MySQL

18 de abril de 2007

Sumário

Parte I

Sobre essa Apostila

Conteúdo

O conteúdo dessa apostila é fruto da compilação de diversos materiais livres publicados na internet, disponíveis em diversos sites ou originalmente produzido no CDTC em *http://www.cdtc.org.br*.

O formato original deste material bem como sua atualização está disponível dentro da licença *GNU Free Documentation License*, cujo teor integral encontra-se aqui reproduzido na seção de mesmo nome, tendo inclusive uma versão traduzida (não oficial).

A revisão e alteração vem sendo realizada pelo CDTC (*suporte@cdtc.org.br*) desde outubro de 2006. Críticas e sugestões construtivas são bem-vindas a qualquer tempo.

Autores

A autoria deste é de responsabilidade de Joao Paulo Claudino de Souza.

O texto original faz parte do projeto Centro de Difusão de Tecnologia e Conhecimento, que vem sendo realizado pelo ITI (Instituto Nacional de Tecnologia da Informação) em conjunto com outros parceiros institucionais, atuando em conjunto com as universidades federais brasileiras que tem produzido e utilizado Software Livre, apoiando inclusive a comunidade Free Software junto a outras entidades no país.

Informações adicionais podem ser obtidas através do email *ouvidoria@cdtc.org.br*, ou da *home page* da entidade, através da URL **http://www.cdtc.org.br**.

Garantias

O material contido nesta apostila é isento de garantias e o seu uso é de inteira responsabilidade do usuário/leitor. Os autores, bem como o ITI e seus parceiros, não se responsabilizam direta ou indiretamente por qualquer prejuízo oriundo da utilização do material aqui contido.

Licença

Parte II

Informações Básicas

Sobre o CDTC

Objetivo Geral

O Projeto CDTC visa a promoção e o desenvolvimento de ações que incentivem a disseminação de soluções que utilizem padrões abertos e não proprietários de tecnologia, em proveito do desenvolvimento social, cultural, político, tecnológico e econômico da sociedade brasileira.

Objetivo Específico

Auxiliar o Governo Federal na implantação do plano nacional de software não-proprietário e de código fonte aberto, identificando e mobilizando grupos de formadores de opinião dentre os servidores públicos e agentes políticos da União Federal, estimulando e incentivando o mercado nacional a adotar novos modelos de negócio da tecnologia da informação e de novos negócios de comunicação com base em software não-proprietário e de código fonte aberto, oferecendo treinamento específico para técnicos, profissionais de suporte e funcionários públicos usuários, criando grupos de funcionários públicos que irão treinar outros funcionários públicos e atuar como incentivadores e defensores de produtos de software não proprietários e código fonte aberto, oferecendo conteúdo técnico on-line para serviços de suporte, ferramentas para desenvolvimento de produtos de software não proprietários e de seu código fonte livre, articulando redes de terceiros (dentro e fora do governo) fornecedoras de educação, pesquisa, desenvolvimento e teste de produtos de software livre.

Guia do aluno

Neste guia, você terá reunidas uma série de informações importantes para que você comece seu curso. São elas:

- Licenças para cópia de material disponível
- Os 10 mandamentos do aluno de Educação a Distância
- Como participar dos foruns e da wikipédia
- Primeiros passos

É muito importante que você entre em contato com TODAS estas informações, seguindo o roteiro acima.

Licença

Os 10 mandamentos do aluno de educação online

- 1. Acesso à Internet: ter endereço eletrônico, um provedor e um equipamento adequado é pré-requisito para a participação nos cursos a distância.
- 2. Habilidade e disposição para operar programas: ter conhecimentos básicos de Informática é necessário para poder executar as tarefas.
- 3. Vontade para aprender colaborativamente: interagir, ser participativo no ensino a distância conta muitos pontos, pois irá colaborar para o processo ensino-aprendizagem pessoal, dos colegas e dos professores.
- 4. Comportamentos compatíveis com a etiqueta: mostrar-se interessado em conhecer seus colegas de turma respeitando-os e fazendo ser respeitado pelo mesmo.
- 5. Organização pessoal: planejar e organizar tudo é fundamental para facilitar a sua revisão e a sua recuperação de materiais.
- 6. Vontade para realizar as atividades no tempo correto: anotar todas as suas obrigações e realizá-las em tempo real.
- 7. Curiosidade e abertura para inovações: aceitar novas idéias e inovar sempre.
- 8. Flexibilidade e adaptação: requisitos necessário à mudança tecnológica, aprendizagens e descobertas.
- 9. Objetividade em sua comunicação: comunicar-se de forma clara, breve e transparente é ponto - chave na comunicação pela Internet.
- 10. Responsabilidade: ser responsável por seu próprio aprendizado. O ambiente virtual não controla a sua dedicação, mas reflete os resultados do seu esforço e da sua colaboração.

Como participar dos fóruns e Wikipédia

Você tem um problema e precisa de ajuda?

Podemos te ajudar de 2 formas:

A primeira é o uso dos fóruns de notícias e de dúvidas gerais que se distinguem pelo uso:

. O fórum de notícias tem por objetivo disponibilizar um meio de acesso rápido a informações que sejam pertinentes ao curso (avisos, notícias). As mensagens postadas nele são enviadas a

todos participantes. Assim, se o monitor ou algum outro participante tiver uma informação que interesse ao grupo, favor postá-la aqui.
Porém, se o que você deseja é resolver alguma dúvida ou discutir algum tópico específico do curso. É recomendado que você faça uso do Forum de dúvidas gerais que lhe dá recursos mais efetivos para esta prática.

. O fórum de dúvidas gerais tem por objetivo disponibilizar um meio fácil, rápido e interativo para solucionar suas dúvidas e trocar experiências. As mensagens postadas nele são enviadas a todos participantes do curso. Assim, fica muito mais fácil obter respostas, já que todos podem ajudar.
Se você receber uma mensagem com algum tópico que saiba responder, não se preocupe com a formalização ou a gramática. Responda! E não se esqueça de que antes de abrir um novo tópico é recomendável ver se a sua pergunta já foi feita por outro participante.

A segunda forma se dá pelas Wikis:

. Uma wiki é uma página web que pode ser editada colaborativamente, ou seja, qualquer participante pode inserir, editar, apagar textos. As versões antigas vão sendo arquivadas e podem ser recuperadas a qualquer momento que um dos participantes o desejar. Assim, ela oferece um ótimo suporte a processos de aprendizagem colaborativa. A maior wiki na web é o site "Wikipédia", uma experiência grandiosa de construção de uma enciclopédia de forma colaborativa, por pessoas de todas as partes do mundo. Acesse-a em português pelos links:

- Página principal da Wiki - http://pt.wikipedia.org/wiki/

Agradecemos antecipadamente a sua colaboração com a aprendizagem do grupo!

Primeiros Passos

Para uma melhor aprendizagem é recomendável que você siga os seguintes passos:

- Ler o Plano de Ensino e entender a que seu curso se dispõe a ensinar;
- Ler a Ambientação do Moodle para aprender a navegar neste ambiente e se utilizar das ferramentas básicas do mesmo;
- Entrar nas lições seguindo a seqüência descrita no Plano de Ensino;
- Qualquer dúvida, reporte ao Fórum de Dúvidas Gerais.

Perfil do Tutor

Segue-se uma descrição do tutor ideal, baseada no feedback de alunos e de tutores.

O tutor ideal é um modelo de excelência: é consistente, justo e profissional nos respectivos valores e atitudes, incentiva mas é honesto, imparcial, amável, positivo, respeitador, aceita as idéias dos estudantes, é paciente, pessoal, tolerante, apreciativo, compreensivo e pronto a ajudar.

A classificação por um tutor desta natureza proporciona o melhor feedback possível, é crucial, e, para a maior parte dos alunos, constitui o ponto central do processo de aprendizagem.' Este tutor ou instrutor:

- fornece explicações claras acerca do que ele espera, e do estilo de classificação que irá utilizar;
- gosta que lhe façam perguntas adicionais;
- identifica as nossas falhas, mas corrige-as amavelmente', diz um estudante, 'e explica porque motivo a classificação foi ou não foi atribuída';
- tece comentários completos e construtivos, mas de forma agradável (em contraste com um reparo de um estudante: 'os comentários deixam-nos com uma sensação de crítica, de ameaça e de nervossismo')
- dá uma ajuda complementar para encorajar um estudante em dificuldade;
- esclarece pontos que não foram entendidos, ou corretamente aprendidos anteriormente;
- ajuda o estudante a alcançar os seus objetivos;
- é flexível quando necessário;
- mostra um interesse genuíno em motivar os alunos (mesmo os principiantes e, por isso, talvez numa fase menos interessante para o tutor);
- escreve todas as correções de forma legível e com um nível de pormenorização adequado;
- acima de tudo, devolve os trabalhos rapidamente;

Parte III

MySQL

Capítulo 1

O que é o Mysql

O Programa MySQL é um sistema de gerenciamento de banco de dados relacionais baseado em comandos SQL (Structured Query Language - Linguagem Estruturada para Pesquisas) que vem ganhando grande popularidade sendo atualmente um dos bancos de dados mais populares, com mais de 4 milhões de instalações.

Capítulo 2

Plano de ensino

2.1 Objetivo

Qualificar técnicos e programadores no SGBD MySQL.

2.2 Público Alvo

Técnicos e Programadores que desejam trabalhar com MySQL.

2.3 Pré-requisitos

Os usuários deverão ser, necessariamente, indicados por empresas públicas e ter conhecimento básico acerca da lógica de programação.

2.4 Descrição

O curso de Banco de Dados MySQL será realizado na modalidade EAD e utilizará a plataforma Moodle como ferramenta de aprendizagem. Ele é composto de cinco Módulos. O material didático estará disponível on-line de acordo com as datas pré-estabelecidas no calendário. A versão utilizada para o MySQL será a 5.0.

O curso está dividido da seguinte maneira:

Duração	Descrição do Módulo
1 semana	Lição 1 - O que é o MySQL Lição 2 - Instalando o MySQL Lição 3 - Aplicabilidade e uso Lição 4 - Criando e selecionando um banco de dados Lição 5 - Criando e manipulando tabelas Lição 6 - Consultando uma tabela Lição 7 - Cláusula ORDER BY Lição 8 - Funções pré-definidas
2 semana	Lição 9 - Conectando e desconectando do Servidor Lição 10 - Adicionando novos usuários ao MySQL Lição 11 - Ferramentas gráficas de administração Lição 12 - Stored procedures e funções

Todo o material está no formato de lições, e estará disponível ao longo do curso. As lições poderão ser acessadas quantas vezes forem necessárias. Aconselhamos a leitura de "Ambientação do Moodle", para que você conheça o produto de Ensino a Distância, evitando dificuldades advindas do "desconhecimento"sobre o mesmo.

Ao final de cada semana do curso será disponibilizada a prova referente ao módulo estudado anteriormente que também conterá perguntas sobre os textos indicados. Utilize o material de cada semana e os exemplos disponibilizados para se preparar para prova.

Os instrutores estarão a sua disposição ao longo de todo curso. Qualquer dúvida deve ser disponibilizada no fórum ou enviada por e-mail. Diariamente os monitores darão respostas e esclarecimentos.

2.5 Metodologia

O curso está dividido da seguinte maneira:

2.6 Cronograma

- Lição 1 - O que é o MySQL
- Lição 2 - Instalando o MySQL
- Lição 3 - Aplicabilidade e uso
- Lição 4 - Criando e selecionando um banco de dados
- Lição 5 - Criando e manipulando tabelas
- Lição 6 - Consultando uma tabela
- Lição 7 - Cláusula ORDER BY
- Lição 8 - Funções pré-definidas
- Lição 9 - Conectando e desconectando do Servidor
- Lição 10 - Adicionando novos usuários ao MySQL

- Lição 11 - Ferramentas gráficas de administração
- Lição 12 - Stored procedures e funções
- Avaliação de aprendizagem
- Avaliação do curso

As lições contém o contéudo principal. Elas poderão ser acessadas quantas vezes forem necessárias, desde que esteja dentro da semana programada. Ao final de uma lição, você receberá uma nota de acordo com o seu desempenho. Responda com atenção às perguntas de cada lição, pois elas serão consideradas na sua nota final. Caso sua nota numa determinada lição for menor do que 6.0, sugerimos que você faça novamente esta lição.
Ao final do curso será disponibilizada a avaliação referente ao curso. Tanto as notas das lições quanto a da avaliação serão consideradas para a nota final. Todos os módulos ficarão visíveis para que possam ser consultados durante a avaliação final.
Aconselhamos a leitura da "Ambientação do Moodle"para que você conheça a plataforma de Ensino a Distância, evitando dificuldades advindas do "desconhecimento"sobre a mesma.
Os instrutores estarão a sua disposição ao longo de todo curso. Qualquer dúvida deverá ser enviada no fórum. Diariamente os monitores darão respostas e esclarecimentos.

2.7 Programa

O curso de Mysql oferecerá o seguinte conteúdo:

- O que é o MySQL
- Instalando o MySQL
- Aplicabilidade e uso
- Criando e selecionando um banco de dados
- Criando e manipulando tabelas
- Consultando uma tabela
- Cláusula ORDER BY
- Funções pré-definidas
- Conectando e desconectando do Servidor
- Adicionando novos usuários ao MySQL
- Ferramentas gráficas de administração
- Stored procedures e funções

2.8 Avaliação

Toda a avaliação será feita on-line.
Aspectos a serem considerados na avaliação:

- Iniciativa e autonomia no processo de aprendizagem e de produção de conhecimento;
- Capacidade de pesquisa e abordagem criativa na solução dos problemas apresentados.

Instrumentos de avaliação:

- Participação ativa nas atividades programadas.
- Avaliação ao final do curso.
- O participante fará várias avaliações referentes ao conteúdo do curso. Para a aprovação e obtenção do certificado o participante deverá obter nota final maior ou igual a 6.0 de acordo com a fórmula abaixo:
- Nota Final = ((ML x 7) + (AF x 3)) / 10 = Média aritmética das lições
- AF = Avaliações

2.9 Bibliografia

- Site official: www.mysql.org
- Guia em Português: http://dev.mysql.com/doc/

Capítulo 3

Mysql

3.1 Visão Geral

O Programa MySQL é um sistema de gerenciamento de banco de dados relacionais baseado em comandos SQL (Structured Query Language - Linguagem Estruturada para Pesquisas) que vem ganhando grande popularidade sendo atualmente um dos bancos de dados mais populares, com mais de 4 milhões de instalações.

3.2 Banco de Dados

3.2.1 Conceitos Fundamentais sobre Banco de Dados

Caso você não tenha tido contato com banco de dados anteriormente, é de fundamental importância a leitura deste livro. Os assuntos abordados vão desde os conceitos sobre banco de dados até normalização. Conceitos importantes para compreensão de relacionamentos, como chaves, também são abordados.

3.2.2 Introdução ao Banco de Dados

O que é um Banco de Dados

Um Banco de Dados é um recurso para a manipulação eficiente de um grande conjunto de informações estruturadas e armazenadas de forma organizada e integrada.

Imagine se fosse necessário encontrar o telefone de determinada pessoa, e que, para isso, não existisse a lista telefônica. Seria muito difícil encontrar o telefone. A lista telefônica, portanto, é um banco de dados onde estão relacionados os nomes e os telefones das pessoas.

Outros exemplos de banco de dados são:

* Lista com acervo de uma biblioteca;
* Lista de preços de uma farmácia;
* Lista de funcionários de uma empresa;

SGBD

Um Sistema Gerenciador de Banco de Dados, é o conjunto de programas (softwares) que compõe a camada responsável pelo armazenamento, e recuperação dos dados no Sistema de Informação. O objetivo principal é retirar da camada da Aplicação a responsabilidade dessas tarefas provendo um ambiente mais seguro, mais fácil de manter-se e mais confiável. A interface entre essas duas camadas é a uma linguagem padrão para consulta, manipulação e controle de acesso aos dados. Atualmente a linguagem mais utilizada para essa interface é o SQL - Structured Query Language.
Exemplos:

* Oracle
* Postgre
* MySQL

Vantagens de um SGBD:

* Rapidez na manipulação e no acesso à informação.
* Controle integrado de informações distribuídas fisicamente.
* Redução de redundância e de inconsistência de informações.
* Compartilhamento de dados.
* Segurança dos dados.
* Redução de problemas de integridade.

Tabelas

Armazena um conjunto de informações referentes a um determinado assunto. (Conjunto de Linhas e Colunas)

Exemplos:

Usuários

cod.	nome	rua	núm	...

Livros

cod	título	editora	ano

Empréstimos

cod_usu.	cod_livro	data_ret	data_dev

Linha (Tupla)

É o elemento do conjunto de uma entidade (Tabela). Cada linha de um Banco de Dados, formada por um conjunto de colunas, representa um registro (ou tupla). Os registros não precisam necessariamente conter dados em todas as colunas, os seus valores podem ser nulos.

Exemplos:

Usuários

cod.	nome	rua	núm	...

Livros

cod	título	editora	ano

Empréstimos

cod_usu.	cod_livro	data_ret	data_dev

Coluna

São dados elementares que descrevem a entidade. É um elemento de uma tupla.

Exemplos:

Usuários

cod.	nome	rua	núm	...

Livros

cod	título	editora	ano

Empréstimos

cod_usu.	cod_livro	data_ret	data_dev

3.2.3 Modelo Entidade Relacionamento

Modelo de Dados

Modelo de Dados é uma imagem gráfica de toda a base de informações necessárias para um determinado empreendimento.

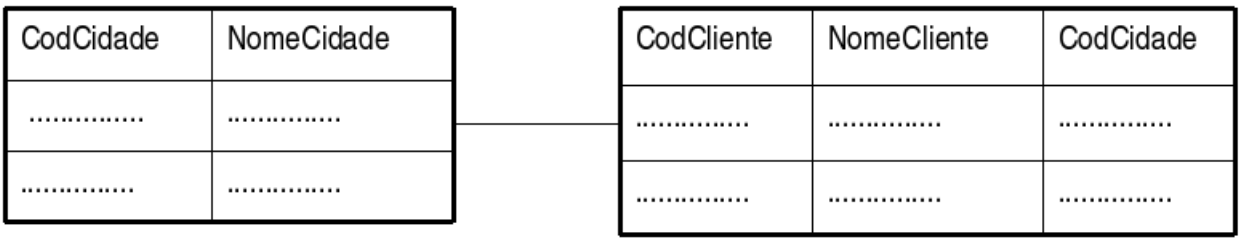
Modelo Relacional

O Modelo de Dados relacional representa os dados contidos em um Banco de Dados através de relações. Estas relações contém informações sobre as entidades representadas e seus relacionamentos.

