REVISTA DIGITAL

SQL

Curta sua pipoca com MySQL

Aprenda alguns truques e conceitos da linguagem SQL aplicada ao MySQL e monte sua locadora

Domine a língua MySQL

Uma abordagem teórica-prática sobre a linguagem de banco de dados mais usada no mundo

DIGERATI

Curta sua pipoca com MySQL

Aprenda alguns truques e conceitos da linguagem SQL aplicada ao MySQL e monte sua locadora

por Michael Granados

Imagine a seguinte situação: você entra em uma locadora, escolhe seu filme e pede à balconista. Ela pergunta seu código, mas você não se lembra. A garota procura seu cadastro por meio de uma parte do seu nome, ou outro dado que você tenha informado em sua ficha. Então, ela insere a locação no sistema com registro de data para controlar quantos dias você ficará com o filme. O sistema dessa locadora não é difícil de se reproduzir, desde que eles saibam utilizar seu banco de dados.

Os recursos citados no parágrafo acima estão neste tutorial, que ensina, de forma prática, como aproveitar bem o MySQL. O sistema possuirá três tabelas básicas: clientes, filmes e aluguéis. Em aluguéis existe apenas a referência aos dados das outras duas tabelas, mas o banco retornará todas as informações juntas, automaticamente.

Ao contrário do que muitos acreditam, o banco de dados não é somente uma ferramenta de armazenamento de informações, mas pode, também, manipular e retornar as informações pré-configuradas. O programador, geralmente, aprofunda-se na linguagem que acha mais interessante, como PHP, ASP ou Java, e termina por não perceber que, cedo ou tarde, terá de colocar as informações em um banco de dados. Não para fazer um backup de todas as matérias do site, mas para dar praticidade aos sistemas. O que ele não sabe é que, por não conhecer bastante a linguagem, ele pode perder funcionalidades importantes do próprio banco.

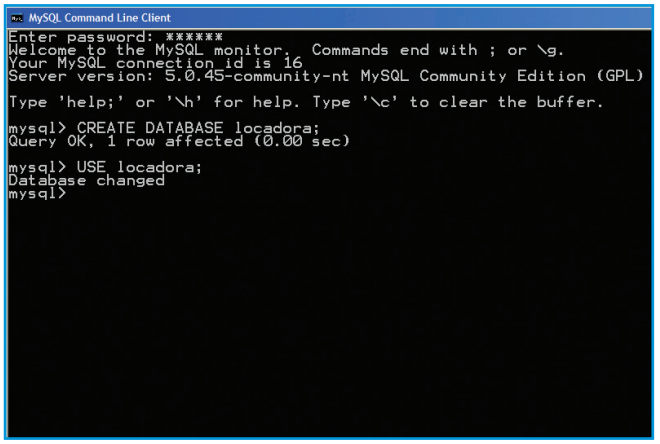
E se fosse possível que o banco de dados retornasse quaisquer números que você gravasse como decimais? E se, para isso, não fosse necessária nenhuma programação extra? E se pudéssemos relacionar uma ou mais tabelas e o banco de dados as retornasse já mescladas com os valores referentes? Conhecendo um pouco mais a linguagem SQL, a programação não precisa ficar reinventando a roda.

Comece aprendendo os tipos de dados que o banco suporta. Se quiser guardar um número, escolha, preferencialmente, o tipo inteiro. Mas se esse número precisa de casas decimais, como um valor monetário, use o tipo de dado float que guarda os números com quantas casas decimais você preferir. Contudo, se esse valor for referente a um telefone ou CEP, na verdade, ele não é um numero, mas um endereçamento, e deve ser tratado como tal, ou seja, pertence ao tipo varchar. Portanto, um número só é um número quando você puder fazer cálculos matemáticos sobre ele.

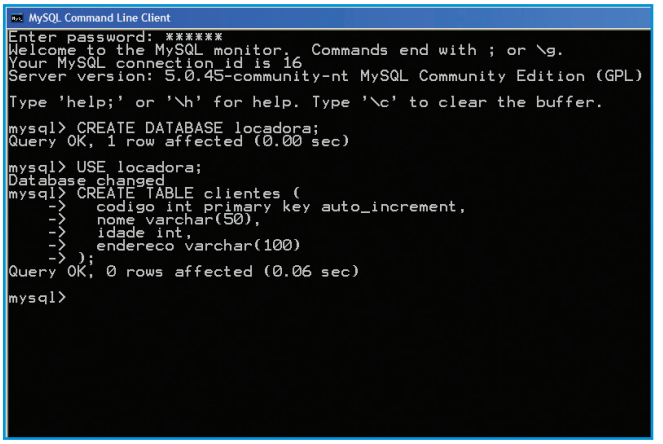
Listas, também conhecidas como ENUM, não guardam diversos valores ao mesmo tempo, elas têm um grupo de palavras ou expressões que são aceitas pelo banco. Se tentar inserir um número diferente, ele cancelará a inclusão do registro.

Data e hora são um problema para qualquer programador: ter que dividir os pedaços de uma data ou hora e analisar cada um em separado (dia, mês, ano), repartindo até chegar ao valor em segundos e diminuir um do outro para, então, reconverter e descobrir qual a diferença entre os dois períodos de tempo. O MySQL consegue fazer esses cálculos e retornar diferenças entre datas que já estão no banco de dados.

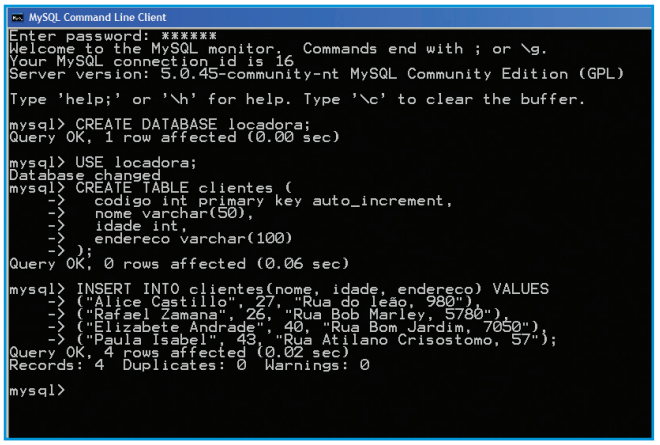
O próximo passo é aprender a mesclar as tabelas sem que seja necessário chamar uma tabela e a cada linha fazer um novo pedido ao banco de dados referente à outra tabela. Este recurso se chama JOIN e poupa os profissionais de programação.



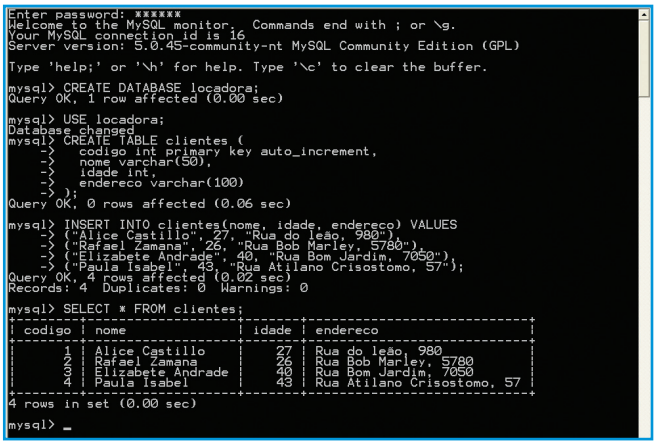
1 – Começamos o sistema criando o banco de dados "locadora" por meio do comando **CREATE DATABASE**. Logo depois, passe a utilizar esse banco com o comando **USE**.



