia para que este conceito se torne ainda mais simples:

Imagine um arquivo de fichas, numa empresa onde há várias caixas, cada uma contendo os dados dos funcionários de um certo setor. Cada caixa possui várias fichas, que são os cadastros dos funcionários − cada ficha contém os dados de apenas um funcionário. Indo mais longe, podemos concluir que cada ficha contém diversas informação sobre o funcionário em questão:

Portanto, cada caixa é uma tabela, contendo diversas fichas, que são os registros, e cada ficha possui várias informações sobre o funcionário, que são os campos.

Como foi dito, há várias caixas, uma para cada departamento, a soma de todas as caixas forma a base de dados.

Observando tudo isto de fora, podemos formar o seguinte esquema:

Base de dados < Tabela < Registro < Coluna (datatype)
Banco de dados < Tabela < Linha < Campo

As duas linhas acima mostram os termos normalmente usados para o que acabamos de aprender. Os campos podem ser de diferentes tipos e tamanhos, permitindo ao programador criar tabelas que satisfaçam ao escopo do projeto. A decisão de quais campos usar e quais não usar é muito importante, pois influi drasticamente na performance da base de dados que estamos desenvolvendo, portanto, é de bom grado um conhecimento sólido destes conceitos.

A etapa de montagem das tabelas, é senão a mais importante, uma das mais importantes etapas da montagem de uma base de dados, pois um bom projeto pode facilitar muito o trabalho de programação.

Campos

Como já sabemos, os campos são a parte fundamental de uma base de dados. É nos campos que as informações ficam armazenadas. Para um otimização da base de dados, antes e utilizar, devemos definir os campos que desejamos usar, e especificar o que cada um pode conter.

Tipos de campos no MySQL

O MySQL oferece os mais comuns campos, que até mesmo um programador novato já deve ter visto. Alguns deles estão aqui listados:

CHAR(M)

Os campos CHAR são usados para caracteres alfanuméricos, como endereços e nomes. Seu tamanho é fixo e instaurado ao ser criado. Um campo do tipo CHAR pode ter de 1 a 255 caracteres.

Exemplo:

endereco_comercial CHAR(10);

Define um campo chamado 'endereco_comercial', que pode conter até dez letras.
Observe que não há acentos no nome do campo, pois muitos servidores não acentuam, e sua tabela teria difícil acesso.

VARCHAR(M)

Sua estrutura e funcionamento é idêntico ao campo anterior, salvo que no tipo CHAR, o tamanho definido é fixo, e mesmo que não usado, aquele espaço em disco é alocado. Já o campo VARCHAR, aloca apenas o espaço necessário para gravação. Contudo, vale lembrar que trocamos espaço por velocidade, pois este campo é 50% mais lento que o anterior.

Exemplo:

endereco_comercial VARCHAR(10);

Define um campo chamado endereco_comercial que pode conter até dez letras. Se você preencher apenas duas, o campo não ocupara todos dez bytes, mas apenas 2.

INT(M) [Unsigned]

O campo INT, que como o próprio número diz, suporta o conjunto dos números inteiros numa variação de −2147483648 a 2147483647. O parâmetro Unsigned pode ser passado, excluindo os números negativos, proporcionando um intervalo de 0 até 4294967295.

Exemplos:

carros_estocados INT;

Inteiro válido: −245
Inteiro válido: 245
Inteiro inválido: 3000000000

carros_estocados INT unsigned;

Inteiro válido: 245
Inteiro inválido: −245
Inteiro inválido: 3000000000

FLOAT[(M,D)]

Os pontos flutuantes (FLOAT) representam pequenos números decimais, e são usados para representar números com maior precisão.

Exemplo:

voltagem_cadeira_eletrica FLOAT(4,2);

Float válido: 324.50

DATE

Campo usado para armazenar informações referentes a data. A forma padrão , é 'AAAA−MM−DD', onde AAAA corresponde ao ano, MM ao mês, e DD ao dia. Ele pode variar de 0000−00−00 a 9999−12−31.