Modelos Entidade Relacionamento

É a principal ferramenta gráfica para representação do Modelo de Dados e foi proposto por Peter Chain. Tem a finalidade de identificar entidades de dados e seus relacionamentos.

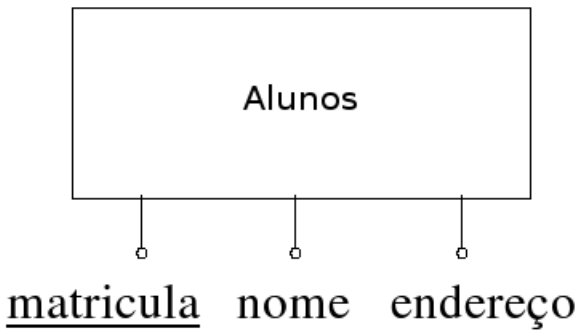
Exemplos:



Entidade

Conjunto de informações referentes a um determinado assunto.

Exemplos:



Atributos

Representam as características de uma Entidade.
Tipos de Atributos:

Determinante:
seu valor representa um elemento da entidade.
seu valor é único para a entidade.
deve ser sublinhado.

Composto:
necessita ser dividido em sub-atributos, para que seu significado seja melhor compreendido.

Multi-valorado:

pode assumir mais do que um valor para cada entidade, é diferenciado com um (*).

Recomendações para criação de um DER

Recomendações para criação de um DER((Diagrama Entidade - Relacionamento).

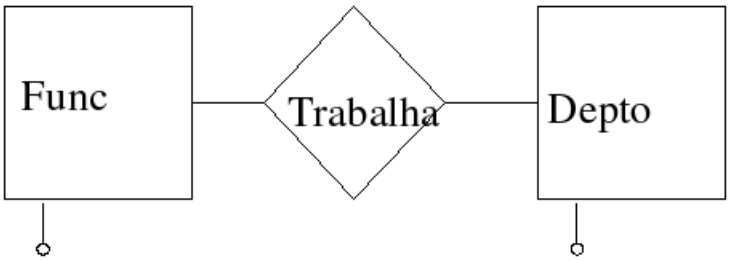
1. Antes de começar a modelar, conheça o "mundo real".
2. Identifique quais são as ENTIDADES.
3. Para cada Entidade represente seus ATRIBUTOS.
4. Confronte cada Entidade consigo mesma e com as demais na procura de possíveis RELACIONAMENTOS
5. Verifique a existência de ATRIBUTOS DE RELACIONAMENTO.
6. Para relacionamentos múltiplos estude a necessidade de AGREGAÇÕES.
7. Desenhe o DER, com todas as Entidades, Atributos, Relacionamentos, Classes e Restrições de Totalidade.
8. Analise cuidadosamente todas as restrições que você impôs.
9. Até que você e os seus usuários estejam convencidos de que o DER reflete fielmente o "mundo real", volte ao item 1.

3.2.4 Tipos de Relacionamento

Relacionamentos

São vínculos ou associações entre entidades

Exemplo:



Geralmente, a chave é um dos campos de um registro. O conceito de chave está também intimamente relacionado aos conceitos de índices e tabelas. Existem vários tipos de chave, por exemplo:

* **Uma chave é dita única** se ela identifica, de forma inequívoca, um determinado item armazenado no repositório. Uma chave não-única, ou ambígua, identifica mais de um item dentro do repositório.

* **Uma chave primária** é uma chave única, utilizada como referência preferencial para identificar um item dentro de um repositório. Por exemplo, o nome de um usuário de um sistema informatizado, ou o número de registro de um carro, podem ser utilizados como chaves primárias para localizar os respectivos itens.

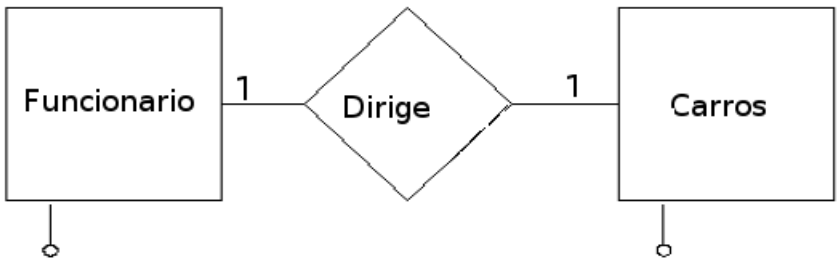
* **Uma chave estrangeira** é uma chave armazenada em um determinado repositório, que permite localizar informações contidas em outros repositórios. É um atributo ou conjunto de atributos de uma entidade que é chave primária em outra entidade

* **Uma chave simples** é associada a um único valor, ou campo, do registro. Uma chave composta corresponde ao valor agregado de vários campos, e pode ser necessária para eliminar a ambigüidade, formando um identificador único.

Tipos de Classes (Cardinalidades)

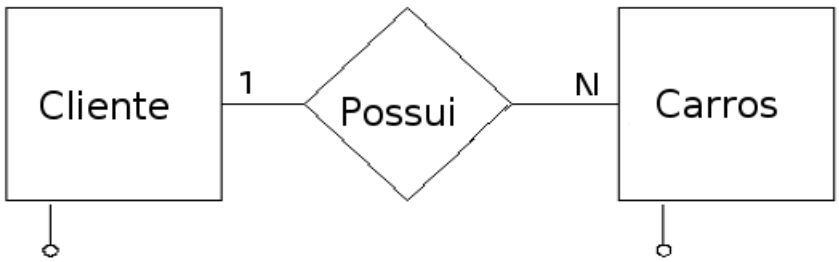
1:1

Em Bases de Dados relacionais, 1 para 1, one to one (um para um ou 1:1) é um dos tipos de relacionamentos que se podem estabelecer entre os campos de duas tabelas. Assim, para cada valor do campo de uma tabela, pode haver um valor no campo da outra tabela e vice-versa.



1:N
Em Bases de Dados relacionais, One to Many, um para muitos, 1 para N ou 1:N é um dos tipos de relacionamentos que se podem estabelecer entre os campos de duas tabelas, em que para cada valor do campo de uma tabela, pode haver N valores no campo da outra tabela.

Exemplos:
Um colégio pode ter várias turmas, mas a cada turma corresponde apenas um colégio.

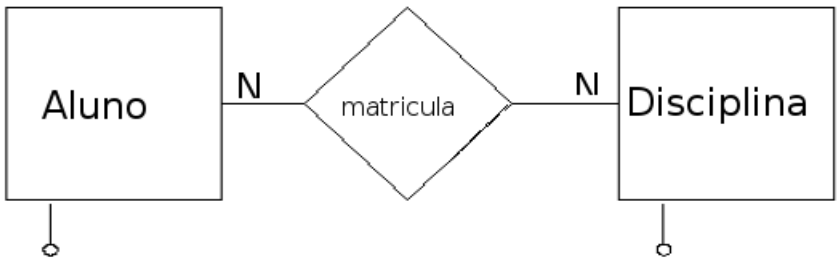


N:N
Em Bases de Dados relacionais, N para N, Many to Many (muitos para muitos ou N:N) é um dos tipos de relacionamentos que se podem estabelecer entre os campos de duas tabelas. Assim, para cada valor do campo de uma tabela, pode haver N valores no campo da outra tabela e vice-versa.

Em bancos de dados relacionais, isto normalmente é feito através de uma tabela de ligação. Ela se liga a cada lado N, com relacionamento 1 para N em cada lado. Cada possibilidade de ligação é registrada isoladamente, ou seja, ao relacionar cinco informações de cada lado, são registradas 25 linhas na tabela de ligação.

A chave primária desta tabela é criada pela junção dos campos chaves das tabelas interligadas.

Exemplos:
Um pedido de compra pode ter várias mercadorias, e um tipo de mercadoria pode ser listada em vários pedidos.



3.2.5 Normalização

Conceito de Normalização

Consiste em definir o formato lógico adequado para as estruturas de dados identificados no projeto lógico do sistema, com o objetivo de evitar redundância e garantir a integridade e confiabilidade das informações.

A normalização é feita, através da análise dos dados que compõem as estruturas utilizando o conceito chamado Formas normais(FN). As FN são conjuntos de restrições nos quais os dados devem ser submetidos.

PRIMEIRA FORMA NORMAL:

UMA RELAÇÃO ESTÁ NA 1FN SE NÃO TEM GRUPOS DE ATRIBUTOS OU ATRIBUTOS REPETITIVOS.

Exemplos:
Aluno (Matricula, Nome, Curso)
A entidade possui um atributo repetitivo, pois, se o aluno fizer mais de um curso será necessário cadastrar os dados Matricula e Nome novamente.

Fatura (nºfat, cod-cliente, nome-cliente, morada-cliente, cod-prod, descrição-prod, preço, quantidade)
A relação Fatura não se encontra na 1FN porque os atributos cod-prod, descrição-prod, preço e quantidade constituem um grupo repetitivo, ou seja, sempre que na mesma Fatura, existe informação referente a um novo produto é imprescindível repetir o nºfat, cod-cliente, nome-cliente, morada-cliente.

Passagem para a 1ªFN:

* Separar a relação Fatura em duas relações, Fatura e Linha-Fatura;
* Fatura (cod-fat, cod-cliente, nome-cliente, morada-cliente);
* Linha-Fatura (cod-fat, cod-prod, descrição-prod, preço, quantidade)

SEGUNDA FORMA NORMAL:

UMA RELAÇÃO ESTÁ NA 2FN SE ESTÁ NA 1FN E SE TODOS OS ATRIBUTOS NÃO CHAVE DEPENDEM DA TOTALIDADE DA CHAVE.

Exemplos:

* Fatura (cod-fat, cod-cliente, nome-cliente, morada-cliente)
* Linha-Fatura (cod-fat, cod-prod, descrição-prod, preço, quantidade)

A relação Linha-Fatura não se encontra na 2FN porque os atributos descrição-prod e preço não dependem de cod-fat, cod-prod mas só de cod-prod.

Passagem para a 2FN:

Separar a relação Linha-Fatura em duas relações, Linha-Fatura e Produto.

* Fatura (cod-fat, cod-cliente, nome-cliente, morada-cliente)
* Linha - Fatura (cod-fat, cod-prod, quantidade)
* Produto (cod-prod, descrição-prod, preço)

TERCEIRA FORMA NORMAL:

UMA RELAÇÃO ESTÁ NA 3FN SE ESTÁ NA 2FN E, TODOS OS SEUS ATRIBUTOS NÃO CHAVE NÃO SÃO IDENTIFICADOS POR UM OUTRO TAMBÉM NÃO CHAVE.
Exemplo:

* Fatura (cod-fat, cod-cliente, nome-cliente, morada-cliente)
* Linha - Fatura (cod-fat, cod-prod, quantidade)
* Produto (cod-prod, descrição-prod, preço)

A relação Fatura não se encontra na 3FN porque os atributos nome-cliente e morada-cliente são identificados por cod-cliente e não por cod-fat.

Passagem para a 3FN:
Separar a relação Fatura em duas relações, Fatura e Cliente.

* Fatura (cod-fact, cod-cliente)
* Cliente (cod-cliente, nome-cliente, morada-cliente)
* Linha - Fatura (cod-fat, cod-prod, quantidade)
* Produto (cod-prod, descrição-prod, preço)

3.3 Lição 1 - O que é o MySQL?

3.3.1 Introdução

O MySQL, um dos mais populares sistemas de gerenciamento de banco de dados SQL Open Source, é desenvolvido, distribuído e tem suporte da MySQL AB. A MySQL AB é uma empresa comercial, fundada pelos desenvolvedores do MySQL, cujos negócios é fornecer serviços relacionados ao sistema de gerenciamento de banco de dados MySQL.

Um banco de dados é uma coleção de dados estruturados. Ele pode ser qualquer coisa desde uma simples lista de compras a uma galeria de imagens ou a grande quantidade de informação da sua rede coorporativa. Para adicionar, acessar, e processar dados armazenados em um banco de dados de um computador, você necessita de um sistema de gerenciamento de bancos de dados como o Servidor MySQL. Como os computadores são muito bons em lidar com grandes quantidades de dados, o gerenciamento de bancos de dados funciona como a engrenagem central na computação, seja como utilitários independentes ou como partes de outras aplicações.

O MySQL é um sistema de gerenciamento de bancos de dados relacional. Um banco de dados relacional armazena dados em tabelas separadas em vez de colocar todos os dados num só local. Isso proporciona velocidade e flexibilidade. A parte SQL do "MySQL" atenta pela "Structured Query Language - Linguagem Estrutural de Consultas". SQL é a linguagem padrão mais

comum usada para acessar banco de dados e é definida pelo Padrão ANSI/ISO SQL. O padrão SQL vem evoluindo desde 1986 e existem diversas versões.

O MySQL é um software Open Source. Open Source significa que é possível para qualquer um usar e modificar o programa. Qualquer pessoa pode fazer download do MySQL pela Internet e usá-lo sem pagar nada. Se você quiser, você pode estudar o código fonte e alterá-lo para adequá-lo as suas necessidades. O MySQL usa a GPL (GNU General Public License - Licença Pública Geral GNU) , para definir o que você pode e não pode fazer com o software em diferentes situações. Se você sentir desconforto com a GPL ou precisar embutir o MySQL em uma aplicação comercial, você pode adquirir a versão comercial licenciada com a MySQL AB.

3.3.2 As Prinipais características do MySQL

A seguinte lista descreve algumas das características mais importantes do Programa de Banco de Dados MySQL.

* Portabilidade.
* Escrito em C e C++.
* Testado com uma ampla faixa de compiladores diferentes.
* Funciona em diversas plataformas.
* Utiliza o GNU Automake, Autoconf, e Libtool para portabilidade.
* APIs para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby e Tcl estão disponíveis.
* Suporte total a multi-threads usando threads diretamente no kernel. Isto significa que se pode facilmente usar múltiplas CPUs, se disponível.
* Fornece mecanismos de armazenamento transacional e não transacional.
* Tabelas em disco (MyISAM) baseadas em árvores-B extremamente rápidas com compressão de índices.
* É relativamente fácil se adicionar outro mecanismo de armazenamento. Isto é útil se você quiser adicionar uma interface SQL a um banco de dados caseiro.
* Um sistema de alocação de memória muito rápido e baseado em processo (thread).
* Joins muito rápidas usando uma multi-join de leitura única otimizada.
* Tabelas hash em memória que são usadas como tabelas temporárias.
* Funções SQL são implementadas por meio de uma biblioteca de classes altamente otimizada e com o máximo de performance. Geralmente não há nenhuma alocação de memória depois da inicialização da pesquisa.
* O código do MySQL foi testado com Purify (um detector comercial de falhas de memória) e também com o Valgrind, uma ferramenta GPL.
* Disponível como versão cliente/servidor ou embutida (ligada).
* Aceita diversos tipos de campos: tipos inteiros de 1, 2, 3, 4 e 8 bytes com e sem sinal, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET e ENUM.
* Registros de tamanhos fixos ou variáveis.
* Completo suporte a operadores e funções nas partes SELECT e WHERE das consultas. Por exemplo:

```
mysql> SELECT CONCAT(first_name, " ", last_name)
-> FROM nome_tbl
-> WHERE income/dependents > 10000 AND age > 30;
```

* Suporte pleno às cláusulas SQL GROUP BY e ORDER BY. Suporte para funções de agrupamento (COUNT(), COUNT(DISTINCT ...), AVG(), STD(), SUM(), MAX() e MIN()).

* Suporte para LEFT OUTER JOIN e RIGHT OUTER JOIN com as sintaxes SQL e ODBC.

* Aliás em tabelas e colunas são disponíveis como definidos no padrão SQL92.

* DELETE, INSERT, REPLACE, e UPDATE retornam o número de linhas que foram alteradas (afetadas). É possível retornar ao número de linhas com padrão coincidentes configurando um parâmetro quando estiver conectando ao servidor.

* O comando específico do MySQL, SHOW, pode ser usado para devolver informações sobre bancos de dados, tabelas e índices. O comando EXPLAIN pode ser usado para determinar como o otimizador resolve a consulta.

* Nomes de funções não conflitam com nomes de tabelas ou colunas. Por exemplo, ABS é um nome de campo válido. A única restrição é que para uma chamada de função, espaços não são permitidos entre o nome da função e o "("que o segue.

* Você pode misturar tabelas de bancos de dados diferentes na mesma pesquisa.

* Um sistema de privilégios e senhas que é muito flexível, seguro e que permite verificação baseada em estações/máquinas. Senhas são seguras porque todo o tráfico de senhas é criptografado quando você se conecta ao servidor.

* Lida com bancos de dados enormes. Usamos o Servidor MySQL com bancos de dados que contém 50.000.000 registros e sabemos de usuários que usam o Servidor MySQL com 60.000 tabelas e aproximadamente 5.000.000.000 de linhas.

* São permitidos até 32 índices por tabela. Cada índice pode ser composto de 1 a 16 colunas ou partes de colunas. O tamanho máximo do índice é de 500 bytes (isto pode ser alterado na compilação do MySQL). Um índice pode usar o prefixo de campo com um tipo CHAR ou VARCHAR.

* Os clientes podem se conectar ao servidor MySQL usando sockets TCP/IP, em qualquer plataforma. No sistema Windows na família NT (NT, 2000 ou XP), os clientes podem se conectar usando named pipes. No sistema Unix, os clientes podem se conectar usando arquivos sockets.

* A interface Connector/ODBC fornece ao MySQL suporte a programas clientes que usam conexão ODBC (Open-DataBase-Connectivity). Por exemplo, você pode usar o MS Access para conectar ao seu servidor MySQL. Os clientes podem ser executados no Windows ou Unix. O fonte do Connector/ODBC está disponível. Todas as funções ODBC são suportadas, assim como muitas outras.

* O servidor pode apresentar mensagem de erros aos clientes em várias línguas.

* Suporte total para vários conjuntos de caracteres, que incluem ISO-8859-1 (Latin1), big5, ujis e mais. Por exemplo, os caracteres Escandinavos "â", "ä", "ö"são permitidos em nomes de tabelas e colunas.

* Todos os dados são armazenados no conjunto de caracteres escolhidos. Todas as comparações em colunas de seqüências caso-insensitivo.

* A ordenação é feita de acordo com o conjunto de caracteres escolhidos (o modo sueco por padrão). É possível alterar isso quando o servidor MySQL é iniciado. Para ver um exemplo de várias ordenações avançadas, procure pelo código de ordenação Tcheca. O Servidor MySQL suporta diversos conjuntos de caracteres que podem ser especificados em tempo de compilação e execução.

* O servidor MySQL foi construído com suporte para instruções SQL que verificam, otimizam e reparam tabelas. Estas instruções estão disponíveis a partir da linha de comando por meio do cliente myisamcheck, O MySQL inclui também o myisamchk, um utilitário muito rápido para realizar estas operações em tabelas MyISAM.