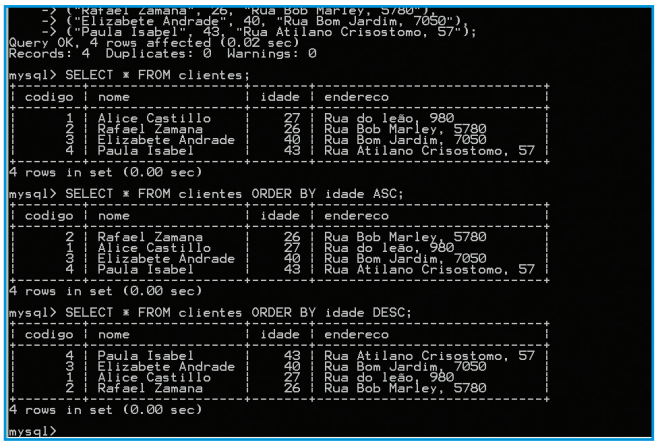
2 – Crie a tabela de clientes. Repare nos tipos de dados: código é inteiro, nome é pedaço de texto (varchar), idade é inteiro e endereço outro varchar maior. **Primary key** indica o campo-chave.



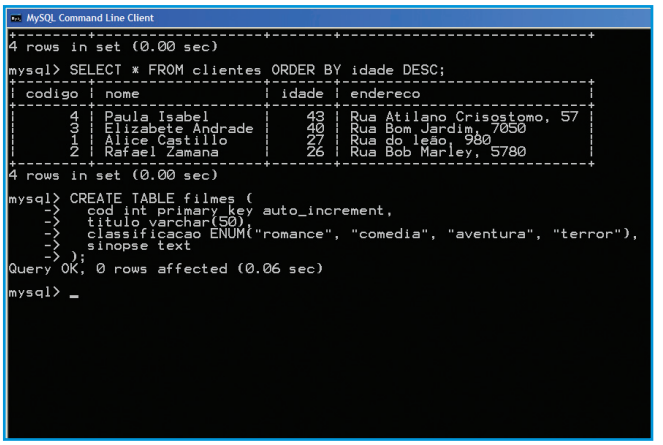
3 – Então, inserimos alguns clientes em nosso banco de dados. Note que a idade é o único valor que não precisa de aspas. Observe, também, que o campo código não foi colocado.



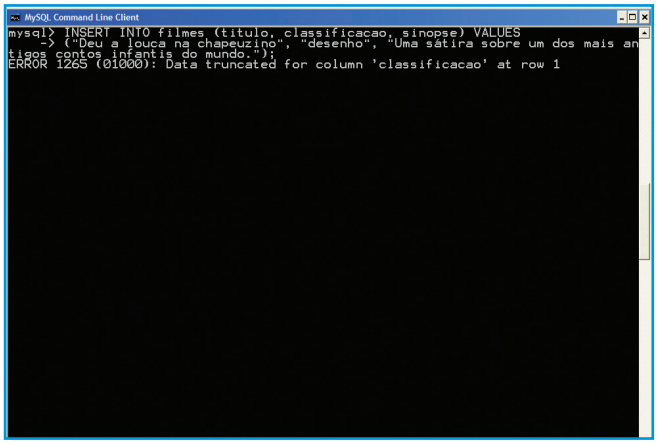
4 – Ao selecionarmos a tabela, note que o código inseriu valores incrementados automaticamente. Isso se deve ao fato de o campo possuir **auto_increment** em sua criação.



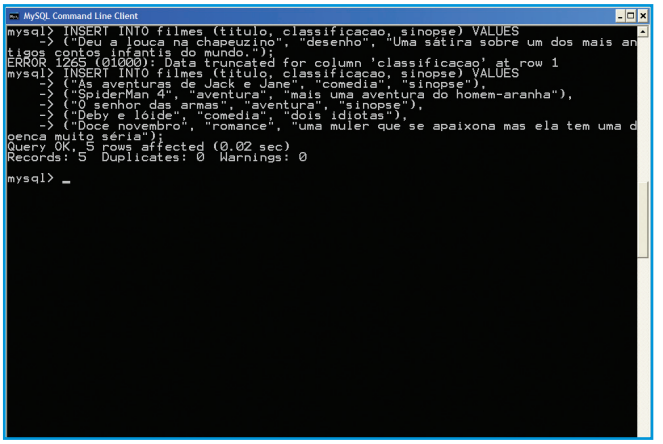
5 – Define-se a ordem na qual o sistema retornará os valores por meio da propriedade **ORDER BY [COLUNA]**. Você pode usar **ASC** para crescente e **DESC** para decrescente. **ASC** é a configuração-padrão.



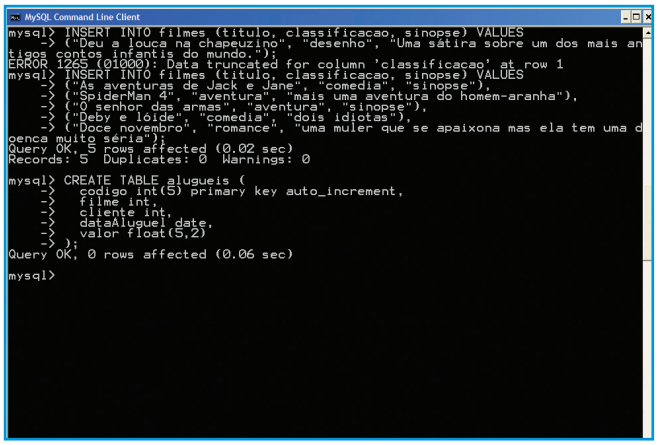
6 – Criemos, então, a segunda tabela: filmes, em que temos novamente o primary key **auto_increment**, um varchar e uma lista **ENUM** com algumas definições e a sinopse do tipo **TEXT**, para textos longos.



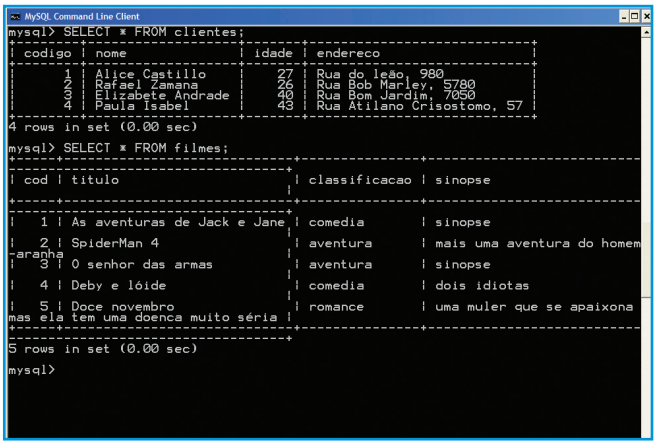
7 – Veja o que acontece quando tentamos inserir um campo diferente dos valores indicados em **ENUM**. A palavra "desenho" não foi definida como parte de classificação.