O MySQL possui um conjunto poderoso de comandos de formatação e manipulação de datas. Consulte documentação adicional, se houver interesse.

Exemplo:

data_de_nascimento DATE;

Data válida: 1999−12−06
Data inválida: 1999−06−12

TEXT/BLOB

Os campos texto e blob são usados para guardar grandes quantidades de caracteres. Podendo conter de 0 a 65535 bytes, os blobs e texts são úteis para armazenar documentos completos, como este que você está lendo. A única diferença entre os campos BLOB e TEXT está no fato de um campo TEXT não ser sensível a letras maiúsculas e minúscula quando uma comparação é realizada, e os BLOBs sim.

Exemplo:

relatorio BLOB;

BLOB válido: 'Minha terra tem palmeiras onde canta o...'

relatorio TEXT;

TEXT válido: 'A que saudades que eu sinto...'

SET

Um campo interessante, que permite que o usuário faça uma escolha dado determinado número de opções. Cada campo pode conter até, 64 opções.

Exemplo:

vicio SET("cafe", "cigarro") NOT NULL;

Neste exemplo este campo pode conter apenas os seguintes itens:

""
"cafe"
"cigarro"
"cafe,cigarro"

ENUM

Um campo com funcionamento semelhante ao SET, com a diferença que apenas um valor pode ser escolhido.

Exemplo:

virtude ENUM("programar", "amar") NOT NULL;

Neste exemplo este campo pode conter os seguintes valores:

""
"programar"

"amar"

Registros

Um conjunto de campos relacionados, forma o que chamamos de registro. Um registro, portanto, pode ter a seguinte estrutura:

```
nome CHAR(15);
email CHAR(25);
telefone INT;
```

Neste exemplo, nosso registro contém três campos, podendo armazenar o email e o telefone de uma determinada pessoa. Observe que o campo nome foi definido como CHAR, portanto poderá conter qualquer tipo de caractere, contudo, o campo telefone, definido como INT, poderá apenas conter números, pois foi configurado como INT.

Tabelas

Um conjunto de registros, forma uma tabela. As tabelas portanto, armazenam grande quantidade de dados. Como no exemplo anterior, poderíamos ter centenas de nomes diferentes cadastrados em nossa tabela de pessoas. Cada conjunto de dados corresponde a um registro.

Antes de usar uma base de dados, ou dar qualquer comando, nós precisamos de uma tabela pelo menos, para armazenar os dados:

Isto pode ser feito usando o comando CREATE TABLE, que recebe como parâmetro o nome da tabela que desejamos usar.

_____

```
mysql> CREATE TABLE teste(
>codigo INT,
>nome CHAR(15),
>email CHAR(25),
>telefone INT);
```

_____

Notas:
− Não é possível criar duas tabelas com o mesmo nome;
− Cada conjunto de dados, quando visto em uma tabela, é também chamado linha;
− Você acaba de criar sua primeira tabela!

Características das linhas:
− O nome de uma coluna pode ter até 64 letras;
− O nome de uma coluna pode começar com um número;
− O nome de uma coluna não pode ser composto de números apenas.

Opções de uma tabela:

As seguintes opções podem ser adicionadas a qualquer campo de uma tabela, concatenando recursos adicionais a estes campos.

−Chave primária:

Usado para que não seja permitido, que o usuário consiga cadastrar dois registros com chaves primárias iguais. Isto é claramente útil, quando não é desejado que seja digitado um segundo registro igual ao primeiro por engano. Para se definir uma chave primária, basta adicionar 'PRIMARY KEY' a definição do campo que se deseja a não duplicidade.

Exemplo:
nome CHAR(15) PRIMARY KEY;

Esta declaração faz com que não seja permitido o cadastro na tabela de dois registros com nomes iguais.

−Auto incremento:

Este recurso, faz com que conforme novos registros são criados, automaticamente estes obtém valores que correspondem ao valor deste mesmo campo no registro anterior, somado a 1.