* Todos os programas MySQL podem ser chamados com as opções –help ou -? para obter ajuda online.

3.4 Lição 2 - Instalando o MySQL

3.4.1 Instalando o MySQL no Debian - 1

APT - Utilitário de empacotamento

No início havia o .tar.gz. Os usuários tinham de penar para compilar cada programa usado em seu sistema GNU/Linux, ou outro qualquer. Quando o Debian foi criado, sentiu-se a necessidade de um sistema de gerenciamento de pacotes instalados no sistema. Deu-se a esse sistema o nome de dpkg. Assim surgiu o famoso "pacote". Logo após a Red Hat resolveu criar seu conhecido sistema rpm.

Rapidamente outro dilema tomou conta das mentes dos produtores de GNU/Linux. Uma maneira rápida, prática e eficiente de se instalar pacotes, gerenciando suas dependências automaticamente e tomando conta de seus arquivos de configuração ao atualizar. Assim, o Debian, novamente pioneiro, criou o APT ou Advanced Packaging Tool.

3.4.2 Instalando o MySQL no Debian - 2

A distribuição Debian possui uma ferramenta chamada "apt"utilizada para instalar softwares. Essa ferramenta é de grande importância durante nosso processo de instalação, pois ela gerenciará todo o processo. No diretório /etc/apt existe um arquivo chamado "sources.list"com a função de armazenar os endereços dos repositórios com os pacotes de instalação. Vejamos um exemplo deste arquivo.

#deb file:///cdrom/ sarge main

deb http://ftp.br.debian.org/debian/ stable main
deb-src http://ftp.br.debian.org/debian/ stable main

deb http://security.debian.org/ stable/updates main

deb http://ftp.br.debian.org/debian/ testing main contrib non-free
deb-src http://ftp.br.debian.org/debian/ testing main contrib non-free
deb ftp://ftp.nerim.net/debian-marillat/ testing main

A primeira linha está comentada, isso se dá pela utilização do carácter #. Caso não estivesse comentada poderíamos instalar softwares diretamente dos cd's de instalação.
As demais linhas são endereços para repositórios que estão na internet.
O pacote com os arquivos do MySQL pode ser encontrado no endereço "deb http://ftp.br.debian.org/debian/ stable main"porém não será a última versão do MySQL.
Instalaremos a última versão do MySQL, e para isso será necessário o repositório "deb ftp://ftp.nerim.net/debia marillat/ testing main", porém pode ser utilizado outros que também possuam a última versão do MySQL.

3.4.3 Instalando o MySQL no Debian - 3

Instalando o MySQL no Debian

Depois de verificado o "sources.list", vamos para instalação. Para verficiar se os pacotes estão realmente à disposição utilizamos o seguinte comando:

shell> apt-cache search mysql-server

A saída será semelhante a esta:

mysql-server - empty transitional package
mysql-server-4.1 - empty transitional package
mysql-server-5.0 - mysql database server binaries
phpbb2-conf-mysql - Automatic configurator for phpbb2 on MySQL database
scoop - Web-based collaborative media application
webmin-mysql - mysql-server control module for webmin

A útlima versão do MySQL-server neste exemplo é a 5.0 . Agora para instalar basta digitarmos o camando:

shell> apt-get install mysql-server-5.0

A saída será semelhante a esta:

Lendo Lista de Pacotes... Pronto
Construindo Árvore de Dependências... Pronto
Os pacotes extra a seguir serão instalados:
gcc-4.0-base libdbd-mysql-perl libdbi-perl libgcc1
libmysqlclient15 libncurses5 libnet-daemon- erl libplrpc-perl
libreadline4 libreadline5 libstdc++6 mysql-client-5.0
mysql-common readline-common
Pacotes sugeridos :
dbishell libcompress-zlib-perl
Os NOVOS pacotes a seguir serão instalados:
gcc-4.0-base libdbd-mysql-perl libdbi-perl libmysqlclient15
libnet-daemon-perl libplrpc-perl libstdc++6 mysql-client-5.0
mysql-common mysql-server-5.0 readline-common
Os pacotes a seguir serão atualizados :
libgcc1 libncurses5 libreadline4 libreadline5
4 pacotes atualizados, 11 pacotes novos instalados, 0 a serem
removidos e 712 não atualizados.
É preciso fazer o download de 26,9MB de arquivos.
Depois de desempacotamento, 59,4MB adicionais de espaço em
disco serão usados.
Quer continuar? [S/n]

Digite S para que os pacotes sejam recebidos.

Uma tela de configuração apareçerá, bastando continuar. Pronto, seu Banco de Dados esta pronto para ser utilizado!

3.4.4 Instalando a partir dos fontes - 1

Instalando a partir dos fontes

Para instalar o MySQL a partir de seu código fonte você precisará das seguintes ferramentas:

* GNU gunzip para descompactar a distribuição.
* Um tar razoável para desempacotar a distribuição. Por exemplo o GNU tar.
* Um compilador ANSI C++ funcional. gcc >= 2.95.2, egcs >= 1.0.2 ou egcs 2.91.66, SGI C++, e SunPro C++ são alguns dos compiladores que funcionam.
* Um bom programa make. GNU make é sempre recomendado e é algumas vezes necessário. Se você tiver problemas, recomendamos tentar o GNU make 3.75 ou mais novo.

3.4.5 Instalando a partir dos fontes -2

Visão Detalhada da Instalação

1. Baixe o arquivo com o código fonte do MySQL em http://dev.mysql.com/get/Downloads/MySQL-5.0/mysql-5.0.22.tar.gz/from/http://www.linorg.usp.br/mysql/
Este é um servidor brasileiro. Caso haja algum problema acesse http://dev.mysql.com/downloads/mysql/5.0.ht

2. Acesse o terminal e escolha um diretório para descompacatar o arquivo tar.gz baixado.

3. Distribuições fontes do MySQL são fornecidas como arquivos tar compactados e tem nomes como mysql-versão.tar.gz. Dentro do diretório escolhido para instalação digite o comando:

* **tar xvzf mysql-versão.tar.gz**

4. Adicione um usuário e grupo para o mysql executar assim:

* **groupadd mysql**
* **useradd -g mysql mysql**

Useradd e groupadd podem mudar em diferentes versões de Unix. Elas podem também ser chamadas adduser e addgroup. Você pode escolher outros nomes para o usuário e grupo em vez de mysql.

6. Acesse o diretório onde o arquivo foi descompactado:

* **cd diretorio**

7. Configure e compile tudo:

* ./**configure –prefix=/usr/local/mysql**
* m**ake**

Obs: –prefix indicará o diretório onde os arquivos binários ficarão armazenados. Neste ponto o mysql já está operacional, porém não está instalado no diretório padrão do sistema.

8. Instalar tudo:

* **make install**

Você deve executar este comando como root.

9. Crie as tabelas de permissões do MySQL.

* **bin/mysql_install_db**

10. Altere o dono dos binários para root e do diretório de
dados para o usuário que irá executar o mysqld:

* **chown -R root /usr/local/mysql**
* **chown -R mysql /usr/local/mysql/var**
* c**hgrp -R mysql /usr/local/mysql**

O primeiro comando altera o atributo de proriedade dos arquivos para o usuário root, o segundo altera o atributo de propriedade do diretório de dados para o usuário mysql, e o terceiro altera o atributo de grupo para o grupo mysql.

11. Inicie o servidor MySQL com o seguinte comando:

* **/usr/local/mysql/bin/mysqld_safe –user=mysql &**

3.4.6 Configuração / Testes

Depois de instalado o mysql, é necessário criar as tabelas de concessões e iniciar o servidor. No terminal, digite os seguintes comandos:

* **shell> diretorio_instalação/bin/mysql_install_db**
* **shell> cd diretorio_instalação_mysql**
* **shell> diretorio_instalação/bin/mysqld_safe –user=mysql &**

O script mysql_install_db cria os bancos mysql, test e as entradas de privilégio para o usuário que utiliza o script e para o usuário root. O banco de dados mysql irá armazenar todos os privilégios dos outros bancos de dados e o banco test será utilizado para testar o MySQL. O mysql_install_db não irá sobrescrever nenhuma tabela de privilégios antiga, então deve ser seguro executá-lo em quaisquer circunstâncias.

O script mysqld_safe inicia o mysqld, daemon do mysql. O daemon é um programa que roda em um computador servidor, atendendo solicitações de outros programas, executando determinadas tarefas e retornando uma resposta adequada.

De agora em diante utilizaremos BINDIR para representar o caminho para a localização na qual os programas como mysqladmin e mysqld_safe estão instalados. Caso você esteja utilizando o Debian, o diretório com os binários provavelmente está localizado em /usr/bin. Caso você tenha instalado o MySQL a partir do código fonte, este diretório foi indicado na etapa de configuração.

Com o **mysqladmin** podemos realizar uma gama de testes para verificar se o mysql está em perfeito funcionamento. Execute o comando abaixo para verificar se o servidor está em funcionamento e respondendo às conexões:

* **shell> BINDIR/mysqladmin version**

A saída de mysqladmin version deve ser algo parecido com o indicado abaixo:

./mysqladmin Ver 8.41 Distrib 5.0.22, for pc-linux-gnu on i686
Copyright (C) 2000 MySQL AB & MySQL Finland AB & TCX DataKonsult AB
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL license

Server version 5.0.22
Protocol version 10
Connection Localhost via UNIX socket
UNIX socket /tmp/mysql.sock
Uptime: 42 sec

Threads: 1 Questions: 2 Slow queries: 0 Opens: 0 Flush tables: 1 Open tables: 6 Queries per second avg: 0.048

Verifique se você pode desligar o servidor:

* **shell> BINDIR/mysqladmin -u root shutdown**

Execute os testes abaixo para verificar se o servidor está funcionando corretamente. Os comandos devem retornar algo parecido com o mostrado abaixo. Caso isto não aconteça, provavelmente as tabelas de concessão não foram criadas e o servidor não foi iniciado.

* **shell> BINDIR/mysqlshow**

+——————–+
| Databases |
+——————–+
| information_schema |
| mysql |
| test |

```
+——————————+
```

* **shell> BINDIR/mysqlshow mysql**

```
+——————————+
| Tables |
+——————————+
| columns_priv |
| db |
| func |
| help_category |
| help_keyword |
| help_relation |
| help_topic |
| host |
| proc |
| procs_priv |
| tables_priv |
| time_zone |
| time_zone_leap_second |
| time_zone_name |
| time_zone_transition |
| time_zone_transition_type |
| user |
+——————————+
```

* **shell> BINDIR/mysql -e "SELECT host,db,user FROM db"mysql**

```
+——+———+——+
| host | db | user |
+——+———+——+
| % | test | |
| % | test_% | |
+——+———+——+
```

3.5 Lição 3 - Aplicabilidade e Uso

3.5.1 Um pouco da história do SQL

Structured Query Language, ou Linguagem de Questões Estruturadas ou SQL, é uma linguagem de pesquisa declarativa para banco de dados relacional (bases de dados relacionais). Muitas das características originais do SQL foram inspiradas no cálculo de tuplos.

Embora o SQL tenha sido originalmente criado pela IBM, rapidamente surgiram vários "dialetos"desenvolvidos por outros produtores. Essa expansão levou à necessidade de ser criado e adaptado um padrão para a linguagem. Esta tarefa foi realizada pela American National Stan-

dards Institute (ANSI) em 1986 e ISO em 1987.

O SQL foi revisto em 1992 e a esta versão foi dado o nome de SQL-92. Foi revisto novamente em 1999 e 2003 para se tornar SQL:1999 (SQL3) e SQL:2003, respectivamente. O SQL:1999 usa expressões regulares de emparelhamento, queries recursivas e gatilhos (triggers). Também foi feita uma adição controversa de tipos não-escalados e algumas características de orientação a objeto. O SQL:2003 introduz características relacionadas ao XML, sequências padronizadas e colunas com valores de auto-generalização (inclusive colunas-identidade).

Tal como dito anteriormente, o SQL, embora padronizado pela ANSI e ISO, possui muitas variações e extensões produzidos pelos diferentes fabricantes de sistemas gerenciadores de bases de dados. Tipicamente a linguagem pode ser migrada de plataforma para plataforma sem mudanças estruturais principais.

Outra aproximação é permitir para código de idioma processual ser embutido e interagir com o banco de dados. Por exemplo, o Oracle e outros incluem Java na base de dados, enquanto o PostgreSQL permite que funções sejam escritas em Perl, Tcl, ou C, entre outras linguagens.

3.5.2 Aplicabilidade e uso

A linguagem SQL é básicamente uma linguagem de consulta a banco de dados. Ela é bem diferente das linguagens comuns de programação, a principal diferença é que a linguagem SQL não é uma linguagem procedural, ao contrário da grande maioria das linguagens de programação. Na linguagem SQL não se especifica como, ou em que ordem, serão executados os processos que irão fornecer os resultados requeridos, na SQL, nós apenas informamos o que queremos e o sistema de banco de dados é o responsável por escolher adequadamente os procedimentos a serem executados, de forma que os resultados sejam obtidos com a maior eficiência possível.

A linguagem SQL é uma linguagem relacional, isto é, ela é ideal para o tratamento de dados relacionados. De uma forma grotesca, dados relacionados são aqueles que podem ser arranjados em uma tabela, onde cada linha forma uma unidade lógica de dados.

3.5.3 Tipos de Comandos

DML - Linguagem de Manipulação de Dados

Primeiro há os elementos da DML (Data Manipulation Language - Linguagem de Manipulação de Dados). A DML é um subconjunto da linguagem usada para selecionar, inserir, atualizar e apagar dados. SELECT é o comumente mais usado do DML, comanda e permite ao usuário especificar uma query como uma descrição do resultado desejado. A questão não especifica como os resultados deveriam ser localizados.

*INSERT é usada para somar uma fila (formalmente uma tupla) a uma tabela existente.

*UPDATE para mudar os valores de dados em uma fila de tabela existente.

*DELETE permite remover filas existentes de uma tabela.

*BEGIN WORK (ou START TRANSACTION, dependendo do dialeto SQL) pode ser usado para marcar o começo de uma transação de banco de dados que pode ser completada ou não.

*COMMIT envia todos os dados das mudanças permanentemente.

*ROLLBACK faz com que as mudanças nos dados existentes desde que o último COMMIT ou ROLLBACK sejam descartadas.

COMMIT e ROLLBACK interagem com áreas de controle como transação e locação. Ambos terminam qualquer transação aberta e liberam qualquer cadeado ligado a dados. Na ausência de um BEGIN WORK ou uma declaração semelhante, a semântica de SQL é dependente da implementação.

DDL - Linguagem de Definição de Dados

O segundo grupo é a DDL (Data Definition Language - Linguagem de Definição de Dados). Uma DDL permite ao usuário definir tabelas novas e elementos associados. A maioria dos bancos de dados de SQL tem extensões proprietárias no DDL.

Os comandos básicos da DDL são:

*CREATE cria um objeto (uma Tabela, por exemplo) dentro do base de dados.

*DROP apaga um objeto do banco de dados.

Alguns sistemas de banco de dados usam o comando ALTER, que permite ao usuário alterar um objeto, por exemplo, adicionando uma coluna a uma tabela existente.

DCL - Linguagem de Controle de Dados

O terceiro grupo é o DCL (Data Control Language - Linguagem de Controle de Dados). DCL controla os aspectos de autorização de dados e licenças de usuários para controlar quem tem acesso para ver ou manipular dados dentro do banco de dados.

Duas palavras-chaves da DCL:

*GRANT - autoriza ao usuário executar ou setar operações.

*REVOKE - remove ou restringe a capacidade de um usuário de executar operações.

3.6 Lição 4 - Criando e selecionando um banco de dados

3.6.1 Comandos de Acesso

Acessando o Servidor MySQL

A partir deste momento começaremos a criar banco de dados utilizando o MySQL, para tanto, precisamos logar no servidor.
Com o terminal aberto digitamos o seguinte comando:

shell> mysql -u root -p
Enter password: ******

Conectaremos como root (administrador), pois ainda não existem usuários cadastrados. Quando apareçer o campo "Enter password", tecle enter. A senha padrão de root vem desabilitada, na próxima seção aprenderemos como trocá-la.

Agora que você já sabe como entrar com os comandos, é hora de acessar um banco de dados.

Suponha que você seja dono de uma vídeo locadora e que deseja melhorar a organização de seus filmes. Você pode fazer isto criando tabelas para armazenar dados referentes aos filmes e, a partir destas tabelas, você será capaz de gerar relatórios e fazer buscas rápidas ao acervo disponvível.

O banco de dados "Locadora", apesar de simples, será de grande utilidade ao dono da lacoradora. Imagine agora empresas que trabalham com fluxos gigantescos de informação, como um sistema de gerenciamento de banco de dados é de fundamental para a organização da instituição.

Verificando a existência de banco de dados

Utilize a instrução SHOW para saber quais bancos de dados existem atualmente no servidor:

mysql> SHOW DATABASES;
+————————+
| Database |
+————————+
| mysql |
| test |
| tmp |
+————————+

A lista de bancos de dados provavelmente será diferente na sua máquina, mas os bancos de dados mysql e test provavelmente estarão entre eles.

Utilize a instrução USE para acessar o banco de dados test :

mysql> USE test

Database changed

A instrução USE pode ser utilizada sem o delimitador ";". Veremos mais a frente que isto não é possível com outros comandos. Outra característica intrínsecas da instrução USE é que ela deve ser utilizada em uma única linha.

3.6.2 Criando e selecionando um banco de dados

Se o administrador criar seu banco de dados quando configurar as suas permissões, você pode começar a usá-lo. Senão, você mesmo precisa criá-lo:

mysql> CREATE DATABASE Locadora;

Para excluirmos o banco de dados utilizamos o seguinte comando:

mysql> DROP DATABASE Locadora;

Este comando também serve para excluir tabelas, basta trocarmos DATABASE por TABLE.

No Unix, nomes de bancos de dados são caso sensitivo (ao contrário das palavras chave SQL). Isto quer dizer que Locadora é direferente de LOCADORA ou locadora.

Criar um bancos de dados não o seleciona para o uso. Faça isto utilizando a instrução use.

mysql> USE Locadora
Database changed

Uma forma alternativa de colocar o banco de dados em uso é passando o nome do banco como parâmetro durante a inicialização do mysql.

shell> mysql -h servidor -u usuario -p Locadora
Enter password: ******

Perceba que -**h** indica o "host"do servidor, ou seja, sua localização na rede. Caso o servidor esteja na máquina local este parâmetro pode ser omitido. -**u** indica o usuário que acessará o sistema e -**p** a senha. Atenção, Locadora não é o parâmetro relativo a senha. Locadora é o banco que desejamos colocar em uso. Caso deseje passar a senha na linha de comando você deve fazê-lo sem usar espaços (por exemplo, -pminhasenha e não como em -p minhasenha). Entrentanto, não é recomendado colocar a senha na linha de comando, visto que a senha fica exposta a outras pessoas.