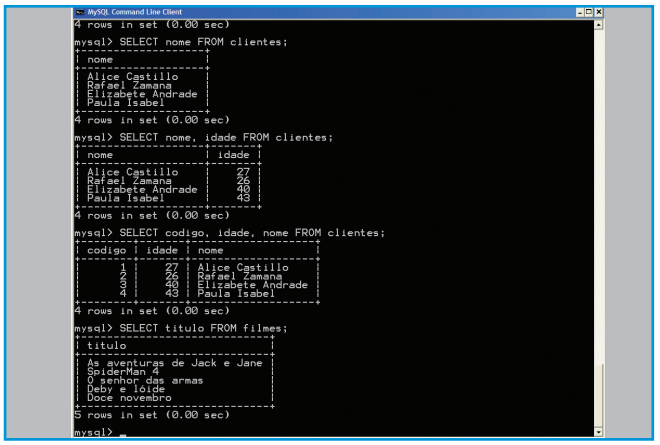
8 – No passo anterior, um erro ocorre. Dessa forma, inserimos alguns filmes. Repare que, novamente, não nos importamos com o código, já que ele é gerado de modo automático.



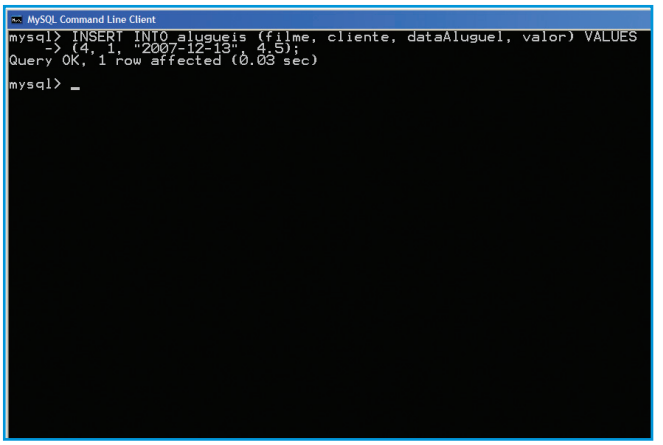
9 – Crie as tabelas relacionais. Tanto **cliente** como **filme** são do mesmo tipo dos campos **codigo** e **cod**, respectivamente. **DataAluguel** será do tipo **DATE**.



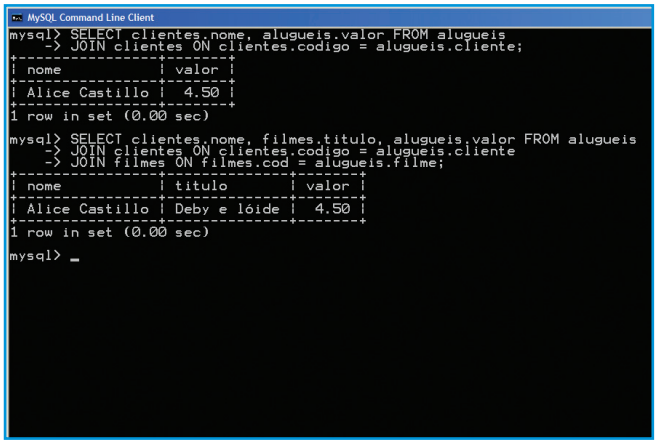
10 – Ainda sobre a figura anterior, **Valor** é do tipo **float** com até cinco dígitos e mais dois decimais. Inicie a instalação. A essa altura, sua configuração deveria ser feita no bloco de notas.



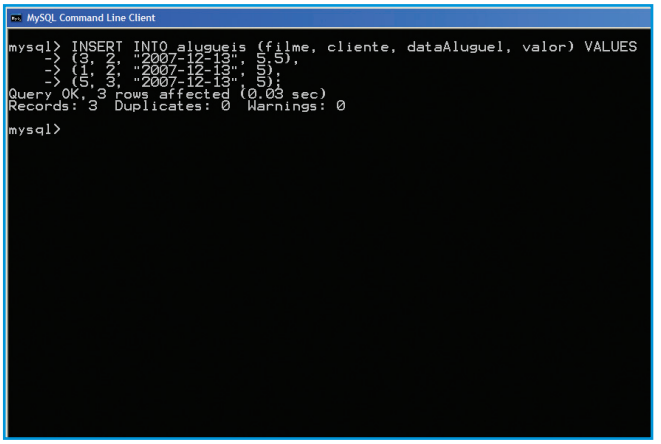
11 – Se você quiser resolver esse problema de forma simples e eficiente, selecione apenas os campos que deseja visualizar logo após a expressão **SELECT**.



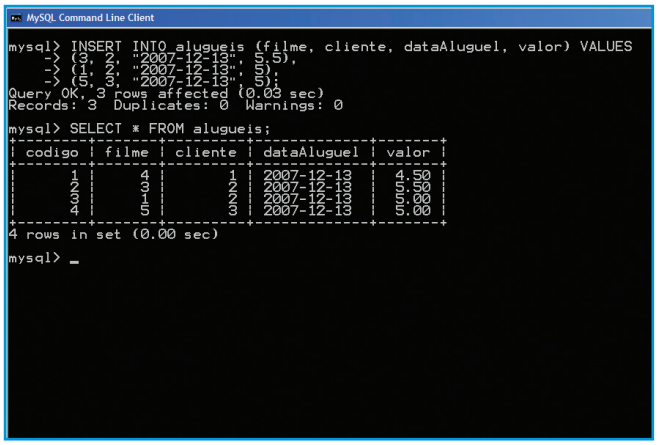
12 – Insira um campo em **aluguéis** para seguir adiante. O campo **date** segue o formato americano. **Valor** foi definido para funcinar com ponto antes dos decimais e com apenas uma casa decimal.



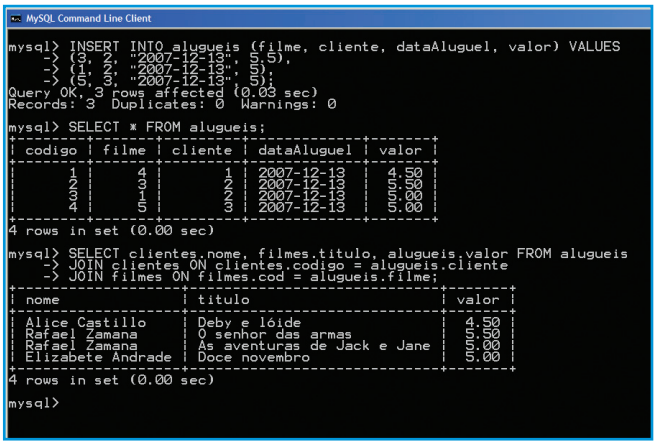
13 – Fazer um join é muito simples, utilize a seguinte sintaxe: **JOIN [TABELA] ON [COMPARAÇÃO ENTRE AS TABELAS]**. Os camos mostrados são definidos pela sintaxe **[TABELA].[COLUNA]**.



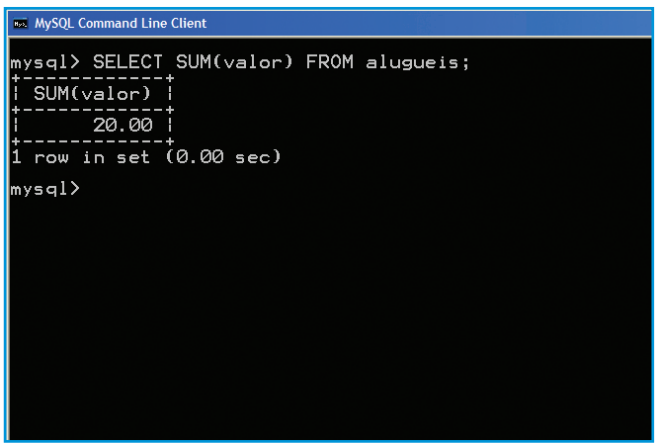
14 – Sobre a figura do passo 13, a primeira comparação requer que a mesclagem seja feita onde o código de cliente for igual ao cliente de aluguéis. Na seqüência, um join com as três tabelas.



