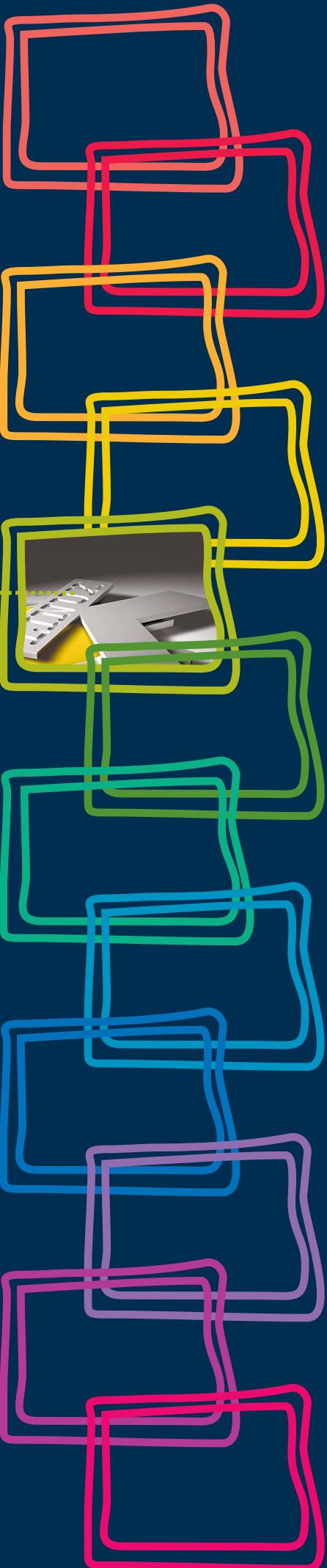




PROJETO ESCOLAS - REFERÊNCIA
Compromisso com a Excelência na Escola Pública

Cadernos de Informática



CURSO DE CAPACITAÇÃO EM INFORMÁTICA INSTRUMENTAL

CURSO DE MONTAGEM E MANUTENÇÃO DE COMPUTADORES

CURSO SOBRE O SISTEMA OPERACIONAL LINUX

CURSO DE PROGRAMAÇÃO EM JAVA

CURSO DE INTRODUÇÃO A BANCOS DE DADOS

CURSO DE CONSTRUÇÃO DE WEB SITES

CURSO DE EDITORAÇÃO ELETRÔNICA

CURSO DE ILUSTRAÇÃO DIGITAL

CURSO DE PRODUÇÃO FONOGRÁFICA

CURSO DE COMPUTAÇÃO GRÁFICA

CURSO DE PROJETO AUXILIADO POR COMPUTADOR

CURSO DE MULTIMÍDIA NA EDUCAÇÃO

CURSO DE INTRODUÇÃO AOS BANCOS DE DADOS

Secretaria de Estado de Educação MG

Cadernos de Informática

CURSO DE INTRODUÇÃO A BANCOS DE DADOS

Éber Machado Duarte

Coordenador

Carlos Eduardo Hermeto Sá Motta



APRESENTAÇÃO

Os computadores que estão sendo instalados pela SEE nas escolas estaduais deverão ser utilizados para propósitos administrativos e pedagógicos. Para isso, desenvolveu-se um conjunto de cursos destinados a potencializar a utilização desses equipamentos. São doze cursos que estão sendo disponibilizados para as escolas para enriquecimento do seu plano curricular. Esses cursos não são profissionalizantes. São cursos introdutórios, de formação inicial para o trabalho, cujo objetivo é ampliar o horizonte de conhecimento dos alunos para facilitar a futura escolha de uma profissão.

Todos os cursos foram elaborados para serem realizados em 40 módulos-aula, cada um deles podendo ser desenvolvidos em um semestre (com 2 módulos-aula semanais) ou em 10 semanas (com 4 módulos-aula semanais). Em 2006, esses cursos deverão ser oferecidos para os alunos que desejarem cursá-los, em caráter opcional e horário extra-turno.

Em 2007, eles cursos deverão ser incluídos na matriz curricular da escola, na série ou séries por ela definida, integrando a Parte Diversificada do currículo.

Esses cursos foram concebidos para dar aos professores, alunos e funcionários uma dimensão do modo como o computador influencia, hoje, o nosso modo de vida e os meios de produção. Para cada curso selecionado pela escola deverão ser indicados pelo menos dois ou, no máximo, três professores (efetivos, de preferência) para serem capacitados pela SEE. Esses professores irão atuar como multiplicadores, ministrando-os a outros servidores da escola e aos alunos.

CURSO DE CAPACITAÇÃO EM INFORMÁTICA INSTRUMENTAL

Este curso será implantado obrigatoriamente em todas as escolas estaduais em que for instalado laboratório de informática. Iniciando pelas Escolas-Referência, todos os professores e demais servidores serão capacitados para que possam fazer uso adequado e proveitoso desses equipamentos tanto na administração da escola como nas atividades didáticas.

É um curso voltado para a desmistificação da tecnologia que está sendo implantada. O uso do computador ainda é algo difícil para muitas pessoas que ainda não estão muito familiarizadas com essas novas tecnologias que estão ocupando um espaço cada vez maior na escola e na vida de todos. Este curso vai motivar os participantes para uma aproximação com essas tecnologias, favorecendo a transformação dos recursos de informática em instrumentos de produção e integração entre gestores, professores e demais servidores. As características dos equipamentos e as funcionalidades dos programas serão apresentadas de maneira gradual e num contexto prático. Essas situações práticas serão apresentadas de maneira que o participante perceba o seu objetivo e o

valor de incorporá-las ao seu trabalho cotidiano. Os participantes serão preparados para navegar e pesquisar na internet; enviar, receber e administrar correspondência eletrônica, além de criar e editar documentos (textos, planilhas e apresentações) de interesse acadêmico e profissional. Esse é um curso fundamental, base e pré-requisito para todos os demais.

CURSO DE MONTAGEM E MANUTENÇÃO DE COMPUTADORES

Este curso será implantado em, pelo menos, uma escola do município sede de cada Superintendência Regional de Ensino. A indicação da escola deverá ser feita pela própria S.R.E, levando-se em conta as condições de infra-estrutura nas Escolas-Referência existentes no município. Nas escolas escolhidas será montado um laboratório de informática especialmente para a oferta desse curso.

O objetivo deste curso é capacitar tecnicamente os alunos de ensino médio que queiram aprender a montar, fazer a manutenção e configurar microcomputadores. Pode ser oferecido para alunos de outras escolas, para professores e demais servidores da escola e para a comunidade, aos finais de semana ou horários em que o laboratório esteja disponível.

Neste curso o participante aprenderá a função de cada um dos componentes do microcomputador. Aprenderá como montar um computador e como configurá-lo, instalando o sistema operacional, particionando e formatando discos rígidos, instalando placas de fax/modem, rede, vídeo, som e outros dispositivos. Conhecerá, ainda, as técnicas de avaliação do funcionamento e configuração de microcomputadores que esteja precisando de manutenção preventiva ou corretiva, além de procedimentos para especificação de um computador para atender as necessidades requeridas por um cliente.

Dos cursos que se seguem, as Escolas-Referência deverão escolher pelo menos dois para implantar em 2006.

No período de 13 a 25 de março/2006, estará disponível no sítio da SEE (www.educacao.mg.gov.br) um formulário eletrônico para que cada diretor das Escolas-Referência possa informar quais os cursos escolhidos pela sua escola e quais os professores que deverão ser capacitados. Durante o período de capacitação, os professores serão substituídos por professores-designados para que as atividades didáticas da escola não sejam prejudicadas.

1. CURSO SOBRE O SISTEMA OPERACIONAL LINUX

É destinado àqueles que desejam conhecer ferramentas padrão do ambiente Unix. É um curso voltado para a exploração e organização de conteúdo. São ferramentas tipicamente usadas por usuários avançados do sistema operacional. Tem por finalidade apresentar alguns dos programas mais simples e comuns do ambiente; mostrar que, mesmo

com um conjunto pequeno de programas, é possível resolver problemas reais; explicar a comunicação entre programas via rede e estender o ambiente através de novos programas. O texto didático deste curso apresenta os recursos a serem estudados e propõe exercícios. É um curso para aqueles que gostam de enfrentar desafios.

Ementa: Histórico e desenvolvimento do Unix e Linux. Login no computador. Explorando o computador (processos em execução, conexões abertas). Descrição dos conceitos de arquivo e diretório. Operações simples sobre arquivos e diretórios. Sistema de permissões e quotas.

Procurando arquivos e fazendo backups. Executando e controlando programas. Processamento de texto. Expressões regulares. Estendendo o ambiente. Trabalho em rede. Um sistema de chat. Comunicação segura no chat (criptografia). Ainda criptografia. Sistema de arquivos como um Banco de Dados. Um programa gráfico. Programando para rede.

2. CURSO DE PROGRAMAÇÃO EM JAVA

É um curso de programação introdutório que utiliza a linguagem Java. Essa linguagem se torna, a cada dia, mais popular entre os programadores profissionais. O curso foi desenvolvido em forma de tutorial. O participante vai construir na prática um aplicativo completo (um jogo de batalha naval) que utiliza o sistema gráfico e que pode ser utilizado em qualquer sistema operacional. Os elementos de programação são apresentados em atividades práticas à medida em que se fazem necessários. Aqueles que desejam conhecer os métodos de produção de programas de computadores terão, nesse curso, uma boa visão do processo.

Ementa: Conceitos de linguagem de programação, edição, compilação, depuração e execução de programas. Conceitos fundamentais de linguagens de programação orientada a objetos.

Tipos primitivos da linguagem Java, comandos de atribuição e comandos de repetição. Conceito de herança e programação dirigida por eventos. Tratamento de eventos. Programação da interface gráfica. Arrays. Números aleatórios.

3. CURSO DE INTRODUÇÃO AO BANCOS DE DADOS

Este curso mostrará aos participantes os conceitos fundamentais do armazenamento, gerenciamento e pesquisa de dados em computadores. Um banco de dados é um repositório de informações que modelam entidades do mundo real. O Sistema Gerenciador do Banco de Dados permite introduzir, modificar, remover, selecionar e organizar as informações armazenadas. O curso mostra como os bancos de dados são criados e estruturados através de exemplos práticos. Ao final, apresenta os elementos da linguagem SQL (Structured Query Language – Linguagem Estruturada de Pesquisa) que é uma

linguagem universal para gerenciamento de informações de bancos de dados e os elementos básicos da administração desses repositórios de informação.. Apesar de ser de nível introdutório, o curso apresenta todos os tópicos de interesse relacionados à área. É um curso voltado para aqueles que desejam conhecer os sistemas que gerenciam volumes grandes e variados de informações, largamente utilizados no mundo empresarial.

Ementa: Modelagem de dados. Normalização. Linguagem SQL. Mecanismos de consulta. Criação e alteração de tabelas. Manipulação e formatação de dados. Organização de resultados de pesquisa. Acesso ao servidor de bancos de dados. Contas de usuários. Segurança. Administração de bancos de dados. Manutenção. Integridade.

4. CURSO DE CONSTRUÇÃO DE WEB SITES

Este curso mostrará aos participantes como construir páginas HTML que forma a estrutura de um “site” na internet. A primeira parte do curso é voltada para a construção de páginas; a segunda parte, para a estruturação do conjunto de páginas que formam o “site”, incluindo elementos de programação. Explicará os conceitos elementares da web e mostrará como é que se implementa o conjunto de páginas que forma o “site” num servidor.

Ementa: Linguagem HTML. Apresentação dos principais navegadores disponíveis no mercado.

Construção de uma página HTML simples respeitando os padrões W3C. Recursos de formatação de texto. Recursos de listas, multimídia e navegação. Tabelas e *Frames*. Folha de Estilo. Elementos de Formulário. Linguagem Javascript. Interação do Javascript com os elementos HTML. Linguagem PHP. Conceitos de Transmissão de Site e critérios para avaliação de servidores.

1. CURSO DE EDITORAÇÃO ELETRÔNICA

Voltado para a produção de documentos físicos (livros, jornais, revistas) e eletrônicos. Apresenta as ferramentas de produção de texto e as ferramentas de montagem de elementos gráficos numa página. O texto é tratado como elemento de composição gráfica, juntamente com a pintura digital, o desenho digital e outros elementos gráficos utilizados para promover a integração dos elementos gráficos.

O curso explora de maneira extensiva os conceitos relacionados à aparência do texto relativos aos tipos de impressão (fontes). Mostra diversos mecanismos de produção dos mais variados tipos de material impresso, de texto comum às fórmulas matemáticas. Finalmente, discute a metodologia de gerenciamento de documentos.

Ementa: Editor de textos. Formatadores de texto. Tipos e Fontes. Gerenciamento de projetos.

Publicações. Programas para editoração. Programas acessórios. Impressão. Desenvolvimento de um projeto.

2. CURSO DE ILUSTRAÇÃO DIGITAL

Desenvolvido sobre um único aplicativo de tratamento de imagens e pintura digital, o GIMP (GNU Image Manipulation Program – Programa de Manipulação de Imagens GNU).

Este curso ensina, passo a passo, como utilizar ferramentas do programa para produzir ilustrações de qualidade que podem ser utilizadas para qualquer finalidade. A pintura digital é diferente do desenho digital. O desenho se aplica a diagramas e gráficos, por exemplo. A pintura tem um escopo muito mais abrangente e é uma forma de criação mais livre, do ponto de vista formal. É basicamente a diferença que há entre o desenho artístico e o desenho técnico. É, portanto, um curso voltado para aqueles que têm interesses e vocações artísticas.

Ementa: A imagem digital. Espaços de cores. Digitalização de imagens. Fotomontagem e colagem digital. Ferramentas de desenho. Ferramentas de pintura. Finalização e saída.

3. CURSO DE PRODUÇÃO FONOGRÁFICA

Curso voltado para aqueles que têm interesse na produção musical. Explica, através de programas, como é que se capturam, modificam e agrupam os sons musicais para produzir arranjos musicais. É um curso introdutório com uma boa visão da totalidade dos procedimentos que levam à produção de um disco.

Ementa: O Fenômeno Sonoro. O Ambiente Sonoro. A Linguagem Musical. Pré-Produção. O Padrão MIDI. A Gravação. A Edição. Pós-processamento. Mixagem. Finalização.

4. CURSO DE COMPUTAÇÃO GRÁFICA

Curso introdutório de modelagem, renderização e animação de objetos tridimensionais.

Esse curso é a base para utilização de animações tridimensionais em filmes. Conduzido como um tutorial do programa BLENDER, apresenta a interface do programa e suas operações elementares. Destinado àqueles que têm ambições de produzir animações de alta qualidade para a educação ou para a mídia.

Ementa: Introdução à Computação Gráfica. Conceitos básicos 2D e 3D. Interface principal do programa Blender. Espaço de trabalho. Navegação em 3D. Modelagem em 3D. Primitivas básicas. Movimentação de objetos. Edição de objetos. Composição de cenas. Materiais e texturas. Aplicação de materiais. UV Mapping. Luzes e Câmeras. Iluminação de cena. Posicionamento e manipulação de câmera. Renderização still frame. Formatos

de saída. Animação básica. Movimentação de câmera e objetos. Renderização da animação. Formatos de saída.

5. CURSO DE PROJETO AUXILIADO POR COMPUTADOR

Os programas de CAD (Computer Aided Design – Projeto Auxiliado por Computador) são utilizados para composição de desenhos técnicos. Diferentemente dos programas de pintura eletrônica (como o GIMP), fornecem ao usuário ferramentas para desenhar com precisão e anotar os desenhos de acordo com as normas técnicas. Além de ensinar ao usuário a utilizar um programa de CAD (QCad), o curso apresenta elementos básicos de desenho técnico e construções geométricas diversas visando preparar o participante para um aprimoramento em áreas típicas das engenharias e da arquitetura..Ementa: Informática aplicada ao desenho técnico. Conceitos básicos: construções geométricas, escalas, dimensionamento, projeções ortográficas e perspectivas. Sistemas de coordenadas cartesiano e polar. Novas entidades geométricas básicas: polígonos e círculos.

Operações geométricas básicas. Tipos de unidades de medida. Criação de um padrão de formato. Organização de um desenho por níveis. Construções geométricas diversas. A teoria dos conjuntos aplicada ao desenho. Propriedades dos objetos. Edição do desenho.

Movimento, rotação, escalamento e deformação de objetos. Agrupamento de objetos em blocos.

6. CURSO DE MULTIMÍDIA NA EDUCAÇÃO

O curso está dividido em três partes: a) utilização da multimídia no contexto educacional; b) autoria de apresentações multimídia; c) projetos de aprendizagem mediada por tecnologia. Este curso é o fundamento para a criação dos cursos de educação a distância.

Apresenta os elementos que compõem os sistemas de multimídia, as comunidades virtuais de aprendizagem, o planejamento e a preparação de uma apresentação e de uma lição de curso e, finalmente, a tecnologia de objetos de aprendizado multimídia.

Ementa: Introdução à Multimídia e seus componentes. Multimídia na Educação. Comunidades Virtuais de Aprendizagem. “Webquest”: Desafios Investigativos baseados na Internet (Web).

Preparação de uma apresentação multimídia.

SUMÁRIO

CAPÍTULO 1	15
1 - Introdução	15
1.1 - Descrição do banco de dados acadêmico	16
1.2 - Atributos definem uma entidade.....	17
1.3 - Os registros são as ocorrências de uma entidade no mundo real....	18
1.4 - Relações e integridades de dados	18
2 - Utilizando o MySQL para criar o banco de dados escolar	19
2.1 - Criando o banco de dados escolar através do MySQL	19
2.2 - Criando e selecionando um banco de dados no MySQL	21
2.3 - Criando as tabelas definidas no modelo	22
2.4 - Inserindo informações no banco de dados	23
2.5 - Exibindo as informações contidas no banco de dados	24
3 - Conclusões	26
4 - Referências Bibliográficas	26
CAPÍTULO 2	27
1 - Introdução à modelagem de dados	27
2 - Descrição de um sistema para controle de uma empresa.....	27
2.1. Descrição de uma empresa de construção civil	27
3 - Aspectos gerais do modelo entidade-relacionamento	28
3.1 - Descrição das entidades problema	29
3.2 - Descrição dos relacionamentos do sistema	30
4 - Modelando as restrições de dados do modelo	32
4.1 - Restrições de atributos	32
4.2 - Integridade referencial.....	32
4.3 - Normalização de dados, eliminando a redundância	33
5 - Conclusões	33
6 - Exercícios de fixação	34
7 - Referências Bibliográficas	34
CAPÍTULO 3	35
1 - Introdução à ferramenta de modelagem DBDesigner4	35
2 - Introdução ao DBDesigner4	36

2.1 - Apresentação dos componentes do DBDesigner4	36
2.1.1 - Descrição da opção File	37
2.1.2 - Descrição da opção Database	38
2.2 - Apresentando a barra de tarefas do DBDesigner4	41
2.2.1 - Criando entidades no DBDesigner4	42
2.2.2 - Definindo os relacionamentos	43
3 - Conclusões	44
4 - Exercícios de fixação	44
5 - Referências Bibliográficas	44
CAPÍTULO 4	45
1 - Introdução	45
2 - Criando o banco de dados e a conexão para o mesmo	45
3 - Construção das entidades do modelo	46
3.1 - Tipos de dados no MySQL	47
3.2 - Definição das entidades do sistema	47
3.3 - Definindo os relacionamentos entre as tabelas	51
3.4 - Convertendo o modelo para o banco de dados curso	54
4 - Conclusões	54
5 - Referências Bibliográficas	54
CAPÍTULO 5	55
1 - Introdução	55
2 - Introdução ao MySQL Query Browser	55
2.1 - Abrindo uma conexão com o banco de dados curso	56
3 - Explorando os principais recursos do MySQL Query Browser	57
3.1 - Utilizando o editor de banco de dados	57
3.2 - Conhecendo o editor de consultas SQL do sistema	59
3.2.1 - Inserindo, alterando e excluindo dados em uma planilha	59
3.2.2 - Leitura de informações armazenadas em uma tabela	61
4 - Conclusões	63
5 - Exercícios de fixação	63
6 - Referências Bibliográficas	63

CAPÍTULO 6 64

1 - Introdução	64
2 - Introdução aos tipos de dados	64
3 - Descrição dos comandos para definição de dados	65
3.1 - Criando e removendo um banco de dados	65
3.2 - Criando e removendo tabelas	66
3.2.1 - Entendendo o conceito de chave primária	67
3.2.2 - Entendendo o conceito de chave estrangeira	69
3.2.3 - Removendo e alterando uma tabela	71
4 - Conclusões	72
5 - Exercícios de fixação	72
6 - Referências Bibliográficas	72

CAPÍTULO 7 73

1 - Introdução	73
2 - Entendendo os comandos DML	73
2.1 - Incluindo dados com o comando INSERT	73
2.2 - Alterando dados com o comando UPDATE	77
2.3 - Excluindo dados com o comando DELETE	78
3 - Introdução aos comandos para leitura de dados	78
3.1 - O comando SELECT básico	79
3.2 - Selecionando registros com o WHERE	80
3.3 - Agrupando dados com o GROUP BY e selecionando com o HAVING	82
3.4 - Ordenando o resultado com o ORDER BY	83
4 - Conclusões	84
5 - Exercícios de fixação	84
6 - Referências Bibliográficas	84

CAPÍTULO 8 85

1 - Introdução	85
2 - Técnicas para extração de dados em múltiplas tabelas-JOIN	85
2.1 - O produto cartesiano, um erro comum na elaboração do JOIN	87
2.2 - INNER JOIN, uma sintaxe alternativa para a junção de tabelas	89
3 - Entendendo os mecanismos de sub-consultas	89

3.1 - Utilizando um SELECT dentro da cláusula SELECT	90
3.2 - Utilizando um SELECT dentro da cláusula FROM	90
3.3 - Utilizando um SELECT dentro da cláusula WHERE	91
3.4 - Agrupando resultados com o comando UNION	92
4 - Utilizando funções para transformação de dados	93
5 - Conclusões	95
6 - Exercícios de fixação.....	95
7 - Referências Bibliográficas	95
 CAPÍTULO 9	 96
1 - Introdução	96
2 - Introdução ao MySQL Administrator	97
3 - Explorando as opções da ferramenta MySQL Administrator	98
3.1 - Controle do serviço MySQL	98
3.2 - Configurando os parâmetros do servidor MySQL	99
3.3 - Administração dos usuários do sistema	100
3.4 - Monitorando a atividade do servidor.....	102
3.5 - Realizando cópias de segurança do banco de dados	104
4 - Conclusões	106
5 - Exercícios de fixação.....	106
6 - Referências Bibliográficas	107
 CAPÍTULO 10	 108
1 - Introdução	108
2 - Entendendo o controle de transações	109
3 - Utilizando a replicação para obter alta disponibilidade.....	113
4 - Trabalhando com rotinas armazenadas no SGBD	113
5 - Conclusões	114
6 - Exercícios de fixação.....	115
7 - Referências Bibliográficas	115

CAPÍTULO 1

1. INTRODUÇÃO

O conceito de banco de dados bem como as tecnologias de bancos de dados têm ganhado mais importância e estão se tornando cada vez mais populares com a expansão da utilização dos computadores. No dicionário encontramos: "Computador adj.s.m. 1 (o) que computa ou calcula, s.m. 2 máquina eletrônica que guarda, analisa e processa dados". O primeiro computador, chamado de ENIAC, foi inventado pelos cientistas Eckert e Mauchly e foi apresentado ao mundo em meados de 1945. Este fato desencadeou uma série de avanços tecnológicos que culminou com a criação dos computadores pessoais. Estes equipamentos possuem um custo reduzido e possibilitam a execução de milhões de cálculos em uma fração de segundos, além de armazenar grandes volumes de informações. Para se ter uma idéia, é possível armazenar em um único CD todo o conteúdo da enciclopédia Balsa.

Assim, começaram a surgir os primeiros sistemas computacionais desenvolvidos para as mais variadas áreas do conhecimento, tais como a engenharia, medicina, bibliotecas, sistemas educacionais, dentre outros.

Esta popularização dos computadores acarretou a geração de um grande volume de dados e informações. Como consequência, tornou-se necessário organizar, armazenar e acessar estas informações de forma ordenada e fácil. Para suprir esta necessidade foram projetados os primeiros sistemas de bancos de dados, objetivando uma sistematização do acesso aos dados.

Um banco de dados é uma coleção de dados relacionados. Entende-se por dado, toda a informação que pode ser armazenada e que apresenta algum significado implícito dentro do contexto ao qual ele se aplica. Por exemplo, ao visualizar uma pessoa conduzindo um veículo pelas ruas está subentendido que a mesma conhece as técnicas para a condução do automóvel. Portanto, o conhecimento acerca do domínio da técnica de direção está implícito na cena apresentada. Exemplificando no cenário de uma base de dados, um cadastro de pessoas poderá apresentar informações diferentes se empregado em sistemas distintos. No caso de um sistema bancário, uma pessoa é identificada pelo seu CPF (cliente). Em um sistema escolar a pessoa é identificada pelo seu número de matrícula (aluno). Além disto, os dados que serão armazenados em cada situação podem se diferir consideravelmente.

Esta definição de banco de dados é bastante genérica, daí pode-se considerar como um banco de dados uma coleção de textos sobre um determinado assunto. No contexto deste curso, utiliza-se uma definição mais restrita para banco de dados, onde o mesmo será sempre a representação de uma situação encontrada no mundo real. Ou seja, um banco de dados é definido dentro de um universo de discurso, isto é, a situação real que ele representa. Para ilustrar, seja um banco de dados para armazenar informações referentes às linhas de ônibus urbanos existentes em uma cidade. Neste cenário, é relevante armazenar o número, placa e cor dos veículos, já que as linhas são identificadas pelas cores dos ônibus. Portanto, as linhas de ônibus urbanas é o universo de discurso desta aplicação, sendo que as informações somente se aplicam a este ambiente. Certamente se for necessário manter dados sobre ônibus intermunicipais, os requisitos de dados mudarão, uma vez que alterou o universo de discurso. O universo de discurso também é entendido como escopo da aplicação, que na verdade são os limites ou propósitos para os quais os dados servirão.

Neste módulo será apresentado um pequeno banco de dados com o intuito de exemplificar o que é um sistema de banco de dados. Para isto será utilizado como exemplo, um sistema acadêmico muito rudimentar, onde serão armazenados os alunos da instituição e as notas obtidas por eles nas avaliações escolares. Estes elementos servirão de base para apresentar e conceituar os elementos básicos que constituem um banco de dados.

1.1. DESCRIÇÃO DO BANCO DE DADOS ACADÊMICO

O banco de dados descrito nesta seção é concebido para representar um pequeno sistema escolar, onde existem basicamente dois componentes que são os alunos matriculados na instituição, bem como as notas obtidas por eles em todas as avaliações realizadas durante um período escolar.

Conforme a definição dada na seção anterior, um banco de dados é projetado para atender os aspectos inerentes ao universo delimitado, neste caso, o sistema escolar. Uma vez definido o escopo da aplicação, ou seja, o seu propósito, o próximo passo é identificar os elementos que a constituem, e por consequência definir todos os dados relevantes para cada item existente. Estes elementos são comumente chamados de entidades, e que por questões de facilidade, são representadas por tabelas.

Uma entidade é todo e qualquer elemento que participa do contexto definido para a aplicação, e que geralmente se refere a um objeto real dentro deste escopo. Para o sistema proposto, percebe-se a existência de duas entidades: 1. Alunos e 2. Pontuação dos alunos nas avaliações. A primeira delas tem como objetivo manter um registro de cada aluno matriculado na escola. A **Tabela 1** ilustra uma estrutura de armazenamento desta entidade.

Tabela 1:
Definição da
ENTIDADE
alunos

MATRÍCULA	NOOME	SÉRIE	TURMA	TELEFONE	DATA DE NASCIMENTO
1	JOSÉ DA SILVA	0ITAVA	1	(31)1234-5678	05-10-1982
2	ANA MARIA	SÉTIMA	1	(31)3421-5678	17-11-1981
3	PAULO SIMON	QUINTA	1	(11)1234-5678	11-04-1983
4	CARLA BEATRIZ	SEXTA	1	(21)3344-5678	30-07-1979
5	ANA PAULA	0ITAVA	2	(92)5555-8888	22-01-1980

A segunda entidade identificada no problema são as pontuações obtidas por cada aluno. Vale ressaltar que durante um período letivo poderão existir várias avaliações, geralmente em datas diferentes, onde deverão ser armazenados os resultados de todos os alunos para cada um destes testes. A **Tabela 2** descreve a entidade pontuações, que serve para o propósito exposto anteriormente.

Tabela 2:
Definição da
ENTIDADE
pontuação

MATRÍCULA	DATA DO TESTE	PONTO
1	25-03-2004	5.5
2	25-03-2004	6
3	25-03-2004	8
4	25-03-2004	10
5	25-03-2004	7.8
1	18-05-2004	4.6
2	18-05-2004	7.2
5	18-05-2004	9.5

Percebe-se que cada entidade é representada por uma tabela, sendo que neste universo de discussão ou modelo, existem apenas duas tabelas e um relacionamento entre elas, já que cada entidade aluno está ligada à entidade pontuação. Em aplicações mais complexas, poderão existir inúmeras tabelas e relacionamentos de forma a permitir a representação do problema abordado.

Dado o banco de dados sugerido anteriormente, as seções seguintes, abordarão os conceitos de atributos, relacionamentos, integridade de dados e instâncias de forma a contextualizá-los dentro do problema proposto.

1.2. ATRIBUTOS DEFINEM UMA ENTIDADE

Uma entidade, no exemplo alunos e pontuações, são representadas por tabelas que por sua vez são constituídas de linhas e colunas. Cada coluna representa um fragmento de dado e o conjunto de todas as colunas constitui a entidade propriamente dita. No contexto de banco de dados cada coluna é chamada de atributo e uma entidade será formada por um ou vários atributos.

Um atributo define uma característica da entidade, por exemplo, um aluno tem nome, idade, altura cor dos cabelos, sexo, dentre outras características. Neste caso o número de atributos existente na tabela, que definem o grau da entidade, dependerá única e exclusivamente de quais dados serão relevantes para representar uma entidade dentro do escopo que está sendo representado. Isto é, no contexto do sistema escolar a cor dos cabelos não é importante, por isto não está representada na tabela.

No sistema descrito a entidade alunos é constituída por seis atributos que são o número de matrícula, nome, a série que está cursando, a sua turma, o seu telefone residencial e a data de nascimento. O atributo matrícula possui um papel importante no modelo servindo como um identificador único para cada aluno. Por exemplo, para pesquisar em um dicionário utiliza-se uma palavra-chave, sendo que uma mesma palavra pode aparecer mais de uma vez, como ocorre com o verbo “casar”. O significado que se deseja para a palavra será determinado pelo contexto onde a mesma é empregada, resolvendo assim a ambigüidade. Em um banco de dados caso ocorram registros com valores idênticos não será possível determinar um contexto que os identifiquem unicamente, como ocorre no dicionário. Por isto, deve existir uma chave ou atributo que identifique unicamente cada registro. Ao observar a **Tabela 1**, percebe-se que não há dois alunos cadastrados com o mesmo número de matrícula. Portanto, este é o atributo chave da entidade, utilizado para a pesquisa de um registro nesta tabela.

A entidade pontuações necessita identificar o aluno, a data da avaliação e a pontuação atingida pelo aluno. Neste caso, como cada aluno é identificado unicamente pela sua matrícula, este atributo será inserido na tabela de pontuações para permitir associar o aluno à nota registrada, conforme visto na **Tabela 2**.

Percebe-se que cada atributo possui um conjunto de valores válidos e aceitáveis, que é definido como domínio do atributo. Todas as informações vistas na tabela são textuais, isto é, seqüências de letras e números, mas é notório que o conjunto de dados contido em cada coluna é diferente umas das outras. No caso da matrícula do aluno, o domínio dos dados é o conjunto dos números inteiros positivos, já que para cada aluno é atribuído um código numérico que denota a ordem em que este foi matriculado na escola. Ou

seja, o texto contido nesta coluna é formado por uma combinação de números, portanto não existem letras.

Desta forma cada coluna apresenta restrições de dados inerentes ao seu domínio. Por exemplo, não existirá um número de matrícula constituído de letras, já que o mesmo tem como objetivo marcar a ordem de inscrição dos alunos na instituição. Outros domínios existentes no modelo, são textos e datas, que são utilizados para representar as demais informações das entidades.

1.3. OS REGISTROS SÃO AS OCORRÊNCIAS DE UMA ENTIDADE NO MUNDO REAL

Além das colunas ou atributos, as tabelas possuem também uma ou mais linhas ou registros, conforme visto nas **Tabelas 1 e 2**. Cada registro representa uma ocorrência daquela entidade no mundo real. Por exemplo, na **Tabela 1**, o registro com o número de matrícula igual a 1 se refere ao aluno José da Silva. Existirão tantos registros na tabela de alunos quantos forem o número de alunos matriculados na escola.

A quantidade de registros existentes em uma tabela define a cardinalidade da entidade, ou seja, o número de elementos no conjunto de alunos. Por exemplo, se a escola possui dez alunos então o conjunto de alunos desta escola possui dez elementos. Daí a sua cardinalidade também será dez. Vale ressaltar que, como não há dois alunos idênticos em uma mesma escola, não haverá dois registros iguais em uma mesma tabela. Isto é garantido pelo código de matrícula que é único e é conhecido como chave primária. Neste caso, a chave primária não permitirá a existência de valores duplicados para este atributo.

1.4. RELAÇÕES E INTEGRIDADE DE DADOS

No modelo exposto, há um relacionamento entre a tabela de alunos e pontuação. Este relacionamento é caracterizado pelo atributo matrícula que está presente em ambas as tabelas. Dado que este atributo é a chave primária da tabela, isto é, não existem dois alunos com o mesmo número de matrícula, esta coluna quando inserida na tabela de pontuações permite relacionar um aluno a uma nota obtida.

Na entidade pontuação, não será possível utilizar apenas a matrícula do aluno como chave primária, já que um mesmo aluno poderá realizar avaliações em datas distintas. Neste caso, o identificador único para a entidade pontuação será a combinação da matrícula com a data do evento. Mais uma vez, não será possível que um mesmo aluno realize mais de uma avaliação no mesmo dia.

Existem outras restrições intrínsecas ao modelo que devem ser respeitadas pelo banco de dados, como por exemplo, não se deve permitir uma ocorrência de uma pontuação para um número de matrícula que não exista cadastrado na tabela de alunos. Este conceito é conhecido como restrição de integridade ou integridade referencial, e deve ser assegurado pelo sistema de banco de dados a fim de se manter as características do mundo representado por ele. Este conceito é implementado através das chaves estrangeiras, que nada mais são que colunas que definem o relacionamento entre entidades. No modelo apresentado a matrícula do aluno representa a chave estrangeira do relacionamento.

Assim, não se deve permitir a inserção de pontuações para alunos que não estejam cadastrados, bem como na remoção de alunos que possuam pontuações cadastradas, deve-se eliminar estes registros para manter-se a consistência das informações. O conceito de consistência está ligado à garantia de que o dado está correto. Por exemplo, em uma aplicação bancária, ao realizar um saque em uma conta deve-se assegurar que o saldo será reduzido da quantia sacada. Caso contrário o dado estará incorreto e poderá acarretar uma análise errada das informações.

2. UTILIZANDO O MySQL PARA CRIAR O BANCO DE DADOS ESCOLAR

Um banco de dados é armazenado e gerenciado por um Sistema Gerenciador de Banco de Dados (SGBD). Este sistema tem o objetivo de permitir a definição da sua estrutura de dados, isto é, tabelas e relacionamentos, e gerenciar o acesso a estas informações. O SGBD possui mecanismos de consultas que possibilitam a inserção, alteração, exclusão e listagem das informações armazenadas por ele.

Este curso utilizará o SGBD MySQL para ilustrar todos os aspectos práticos de um sistema de banco de dados. A escolha deste SGBD se dá pelo fato de o mesmo ser distribuído gratuitamente e pela sua simplicidade e facilidade de uso. Este sistema pode ser obtido a partir do site <http://www.mysql.com/downloads>.

2.1. CRIANDO O BANCO DE DADOS ESCOLAR ATRAVÉS DO MySQL

O MySQL é um SGBD que será utilizado durante todo o curso com o intuito de ilustrar o funcionamento prático de um banco de dados. Além de ser obtido gratuitamente a sua utilização é bastante simples, facilitando o entendimento do mesmo e possibilitando uma melhor aprendizagem dos conceitos discutidos neste material.

O objetivo desta seção é criar, dentro do MySQL, o banco de dados descrito na seção 1. Aqui serão apresentados os passos a serem seguidos a fim de se realizar esta tarefa.

Para manipular o SGBD é preciso iniciar uma conexão com o mesmo. Esta conexão estabelece a ligação entre o SGBD e a aplicação que manipulará os dados. Vale ressaltar que o SGBD pode estar em uma máquina colocada em qualquer lugar do planeta, e ainda assim ser acessada remotamente pela aplicação.