3.7 Lição 5 - Criando e manipulando Tabelas

3.7.1 Tipos de Tabelas

Antes de criarmos nossa tabela e continuarmos com o exemplo da lição anterior, vamos aprender os tipos de tabelas suportado pelo MySQL se suas características:

No MySQL podemos definir vários tipos de tabelas, sendo que um banco de dados pode conter diferentes tipos de tabelas.
Quando você cria uma nova tabela, você pode dizer ao MySQL que tipo de tabela criar. O tipo padrão é o MyISAM.

Abaixo segue um descrição rápida dos tipos de tabelas suportados pelo MySQL:

MyISAM: MyISAM é o tipo de tabela padrão no MySQL. Ela é baseada no código ISAM e possui várias extensões úteis. O índice é armazenado em um arquivo com extensão .MYI (MYIndex), e os dados são armazenados em um arquivo com a extensão .MYD (MYData). Você pode verificar/reparar tabelas MyISAM com o utilitário myisamchk.

Merge: Uma tabela MERGE (também conhecida como tabela MRG_MyISAM) é uma coleção de tabelas MyISAM idênticas que podem ser usadas como uma. Você só pode fazer SELECT, DELETE, e UPDATE da coleção de tabelas. Se você fizer um DROP na tabela MERGE, você só está apagando a especificação de MERGE.

HEAP: As tabelas HEAP do MySQL utilizam hashing 100% dinâmico sem áreas em excesso. Não há espaços extras necessários para listas livres. Tabelas HEAP também não têm problemas com deleção + inserção, o que normalmente é comum em tabelas com hash:

InnoDB: O InnoDB prove o MySQL com um mecanismo de armazenamento seguro com transações (compatível com ACID) com commit, rollback, e recuperação em caso de falhas. InnoDB faz bloqueio a nível de registro e também fornece uma leitura sem bloqueio em SELECT em um estilo consistente com Oracle. Estes recursos aumentam a performance e a concorrência de multi-usuários. InnoDB é o primeiro gerenciador de armazenamento no MySQL que suportam restrições FOREIGN KEY.

BerkeleyDB ou BDB: disponível em http://www.sleepycat.com/ tem provido o MySQL com um mecanismo de armazenamento transacional. Tabelas BDB podem ter maior chance de sobrevivência a falhas e também são capazes de realizar operações COMMIT e ROLLBACK em transações.

3.7.2 Criando uma tabela

A criação de uma tabela é feita com a instrução CREATE TABLE. O MySQL possui uma característica um pouco diferente de outros SGBD's. Durante a criação da tabela podemos definir o tipo desta tabela.
Exemplo:

CREATE TABLE teste (
id INT NOT NULL,
texto CHAR(30) NOT NULL,
PRIMARY KEY (id)
) TYPE=MyISAM;

No comando acima, TYPE=MyISAM, indica que a tabela criada será do tipo MyISAM, que é o valor default caso não seja informado o TYPE. Através deste exemplo observamos que a sintaxe do CREATE TABLE é muito simples, sendo necessário apenas informar o nome da tabela, no caso "teste", os campos da tabela e o tipo da tabela, podendo este último ser ocultado.

Continuando com o exemplo que vínhamos seguindo na lição anterior, com a criação do banco de dados "Locadora", o comando SHOW TABLES nos mostrará as tabelas existentes neste banco de dados.

mysql> SHOW TABLES;
Empty set (0.00 sec)

A parte mais difícil é decidir qual a estrutura que seu banco de dados deve ter: quais tabelas você precisará e que colunas estarão em cada uma delas.

Você irá precisar de uma tabela para guardar os registros de cada um de seus filmes. A tabela poderá se chamar "Acervo"e deverá conter os campos que identificam os filmes. Campos com nome, gênero, censura e duração.

Você pode usar o banco de dados para tarefas como gerar avisos de atraso de devolução, ou até mesmo para gerar listas com informações de todos os clientes da locadora. Logo teríamos mais uma tabela com informações de todos os clientes e esta tabela poderia se relacionar com a tabela Acervo através de algum campo. Vejam como podemos modelar uma estrutura organizacional de tal forma que sistemas cada vez mais complexos sejam construídos a partir de base de dados.

Você provavelmente pode pensar em outros tipos de informações que poderão ser úteis na tabela Acervo, mas as identificadas até o momento são suficientes por agora: nome, gênero, censura e duração.

Utilize a senteça CREATE TABLE para especificar o layout de sua tabela:

mysql> CREATE TABLE Acervo(nome VARCHAR(20), gênero VARCHAR(20), censura INT, duração TIME);
Query OK, 0 rows affected (0,00 sec)

VARCHAR é uma boa escolha para os campos nome e gênero porque os valores da coluna são de tamanho variável. Os tamanhos destas colunas não precisam necessariamente de ser os mesmos e não precisam ser 20.

O campo censura é bem representando com o tipo inteiro, visto que as idades são dadas em números inteiros. O mesmo acontece com o campo duração, onde TIME é um tipo utilizado para armazenar dados no formato 'HH:MM:SS' (Veremos mais a frente todos os tipos de campos).

Agora que você criou uma tabela, a instrução SHOW TABLES deve produzir alguma saída:

```
mysql> SHOW TABLES;
+-------------+
| Tables_in_Locadora |
+-------------+
| Acervo |
+-------------+
1 row in set (0,02 sec)
```

Para verificar se sua tabela foi criada da forma que você esperava, utilize a instrução DESCRIBE:

```
mysql> DESCRIBE Acervo;

+-------+----------+----+---+---------+-----+
| Field | Type | Null | Key | Default | Extra |
+-------+----------+----+---+---------+-----+
| nome | varchar(20) | YES | | NULL | |
| gênero | varchar(20) | YES | | NULL | |
| censura | int(11) | YES | | NULL | |
| duração | time | YES | | NULL | |
+-------+----------+----+---+---------+-----+
4 rows in set (0,00 sec)
```

Você pode usar DESCRIBE a qualquer hora, por exemplo, se você esquecer os nomes das colunas na sua tabela ou de que tipos elas têm.

3.7.3 Tipos de Campos

Para cada campo de cada uma da tabelas, é necessário determinar o tipo de dados que contem, para poder ajustar a estrutura da base de dados, e conseguir um armazenamento com a menor utilização de espaço.

Os tipos de dados que pode ter um campo, podem-se agrupar em três grandes grupos:

* Tipos numéricos
* Tipos de Data
* Tipos de Cadeia

Tipos numéricos:
Existem tipos de dados numéricos, que se podem dividir em dois grandes grupos, os que estão em vírgula flutuante (com decimais) e os que não.

* TinyInt: é um número inteiro com ou sem sinal. Com sinal à margem de valores válidos é desde -128 até 127. Sem sinal, a margem de valores é de 0 até 255

* Bit ou Bool: um número inteiro que pode ser 0 ou 1.
* SmallInt: número inteiro com ou sem sinal. Com sinal à margem de valores válidos é desde -32768 até 32767. Sem sinal, a margem de valores é de 0 até 65535.
* MediumInt: número inteiro com ou sem sinal. Com sinal à margem de valores válidos é desde -8.388.608 até 8.388.607. Sem sinal, a margem de valores é de 0 até 16777215.
* Integer, Int: número inteiro com ou sem sinal. Com sinal à margem de valores válidos é desde -2147483648 até 2147483647. Sem sinal, a margem de valores é de 0 até 429.496.295
* BigInt: número inteiro com ou sem sinal. Com sinal à margem de valores válidos é desde -9.223.372.036.854.775.808 até 9.223.372.036.854.775.807. Sem sinal, à margem de valores é de 0 até 18.446.744.073.709.551.615. * Float: número pequeno em vírgula flutuante de precisão simples. Os valores válidos vão desde -3.402823466E+38 até -1.175494351E-38,0 ou desde 175494351E-38 até 3.402823466E+38.
* xReal, Double: número em vírgula flutuante de dupla precisão. Os valores permitidos vão desde -1.7976931348623157E+308 até -2.2250738585072014E-308, 0 e desde 2.2250738585072014E-308 até 1.7976931348623157E+308
* Decimal, Dec, Numeric: Número em vírgula flutuante desempacotado. O número armazena-se como uma cadeia.

Tipos de data:
Na hora de armazenar datas, o MySQL não verifica de uma maneira restrita se uma data é válida ou não. Simplesmente comprova que o mês está compreendido entre 0 e 12 e que o dia está compreendido entre 0 e 31.

* Date: tipo data, armazena uma data. A margem de valores vai desde o 1 de Janeiro de 1001 ao 31 de dezembro de 9999. O formato de armazenamento é de ano-mes-dia.
* DateTime: Combinação de data e hora. A margem de valores vai desde o 1 de Janeiro de 1001 às 0 horas, 0 minutos e 0 segundos ao 31 de Dezembro de 9999 às 23 horas, 59 minutos e 59 segundos. O formato de armazenamento é de ano-mes-dia horas:minutos:segundos.
* TimeStamp: Combinação de data e hora. A margem vai desde o 1 de Janeiro de 1970 ao ano 2037.
* Time: armazena uma hora. A margem de horas vai desde -838 horas, 59 minutos e 59 segundos. O formato de armazenamento é 'HH:MM:SS'.
* Year: armazena um ano. A margem de valores permitidos vai desde o ano 1901 ao ano 2155. O campo pode ter tamanho dois ou tamanho 4 dependendo de se queremos armazenar o ano com dois ou quatro algarismos.

Tipos de cadeia:

* Char(n): armazena uma cadeia de longitude fixa. A cadeia poderá conter desde 0 até 255 caracteres.
* VarChar(n): armazena uma cadeia de longitude variável. A cadeia poderá conter desde 0 até 255 caracteres. Dentro dos tipos de cadeia pode-se distinguir dois subtipos, os tipo Test e os tipo Blob (Binary Large Object) A diferença entre um tipo e outro é o tratamento que recebem na hora de ordená-los e compará-los. No tipo test ordena-se sem ter importância as maiúsculas e as minúsculas e no tipo blob ordena-se tendo em conta as maiúsculas e minúsculas. Os tipos blob utilizam-se para armazenar dados binários como podem ser ficheiros.
* TinyText e TinyBlob: Coluna com uma longitude máxima de 255 caracteres.
* Blob e Text: um texto com um máximo de 65535 caracteres.

* MediumBlob e MediumText: um texto com um máximo de 16.777.215 caracteres.
* LongBlob e LongText: um texto com um máximo de caracteres 4.294.967.295.

Há que ter em conta que devido aos protocolos de comunicação os pacotes podem ter um máximo de 16 Mb.

* Enum: campo que pode ter um único valor de uma lista que se especifica. O tipo Enum aceita até 65535 valores diferentes.
* Set: um campo que pode conter nenhum, um ou vários valores de uma lista. A lista pode ter um máximo de 64 valores.

3.7.4 Alterando campos da tabela

Suponha que durante a criação da nossa tabela Acervo não fizessemos uma boa escolha com o tamanho do campo nome(varchar(20)). Poderíamos remover a tabela e criar outra semelhante com os comandos abaixo:

DROP TABLE Acervo;

mysql> CREATE TABLE Acervo(nome VARCHAR(40), gênero VARCHAR(20), censura INT, duração TIME);

Esta é uma das alternativas, porém, não a melhor. Imaginem se dados já tivessem sido inseridos na tabela. Teríamos que refazer todo o processo de inserção de dados, o que muitas vezes é inviável. O Comando ALTER TABLE nos permite realizar uma infinidade de operações com a tabela como por exemplo:

* Modificar o nome da tabela;
* Adicionar uma coluna;
* Remover uma coluna ;
* Alterar a estrutura da coluna;

Para alterarmos o campo "nome"da tabela Acervo, utilizamos a seguinte instrução:

* **ALTER TABLE Acervo MODIFY nome varchar(40);**

Agora vamos inserir uma nova coluna, por exemplo "data_lançamento".

* **ALTER TABLE Acervo ADD data_lançamento date;**

Para remover basta inserirmos DROP no lugar de ADD e retirarmos campo.

* **ALTER TABLE Acervo DROP data_lançamento;**

Viu como é simples? Existem outras funcionalidades do ALTER TABLE que não foram exemplificadas, para maiores informações consulte a documentação do MySQL no site http://www.mysql.org.

3.7.5 Primary Key e Foreign Key

Quando estamos trabalhando com banco de dados um conceito importantíssimo é o conceito de chave. A partir do MySQL 5.0 foi introduzido o conceito de integridade referencial as relações entre tabelas no banco de dados.

Chave Primária ou Primary key é um campo da tabela com valores únicos que caracterizam os registros, por exemplo:
Uma vídeo locadora possui um banco de dados com a tabela "FILME"e com os campos codFilme , nome, censura, genero e duracao. codFilme nesta tabela podería representar a primary key desta tabela, pois seus valores seriam únicos e identificariam os registros.

Já a **chave estrangeira ou foreign key** nada mais é do que a chave primária de outra tabela, que servirá como elo entra as tabelas.
Seguindo com o exemplo dado anteriormente poderíamos, ao invés de ter o campo genero na tabela "FILME", criar outra tabela chamada "GENERO"com os campos codGenero (primary key), nomeGenero. Agora poderiamos, introduzir na tabela "FILME"uma foreign key, que no caso seria a primary key da tabela "GENERO".

Por que utilizar chaves?
Imagine se todas as vezes que você for cadastrar um filme seja necessário digitar o nome do gênero. Poderiam haver erros durante a digitação do nome, como acentuação e abreviação do mesmo. Por isso é mais importante ter uma tabela "GENERO"com os dados previamente cadastrados, sendo necessário apenas informar o código deste filme.

Quando estamos trabalhando com integridade referencial, o que estamos fazendo é adicionar restrições às chaves estrangeiras para que as tabelas sejam relacionadas de tal forma que não seja possível realizar determinadas operações caso isto seja prejudicial a estrutura do banco de dados. Tabelas do tipo innoDB suportam integridade referencial.

O InnoDB implementa as restrições de integridade CASCADE, RESTRICT, SET NULL e SET DEFAULT. No primeiro caso, ao se remover um registro da tabela referenciada pela chave estrangeira os registros relacionados àquele removido serão eliminados em todas as tabelas relacionadas. O RESTRICT não permite a remoção de registros que possuam relacionamentos em outras tabelas. Os dois últimos atribuem os valores DEFAULT ou NULL para as chaves estrangeiras cujos registros relacionados foram excluídos. O exemplo abaixo ilustra algumas tabelas que utilizam regras de integridade:

CREATE TABLE genero(CodGenero INT auto increment, nomeGenero VARCHAR(20), PRIMARY KEY(CodGenero)) type=innoDB;

CREATE TABLE Acervo(CodFilme INT, nome VARCHAR(20), censura INT, duracao TIME, CodGenero INT, PRIMARY KEY(CodFilme),FOREIGN KEY(CodGenero) REFERENCES genero(CodGenero) ON DELETE RESTRICT) type=innoDB;

Perceba que não conseguimos criar a tabela filme antes da tabela gênero, isso acontece porque ainda não podemos relacionar a chave primária da tabela gênero com a chave estrangeira da tabela filme, sendo possível após a criação da tabela gênero. A sintaxe é bem simples de ser

observada:
PRIMARY KEY(coluna)
FOREIGN KEY(coluna_da_tabela_atual) REFERENCES
tabela(coluna_da_tabela_referenciada) ON restrição

3.7.6 Carregando dados em uma tabela

Depois de criar sua tabela, você precisará povoá-la. As instruções LOAD DATA e INSERT são úteis para isto.

Suponha que seu registro de Filmes possa ser descrito como é abaixo:

nome	gênero	censura	duracao
Rambo	ação	10	1:20:00
Titanic	romance	12	3:20:00

Como você está começando com uma tabela vazia, uma forma simples de povoá-la é criar um arquivo texto contendo uma linha para cada um de seus filmes, e depois carregar o conteúdo do arquivo para a tabela com uma simples instrução.

Você pode criar um arquivo texto acervo.txt contendo um registro por linha, com valores separados por tabulações e na mesma ordem em que as colunas foram listadas na instrução CREATE TABLE. Caso você não saiba que informação colocar em algum campo, você pode usar valores NULL. Para representá-lo em seu arquivo texto, use \N (barra invertidam N maíusculo). Por exemplo, o registro para o filme "Era do gelo":

nome	gênero	censura	duracao
Era do Gelo	comédia	0	\N

Para carregar o arquivo texto acervo.txt na tabela Acervo, este arquivo deverá estar na pasta do banco de dados Locadora que geralmente é /var/lib/mysql/Locadora/nome.txt, ou você pode explicitar o caminho completo para o arquivo. Use este comando:

mysql> LOAD DATA INFILE "acervo.txt"INTO TABLE Acervo;

Para adicionar registros um a um, basta utilizar a instrução INSERT. Caso você queira adicionar mais um filme à tabela "Acervo"basta utiilzar a instrução a seguir:

mysql> INSERT INTO Acervo VALUES ("nomedofilme","genero",18,NULL);

Observe a utilização de aspas duplas nos campos do tipo varchar() . Com o INSERT você também pode inserir NULL diretamente para representar um valor em falta. Não pode ser usado \N como você fez com LOAD DATA.

Também é possível inserir dados em colunas específicas da tabela, bastando para isso identificá-la.

mysql> INSERT INTO Acervo(nome) VALUES ('Pânico');

No MySQL quando estamos criando uma tabela podemos utilizar a palavra chave AUTO_INCREMENT para atribuir valores automaticamente às colunas. por exemplo:

CREATE TABLE Acervo2(cod_filme int AUTO_INCREMENT, nome VARCHAR(20), gênero VARCHAR(20), censura INT, duração TIME, PRIMARY KEY(cod_filme));

Na tabela acima, criamos uma nova tabela, Acervo2, com o campo cod_filme. Perceba que não é possível utilizar a palavra chave AUTO_INCREMENT sem informar que a coluna é PRIMARY KEY. Este campo receberá valores automaticamente, sendo necessário informar apenas a palavra default quando utilizada a instrução insert:

INSERT INTO Acervo2 VALUES(default,"código da vinci","drama",NULL,NULL);

Desta forma, toda vez que for inserido um novo registro, o campo cod_filme, o campo cod_filme será incrementado automaticamente.