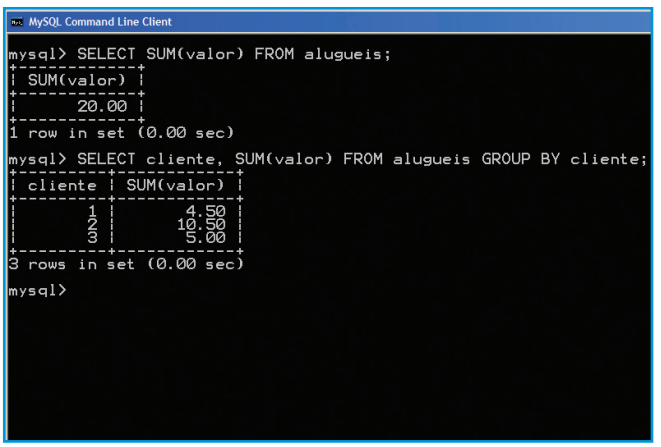
15 – Observe que a tabela em si não trabalha com nenhum texto, apenas faz referências às outras tabelas. Vamos seguir com as demais tabelas.



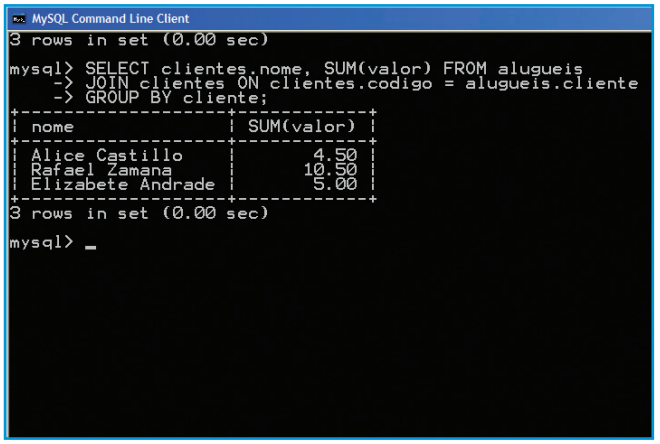
16 – Por fim, o join das três tabelas. Veja a facilidade em se trabalhar com campos de dados relacionais, o que traz grandes benefícios ao seu projeto.



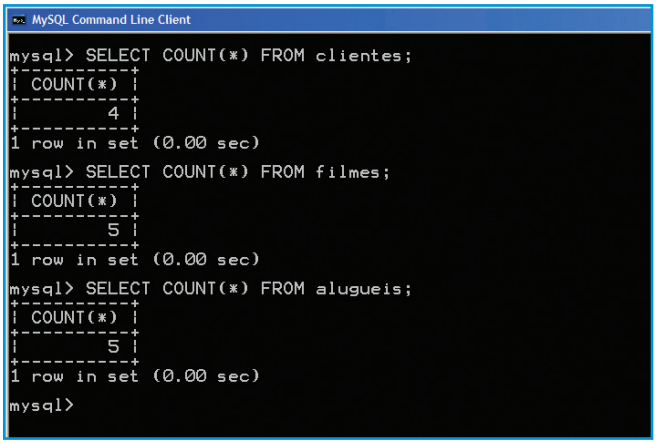
17 – Se quisermos selecionar a soma dos valores da tabela aluguéis, use **SUM(CAMPO)**. Esse tipo de dado pode ser obtido a qualquer momento.



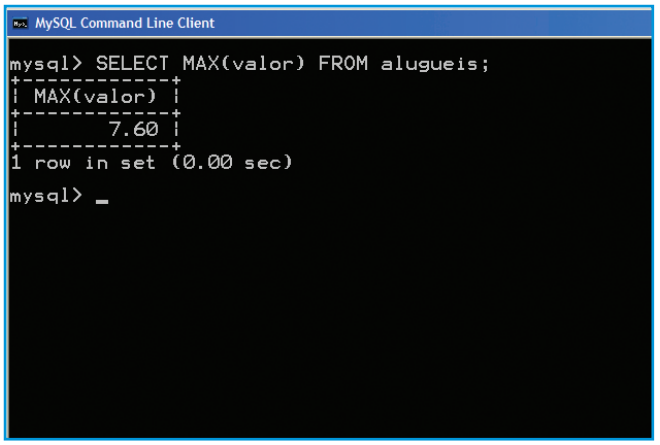
18 – Agora, com a propriedade **GROUP BY [CAMPO]**, saberemos quanto cada cliente deve pagar. Note que somente a tabela aluguéis foi selecionada.



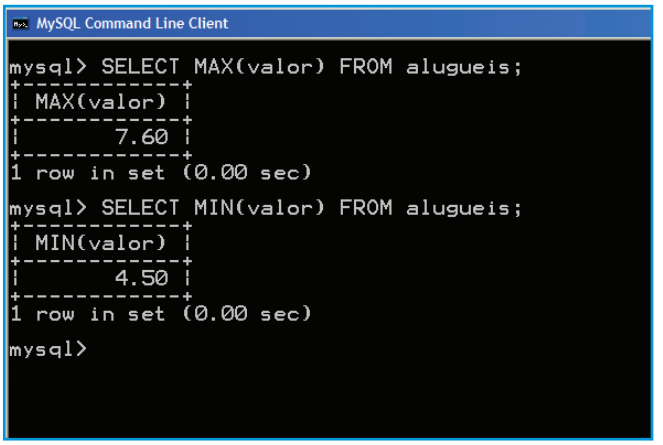
19 – Agora, inserimos o JOIN em meio ao exemplo anterior. Essa tarefa vai testar mais funcionalidades do banco de dados relacional que está sendo criado.



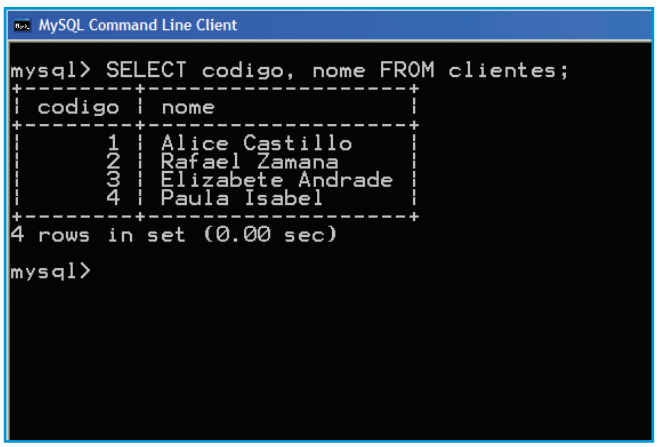
20 – Quantos filmes, clientes e locações a locadora possui? Com um simples **COUNT(*)** resolvemos o problema de modo muito simples.