Para estabelecer a conexão é preciso ter um usuário e uma senha para realizar o acesso. Este sistema possui um usuário padrão chamado root e que não possui senha, não é necessário fornecer uma senha para efetuar a conexão com o servidor. Este usuário root é o administrador do banco de dados e possui autorização para realizar qualquer operação dentro do MySQL. Este é diferente do root do Linux, que é o responsável pela administração do sistema operacional. Possivelmente o acesso ao Linux durante o curso não será feito através deste usuário. Porém, como existe o administrador do banco, que por coincidência se chama root, será possível realizar qualquer operação no MySQL, mesmo não tendo acesso de administrador no Linux.

Para estabelecer esta conexão, primeiro deve-se executar um terminal no Linux. Neste ambiente existem aplicações gráficas, apresentando janelas, ícones, imagens e etc. Por

outro lado existem aplicações que não apresentam estes elementos gráficos e, portanto, são executadas a partir de um terminal. Utilizando-se a interface gráfica KDE, deve-se proceder conforme indicado na **Figura 1**.

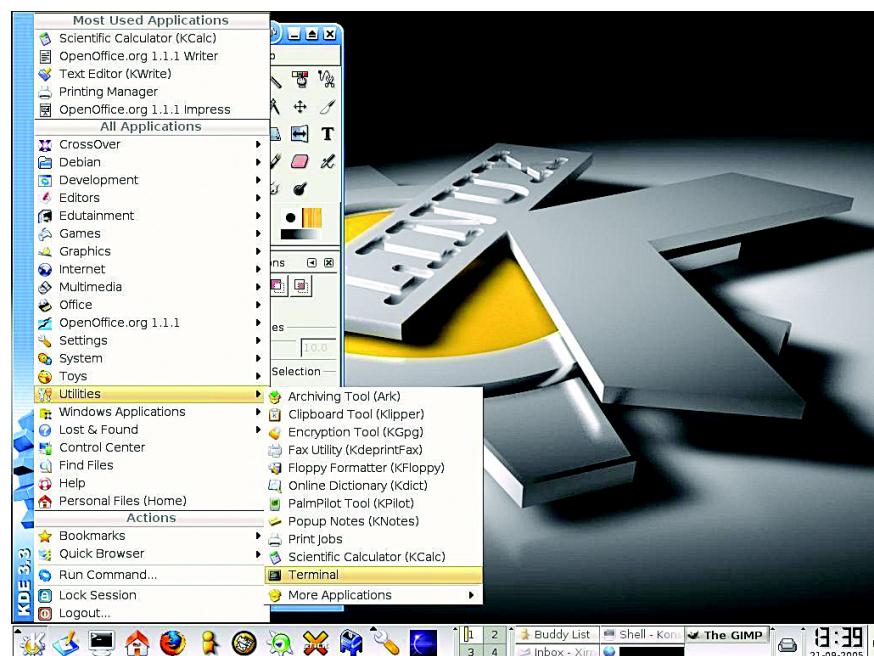


Figura 1
Abrindo um
terminal no
Linux

Feito isto, o terminal se abrirá e então a conexão com o MySQL poderá ser estabelecida conforme a **Figura 2**.

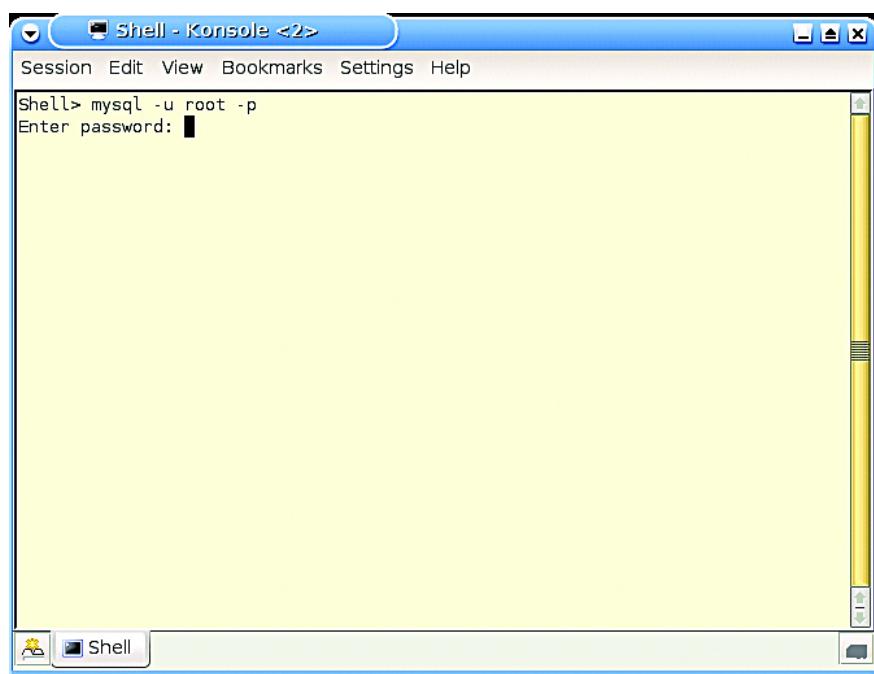
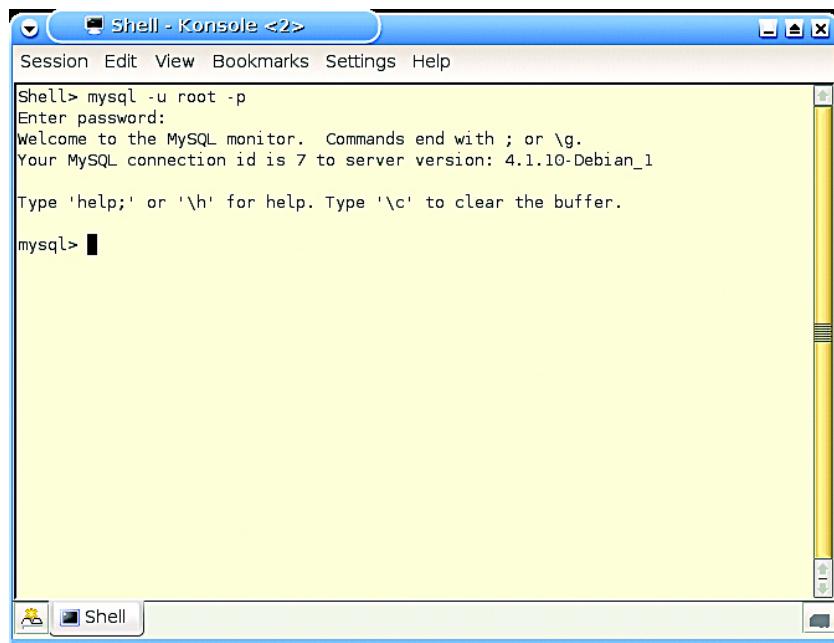


Figura 2:
Estabelecendo
a conexão com
o MySQL

O sistema solicitará uma senha para o usuário root, como a mesma é vazia, basta digitar a tecla <ENTER>, e a tela exibida na **Figura 3** será mostrada.

Figura 3:
Conexão com o MySQL estabelecida com sucesso



```
Shell - Konsole <2>
Session Edit View Bookmarks Settings Help
Shell> mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 7 to server version: 4.1.10-Debian_1
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

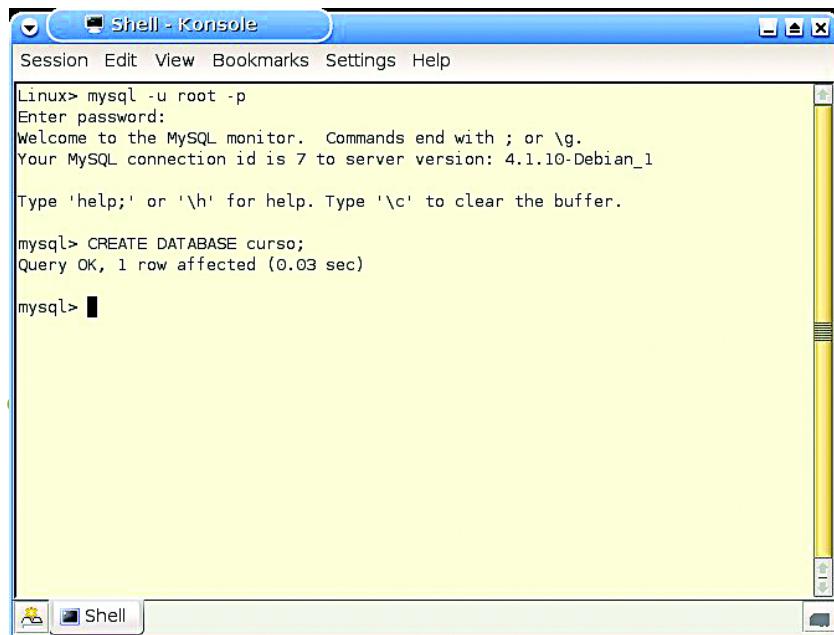
mysql>
```

Percebe-se que o sistema exibe uma mensagem de boas-vindas e algumas informações a respeito do SGBD, tais como a sua versão. A partir deste momento, é possível submeter os comandos para a criação do banco de dados e tabelas. Vale observar que o prompt "mysql>", indica que o SGBD MySQL está sendo utilizado.

2.2. CRIANDO E SELECIONANDO UM BANCO DE DADOS NO MySQL

Uma vez estabelecida uma conexão com o SGBD, o próximo passo é criar um banco de dados e selecioná-lo para uso. Para isto devem-se utilizar os comandos CREATE DATABASE e USE, conforme ilustrado na **Figura 4**.

Figura 4: Criando e selecionando o banco de dados curso



```
Shell - Konsole <2>
Session Edit View Bookmarks Settings Help
Linux> mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 7 to server version: 4.1.10-Debian_1
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

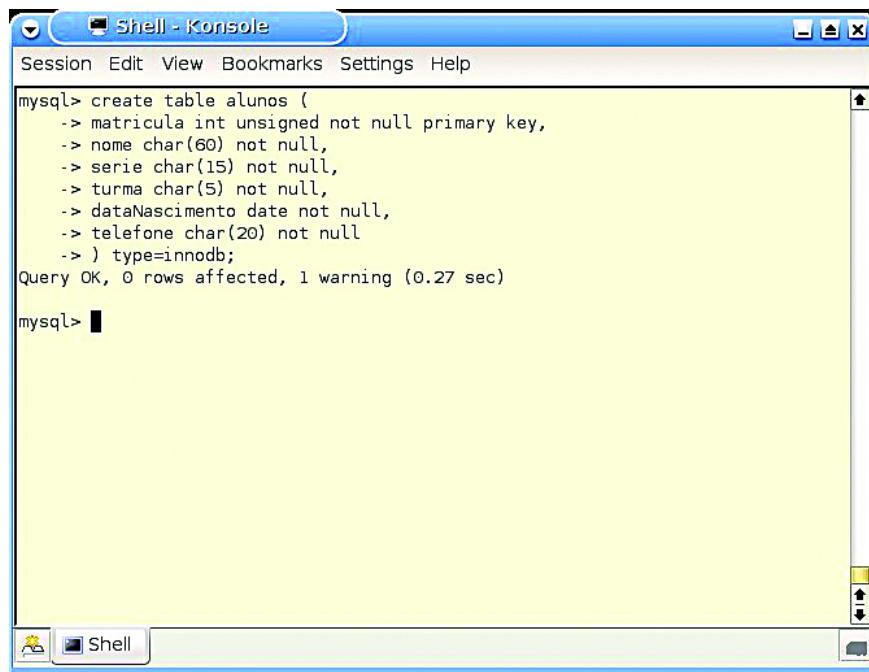
mysql> CREATE DATABASE curso;
Query OK, 1 row affected (0.03 sec)

mysql>
```

O esquema ilustrado define um banco de dados chamado curso e o seleciona para ser utilizado nos passos seguintes. Vale ressaltar que um banco de dados neste SGBD representa uma coleção de tabelas.

2.3. CRIANDO AS TABELAS DEFINIDAS NO MODELO

Uma vez criado e selecionado o banco de dados, é possível definir as tabelas que constituem o modelo. O comando `CREATE TABLE` é empregado para especificar os atributos e seus domínios (ou tipos), bem como as restrições de chave primária e estrangeiras conforme discutido nas seções 1.2 e 1.4. A **Figura 5** ilustra o comando para a criação da tabela alunos.

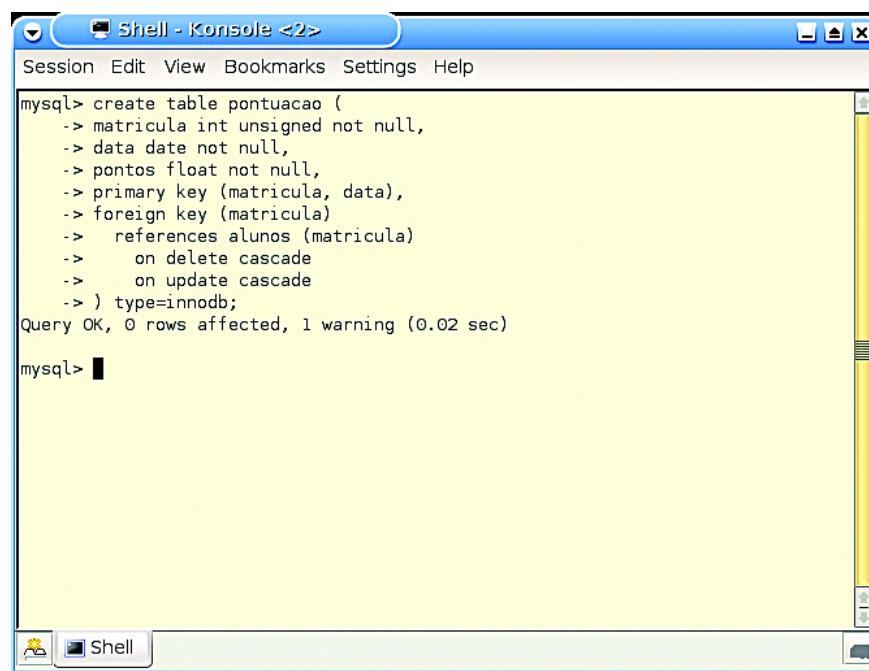


```
Shell - Konsole
Session Edit View Bookmarks Settings Help
mysql> create table alunos (
-> matricula int unsigned not null primary key,
-> nome char(60) not null,
-> serie char(15) not null,
-> turma char(5) not null,
-> dataNascimento date not null,
-> telefone char(20) not null
-> ) type=innodb;
Query OK, 0 rows affected, 1 warning (0.27 sec)

mysql> ■
```

Figura 5:
Criação da
tabela de
alunos

A tabela que armazenará a relação de pontos dos alunos é criada conforme a **Figura 6**. Observe que nela estão definidas as restrições de chave estrangeira desta entidade.



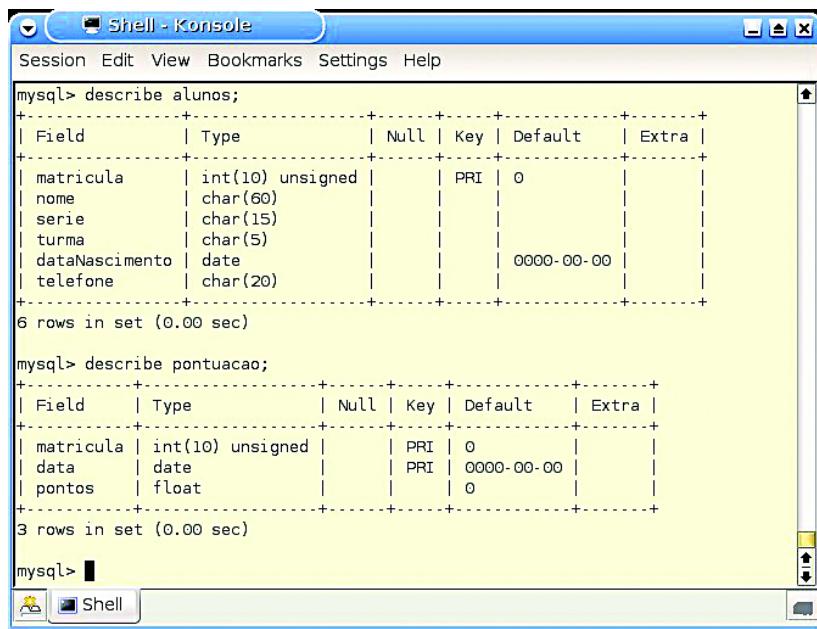
```
Shell - Konsole <2>
Session Edit View Bookmarks Settings Help
mysql> create table pontuacao (
-> matricula int unsigned not null,
-> data date not null,
-> pontos float not null,
-> primary key (matricula, data),
-> foreign key (matricula)
-> references alunos (matricula)
-> on delete cascade
-> on update cascade
-> ) type=innodb;
Query OK, 0 rows affected, 1 warning (0.02 sec)

mysql> ■
```

Figura 6:
Criação da
tabela de
pontuações

Tendo executado os comandos para a criação das tabelas, podem-se examinar as suas estruturas através do comando `DESCRIBE`, como ilustrado na **Figura 7**.

Figura 7:
Verificando a estrutura das tabelas através do comando DESCRIBE



```

Shell - Konsole
Session Edit View Bookmarks Settings Help
mysql> describe alunos;
+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| matricula | int(10) unsigned |    | PRI | 0      |
| nome       | char(60)          |    |      |         |
| serie      | char(15)          |    |      |         |
| turma     | char(5)           |    |      |         |
| dataNascimento | date |    |      | 0000-00-00 |
| telefone   | char(20)          |    |      |         |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> describe pontuacao;
+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| matricula | int(10) unsigned |    | PRI | 0      |
| data       | date             |    | PRI | 0000-00-00 |
| pontos    | float            |    |      | 0      |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

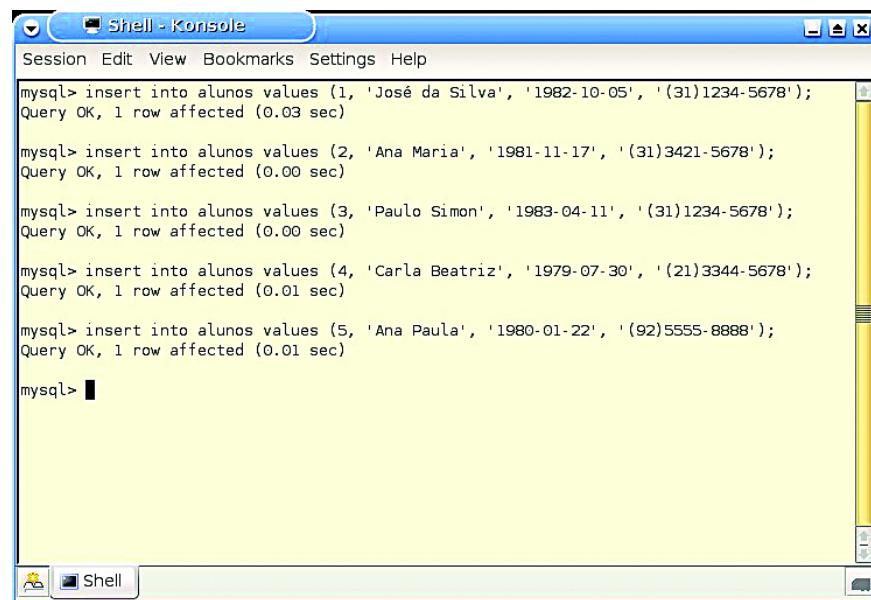
mysql> ■

```

2.4. INSERINDO INFORMAÇÕES NO BANCO DE DADOS

O objetivo desta seção é ilustrar a inserção dos registros exibidos nas **Tabelas 1 e 2**, apresentadas na seção 1. A inserção de dados consiste em informar valores para cada atributo da entidade, armazenando as ocorrências de cada conjunto na base de dados. Para realizar esta operação utiliza-se o comando **INSERT**, conforme ilustram as **Figuras 8 e 9**.

Figura 8:
Inserção dos dados na tabela de alunos



```

Shell - Konsole
Session Edit View Bookmarks Settings Help
mysql> insert into alunos values (1, 'José da Silva', '1982-10-05', '(31)1234-5678');
Query OK, 1 row affected (0.03 sec)

mysql> insert into alunos values (2, 'Ana Maria', '1981-11-17', '(31)3421-5678');
Query OK, 1 row affected (0.00 sec)

mysql> insert into alunos values (3, 'Paulo Simon', '1983-04-11', '(31)1234-5678');
Query OK, 1 row affected (0.00 sec)

mysql> insert into alunos values (4, 'Carla Beatriz', '1979-07-30', '(21)3344-5678');
Query OK, 1 row affected (0.01 sec)

mysql> insert into alunos values (5, 'Ana Paula', '1980-01-22', '(92)5555-8888');
Query OK, 1 row affected (0.01 sec)

mysql> ■

```

Figura 9:
Inserção dos
dados na
tabela de
pontuações

```

Shell - Konsole
Session Edit View Bookmarks Settings Help
mysql> insert into pontuacao values (1, '2004-03-25', 5.5);
Query OK, 1 row affected (0.00 sec)

mysql> insert into pontuacao values (2, '2004-03-25', 6);
Query OK, 1 row affected (0.00 sec)

mysql> insert into pontuacao values (3, '2004-03-25', 8);
Query OK, 1 row affected (0.00 sec)

mysql> insert into pontuacao values (4, '2004-03-25', 10);
Query OK, 1 row affected (0.00 sec)

mysql> insert into pontuacao values (5, '2004-03-25', 7.8);
Query OK, 1 row affected (0.00 sec)

mysql> insert into pontuacao values (1, '2004-05-18', 4.6);
Query OK, 1 row affected (0.01 sec)

mysql> insert into pontuacao values (2, '2004-05-18', 7.2);
Query OK, 1 row affected (0.00 sec)

mysql> insert into pontuacao values (3, '2004-05-18', 9.5);
Query OK, 1 row affected (0.00 sec)

```

Neste caso, foram inseridos cinco alunos, bem como os oito registros de notas destes alunos. Vale ressaltar que o MySQL armazena datas no formato “ano-mês-dia”, por isto, na hora de informar a data de nascimento dos alunos, bem como a data das avaliações, as mesmas foram apresentadas no formato compreendido pelo SGBD. Com isto, tem-se a base de dados preenchida e a partir daí torna-se possível extrair as informações nela contida.

2.5. EXIBINDO AS INFORMAÇÕES CONTIDAS NO BANCO DE DADOS

Nesta seção é ilustrado o mecanismo de consultas do banco de dados. O objetivo deste processo de extração de informações é de extrema importância para que se possa acessar o conteúdo do banco de dados. Para consultar as informações deve-se definir 3 elementos básicos: 1- as colunas ou atributos que se deseja listar, 2- a(s) tabela(s) que possui(em) 3- os dados, e finalmente os critérios para a busca dos registros.

A **Figura 10** exibe duas consultas que listam todas as colunas e registros das tabelas de alunos e pontuações, respectivamente.

Figura 10:
Listando o
conteúdo das
tabelas de
alunos e
pontuações

```

Shell - Konsole
Session Edit View Bookmarks Settings Help
mysql> select * from alunos;
+-----+-----+-----+
| matricula | nome      | dataNascimento | telefone      |
+-----+-----+-----+
| 1 | José da Silva | 1982-10-05 | (31)1234-5678 |
| 2 | Ana Maria    | 1981-11-17 | (31)3421-5678 |
| 3 | Paulo Simon   | 1983-04-11 | (31)1234-5678 |
| 4 | Carla Beatriz | 1979-07-30 | (21)3344-5678 |
| 5 | Ana Paula    | 1980-01-22 | (92)5555-8888 |
+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> select * from pontuacao;
+-----+-----+-----+
| matricula | data      | pontos |
+-----+-----+-----+
| 1 | 2004-03-25 | 5.5 |
| 1 | 2004-05-18 | 4.6 |
| 2 | 2004-03-25 | 6 |
| 2 | 2004-05-18 | 7.2 |
| 3 | 2004-03-25 | 8 |
| 3 | 2004-05-18 | 9.5 |
| 4 | 2004-03-25 | 10 |
| 5 | 2004-03-25 | 7.8 |
+-----+-----+-----+
8 rows in set (0.00 sec)

mysql>

```

Para ilustrar a utilização de critérios para a recuperação dos dados, suponha uma listagem do nome de todos os alunos que nasceram depois de 1980. A **Figura 11** ilustra a consulta para a resolução desta pergunta.

Figura 11:
Listagem do
nome dos
alunos que
nasceram
depois de
1980

```
Shell - Konsole
Session Edit View Bookmarks Settings Help
mysql> select nome from alunos where dataNascimento >= '1980-01-01';
+-----+
| nome |
+-----+
| José da Silva |
| Ana Maria |
| Paulo Simon |
| Ana Paula |
+-----+
4 rows in set (0.00 sec)

mysql> ■
```

Finalmente, é possível extrair informações armazenadas em tabelas relacionadas, especificando mais de uma tabela na cláusula FROM. Desta forma, a ligação será feita pelos atributos em comum entre as entidades. No exemplo, a coluna matrícula é este meio de ligação. Portanto, deseja-se listar o nome dos alunos e as notas obtidas por eles em todos os testes aplicados no mês de março de 2004. A **Figura 12** fornece a solução para este problema.

Figura 12:
Listagem do
nome e dos
pontos obtidos
em março de
2004

```
Shell - Konsole <2>
Session Edit View Bookmarks Settings Help
mysql> select nome, pontos
    -> from alunos,pontuacao
    -> where alunos.matricula=pontuacao.matricula AND
    -> data between '2004-03-01' AND '2004-03-31';
+-----+-----+
| nome | pontos |
+-----+-----+
| José da Silva | 5.5 |
| Ana Maria | 6 |
| Paulo Simon | 8 |
| Carla Beatriz | 10 |
| Ana Paula | 7.8 |
+-----+-----+
5 rows in set (0.00 sec)

mysql> ■
```

Esta seção mostra os mecanismos de consultas do SGBD, que serão abordados com maiores detalhes em seções futuras. O objetivo aqui é apenas de introduzir os mecanismos de consultas do banco de dados.

3. CONCLUSÕES

Este capítulo discutiu os aspectos básicos de um banco de dados, ilustrando a criação de um sistema simplificado. Estes conhecimentos servem como introdução e facilitam o entendimento de outros aspectos que serão abordados nos próximos capítulos. Além disto, alguns elementos discutidos sucintamente neste módulo serão retomados adiante, a fim de que se tenha um entendimento mais aprofundado dos mesmos. O objetivo principal deste módulo é a introdução e a familiarização do aluno com os conceitos básicos de uma aplicação de banco de dados real.

4. REFERÊNCIAS BIBLIOGRÁFICAS

- Houaiss, Antônio; Villar, Mauro de Salles. **Houaiss Dicionário da Língua Portuguesa.** Rio de Janeiro: Objetiva, 2003.
- MySQL AB: **MySQL 5.0 Reference Manual.** Disponível em: <<http://www.mysql.com/documentation>>. Acesso em: 20 dez. 2005.
- Abiteboul, S. Hull, R., and Vianu, V. [1995] **Foundations of Databases**, Addison-Wesley, 1995.
- Elmasri, Ramez A.; Navathe, Shamkant (200). **Fundamentals of Database System**, Third Edition, Addison-Wesley, Menlo Park, CA.

CAPÍTULO 2

1. INTRODUÇÃO À MODELAGEM DE DADOS

No capítulo 1 foi abordada uma pequena aplicação de banco de dados, onde o objetivo era armazenar informações sobre alunos de uma determinada escola. Vale lembrar que um banco de dados é na verdade uma representação de entidades inerentes a um problema real. Desta forma, é preciso que o banco de dados contenha informações que permitam identificar as peculiaridades deste universo ao qual se pretende modelar.

Os sistemas de grande porte geralmente contêm uma grande quantidade de entidades, isto requer mecanismos que permitam projetar estes bancos de dados de forma fácil e sem perder informações indispensáveis a respeito do problema. Para ilustrar, um sistema para controle de uma empresa possui várias entidades, tais como funcionários, fornecedores, clientes, folha de pagamentos, além do controle de contas a pagar e a receber. Somado a este conjunto de entidades não se pode esquecer das dependências existentes entre as mesmas, e que definem o comportamento do banco de dados a ser desenvolvido. Em geral, para projetar aplicações de grande porte é preciso constituir equipes de pessoas, onde cada um será responsável por uma parte do sistema.

Neste caso, haverá equipes responsáveis pelo levantamento dos requisitos de dados, isto é, identificar as entidades envolvidas e as restrições que se aplicam ao problema. Definida esta estrutura, eventualmente uma outra equipe será responsável por codificar a aplicação que utilizará este banco de dados. É notória a necessidade de comunicação entre equipes diferentes para completar a tarefa de projetar o grande sistema. Daí torna-se imprescindível a existência de uma linguagem padrão, de fácil entendimento, que possibilite que a comunicação se dê de forma precisa e sem que haja perda de informações relevantes.

Portanto, para a modelagem de um banco de dados são utilizados diagramas que permitem descrever de forma simples e universal, todos os aspectos importantes do sistema que se deseja representar.

Este capítulo tem como objetivo apresentar os principais aspectos da modelagem de banco de dados, introduzindo conceitos de integridade referencial, normalização, bem como o modelo Entidade-Relacionamento (ER), que é amplamente aplicado na construção e documentação de sistemas de bancos de dados relacionais.

2. DESCRIÇÃO DE UM SISTEMA PARA CONTROLE DE UMA EMPRESA

Para facilitar o entendimento e ilustrar os elementos que definem um diagrama ER, será abordado um problema relacionado a uma pequena empresa de construção civil. Assim, será introduzido o escopo do problema, ou seja, o seu propósito e as necessidades que este deve atender. Assim, a partir desta descrição deve-se construir um modelo ER para esta aplicação. O problema da construção civil é abordado com maiores detalhes ao longo das próximas seções.

2.1. DESCRIÇÃO DE UMA EMPRESA DE CONSTRUÇÃO CIVIL

A empresa de construção civil tem como objetivo projetar e construir obras tais como prédios, casas, pontes, estradas, para citar algumas de suas atividades. Para isto, é

preciso que a empresa possua pessoas ou funcionários capazes de desempenhar as diversas tarefas relacionadas a este ramo de negócios. Por exemplo, é necessário que a empresa contenha em seu quadro de funcionários engenheiros e arquitetos responsáveis pelo projeto e cálculo da infra-estrutura da obra. Além disto, é necessário ainda que haja pedreiros e mestres de obras que serão incumbidos de executar o projeto definido pela equipe de engenheiros. Finalmente, devem-se ter profissionais como eletricistas, bombeiros hidráulicos e carpinteiros para que a obra possa ser executada com sucesso.

Para facilitar a coordenação dos trabalhos destes profissionais, a empresa organiza as pessoas em equipes de acordo com as suas especialidades. Desta forma, estas equipes são alocadas em uma ou mais obras que estejam sendo desenvolvidas pela empresa de construção civil. Vale ressaltar que para fazer parte de uma equipe a pessoa deve fazer parte do quadro de funcionários da empresa, e caso um funcionário seja afastado da empresa, o mesmo deve ser imediatamente retirado da equipe à qual ele pertença.

Cada equipe possui um gerente, responsável por coordenar os trabalhos delegados a ela, sendo que este deve ser necessariamente um funcionário da própria empresa. É preciso salientar que ocorrem situações onde há mais de uma obra em andamento, simultaneamente. Daí, cada obra terá várias equipes envolvidas, já que várias habilidades são necessárias para executar a construção da mesma.

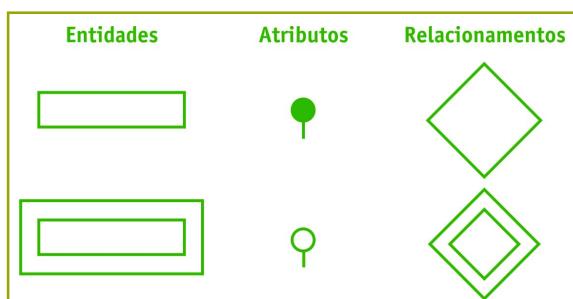
Existem equipes que participam de várias obras ao mesmo tempo, já que as suas tarefas não requerem dedicação exclusiva a um projeto. Este é o caso dos engenheiros, que podem projetar e acompanhar o desenvolvimento de diversas obras simultaneamente, sem que haja comprometimento na qualidade do seu trabalho ou até mesmo prejuízos para o cronograma de execução das mesmas.

Este é um cenário que descreve uma aplicação real de um sistema de banco de dados. Percebe-se que este apresenta várias entidades e restrições que devem ser respeitadas a fim de que o sistema funcione da forma esperada. Para isto, será criado nas próximas seções, um modelo ER que descreva todas as particularidades expostas anteriormente, servindo de base para ilustrar os conceitos que envolvem a modelagem de um banco de dados relacional.

3. ASPECTOS GERAIS DO MODELO ENTIDADE-RELACIONAMENTO

Figura 1:
Símbolos utilizados para o desenho de um diagrama ER

O modelo Entidade-Relacionamento (ER ou MER) constitui-se em uma simbologia que permite representar de forma gráfica os elementos que definem um sistema de banco de dados. Basicamente existem símbolos para representar as entidades e os seus atributos, bem como os seus relacionamentos e restrições. A **Figura 1** ilustra estes componentes empregados para elaborar um diagrama ER.



Na figura o retângulo simples representa uma entidade forte, enquanto o retângulo com borda dupla referencia uma entidade fraca. As entidades fortes são aquelas que não dependem de outras entidades para que existam. Por outro lado as entidades fracas só serão encontradas quando houver uma entidade forte associada a ela, ou seja, são dependentes de outras entidades para que existam.

Como discutido no capítulo 1, toda entidade é descrita por um conjunto de atributos ou colunas, representando as suas qualidades ou características. No exemplo adotado, os alunos eram definidos por um nome, número de matrícula, série, turma e data de nascimento, sendo que o número de matrícula era o identificador único do aluno, conhecido também como chave primária.

Os atributos são representados na **Figura 1** por círculos preenchidos e não preenchidos, posicionados sobre uma pequena linha. Estes símbolos são colocados sobre as entidades as quais eles pertencem, onde sobre eles são especificados os nomes de cada atributo. Os atributos que formam a chave primária da tabela são representados pelos círculos preenchidos, enquanto os demais atributos são representados pelos círculos em branco.

Além das entidades e seus atributos ou colunas, existem ainda relacionamentos que ocorrem entre elas, como é o caso de um funcionário que gerencia uma equipe. Estes relacionamentos são representados por losangos que são conectados por linhas às entidades envolvidas. Da mesma forma que acontece com a representação de entidades, um losango com borda simples ou dupla denota, respectivamente, relacionamentos fortes e fracos. Um relacionamento fraco ocorre quando há a participação de pelo menos uma entidade dependente ou fraca, caso contrário esta relação será considerada forte.

É importante salientar que os relacionamentos assim como as entidades podem conter atributos. Como exemplo, se for necessário armazenar as data de início e término dos trabalhos de uma equipe em uma determinada obra, estas informações pertencem ao relacionamento entre as entidades e não às entidades envolvidas.

Finalmente, no diagrama indica-se a quantidade de participantes de cada entidade no relacionamento através de números colocados sobre as arestas, ou linhas, que conectam as entidades ao relacionamento. Estes números definem a cardinalidade do relacionamento. No exemplo, um funcionário poderá gerenciar apenas uma equipe.

Com isto estão disponíveis os principais elementos que serão utilizados para a construção de um modelo ER. Assim, as seções seguintes elaboram o modelo ER do sistema de construção civil, utilizando os símbolos apresentados e discutidos até aqui. Este diagrama constitui uma linguagem universal, utilizada pelos arquitetos e engenheiros de software para a documentação dos requisitos de cada aplicação modelada.

3.1. DESCRIÇÃO DAS ENTIDADES PROBLEMA

O problema proposto é constituído de várias entidades que estão apresentadas de forma resumida na **Tabela 1**.

ENTIDADE	TIPO	ATRIBUTOS
EMPRESA	FORTE	CÓDIGO, NOME, CNPJ E TELEFONE
OBRAS	FORTE	CÓDIGO, NOME, DATAS DE INÍCIO E TÉRMINO
FUNCIONÁRIOS	FORTE	CPF, NOME, DATA DE NASCIMENTO E ENDEREÇO
CARGOS	FORTE	CÓDIGO E DESCRIÇÃO
EQUIPES	FRACA	CÓDIGO E NOME

Tabela 1: Entidades do banco de dados da construção civil

Para facilitar o trabalho de elaboração do diagrama ER, destacam-se na tabela o tipo de cada entidade, bem como os atributos que as constituem. Estes foram definidos de forma arbitrária, e a definição dos mesmos estará sempre vinculada ao problema abordado, não existindo nenhuma regra formal para a definição de quais atributos existirão.