3.8 Lição 6 - Consultando uma tabela

3.8.1 A instrução SELECT

A instrução SELECT é usada para recuperar informações de uma tabela. A forma geral da instrução é:

SELECT o_que_mostrar
FROM de_qual_tabela
WHERE condições_para_satisfazer;

o_que_mostrar indica o que você deseja ver. Isto pode ser uma lista de colunas ou * para indicar "todas colunas."de_qual_tabela indica a tabela de onde você deseja recuperar os dados. A cláusula WHERE é opcional. Se estiver presente, condições_para_satisfazer especificam as condições que os registros devem satisfazer para fazer parte do resultado.

3.8.2 Selecionando todos os dados / UPDATE E DELETE

Abaixo mostramos a forma mais simples do SELECT recuperar tudo de uma tabela:

mysql> SELECT * FROM Acervo;
+———+———————+———————+———+———-+
| cod_filme | nome | gênero | censura | duração |
+———+———————+———————+———+———-+
| 1 | Titanic | Romance | 12 | 03:20:00 |
| 2 | Código da Vinci | Ação | 12 | 02:20:00 |

| 3 | Todo mundo em Pânico | Comédia | 14 | 01:30:00 |
| 4 | O chamado | Terror | 14 | 01:50:00 |
| 5 | Star Wars | ficção científica | 14 | 01:50:00 |
+———+——————+——————+———+———+
5 rows in set (0,00 sec)

Esta forma de busca é muito útil quando desejamos ver tabela inteira. Ela retorna todos os campos da tabela, ajudando a encontrar erros de inserção de dados. Por exemplo, consultando informções sobre o filme Titanic você percebeu que as informações sobre a duração do filme estavam erradas. Para corrigir este problema vamos aprender duas novas instruções.

Delete e Update

Edite o arquivo acervo.txt para corrigir o erro, depois limpe a tabela e recarregue-o usando DELETE e LOAD DATA:
mysql> DELETE FROM Acervo; (apagará todos os dados da tabela)
mysql> LOAD DATA INFILE "acervo.txt"INTO TABLE pet;

Ou podemos corrigir somente o registro errado com uma instrução UPDATE:

mysql> UPDATE Acervo SET duração = "03:00:00"WHERE nome = "Titanic";

O UPDATE altera apenas o registro em questão e não exige que você recarregue a tabela.

3.8.3 Selecionando registros específicos

Como foi mostrado na seção anterior, é fácil recuperar uma tabela inteira. Apenas omita a cláusula WHERE da instrução SELECT. Mas normalmente você não quer ver toda a tabela, particularmente quando a tabela fica grande. Em vez disso, você estará mais interessado em ter a resposta de uma questão em particular, na qual você especifica detalhes da informação que deseja. Vamos ver algumas consultas de seleção nos termos das questões sobre seus animais.

Como foi mostrado anteriormente é muito simples recuperar informações de tabela inteira. Porém, normalmente você quer buscar informações específicas, principalmente se a tabela for grande. Vamos ver algumas consultas de seleção que retornam valores específicos.

Você pode selecionar apenas registros específicos da sua tabela. Por exemplo, se você deseja verificar a alteração que fez o filme Titanic, utilize a seguinte instrução:

mysql> SELECT * FROM Acervo WHERE nome = "Titanic";
+———+———+———+———+———+
| cod_filme | nome | gênero | censura | duração |
+———+———+———+———+———+
| 1 | Titanic | Romance | 12 | 03:00:00 |
+———+———+———+———+———+
1 row in set (0,05 sec)

São retornadas todas as colunas com informações referentes ao filme Titanic. Comparações de strings normalmente são caso insensitivo, então você pode especificar o nome como "Titanic", "TITANIC", etc. Você pode especificar condições em qualquer coluna. Por exemplo, pode-se fazer uma pesquisa que retorne os filmes com censura maior ou igual à 12 anos.

```
    mysql> SELECT * FROM Acervo WHERE censura >= 12;
+————+————————+————————+———+———+
| cod_filme | nome | gênero | censura | duração |
+————+————————+————————+———+———+
| 1 | Titanic | Romance | 12 | 03:00:00 |
| 2 | Código da Vinci | Ação | 12 | 02:20:00 |
| 3 | Todo mundo em Pânico | Comédia | 14 | 01:30:00 |
| 4 | O chamado | Terror | 14 | 01:50:00 |
| 5 | Star Wars | ficção científica | 14 | 01:50:00 |
+————+————————+————————+———+———+
5 rows in set (0,04 sec)
```

Você pode combinar condições, por exemplo, para encontrar filmes com censura 12 anos e gênero Ação:

```
mysql> SELECT * FROM Acervo WHERE gênero = "Ação"AND censura = 12;
+————+————————+———+———+———+
| cod_filme | nome | gênero | censura | duração |
+————+————————+———+———+———+
| 2 | Código da Vinci | Ação | 12 | 02:20:00 |
+————+————————+———+———+———+
1 row in set (0,08 sec)
```

A consulta anterior utiliza o operador lógico AND (e). Existe também um operador OR (ou):

```
mysql> SELECT * FROM Acervo WHERE gênero = "Ação"OR censura = 12;
+————+————————+———+———+———+
| cod_filme | nome | gênero | censura | duração |
+————+————————+———+———+———+
| 1 | Titanic | Romance | 12 | 03:00:00 |
| 2 | Código da Vinci | Ação | 12 | 02:20:00 |
+————+————————+———+———+———+
2 rows in set (0,00 sec)
```

AND e OR podem ser misturados, porém AND terá maior precedência. A utilzação de parênteses auxilia na organização da consulta.

mysql> SELECT * FROM Acervo WHERE (gênero = "Ação"AND censura >=12) OR (gênero = "Comédia"AND duração >= "01:00:00");

```
+————+————————+———+———+———+
| cod_filme | nome | gênero | censura | duração |
+————+————————+———+———+———+
| 2 | Código da Vinci | Ação | 12 | 02:20:00 |
| 3 | Todo mundo em Pânico | Comédia | 14 | 01:30:00 |
+————+————————+———+———+———+
2 rows in set (0,00 sec)
```

Percebam que é possível fazer comparações com o campo "time"e as aspas devem obrigatoriamente ser colocadas.

3.8.4 Selecionando colunas específicas

Se você não desejar ver todo o registro de sua tabela, especifique as colunas em que você estiver interessado, separado por vírgulas. Por exemplo, se você deseja realizar uma consulta que retorne somente os nomes dos filmes juntamente com o gênero utilize a seguinte instrução:

mysql> SELECT nome,gênero FROM Acervo;

```
+————————+————————+
| nome | gênero |
+————————+————————+
| Titanic | Romance |
| Código da Vinci | Ação |
| Todo mundo em Pânico | Comédia |
| O chamado | Terror |
| Star Wars | ficção científica |
+————————+————————+
5 rows in set (0,00 sec)
```

Insira mais um campo à tabela com o gênero comédia:

mysql> INSERT INTO Acervo VALUES (default,"A era do gelo","Comédia",0,"1:30");

Para realizar uma query(consulta) que retorne todos os gêneros possíves utilize a intrução abaixo.

mysql> SELECT gênero FROM Acervo;

```
+————————+
| gênero |
+————————+
| Romance |
| Ação |
| Comédia |
| Terror |
| ficção científica |
| Comédia |
```

+————————-+
6 rows in set (0,00 sec)

Entretanto, perceba que o gênero Comédia apareçeu duas vezes. Para minimizar a saída, recupere cada registro apenas uma vez, adicionando a palavra chave DISTINCT:

mysql> SELECT DISTINCT gênero FROM Acervo;
+————————-+
| gênero |
+————————-+
| Romance |
| Ação |
| Comédia |
| Terror |
| ficção científica |
+————————-+
5 rows in set (0,17 sec)

3.8.5 Utilizando múltiplas tabelas

A tabela Acervo mantém informações de quais filmes você tem. Porém, uma vídeo-locadora necessita de informações sobre seus clientes. Informações como:

* Nome do cliente.
* Uma data para que você saiba quando ocorreu a locação.
* Telefone residencial.
* Endereço.
* Filme locado (referência a tabela Acervo)

A instrução CREATE TABLE para a tabela Cliente deve se parecer com isto:

mysql> CREATE TABLE Cliente(cod_cliente int AUTO_INCREMENT,
nome VARCHAR(20),
telefone VARCHAR(20),
endereço VARCHAR(50),
filme_locado INT,
data_locação DATE,
PRIMARY KEY (cod_cliente),
FOREIGN KEY (filme_locado) REFERENCES Acervo(cod_filme) ON DELETE RESTRICT)
ENGINE = InnoDB;

Observe que o tipo da nova tabela é InnoDB. Caso a tabela Acervo não seja deste mesmo tipo ocorrerá um erro. Para alterar o tipo da tabela Acervo utilize a seguinte instrução:

ALTER TABLE Acervo TYPE = InnoDB;

Como na tabela Acervo, é mais fácil carregar os registros iniciais criando um arquivo texto delimitado por tabulações contendo a informação, porém fica a cargo do administrador escolher o método de inserção de dados.

Carregue os registros:

mysql> LOAD DATA INFILE "cliente.txt"INTO TABLE cliente;

Neste ponto você já está apto a realizar consultas em uma única tabela. No exemplo da vídeolocadora você já possui duas tabelas, e em alguns casos será necessário fazer múltiplas consultas. Vamos aprender como realizar estes tipos de consulta. Realize uma busca na tabela Cliente para observar o seu formato.

mysql> SELECT * FROM Cliente;

```
+————————+———+———————+———————————+————————+————————+
| cod_cliente | nome | telefone | endereço | filme_locado | data_locação |
+————————+———+———————+———————————+————————+————————+
| 1 | João | 351-1551 | SQS 414 Bl. B apt 103 | 1 | 2006-06-24 |
| 2 | João | 351-1551 | SQS 414 Bl. B apt 103 | 1 | 2006-04-25 |
| 3 | Pedro | 344-1551 | SQS 415 Bl. A apt 108 | 1 | 2006-05-25 |
| 4 | Ana | 328-1451 | SQS 413 Bl. A apt 415 | 2 | 2006-05-25 |
| 5 | Maria | 328-1451 | SQS 413 Bl. A apt 415 | 3 | 2006-05-25 |
| 6 | Marcio | 328-1451 | SQS 413 Bl. A apt 415 | 4 | 2006-06-24 |
+————————+———+———————+———————————+————————+————————+
6 rows in set (0,00 sec)
```

Observe que o campo filme_locado não mostra o nome do filme, e sim uma identificação relativa a outra tabela. Para realizar uma consulta que busque o nome dos filmes locados pelos clientes em vez deste identificador utiliza-se a seguinte instrução:

mysql> SELECT Cliente.nome,Cliente.telefone,Acervo.nome FROM Acervo,Cliente WHERE Cliente.filme_locado=Acervo.cod_filme;

```
+———+———————+———————————+
| nome | telefone | nome |
+———+———————+———————————+
| João | 351-1551 | Titanic |
| João | 351-1551 | Titanic |
| Pedro | 344-1551 | Titanic |
| Ana | 328-1451 | Código da Vinci |
| Maria | 328-1451 | Todo mundo em Pânico |
| Marcio | 328-1451 | O chamado |
+———+———————+———————————+
6 rows in set (0,11 sec)
```

Existem várias coisas que devem ser percebidas sobre esta consulta: A cláusula FROM lista as duas tabelas porque a consulta precisa extrair informação de ambas.
Para combinar informações de múltiplas tabelas, você precisa especificar se os registros em uma

tabela coincidem com os registros em outra (WHERE Cliente.filme_locado = Acervo.cod_filme). Como a coluna nome ocorre em ambas tabelas, você deve especificar qual a tabela a que você está se referindo. Isto é feito usando o nome da tabela antes do nome da coluna separados por um ponto (.).

A consulta acima poderia ser feita da seguinte maneira:

SELECT A.nome,A.telefone,B.nome FROM Acervo as B,
Cliente as A WHERE A.filme_locado=B.cod_filme;

Nesta consulta, nós criamos apelidos para os nomes das tabelas chamando Acervo de B (Acervo as B) e Cliente de A (Cliente as A).

3.9 Lição 7 - A cláusula ORDER BY

3.9.1 Ordenando Registros

Durante uma query (busca), geralmente é necessário que as informações sejam mostradas de um forma organizada. Nos exemplos anteriores os valores eram retornados seguindo a ordem de inserção na tabela. A partir de agora será utilizada a cláusula ORDER BY para ordenar os arquivos. No exemplo abaixo é feita uma busca nas tabelas "Cliente"e "Acervo"retornando os campos ordenados pela data de locação:

mysql> SELECT A.nome,A.telefone,B.nome AS filme,A.data_locação FROM Cliente as A, Acervo as B WHERE A.filme_locado = B.cod_filme ORDER BY data_locação;

```
+------+----------+-----------------+-----------+
| nome | telefone | filme           | data_locação |
+------+----------+-----------------+-----------+
| João | 351-1551 | Titanic         | 2006-04-25 |
| Pedro | 344-1551 | Titanic         | 2006-05-25 |
| Ana | 328-1451 | Código da Vinci | 2006-05-25 |
| Maria | 328-1451 | Todo mundo em Pânico | 2006-05-25 |
| João | 351-1551 | Titanic         | 2006-06-24 |
| Marcio | 328-1451 | O chamado       | 2006-06-24 |
+------+----------+-----------------+-----------+
6 rows in set (0,01 sec)
```

Em campos de texto a comparação entre os campos é feita no modo caso insensitivo. Para forçar uma busca que diferencie maiúsculas de minúsculas utilize a coerção BINARY: ORDER BY BINARY(campo).

OBS: Percebam que apelidamos a coluna B.nome de filme **(B.nome AS filme)** . Se isto não fosse feito duas colunas chamadas "nome"iriam aparecer.

3.9.2 Ordenando de forma decrescente

Como foi observado no exemplo anterior, a query retornou os campos ordenados de forma crescente. Para ordernar de forma decrescente utilize a palavra chave DESC ao nome da coluna que será ordenada.

Exemplo: Busca feita nas tabelas "Cliente"e "Acervo", ordenada pelo campo "filme".

mysql> SELECT A.nome,A.telefone,B.nome AS filme,A.data_locação FROM Cliente as A, Acervo as B WHERE A.filme_locado = B.cod_filme ORDER BY filme DESC;

```
+———+———-+——————-+————+
| nome | telefone | filme | data_locação |
+———+———-+——————-+————+
| Maria | 328-1451 | Todo mundo em Pânico | 2006-05-25 |
| João | 351-1551 | Titanic | 2006-06-24 |
| João | 351-1551 | Titanic | 2006-04-25 |
| Pedro | 344-1551 | Titanic | 2006-05-25 |
| Marcio | 328-1451 | O chamado | 2006-06-24 |
| Ana | 328-1451 | Código da Vinci | 2006-05-25 |
+———+———-+——————-+————+
6 rows in set (0,00 sec)
```

OBS: Como apelidados "B.nome"de "filme", foi possível apenas utilizar a referência filme na cláusula ORDER BY.

3.9.3 Ordenando por múltiplas colunas

É possível utilizar a cláusula ORDER BY em múltiplas colunas para ordenar seguindo padrões diferentes. Exemplificando, podemos realizar uma busca nas tabelas "Acervo"e "Cliente"de tal forma que o campo "nome"seja ordenado de forma crescente e o campo "data_locação"seja ordenado de forma decrescente.

mysql> SELECT A.nome,A.telefone,B.nome AS filme,A.data_locação FROM Cliente as A, Acervo as B WHERE A.filme_locado = B.cod_filme ORDER BY A.nome, A.data_locação DESC;

```
+———+———-+——————-+————+
| nome | telefone | filme | data_locação |
+———+———-+——————-+————+
| Ana | 328-1451 | Código da Vinci | 2006-05-25 |
| João | 351-1551 | Titanic | 2006-06-24 |
| João | 351-1551 | Titanic | 2006-04-25 |
| Marcio | 328-1451 | O chamado | 2006-06-24 |
| Maria | 328-1451 | Todo mundo em Pânico | 2006-05-25 |
| Pedro | 344-1551 | Titanic | 2006-05-25 |
+———+———-+——————-+————+
6 rows in set (0,00 sec)
```

Observe que a palavra chave DESC é aplicada somente à coluna data_locação, ela não afeta a ordenação da coluna nome.

3.10 Lição 8 - Funções pré-definidas

3.10.1 Cálculo de Datas

Existe uma gama de funções disponibilizadas pelo MySQL que facilitam a vida do administrador do banco. Funções como SUM(), CURDATE(), AVG() e muitas outras serão explicadas nesta lição.
Para determinar, por exemplo, quanto tempo se passou (em dias) desde a última locação de um cliente, podemos utilizar a seguinte instrução:

```
mysql> SELECT nome,telefone,CURDATE() AS "Data atual",data_locação,DATEDIFF(CURDATE(),data_
as "Tempo decorrido"FROM Cliente;
+———+———-+————+————+————————+
| nome | telefone | Data atual | data_locação | Tempo decorrido |
+———+———-+————+————+————————+
| João | 351-1551 | 2006-07-11 | 2006-06-24 | 17 |
| João | 351-1551 | 2006-07-11 | 2006-04-25 | 77 |
| Pedro | 344-1551 | 2006-07-11 | 2006-05-25 | 47 |
| Ana | 328-1451 | 2006-07-11 | 2006-05-25 | 47 |
| Maria | 328-1451 | 2006-07-11 | 2006-05-25 | 47 |
| Marcio | 328-1451 | 2006-07-11 | 2006-06-24 | 17 |
+———+———-+————+————+————————+
6 rows in set (0,00 sec)
```

Podemos observar neste exemplo a utilização de duas funções, CURDATE() e DATEDIFF. A primeira retorna o valor da data atual (data do sistema) e a segunda retorna a diferença, em dias, entre duas datas. Observem também que utilizamos a função CURDATE() como parâmetro para a função DATEDIFF(). Podemos fazer isto pois o retorno da função CURDATE() é do tipo "date".

Para verificar qual pessoa está a mais tempo sem alugar um filme, podemos realizar a seguinte query.

```
mysql> SELECT nome,telefone,CURDATE() AS "Data atual",data_locação,DATEDIFF(CURDATE(),data_
as "Tempo decorrido"FROM Cliente ORDER BY "Tempo decorrido"DESC LIMIT 1;
+———+———-+————+————+————————+
| nome | telefone | Data atual | data_locação | Tempo decorrido |
+———+———-+————+————+————————+
| João | 351-1551 | 2006-07-11 | 2006-04-25 | 77 |
+———+———-+————+————+————————+
1 row in set (0,05 sec)
```

Observem que utilizamos a cláusula ORDER BY para ordenar os campos de forma decrescente em relação ao Tempo decorrido. Uma nova cláusula foi apresentada. LIMIT é utilizado para restringir o número de linhas que o servidor retorna para um cliente.