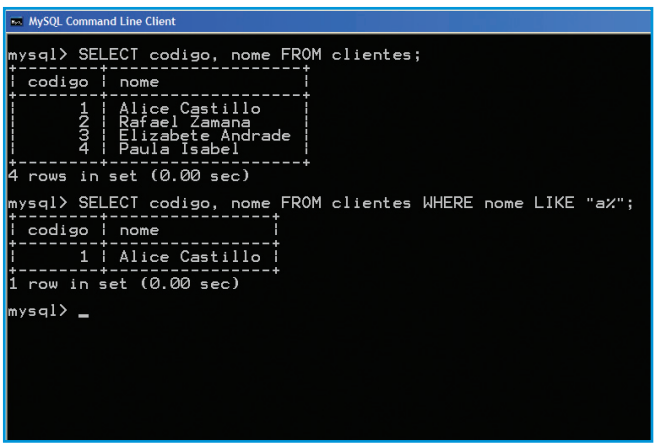
21 – Qual é o preço mais alto cobrado na locadora? Use **MAX(CAMPO)** para ter esse resultado. Outra tarefa muito simples de ser realizada.



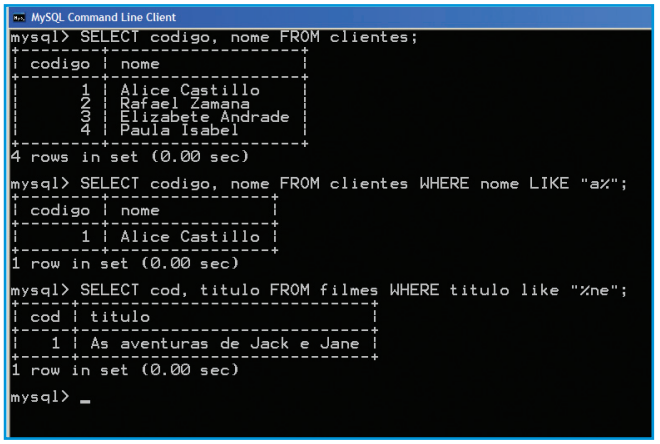
22 – O mesmo acontece quando usamos o argumento **MIN(CAMPO)** para valor mínimo. É um excelente modo de obter dados importantes para a locadora.



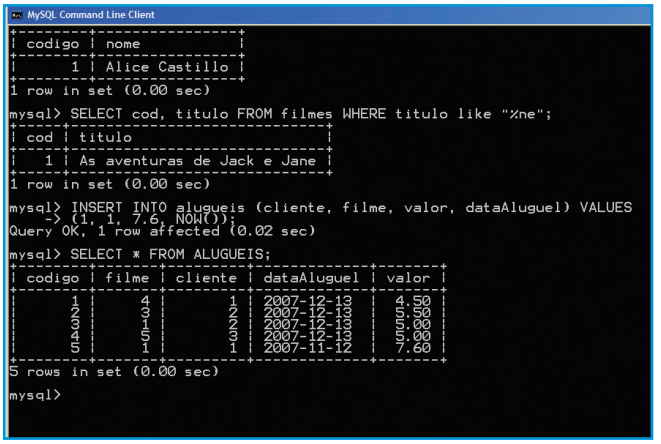
23 – O usuário não sabe o próprio codigo, mas não é nada prático procurá-lo dentro de um banco de dados extenso como o de uma locadora em situação real, com mais de 500 clientes.



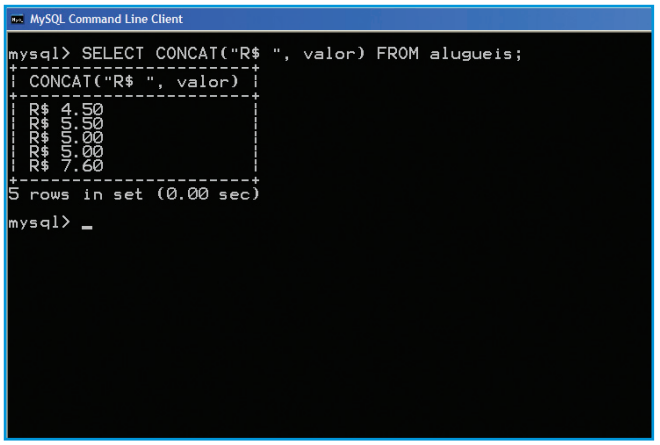
24 – Então, fazemos uma busca com o comando **LIKE**. Ele aceita determinados caracteres especiais, como %, que indica que qualquer espécie de texto pode estar no lugar.



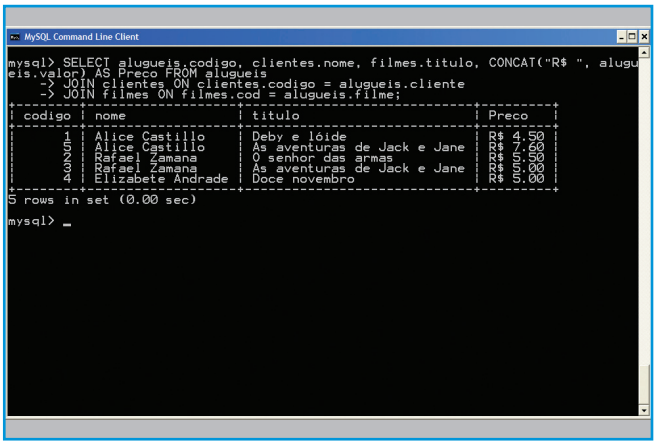
25 – Faça o mesmo para filmes, agora, ache algum que termine com "de". Você pode buscar **%andrade%** para procurar alguém que tenha Andrade em um de seus sobrenomes, por exemplo.



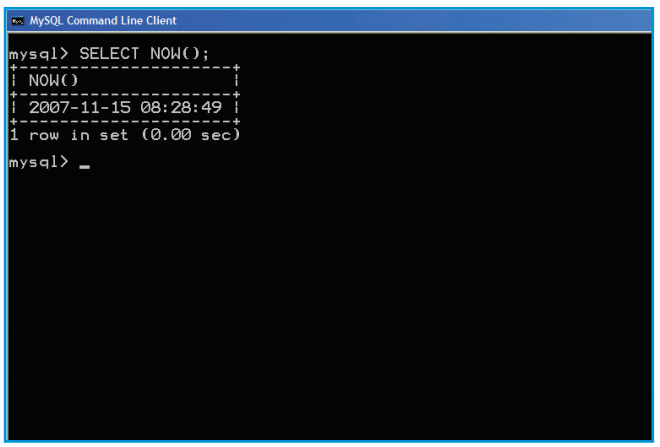
26 – Insira um novo filme para o cliente encontrado. Em vez de escrever a data na qual o filme foi alugado, foi usada a referência **NOW()**, que indica que o tempo a ser inserido é o da data atual.



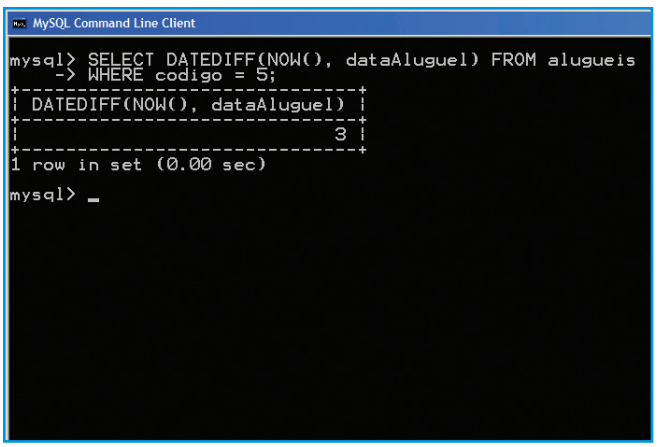
27 – Ao utilizar a função **CONTAT(VALOR1, VALOR2, VALOR3, VALORN)** é possível juntar um texto a outro, e se tal texto for um número, ele será convertido para texto.