Observa-se a partir da tabela que os funcionários são uma entidade forte já que existem independentemente das demais entidades do modelo. Já as equipes são consideradas entidades fracas, visto que só existirão caso haja funcionários para a sua constituição.

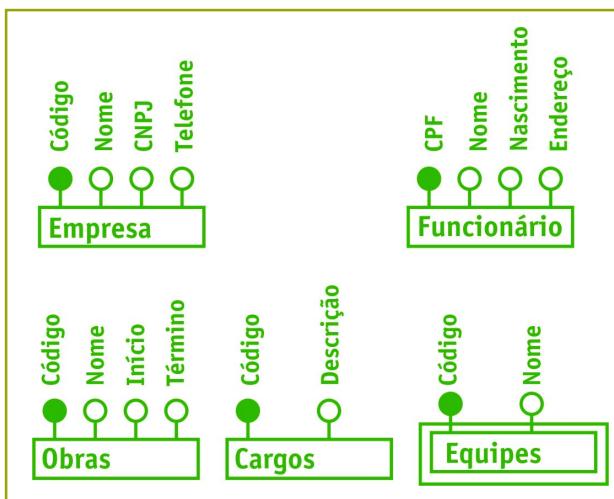


Figura 2:
Diagrama contendo as entidades do modelo e seus atributos

a escolha dos atributos, a definição de qual (is) atributo (s) comporá (ão) a chave primária depende única e exclusivamente das necessidades específicas da aplicação. Ou seja, os requisitos de dados do problema determinarão quais os atributos deverão existir para cada entidade do modelo, bem como quais serão o identificador único ou chave primária da entidade.

3.2. DESCRIÇÃO DOS RELACIONAMENTOS DO SISTEMA

Um relacionamento define uma interação entre duas entidades do modelo. No exemplo, existe um relacionamento entre as entidades funcionários e equipes definindo que toda equipe é gerenciada por um funcionário, e que um funcionário poderá gerenciar apenas uma equipe. Na **Figura 3** está apresentado o diagrama que descreve esquematicamente esta situação.

Todo relacionamento possui uma cardinalidade que define o número de participantes de cada entidade que estarão presentes nesta interação. Neste caso, existem três tipos de relacionamentos:

- 1- Um para um
- 2- Um para muitos
- 3- Muitos para muitos

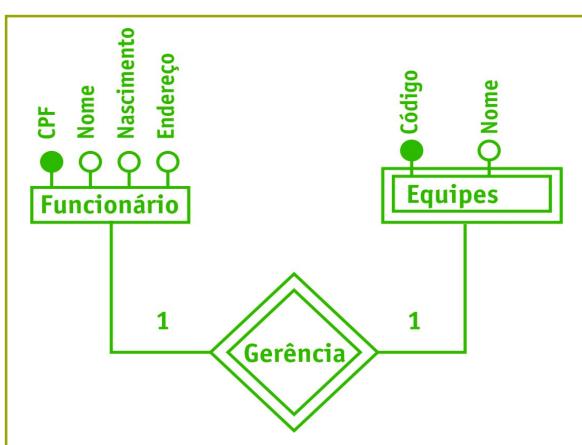


Figura 3:
Relacionamento funcionário gerencia equipe

No relacionamento “funcionário Gerencia equipe”, percebe-se que o mesmo possui cardinalidade de um para um (denotado por 1:1). A cardinalidade é representada por números colocados sobre as arestas que conectam o relacionamento às entidades participantes, conforme ilustra a **Figura 3**. Isto implica que uma equipe possui apenas um

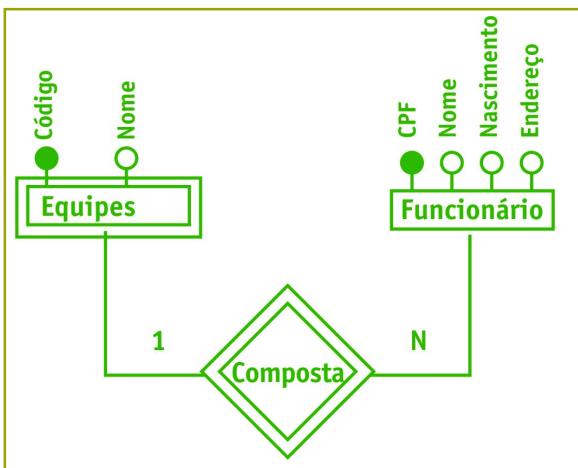


Figura 4:
Relacionamento
equipe é
composta por
funcionários

relacionamentos deste tipo, como é o caso do relacionamento entre empresa e obras. Isto é, a empresa pode desenvolver várias obras simultaneamente.

O terceiro e último tipo de relacionamento são aqueles onde podem existir várias ocorrências de ambas as entidades que participam desta colaboração. Neste caso, a cardinalidade do relacionamento é dita de muitos para muitos, e é denotada por N:M. A **Figura 5** trás o diagrama que representa um relacionamento muitos para muitos, ocorrido entre equipes e obras.

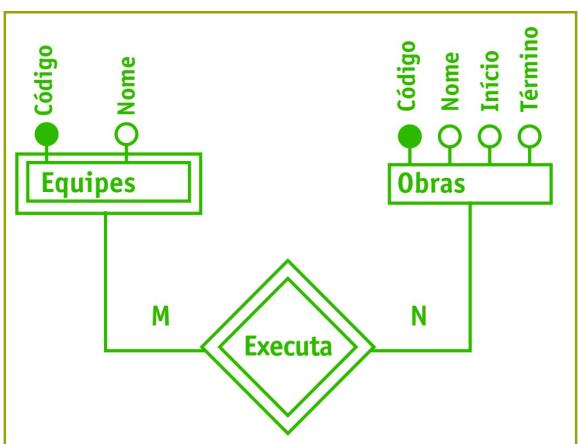


Figura 5:
Relacionamento
de equipes que
executam obras

de uma obra ao mesmo tempo, sem que haja comprometimento do serviço prestado.

Uma vez discutidas as questões ligadas aos relacionamentos, é possível construir então, o diagrama ou modelo ER completo para a aplicação proposta. Assim, deve constar no modelo todas as entidades existentes no problema, seus atributos e relacionamentos. Além disto, é preciso salientar no diagrama a cardinalidade de cada relacionamento, possibilitando que as restrições inerentes ao problema sejam documentadas, propagadas e controladas.

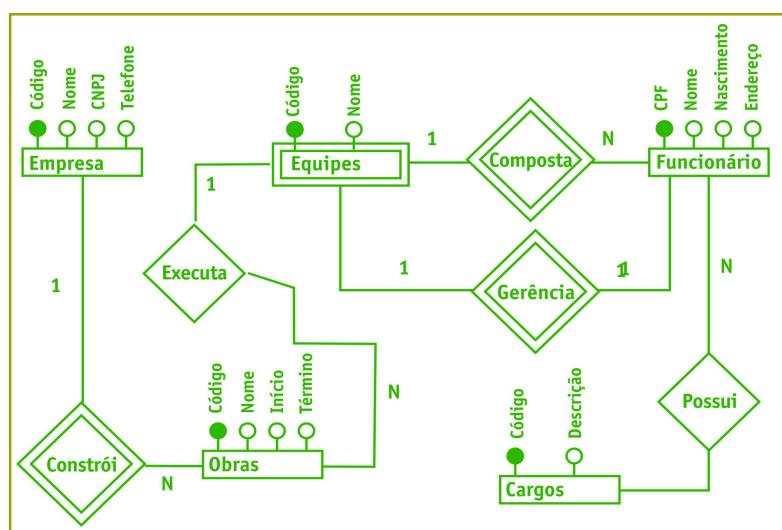


Figura 6:
Modelo ER
completo da
aplicação
proposta

gerente, e ao mesmo tempo um funcionário poderá gerenciar apenas uma equipe.

Existem situações em que pode haver a participação, no relacionamento, de um ou mais elementos de uma das entidades presentes na interação. Se considerar a formação de uma equipe, nota-se que uma equipe é composta por um grupo de funcionários. Neste caso, o relacionamento é conhecido como um para muitos, sendo denotado por 1:N. A **Figura 4** ilustra esta situação.

De um modo geral, os relacionamentos um para muitos são os mais comumente encontrados em sistemas de bancos de dados. No modelo existem outros

relacionamentos deste tipo, como é o caso do relacionamento entre empresa e obras. Isto é, a empresa pode desenvolver várias obras simultaneamente.

O terceiro e último tipo de relacionamento são aqueles onde podem existir várias ocorrências de ambas as entidades que participam desta colaboração. Neste caso, a cardinalidade do relacionamento é dita de muitos para muitos, e é denotada por N:M. A **Figura 5** trás o diagrama que representa um relacionamento muitos para muitos, ocorrido entre equipes e obras.

No exemplo apresentado na **Figura 5**, é notório que uma obra pode ser executada por várias equipes ao mesmo tempo, de forma a desempenhar as tarefas das mais diversas especialidades. Da mesma forma, uma equipe pode estar desenvolvendo trabalhos em diversas obras, como é o caso da equipe de engenharia. Ou seja, os engenheiros podem monitorar mais

de uma obra ao mesmo tempo, sem que haja comprometimento do serviço prestado.

Uma vez discutidas as questões ligadas aos relacionamentos, é possível construir então, o diagrama ou modelo ER completo para a aplicação proposta. Assim, deve constar no modelo todas as entidades existentes no problema, seus atributos e relacionamentos. Além disto, é preciso salientar no diagrama a cardinalidade de cada relacionamento, possibilitando que as restrições inerentes ao problema sejam documentadas, propagadas e controladas.

das e respeitadas pelos desenvolvedores do sistema. A **Figura 6** contém o diagrama ER completo para a aplicação da construção civil, descrita na seção 2.1.

4. MODELANDO AS RESTRIÇÕES DE DADOS DO MODELO

A fase de modelagem de um banco de dados consiste em identificar os elementos que compõem o problema, bem como a forma como se dá a interação entre os mesmos. Todo modelo deve ressaltar as restrições que se aplicam sobre os elementos que constituem o sistema. Basicamente existem dois tipos de restrições que são aquelas aplicadas aos atributos de cada entidade e as restrições de relacionamento. As seções seguintes apresentam mais detalhes sobre estes aspectos.

4.1. RESTRIÇÕES DE ATRIBUTOS

Conforme descrito anteriormente, uma entidade é definida por um conjunto de atributos ou colunas. Estes atributos armazenam os dados referentes à entidade, por exemplo, cada funcionário tem um nome, CPF, endereço, telefone e data de nascimento. Estas são as informações que definem um funcionário no sistema proposto.

Vale ressaltar que não existem duas pessoas com o mesmo CPF, por isto este atributo é chamado de chave primária, pois identifica unicamente o registro ou funcionário, e não aceita valores duplicados. Esta é a forma utilizada pelos sistemas gerenciadores de banco de dados para garantir a unicidade dos valores armazenados em uma determinada coluna. No diagrama, a chave primária é representada por um círculo preenchido, onde a mesma pode ser composta por mais de um atributo.

Além desta restrição imposta pelo modelo, existem outras regras que devem ser obedecidas. No exemplo, não é permitido armazenar uma data de nascimento que não exista, tal como, 30 de fevereiro. Esta restrição é conhecida como restrição de domínio do atributo, e deve ser garantida pela aplicação. Não há como representar este tipo de informação no diagrama lógico. Este trabalho é realizado ao projetar o modelo ER utilizando uma ferramenta de modelagem, como é o caso do DBDesigner4, que será apresentado no próximo capítulo. Desta forma, cada atributo terá o seu tipo definido pelo conjunto de dados suportados pelo Sistema Gerenciador de Banco de Dados (SGBD), que será utilizado para a solução do problema.

4.2. INTEGRIDADE REFERENCIAL

A integridade referencial se refere ao cumprimento das restrições existentes nos relacionamentos do modelo, isto é, a cardinalidade. Portanto, quando é dito no modelo que uma equipe é composta de funcionários, a aplicação ou o SGBD deve garantir que não haverá nenhuma equipe composta por pessoas que não sejam funcionários. Ou ainda, deve-se fazer com que ao retirar um funcionário que esteja inserido em uma equipe, a referência dele seja também removida da equipe que ele eventualmente participar. Um exemplo clássico da restrição de integridade é o sistema para armazenar os pais e os respectivos filhos de pessoas que trabalham em uma determinada empresa. Ao remover

um pai cadastrado no banco devem-se remover os filhos dele, pois do contrário ficaria um registro de um filho para um pai desconhecido pelo sistema. Isto geraria uma inconsistência na base de dados, já que todo filho possui um pai, mesmo que ele não o conheça por quaisquer razões.

As restrições de integridade podem ser construídas utilizando o conceito de chaves estrangeiras presente na maioria dos SGBD disponíveis no mercado. Neste caso, ao definir as chaves estrangeiras do modelo, na verdade são determinados os comportamentos de cada entidade caso haja uma remoção ou alteração de um dado que participe de um relacionamento qualquer. Desta forma, garante-se que não haverá violação das restrições impostas pelo modelo.

Estas restrições são apresentadas no modelo através da cardinalidade, que é exibida para cada relacionamento. A construção das chaves estrangeiras foi introduzida no capítulo 1 e serão detalhadas no módulo que trata do mapeamento do modelo ER para as tabelas do banco de dados.

4.3. NORMALIZAÇÃO DE DADOS, ELIMINANDO A REDUNDÂNCIA

O conceito de normalização está ligado ao fato de que não se devem manter informações duplicadas dentro do banco de dados. Neste caso, durante o processo de criação do banco a partir do modelo lógico proposto, é preciso tomar o cuidado de eliminar informações redundantes ou duplicadas, nas tabelas que serão criadas.

Para ilustrar esta questão, retoma-se o exemplo do relacionamento entre funcionário e equipe. Ao criar as tabelas de funcionários e equipes o endereço do funcionário poderia ser colocado, erroneamente, em abas as tabelas. Isto seria uma redundância que poderia gerar inconsistências na base de dados. Caso haja uma alteração do endereço do funcionário, esta informação deverá ser modificada na tabela de equipes, o que gera um esforço maior e pode gerar problemas, caso a alteração não seja efetuada em ambos os locais. Neste caso, existem no sistema dois endereços distintos para um mesmo funcionário, o que é incorreto ou inconsistente e pode levar a uma conclusão errada.

Além disto, esta abordagem geraria um desperdício de espaço para armazenar a mesma informação duas vezes. Para evitar estas situações, deve-se eliminar a redundância de dados a fim de se fazer um uso racional do banco de dados. Esta questão será abordada com mais detalhes no capítulo seguinte, que trata da criação do banco de dados a partir do modelo ER.

Vale ressaltar que o modelo ER não apresenta nenhuma simbologia que indique qual o nível de normalização desejado. Mas, existem algumas regras que devem ser aplicadas durante o mapeamento físico do banco de dados. Estas regras serão abordadas nos capítulos seguintes.

5. CONCLUSÕES

Este capítulo apresentou os fatores relacionados à modelagem do banco de dados, exibindo as notações e símbolos empregados para desenhar o modelo lógico do banco de dados. O objetivo é prover o conhecimento mínimo a respeito dos constituintes de um diagrama ER, bem como salientar as restrições que devem estar presentes neste

desenho.

Além disto, os conhecimentos vistos neste capítulo serão amplamente utilizados nos capítulos seguintes para que se possa construir o banco de dados para a aplicação proposta.

6. EXERCÍCIOS DE FIXAÇÃO

- 1- O que você entende por atributo?
- 2- Para que serve uma chave primária de uma entidade?
- 3- Explique a diferença entre entidades fracas e fortes.
- 4- Dê um exemplo de cada uma das entidades do exercício anterior.
- 5- Existem três tipos de relacionamentos, cite-os.
- 6- O que é um relacionamento? Ilustre com um exemplo.
- 7- Crie um modelo ER para representar um sistema para armazenar o nome, endereço (rua, bairro, cidade e estado), telefone e celular de todos os seus amigos.
- 8- Amplie o ER construído no exercício anterior, de forma que seja possível relacionar os seus amigos com as escolas em que eles estudam. Para isto, deverá ser criada a entidade escola e o relacionamento entre elas. Os atributos da entidade escola devem ser escolhidos por você.

7. REFERÊNCIAS BIBLIOGRÁFICAS

- Batini, C., Ceri, S., and Navathe, S. [1992] Database Design: An Entity-Relationship Approach, Benjamin/Cummings, 1992.
- Campbell, D., Embley, D., and Czejdo, B. [1985] "A Relationally Complete Query Language for the Entity-Relationship Model," in ER Conference [1985].
- Dumpala, S., and Arora, S. [1983] "Schema Translation Using the Entity-Relationship Approach," in ER Conference [1983].

CAPÍTULO 3

1. INTRODUÇÃO À FERRAMENTA DE MODELAGEM DBDESIGNER4

O Capítulo 2 apresentou uma visão geral dos mecanismos de modelagem de dados, apresentando os conceitos de entidades, atributos e relacionamentos. Foi discutido o modelo Entidade-Relacionamento (ER), bem como a simbologia empregada neste tipo de diagrama, de forma a prover uma comunicação uniforme sobre a estrutura de dados do problema abordado. Para ilustrar um diagrama ER, foi elaborado um esquema lógico para uma aplicação de uma empresa de construção civil. O final do processo culminou em um diagrama ER completo para a aplicação em questão.

O objetivo principal de um modelo lógico de banco de dados é permitir a sua implantação utilizando um Sistema Gerenciador de Banco de Dados (SGBD) qualquer. Para isto, o modelo deve ser convertido em tabelas, de forma a representar as entidades, atributos, relacionamentos e restrições impostas pelo modelo. Para isto, existem ferramentas ou programas que são construídos com o intuito de facilitar a elaboração destes esquemas facilitando a sua aplicação em um SGBD.

De um modo geral, estas ferramentas apresentam símbolos que permitem representar todos os aspectos de um modelo ER. Como cada entidade de um diagrama ER é compreendida pelo SGBD como uma tabela, estas ferramentas permitem a criação destas tabelas, além de definir as suas colunas ou atributos, bem como definir os seus tipos de dados. Ainda é possível salientar neste modelo quais são os atributos que definem as chaves primárias e estrangeiras pertinentes ao problema.

Desta forma, como existem particularidades entre os diversos SGBD disponíveis no mercado, é preciso escolher a ferramenta de modelagem, geralmente chamadas de ferramentas CASE, que possua suporte para os recursos presentes no SGBD que será utilizado para o desenvolvimento da aplicação.

O objetivo deste capítulo é ilustrar uma ferramenta de modelagem de dados, utilizando-a para a elaboração de um modelo lógico para um problema simplificado. Com isto, formam-se uma base de conhecimentos para que se possa no capítulo seguinte, elaborar o diagrama que representa o sistema de construção civil. Isto é, o modelo ER proposto no capítulo 2 será construído dentro de uma ferramenta de modelagem.

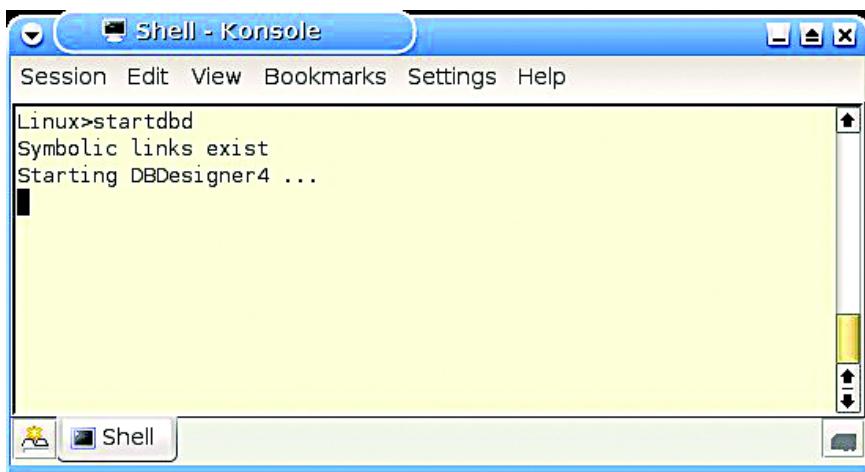
Neste curso, o MySQL foi adotado como o SGBD padrão para ilustrar as aplicações dos bancos de dados propostos no decorrer dos capítulos. Neste cenário, é preciso fazer uso de um software que permita a interface com o SGBD MySQL, permitindo explorar todos os recursos disponíveis no mesmo. Daí optou-se pela ferramenta chamada DBDesigner4, que é distribuída gratuitamente e pode ser encontrada no site <http://fabforce.net>. Esta ferramenta é desenvolvida especificamente para o MySQL e possui suporte ao Sistema Operacional Linux e Windows.

Assim, as seções seguintes fornecem uma visão geral da ferramenta DBDesigner4, apresentando os seus principais recursos, e ilustrando um passo a passo para a sua utilização na elaboração de um modelo simples de banco de dados.

2. INTRODUÇÃO AO DBDESIGNER4

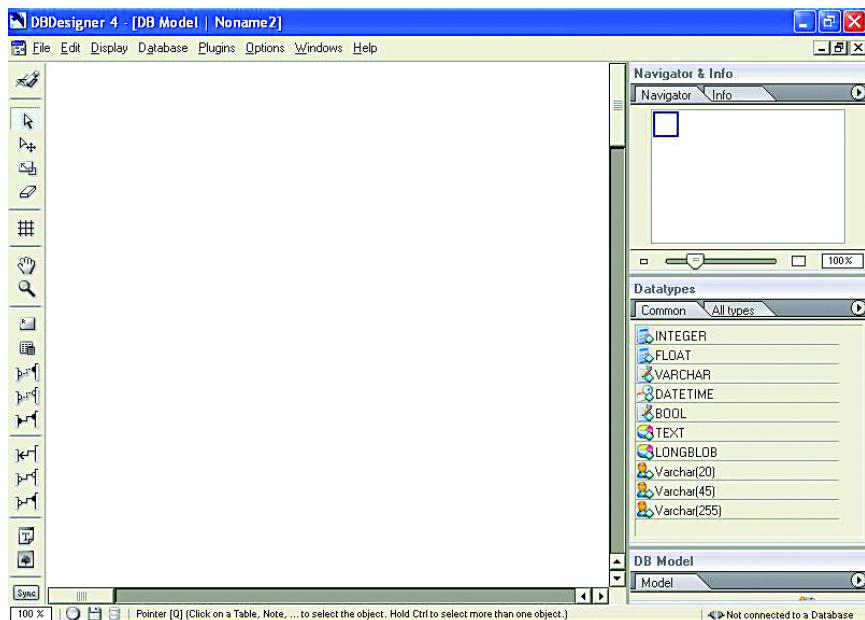
Nesta seção serão explorados os principais recursos desta ferramenta de modelagem, que servirá como instrumento de trabalho no decorrer deste curso. Para iniciar o uso desta ferramenta, é preciso invocá-la a partir de um terminal do Linux, conforme ilustrado na **Figura 1**. Vale ressaltar que o mesmo deve estar configurado no PATH do Linux.

Figura 1:
Iniciando o
uso do
DBDesigner4



Ao executar o comando exibido na figura anterior, o sistema abrirá uma tela com um novo modelo em branco, conforme visto na **Figura 2**.

Figura 2: Janela
principal do
DBDesigner4

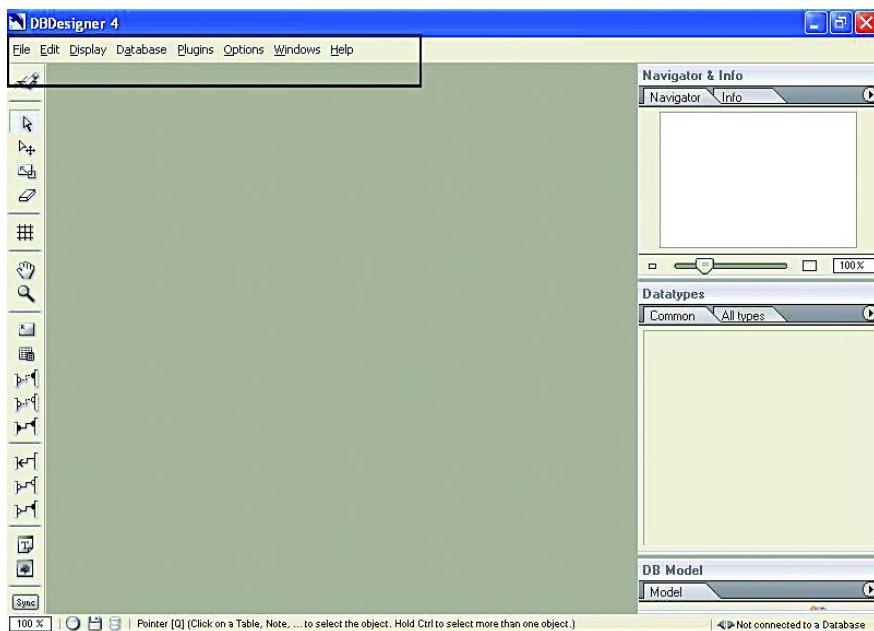


Nas subseções seguintes, serão apresentados os principais recursos disponíveis nesta ferramenta, de forma a permitir uma utilização mínima de seus componentes para efetuar a construção de um modelo de banco de dados.

2.1. APRESENTAÇÃO DOS COMPONENTES DO DBDESIGNER4

A tela do DBDesigner4 exibida na **Figura 2** apresenta um conjunto de elementos que provêm acesso aos mais variados recursos do sistema. No alto da tela encontra-se um conjunto de opções que dá acesso às principais funcionalidades do sistema. A **Figura 3** apresenta a janela do sistema com o destaque para este conjunto de opções.

Figura 3: Menu de opções do DBDesigner4



Existem basicamente oito tens de menu:

- 1- File
- 2- Edit
- 3- Display
- 4- Database
- 5- Plugins
- 6- Options
- 7- Windows
- 8- Help

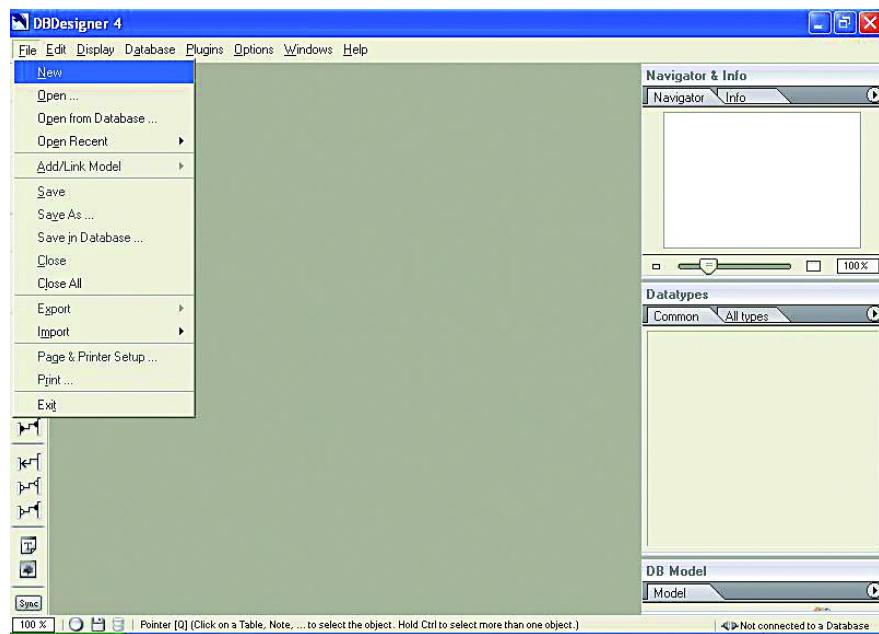
Estas significam respectivamente em português: 1. Arquivo, 2. Editar, 3. Exibir, 4. Banco de dados, 5. Acessórios, 6. Opções, 7. Janelas e 8. Ajuda. As opções File e Database serão descritas com maiores detalhes nas próximas seções, por serem importantes para a construção do modelo. Em relação às demais opções, o Edit apresenta funções para a manipulação do modelo, tais como refazer ou desfazer ações, dentre outras. A função Display permite controlar o formato de exibição do diagrama, isto é, a simbologia para a representação das entidades e relacionamentos. As opções Plugins e Options permitem, respectivamente, adicionar funcionalidades acessórias ao sistema e controlar as configurações do seu sistema de modelagem.

Finalmente, as opções Windows e Help fornecem mecanismos para controlar a disposição das janelas do sistema, bem como o acesso a documentação detalhada do mesmo. Esta documentação pode ser consultada para uma compreensão mais aprofundada do sistema como um todo.

2.1.1. DESCRIÇÃO DA OPÇÃO FILE

Esta opção provê os principais mecanismos para a elaboração de um modelo de banco de dados. Ao acionar este item de menu, será exibido um conjunto de ações do sistema, conforme ilustra a **Figura 4**.

Figura 4:
Explorando a
opção File



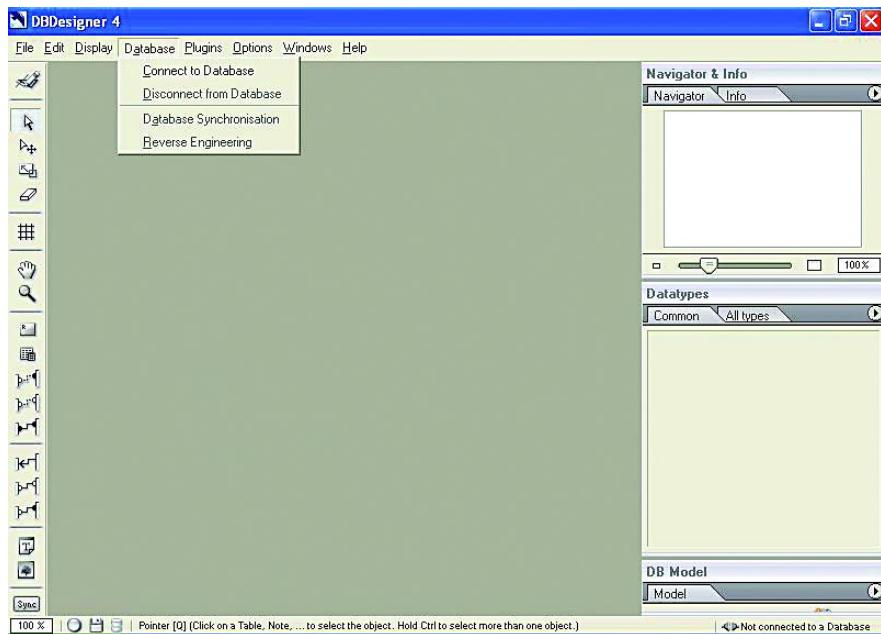
Através do menu File é possível criar um novo modelo por meio do botão New, que em português significa Novo. Um modelo já existente pode ser aberto utilizando as funções Open (em português significa Abrir) ou Open recent (em português significa Abrir recente). Desta forma, é possível manipular um modelo previamente construído. Durante a manipulação do seu modelo é permitido armazenar ou salvar as alterações realizadas por meio das ações Save (em português quer dizer Salvar) e Save as (que quer dizer Salvar como, em português). Além disto, é possível exportar e importar um modelo a partir de um arquivo texto, bem como fechar o modelo (Close, ou fechar em português) atual ou todos os modelos abertos no sistema (Close all, que é Fechar todos em português). Por último, para encerrar o sistema o comando Exit, que quer dizer Sair na língua portuguesa deve ser utilizado.

2.1.2. DESCRIÇÃO DA OPÇÃO “DATABASE”

O DBDesignr4 apresenta dentre outras características interessantes um mecanismo de comunicação do seu modelo lógico com o SGBD que o representa, de forma a possibilitar a sincronização dos dados em ambos os locais. O sincronismo da base de dados permite recuperar eventuais alterações que tenham sido feitas nas tabelas ou no modelo, por outras pessoas que utilizam o sistema. A sincronização neste caso é similar ao ajuste de dois relógios, onde ambos são colocados para apresentar o mesmo horário. Desta forma, o modelo representará a nova base de dados, eventualmente modificada, e vice-versa.

Estas funcionalidades estão disponíveis através do menu Database (banco de dados) exibido na **Figura 5**.

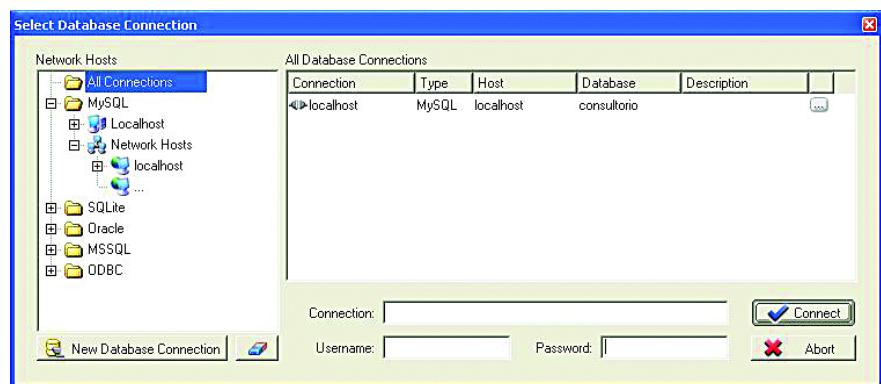
Figura 5:
Explorando a
opção Database



Existem basicamente três funcionalidades importantes em relação a esta opção. A primeira delas permite abrir uma conexão com o banco de dados Connect to database, que quer dizer em português, Conecte-se ao banco de dados. Neste caso, a conexão representa uma ligação estabelecida entre o DBDesigner4 e o MySQL, da mesma forma ilustrada no capítulo 1, utilizando-se o cliente mysql. A segunda opção permite encerrar esta conexão previamente estabelecida Disconnect from database, ou traduzindo, Desconectar-se do banco de dados.

Ao açãoar a opção para conectar-se ao banco de dados a tela exibida na **Figura 6** aparecerá.

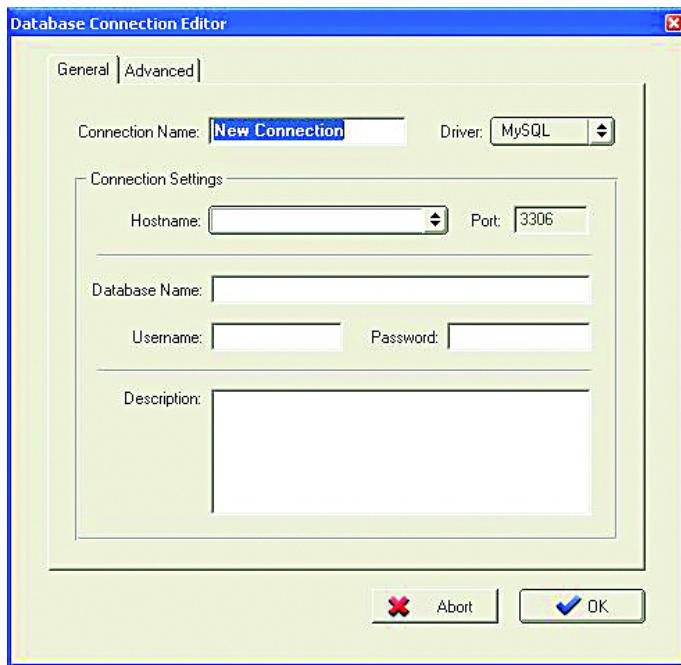
Figura 6: Abrindo
uma conexão com
o banco de dados



Nesta janela serão exibidas todas as conexões disponíveis, e caso haja a necessidade de criar novas conexões, basta utilizar o New Database Connection, significando Nova conexão de banco de dados. A criação de novas conexões é de extrema importância, visto que a ferramenta pode ser utilizada para controlar modelos ER armazenados em servidores de bancos de dados espalhados geograficamente.

Ao criar uma nova conexão será exibida uma janela, conforme ilustra a **Figura 7**.

Figura 7:
Criando uma
nova conexão
com o banco
de dados

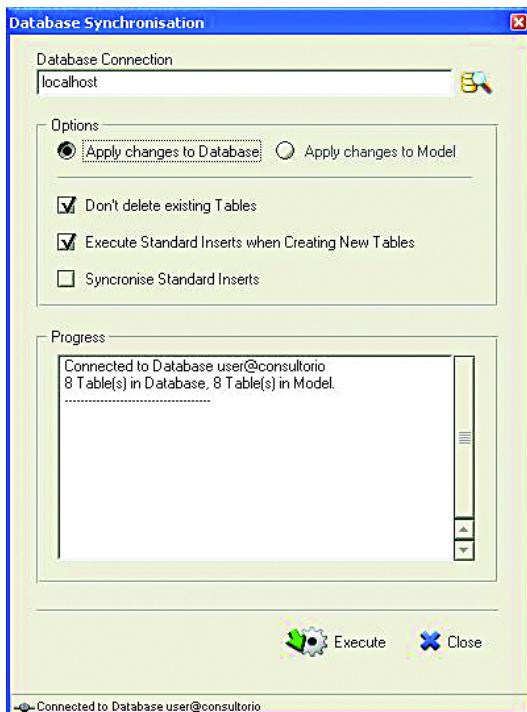


Nesta tela deverão ser informados o nome da conexão (Connection name) e do banco de dados que será manipulado (Database name), além do endereço ou Host onde se encontra o SGBD. Quando o servidor MySQL e o cliente, neste caso o DBDesigner4, encontram-se na mesma máquina utiliza-se o endereço localhost. Finalmente, deve-se informar o usuário (Username) e a senha (Password) para a conexão com o SGBD. Perceba que para o propósito deste curso o Driver será o MySQL. Finalizada a entrada de dados

acione o botão “Ok” e o sistema retornará para a tela exibida na **Figura 6**, só que agora contendo a conexão recém criada. Selecione a conexão desejada e aperte o botão Connect (Conectar), isto iniciará a conexão com o MySQL.