Para retornar ao mês em que ocorreu a locação do filme, pode-se utilizar a função MONTH(). Esta função extrai o mês de uma campo do tipo date.

```
mysql> SELECT nome,data_locação,MONTH(data_locação) AS "Mês"FROM Cliente
; +———+————————+———+
| nome | data_locação | Mês |
+———+————————+———+
| João | 2006-06-24 | 6 |
| João | 2006-04-25 | 4 |
| Pedro | 2006-05-25 | 5 |
| Ana | 2006-05-25 | 5 |
| Maria | 2006-05-25 | 5 |
| Marcio | 2006-06-24 | 6 |
+———+————————+———+
6 rows in set (0,00 sec)
```

Encontrar pessoas que alugaram algum filme no mês de junho também é simples:

```
mysql> SELECT nome,data_locação AS "Mês"FROM Cliente WHERE MONTH(data_locação)=6;
+———+————————+
| nome | Mês |
+———+————————+
| João | 2006-06-24 |
| Marcio | 2006-06-24 |
+———+————————+
2 rows in set (0,01 sec)
```

Existe uma infinidade de funções para manipulação de campos date/time. Caso desejem conheçer novas funções acessem http://dev.mysql.com/doc/refman/4.1/pt/date-and-time-functions.html

3.10.2 Contando Registros / GROUP BY

Geralmente precisamos saber com que tipo de frequência determinados dados apareçem. Seguindo o exemplo da Locadora, caso fosse necessário saber o número de filmes disponíveis no acervo, a função COUNT() ajudaria nesta tarefa. Podemos utilizar a função COUNT() para realizar uma série de pesquisas estatísticas, como veremos agora.

Saber o número de filmes disponíveis na locadora é o mesmo que saber o número de registros existentes na tabela "Acervo". A seguinte busca retornará o número de registros das tabelas.

```
mysql> SELECT COUNT(*) FROM Acervo;
```

```
+———-+
| COUNT(*) |
+———-+
| 6 |
+———-+
1 row in set (0,00 sec)
```

Para retornar ao número de filmes por gênero:

```
mysql> SELECT gênero,COUNT(*) FROM Acervo GROUP BY gênero;
+————————-+———-+
| gênero | COUNT(*) |
+————————-+———-+
| Ação | 1 |
| Comédia | 2 |
| ficção científica | 1 |
| Romance | 1 |
| Terror | 1 |
+————————-+———-+
5 rows in set (0,01 sec)
```

Perceba o uso de GROUP BY para agrupar todos os registros para cada gênero. Sem o GROUP BY seria retornado o seguinte erro:

```
mysql> SELECT gênero,COUNT(*) FROM Acervo;
ERROR 1140 (42000): Mixing of GROUP columns (MIN(),MAX(),COUNT(),...) with no GROUP
columns is illegal if there is no GROUP BY clause
```

COUNT() e GROUP BY podem ser utilizados para realizar uma infinidade de buscas. Por exemplo, para realizar uma pesquisa que retorne quantas vezes um filme já foi alugado:

```
mysql> SELECT A.nome AS "Nome do Filme",COUNT(*) FROM Acervo AS A, Cliente AS
B WHERE A.cod_filme = B.filme_locado GROUP BY B.filme_locado;
+———————————-+———-+
| Nome do Filme | COUNT(*) |
+———————————-+———-+
| Titanic | 3 |
| Código da Vinci | 1 |
| Todo mundo em Pânico | 1 |
| O chamado | 1 |
+———————————-+———-+
4 rows in set (0,09 sec)
```

Observem que foram utlizadas duas tabelas. A tabela Acervo contém o nome do filme e a tabela Cliente o campo filme_locado, que foi agrupado para retornar o número de vezes que este filme havia sido alugado.

Para retornar todas as pessoas da tabela Cliente que têm telefone com prefixo 351, utilizamos a seguinte instrução:

```
mysql> SELECT nome,telefone FROM Cliente WHERE telefone LIKE "351%";
+——+———-+
| nome | telefone |
+——+———-+
| João | 351-1551 |
| João | 351-1551 |
+——+———-+
2 rows in set (0,00 sec)
```

Devemos observar um ponto importante. A utilização do comparador LIKE. Utilizamos o LIKE para comparar uma string que possa ter campos ainda não explicitados na consulta. O "%"serve justamente para isto. Esta busca retorna todos os telefones com início 351 e o "%"indica que o final da string pode ser qualquer coisa.

Agora para retornar o número de pessoas que têm telefone com prefixo 351 basta utilizar a função COUNT().

```
mysql> SELECT COUNT(*) AS "Prefixo 351"FROM Cliente WHERE telefone LIKE "351%";
+————-+
| Prefixo 351 |
+————-+
| 2 |
+————-+
1 row in set (0,00 sec)
```

Como mostrado acima, não precisamos contar todos os campos da tabela:

```
mysql> SELECT data_locação,COUNT(*) FROM Cliente WHERE data_locação = "2006:05:25"GROUP
BY data_locação;
+————-+———-+
| data_locação | COUNT(*) |
+————-+———-+
| 2006-05-25 | 3 |
+————-+———-+
1 row in set (0,00 sec)
```

3.10.3 Outras Funções básicas

Funções básicas

O MySQL possui uma série de funções que podem ser utilizadas para auxiliar consultas, abaixo listamos algumas bastante conhecidas. Para maiores informações, acesse o manual do MySQL no site www.mysql.org.

AVG(Média): Retorna à média de uma coluna;
exemplos:

```
mysql> SELECT AVG(censura) FROM Acervo;
+————————+
| AVG(censura) |
+————————+
| 11.0000 |
+————————+
1 row in set (0,11 sec)
```

SUM(Soma): Retorna ao somatório das colunas;
exemplos:

```
mysql> SELECT SUM(censura) FROM Acervo;
+————————+
| SUM(censura) |
+————————+
| 66 |
+————————+
1 row in set (0,00 sec)
```

MIN(Mínimo): Retorna ao menor valor para uma coluna;
exemplos:

```
mysql> SELECT MIN(censura) FROM Acervo;
+————————+
| MIN(censura) |
+————————+
| 0 |
+————————+
1 row in set (0,00 sec)
```

MAX(Máximo): Retorna ao valor máximo para uma coluna;

```
mysql> SELECT MAX(censura) FROM Acervo;
+————————+
| MAX(censura) |
+————————+
| 14 |
+————————+
1 row in set (0,01 sec)
```

3.11 Lição 9 - Conectando e Desconectando do Servidor

3.11.1 Configurando os Privilégios Iniciais do MySQL

Após instalar o MySQL executamos dois scripts básicos. Um deles é o script mysql_install_db que cria as tabelas de concessões e o outro é o mysqld_safe para iniciar o servidor.
Ao executar o script mysql_install_db, um banco de dados chamado mysql é criado, e dentro deste banco existe uma tabela chamada user. O usuário root é criado durante a execuação deste script.
Para verificar o Conteúdo da tabela user realize a seguinte busca:

```
mysql> SELECT Host,User,Password FROM user;
+———–+——+———-+
| Host | User | Password |
+———–+——+———-+
| localhost | root | |
| debian | root | |
| debian | | |
| localhost | | |
+———–+——+———-+
4 rows in set (0,00 sec)
```

O usuário root é o super-usuário, podendo alterar qualquer coisa no sistema. Ele tem permissão de acesso local, não podendo acessar o banco remotamente. Deve-se atentar para o fato de que a senha inicial de root é vazia, e qualquer pessoa terá acesso ao banco.

Um usuário anônimo também é criado. Este usuário tem acesso ao banco de dados "test". Os usuário locais do sistema podem acessar o MySQL como este usuário anônimo.

Deixar a senha de root vazia não é uma boa coisa. Após a instalação, a primeira coisa que devemos fazer é trocar a senha de root. Veremos três maneiras de modificar esta senha.

1) Utilizando a função PASSWORD

* **shell> mysql -u root mysql**
* **mysql> SET PASSWORD FOR root@localhost=PASSWORD('nova_senha');**

Substitua 'nova_senha' pela senha que você deseja usar.

2)Você também pode manipular diretamente a tabela privilégios:

* **shell> mysql -u root mysql * mysql> UPDATE user SET Password=PASSWORD('nova_senha')**
* **-> WHERE user='root'; * mysql> FLUSH PRIVILEGES;**

FLUSH PRIVILEGES diz ao servidor para reler as tabelas de permissões.

3)Outra forma de configurar a senha é utilizando o comando mysqladmin:

*** shell> mysqladmin -u root password nova_senha**

De agora em diante a senha deverá ser especificada para se ter acesso ao servidor.

Para recriar as tabelas de permissões completamente, remova todos os arquivos .frm .MYI e .MYD no diretório contendo o banco de dados mysql. Caso o MySQL tenha sido instalado no debian através do APT este banco se encontra em /var/lib/mysql. Caso a instalação tenha sido feita a partir do código-fonte, este banco se encontra no diretório raiz do MySQL, dentro da pasta var.

3.11.2 Conectando e Desconectando do Servidor

Para conectar ao MySQL utilizamos o seguinte comando:

shell> mysql -h servidor -u usuario -p
Enter password: ********

Neste comando, -h indica a localização onde o servidor MySQL está instalado. Caso você esteja acessando localmente, este parâmetro pode ser omitido. -u indica o usuário e -p a senha. Percebam que a senha não foi passada como parâmetro. A senha será requisitada logo em seguida.

shell> mysql -h host -u user -p
Enter password: ********
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 5.0.22

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

mysql>

A senha pode ser passada como parâmetro da seguinte maneira:
shell> mysql -h host -u user -psuasenha

A senha deve ser colocada junto ao argumento -p. Caso algum espaço seja colocado, na verdade a senha será dada como o banco que seria iniciado junto ao MySQL.

Para se conectar como usuário anônimo basta digitar o comando:
shell> mysql

Para disconectar basta utilizar a instrução QUIT (ou \q) no prompt mysql>:
mysql> QUIT
Bye

3.12 Licão 10 - Adicionando Novos Usuários ao MySQL

3.12.1 Um pouco sobre as intruções REVOKE E GRANT

Grant e Revoke são duas instruções com uma sintaxe um pouco complicada, nesta lição não iremos nos aprofundar muito na sintaxe das intruções. Será apenas uma breve introdução.

Os comandos GRANT e REVOKE permitem aos administradores do sistema criar usuários e conceder e revogar direitos aos usuários do MySQL em quatro níveis de privilégios. Sua sintaxe e dada por:

GRANT "Privilégios"ON "banco_de_dados"TO "usuario@servidor"IDENTIFIED BY "alguma_senha"

REVOKE "Privilégios"
ON "banco_de_dados"
FROM "usuario"

Podem ser passados mais argumentos a instrução GRANT. Para maiores informações, pode ser acessado o site com a documentação do MySQL, www.mysql.org.

Os níveis de privilégios são:

Nível Global:

Este tipo nível de privilégio é aplicado a todos os bancos do servidor.
Tabelas onde os previlégios são armazenados: mysql.user.
Sintaxe: GRANT ALL ON *.* e REVOKE ALL ON *.*

Nível dos bancos de dados:

Neste nível, os privilégios são aplicados à todas as tabelas de um determinado banco de dados.
Tabelas onde os privilégios são armazenados: mysql.db e mysql.host.
Sintaxe: GRANT ALL ON db.* e REVOKE ALL ON db.*

Nível das tabelas:

Neste nível, os privilégios são aplicados à todas as colunas de uma determinada tabela.
Tabelas onde os privilégios são armazenados: mysql.tables_priv.
Sintaxe: GRANT ALL ON db.table e REVOKE ALL ON db.table

Nível das colunas:

Privilégios de colunas aplicam-se a uma única coluna em uma determinada tabela. Tabelas onde os privilégios são armazenados: mysql.columns_priv.

Para as instruções GRANT e REVOKE, os previlégios podem ser especificados como:
* ALL [PRIVILEGES] : Configura todos os privilégios simples exceto WITH GRANT OPTION

* ALTER: Permite o uso de ALTER TABLE
* CREATE: Permite o uso de CREATE TABLE
* CREATE TEMPORARY TABLES: Permite o uso de CREATE TEMPORARY TABLE
* DELETE: Permite o uso de DELETE
* DROP: Permite o uso de DROP TABLE.
* EXECUTE: Permite que o usuário execute stored procedures (MySQL 5.0)
* FILE: Permite o uso de SELECT ... INTO OUTFILE e LOAD DATA INFILE.
* INDEX: Permite o uso de CREATE INDEX e DROP INDEX
* INSERT: Permite o uso de INSERT
* LOCK TABLES: Permite o uso de LOCK TABLES em tabelas nas quais se tem o privilégio SELECT.
* PROCESS: Permite o uso de SHOW FULL PROCESSLIST
* REFERENCES: Para o futuro
* RELOAD: Permite o uso de FLUSH
* REPLICATION CLIENT: Dá o direito ao usuário de perguntar onde o slave/master está.
* REPLICATION SLAVE: Necessário para a replicação dos slaves (para ler logs binário do master).
* SELECT: Permite o uso de SELECT
* SHOW DATABASES: SHOW DATABASES exibe todos os banco de dados.
* SHUTDOWN: Permite o uso de mysqladmin shutdown
* SUPER: Permite a conexão (uma vez) mesmo se max_connections tiverem sido alcançados e executa o comando CHANGE MASTER, KILL thread, mysqladmin debug, PURGE MASTER LOGS e SET GLOBAL
* UPDATE: Permite o uso de UPDATE
* USAGE: Sinônimo para “sem privilégios.”
* GRANT OPTION: Sinônimo para WITH GRANT OPTION
* USAGE: pode ser usado quando você quer criar um usuário sem privilégios.

3.12.2 Nomes de Usuários e Senhas do MySQL

Existem alguns fatos que devemos atentar quando estamos falando de Nomes de Usuários utilizados pelo MySQL.

* Os nomes de usuários do MySQL não têm nenhuma ligação com os usuários do sistema operacional.
* Nomes de usuários MySQL podem ter o tamanho de até 16 caracteres; Nomes de usuário Unix normalmente são limitados até 8 caracteres.
* Senhas MySQL não tem nenhuma relação com senhas do sistema operacional.
* O MySQL criptografa senhas utilizando um algorítimo diferente que o utilizado pelo processo de login do Unix.

O MySQL é seguro contra sniffs (programas que capturam senha pela rede) e até mesmo quanto a captura do banco "mysql".

Abaixo estão listadas diversas formas de passar a senha como parâmetro para o servidor:

*** mysql –user=usuario –password=senha nome_do_banco**

*** mysql –user=usuario –password nome_do_banco**
*** mysql -u usuario -p nome_do_banco**

Perceba que nos dois últimos exemplos a senha não é 'nome_do_banco'.

3.12.3 Adicionando Novos Usuários ao MySQL

Vamos aprender duas maneiras de adicionar usuários no mysql, utilizando instruções GRANT ou manipulando as tabelas de permissões do MySQL diretamente. O método preferido é utilizar instruções GRANT, porque elas são mais concisas e menos propensas a erros.

Existem vários programas de colaboradores (como o phpMyAdmin) que podem ser utilizados para criar e administrar usuários.

Acesse o MySQL como root:

shell> mysql –user=root mysql

Abaixo estão três exemplos de como adicinoar usuários com instruções GRANT:
Usuário 1:
mysql> GRANT ALL PRIVILEGES ON *.* TO usuario1@localhost
IDENTIFIED BY 'alguma_senha' WITH GRANT OPTION;
mysql> GRANT ALL PRIVILEGES ON *.* TO usuario1@'%'
IDENTIFIED BY 'alguma_senha' WITH GRANT OPTION;

Usuário 2:
mysql> GRANT RELOAD,PROCESS ON *.* TO usuario2@localhost;

Usuário 3:
mysql> GRANT USAGE ON *.* TO usuario3@localhost;

Estas instruções GRANT configuram três novos usuários:

*** usuario1:**

Um superusuário completo que pode conectar no servidor de qualquer lugar e possui todos os privilégios.

*** usuario2:**

Um usuário que possa conectar da máquina local sem uma senha e que é concedido os privilégios administrativos reload e process.

*** usuario3**

Um usuário que pode conectar sem uma senha, mas somente na máquina local. Não é concedido nenhum privilégio, o tipo de privilégio USAGE permite a criação de um usuário sem privilégios.

Também é possível adicionar a mesma informação de acesso do usuário diretamente, utilizando instruções INSERT e depois dizendo ao servidor para recarregar as tabelas de permissões:

```
shell> mysql –user=root mysql
mysql> INSERT INTO user VALUES('localhost','usuario1',PASSWORD('alguma_senha'),
'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO user VALUES('%','usuario1',PASSWORD('alguma_senha'),
'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');

mysql> INSERT INTO user SET Host='localhost',User='usuario2',
Reload_priv='Y', Process_priv='Y';

mysql> INSERT INTO user (Host,User,Password)
VALUES('localhost','usuario3','');
mysql> FLUSH PRIVILEGES;
```

Para configurar o superusuário, você só precisar criar uma entrada na tabela user com os campos de privilégios configurados para 'Y'.

Na última instrução INSERT (para o usuário dummy), apenas as colunas Host, User e Password nos registros da tabela user tem valores atribuídos. O MySQL atribui a todas as outras colunas o valor padrão de 'N'.

O exemplo abaixo adiciona um usuário com senha 'novasenha', e que tem permissões de acesso ao banco CDTC apenas da máquina local, ao banco ITI apenas do host cdtc.gov.br:

```
shell> mysql –user=root mysql
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
-> ON CDTC.*
-> TO usuario@localhost
-> IDENTIFIED BY 'novasenha';

mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
-> ON ITI.*
-> TO usuario@'cdtc.gov.br'
-> IDENTIFIED BY 'obscure';
```

Se você deseja fornecer a um usuário específico acesso de qualquer máquina em um determinado domínio (por exemplo, meudomínio.com), você pode utilizar uma instrução GRANT como a seguir:

```
mysql> GRANT ...
-> ON *.*
-> TO meusername@'%.mydomain.com'
-> IDENTIFIED BY 'mypassword';
```

Para realizar a mesma coisa modificando diretamente as tabelas de permissões, faça isto:

mysql> INSERT INTO user VALUES ('%.meudominio, 'meunomedeusuario'
PASSWORD('minhasenha'),...);
mysql> FLUSH PRIVILEGES;

Deletando Usuários do MySQL

DROP USER nome_usuario
Este comando foi adicionado ao MySQL 4.1.1.
Ele apaga um usuário que não possua nenhum privilégio.