28 – Agora, a versão final com o preço já convertido para reais e mesclado com as outras tabelas. Fica a seu critério colocar um filtro ou outros campos, como aluguel, que indiquem o *status* do filme.



29 – Ao alterar a data do meu sistema, o banco de dados receberá a nova data, logo, a função **NOW()** retorna uma nova data, como visto na imagem.



30 – Utilize a função DATEDIFF para saber qual a diferença entre duas datas. É uma forma de calcular eventuais multas por atraso.

Domine a língua MySQL

Uma abordagem teórica-prática sobre a linguagem de banco de dados mais usada no mundo

por Michael Granados

Organizar e agrupar parecem ser tendências naturais do ser humano. Sempre procuramos manter a ordem e o equilíbrio sobre todas as coisas ao nosso redor. Essa técnica é utilizada em quase todos os lugares por onde passamos. Olhe em sua volta: mesmo que não queira você acaba de agrupar alguns elementos por cor, forma ou tamanho. Assim como os objetos físicos, as informações também podem ser agrupadas. Uma dona de casa procura em seu caderno o prato que mais possa agradar seus convidados. O caderno é um banco cheio de informações úteis. Esse é um belo exemplo de banco de dados, uma vez que informações são conhecidas como dados, em informática.

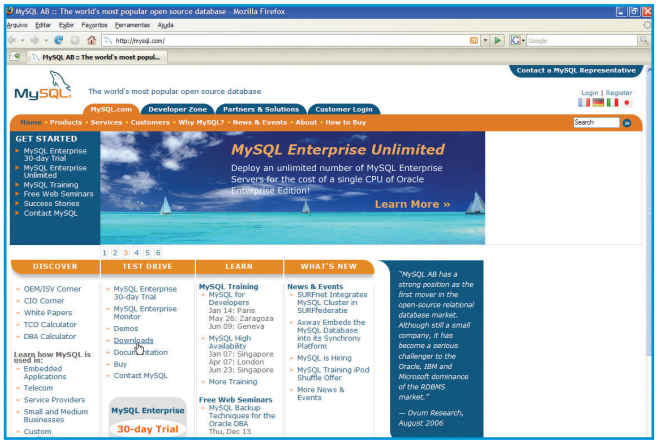
Contudo, as informações (principalmente em grande escala) não têm valor, se não houver alguma maneira de interagir com os dados adicionando novos registros, contando quantos registros de determinado valor existem, definindo classificação, como ordem alfabética ou filtrando dados por um critério. Essas funcionalidades, que nos ajudam bastante no dia-a-dia, ficam a cargo dos sistemas de bancos de dados.

Claro que não existe apenas um tipo de sistema de banco de dados, diversas empresas produzem inúmeros bancos de dados e disponibilizam, vendem ou mesmo alugam os sistemas. Os sistemas, em sua maioria, trabalham com uma linguagem em comum: SQL. Essa linguagem é a mais usada por ser simples e de fácil aprendizado. Além disso, os comandos dela são muito humanizados, o que a torna tão popular. Mesmo quem não entende nada de informática pode compreender facilmente certos comandos, como “SELECT * FROM vendas” ou, em português, “selecione tudo das vendas”.

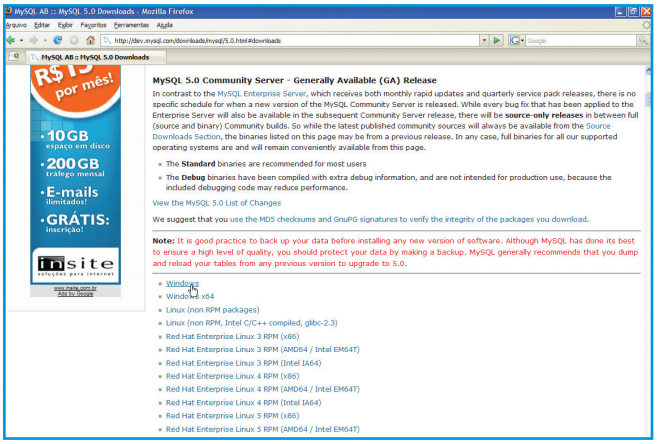
Um dos bancos de dados mais usados no mundo é o MySQL, que ganhou força por meio da Internet por se integrar a várias linguagens de programação, como ASP, PHP, Perl ou Python. O MySQL também se comporta extremamente bem quando o assunto é portabilidade. Ele funciona na maioria dos sistemas operacionais, como Linux, Windows, MacOS, entre outros. Também adquiriu preferência por ser um sistema livre para desenvolvedores.

Geralmente, o MySQL é usado para sistemas na Internet em conjunto com a linguagem de programação PHP no servidor Apache rodando na plataforma Linux. Esse time de ferramentas é conhecido como LAMP (Linux + Apache + MySQL + PHP). Sistemas de grande porte foram construídos com o LAMP, por exemplo, a Wikipédia. Mas isso não impede que o MySQL funcione em desktop também com o Delphi ou o VB por exemplo, além de rodar perfeitamente sem o auxílio de qualquer interpretador.

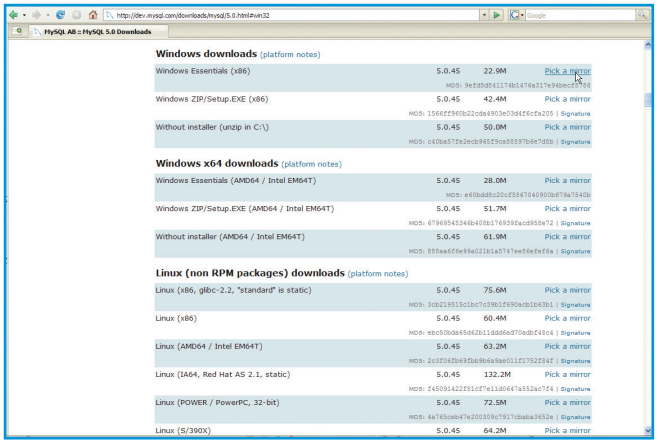
Neste tutorial, você vai aprender a instalar, configurar e dar seus primeiros passos com o MySQL, criar um banco de dados, uma tabela e inserir e selecionar dados da mesma. Após ler a revista, você perceberá que não há motivo para depender de programas que fazem tudo para facilitar e deixam código sujo com inúmeras funções que talvez não sejam úteis, porque você só precisa de ações simples. Aprendendo a linguagem do MySQL naturalmente, você também aprende muitas funcionalidades de outros bancos de dados como o Postgree ou SQLite, já que todas essas linguagens se baseiam no padrão SQL. Então, arregace as mangas e mãos à obra!



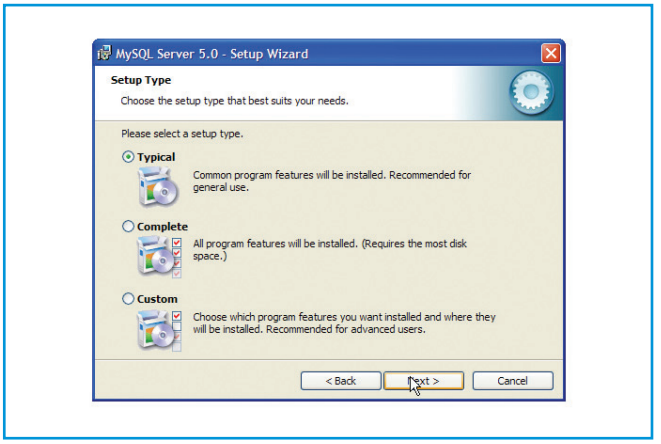
1 – Vá até o site do MySQL (*www.mysql.com*) para efetuar o download do sistema de banco de dados que será instalado no servidor.