A segunda funcionalidade relacionada ao menu Database é a sincronização com o SGBD, através do item Database synchronisation, que significa Sincronização do banco de dados. Este recurso permite converter o modelo lógico construído em tabelas do seu SGBD. Assim, toda alteração feita no seu modelo será propagada para o seu SGBD de forma automática. Para isto, execute os passos para iniciar a conexão com o banco de dados e escolha a opção Database synchronisation. A tela exibida na **Figura 8** será mostrada neste momento.

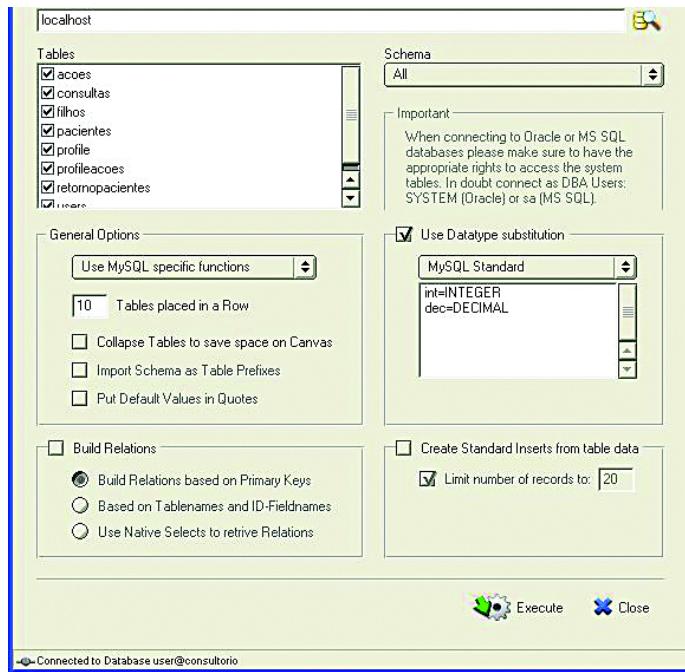
Figura 8:
Sincronização
do modelo com
o SGBD



Portanto, devem ser selecionadas as opções para a sincronização, isto é, se o sistema irá apagar as tabelas existentes no SGBD ou não (Don't delete existing tables, que quer dizer Não apagar tabelas existentes). Definido estes parâmetros basta apertar o botão Execute (Executar) e observar os detalhes da execução do processo, que são exibidos na janela Progress (Progresso).

O terceiro e último recurso relativo à manipulação do banco de dados é o oposto do que foi apresentado anteriormente. Existem situações em que é desejável fazer a conversão da estrutura dos dados armazenada no seu SGBD para um modelo lógico dentro do DBDesigner4. Esta função é realizada pelo Reverse engineering,

Figura 9:
Engenharia
reversa de um
banco de
dados



SGBD que armazenará as informações deste modelo. Desta forma é permitida a manutenção da consistência entre o modelo lógico e físico do seu banco de dados.

ou Engenharia reversa, que é mostrada na **Figura 9**.

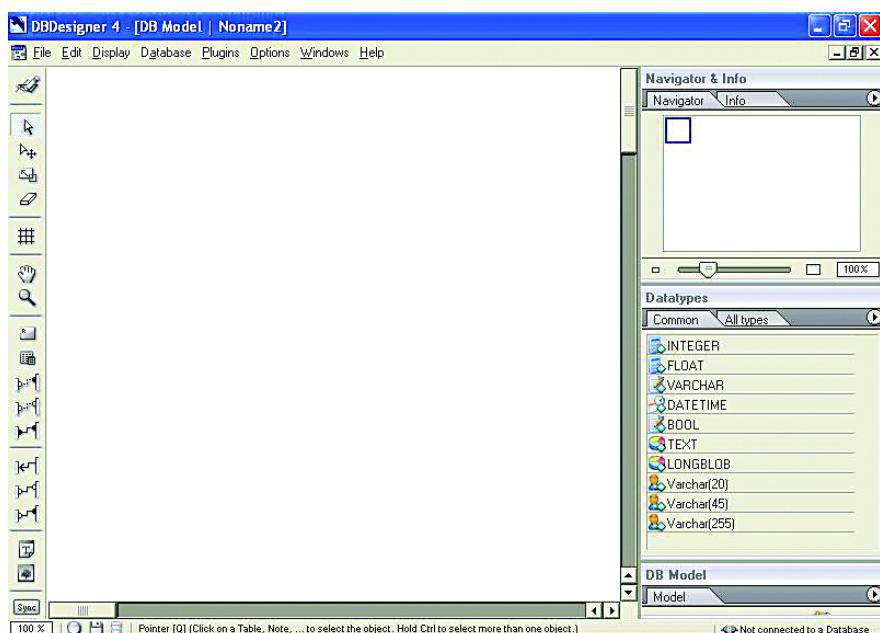
Nesta janela deverão ser informados os parâmetros para que o DBDesigner4 faça a engenharia reversa do seu banco de dados, isto é, construa o modelo a partir dos dados. Basicamente serão informadas as tabelas a serem convertidas e a forma com a qual o DBDesigner4 criará os relacionamentos entre estas tabelas (Build relations – Construir relacionamentos).

Com isto, é possível criar uma interface entre o modelo gerado pelo DBDesigner4 e o

2.2. APRESENTANDO A BARRA DE TAREFAS DO DBDESIGNER4

A barra de tarefas do DBDesigner4 fica posicionada na parte esquerda do sistema, e provê acesso aos principais elementos utilizados para a elaboração do modelo ER, tais como as entidades e os relacionamentos. A **Figura 10** trás uma janela da aplicação com um destaque para esta barra de tarefas. Os componentes desta parte do sistema serão discutidos nas próximas subseções.

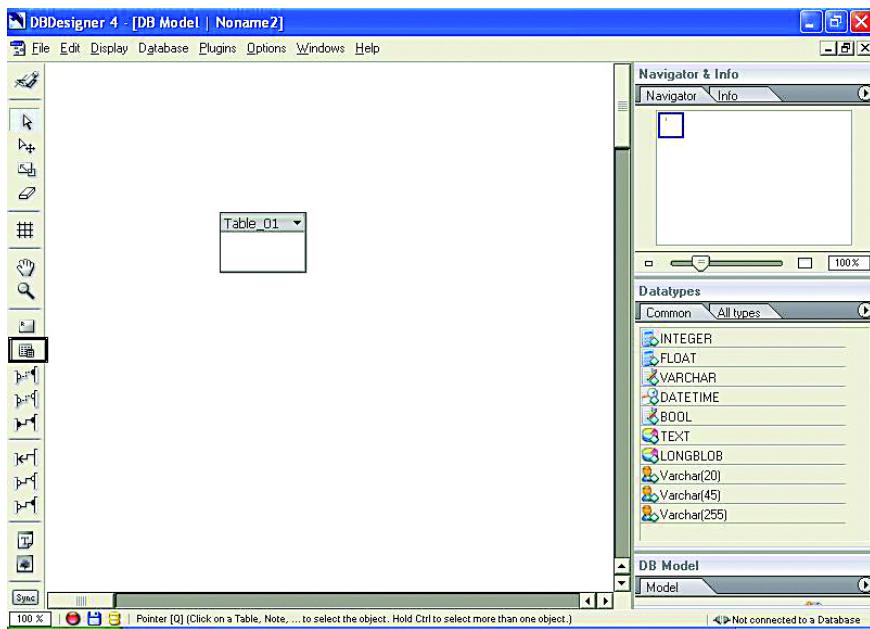
Figura 10: Barra
de tarefas do
DBDesigner4



2.2.1. CRIANDO ENTIDADES NO DBDESIGNER4

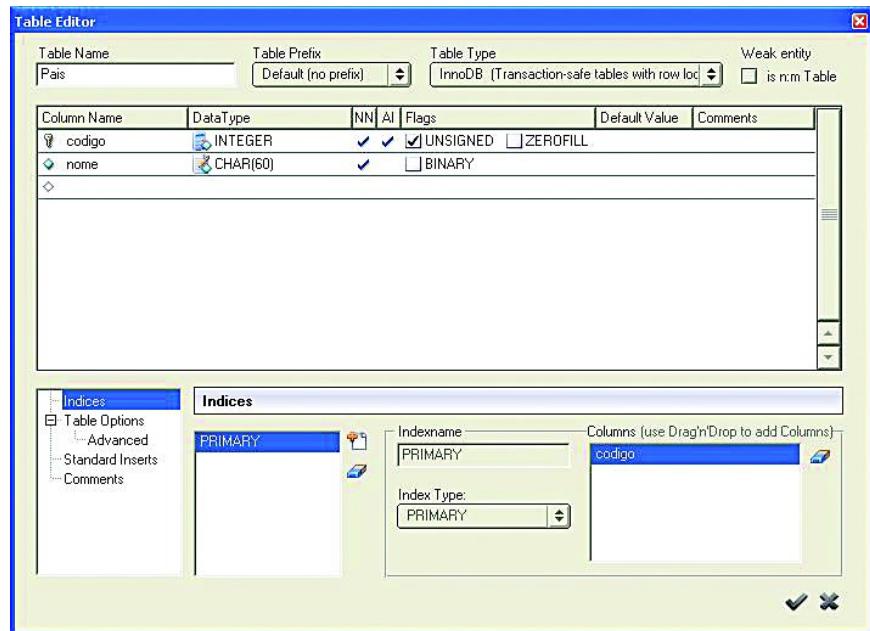
Em uma ferramenta de modelagem toda entidade é representada por uma tabela, facilitando assim a conversão do modelo para dentro do SGBD. Deste modo, para criar uma tabela no modelo basta clicar no botão referente à tabela e clicar na área de desenho da ferramenta, conforme ilustra a **Figura 11**.

Figura 11:
Criando uma
tabela



Perceba que o sistema cria por padrão a tabela com o nome **Table_01**. Para alterar o seu nome e definir os seus atributos basta utilizar um clique duplo sobre a mesma. Desta forma aparecerá a tela ilustrada pela **Figura 12**.

Figura 12:
Tabela para
armazenar os
país



Na figura anterior o nome da tabela ou entidade foi alterado para **Pais**, e foram definidos dois atributos: o código que é um número inteiro e o nome que é uma cadeia de caracteres (texto) de tamanho máximo 60. Vale ressaltar que no editor de atributos as

propriedades variam de acordo com o tipo escolhido. Por exemplo, o INT possui “AI”, que significa Auto Incremento que possibilita a geração de números seqüenciais. Enquanto as colunas textuais não possuem esta propriedade.

Outro detalhe é que os atributos chave primária são identificados por uma chave desenhada ao lado do seu nome, os demais não apresentam esta marcação. No exemplo a chave primária da tabela de pais é o código numérico. Ao definir todos os atributos da tabela basta acionar o botão no formato de um “V” para confirmar a alteração ou o “X” para descartar as mudanças.

Desta forma, para construir todas as entidades do modelo, basta executar o procedimento realizado para a tabela “Pais” para cada tabela do seu sistema. Vale ressaltar que as mesmas podem ser posicionadas em qualquer local da tela de edição do modelo.

2.2.2. DEFININDO OS RELACIONAMENTOS

Existem três tipos de relacionamentos entre as entidades que são um para um, um para muitos e muitos para muitos. No DBDesigner4, para criar este tipo de relacionamento, basta selecionar na barra de tarefas o tipo de relacionamento desejado e clicar nas entidades que formam a relação. A **Figura 13** exibe os componentes para a definição destes elementos.

Para ilustrar a criação de um relacionamento, suponha que fosse necessário armazenar os Pais e os seus respectivos filhos. Desta forma, é natural que o relacionamento seja do tipo um para muitos, já que um pai pode ter mais de um filho associado. Para realizar esta tarefa, acione o relacionamento 1:N, clique na entidade com cardinalidade 1, neste caso Pais. Depois clique na segunda entidade, isto é, com cardinalidade N, que são os Filhos. Feito isto aparecerá a situação ilustrada pela **Figura 14**.

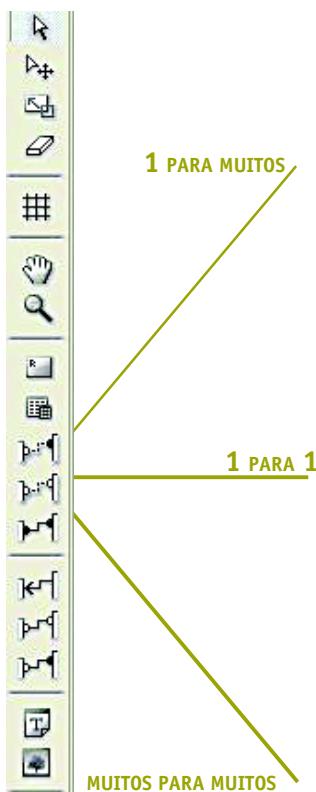


Figura 13: Tipos de relacionamentos no DBDesigner4

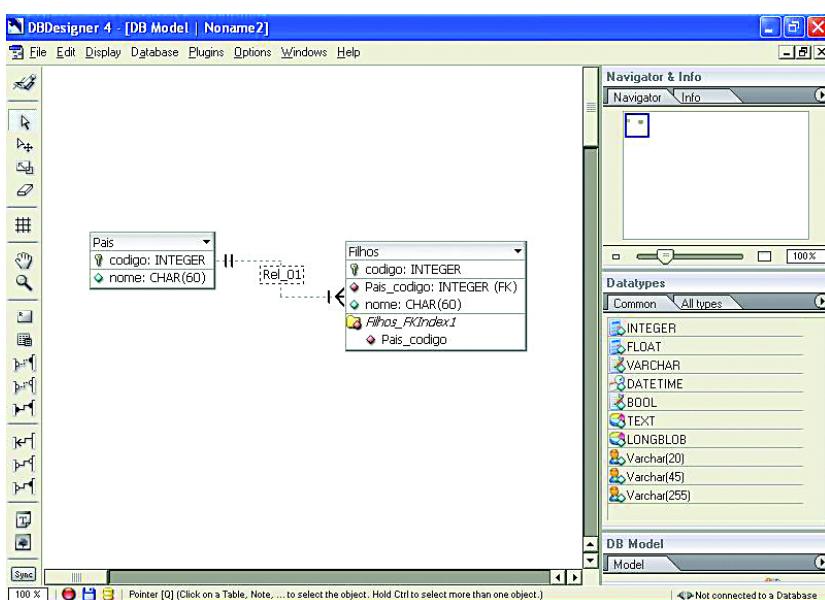
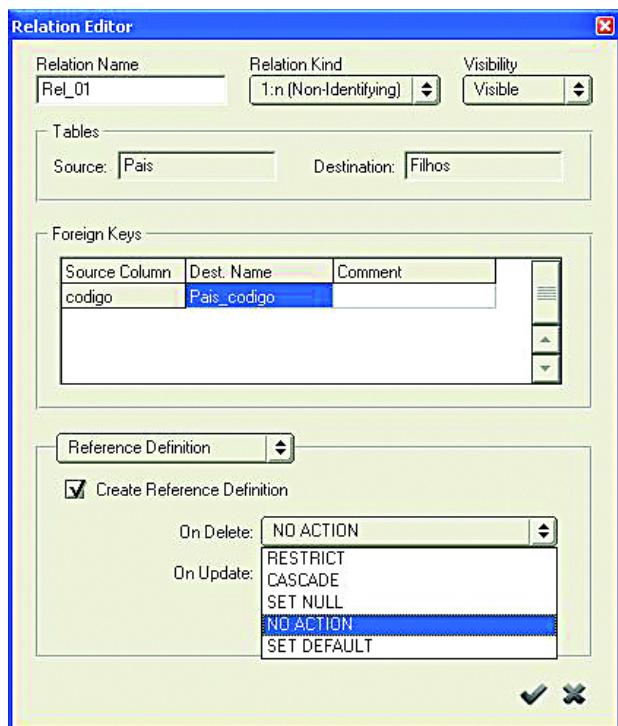


Figura 14:
Relacionamento 1:N
entre País e Filhos



Assim, nota-se o aparecimento de uma conexão entre a tabela Pais e Filhos, com uma marcação que define os lados 1 e N da relação. Para editar as propriedades do relacionamento basta utilizar um clique duplo sobre o mesmo e a janela apresentada na **Figura 15** aparecerá.

Neste caso, é possível definir o nome do relacionamento, bem como as restrições de integridade do mesmo. No exemplo, ao remover ou alterar um Pai que tenha Filhos o sistema poderá assumir a postura de modificar ou excluir todos os filhos (Cascade), ou não permitir a alteração/exclusão de um pai que possua filhos (Restrict). As demais restrições apresentadas na figura são pouco utilizadas e não serão abordadas no contexto deste livro.

Figura 15: Definindo as propriedades do relacionamento Pais e Filhos

3. CONCLUSÕES

O objetivo principal deste módulo foi introduzir os conhecimentos básicos sobre a ferramenta de modelagem DBDesigner4. Desta forma, têm-se condições de compreender e utilizar os elementos que definem as entidades e relacionamentos, dentro do contexto desta ferramenta, e possibilita a sua utilização posterior. Portanto, no capítulo seguinte, esta ferramenta será empregada para o desenho do modelo ER da aplicação concebida no Capítulo 2. Além disto, serão utilizados os recursos de manipulação do banco de dados a fim de mapear o modelo gerado para dentro do SGBD MySQL.

4. EXERCÍCIOS DE FIXAÇÃO

- 1- Descreva com suas palavras qual é o objetivo da ferramenta DBDesigner4.
- 2- Como pode ser obtida esta ferramenta?
- 3- Por que o DBDesigner4 foi escolhido para ser utilizado neste curso? Existe alguma relação com o uso do MySQL?
- 4- Quantas são as opções de menu do DBDesigner4. Cite-as.
- 5- O que você entende por sincronização de banco de dados?
- 6- Descreva o procedimento para criar uma nova conexão com o banco de dados a partir do DBDesigner4.
- 7- Crie uma tabela para armazenar o nome, a idade e o telefone de todos os seus familiares.
- 8- Crie um modelo ER, utilizando o DBDesigner4, para representar o sistema que você construiu no exercício 8 do capítulo 2.

5. REFERÊNCIAS BIBLIOGRÁFICAS

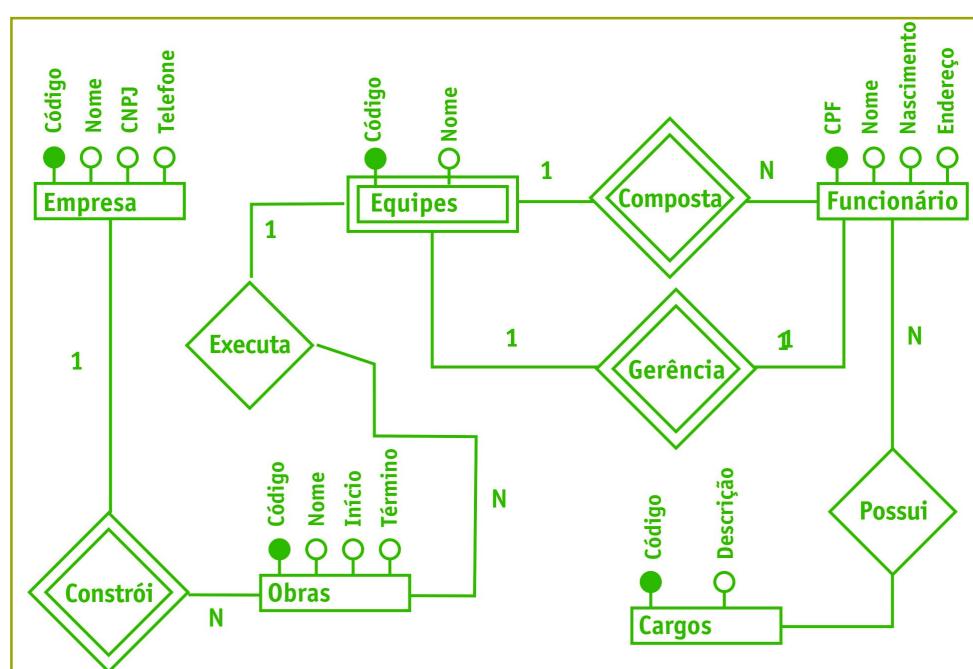
- FabForce.net: DBDesigner4 online documentation. Disponível em: <http://www.fabforce.net/dbdesigner4/doc/index.html>. Acesso em: 20 dez. 2005.

CAPÍTULO 4

1. INTRODUÇÃO

Os capítulos iniciais deste livro tiveram como objetivo a conceituação de um modelo lógico de banco de dados, bem como apresentar uma pequena introdução às aplicações que utilizam estas estruturas de dados. Com o intuito de formalizar os conceitos da modelagem de um sistema real, foi apresentada uma aplicação para controle de uma empresa de construção civil. A discussão culminou com o diagrama ER ou modelo lógico do banco de dados, que por comodidade é ilustrado novamente pela **Figura 1**.

Figura 1: Modelo Entidade-Relacionamento do sistema de construção civil



Posteriormente à definição do diagrama ER, abordou-se os aspectos práticos da utilização de uma ferramenta de modelagem. Este tópico ilustrou os mecanismos para a elaboração de um modelo lógico dentro de um ambiente que permitia a comunicação com o Sistema Gerenciador de Banco de Dados (SGBD).

Portanto, neste capítulo serão utilizados os conhecimentos adquiridos até o momento para que se possa criar o modelo exibido na **Figura 1**, utilizando a ferramenta DBDesigner4. O objetivo final é projetar o diagrama ER e aplicar este modelo ao MySQL, criando um banco de dados que será chamado de curso.

Assim, as seções seguintes ilustram passo a passo, os caminhos que devem ser seguidos para a obtenção deste banco de dados.

2. CRIANDO O BANCO DE DADOS E A CONEXÃO PARA O MESMO

O primeiro passo a ser dado no sentido de construir o banco de dados para a aplicação de construção civil, é justamente criar o banco de dados no MySQL e criar uma conexão no DBDesigner4 que permita acessá-lo. Desta forma, será possível criar um interface com o SGBD, e permitir sincronizar com o MySQL o modelo lógico gerado.

Para criar um banco de dados, inicie uma conexão com o MySQL através de um terminal no Linux e execute o comando CREATE DATABASE, conforme ilustra a **Figura 2**.

Figura 2:
Utilizando o
MySQL para a
criação do banco
de dados curso

```

Linux> mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7 to server version: 4.1.10-Debian_1

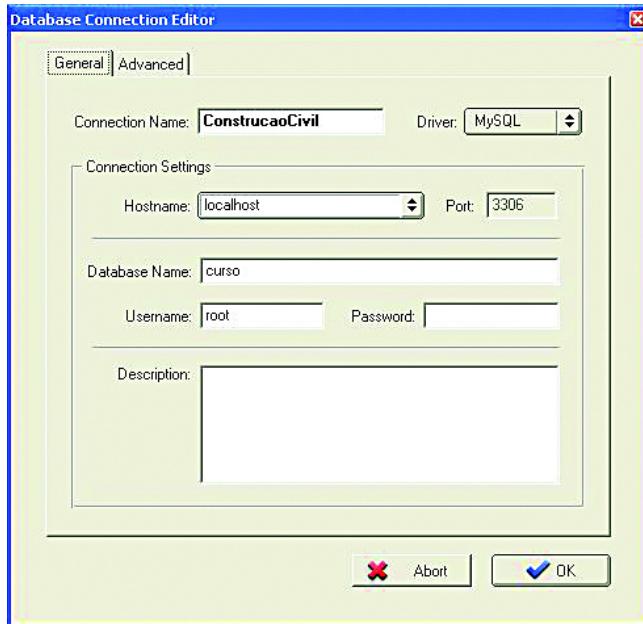
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> CREATE DATABASE curso;
Query OK, 1 row affected (0.03 sec)

mysql> 
```

Feito isto, o passo seguinte é criar uma conexão no DBDesigner4, permitindo que este se comunique com o banco de dados recém criado. A **Figura 3** fornece as configurações da conexão chamada de “ConstrucaoCivil”.

Figura 3:
Conexão
para acesso
ao banco
de dados
curso



Vale ressaltar que o endereço do servidor MySQL é o mesmo onde se encontra o DBDesigner4, por isto foi utilizado “localhost”. O banco de dados é o curso, e o usuário root, sem senha será utilizado para estabelecer a conexão com o MySQL. Feito isto, basta clicar em “Ok”, selecionar a conexão “ConstrucaoCivil” e acionar o botão Connect (Conectar). Assim, estará estabelecida a comunicação do DBDesigner4 com o banco de dados curso.

3. CONSTRUÇÃO DAS ENTIDADES DO MODELO

Nesta seção serão criadas as entidades do modelo proposto, que são Empresa, Cargos, Funcionários, Equipes e Obras. Para isto, a subseção 3.1 faz uma pequena introdução aos tipos de dados suportados pelo MySQL de forma a permitir a criação dos atributos ou colunas de cada entidade identificada anteriormente.

3.1. TIPOS DE DADOS NO MySQL

O MySQL apresenta um vasto conjunto de dados que permite a representação das mais variadas informações existentes no mundo real. Por exemplo, é possível criar uma coluna para armazenar uma música, um vídeo ou até mesmo uma imagem. Além de dados mais comuns, tais como datas, números, letras e textos, dentre outros.

Para simplificar a discussão, serão descritos aqui apenas os tipos de dados relevantes para a solução do problema proposto no início do capítulo. Portanto, nem todos os tipos de dados existentes no DBDesigner4, e por consequência no MySQL, serão descritos aqui. Maiores informações sobre todos os tipos de dados existentes no MySQL podem ser encontradas no site <http://www.mysql.com/documentation>.

Os tipos de dados podem ser agrupados em três grandes grupos que são os números, textos e datas e horas. A **Tabela 1** apresenta um resumo dos tipos de atributos que serão utilizados no modelo proposto.

CATEGORIA	TIPO	DESCRIÇÃO
NÚMEROS	INTEGER	NÚMEROS INTEIROS
	DOUBLE	NÚMEROS REAIS OU PONTO-FLUTUANTE
TEXTOS	CHAR(X)	TEXTO COM TAMANHO MÁXIMO DE X CARACTERES
DATA E HORA	DATE	DATA NO FORMATO AAAA-MM-DD
	TIME	HORA NO FORMATO HH:MM:SS

Tabela 1: Principais tipos de dados do MySQL

Desta forma, é possível definir as restrições de domínio dos seus atributos, uma vez que os mesmos deverão respeitar o tipo de dados definido para ele. Isto é, não será possível armazenar uma informação textual em uma coluna de data, eliminando assim as inconsistências. Tendo em vista os tipos de dados ilustrados nesta seção é possível elaborar o diagrama para todas as entidades do sistema, conforme mostrado na seção 3.2.

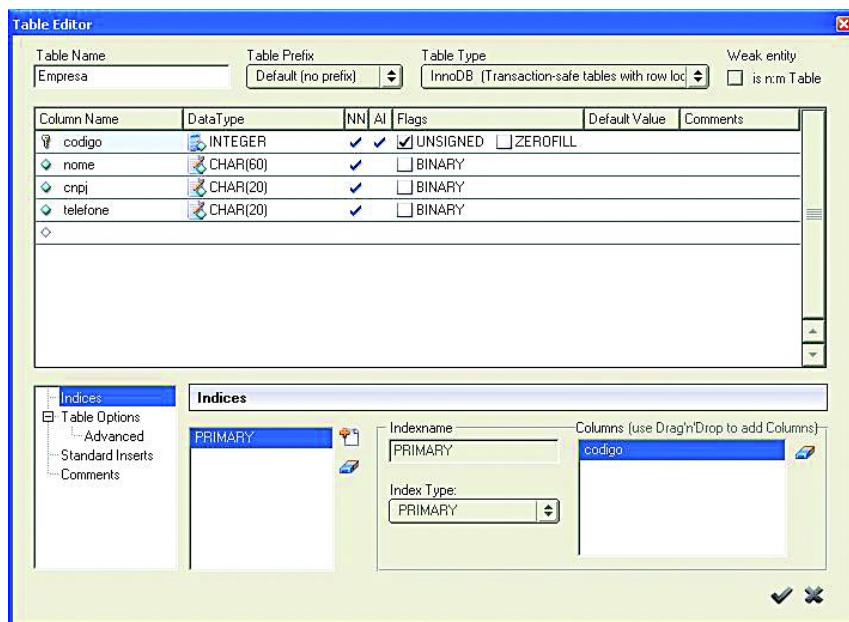
3.2. DEFINIÇÃO DAS ENTIDADES DO SISTEMA

Esta seção descreve a criação das cinco entidades do sistema de construção civil a partir da utilização do DBDesigner4. Vale lembrar que esta ferramenta considera uma entidade como sendo uma tabela, onde as colunas definem os atributos, que por sua vez necessitam de um tipo de dado. Por isto, os nomes entidades e tabelas serão empregados neste contexto como sinônimos, isto é, representam uma mesma informação.

O procedimento completo para a criação de uma tabela está descrito no capítulo 3, e por isto não será exposto novamente. Vale lembrar que todas as tabelas devem ser criadas como InnoDB para que a integridade referencial funcione com sucesso.

A **Figura 4** apresenta a tela de edição de tabelas do aplicativo, onde estão descritos todos os atributos da entidade Empresa.

Figura 4:
Definição da
tabela
Empresa



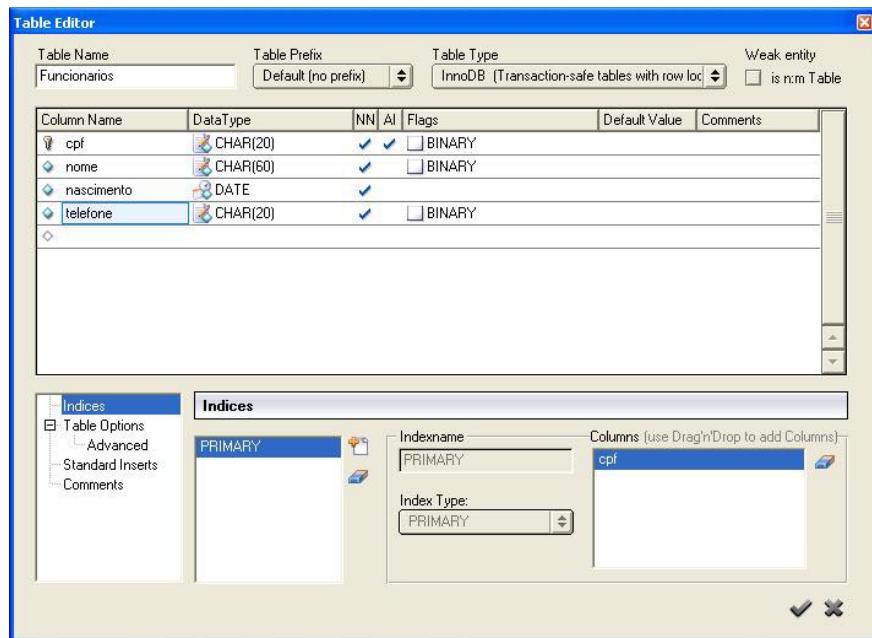
Portanto a empresa é constituída de quatro colunas, uma do tipo numérico representando a chave primária da tabela, e três atributos do tipo texto para armazenar as demais informações. Percebe-se na figura a expansão das opções de tipos para a coluna telefone. É notória a existência de outros tipos que não foram apresentados na **Tabela 1**.

A segunda entidade do modelo são as obras, que também são descritas por um código numérico, um nome e as datas de início e término das mesmas. A **Figura 5** descreve esta tabela.

Figura 5:
Definição da
tabela Obras

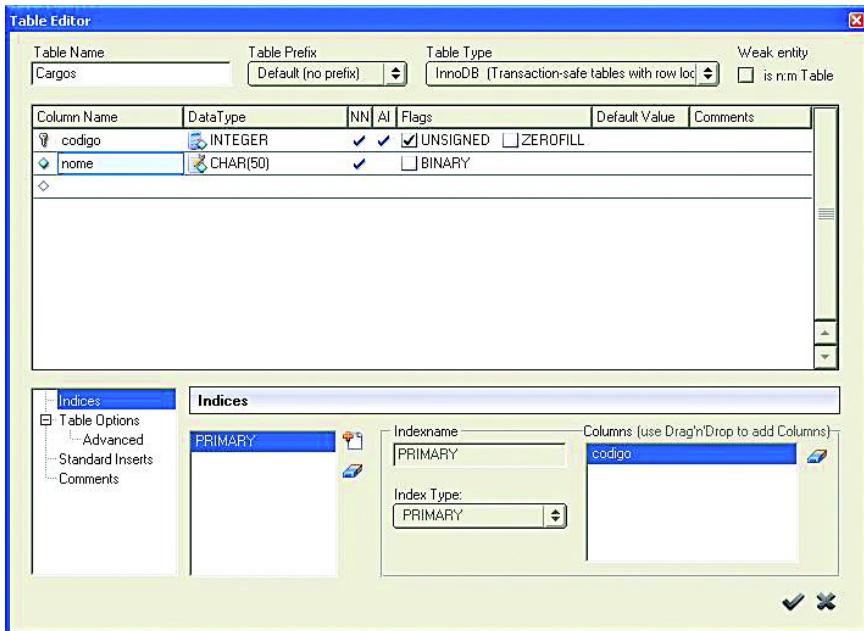
A próxima tabela do sistema é a que armazena os funcionários da empresa. Esta tabela apresenta uma chave primária definida pelo atributo CPF, que será representado por um texto de tamanho 20. Além disto, existem os atributos nome, data de nascimento e telefone de contato do funcionário. A **Figura 6** fornece o esquema deste objeto.

Figura 6:
Definição da
tabela
Funcionários



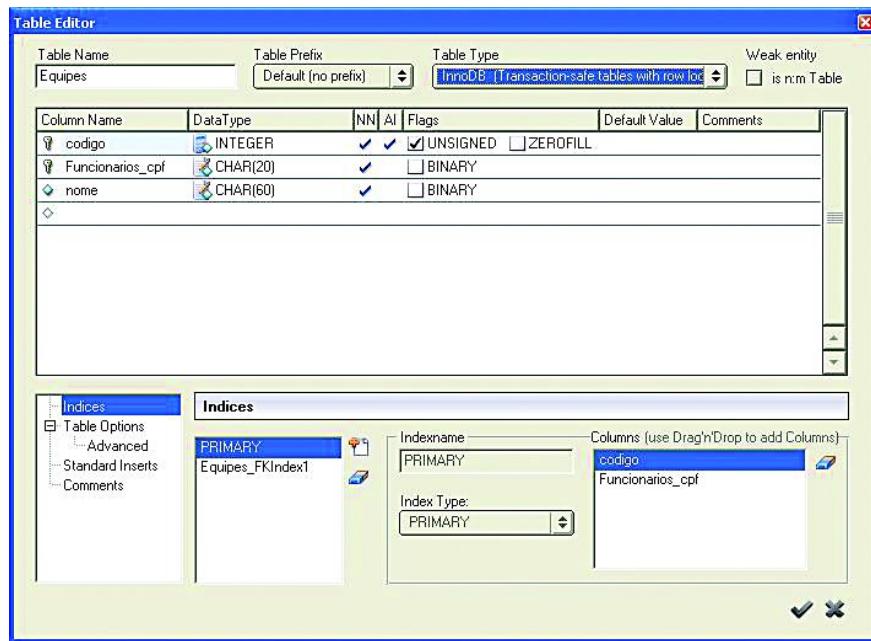
As informações de cargos dos funcionários serão armazenadas na tabela Cargos, que apresenta apenas dois atributos. O primeiro é um código numérico que identifica o cargo, e o segundo é o nome ou descrição do cargo. A **Figura 7** ilustra a definição desta tabela.