3.13 Lição 11 - Ferramentas Gráficas de Adminstração

3.13.1 Conheçendo algumas ferramentas

Como podemos observar administrar contas de usuário por linhas de comando é muito complicado. Para facilitar esta tarefa, existem ferramentas gráficas que auxiliam na administração do Mysql. Mas vale ressaltar, que quando utilizamos estas ferramentas estamos sujeitos a falhas de segurança, sendo de responsabilidade do administrador verificar o melhor método a ser utilizado.

Existem várias ferramentas gráficas que auxiliam nesse processo. Podemos citar como exemplo:

* MySQL-Front: MySQL-Front é um front-end gráfico para a base de dados do MySQL. Porque é uma aplicação "real", pode oferecer uma interface com o usuário mais refinada do que é possível com os sistemas construídos em PHP e em HTML.
* PhpMyAdmin: é um script em PHP para gerenciamento do MySQL (a mesma coisa que faz o MySQLFront), só que geralmente é usado nos servidores por ser PHP e roda no Browser do usuário.
* MySQL-Admin: Mais uma ferramenta para administração. Com uma interface bem amigável, nos permite excluir e adicionar usuários, parar o servidor, configurar arquivos como o mysqld_safe e muitas outras opções.

Neste curso aprendermos a utilizar algumas funções do PhpMyAdmin.

3.13.2 Instalando o PhpMyAdmin

A instalação do PhpMyAdmin é muito simples, uma vez que temos os caminhos para os repositórios configurados corretamente a instalação é feita da seguinte maneira:

*** shell> apt-get install PhpMyAdmin**

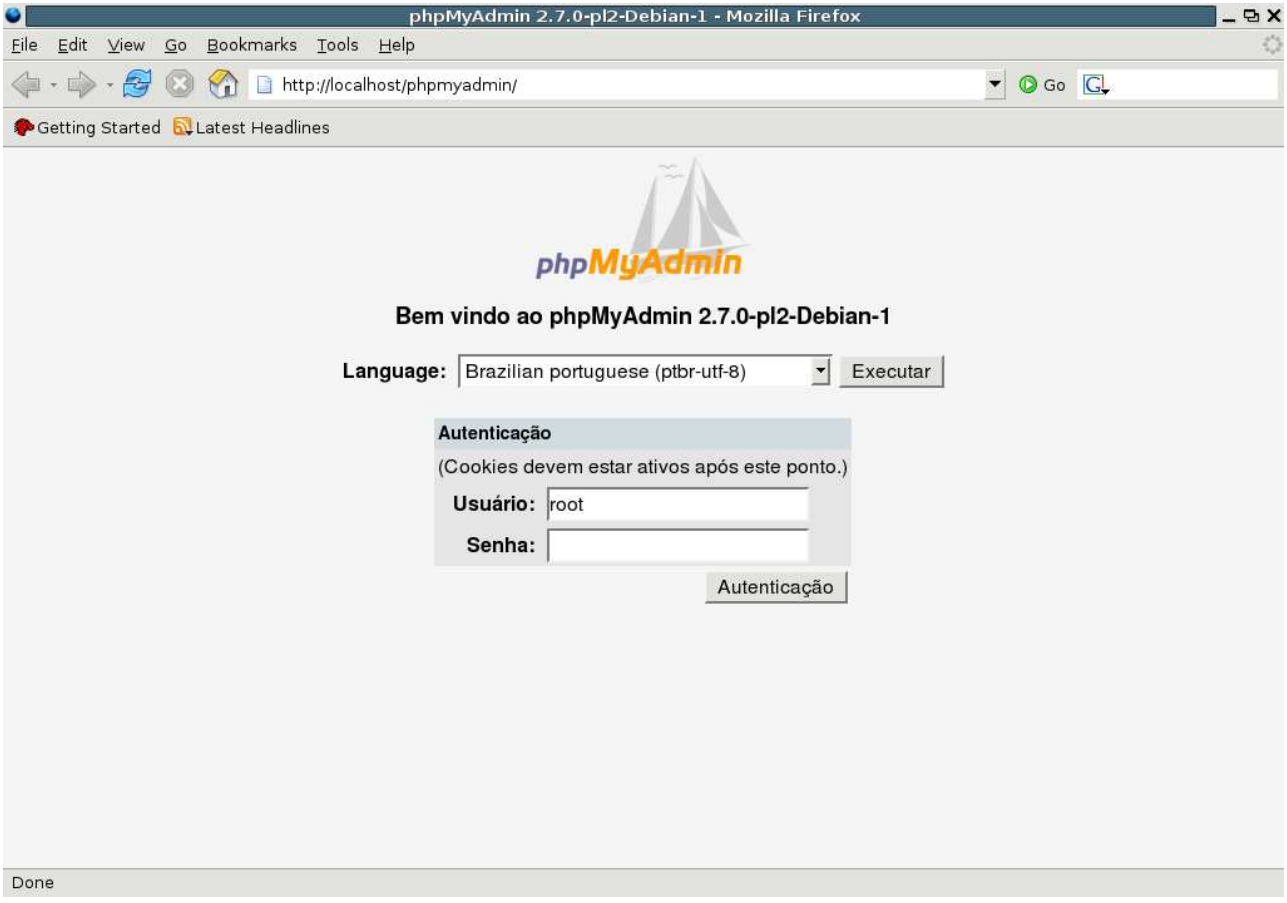
Basta responder sim para algumas perguntas e pronto, estamos com um front-end para mysql instalado.

Também será necessário instalar o Apache, porém esse processo é automatizado.

3.13.3 Acessando o PhpMyAdmin

Depois de instalado, acessamos o PhpMyAdmin pelo navegador digitando my-domain/phpmyadmin/ na barra de endereço, onde my-domain é o endereço do servidor MySQL.

Caso o servidor esteja na máquina onde você está acessando o PhpMyAdmin, basta digitar localhost/phpmyadmin. Uma tela apareçerá pedindo seu login e senha.

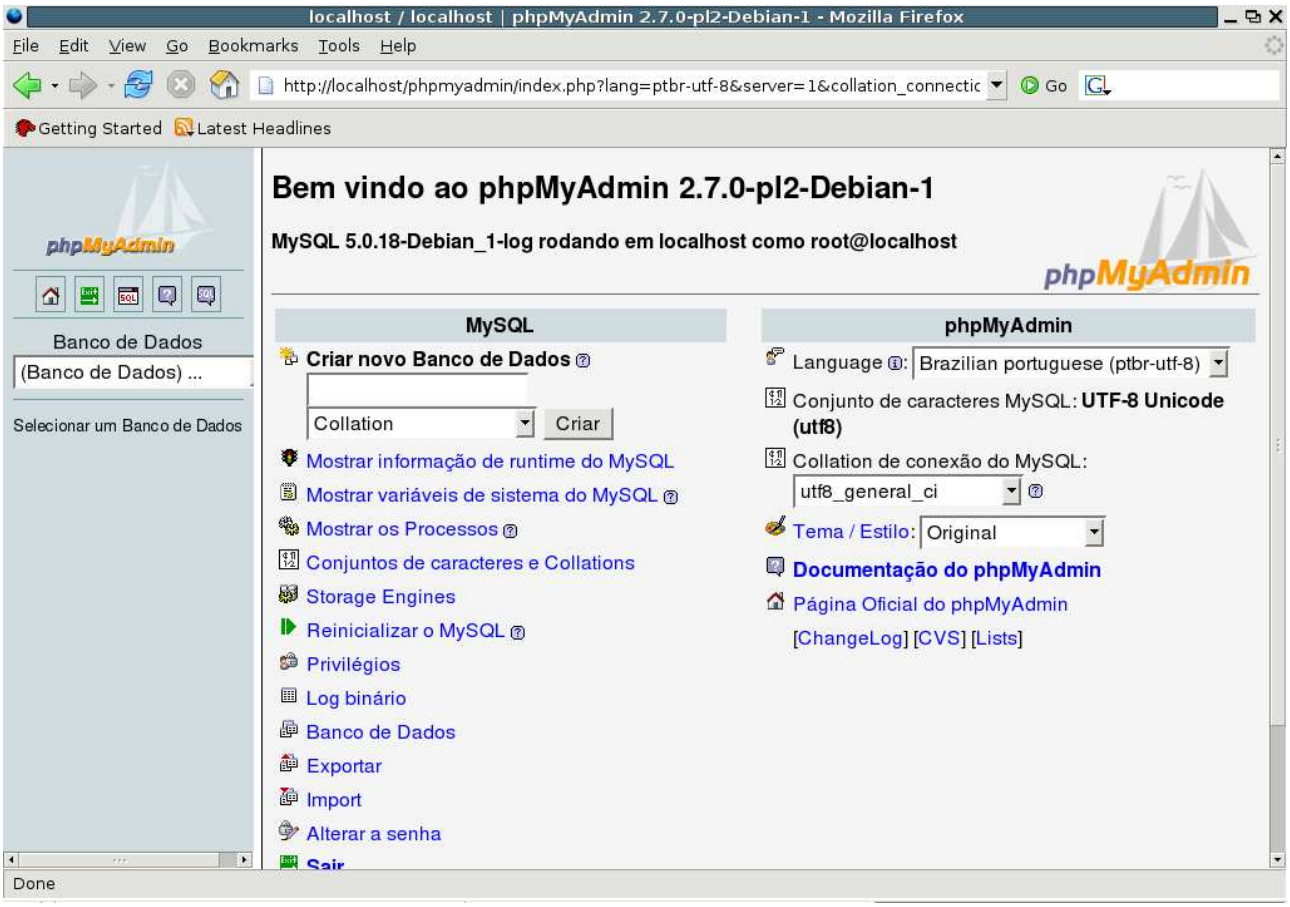


Acesse como administrador, caso o MySQL seja recém instalado o usuário será root e não haverá senha.

3.13.4 Tela de apresentação

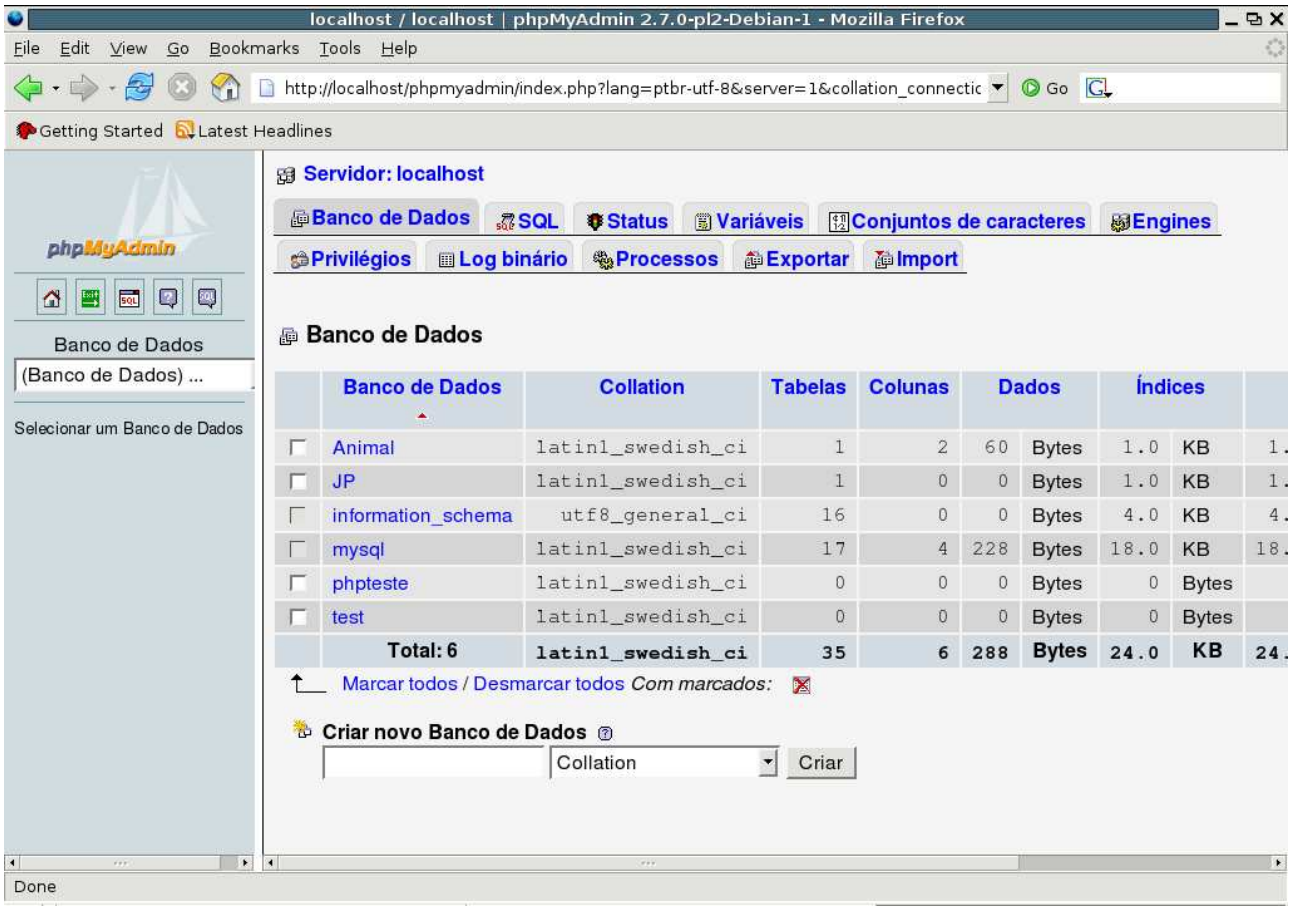
Uma vez que você esteja logado, uma tela do PhpMyAdmin como a mostrada abaixo aparecerá.

Esta seção com vários menus é o local a partir do qual você configurará as principais funções do PhpMyAdmin.

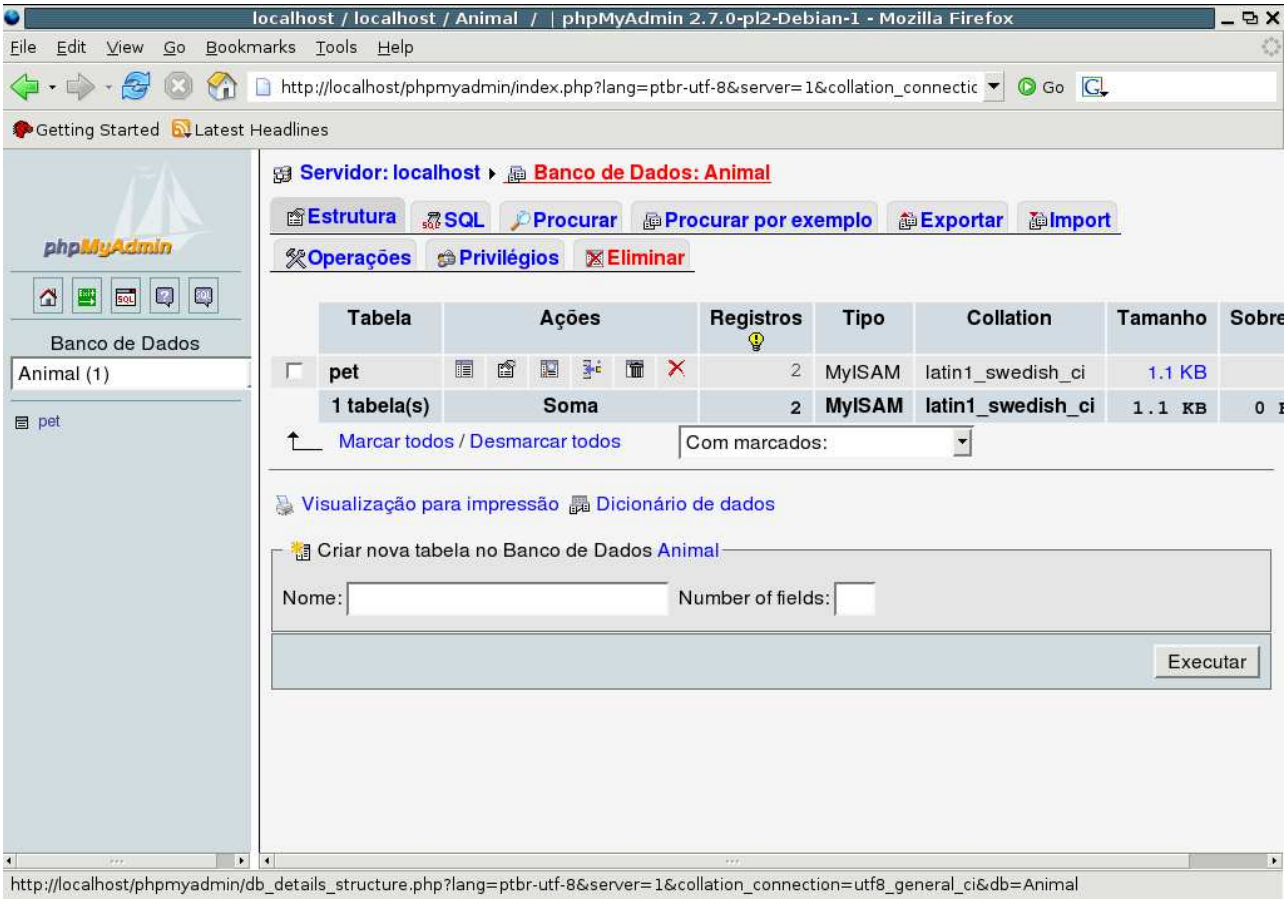


3.13.5 Criando banco de Dados no PhpMyAdmin

Como o foco do curso não é ensinar como utilizar o PhPMyAdmin, apresentaremos somente algumas funções primordiais desta ferramenta. A primeira delas será criar um banco de dados. Cliclando no menu banco de dados seremos redirecionados para seguinte tela:



Agora basta digitarmos o nome do banco de dados a ser criado. Os bancos de dados ja existentes também são apresentados. Para acessá-los basta clicar sobre o nome.