2 – Na área de download, escolha seu sistema operacional para que o banco de dados se adapte perfeitamente ao local em que será instalado.

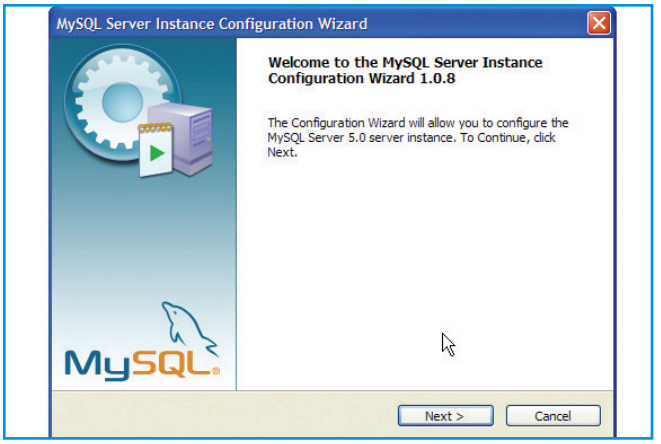


3 – Para Windows, existem três versões: apenas o essencial, completa ou sem instalador, para que você possa fazer a instalação na unha. Este tutorial utiliza a primeira versão.



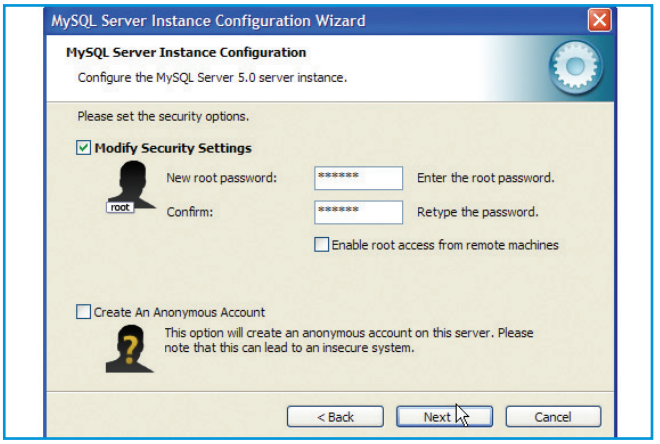
4 – Inicie a instalação normalmente e use as configurações-padrão. A instalação do MySQL era complicada e sua configuração costumava ser feita no bloco de notas. Agora, tudo é mais fácil.

5 – Ao final da instalação, o sistema vai perguntar se você deseja configurar o banco de dados. Deixe essa caixa marcada e conclua a instalação.

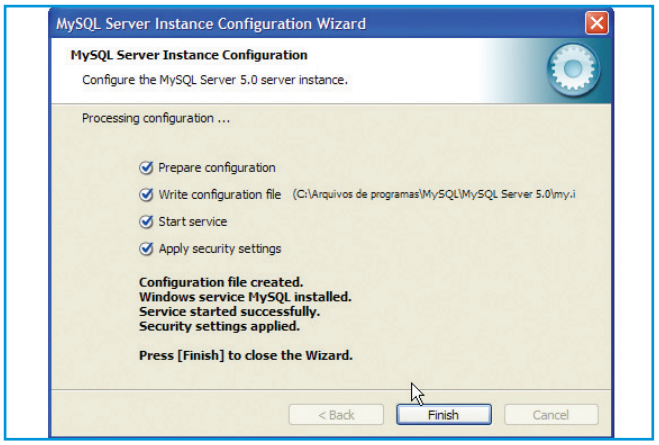


6 – Automaticamente, o instalador lançará o configurador do MySQL. Veja, na imagem, a tela que você deve visualizar a esta altura da atividade.

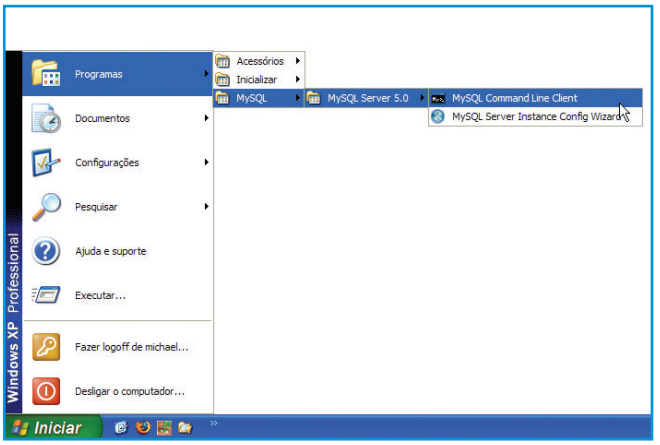
7 – Nesta tela, você configura seu sistema operacional para utilizar o MySQL como serviço, deixando como opção que ele se inicie junto ao Windows.



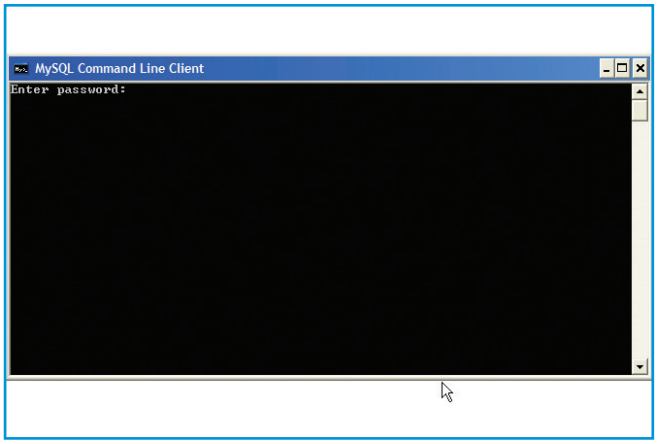
8 – Defina uma senha para que o usuário root possa utilizar o banco de dados. Em sistemas de informática, root é o usuário principal que tem o controle sobre tudo o que ocorre no sistema.



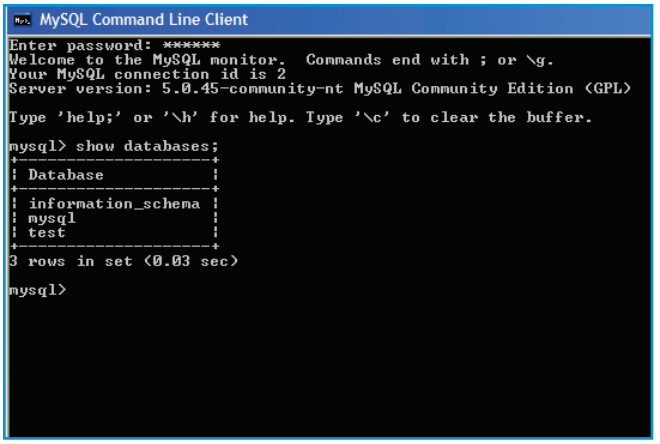
9 – Ao final, o programa demonstra seu relatório de processos: preparar a configuração; escrever o arquivo de configuração; iniciar o serviço e aplicando as configurações do banco de dados.