Figura 7:
Definição da
tabela
Cargos



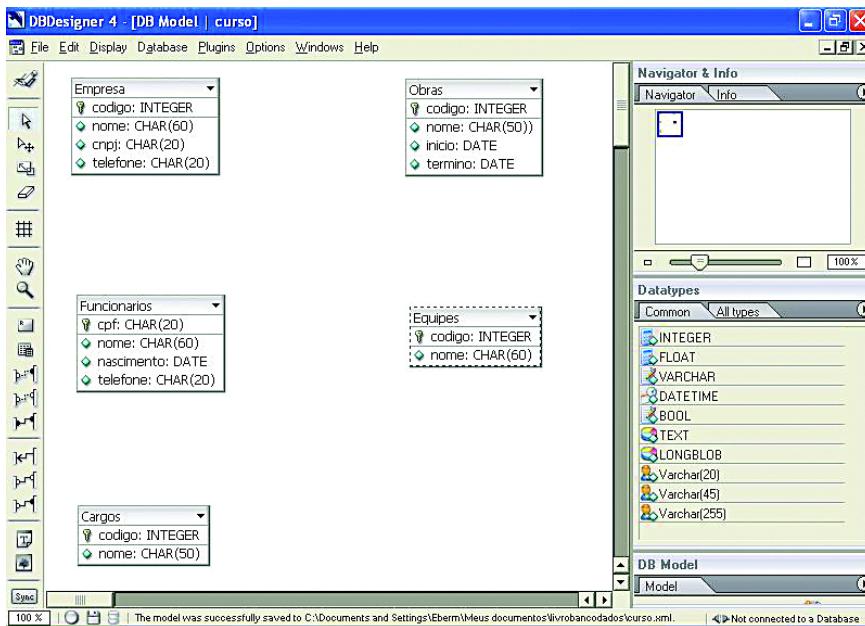
Finalmente, a **Figura 8** fornece a representação da entidade equipe que é responsável por agrupar um conjunto de funcionários de uma determinada especialidade. Assim como a entidade obras, uma equipe contém apenas um nome e um código que a identifica. Vale ressaltar que não há nenhuma forma de identificar uma entidade fraca quando se utiliza uma ferramenta de modelagem.

Figura 8:
Definição da
tabela Equipes



Este ponto marca o fim da definição das entidades ou tabelas do sistema. Assim, a situação em que se encontra o modelo é exemplificada pela **Figura 9**.

Figura 9:
Ilustração das
5 tabelas do
sistema para
construção
civil



Assim, tendo construído todas as tabelas do sistema resta ainda definir os relacionamentos entre os objetos, bem como as restrições e cardinalidades pertinentes ao problema. Esta tarefa é descrita com detalhes na próxima seção.

3.3. DEFININDO OS RELACIONAMENTOS ENTRE AS TABELAS

O objetivo desta seção é elaborar os relacionamentos existentes entre as entidades do sistema. Existem basicamente cinco relacionamentos, que são:

- 1- Funcionário gerencia equipe
- 2- Equipe é composta de funcionários
- 3- Funcionários possuem cargos
- 4- Equipes executam obras
- 5- Empresa constrói obras

Cada um destes relacionamentos apresenta a cardinalidade descrita no modelo da **Figura 1**. Os elementos do DBDesigner4 relativos às relações foram expostos no capítulo anterior e não serão vistos novamente.

O primeiro relacionamento que será definido é o funcionário gerencia equipe. Este possui cardinalidade de um para um, e a ferramenta criará um atributo em uma das tabelas de forma a identificar a interação entre eles. Portanto, é razoável que seja colocada na tabela de equipes o código do funcionário que a gerencia. Para que isto se verifique, deve-se definir a tabela de equipes como o término da relação, isto é, clicar no elemento 1:1, depois na tabela funcionários e então na tabela de equipes. Além disto, deve-se garantir que caso haja remoção ou alteração de um funcionário, se o mesmo fizer parte de alguma equipe, esta referência deverá ser excluída também. Ou seja, nos campos “On delete” e “On update” utiliza-se a restrição “CASCADE”, que realiza exatamente esta tarefa. A **Figura 10** ilustra a situação descrita.

Para o caso de relacionamentos do tipo um para muitos, a chave primária da entidade do lado um será colocada na tabela com várias ocorrências (N). Com isto é possível identificar a interdependência entre as entidades. Neste caso, aciona o elemento 1:N, clica na entidade 1 e depois na entidade N para construir o relacionamento corretamente. As **Figuras 11, 12, e 13**, representam respectivamente os relacionamentos Equipes são compostas por Funcionários, Funcionários possuem Cargos e Empresa constrói Obras.

Figura 10:
Relacionamento
Funcionário
gerencia Equipe

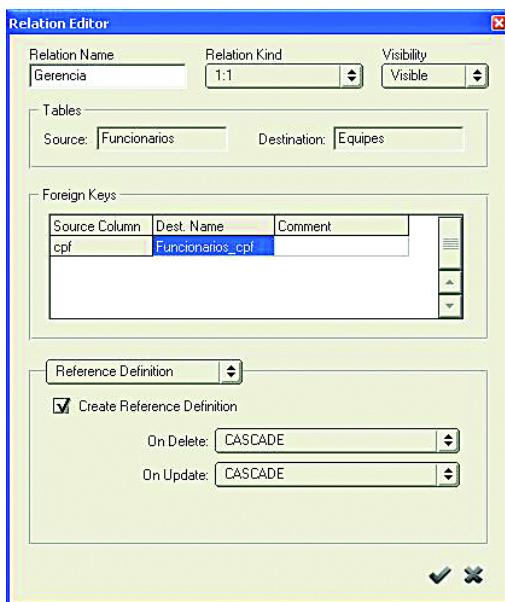


Figura 11:
Relacionamento Equipes
são compostas por
Funcionários

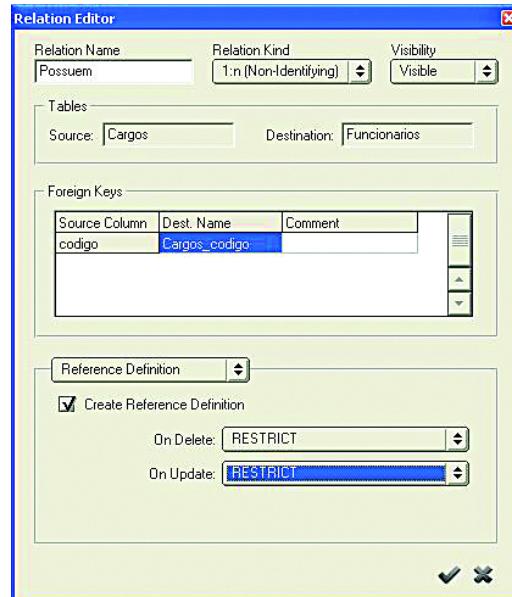


Figura 12:
Relacionamento
Funcionários
possuem
Cargos

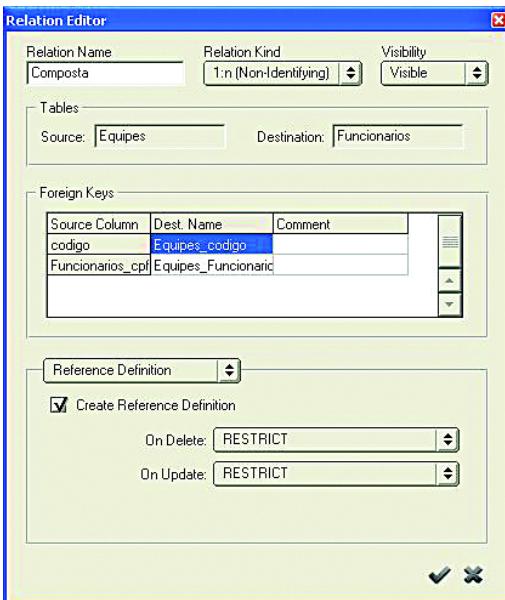
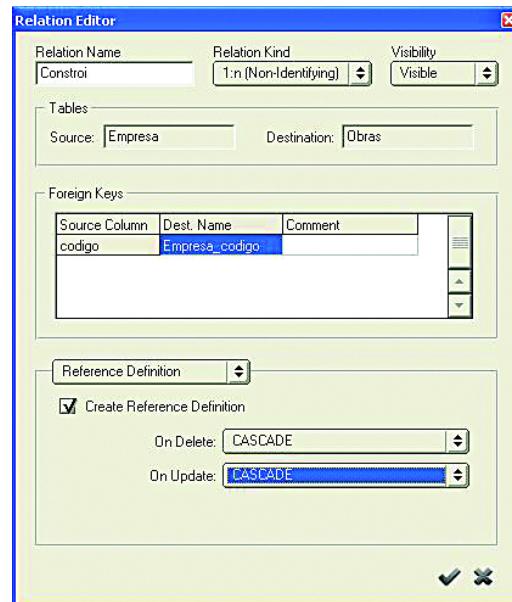


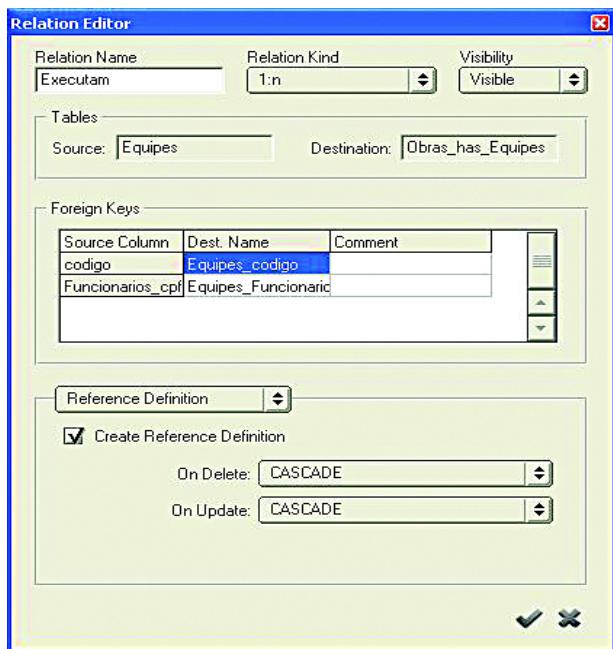
Figura 13:
Relacionamento
Empresa
constrói Obras



Vale ressaltar que para todos os relacionamentos foi aplicada a opção “RESTRICT”. Este recurso garante que nenhuma alteração ou exclusão poderá ser realizada em registros que apresentem correspondência na tabela que está no lado N da relação. Já o relacionamento ilustrado na **Figura 13**, a restrição utilizada foi o “CASCADE”, propagando assim toda alteração/exclusão ocorrida. Isto significa dizer que se remover ou alterar uma entidade no lado 1 as referências existentes no lado N serão automaticamente alteradas/excluídas.

O último relacionamento é: equipes executam obras. Neste caso o tipo do relacionamento é de muitos para muitos, ou seja, a obra é executada por várias equipes, e por consequência, uma equipe participa de várias obras. Ao acionar o elemento N:M da ferramenta de modelagem e clicar nas duas tabelas, será criada uma nova tabela que conterá as chaves primárias de ambas as entidades. A **Figura 14** ilustra esta situação.

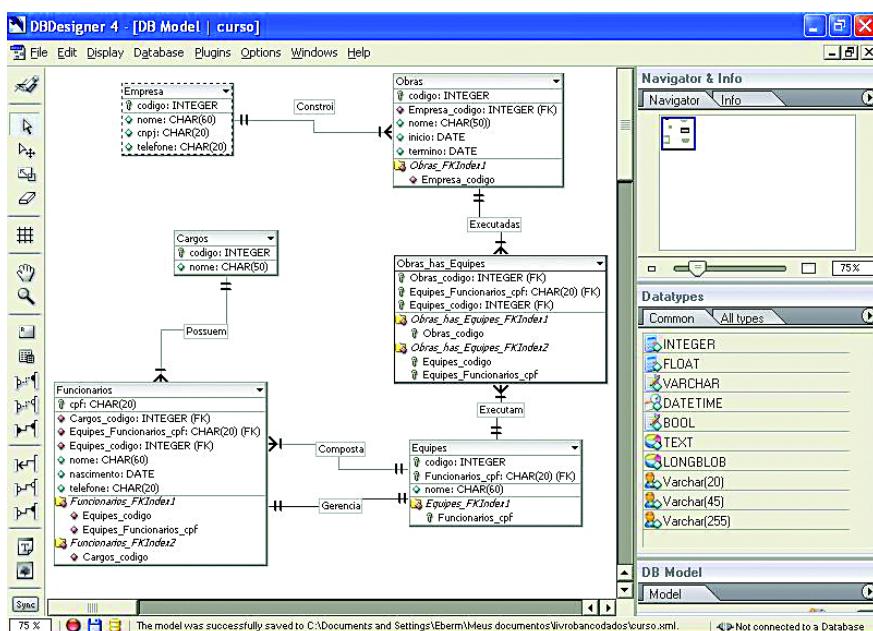
Figura 14:
Relacionamento
Equipes
executam
Obras e Obras
são
executadas por
Equipes



Percebe-se na realidade que um relacionamento de muitos para muitos é convertido automaticamente para dois relacionamentos do tipo um para muitos. Isto é feito a partir da introdução de uma nova tabela, que no exemplo foi chamada de *Obras_has_Equipes*.

Com isto, está concluído o modelo do banco de dados da construção civil e o diagrama gerado é apresentado na **Figura 15**.

Figura 15:
Diagrama
completo da
aplicação



3.4. CONVERTENDO O MODELO PARA O BANCO DE DADOS CURSO

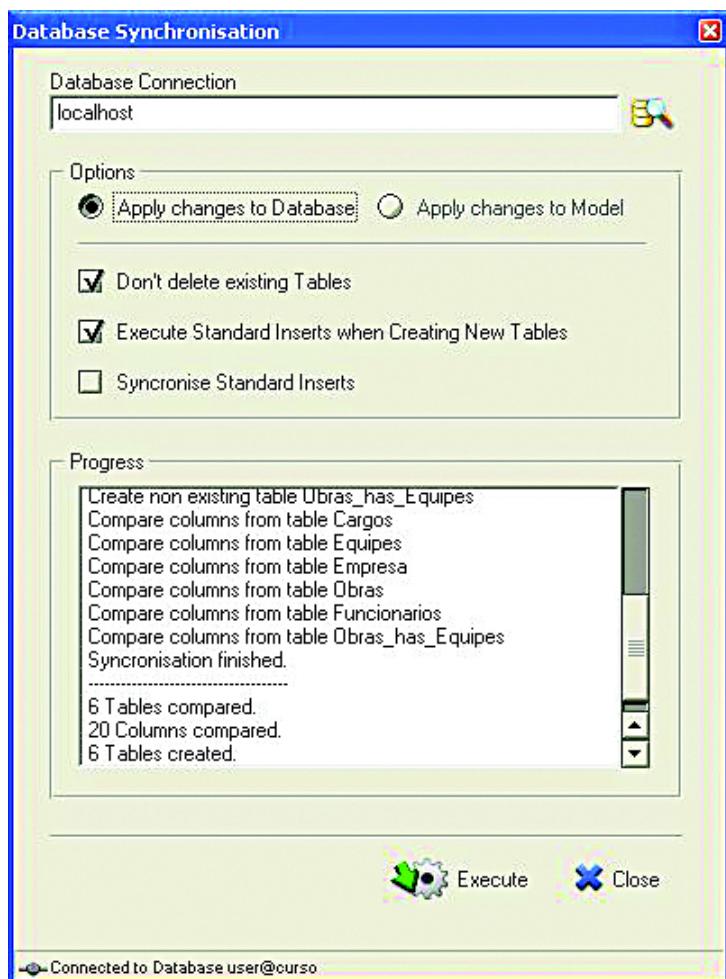


Figura 16:
Sincronização
do modelo com
o banco curso

civil, a partir da utilização de uma ferramenta de modelagem de dados. Com isto, finaliza-se a discussão sobre os aspectos do desenho da aplicação, possibilitando a extensão destes conhecimentos para desenvolver novos sistemas.

Os capítulos posteriores abordarão questões mais ligadas à utilização dos dados armazenados no SGBD, isto é, técnicas de consultas e extração de dados destes bancos de dados projetados até aqui.

5. REFERÊNCIAS BIBLIOGRÁFICAS

- MySQL AB: MySQL 5.0 Reference Manual. Disponível em: <<http://www.mysql.com/documentation>>. Acesso em: 20 dez. 2005.
- FabForce.net: DBDesigner4 online documentation. Disponível em: <http://www.fabforce.net/dbdesigner4/doc/index.html>. Acesso em: 20 dez. 2005.

Uma vez concluída a modelagem do sistema é possível utilizar a funcionalidade de sincronização de dados do DBDesigner4 para criar as tabelas dentro do MySQL. O procedimento para realizar esta tarefa foi apresentado com detalhes no capítulo anterior, portanto não será exposto novamente. A **Figura 16** ilustra a execução da rotina de sincronização de dados.

Desta forma, está disponível no banco curso gerenciado pelo MySQL, todas as tabelas e atributos descritos no modelo lógico. Vale lembrar que toda alteração feita no modelo poderá ser refletida no banco curso através desta prática. Isto seria necessário no caso de outros usuários do banco terem realizados alterações no modelo ou nas tabelas e desejarem obter a versão mais atual do mesmo.

4. CONCLUSÕES

Neste capítulo foi elaborado um modelo lógico do sistema de construção

CAPÍTULO 5

1. INTRODUÇÃO

A utilização de um Sistema Gerenciador de Banco de Dados (SGBD) tem como objetivo principal facilitar o armazenamento e a manipulação das informações armazenadas por ele. Para isto inicialmente é preciso projetar o banco de dados e documentá-lo, conforme descrito nos capítulos anteriores, onde foram expostos os detalhes da modelagem Entidade-Relacionamento.

Uma vez elaborado o projeto lógico do banco de dados e tendo sido realizado o seu mapeamento para um SGBD qualquer, torna-se possível a sua utilização para o armazenamento das informações pertinentes ao sistema em questão. Neste caso, não basta apenas criar o repositório de dados, é preciso também que este SGBD possua recursos que permitam o acesso a estas informações.

Basicamente, um SGBD provê um mecanismo de consultas capaz de criar uma interface simples para adicionar, alterar, remover e extrair informações destas grandes bases de dados. Tudo isto é feito a partir da utilização de uma linguagem de consulta padrão conhecida como SQL (Structured Query Language), que significa em português, Linguagem de Consulta Estruturada. Esta linguagem foi introduzida de forma rápida no capítulo 1, e será abordada com detalhe nos próximos capítulos. Desta forma, deseja-se possibilitar a sua utilização em uma situação real, isto é, para a manipulação do banco de dados curso, projetado anteriormente.

A linguagem SQL é dividida em duas partes: Linguagem para Definição de Dados (DDL – Data Definition Language), e Linguagem para Manipulação de Dados (DML – Data Manipulation Language). Na DDL encontram-se comandos para a criação e alteração de dados, tais como bancos de dados e tabelas, permitindo, por exemplo, a definição de uma nova tabela ou até mesmo a inclusão de uma coluna em uma tabela já existente. Já a DML apresenta os comandos para a inserção, alteração, exclusão e leitura dos dados contidos nestas tabelas.

Nos próximos capítulos serão discutidos os detalhes da linguagem SQL, mas para isto, neste capítulo será introduzida uma ferramenta gráfica para a elaboração destas consultas. O sistema que será utilizado neste livro será o MySQL Query Browser que pode ser obtido gratuitamente a partir do site <http://www.mysql.com/downloads>. Este produto possui suporte para os Sistemas Operacionais (SO) Linux e Windows e servirá como ambiente de suporte ao desenvolvimento de consultas SQL no decorrer deste curso.

Portanto, o objetivo deste capítulo é apresentar as principais características desta ferramenta de forma a possibilitar o seu entendimento e utilização para a consulta ao banco de dados.

2. INTRODUÇÃO AO MySQL QUERY BROWSER

O MySQL Query Browser, assim como o DBDesigner4, é conhecido como um cliente MySQL, já que permite a comunicação entre o usuário do SGBD, que pode ser uma pessoa ou sistema que utilize o MySQL como repositório de dados, e o próprio SGBD. Neste caso, o MySQL Query Browser fornece uma interface amigável para a visualização e manipulação dos bancos de dados mantidos pelo MySQL.

Para utilizar esta ferramenta, inicialmente deve-se estabelecer uma conexão com o SGBD MySQL, assim como ocorre no DBDesigner4 e no cliente mysql utilizado a partir de um terminal Linux. Este processo foi empregado para a criação do banco de dados curso, no capítulo anterior.

O MySQL Query Browser apresenta diversas características importantes, dentre as quais se destacam o acesso a todos os bancos de dados ou esquemas armazenados pelo MySQL. Além disto, possui o manual do MySQL, onde pode-se consultar a sintaxe de comandos e funções pré-definidas, além de um editor visual de consultas que permite a criação de consultas SQL de forma intuitiva, isto é, através da seleção de atributos e tabelas exibidos pela ferramenta. Neste caso, não é necessário escrever o comando SQL, a ferramenta o gera automaticamente.

As subseções seguintes apresentam uma visão geral de como se conectar ao SGBD a partir do MySQL Query Browser, bem como exibir o banco de dados curso criado anteriormente, e finalmente apresentar as principais funcionalidades desta ferramenta.

2.1. ABRINDO UMA CONEXÃO COM O BANCO DE DADOS CURSO

Para iniciar o uso do MySQL Query Browser, basta abrir um terminal no Linux e invocar o programa digitando o seu nome no prompt de comandos. Neste ponto deve-se garantir que o mesmo está instalado e configurado para que esteja disponível no PATH do Linux. Assim, ao digitar “mysql-query-browser” no terminal do Linux, a tela exibida na **Figura 1** aparecerá imediatamente.

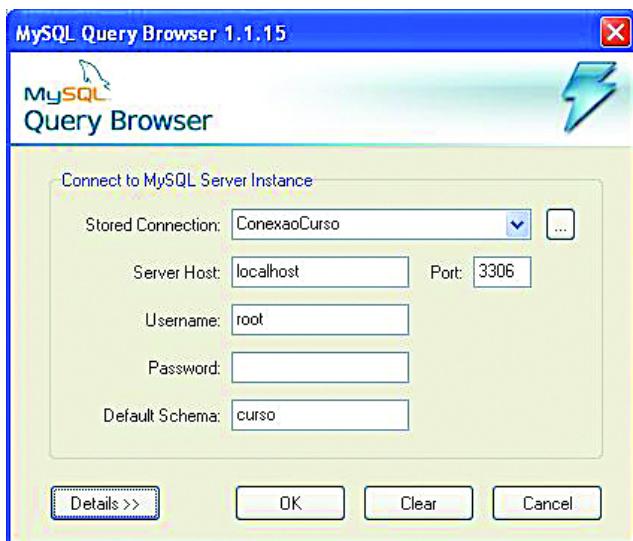


Figura 1: Tela para conexão do MySQL Query Browser com o MySQL

No campo Stored Connection (Conexões armazenadas), será informado um nome para a conexão que poderá ser utilizado para os futuros acessos ao sistema, sem ter que informar todos os parâmetros novamente. Além disto, deve-se informar o endereço da máquina onde se encontra o MySQL (Server Host, que quer dizer Endereço do servidor). Neste caso, o servidor MySQL e o MySQL Query Browser estão instalados na mesma máquina, por isto utiliza-se localhost para identificar o endereço do servidor MySQL. Outro parâmetro para a conexão é a porta que o MySQL utiliza, por padrão é a 3306, conforme indicado na **Figura 1**.

Toda conexão ao SGBD requer um usuário Username (nome do usuário), que neste caso será utilizado o root, da mesma forma que as situa-

ções anteriores, e a senha digitada no campo Password (Senha). No caso o usuário root não possui senha, portanto este campo deverá ser mantido vazio. Finalmente, deve-se escolher qual o “Default Schema”, ou seja, o Esquema de banco de dados padrão. No contexto de um SGBD um esquema se refere à todos os objetos que compõe a base de dados do sistema. Neste caso, seriam o banco de dados e suas tabelas, procedimentos, visões, dentre outros. No caso do MySQL, um esquema é compreendido como um banco de dados. Neste caso será informado o nome do banco de dados que se deseja trabalhar durante a conexão com o SGBD. Vale ressaltar que este banco de dados pode ser alterado

posteriormente caso seja necessário. Para efeito de ilustrar a ferramenta, será utilizado o banco de dados curso construído no capítulo 4.

Definidos os parâmetros basta acionar o botão Ok para estabelecer a conexão com o banco de dados desejado. Assim, a tela exibida na **Figura 2** aparecerá e é possível ver à direita o banco de dados curso selecionado, bem como as tabelas que o compõe.

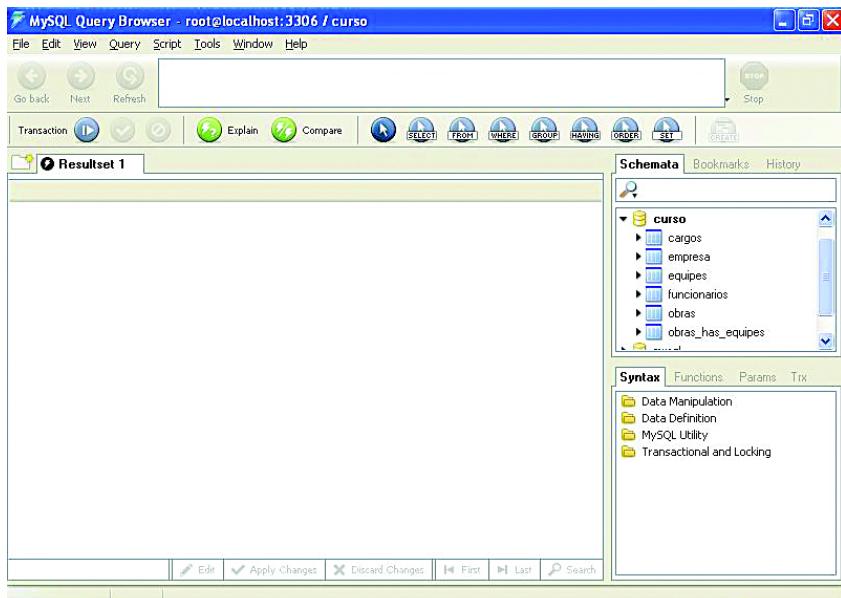


Figura 2:
Conexão com
o banco de
dados curso
através do
MySQL Query
Browser

Os principais elementos do sistema são o editor de consultas, localizado no topo da janela e o editor de banco de dados na lateral direita da tela, onde são exibidos todos os bancos de dados disponíveis no SGBD. Percebe-se a existência de apenas três bancos, e em destaque o banco de dados curso, selecionado durante a conexão com o SGBD. Além disto, a janela ResultSet (Conjunto Resultante), é o local onde os resultados das consultas serão exibidos pelo sistema.

Na seção 3 estes elementos do sistema serão apresentados com maior detalhe, salientando as suas utilizações em situações de acesso ao banco de dados curso.

3. EXPLORANDO OS PRINCIPAIS RECURSOS DO MySQL QUERY BROWSER

O objetivo desta seção é ilustrar o editor de banco de dados desta ferramenta exibindo a sua capacidade para a definição e alteração de tabelas e, o editor de consultas SQL. Este permite criar manualmente ou automaticamente as consultas a serem submetidas ao SGBD, facilitando a manipulação das informações armazenadas.

3.1. UTILIZANDO O EDITOR DE BANCO DE DADOS

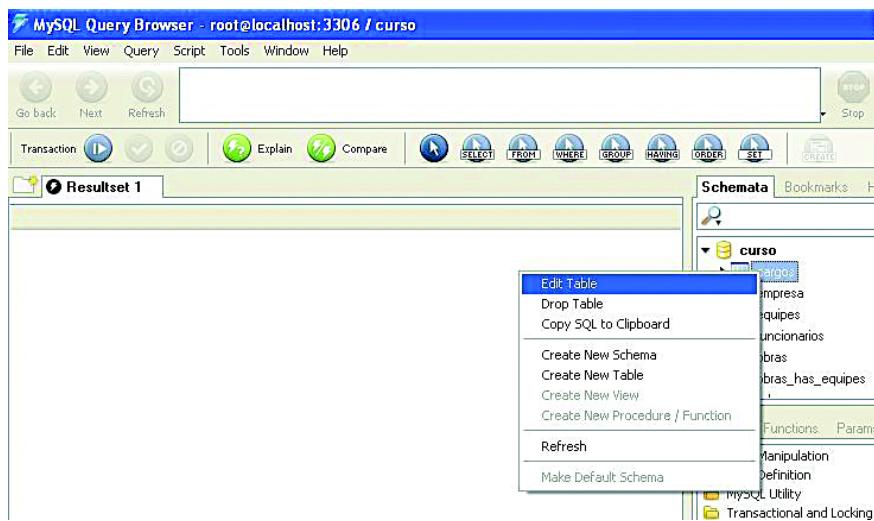
A **Figura 3** apresenta em destaque o editor de banco de dados do MySQL Query Browser, que fornece acesso aos diversos repositórios de dados gerenciados pelo MySQL.

Figura 3: Detalhe do editor de banco
de dados do MySQL Query Browser



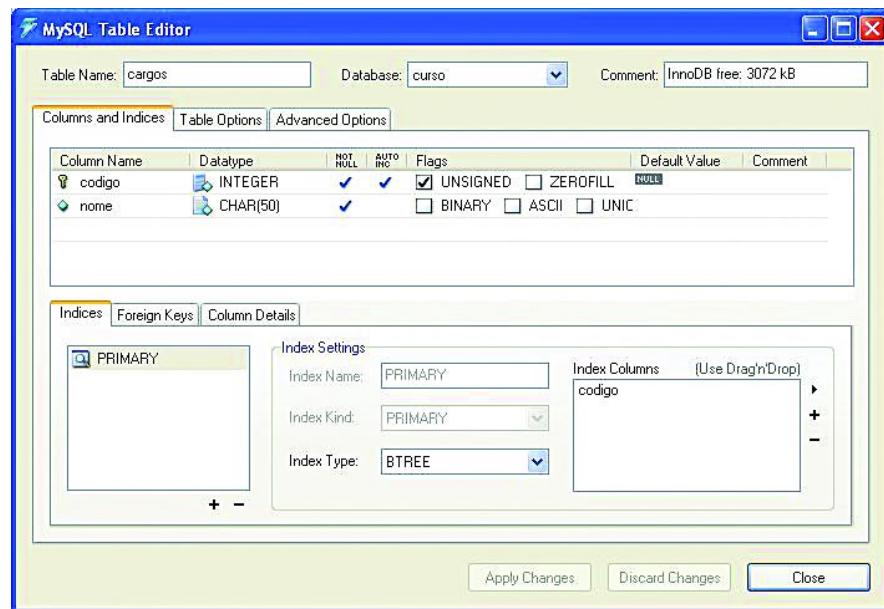
Observa-se na figura a existência de três bancos de dados, que são curso – que está selecionado – o mysql e o test que vêm criados por padrão no MySQL. Este editor provê, além da capacidade de visualizar as tabelas de um banco de dados, as funções de criar e excluir um esquema ou banco de dados, ou criar, excluir e alterar uma tabela, bem como copiar o comando SQL para a criação da tabela. Assim como feito através do DBDesigner4, é possível criar e alterar a estrutura de uma determinada tabela para isto basta açãoar o botão da direita do mouse sobre a tabela desejada e selecionar Edit table (Editar tabela), conforme indica a **Figura 4**.

Figura 4:
Acionando o editor de tabela do MySQL Query Browser



A **Figura 5** ilustra a edição da tabela de cargos a partir do editor de tabelas do MySQL Query Browser.

Figura 5: Editor de tabelas do MySQL Query Browser



Desta forma é possível criar e alterar toda a estrutura de dados do banco de dados de forma simples, e se necessário gerar os comandos SQL, isto é DDL, relativos à tarefa. Isto facilita sobremaneira a criação dos dados, caso não haja uma ferramenta de modelagem com recursos de sincronização de dados, conforme o DBDesigner4 provê.

3.2. CONHECENDO O EDITOR DE CONSULTAS SQL DO SISTEMA

O MySQL Query Browser apresenta mecanismos sofisticados para a elaboração de consultas tanto para a escrita quanto para a leitura de dados. O grande benefício desta ferramenta é a possibilidade de manipulação das informações sem que se tenham profundos conhecimentos da linguagem de consulta SQL. Nas subseções seguintes serão explorados os recursos para a elaboração de consultas disponíveis dentro desta ferramenta.

3.2.1. INSERINDO, ALTERANDO E EXCLUINDO DADOS EM UMA TABELA

Para ilustrar a inserção de dados em uma tabela a partir do MySQL Query Browser, considera-se a inclusão dos cargos de engenheiro, arquiteto e pedreiro, com os códigos 1, 2 e 3, respectivamente. O primeiro passo é ir até o editor de banco de dados e dar um duplo clique sobre a tabela em que se deseja inserir as informações, neste caso a tabela cargos. Neste momento, aparecerá no editor de resultados “ResultSet 1” uma tabela vazia com duas colunas código e nome, que são os atributos da tabela de cargos. Além disto, no editor de consultas no alto da tela, aparece a consulta SQL que lista todas as colunas da tabela, gerada automaticamente pela ferramenta, que é “select * from cargos c”. A **Figura 6** ilustra esta situação.

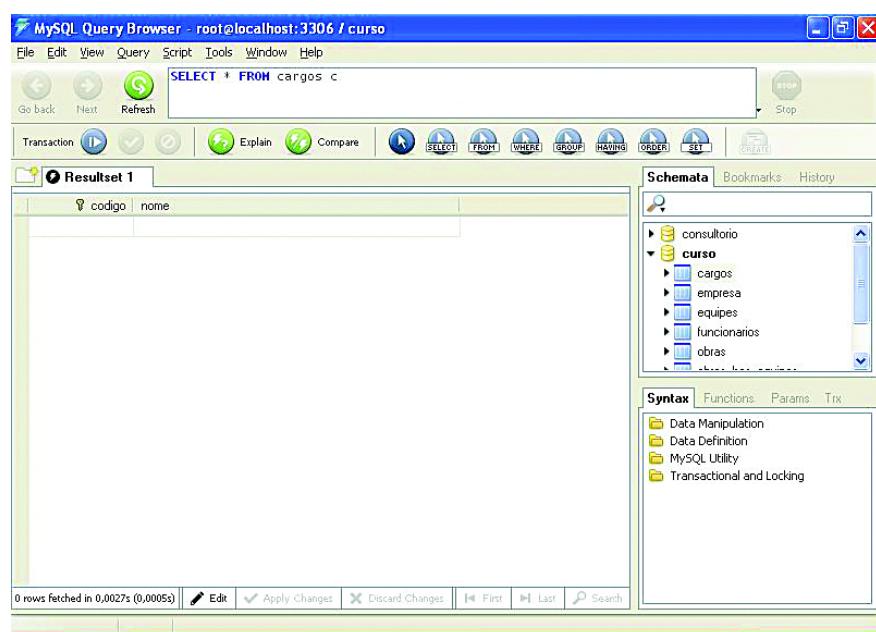
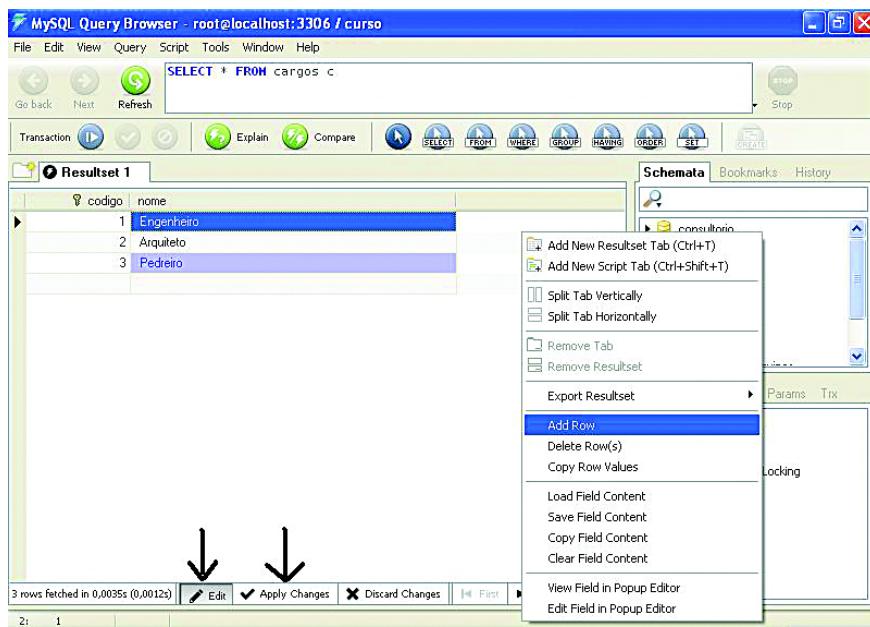


Figura 6:
Tela com a
tabela de
cargos sem
registros

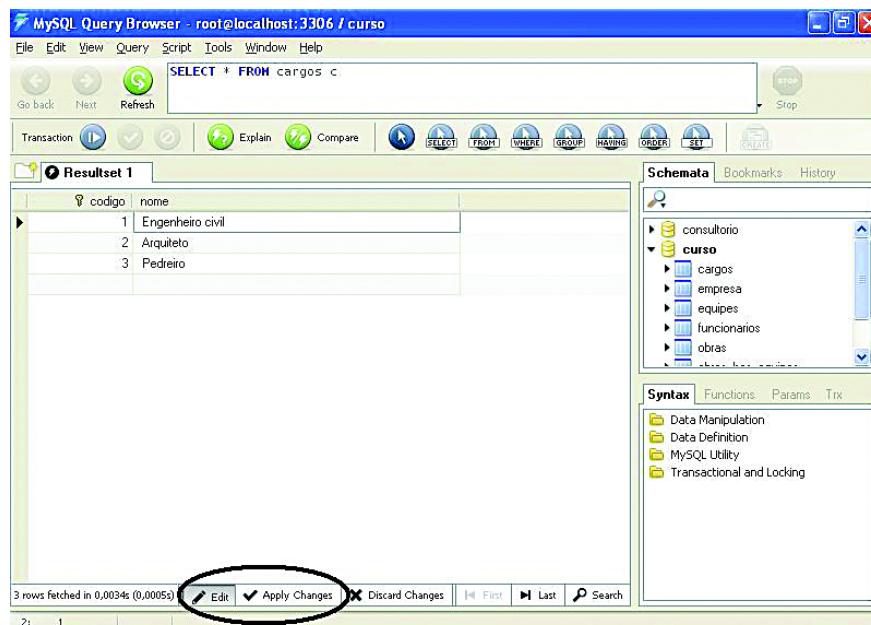
Para inserir os dados, clique no botão Edit (Editar), localizado na parte inferior da tabela de resultados e em seguida acione o botão direito do mouse sobre a caixa de resultados invocando o botão Add Row (Incluir linha). Neste momento o cursor estará disponível no campo código, que poderá ser preenchido. Depois de informado o código aperte a tecla <tab> e então digite o nome do cargo. Continue este procedimento até que todos os registros tenham sido inseridos e então aperte a tecla Apply Changes (Aplicar alterações), para que as modificações persistam. A **Figura 7** destaca os botões a serem utilizados nesta operação e o resultado final da inclusão dos três registros.