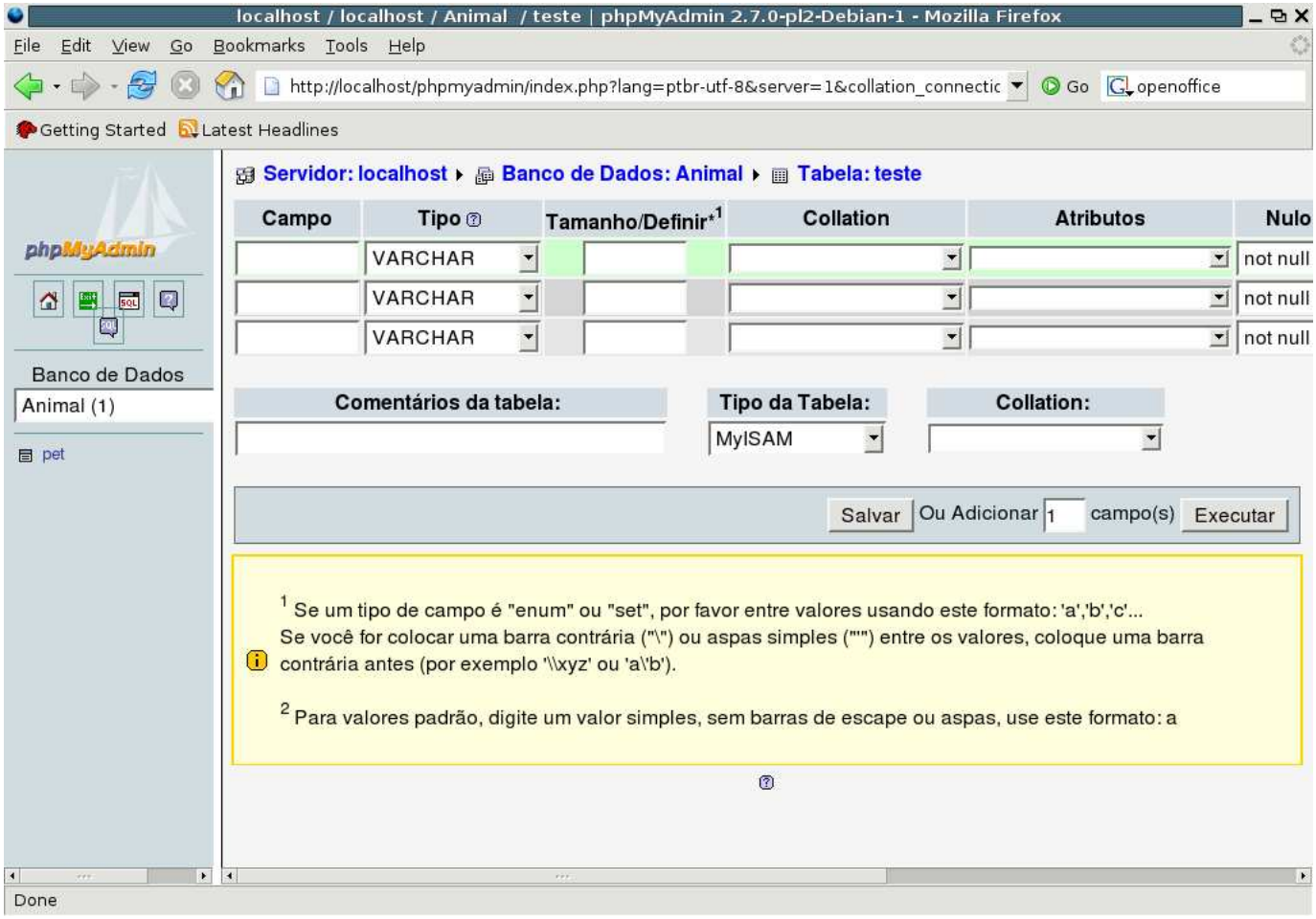


3.13.6 Criando tabelas no PhpMyAdmin

Continuando com o exemplo, para criarmos tabelas também é muito simples.

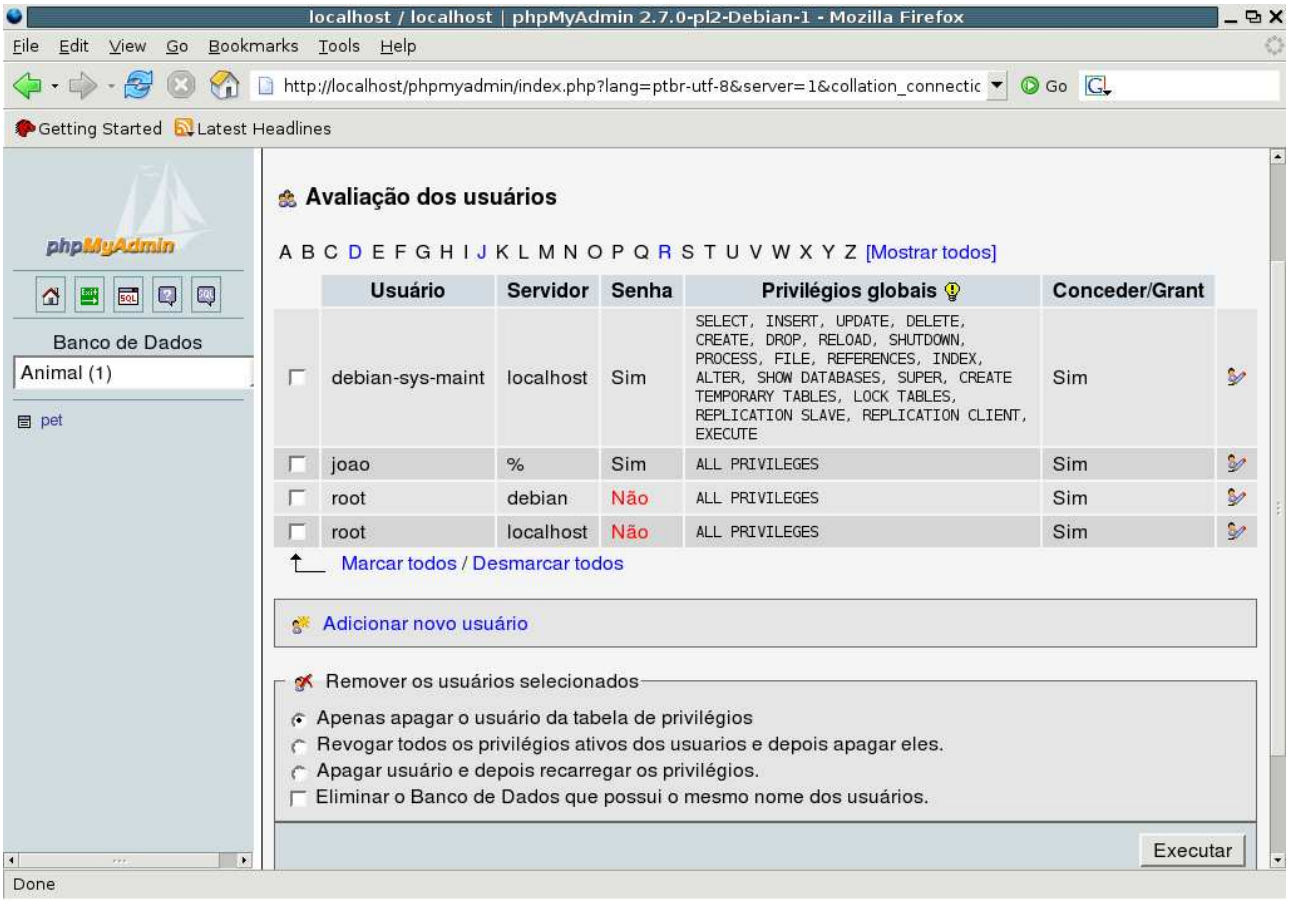
Para criar tabelas dentro do banco de dados selecionado também é bastante intuitivo. Basta digitarmos o nome da tabela a ser criada e o número de campos que a tabela terá.

Agora basta colocar os dados relativos a tabela e clicar em salvar.

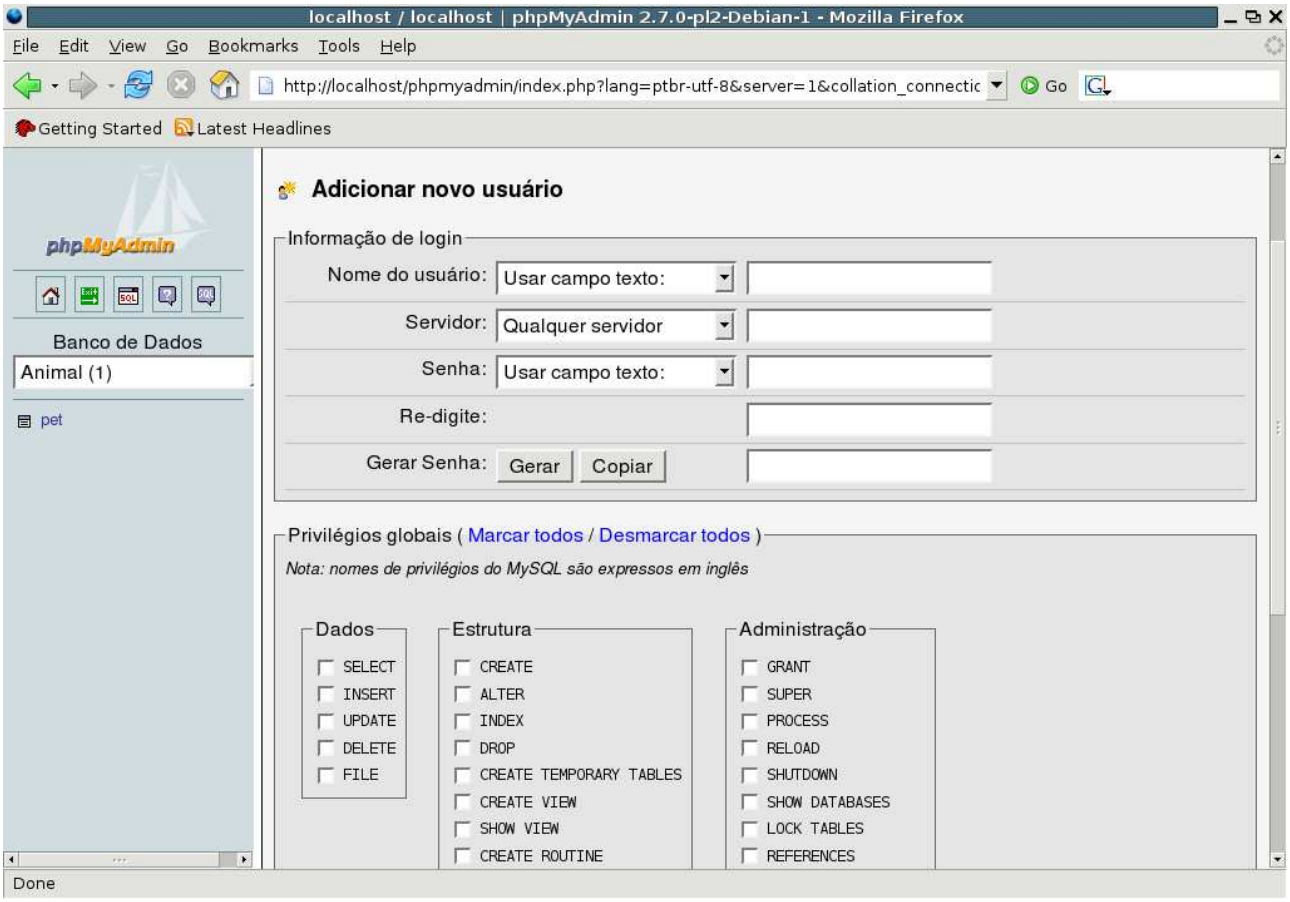


3.13.7 Adicionando Novos Usuários no PhpMyAdmin

Vamos aprender agora como adicionar novos usuários ao banco de dados. Voltemos à tela principal do PhpMyAdmin clicando em servidor:localhost no topo da tela. Agora clicando no menu Privilégios acessamos a seção onde poderemos adicionar e remover usuários.



Para adicionar usuários basta cilicar na aba "Adicionar Novo Usuário"



Digite os dados necessários e as permissões que o novo usuário possuirá. Clique em executar e pronto, temos um novo usuário cadastrado no Servidor MySQL.

Neste ponto finalizamos um breve introdução desta ótima ferramenta de administração. Caso o usuário queira encontrar mais informações sobre o PhpMyAdmin pode acessar a página do projeto em http://www.phpmyadmin.net/home_page/index.php.

3.14 Lição 12 - Stored Procedures e Funções

Stored Procedures e funções, são rotinas a partir de instruções SQL que podem ser armazenadas no servidor MySQL. Para quem já tem experiência na área de programação, as implementações mostradas nesta lição serão de fácil entendimento.

Uma vez que estas rotinas estejam armazenadas no servidor, o administrador do banco de dados não necessitará repeti-las quando for necessário utilizá-las. Este armazenamento de informações é muito útil, visto que o fluxo de informações entre o servidor e o cliente é reduzido. Porém, a carga no sistema do servidor aumenta, já que a maior parte do trabalho é feita no servidor.

Situações onde stored procedures são utilizadas:

* Quando aplicações cliente são escritas em linguagens diferentes
* Operações repetitivas

* Rotinas críticas, quando a segurança é prioritária. Stored procedures fornecem um ambiente que pode assegurar que cada operação seja registrada de forma apropriada.

É necessário que a tabela proc exista no banco de dados mysql. Esta tabela é criada durante a instalação do MySQL 5.0. Caso esta tabela não exista certifique-se de atulalizar as tabelas de concessões.

3.14.1 Sintaxe de Stored Procedure

Para criar Stored procedure e funções utiliza-se as instruções:

* **CREATE PROCEDURE e CREATE FUNCTION**

Um procedimento é chamado usando uma instrução CALL e só pode passar valores de retorno usando variáveis de saída.
Funções podem retornar valores escalares e são chamadas dentro de uma instrução. Exemplo:

* **SELECT FUNCÃO(parâmetros);**

Atualmente o MySQL só preserva o contexto para o banco de dados padrão. Isto é, se você usar USE dbname dentro de um procedimento, o banco de dados original é restaurado depois da saída da rotina. Uma rotina herda o banco de dados padrão de quem a chama, assim geralmente as rotinas devem utilizar uma instrução USE dbname, ou especifique todas as tabelas com uma referência de banco de dados explicita, ex. dbname.tablename.

Sintaxe:

CREATE PROCEDURE e CREATE FUNCTION

CREATE PROCEDURE sp_name ([parâmetros[...]])[característica ...] corpo_da_rotina

CREATE FUNCTION sp_name ([parâmetros[...]])[RETORNA tipo] [característica...] corpo_da_rotina

parameter: [IN | OUT | INOUT] nome_tipo_parâmetro

tipo:
Qualquer tipo válido do MySQL

característica:
LANGUAGE SQL
| [NOT] DETERMINISTIC
| SQL SECURITY DEFINER | INVOKER
| COMMENT string

corpo da rotina:
Indicação válida do procedimento do SQL
———————————————————————————-

RETURNS pode ser especificada apenas por uma FUNCTION. É usada para indicar o tipo de retorno da função.

A lista de parâmetros entre parênteses deve estar sempre presente. Se não houver parâmetros, uma lista de parâmetros vazia de () deve ser usada. Existem três tipos de parâmetros:

* IN : parâmetros de entrada
* OUT : parâmetros de saída
* INOUT : parâmetros de entrada e saída

O MySQL também permite rotinas contendo instruções DDL (como CREATE e DROP) e instruções de transação SQL (como COMMIT).

Esses conceitos podem paraçer um pouco confusos no momento, mas com os exemplos mostrados no decorrer da lição, boa parte das dúvidas serão esclarecidas.

3.14.2 Exemplos de Stored Procedures e Functions

Abaixo temos um exemplo de uma stored procedure simples que usa um parâmetro OUT. O camando **delimiter** serve para alterar o delimitador de instrução para antes da definição do procedure (troca ";"por "|").

```
mysql> delimiter |

mysql> create procedure simpleproc(out numero int)
-> begin
-> set numero=10;
-> end
-> |
Query OK, 0 rows affected (0.02 sec)

mysql> CALL simpleproc(@a)|
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SELECT @a|
+——+
| @a |
+——+
| 10 |
+——+
1 row in set (0.00 sec)
```

Neste procedimento, simplesmente atribuimos o valor 10 à variável @a.

A seguir está um exemplo de uma função que utiliza um parâmetro, realiza uma operação usando uma função SQL e retorna o resultado:

```
mysql> delimiter |

mysql> CREATE FUNCTION hello (s CHAR(20)) RETURNS CHAR(50)
-> RETURN CONCAT('Hello, ',s,'!');
-> |
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT hello('world')|
+————————-+
| hello('world') |
+————————-+
| Hello, world! |
+————————-+
1 row in set (0.00 sec)
```

3.14.3 Declarando Variáveis

Também é permitido declarar variáveis dentro de funções e procedimentos.
Vejamos um exemplo:

```
mysql> CREATE PROCEDURE declarandovariaveis()
> BEGIN
> DECLARE a INT;
> DECLARE b INT;
> SET a=5;
> SET b=5;
> INSERT INTO t VALUES (a,b);
> END
> |
```

O procedimento acima declara duas variáveis, a e b, atribuindo o valor 5 a elas. Depois de atribuido os valores, o procedimento os insere em uma tabela genérica t.

Como observamos declarar variáveis é muito simples e de grande utilidade quando é preciso elaborar procedimentos mais complexos.

3.14.4 Condições

Assim como em outras linguagens, podemos estabelcer condições para rotinas em nossos procedimentos. Vejamos o exemplo:

```
mysql> create procedure condição(in parametro int)
> begin
> if parametro=0 then select "é igual a zero";
> else select "é diferente de zero"
> end if;
> end
> |
```

No exemplo acima, o procedimento recebe um número como parâmetro, e é observado se este número é igual ou diferente de zero.

Acima foi utilizada a expresão if-else, agora vejamos um exemplo da estrutura case:

```
mysql> create procedure condição2(in parametro int)
> begin
>case parametro
>when 0 then select "zero";
>when 1 then select "um";
>else select "não é zero nem um";
> end case;
>end
> |
```

Agora veficamos o valor do parâmetro utilizando a instrução case. Poderiamos continuar com comparações caso fosse necessário. Por exemplo:

```
mysql> create procedure condição2(in parametro int)
> begin
>case parametro
>when 0 then select "zero";
>when 1 then select "um";
>when 3 then select "tres";
>else select "outro numero";
> end case;
>end
> |
```

3.14.5 Laços de Repetição

Podemos também colocar laços de repetição em nossos procedimentos.Veremos aqui duas estruturas que têm essa funcionalidade.
Para observarmos melhor estes exemplos, precisamos criar uma tabela qualquer, utilize o camando create table, e crie uma tabela com um campo inteiro.

While:

```
mysql> create procedure laço(in parametro int)
-> begin
-> while parametro>=0 do
-> insert into tabela_criada values (parametro);
-> set parametro = parametro - 1;
->end while;
-> end
-> |
```

Este procedimento, quando chamado, irá inserir na tabela criada valores de 0 ao número passado como parâmetro.

Repeat:

```
mysql> create procedure laço1(in parametro int)
-> begin
-> repeat
-> select parametro;
-> set parametro = parametro + 1;
->until parametro >= 5
-> end repeat;
-> end
```

Este procedimento repetirá do valor do número passado como parâmetro até 5, porém se o número for maird o que 5 ele simplesmente imprimirá o número na tela.