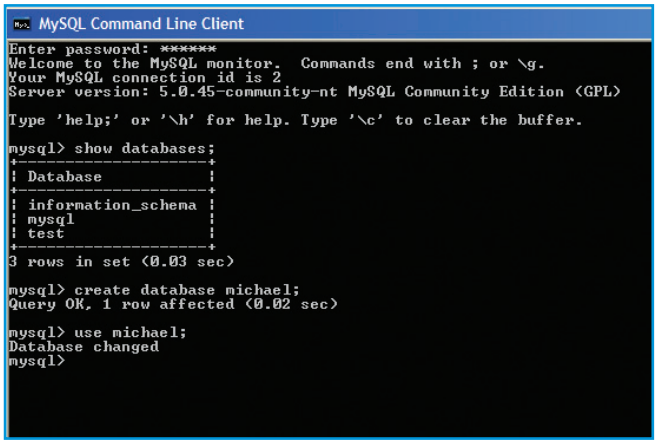
10 – Inicie o prompt de comando do MySQL. Perceba que, se você quiser configurar seu banco de dados novamente, poderá fazer isso com a ferramenta de configuração contida neste mesmo menu.



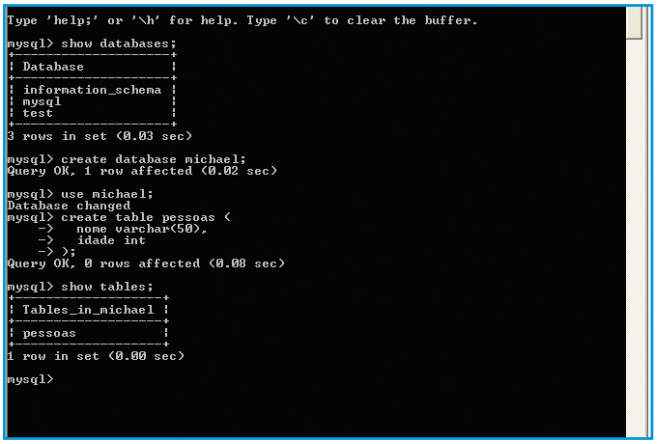
11 – A primeira coisa que você deve fazer para começar a brincar com o banco de dados é entrar com a senha do root. Lembra daquela senha alguns passos atrás?



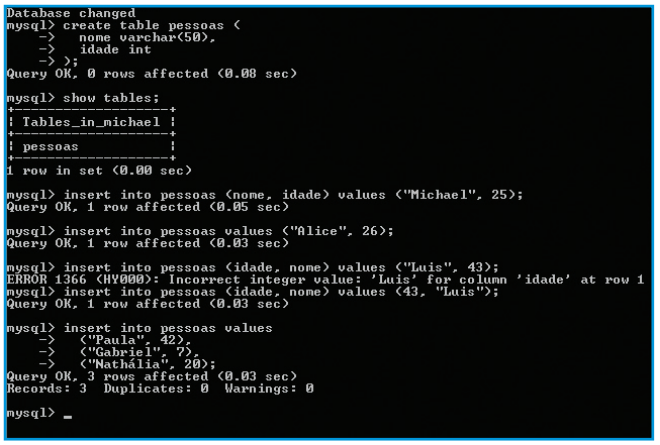
12 – Depois de fazer o login, use o comando **SHOW DATABASES** para ver quais bancos de dados é possível utilizar para fazer novos projetos.



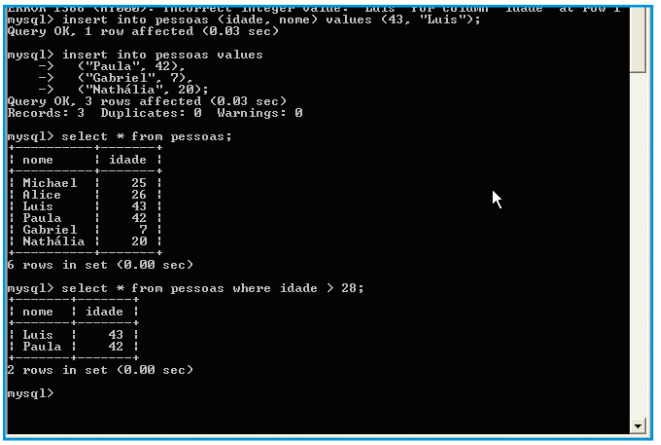
13 – Com os comandos **CREATE DATABASE** <NOME DO BANCO> e **USE** <NOME DO BANCO>, respectivamente, crie um novo banco e passe a usá-lo como referência para os outros passos.



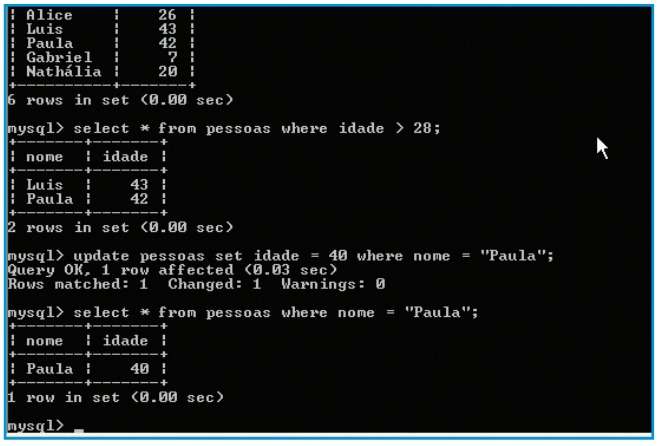
14 – Com o **CREATE TABLE**, crie uma tabela de pessoas com suporte a dois fatores: nome e idade. Veja que para cada tipo de informação, utiliza-se um atributo.



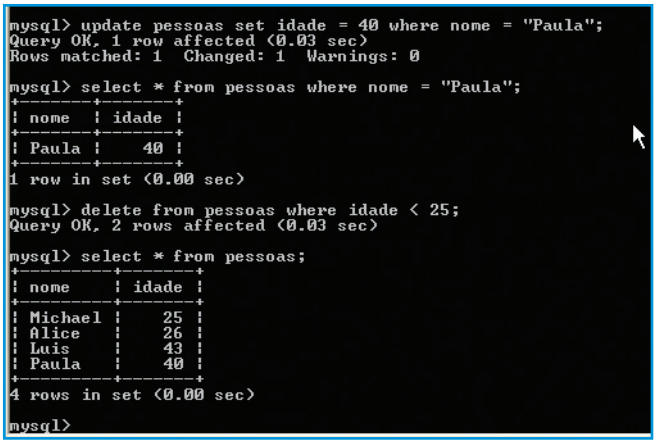
15 – Insira dados nesse banco. Veja que ocorreu um erro ao tentarmos alocar um texto no campo de valor inteiro. Logo, a escolha do tipo de dado é importante.



16 – Selecione as pessoas com o comando **SELECT** para saber quais delas existem na tabela. Na seqüência, foi usado um filtro chamado **WHERE**, para verificar as pessoas acima de 28 anos.



17 – Para atualizar alguma informação, use o comando **UPDATE**, configurando valores com a diretiva **SET.** Especifique onde acontecerá essa atualização no banco de dados com o filtro **WHERE**.



18 – Para excluir dados, utilize o comando **DELETE** em conjunto com o filtro **WHERE** e o **UPDATE**. Para apagar tudo, use os comando "**DROP TABLE pessoa**s" e "**DROP DATABASE michael**".