Exemplo:
codigo INT AUTO_INCREMENT;

Soma um a cada registro automaticamente neste campo. Começando de 1, com inserção subsequente.

Comandos relativos as tabelas:

Podemos executar comandos para saber as condições que as tabelas se encontram, e também manipulá−las. Abaixo, poderemos conhecer os fundamentais:

−Mostrar tabelas
Função:
Lista todas as tabelas existentes no banco de dados atual.
Comando:
mysql>show tables;

−Mostrar colunas
Função:
Mostra as informações referentes a estrutura, ou seja, as colunas da tabelas desejada.
Comando:
mysql>show columns from teste;

Lembre−se de dedicar algum tempo a testes. Não tenha medo de danificar a base de dados, pois o Administrador do servidor que você usa não daria a você controle total dos dados, e se o servidor pertencer a você..."vai, faça o que tu queres".

Totalmente imerso em MySQL

Manipulando a base de dados

Uma base de dados pode ser manipulada com quatro operações básicas: Incluir, Apagar, Altera, e Pesquisar. Estes tópicos serão brevemente abordados nas seções seguintes, mas antes de mais nada, eu gostaria de lembrar que como toda linguagem para computadores, o MySQL tem suas regras. Um erro de parênteses que seja pode resultar no inverso do que você espera. Portanto, fique atento a sintaxe de seus comandos.
É bom saber também, que faremos todas experiências a partir de agora, usando a tabela teste, criada no capítulo anterior. Portanto se você ainda não a criou, esta é uma boa hora.

Aqui está ela novamente como uma rápida referência:

Seleciona base de dados:

−−−

use teste;

−−−

Cria tabela:

−−−

```
mysql> CREATE TABLE teste(
>codigo INT,
>nome CHAR(15),
>email CHAR(25),
>telefone INT);
```

−−−

Inserindo registros

Para se adicionar dados a uma tabela, usamos o comando INSERT, que diz por si só sua função, como o exemplo que segue:

−−−

```
mysql>INSERT INTO teste VALUES

mysql>(NULL, 'Ernesto', 'ernesto@nbsnet.com.br',

mysql>2742729);
```

−−−

Observações:

−Apostrofes foram colocadas na entrada de campos tipo CHAR. Todos os campos que contém texto, ou seja, CHAR, VARCHAR, BLOB, TEXT, etc. têm de ficar entre apóstrofes, ou um erro ocorrerá;

−Para campos do tipo número, não se usam apóstrofes.

−A entrada NULL em um campo do tipo auto−incremento, permite que o MySQL providencie o conteúdo deste campo de forma automática. No caso do primeiro campo, o valor será 1, no segundo 2, no terceiro 3 e assim consecutivamente.

−Se possuíssemos um campo DATE, a entrada NULL faria com que o valor gravado no registro se torne a data atual.

Nota:
É importante lembrar−se sempre de passar para o camando INSERT um número de parâmetros igual ao número de campos na tabela que está recebendo os dados. Caso contrario, você obterá uma mensagem de erro. O mesmo erro também ocorre ao tentar inserir mais parâmetros do que o número de campos suportado pela tabela.

Nota 2:
Uma das grandes vantagens do MySQL é a capacidade de facilmente converter sem problemas entre campos. O sistema automaticamente converte entre números, caracteres, e datas sem problemas.

Pesquisando registros

Imagine que possuímos um carro de última geração, com todos os recursos que a tecnologia atual pode permitir. Contudo, nosso carro não tem roda, ou seja, não serve para nada, a não ser ostentar. (Opinião pessoal: assim são os servidores M$).

A intenção de uma base de dados, é ser útil, portanto, em contradição ao exemplo, ela precisa ter rodas. No

caso de qualquer base de dados, uma das roda é a pesquisa, afinal de contas, o que seria um site de busca na Internet sem pesquisa?

As pesquisas no MySQL são feitas através do comando SELECT. Este comando pode fazer desde uma simples e trivial pesquisa até uma pesquisa extremamente complexa.