Figura 7:
Ilustração da inserção de três registros na tabela cargos



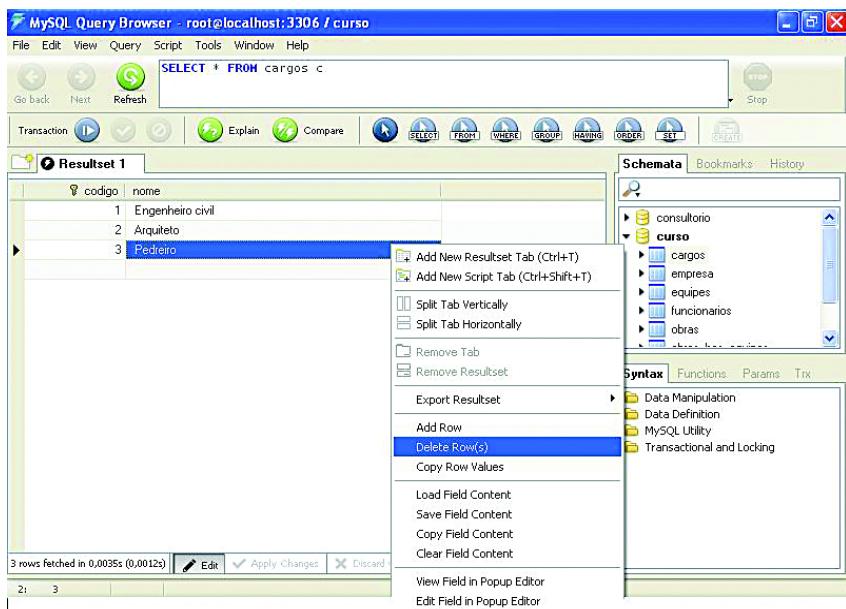
Supondo que o cargo de engenheiro desse ser modificado para engenheiro civil, de forma a retratar de forma mais específica os cargos dos funcionários, deve-se utilizar o recurso de alteração de dados para realizar esta ação. Para isto, basta clicar no botão Edit e então dar um duplo clique sobre o nome do cargo que se deseja modificar. Feito isto, digite o novo nome do cargo e então acione o botão Apply Changes para confirmar a alteração da informação. A **Figura 8** ilustra este procedimento e o registro modificado.

Figura 8:
Alteração do cargo de engenheiro para engenheiro civil



Além da possibilidade de inserir e alterar um registro de uma tabela é preciso que a ferramenta forneça recursos para a exclusão de registros ou linhas da tabela. Assim como os demais procedimentos descritos anteriormente, acione o botão Edit, selecione o registro a ser excluído e então acionando o botão direito do mouse escolha a opção Delete Row(s) (Remover registro(s)), conforme ilustra a **Figura 9**.

Figura 9:
Exclusão do
cargo pedreiro



Finalmente para confirmar a operação ação o botão Apply Changes, localizado na parte inferior da tela de resultados, e assim o registro será eliminado da tabela definitivamente.

Desta forma, conclui-se a execução das principais tarefas para a manutenção de informações contidas em um banco de dados, a partir da interface fornecida pelo MySQL Query Browser. Vale ressaltar que este procedimento é relativamente simples e intuitivo, já que não exige conhecimentos avançados sobre o SQL para realizar as operações. Este é um dos benefícios desta ferramenta gráfica.

3.2.2. LEITURA DE INFORMAÇÕES ARMAZENADAS EM UMA TABELA

Um banco de dados é um repositório de dados cujas informações estão acessíveis para os seus usuários. Portanto, é preciso realizar a leitura das informações armazenadas neste conjunto de tabelas, e o MySQL Query Browser fornece recursos para executar esta tarefa.

A leitura de todas as informações de uma tabela pode ser feita a partir de um duplo clique sobre o nome da tabela. Neste caso são mostrados todos os registros e colunas da tabela, no editor de resultados, conforme visto na subseção anterior. Porém, em situações reais torna-se necessário especificar critérios ou filtros para a seleção de dados.

A filtragem de dados consiste em escolher um subconjunto de dados dentro de um conjunto maior de informações. Por exemplo, uma pessoa pode optar por utilizar um determinado ônibus dentre as várias linhas existentes na sua cidade. Isto é a filtragem de dados. A escolha de qual ônibus utilizar é baseada no destino ao qual se pretende chegar, ou seja, dentre as diversas linhas existentes selecionam-se apenas aquelas que levam até ao local desejado. Neste caso, o destino da viagem representa o critério de filtragem.

Além de filtrar os registros de uma tabela pode-se ainda restringir as colunas que serão retornadas pela consulta. Por exemplo, deseja-se exibir apenas o nome dos funcionários, em vez de todas as colunas. Neste caso, o MySQL Query Browser oferece um conjunto de ferramentas para a elaboração de consultas mais complexas. Estes componentes são ilustrados na **Figura 10**.

Figura 10:
Componentes para a construção de consultas para leitura de dados



Para construir uma consulta, acione o botão SELECT e então clique sobre as colunas que devem ser exibidas. Considerando que se deseja listar o nome de todos os cargos cadastrados, inicia-se o processo acionando o botão SELECT, e então clicando na coluna nome da tabela cargos. Feito isto aparecerá o comando SQL na parte superior da janela, então basta apertar as teclas <ctrl> e <enter> simultaneamente e o resultado será exibido na janela de resultados, conforme ilustra a **Figura 11**. Vale chamar atenção para o fato de que o comando “select c.nome from cargos c” foi gerado automaticamente pela ferramenta, possibilitando listar a penas o nome dos cargos.

Figura 11:
Listagem dos nomes de todos os cargos

Caso seja necessário informar um critério de seleção de dados, basta acionar o botão WHERE e então selecionar a coluna que será utilizada como critério de filtragem. Supondo a exibição apenas do cargo com o código 1, clique em WHERE, depois na coluna código, e então digite no editor de consultas o critério “=1”. Assim, o comando “select c.nome from cargos c where c.codigo=1” será gerado, digite as teclas <ctrl> e <enter> e o resultado ilustrado pela **Figura 12** será exibido.

Figura 12:
Seleção do nome do cargo com o código 1

Assim, é possível fazer a leitura dos dados armazenados no banco de dados a partir da utilização de recursos gráficos e automáticos de navegação, providos pela ferramenta apresentada neste capítulo.

4. CONCLUSÕES

Um banco de dados tem a função de armazenar informações e possibilitar o acesso às mesmas. Para isto, deve-se utilizar uma linguagem de consulta para realizar esta tarefa. O objetivo deste capítulo foi apresentar uma ferramenta de consulta que permita a interação com o banco de dados de forma automática e sem que seja necessário o conhecimento profundo desta linguagem de consulta.

Além disto, este capítulo forneceu uma visão geral dos recursos para a manipulação das informações de um banco de dados, gerando o conhecimento básico para o entendimento dos conceitos da linguagem SQL que será abordada com detalhe nos próximos capítulos.

5. EXERCÍCIOS DE FIXAÇÃO

- 1- Onde pode ser encontrada a ferramenta MySQL Query Browser?
- 2- Qual o objetivo desta ferramenta?
- 3- Utilizando o editor de tabelas do MySQL Query Browser, crie uma tabela para armazenar o nome, a data de nascimento, o endereço e cidade de todos os seus amigos.
- 4- Adicione à tabela criada no exercício anterior, a coluna para armazenar o CEP dos seus amigos. Utilize o editor de tabelas disponível na ferramenta.

6. REFERÊNCIAS BIBLIOGRÁFICAS

- MySQL AB: MySQL Query Browser. Disponível em: <<http://dev.mysql.com/doc/query-browser/en/index.html>>. Acesso em: 20 dez. 2005.
- MySQL AB: MySQL 5.0 Reference Manual. Disponível em: <<http://www.mysql.com/documentation>>. Acesso em: 20 dez. 2005.

CAPÍTULO 6

1. INTRODUÇÃO

No capítulo 5 foram introduzidos os primeiros passos para a utilização dos mecanismos de consultas disponíveis em um sistema de banco de dados relacional. Esta interação foi realizada a partir de uma ferramenta de consulta que provê uma interface de manipulação de dados que omite a linguagem nativa de comunicação com o SGBD (Sistema Gerenciador de Banco de Dados). Na verdade, toda a navegação é feita de forma visual e automática, o que facilita a utilização do banco de dados mesmo para usuários com pouca experiência no assunto.

De um modo geral, existem situações em que pode ser necessária a interação com o sistema de banco de dados sem a utilização de uma ferramenta como o MySQL Query Browser, e neste caso, deve-se empregar a linguagem de consulta reconhecida pelo sistema. Existe um comitê chamado ANSI, que cuida da padronização de uma linguagem de consulta universal, que permita a comunicação com todos os SGBD que seguem estas normas. Esta linguagem é conhecida como SQL (Structured Query Language), que significa em português, Linguagem de Consulta Estruturada.

Uma língua significa um conjunto de símbolos e expressões que representam uma forma de comunicação entre povos e nações. Da mesma forma, a linguagem SQL é constituída de um conjunto de símbolos que expressam a linguagem ou dialeto de comunicação com os mais variados sistemas de bancos de dados relacionais.

Todas as tarefas realizadas no capítulo anterior através do MySQL Query Browser, podem ser traduzidas em comandos SQL possíveis de serem executados em qualquer banco de dados que suporte este padrão. Para facilitar o seu entendimento costuma-se dividir a linguagem SQL em duas partes. A primeira parte conhecida como DDL (Data Definition Language), isto é Linguagem para Definição de Dados, contém os comandos para a criação de bancos de dados, tabelas, bem como a alteração e exclusão dos mesmos. A segunda parte é a DML (Data Manipulation Language), ou Linguagem para Manipulação de Dados, que permite a inserção, alteração, exclusão e leitura das informações armazenadas no banco de dados.

O objetivo deste capítulo e dos próximos é fornecer uma visão mais ampla destes comandos, permitindo o entendimento e a sua utilização em situações reais. Neste capítulo serão discutidos os aspectos relacionados à DDL, enquanto os capítulos posteriores tratam da DML. Portanto, toda a definição de dados feita utilizando o DBDesigner4 e o MySQL Query Browser será apresentada dentro do contexto da linguagem SQL.

2. INTRODUÇÃO AOS TIPOS DE DADOS

O processo de modelagem de um banco de dados consiste em representar situações do mundo real através de um sistema. É razoável imaginar que diversas informações podem ser armazenadas em um banco de dados, tais como endereços, imagens, datas, números, moedas, dentre outras. Conforme descrito anteriormente, todo atributo de uma tabela necessita ter um domínio ou conjunto de valores aceitáveis. Por exemplo, ao definir a data de nascimento de uma pessoa não é desejável que o sistema permita a inserção de um texto neste campo. Em uma coluna que armazena os salários dos funcionários devem-se ter apenas valores numéricos e não datas e horas, por exemplo.

Portanto, toda definição de tabelas passa pela construção dos seus atributos com seus respectivos tipos de dados ou domínios. Como o MySQL é o sistema de banco de dados adotado neste livro, a **Tabela 1** apresenta um subconjunto dos possíveis tipos de dados a serem empregados, bem como os tipos de informações que podem ser armazenados nestas colunas.

TIPO DE DADO	FORMATO	DESCRIÇÃO
INTEGER	NÚMEROS	NÚMEROS INTEIROS
DECIMAL(T,D)	T = TAMAÑO D = NÚMERO	NÚMEROS COM PRECISÃO FIXA TAIS COMO MOEDA., LEMBRANDO QUE O PONTO É O SEPARADOR DA DE CASAS DECIMAS PARTE FRACIONÁRIA OU CENTAVOS
FLOAT	X.XXXXXXXXXX	NÚMEROS REAIS COM QUANTIDADE DE CASAS DECIMAIS ELEVADA.
DATE	AAAA-MM-DD	ARMAZENA A DATA NO FORMATO ANO-MÊS-DIA
DATETIME	AAAA-MM-DD HH:MM:SS	ARMAZENA DATA E HORA EM UM ÚNICO CAMPO NO FORMATA ANO- MÊS-DIA HORA:MINUTO:SEGUNDOS
CHAR(T)	TEXTOS	INFORMAÇÕES TEXTUAIS COM NO MÁXIMO T CARACTERES
TEXT	TEXTOS LONGOS	INFORMAÇÕES TEXTUAIS LONGAS, TAIS COMO CURRÍCULOS, POEMAS, ETC
BLOB	BINÁRIOS VÍDEOS E IMAGENS	ARMAZENAM DADOS NO FORMATO BINÁRIO, TAIS COMO MÚSICAS,

Tabela 1: Principais tipos de dados do MySQL

Ao inserir uma linha ou registro em uma tabela deve-se informar um valor para cada coluna, respeitando o tipo de dados definido para aquele atributo. Existem situações em que o valor a ser colocado em uma coluna não é conhecido no momento da inclusão dos dados na tabela. Por exemplo, se existir uma coluna que armazena a data de falecimento dos funcionários, esta data em geral não é conhecida no momento da inclusão do registro.

Para contornar esta situação existe um valor especial conhecido como NULL que denota o fato de que a informação não é conhecida. No caso da data de falecimento, pode-se utilizar o valor NULL para os funcionários que ainda estão vivos. É possível definir no momento da criação da tabela se as colunas aceitam ou não este valor especial. Isto é feito através da cláusula NOT NULL, que deve ser colocada nas colunas que não aceitarão o NULL, lembrando que por padrão qualquer coluna aceita o NULL. É importante lembrar que colunas de qualquer tipo podem receber o valor NULL.

3. DESCRIÇÃO DOS COMANDOS PARA DEFINIÇÃO DE DADOS

O objetivo desta seção é apresentar a sintaxe dos comandos para a criação de bancos de dados e tabelas, expondo todos os comandos que constituem a linguagem SQL – DDL.

3.1. CRIANDO E REMOVENDO UM BANCO DE DADOS

Um banco de dados é entendido como uma coleção de tabelas. De fato é uma forma de se organizar as informações dentro do SGBD, isto é, cada aplicação pode ter o seu próprio banco de dados, onde estarão apenas as tabelas que fazem parte daquele problema. A **Listagem 1** fornece a sintaxe do comando para a criação do banco de dados.

```
CREATE DATABASE nome_do_banco
```

Listagem 1: Sintaxe do comando CREATE DATABASE

Por convenção, todos os comandos que se referem à linguagem SQL serão colocados em letras maiúsculas, enquanto os valores que serão informados pelos usuários serão exibidos em itálico. No exemplo anterior *nome_do_banco* deverá ser substituído pelo nome do banco que se deseja criar. Vale ressaltar que este comando pode ser submetido ao SGBD a partir de qualquer cliente SQL, como o MySQL Query Browser, ou o cliente mysql utilizado a partir de um terminal do Linux.

Para selecionar um banco de dados deve-se utilizar o comando USE, conforme ilustra a **Listagem 2**.

```
USE nome_do_banco
```

Listagem 2: Selecionando um banco de dados

Uma vez selecionado o banco e dados, pode-se manipular as tabelas contidas nele, ou até mesmo criar novas tabelas dentro deste banco de dados.

Para remover um banco de dados e todo o seu conteúdo, o comando DROP DATABASE deve ser utilizado. A **Listagem 3** fornece a sintaxe deste comando.

```
DROP DATABASE nome_do_banco
```

Listagem 3: Removendo um banco de dados e todo seu conteúdo

O comando anterior remove o banco de dados e todas as tabelas contidas nele, portanto deve-se agir com prudência a fim de que não sejam obtidos resultados indesejados.

Este é o primeiro passo para a construção de uma base de dados, lembrando que o banco de dados curso utilizado nos capítulos anteriores foi criado a partir da execução do comando exibido na **Listagem 1**. Para isto, deve-se apenas substituir o termo *nome_do_banco* por *curso*, que é o nome do banco desejado.

3.2. CRIANDO E REMOVENDO TABELAS

Para criar uma tabela é preciso primeiro identificar os atributos e seus tipos de dados, bem como as restrições em relação ao valor NULL, isto é, se haverão colunas com valores indefinidos. Além disto, é necessário identificar a chave primária da tabela, que é o conjunto de colunas que referenciam de forma única cada registro da tabela.

Finalmente, para as tabelas que participam de algum relacionamento é necessário determinar as chaves estrangeiras e as restrições que se aplicam sobre elas. Neste caso, o objetivo da chave estrangeira é identificar os registros que participam da relação e impor as regras de integridade que regem o relacionamento. Por exemplo, em um relacionamento entre funcionários e equipes, deve-se garantir que não haverá um membro da equipe que não esteja cadastrado na tabela de funcionários. Ou em um relacionamento entre pais e filhos, deve-se garantir que não haverá um filho sem um pai.

Para a criação de uma tabela utiliza-se o comando CREATE TABLE, cuja sintaxe básica está apresentada na **Listagem 4**.

```

CREATE TABLE nome_da_tabela (
    Coluna1 TIPO_COLUNA_1,
    Coluna 2 TIPO_COLUNA_1,
    ...
    Coluna N TIPO_COLUNA_N,
    PRIMARY KEY (colunas),
    [FOREIGN KEY (colunas) RESTRIÇÕES]
) [ENGINE=tipo];

```

Listagem 4: Sintaxe do comando CREATE TABLE

Como descrito na listagem anterior para criar uma tabela deve-se especificar o nome de cada coluna ou atributo que a constitui, seus tipos e a sua chave primária. Nota-se que a chave estrangeira é opcional, portanto aparece entre colchetes ([FOREIGN KEY]). No MySQL é possível escolher o tipo de tabela a ser criado (ENGINE), que para efeito deste livro será sempre utilizado o InnoDB, que possui suporte ao conceito de restrições de chaves estrangeiras. A explicação dos tipos de tabelas do MySQL ficam fora do escopo deste livro.

O TIPO_COLUNA deverá ser substituído por alguns dos tipos de dados suportados pelo SGBD utilizado. No caso do MySQL pode-se utilizar os tipos descritos na **Tabela 1** ou qualquer outro tipo de dado suportado por ele. Nas subseções seguintes serão apresentados exemplos da utilização deste comando.

3.2.1 ENTENDENDO O CONCEITO DE CHAVE PRIMÁRIA

A **Listagem 5** fornece os comandos para a criação das tabelas cargo, empresa e equipes, que seguem a sintaxe apresentada anteriormente.

```

CREATE TABLE cargos (
    codigo integer unsigned NOT NULL auto_increment,
    nome char(50) NOT NULL,
    PRIMARY KEY (codigo)
) ENGINE=InnoDB;

```

```

CREATE TABLE empresa (
    codigo integer unsigned NOT NULL auto_increment,
    nome char(60) NOT NULL,
    cnpj char(20) NOT NULL,
    telefone char(20) NOT NULL,
    PRIMARY KEY (codigo)
) ENGINE=InnoDB;

```

```

CREATE TABLE equipes (
    codigo integer unsigned NOT NULL auto_increment,

```

```

        nome char(60) NOT NULL,
        PRIMARY KEY (codigo)
    ) ENGINE=InnoDB;

```

Listagem 5: Comandos para a criação das tabelas cargo, empresa e equipes

No exemplo, todas as colunas foram criadas com a opção NOT NULL, isto é, não é possível informar o valor NULL para nenhuma das colunas. Todas as tabelas têm uma coluna código que é a chave primária da tabela. Isto significa dizer que não há nestas tabelas dois registros com o mesmo código, caso contrário não seria possível encontrar um determinado registro na base de dados devido à ambigüidade. Caso ocorra uma tentativa de inserir dois registros com o mesmo código numérico, o SGBD emitirá uma mensagem de chave duplicada (Duplicate key entry), e inibirá a inserção do mesmo. A **Figura 1** ilustra a situação onde é feita a tentativa de inserir o cargo de pedreiro utilizando o código idêntico ao do cargo de arquiteto. Percebe-se, em destaque na figura, a mensagem de erro emitida pelo SGBD ao tentar executar a inserção através do acionamento do botão **Apply Changes** (Aplicar alterações).

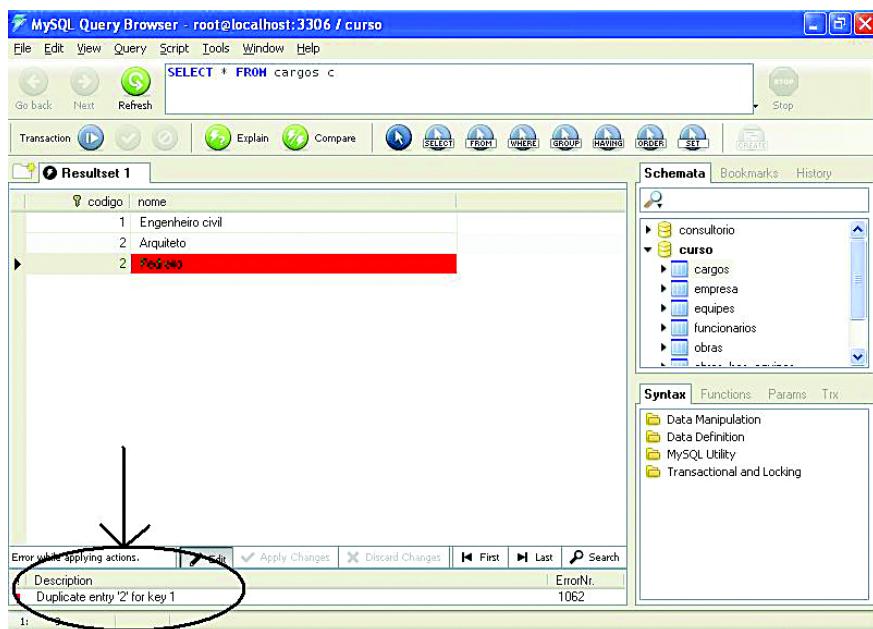


Figura 1: Erro na inserção de dois cargos com o código 2

Na definição dos códigos foi utilizado o atributo AUTO_INCREMENT, cuja função é gerar um número seqüencial automático. Isto significa dizer que durante a inserção, se o código for omitido o SGBD criará um código a partir do maior código cadastrado acrescido de um. Ou seja, se o maior código de cargos é o valor dois, a próxima inserção de cargos gerará o código três, e assim sucessivamente. Isto reduz a possibilidade de erros devido a chaves duplicadas, já que o código será gerado automaticamente pelo sistema e nunca se repetirá.

Para utilizar o recurso do AUTO_INCREMENT, a coluna deve ser declarada como do tipo inteiro e deverá ser a chave primária da tabela. Caso contrário, o sistema não permitirá a sua criação. Vale lembrar que no DBDesigner4 a criação da coluna auto-incremento é feita a partir da seleção do atributo "AI", presente no editor de atributos.

3.2.2 ENTENDENDO O CONCEITO DE CHAVE ESTRANGEIRA

As demais tabelas do banco curso apresentam relacionamentos entre elas, obedecendo às regras de integridade definidas pelo modelo lógico da aplicação. A **Listagem 6** contém os comandos para a criação das tabelas funcionários, obras e obras_tem_equipes.

```
CREATE TABLE funcionarios (
    cpf char(20) NOT NULL,
    Cargos_codigo int(10) unsigned NOT NULL,
    nome char(60) NOT NULL,
    nascimento date NOT NULL,
    telefone char(20) NOT NULL,
    PRIMARY KEY (cpf),
    FOREIGN KEY (Cargos_codigo) REFERENCES cargos (codigo)
) ENGINE=InnoDB;
```

```
CREATE TABLE obras (
    codigo int(10) unsigned NOT NULL auto_increment,
    Empresa_codigo int(10) unsigned NOT NULL,
    nome char(50) NOT NULL,
    inicio date NOT NULL,
    termino date NOT NULL,
    PRIMARY KEY (codigo),
    FOREIGN KEY (Empresa_codigo) REFERENCES empresa (codigo) ON DELETE
    CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;
```

```
CREATE TABLE obras_tem_equipes (
    Obras_codigo int(10) unsigned NOT NULL,
    Equipes_codigo int(10) unsigned NOT NULL,
    PRIMARY KEY (Obras_codigo,Equipes_codigo),
    FOREIGN KEY (Obras_codigo) REFERENCES obras (codigo) ON DELETE CASCADE
    ON UPDATE CASCADE,
    FOREIGN KEY (Equipes_codigo) REFERENCES equipes (codigo) ON DELETE
    CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB;
```

Listagem 6: Comandos para a criação das tabelas funcionário, obras e obras_tem_equipes

Percebe-se que nestas tabelas existe além da chave primária a definição de chaves estrangeiras, já que estas tabelas participam de relacionamentos. No caso da tabela de

funcionários, os registros desta tabela estão associados a registros existentes na tabela de cargos. Ou seja, todo funcionário tem um cargo, onde esta relação é salientada pela coluna Cargos_codigo da tabela de funcionários.

Uma chave estrangeira nada mais é que a chave primária de uma tabela colocada em outra tabela para identificar a relação entre elas. Para criá-la deve-se indicar qual o conjunto de colunas que a compõe, bem como a tabela e coluna que esta referencia. No exemplo têm-se FOREIGN KEY (Cargos_codigo) REFERENCES cargos (codigo). Isto significa dizer que os valores armazenados na coluna Cargos_codigo da tabela de funcionários, deve ser um valor contido na coluna código da tabela de cargos. Desta forma, o SGBD assegura que nenhum funcionário terá um cargo que não esteja cadastrado na tabela de cargos. A **Figura 2** ilustra uma tentativa de inserção de um funcionário com um cargo inválido.

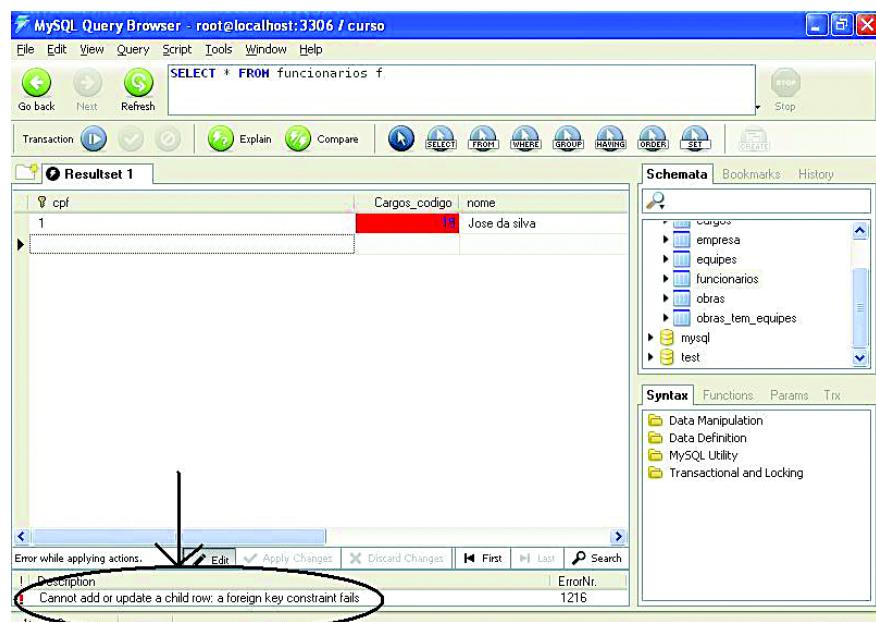


Figura 2: Erro na inserção de funcionário com cargo inexistente

Percebe-se em destaque na figura que o sistema emitiu uma mensagem de erro indicando que uma restrição de integridade foi violada (“Cannot add or update a child row: a foreign key constraint fails”, em português, “Não é possível adicionar ou alterar o registro: Falha de restrição de chave estrangeira”), e desta forma o comando não é executado.

Existe ainda a situação onde uma alteração ou exclusão de um cargo pode levar a uma inconsistência de dados na tabela de funcionários. No caso de uma remoção ou modificação de um cargo para o qual existam funcionários cadastrados, deve-se garantir que o funcionário não ficará com um cargo inválido.

Para isto, foram especificadas as restrições de chave estrangeira ON UPDATE CASCADE e ON DELETE RESTRICT. Isto é, quando o código de um cargo for alterado o SGBD propagará automaticamente a modificação para todos os funcionários que estejam cadastrados com este cargo (CASCADE). Já no caso da remoção, o sistema não permitirá a exclusão de cargos que apresentem funcionários associados a ele (RESTRICT). Vale ressaltar que a opção CASCADE ou RESTRICT pode ser aplicada às cláusulas UPDATE e DELETE de acordo com a restrição imposta pelo modelo.

Para ilustrar a utilização das cláusulas RESTRICT e CASCADE, considere um cadastro de funcionários e os seus dependentes ou filhos. Se o CPF do funcionário, que é a chave primária, for alterado é necessário alterar o CPF na tabela de filhos, pois do contrário a relação entre as duas entidades se perderia, já que o filho estaria associado com um CPF de um funcionário inexistente. Desta forma, pode-se inibir a alteração do CPF do pai durante um UPDATE, empregando a cláusula RESTRICT. Ou pode-se possibilitar a alteração automática do CPF em ambas as tabelas através do CASCADE.

O mesmo raciocínio se aplicaria no momento da exclusão de um funcionário, isto é no comando DELETE. Ao excluir um funcionário não se pode manter os registros e seus eventuais dependentes, pois desta forma teríamos registros órfãos na tabela de dependentes. Portanto, pode-se inibir a remoção dos funcionários que tenham filhos com o RESTRICT, ou forçar a exclusão dos filhos com a opção CASCADE. Todo este mecanismo visa a manutenção da consistência das informações, que já foi discutida nos capítulos anteriores.

Além disto, uma única tabela pode conter mais de uma chave estrangeira dependendo de como estão organizados os relacionamentos entre elas. Isto é o que ocorre no caso da tabela *obras_tem_equipes*, que se relaciona com as tabelas *obras* e *equipes*, simultaneamente. Por isto apresenta duas chaves estrangeiras, referenciando a chave primária de cada tabela.

O grande benefício das chaves estrangeiras é o fato de que o próprio SGBD assegura que as restrições de integridade pertinentes ao modelo serão aplicadas, mesmo que o usuário do banco desconheça estas regras.

3.2.3 REMOVENDO E ALTERANDO UMA TABELA

Finalmente, para remover uma tabela utiliza-se o comando `DROP TABLE` conforme ilustra a **Listagem 7**.

`DROP TABLE nome_da_tabela`

Listagem 7: Comando para a exclusão de uma tabela

Ao executar este comando a tabela será apagada por completo, juntamente com os dados que por ventura estejam armazenados nela. Por isto, deve-se ter cuidado na utilização do mesmo para evitar resultados indesejados.

Outra situação comum no dia a dia da utilização de um sistema de banco de dados é a necessidade de alterar a estrutura de uma tabela já existente. Por exemplo, supondo uma alteração no modelo lógico da aplicação de forma que se tenha que armazenar a data de entrada do funcionário na empresa, o que não era necessário anteriormente. Para contemplar esta situação seria necessária a inclusão de uma nova coluna do tipo data à tabela de funcionários, previamente criada.

Em outro cenário pode-se remover uma coluna, alterar o seu nome, ou até mesmo o tipo de dados que ela armazena. Imaginando que o nome do funcionário tenha sido definido inicialmente com tamanho máximo de 15 caracteres ou letras, e surgiu um novo funcionário com um nome extenso e que requer mais que 15 caracteres para ser armazenado. Neste caso, o tipo do dado deverá ser alterado para satisfazer esta nova condição.

O comando para modificar a estrutura de tabelas é o `ALTER TABLE`, que pode ser utilizado para os propósitos apresentados anteriormente. A **Listagem 8** fornece a sintaxe

para a utilização do comando ALTER TABLE.

```
ALTER TABLE nome_da_tabela ADD nome_da_coluna TIPO;
ALTER TABLE nome_da_tabela DROP nome_da_coluna;
ALTER TABLE nome_da_tabela MODIFY nome_da_coluna TIPO;
ALTER TABLE nome_da_tabela CHANGE coluna_antiga coluna_nova TIPO;
```

Tabela 2: Comandos para incluir, remover, trocar o tipo e alterar nome e tipo de colunas de uma tabela

Com isto, finaliza-se a descrição da parte de definição de dados da linguagem SQL, tornando possível o entendimento e aplicação destes mecanismos para a construção de bases de dados reais.

4. CONCLUSÕES

Neste capítulo foram explorados os conceitos associados à linguagem de definição de dados do SQL. Neste ponto alguns conceitos relacionados à integridade referencial e restrições de dados foram abordados com mais propriedade, tornando clara a sua importância para a resolução de problemas relacionados a aplicações reais.

Deste ponto em diante é possível empregar os recursos da linguagem SQL a fim de se criar bases de dados para suportar as aplicações que necessitem destes repositórios de dados. Tendo como base estes conhecimentos acerca da construção do banco de dados, os capítulos seguintes tratam da manipulação dos dados armazenados nestes bancos de informações. Portanto, serão apresentados os comandos para inserção, exclusão, alteração e leitura de dados, dentro da sintaxe de consulta universal que é o SQL.

5. EXERCÍCIOS DE FIXAÇÃO

- 1- Qual o significado do termo SQL?
- 2- Para que serve o SQL?
- 3- O SQL é dividido em duas partes, quais são elas? Descreva o objetivo de cada uma destas linguagens.
- 4- Crie um banco de dados chamado TESTE, e crie dentro dele as tabelas cujos comandos foram apresentados ao longo deste capítulo.
- 5- Adicione uma coluna para armazenar a cidade natal de todos os funcionários.

6. REFERÊNCIAS BIBLIOGRÁFICAS

- MySQL AB: MySQL 5.0 Reference Manual. Disponível em: <<http://www.mysql.com/documentation>>. Acesso em: 20 dez. 2005.
- Oliveira, Celso H. P. de (2002), SQL Curso Prático, Novatec.
- Jesus, João Batista de (2004) ANSI: SQL 89/92, Axcel Books.
- Costa, Rogério L. de C. (2004), SQL: Guia Prático, Brasport.
- Stephens, Ryan (2003), Aprenda em 24 horas SQL, 3 edição, Campus.
- Gennick, Jonathan (2004), SQL Pocket Guide, O'Reilly.
- Gulutzan, Peter; Pelzer, Trudy (1999), SQL-99 Complete, Really, CMP Books.
- Gulutzan, Peter; Pelzer, Trudy (2002), SQL Performance tuning, 1st edition, Addison-Wesley.

CAPÍTULO 7

1. INTRODUÇÃO

Os mecanismos de consultas de um sistema de banco de dados são concebidos de forma a possibilitar a criação da estrutura de dados, bem como o preenchimento e a manipulação das informações contidas nele. No capítulo 6 foi introduzida a linguagem SQL, mais especificamente a parte para a definição de dados (DDL). Assim, foram apresentados os comandos para a criação de bancos de dados, tabelas, além dos comandos para a alteração da estrutura de uma tabela.

Entretanto, existe a segunda parte da SQL que se refere à manipulação de dados, conhecida como DML (Data Manipulation Language), que significa Linguagem para Manipulação de Dados. Basicamente esta linguagem permite a inserção, alteração, exclusão e leitura das informações mantidas nas diversas tabelas de um banco de dados.

Utilizando a ferramenta de consulta MySQL Query Browser é possível realizar estas operações de forma a não utilizar comandos SQL explicitamente, já que o sistema constrói automaticamente os comandos SQL relacionados a cada tarefa. Desta forma os detalhes da sintaxe da linguagem ficam escondidos dos usuários e facilitam o acesso aos dados, uma vez que não é necessário entender profundamente a estrutura desta linguagem.

Neste capítulo serão apresentados e discutidos todos os comandos que compõem a DML. Para facilitar o entendimento destes comandos será utilizado o banco de dados curso construído anteriormente. Assim, o objetivo final é popular esta base de dados e introduzir alguns relatórios básicos empregando a linguagem SQL. No capítulo seguinte serão discutidos os relatórios avançados, bem como os comandos avançados para a manipulação de dados, tais como JOINs e sub-consultas, ou seja como extrair dados em várias tabelas.

Vale ressaltar que a DML apresenta comandos para a escrita e alteração de dados, bem como para a leitura de informações. Assim, inicialmente serão discutidos os comandos para a escrita de dados, permitindo preencher o banco de dados, e posteriormente, será apresentado o comando para ler as informações previamente armazenadas.

2. ENTENDENDO OS COMANDOS DML

A linguagem SQL apresenta basicamente três comandos para alteração de dados, que permitem realizar as tarefas de inclusão, modificação e exclusão de dados em uma tabela qualquer. Nesta seção serão ilustradas a sintaxe e exemplos de cada um destes comandos.

2.1. INCLUINDO DADOS COM O COMANDO INSERT

Para a inclusão de informações em uma tabela a linguagem SQL define o comando **INSERT**, cuja sintaxe é apresentada na **Listagem 1**.

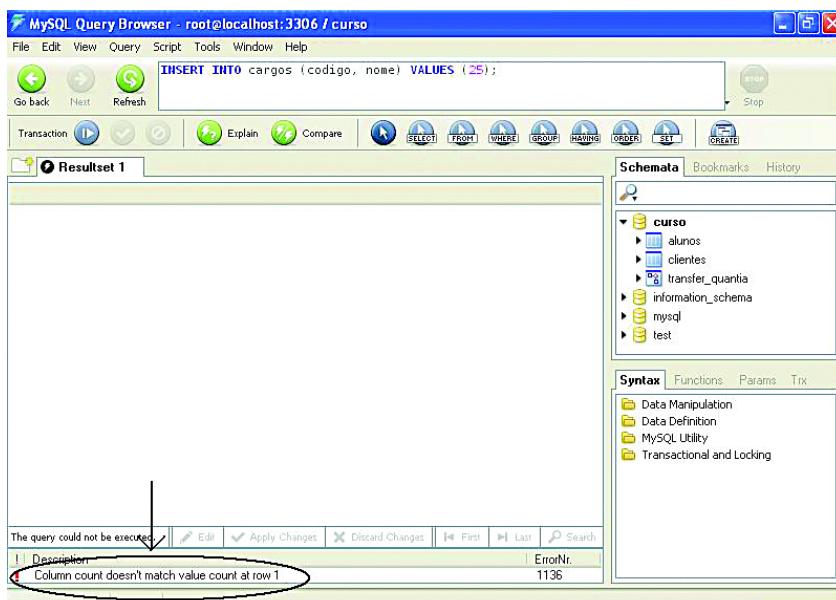
```
INSERT INTO nome_da_tabela (lista_de_colunas) VALUES (lista_de_valores)
```

Listagem 1: Sintaxe do comando **INSERT**

O comando **INSERT** consiste em informar o nome da tabela que se deseja inserir os dados, uma lista com o nome das colunas desta tabela para as quais serão informados os dados, e finalmente os dados para cada coluna informada anteriormente. Vale ressaltar que as colunas e os valores são separados por vírgulas, e o número de colunas

deve coincidir com o número de valores informados, caso contrário o sistema emitirá uma mensagem de erro e não executará a inserção. A **Figura 1** ilustra uma tentativa incorreta de inserir dados na tabela de cargos. O comando digitado é `INSERT INTO cargos (codigo, nome) VALUES (25)`.

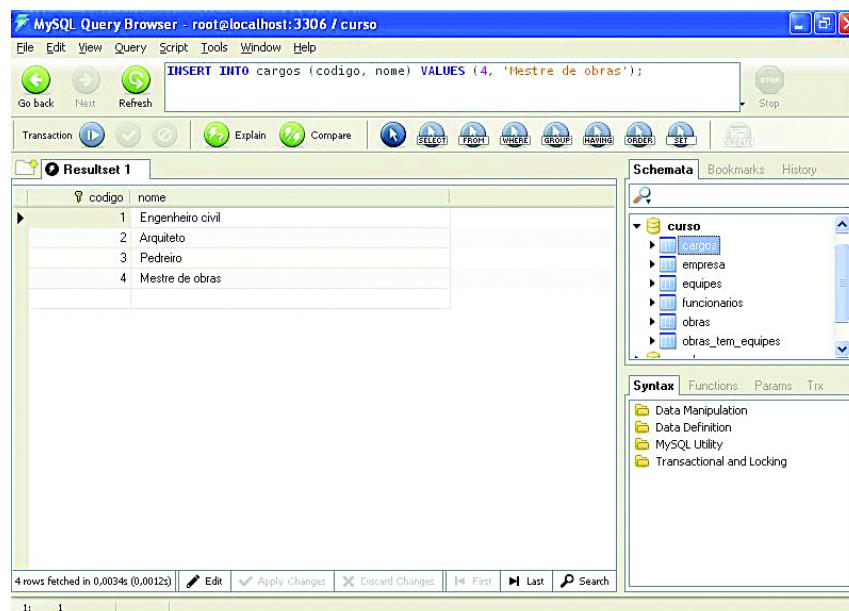
Figura 1:
INSERT com
maior número
de colunas que
valores



Na figura anterior percebe-se em destaque a mensagem “Column count doesn’t match value count at row 1”, que significa “Contagem de colunas não confere com a contagem de valores no registro 1”. Isto é, foram informadas duas colunas e apenas um valor para a coluna código, por isto a mensagem de erro.

Para ilustrar e facilitar o entendimento deste comando considera-se a inserção de mais um cargo na tabela de cargos do sistema. Portanto, um novo cargo será inserido com o código quatro e com o nome Mestre de obras. A **Figura 2** ilustra o comando `INSERT INTO cargos (codigo, nome) VALUES (4, 'Mestre de obras')`, executado a partir do MySQL Query Browser.

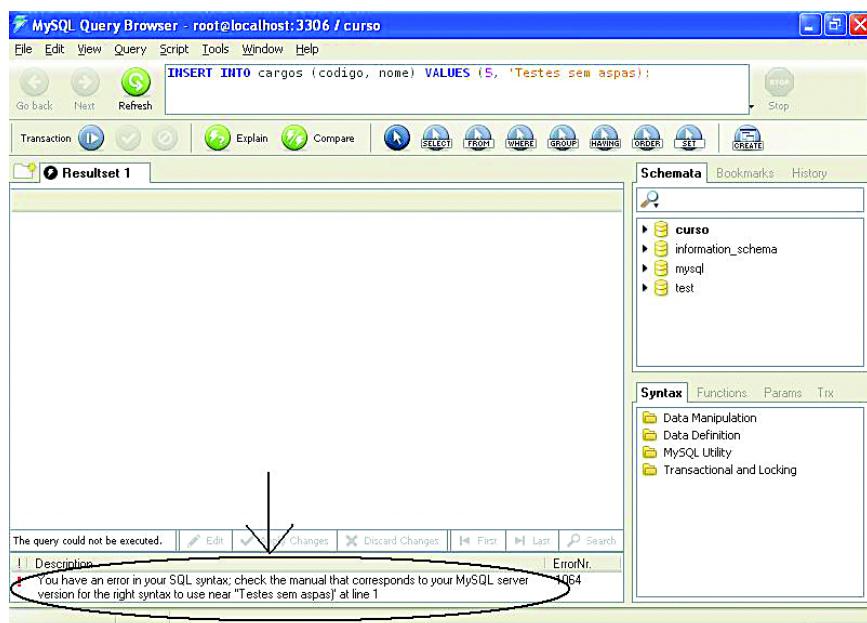
Figura 2:
Exemplo de
inserção do
cargo Mestre de
obras



No exemplo foi informado a lista de colunas (codigo e nome), e os valores 4 e 'Mestre de obras', representando as informações a serem armazenadas em cada uma das colunas. É importante perceber que a ordem dos valores informados segue a ordem das colunas. Isto é, primeiro aparece uma coluna código então o primeiro valor informado deve ser o código do cargo, e assim sucessivamente.

Outro detalhe importante de ser salientado é que para as colunas do tipo texto ou data, os valores devem ser informados entre aspas simples ('') ou aspas duplas (""), caso contrário o SGBD emitirá uma mensagem de erro e não executará a tarefa. A **Figura 3** ilustra a inserção do nome onde foi omitida uma das aspas, o comando executado foi `INSERT INTO cargos (codigo, nome) VALUES (5, 'Testes sem aspas')`.

Figura 3: Erro de sintaxe pela ausência de uma das aspas no nome



A mensagem "You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near "Testes sem aspas)", que significa "Você tem um erro de sintaxe no seu SQL; verifique o manual que corresponde à versão do seu servidor MySQL para a sintaxe correta próximo de "Testes sem aspas)", indica a ausência das aspas para encerrar o nome.

Já para a inserção de dados numéricos não é preciso utilizar as aspas, conforme ilustra o exemplo anterior.

Como a base de dados está vazia e tendo em vista o preenchimento dos dados para que possam ser feitos relatórios mais adiante, a **Tabela 1** fornece os comandos para a inserção de empresas, funcionários, equipes, obras, bem como a inclusão de equipes em algumas das obras cadastradas, isto é na tabela `obras_tem_equipas`.

Inserção de empresas

```

INSERT INTO `empresa` VALUES (1,'Empresa de banco LTDA',
'888.888.8888-0001/88', '(31)3333-4444');
INSERT INTO `empresa` VALUES (2,'Empresa do curso LTDA',
'999.999.9999-0001/88', '(31)4444-55555');

```

Inserção de equipes

```
INSERT INTO equipes (codigo, nome) VALUES (1, 'Equipe engenheiro');  
INSERT INTO equipes (codigo, nome) VALUES (2, 'Equipe pedreiros');  
INSERT INTO equipes (codigo, nome) VALUES (3, 'Equipe arquitetos');
```

Inserção de funcionários

```
INSERT INTO funcionarios (cpf, Cargos_codigo, nascimento, nome, telefone,  
Codigo_equipe) VALUES ('111.111.111-11', 1, '1970-10-12', 'José de Alencar',  
'(31)3333-3333', 1);
```

```
INSERT INTO funcionarios (cpf, Cargos_codigo, nascimento, nome, telefone,  
Codigo_equipe) VALUES ('222.222.222-22', 3, '1967-11-21', 'Paulo Goulart',  
'(31)4444-4444', 2);
```

```
INSERT INTO funcionarios (cpf, Cargos_codigo, nascimento, nome, telefone,  
Codigo_equipe) VALUES ('333.333.333-33', 3, '1987-11-21', 'Antônio Pereira',  
", 2);
```

```
INSERT INTO funcionarios (cpf, Cargos_codigo, nascimento, nome, telefone,  
Codigo_equipe) VALUES ('444.444.444-44', 3, '1975-12-10', 'Carlos de  
Nóbrega', '(31)5555-5544', 2);
```

```
INSERT INTO funcionarios (cpf, Cargos_codigo, nascimento, nome, telefone,  
Codigo_equipe) VALUES ('555.555.555-55', 2, '1977-06-23', 'Maria de Souza  
Melo', '(31)7777-5544', 3);
```

```
INSERT INTO funcionarios (cpf, Cargos_codigo, nascimento, nome, telefone,  
Codigo_equipe) VALUES ('666.666.666-66', 2, '1975-01-03', 'Antônio César',  
'(31)9999-5544', 3);
```

```
INSERT INTO funcionarios (cpf, Cargos_codigo, nascimento, nome, telefone,  
Codigo_equipe) VALUES ('777.777.777-77', 4, '1970-05-15', 'Carlos de Castro  
Silva', '(31)4567-8901', 2);
```

Inserção de obras

```
INSERT INTO `obras` VALUES (1,1,'Ponte','2005-01-01','2005-06-01');  
INSERT INTO `obras` VALUES (2,1,'Prédio','1999-10-15','2002-05-10');  
INSERT INTO `obras` VALUES (3,1,'Praça','2002-05-10','0000-00-00');
```

Inserção de equipes em obras

```
INSERT INTO `obras_tem_equipes` VALUES (1,1);  
INSERT INTO `obras_tem_equipes` VALUES (1,2);  
INSERT INTO `obras_tem_equipes` VALUES (1,3);  
INSERT INTO `obras_tem_equipes` VALUES (2,1);  
INSERT INTO `obras_tem_equipes` VALUES (2,2);  
INSERT INTO `obras_tem_equipes` VALUES (3,3);
```

Tabela 1: Inserção dos dados iniciais do banco de dados curso

Os comandos apresentados anteriormente podem ser executados via MySQL Query Browser de forma a construir a base de dados do curso. Um detalhe importante a ser lembrado é o fato de que a data deve ser informada de forma invertida, respeitando a regra 'ano-

mês-dia', conforme ocorre na data de nascimento dos funcionários.

Esta é a sintaxe básica do comando INSERT, existem variações aceitáveis deste comando, mas estas ficam fora do escopo deste livro, visto que este tem como objetivo apenas fornecer uma visão geral acerca da linguagem SQL.

2.2. ALTERANDO DADOS COM O COMANDO UPDATE

Existem situações em que há a necessidade de modificar uma informação armazenada na tabela, por exemplo, considera-se a alteração de telefone de um determinado funcionário, ou ainda a mudança na data de término de uma determinada obra. Neste caso, a tarefa de alterar o dado de uma coluna é realizada pelo comando UPDATE, cuja sintaxe básica é apresentada na **Listagem 2**.

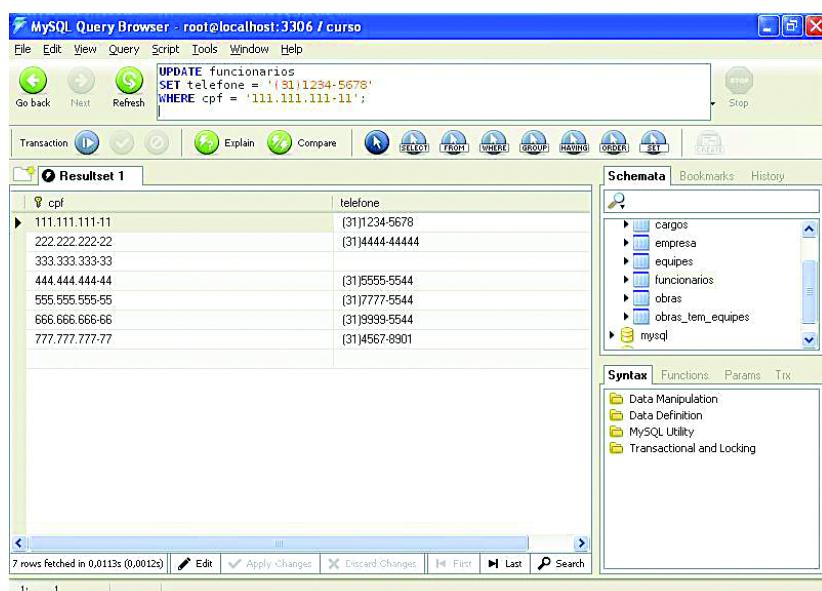
```
UPDATE nome_data_tabela
SET coluna=valor [, coluna=valor]
[WHERE critério_de_seleção]
```

Listagem 2: Sintaxe do comando UPDATE

O primeiro passo para a execução do comando UPDATE é definir a tabela que será modificada. Feito isto, informam-se as colunas e os novos valores que serão atribuídos a elas, especificando-as dentro da cláusula SET. Percebe-se que podem ser alteradas mais de uma coluna separando-as por vírgulas, sendo que este recurso é opcional, por isto está entre colchetes ([]).

A atualização de dados pode ser restrita a um conjunto de registros ou se aplicar a todas as linhas da tabela. Neste caso, existe a cláusula opcional WHERE (exibida entre colchetes ([])), que permite informar o critério de seleção de linhas, de forma a identificar apenas os registros que serão modificados pelo comando. Caso esta cláusula seja omitida, todos os registros da tabela serão alterados para o novo valor. A **Figura 4** fornece o comando para alterar o telefone do funcionário cujo CPF é igual a 111.111.111-11, isto é, UPDATE funcionarios SET telefone = '(31) 1234-5678' WHERE cpf = '111.111.111-11'.

Figura 4:
Atualizando o
telefone do
funcionário com
CPF
111.111.111-11



No exemplo anterior se o WHERE for omitido, o telefone de todos os funcionários passa a ser (31)1234-5678. Portanto, deve-se ter cuidado ao executar este comando para

que não seja obtido um resultado indesejado.

2.3. EXCLUINDO DADOS COM O COMANDO DELETE

Finalmente, os dados armazenados em uma base de dados podem ser excluídos por várias razões. Por exemplo, a saída de um funcionário da empresa deve eliminar o registro deste indivíduo da tabela de funcionários. Outra situação seria a saída de uma equipe da execução de uma determinada obra, isto é, o registro que relata este fato deve ser eliminado da tabela Obras_tem_equipes. Para a exclusão de linhas de uma tabela utiliza-se o comando DELETE, cuja sintaxe é apresentada na **Listagem 3**.

`DELETE FROM nome_data_tabela`

`[WHERE critério_de_seleção]`

Listagem 3: Sintaxe do comando DELETE

Assim, para a execução do comando DELETE deve-se informar o nome da tabela cujos registros serão eliminados, e opcionalmente o critério de seleção de linhas de forma a excluir apenas aquelas que atendam a um determinado requisito. Vale ressaltar que na omissão do WHERE serão retirados todos os registros da tabela. A **Figura 5** ilustra a remoção da empresa com o código igual a dois, cujo comando é `DELETE FROM empresa WHERE codigo = 2;`

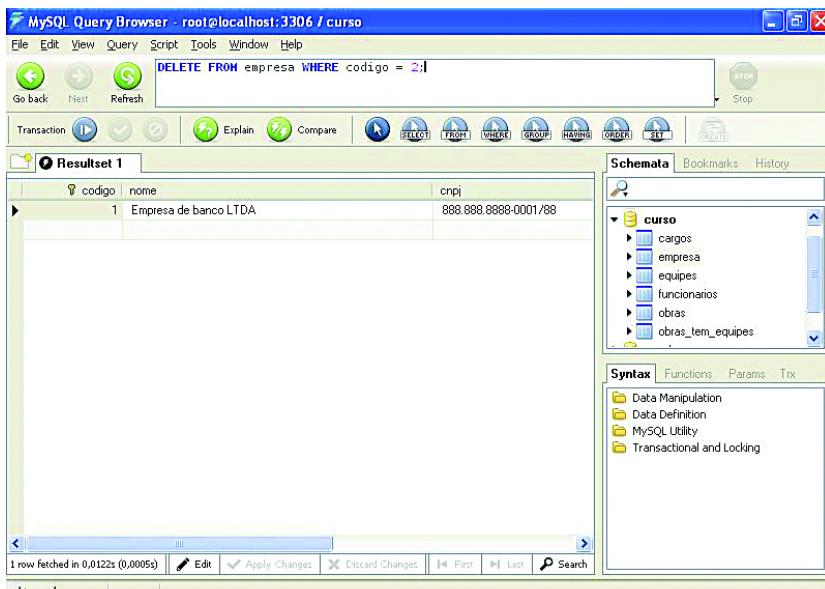


Figura 5:
Exclusão da
empresa com
o código
igual a 2

Assim como o UPDATE, no comando DELETE a cláusula WHERE é opcional por isto deve-se agir com prudência para que não sejam removidos ou alterados dados que não devem ser modificados.

3. INTRODUÇÃO AOS COMANDOS PARA LEITURA DE DADOS

A tarefa de acessar o conteúdo armazenado em uma base de dados é de extrema importância, já que sem o mesmo seria impossível utilizar estas informações como ferramenta de análise e controle do sistema que ela representa. O objetivo desta seção é fornecer uma introdução ao comando SELECT que possibilita a leitura dos dados existentes nas

tabelas do banco de dados. Aqui serão abordados apenas os relatórios simples, ou seja, que acessam apenas uma tabela, bem como a utilização de funções, que serão explicadas e exemplificadas no contexto em que forem empregadas.

3.1. O COMANDO SELECT BÁSICO

O comando SELECT é a estrutura da linguagem SQL que permite ler as informações de uma tabela, ou de um conjunto de tabelas que se relacionam. A sintaxe básica deste comando é apresentada na **Listagem 4**, e a explicação de cada cláusula que o constitui serão apresentados no decorrer desta seção.

```
SELECT lista_de_colunas
      FROM nome_da_tabela
        WHERE critério_de_seleção
          GROUP BY lista_de_colunas
            HAVING critério_de_seleção_no_resultado
              ORDER BY lista_de_colunas
```

Listagem 4: Sintaxe do comando SELECT

Na cláusula SELECT serão colocadas as colunas que serão exibidas pelo relatório, por exemplo, se for necessário mostrar apenas as colunas nome e CPF dos clientes, apenas estas colunas deverão ser escritas no SELECT. Caso queira exibir todas as colunas pode-se escrever o nome de todas as colunas ou, para simplificar a digitação, utilizar o caractere *, que significa todas as colunas.

Na cláusula FROM será especificada a tabela de onde os dados serão extraídos. Neste capítulo serão abordados apenas os relatórios com apenas uma tabela, mas podem aparecer várias tabelas nesta cláusula, permitindo ler dados de tabelas que se relacionam. Estes recursos serão explorados no próximo capítulo.

Para ilustrar estes mecanismos, considera-se o relatório que fornece o nome e o CPF de todos os funcionários cadastrados. A consulta que resolve esta pergunta está apresentada na **Figura 6**, representada pelo comando SELECT nome, cpf FROM funcionarios.

Figura 6: Exibir o nome e o CPF de todos os funcionários

The screenshot shows the MySQL Query Browser interface. The query window contains the command:

```
SELECT nome, cpf
FROM funcionarios;
```

The results are displayed in a table titled "Resultset 1". The columns are "nome" and "cpf". The data is as follows:

nome	cpf
José de Alencar	111.111.111-11
Paulo Goulart	222.222.222-22
Antônio Pereira	333.333.333-33
Carlos de Nobreza	444.444.444-44
Maria de Souza Melo	555.555.555-55
Antônio César	666.666.666-66
Carlos de Castro Silva	777.777.777-77

The bottom status bar indicates "7 rows fetched in 0.0109s (0.00130)".

Para ilustrar a utilização do *, seja exibir todos os atributos de todas as equipes cadastradas no sistema. A consulta SELECT * FROM equipes fornece este resultado e é ilustrada pela **Figura 7**.

The screenshot shows the MySQL Query Browser interface. The query window contains the following SQL code:

```
SELECT *
FROM equipes;
```

The results pane displays a table titled "Resultset 1" with two columns: "codigo" and "nome". The data is as follows:

codigo	nome
1	Equipe engenheiros
2	Equipe pedreiros
3	Equipe arquitetos

The status bar at the bottom indicates "3 rows fetched in 0.0031s (0.0013s)".

Figura 7:
Exibir todos
os atributos
de todas as
equipes

3.2. SELECIONANDO REGISTROS COM O WHERE

Como ocorre nos comandos UPDATE e DELETE é possível listar os dados de uma tabela de acordo com um critério de filtragem qualquer. De um modo geral, estes critérios são definidos em função dos atributos das tabelas utilizadas na cláusula FROM. Para ilustrar o fato, considera-se a listagem do nome de todos os funcionários cujo código do cargo seja igual a três, isto é, listar todos os funcionários que atuam como pedreiros. A **Figura 8** ilustra o comando SELECT * FROM funcionários WHERE Cargos_codigo = 3, que soluciona este problema.

The screenshot shows the MySQL Query Browser interface. The query window contains the following SQL code:

```
SELECT *
FROM funcionarios
WHERE Cargos_codigo = 3;
```

The results pane displays a table titled "Resultset 1" with three columns: "cpf", "Cargos_codigo", and "nome". The data is as follows:

cpf	Cargos_codigo	nome
222.222.222-22	3	Paulo Goulart
333.333.333-33	3	Antônio Pereira
444.444.444-44	3	Carlos de Nóbrega

The status bar at the bottom indicates "3 rows fetched in 0.0129s (0.0015s)".

Figura 8: Listar
o nome dos
funcionários que
atuam como
pedreiro

Podem existir ainda critérios mais complexos, como exibir apenas os funcionários que atuam como pedreiro e que tenham nascido depois do ano de 1970. Neste caso, haverá uma combinação de critérios para selecionar os registros desejados. A **Figura 9** fornece

a consulta `SELECT nome, Cargos_codigo, nascimento FROM funcionarios WHERE Cargos_codigo = 3 AND nascimento >= '1970-01-01'`, que responde a esta pergunta.

The screenshot shows the MySQL Query Browser interface. The query window contains:

```
MySQL Query Browser - root@localhost:3306 / curso
File Edit View Query Script Tools Window Help
SELECT nome, Cargos_codigo, nascimento
FROM funcionarios
WHERE Cargos_codigo = 3 AND nascimento >= '1970-01-01';
```

The Resultset 1 pane displays the following data:

nome	Cargos_codigo	nascimento
Antônio Pereira	3	1987-11-21
Carlos de Nóbrega	3	1975-12-10

Below the table, the status bar indicates "2 rows fetched in 0,0034s (0,0455s)".

Figura 9: Listar o nome dos funcionários que atuam como pedreiro e que nasceram depois de 1970

Neste exemplo, a cláusula `AND` foi utilizada para separar os vários critérios de seleção de dados, lembrando que não há limite em relação à quantidade de critérios de filtragem de dados. Existem situações onde existem vários critérios e caso um deles se verifique já seria suficiente para satisfazer a busca. Por exemplo, deseja-se listar todos os funcionários que sejam pedreiros ou que se chamem José de Alencar. A **Figura 10** ilustra a consulta que resolve este problema, neste caso, os registros que atendam a algum dos critérios apareceram no resultado. Se fosse utilizado o `AND`, apenas os registros que satiszessem a ambos os critérios seriam listados. O comando ilustrado é `SELECT * FROM funcionarios WHERE Cargos_codigo = 3 OR nome = 'José de Alencar'`.

The screenshot shows the MySQL Query Browser interface. The query window contains:

```
MySQL Query Browser - root@localhost:3306 / curso
File Edit View Query Script Tools Window Help
SELECT *
FROM funcionarios
WHERE Cargos_codigo = 3;
```

The Resultset 1 pane displays the following data:

cpf	Cargos_codigo	nome
222.222.222-22	3	Paulo Goulart
333.333.333-33	3	Antônio Pereira
444.444.444-44	3	Carlos de Nóbrega

Below the table, the status bar indicates "3 rows fetched in 0,0129s (0,0015s)".

Figura 10: Listar o nome dos funcionários que atuam como pedreiro ou que se chamam José de Alencar

Para colunas textuais é possível realizar consultas para comparar apenas uma parte do texto. Por exemplo, deseja-se listar o nome de todos os funcionários que comecem com Carlos. A consulta `SELECT nome FROM funcionarios WHERE nome LIKE 'Carlos%'` fornece este resultado e é apresentada na **Figura 11**.

Figura 11:
Listar o nome dos funcionários com nome começando com Carlos

The screenshot shows the MySQL Query Browser interface. In the query editor, the following SQL command is entered:

```
SELECT nome
FROM funcionarios
WHERE nome LIKE 'Carlos%';
```

The results are displayed in the Resultset 1 tab, showing two rows:

nome
Carlos de Castro Silva
Carlos de Nóbrega

Below the table, it says "2 rows fetched in 0,0313s (0,0013s)".

3.3. AGRUPANDO DADOS COM O GROUP BY E SELECIONANDO COM O HAVING

Existem relatórios que têm como objetivo calcular a soma ou totalização de dados dentro de um critério específico. Para ilustrar esta situação deseja-se saber a quantidade de equipes que atuaram em cada uma das obras da empresa. Para isto, deve-se acessar a tabela de obras_tem_equipes e contar, para cada obra, quantos registros existem para cada equipe. A **Figura 12** contém a solução para esta pergunta, que é dada pelo comando `SELECT.obras_codigo, COUNT(*) FROM obras_tem_equipes GROUP BY obras_codigo;`

Figura 12:
Listar as obras e o total de equipes envolvidas

The screenshot shows the MySQL Query Browser interface. In the query editor, the following SQL command is entered:

```
SELECT obras_codigo, COUNT(*)
FROM obras_tem_equipes
GROUP BY obras_codigo;
```

The results are displayed in the Resultset 1 tab, showing three rows:

obras_codigo	COUNT(*)
1	3
2	2
3	1

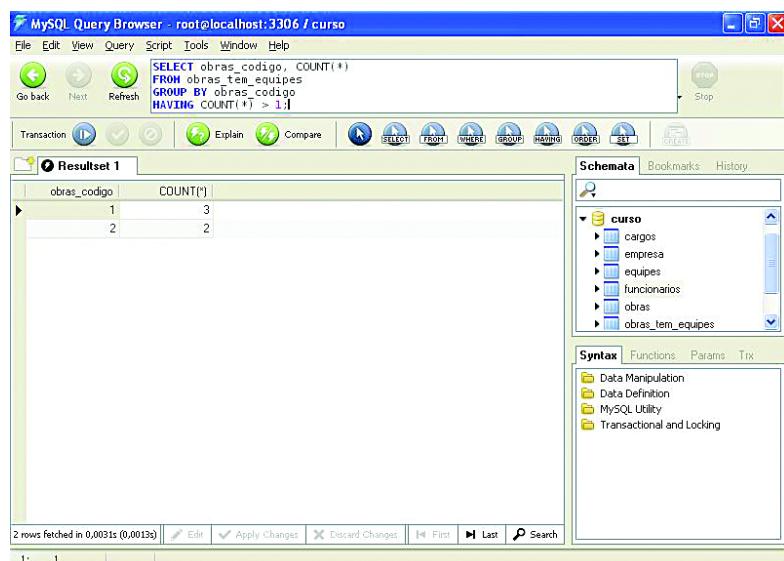
Below the table, it says "3 rows fetched in 0,0049s (0,0348s)".

No exemplo, utiliza-se a função `COUNT(*)` que realiza a contagem dos registros encontrados dentro do critério de agrupamento `GROUP BY`. Ou seja, para cada obra distinta contam-se as ocorrências distintas de equipes. O mesmo raciocínio é aplicado para a soma dos valores de uma coluna, feito com a função `SUM(nome_da_coluna)`.

Além do agrupamento de dados, pode ser necessário estabelecer critérios sobre o resultado calculado, neste caso, o filtro não aparecerá no `WHERE`, mas sim na cláusula `HAVING`.

Isto é, sempre que houver o GROUP BY e for desejável a pesquisa sobre o resultado gerado, utiliza-se o HAVING e não o WHERE como nos exemplos anteriores. Para ilustrar a utilização do HAVING a **Figura 13** fornece a consulta que exibe o código das obras que tiveram mais que uma equipe envolvida em sua elaboração. O comando ilustrado é SELECT obras_codigo, COUNT(*) FROM obras_tem_equipes GROUP BY obras_codigo HAVING COUNT(*) > 1.

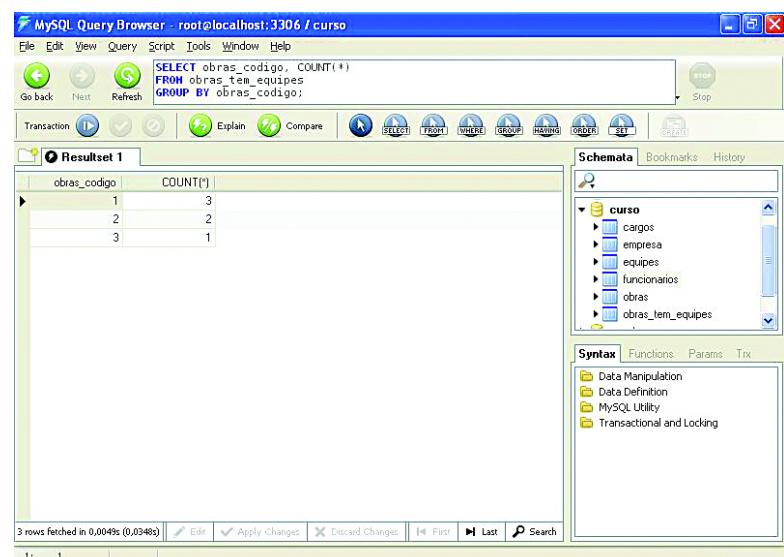
Figura 13:
Listar as obras
que tenham
mais que uma
equipe
envolvida



3.4. ORDENANDO O RESULTADO COM O ORDER BY

Finalmente, a última cláusula do comando SELECT se refere à ordenação dos dados da consulta. Pode-se exibir a listagem de acordo com o critério de ordenação que for mais conveniente para o relatório. No caso da listagem de funcionários pode-se ordená-la por nome, data de nascimento, ou até mesmo pelo telefone. Esta tarefa é feita por meio do ORDER BY, que faz referência a uma coluna ou um conjunto de colunas separadas por vírgulas, caso tenha mais de um critério. A **Figura 14** ilustra uma lista de todos os cargos ordenando-os pelo nome. O comando para este propósito é o SELECT * FROM cargos ORDER BY nome.

Figura 14:
Listar os
cargos
ordenados por
nome



4. CONCLUSÕES

Este capítulo encerra a apresentação de toda a linguagem SQL, tanto a definição quanto a manipulação de dados. Neste contexto foram discutidos os comandos para escrita de dados, bem como as técnicas para a criação de relatórios básicos utilizando-se a linguagem SQL. A estrutura do SELECT apresentada neste capítulo será utilizada em situações mais sofisticadas nos capítulos seguintes, mas todas as cláusulas que o define foram estudadas neste capítulo.

Vale ressaltar que o objetivo deste curso é formar uma visão geral do que é a linguagem SQL e como esta pode ser utilizada na prática. Portanto, existem variações ou incrementos para estes comandos que foram omitidos para simplificar a discussão. Mas, mesmo assim, estas sintaxes básicas são suficientes para utilizar um banco de dados em um sistema qualquer.

5. EXERCÍCIOS DE FIXAÇÃO

- 1- Listar o nome de todas as equipes que começam com a letra p.
- 2- Listar o nome de todos os funcionários que trabalham como engenheiro.
- 3- Listar o total de registros da tabela de obras.
- 4- Insira você como um funcionário da empresa, no cargo de arquiteto.
- 5- Altere o seu cargo para engenheiro.
- 6- Remova da tabela todos os funcionários que começam com a letra C.

6. REFERÊNCIAS BIBLIOGRÁFICAS

- MySQL AB: MySQL 5.0 Reference Manual. Disponível em: <<http://www.mysql.com/documentation>>. Acesso em: 20 dez. 2005.
- Oliveira, Celso H. P. de (2002), SQL Curso Prático, Novatec.
- Jesus, João Batista de (2004) ANSI: SQL 89/92, Axcel Books.
- Costa, Rogério L. de C. (2004), SQL: Guia Prático, Brasport.
- Stephens, Ryan (2003), Aprenda em 24 horas SQL, 3 edição, Campus.
- Gennick, Jonathan (2004), SQL Pocket Guide, O'Reilly.
- Gulutzan, Peter; Pelzer, Trudy (1999), SQL-99 Complete, Really, CMP Books.
- Gulutzan, Peter; Pelzer, Trudy (2002), SQL Performance tuning, 1st edition, Addison-Wesley.

CAPÍTULO 8

1. INTRODUÇÃO

O capítulo 7 abordou a linguagem de manipulação de dados do SQL, descrevendo os principais comandos para escrita e leitura no banco de dados. Mais especificamente o comando SELECT básico foi apresentado, sendo discutidas as principais cláusulas que o constitui. O objetivo inicial era introduzir os recursos básicos para a leitura de informações, extraíndo dados de uma única tabela do sistema. Para isto foram exibidas consultas simples, onde foi possível compreender as técnicas para a construção de filtros de dados através do WHERE, definição de agrupamentos por meio do GROUP BY, e finalmente, ordenar o resultado da pesquisa através do ORDER BY.

Desta forma foram apresentados todos os aspectos da linguagem SQL, possibilitando o entendimento e a utilização do comando SELECT para a construção de relatórios simplificados. Porém, em sistemas reais é comum a existência de relatórios mais complexos, já que há um conjunto grande de tabelas e relacionamentos entre elas. Portanto, a extração de dados a partir do acesso a uma única tabela é bastante limitada em alguns casos, e não é capaz de prover todas as informações necessárias para os usuários do sistema. Tomando como base o banco de dados sobre a empresa de construção civil, seria bastante razoável que um usuário quisesse listar o nome de todos os funcionários e equipes às quais estes pertencem. Ou ainda, exibir o nome das equipes que participaram do desenvolvimento de uma obra específica. E assim, poderiam ser criados os mais diversos relatórios baseando-se nas tabelas que armazenam informações relacionadas.