Exemplos:

Ação:

```
mysql>SELECT * FROM teste;
```

Resultado:
Lista todos registros da tabela teste.

Ação:

```
mysql>SELECT * FROM teste
mysql>WHERE (nome = "Ernesto");
```

Resultado:
Lista todos os registros da tabela teste que contém "Ernesto" no campo nome.

Nota:
Consulte maiores detalhes, e estude mais itens relacionados a pesquisas SQL. Você pode economizar centenas de linha de código em seus programas usando recursos de pesquisa.

Apagando registros

Quando um registro não nos é mais útil, podemos apagá−lo, para que a tabela não contenha dados obsoletos.

Isto pode ser facilmente feito usando o comando DELETE, que tem o funcionamento semelhante ao comando INSERT.

Ação:

```
mysql>DELETE FROM teste
mysql>WHERE (telefone = 2744747);
```

Resultado:
Apaga da tabela teste todos os registros que têm o conteúdo "2744747" no campo telefone.

Alterando registros

Suponhamos agora que sua namorada trocou o telefone. A primeira providência, é alterar sua tabela para que você não possua dados desatualizados. Para tal, existe o camando UPDATE. Observe os exemplos:

Ação:

```
mysql>UPDATE teste SET nome = 'ErnestaoOBao'
mysql>WHERE nome = 'Ernesto';
```

Resultado:
Procura na tabela um registro que contenha no campo nome o conteúdo 'Ernestao', definido pelo comando WHERE. Encontrado o registro, ele é substituido pelo nome definido no comando SET, que é 'ErnestaoOBao'.

Afogando−se em MySQL

Comando avançados

O que aprendemos até agora, quando comparado a quantidade de recursos do MySQL, não chega a ser a ponta de um gigante ice−berg. O MySQL é repleto de comandos avançados que permitem fazer operações para todas as necessidades. Veremos agora alguns comandos avançados do MySQL:

Operadores lógicos:

O MySQL suporta todas operações lógicas, estando abaixo listadas as mais conhecidas:

AND (&&)
O operador lógico AND, ou E, deve ser usado em uma pesquisa que se deseja entrar dois valores. O AND, verifica ambas as clausulas da comparação, e só retorna algum valor se as duas tiverem uma resposta verdadeira. Observe o exemplo:

−−

```
mysql>SELECT * FROM teste WHERE
mysql>(nome = "Aline") AND
mysql>(telefone = 2728918);
```

−−

Esta pesquisa mostrara todos os registros que contém no campo nome e conteúdo "Aline", E (AND) no campo telefone, o conteúdo 2728988.

OR (||)
O operador lógico OR, ou OU, deve ser usado em uma pesquisa que se deseja entrar dois valores. O OR,

verifica ambas as clausulas da comparação, e retorna valores se qualquer um dos membros obtiver resultado.

−−

```
mysql>SELECT * FROM teste WHERE
mysql>(nome = "Aline") OR
mysql>(telefone = 2728918);
```

−−

Este pesquisa fará com que todos os resultados que contenham o conteúdo "Aline" no campo nome, OU 2728918 sejam exibidos na tela.

NOT (!)

O operador lógico NOT, ou NÃO, realiza uma pesquisa, excluindo valores determinados do resultado.

−−

```
mysql>SELECT * FROM teste WHERE
mysql>(nome != "Aline");
```

−−

Esta pesquisa listara todos os registros da base de dados teste, NÃO (NOT) mostrando aqueles que possuem "Aline" como conteúdo do campo nome.

ORDER BY

O operador lógico ORDER BY, ou ORDENAR POR, simplesmente lista os registros, colocando−os em ordem de acordo com o campo solicitado.

−−

```
mysql>SELECT * FROM teste WHERE
mysql>(nome = "Aline")
mysql>ORDER BY telefone;
```

−−

O resultado desta busca resultara em todos os registros contendo "Aline" no campo nome, e a listagem será organizada de acordo com a ordem do telefone.