A linguagem SQL provê mecanismos para resolver este tipo de questão através da especificação de uma lista de tabelas na cláusula FROM do SELECT. Este recurso permite extrair informações de mais de uma tabela, desde que haja um relacionamento entre elas, isto é, caso haja uma coluna em comum entre estas entidades. Este recurso é conhecido como JOIN (junção), e será descrito com detalhes no decorrer deste capítulo.

Além dos agrupamentos de tabelas, a linguagem SQL fornece uma abordagem conhecido como sub-consulta, isto é, a estrutura do comando SELECT pode ser utilizada dentro das cláusulas SELECT, FROM e WHERE de um outro comando SELECT. Este é um recurso poderoso e também será estudado com detalhes neste capítulo.

Desta forma torna-se possível elaborar consultas sofisticadas para resolver questões mais complexas em sistemas de bancos de dados. Além disto, será introduzido o conceito de funções, que na verdade são artifícios para realizar cálculos ou transformações de dados. Um exemplo de função é o DATE_FORMAT que é utilizado para formatar uma data de acordo com o padrão de data conhecido, e não no estilo “ano-mês-dia”, utilizando pelo MySQL.

2. TÉCNICAS PARA EXTRAÇÃO DE DADOS EM MÚLTIPLAS TABELAS - JOIN

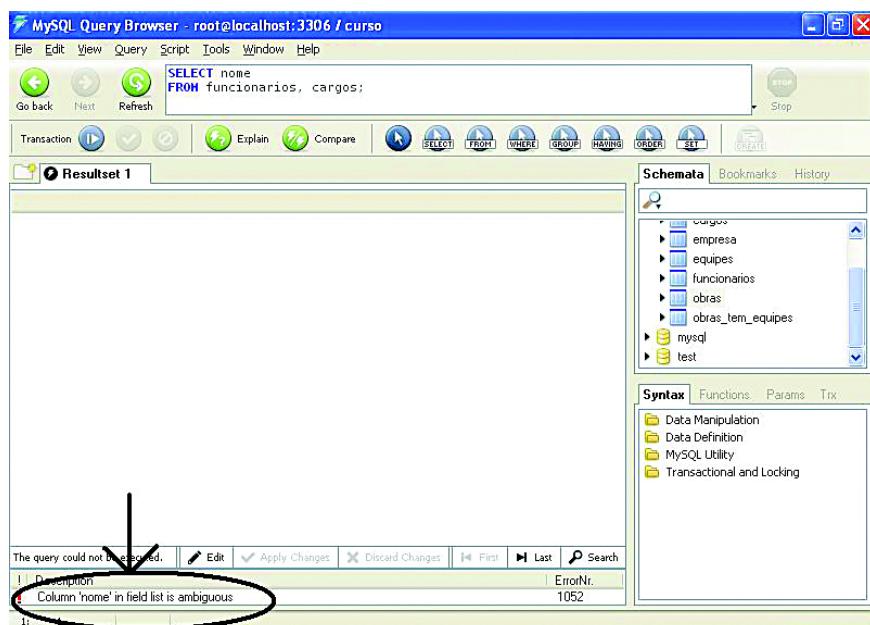
Em um sistema de banco de dados relacional as informações são armazenadas em diversas tabelas, e na maioria dos casos, o conteúdo destas tabelas compõe um descriptivo complexo a respeito da aplicação que ele representa. Na modelagem entidade-relacionamento define-se os elementos que constituem o sistema, bem como a forma como eles se relacionam. Estes relacionamentos são identificados pelas chaves primárias e

estrangeiras, que são na verdade os atributos comuns entre duas tabelas.

Para ilustrar esta situação, se considerada as tabelas cargos e funcionários do sistema de construção civil, existe um relacionamento entre estas entidades, já que cada funcionário da empresa possui um cargo. Esta interação entre os dois objetos é descrita pelo atributo código, que é chave primária da tabela cargos, e que é inserido na tabela de funcionários, identificando o cargo do mesmo (coluna Cargos_codigo).

Partindo do princípio de que há algo em comum entre tabelas relacionadas torna-se possível a elaboração de uma estrutura de pesquisa que faça a combinação destes registros e traga o resultado desejado. Este mecanismo é conhecido como JOIN, e para ilustrar o seu funcionamento, seja exibir o nome do cargo de todos os funcionários da empresa. Basicamente, o que precisa ser feito é informar as tabelas cargos e funcionários na cláusula FROM do comando SELECT. A **Figura 1** ilustra a primeira tentativa de resolver o problema da listagem dos cargos de funcionários, através da submissão do comando SELECT nome FROM funcionários, cargos.

Figura 1:
Utilizando o
JOIN, problema
de ambigüidade
de nomes de
colunas.



Na figura percebe-se que no FROM aparecem as duas tabelas separadas por vírgulas, e que conforme o destaque, o sistema emitiu uma mensagem de erro “Column ‘nome’ in field list is ambiguous”, que quer dizer “Coluna ‘nome’ na lista de campos é ambígua”. Isto é, a coluna nome é ambígua e a consulta não pode ser processada. Este é um problema típico de consultas que fazem acesso a mais de uma tabela. Como as tabelas cargos e funcionários possuem uma coluna chamada nome, o sistema não sabe qual delas listar, ou seja, o nome do funcionário ou o nome do cargo. Neste caso, é preciso prefixar o nome da coluna com o nome da tabela, resolvendo assim a ambigüidade. Portanto, todas as vezes que for necessário relacionar duas tabelas que possuam nomes iguais, deve-se utilizar a sintaxe nome_da_tabela.nome_da_coluna, conforme ilustra a **Figura 2**, com o comando SELECT funcionários.nome, cargos.nome FROM funcionários, cargos.

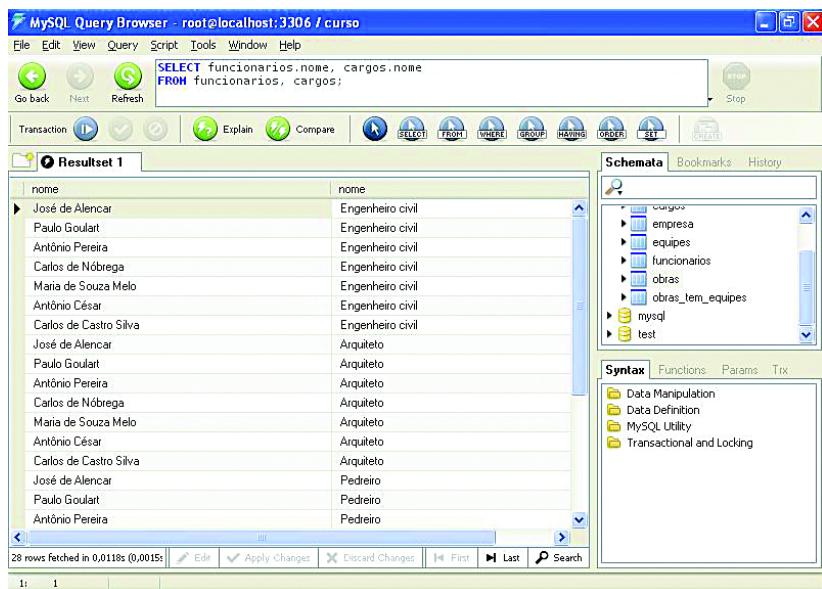


Figura 2:
Resolvendo a
ambigüidade
de nomes de
colunas

2.1. O PRODUTO CARTESIANO, UM ERRO COMUM NA ELABORAÇÃO DO JOIN

A **Figura 2** ilustra uma situação comum que ocorre com iniciantes na linguagem SQL, que é conhecido como produto cartesiano. Percebe-se que o resultado gerado associa todos os cargos com todos os funcionários cadastrados, ou seja, é uma combinação de todos os registros das tabelas envolvidas no JOIN. No caso o conjunto resultante contém 28 registros, que é a combinação dos quatro cargos com os sete funcionários existentes na base de dados. Vale ressaltar que este relatório não possui significado, já que associa um único funcionário com vários cargos. Desta forma, é impossível saber qual é o cargo real do mesmo. É possível perceber esta situação se observado o funcionário José de Alencar, exibido como engenheiro, arquiteto, pedreiro e mestre de obras, o que está incorreto ou inconsistente.

Além de não fornecer um resultado válido, esta consulta apresenta um alto custo para ser executada, pois exige que o SGBD calcule a combinação dos registros existentes em todas as tabelas. Para resolver esta situação é preciso informar no comando SQL, qual o critério comum para as tabelas, de forma a evitar o produto cartesiano. No exemplo, as colunas código da tabela cargos e Cargos_codigo da tabela de funcionários constituem o elo entre estas tabelas. Assim, deve-se especificar na cláusula WHERE, que apenas os registros que apresentem valores iguais nestas colunas devem ser exibidos. A **Figura 3** ilustra o comando que lista corretamente o nome dos funcionários e os seus cargos, que é o `SELECT funcionarios.nome, cargos.nome FROM funcionarios, cargos WHERE codigo = cargos_codigo`.

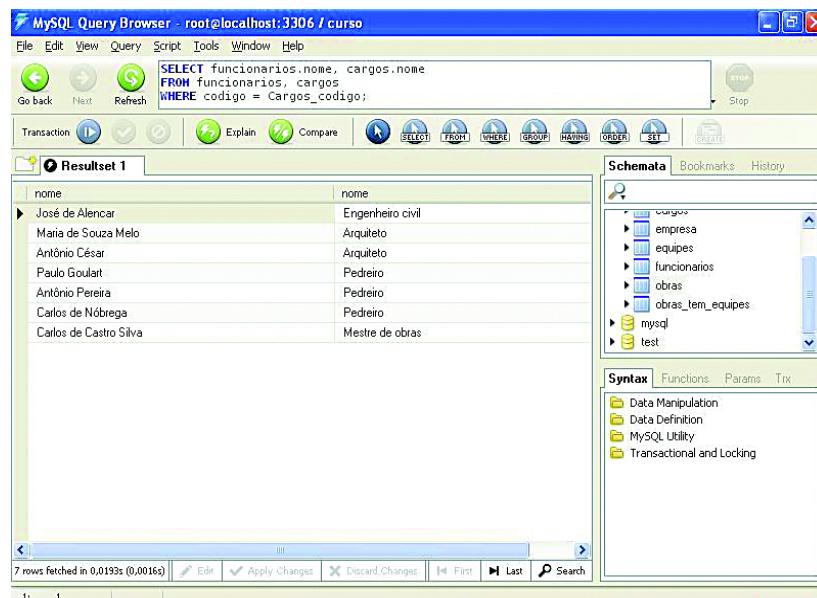


Figura 3:
Listando o nome e os cargos dos funcionários

Neste caso, torna-se possível extrair as informações contidas em tabelas que se relacionam. Vale destacar que podem existir relacionamentos entre mais de duas tabelas, como é o caso das tabelas cargos, funcionários e equipes. Portanto, o raciocínio utilizado para duas tabelas pode ser estendido para qualquer número de tabelas, desde que os critérios de ligação entre elas sejam estabelecidos na cláusula WHERE.

A **Figura 4** exemplifica um relatório onde deve ser mostrado o nome do funcionário, o seu cargo e o nome da equipe ao qual ele pertence. Neste caso, três tabelas deverão ser relacionadas e o critério de junção das mesmas são as colunas código (tabela cargos), Cargos_codigo e Equipes_codigo da tabela de funcionários e código da tabela de equipes. O comando ilustrado é o SELECT funcionários.nome, cargos.nome, equipes.nome FROM funcionários, cargos, equipes WHERE cargos.codigo = cargos_codigo AND equipes.codigo = Código_equipe.

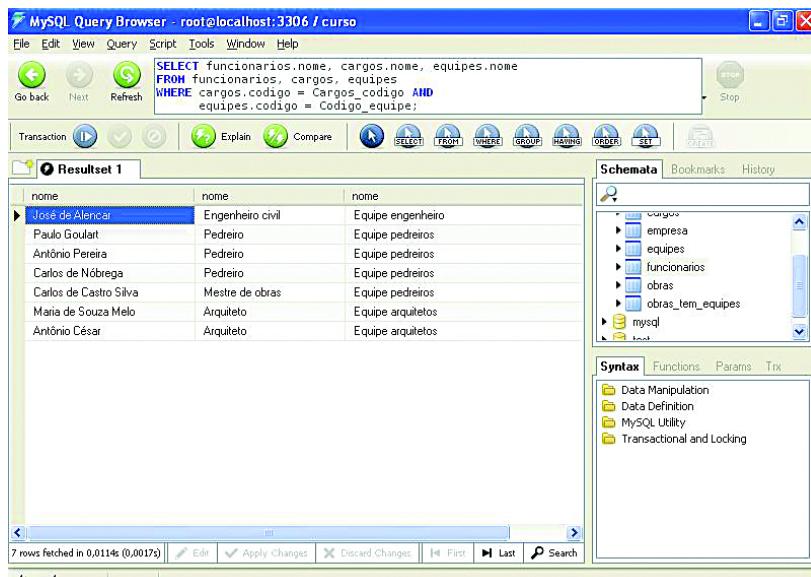


Figura 4:
Listando o nome e os cargos dos funcionários e as suas equipes

Desta forma, é possível construir relatórios complexos através da união de diversas tabelas em um único comando SELECT.

2.2. INNER JOIN, UMA SINTAXE ALTERNATIVA PARA A JUNÇÃO DE TABELAS

A junção de tabelas apresentada na seção anterior permite a solução de problemas complexos, mas trás um problema em relação à organização do comando. Neste caso os critérios de relacionamento das tabelas e os filtros de dados são colocados juntos na cláusula WHERE, o que dificulta sobremaneira a leitura e o entendimento do comando.

Para uma abordagem mais didática e mais organizada para a criação de junções de tabelas, existe a sintaxe do INNER JOIN, conforme ilustra a **Figura 5**. Esta figura exibe um relatório com todos os funcionários que nasceram depois de 1970, exibindo também os seus cargos e equipes. O comando é SELECT funcionários.nome, cargos.nome, equipes.nome FROM funcionários INNER JOIN cargos ON (cargos.codigo = cargos.codigo) INNER JOIN equipes ON (equipes.codigo = Código_equipe) WHERE nascimento > '1970-01-01'.

Figura 5:
Relatório de
funcionários,
cargos e
equipes
utilizando o
INNER JOIN

The screenshot shows the MySQL Query Browser interface. The query window contains the following SQL code:

```
SELECT funcionários.nome, cargos.nome, equipes.nome
FROM funcionários INNER JOIN cargos ON (cargos.codigo = Cargos_codigo)
INNER JOIN equipes ON (equipes.codigo = Código_equipe)
WHERE nascimento > '1970-01-01';
```

The results window displays a table titled "Resultset 1" with three columns: nome, nome, and nome. The data is as follows:

	nome	nome	nome
▶	José de Alencar	Engenheiro civil	Equipe engenheiro
	Antônio Pereira	Pedreiro	Equipe pedreiros
	Carlos de Nóbrega	Pedreiro	Equipe pedreiros
	Carlos de Castro Silva	Mestre de obras	Equipe pedreiros
	Maria de Souza Melo	Arquiteto	Equipe arquitetos
	Antônio César	Arquiteto	Equipe arquitetos

The bottom status bar indicates "6 rows fetched in 0,0113s (0,0005s)".

No comando é fácil identificar como as tabelas se relacionam, já que o critério de junção é especificado imediatamente após as tabelas utilizando-se a cláusula ON. Neste caso apenas os filtros serão colocados na cláusula WHERE isolando os critérios de relacionamento entre tabelas. Vale lembrar que o resultado obtido com a vírgula é o mesmo do INNER JOIN, utiliza-se o último apenas por questões de estilo e para facilitar a manutenção da consulta gerada.

3. ENTENDENDO OS MECANISMOS DE SUB-CONSULTAS

Os recursos da linguagem SQL para a junção de tabelas permitem elaborar relatórios envolvendo diversas tabelas, como ilustra a seção anterior. Contudo, os JOINS não são a única possibilidade para a extração de dados em várias tabelas. A linguagem SQL fornece uma estrutura conhecida como sub-consulta ou sub-selects onde é possível utilizar um comando SELECT dentro de outro comando SELECT.

Neste caso, o comando SELECT ilustrado até aqui poderá compor a cláusula SELECT, FROM ou WHERE de um outro comando SELECT, criando assim uma hierarquia de comandos que permite resolver problemas bastante complexos, e que em algumas situações

não são passíveis de serem resolvidos via JOIN. Nesta seção serão apresentadas três formas de sub-consultas, bem como o comando UNION que permitem agrupar o resultado de dois comandos SELECT de forma a compor um único conjunto resultante.

3.1. UTILIZANDO UM SELECT DENTRO DA CLÁUSULA SELECT

Para ilustrar o uso de uma consulta dentro da cláusula SELECT, considera-se o seguinte relatório: listar o nome dos funcionários e as suas datas de nascimento. A **Figura 6** ilustra a solução deste problema utilizando-se uma sub-consulta. O comando ilustrado é o SELECT nome, (SELECT nascimento FROM funcionários f WHERE f.cpf = f1.cpf) FROM funcionários f1.

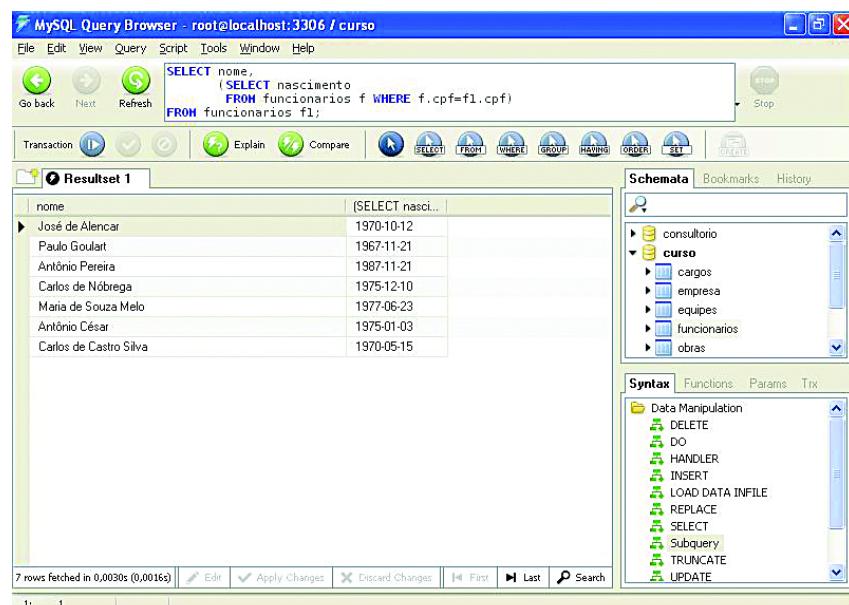


Figura 6: Exibindo o nome e a data de nascimento dos funcionários

The screenshot shows the MySQL Query Browser interface. The query window contains the following SQL code:

```
MySQL Query Browser - root@localhost:3306 / curso
File Edit View Query Script Tools Window Help
SELECT nome,
       (SELECT nascimento
        FROM funcionarios f WHERE f.cpf=f1.cpf)
  FROM funcionarios f1;
```

The results pane displays a table titled "Resultset 1" with the following data:

nome	(SELECT nasci...)
José de Alencar	1970-10-12
Paulo Goulart	1967-11-21
Antônio Pereira	1987-11-21
Carlos de Nóbrega	1975-12-10
Maria de Souza Melo	1977-06-23
Antônio César	1975-01-03
Carlos de Castro Silva	1970-05-15

The bottom status bar indicates "7 rows fetched in 0.0030s (0.0016s)".

É óbvio que a solução trivial seria listar diretamente a coluna nascimento, mas o objetivo aqui é apresentar o funcionamento de uma consulta SELECT colocada dentro da SELECT de outra consulta.

3.2. UTILIZANDO UM SELECT DENTRO DA CLÁUSULA FROM

Uma consulta pode aparecer dentro da cláusula FROM de outra consulta e, neste caso, a consulta colocada dentro da cláusula FROM será resolvida e seu resultado será entendido pela consulta externa como se fosse uma nova tabela. Esta situação pode ser ilustrada pela consulta que lista o nome de todos os funcionários que compõem a equipe de pedreiros. A **Figura 7** ilustra esta consulta através do comando SELECT * FROM (SELECT funcionários.nome FROM equipes INNER JOIN funcionários ON (código = Código_equipe) WHERE equipes.nome = 'Equipe pedreiros') t.

The screenshot shows the MySQL Query Browser interface. The query window contains the following SQL code:

```
SELECT *
FROM (SELECT funcionarios.nome
      FROM equipes INNER JOIN funcionarios ON (codigo=Codigo_equipe)
      WHERE equipes.nome = 'Equipe pedreiros') t;
```

The results pane, titled "Resultset 1", displays a table with one column "nome" containing four rows: Paulo Goulart, Antônio Pereira, Carlos de Nóbrega, and Carlos de Castro Silva.

The right side of the interface includes a "Schema" tree view showing database structures like consultorio, curso, cargos, empresa, equipes, funcionários, and obras. A "Syntax" tab is also visible.

Figura 7:
Listar o nome
de todos os
funcionários
da equipe de
pedreiros

Vale observar que esta consulta poderia ser resolvida facilmente através do JOIN, mas esta abordagem pode ser mais atraente ou intuitiva para aqueles que estão iniciando o uso da linguagem SQL.

3.3. UTILIZANDO UM SELECT DENTRO DA CLÁUSULA WHERE

As consultas dentro da cláusula WHERE são de grande utilidade especialmente para extrair informações em relacionamentos do tipo muitos para muitos, pois facilita comparar uma determinada coluna com uma lista de valores retornados por uma outra SELECT. Um exemplo prático da utilização deste tipo de consulta é descrito pela **Figura 8**, onde serão exibidos os nomes de todas as equipes que possuem funcionários associados a elas, através do comando `SELECT equipes.nome FROM equipes WHERE EXISTS (SELECT * FROM funcionários WHERE codigo=Código_equipe)`.

The screenshot shows the MySQL Query Browser interface. The query window contains the following SQL code:

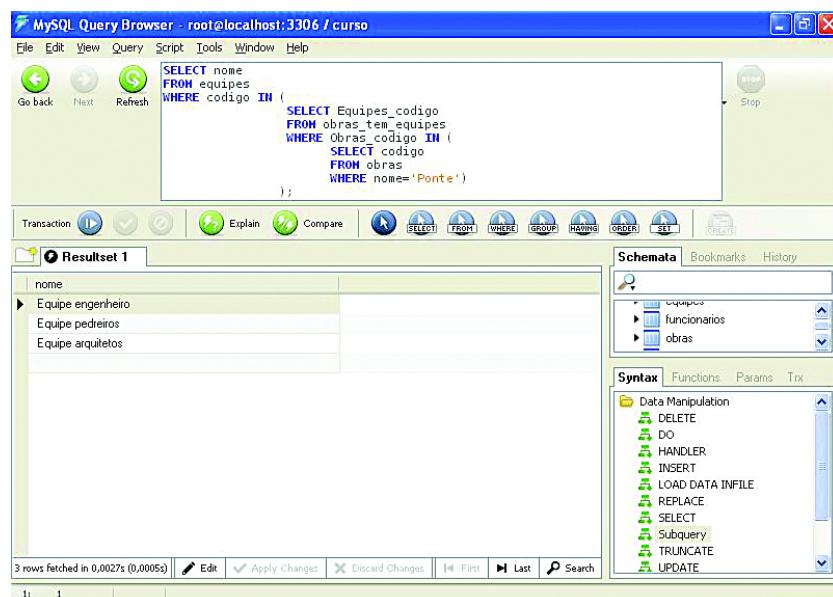
```
SELECT equipes.nome
FROM equipes
WHERE EXISTS (SELECT *
              FROM funcionários
              WHERE codigo=Código_equipe);
```

The results pane, titled "Resultset 1", displays a table with one column "nome" containing three rows: Equipe engenheiro, Equipe pedreiros, and Equipe arquitetos.

The right side of the interface includes a "Schema" tree view showing database structures like consultorio, curso, cargos, empresa, equipes, and funcionários. A "Syntax" tab is also visible.

Figura 8: Nome
das equipes
contendo
funcionários

No exemplo anterior foi utilizada a estrutura EXISTS para verificar se existem registros na consulta externa associados ao resultado da consulta interna. Vale ressaltar que qualquer tipo de operador pode ser utilizado neste caso, como na lista dos nomes das equipes que participaram da obra denominada Ponte. Este relatório é apresentado na **Figura 9**, definido pelo comando SELECT nome FROM equipes WHERE codigo IN (SELECT equipes_codigo FROM obras_tem_equipes WHERE obras_codigo IN (SELECT codigo FROM obras WHERE nome = 'Ponte')).



```

MySQL Query Browser - root@localhost:3306 / curso
File Edit View Query Script Tools Window Help
Go back Next Refresh Stop
SELECT nome
FROM equipes
WHERE codigo IN (
    SELECT Equipes_codigo
    FROM obras_tem_equipes
    WHERE obras_codigo IN (
        SELECT codigo
        FROM obras
        WHERE nome='Ponte'
    )
);

```

Resultset 1

nome
Equipe engenheiro
Equipe pedreiros
Equipe arquitetos

3 rows fetched in 0,0027s (0,0005s)

Schemata Bookmarks History

Syntax Functions Params Trix

Data Manipulation

- DELETE
- DO
- HANDLER
- INSERT
- LOAD DATA INFILE
- REPLACE
- SELECT
- Subquery
- TRUNCATE
- UPDATE

Figura 9: Nome das equipes envolvidas na obra Ponte

No exemplo é possível perceber que existem mais de uma consulta dentro da outra, e representa o fato de que não há limite para a utilização de sub-consultas. Isto é, podem existir tantas sub-selects quantas forem necessárias para solucionar a questão, da mesma forma que ocorre no JOIN.

De um modo geral, emprega-se as sub-consultas para a solução de problemas que não sejam solucionáveis via JOIN, ou que a solução através da junção de tabelas seja muito complexa e de difícil construção. No entanto, a maioria dos casos podem ser resolvidos com o JOIN, e o custo para a execução destes é, em geral, menor que o das sub-consultas. Portanto, ambos podem ser utilizados, e a opção por uma ou outra abordagem fica a critério do programador. De um modo geral a decisão será tomada de forma arbitrária, levando em conta a capacidade e experiência do programador para utilizar estes mecanismos de consultas.

3.4. AGRUPANDO RESULTADOS COM O COMANDO UNION

Por último, existem situações em que se deseja agrupar o resultado de duas consultas formando um único conjunto resultante. O comando que possibilita esta tarefa é o UNION e a sua utilização é ilustrada pela **Figura 10**, cujo comando é o (SELECT nome FROM funcionários WHERE Cargos_codigo = (SELECT codigo FROM cargos WHERE nome = 'Pedreiro')) UNION (SELECT nome FORM funcionários WHERE Cargos_codigo = (SELECT código FROM cargos WHERE nome='Engenheiro')).

Figura 10: Nome dos funcionários pedreiros e engenheiros

The screenshot shows the MySQL Query Browser interface. The query window contains the following SQL code:

```

SELECT nome
FROM funcionarios
WHERE Cargos_codigo = (SELECT codigo FROM cargos
                       WHERE nome='Pedreiro')
UNION
SELECT nome
FROM funcionarios
WHERE Cargos_codigo = (SELECT codigo FROM cargos
                       WHERE nome='Engenheiro')
    
```

The Resultset window displays the names of employees in the 'funcionarios' table who have the 'Pedreiro' or 'Engenheiro' job title. The results are:

nome
Paulo Goulart
Antônio Pereira
Carlos de Nóbrega

The Schema window shows the database structure with tables 'consultorio' and 'curso'. The Syntax window lists various SQL commands.

A consulta do exemplo anterior lista o nome dos funcionários com os cargos de pedreiro e engenheiro, visando apresentar esta estrutura do SQL. É importante ressaltar que a melhor solução para este relatório é utilizar uma única consulta com o filtro pelos cargos desejados. O objetivo desta consulta é apenas introduzir o comando UNION.

4. UTILIZANDO FUNÇÕES PARA TRANSFORMAÇÃO DE DADOS

A maioria dos SGBD fornecem um recurso importante para a elaboração das consultas SQL que são as funções. Uma função é utilizada na prática para realizar transformações nos dados. Pegando como exemplo a data do MySQL, esta é armazenada no formato “aaaa-mm-dd”, ou seja ano, mês e dia. É fácil perceber que este não é um formato de data com o qual as pessoas estejam habituadas. Portanto seria de extrema relevância que as datas armazenadas pelo MySQL fossem exibidas em relatórios no formato “dd/mm/aaaa”, ou seja, dia, mês e ano separados por barras.

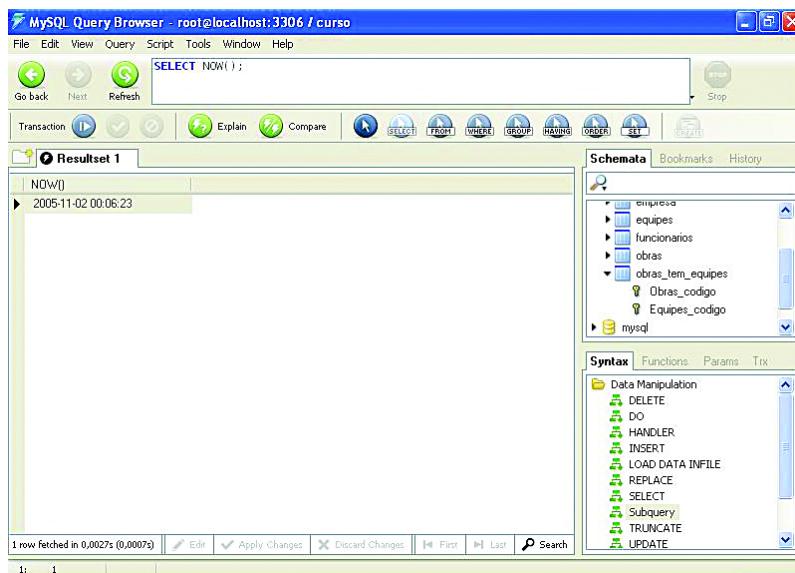
Para resolver este tipo de problema existem as funções, que são projetadas para os mais variados propósitos, tais como formatação de datas, manipulação de textos, cálculos matemáticos, dentre outras. Uma função é definida por três elementos que são o nome, os argumentos e o valor de retorno, que constituem a sua assinatura. A **Listagem 1** apresenta a sintaxe de uma função.

valor_de_retorno nome_da_função(lista_de_argumentos)

Listagem 1: Sintaxe de uma função

Uma função pode ter um ou vários argumentos, bem como pode não ter argumentos, como é o caso da função NOW, que retorna a data e hora atual do sistema, conforme ilustra a **Figura 11**, com o comando SELECT NOW().

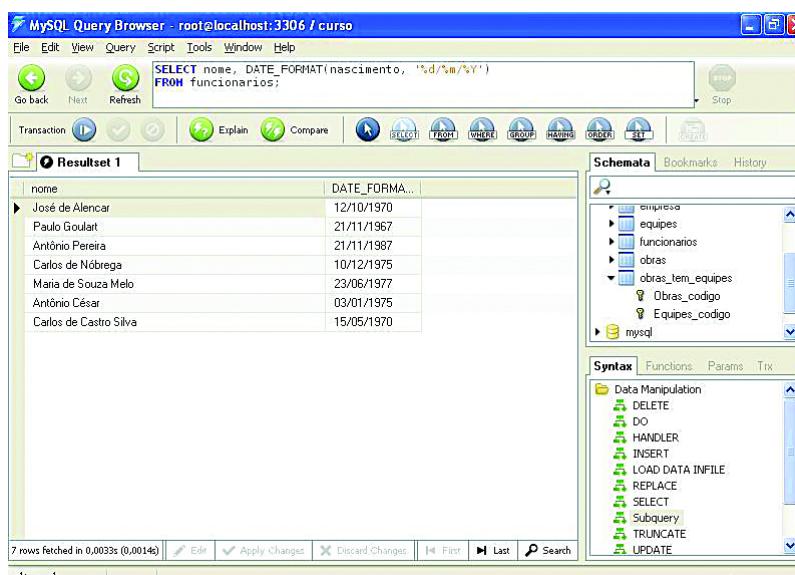
Figura 11:
Retornando a
data e hora do
sistema com a
função NOW



Uma função retorna um valor como ocorre na função NOW, ou podem existir situações em que nenhum valor é retornado, como ocorre na função LOAD_FILE, que carrega um arquivo para uma coluna da tabela.

Existe um vasto conjunto de funções no MySQL, mas fica fora do escopo deste livro abordar todas elas. Portanto, estas podem ser encontradas no manual do MySQL Query Browser. Apenas para ilustrar a questão da data, a **Figura 12** fornece um relatório com o nome e a data de nascimento de todos os funcionários, exibindo a data no formato "dd/mm/aaaa". A consulta para isto é `SELECT nome, DATE_FORMAT(nascimento, '%d/%m/%Y') FROM funcionários;`

Figura 12:
Utilizando a
função
DATE_FORMAT
para formatar
uma data



A função DATE_FORMAT recebe dois argumentos que são a data a ser transformada e o padrão de formatação. Neste caso, existem vários caracteres de formatação, tais como o %d que fornece o dia com dois dígitos, o %m para construir o mês com dois dígitos e, finalmente o %Y que retorna o ano com quatro dígitos. Outros caracteres de formatação e outras funções podem ser vistas no manual do MySQL Query Browser, encontrado no canto inferior direito desta ferramenta.

5. CONCLUSÕES

Este capítulo ilustra a leitura de dados através da linguagem SQL, e fornece uma visão geral de como construir relatórios avançados através desta linguagem. É importante ressaltar que o objetivo deste livro é introduzir os recursos desta linguagem, apresentando exemplos simples de utilização da mesma. Portanto, é possível com os conhecimentos vistos ao longo dos últimos capítulos, entender e elaborar relatórios em um banco de dados qualquer. Além disto, este capítulo forneceu a estrutura básica do SQL, possibilitando a sua utilização em situações reais.

Nos próximos capítulos serão abordados os aspectos da administração do banco de dados, discutindo as principais tarefas que devem ser realizadas pelo administrador do sistema. Além disto, serão abordados os recursos avançados de um sistema de banco de dados, tais como replicação de dados e controle de transação.

6. EXERCÍCIOS DE FIXAÇÃO

- 1- Escreva um relatório (SELECT) para listar o nome, a data de nascimento e nome da equipe para todos os funcionários que nasceram no ano de 1970?
- 2- Escreva um relatório (SELECT) para listar o nome de todas as equipes e o total de funcionários em cada uma delas.
- 3- Crie um relatório (SELECT) com o nome de todas as obras e o total de equipes que participaram de sua execução.

7. REFERÊNCIAS BIBLIOGRÁFICAS

- MySQL AB: MySQL 5.0 Reference Manual. Disponível em: <<http://www.mysql.com/documentation>>. Acesso em: 20 dez. 2005.
- MySQL AB: MySQL Query Browser. Disponível em: <<http://dev.mysql.com/doc/query-browser/en/index.html>>. Acesso em: 20 dez. 2005.
- Oliveira, Celso H. P. de (2002), SQL Curso Prático, Novatec.
- Jesus, João Batista de (2004) ANSI: SQL 89/92, Axcel Books.
- Costa, Rogério L. de C. (2004), SQL: Guia Prático, Brasport.
- Stephens, Ryan (2003), Aprenda em 24 horas SQL, 3 edição, Campus.
- Gennick, Jonathan (2004), SQL Pocket Guide, O'Reilly.
- Gulutzan, Peter; Pelzer, Trudy (1999), SQL-99 Complete, Really, CMP Books.
- Gulutzan, Peter; Pelzer, Trudy (2002), SQL Performance tuning, 1st edition, Addison-Wesley.

CAPÍTULO 9

1. INTRODUÇÃO

Neste capítulo serão estudados os aspectos relacionados à administração de um Sistema Gerenciador de Banco de Dados (SGBD), mais especificamente o MySQL, que é a ferramenta utilizada neste livro para ilustrar uma aplicação real de um SGBD.

Um SGBD constitui um serviço ou engrenagem responsável por armazenar e prover acesso às informações através da linguagem SQL. Para isto, estes sistemas mantêm as suas tabelas e bancos de dados dentro do sistema de arquivos fornecido pelo Sistema Operacional (SO). Este por sua vez é o responsável por controlar todos os recursos da máquina, bem como a execução de todos os programas instalados nela. Portanto, o SGBD cria um conjunto de arquivos e diretórios a fim de armazenar a sua estrutura de dados.

Com isto, é preciso configurar o SGBD indicando os parâmetros para utilização de discos, ou seja, quais arquivos e diretórios serão criados e onde serão colocados. Além disto, é preciso definir a quantidade de memória que o sistema utilizará, bem como definir as políticas de acessos aos dados e os usuários que terão acesso a estas informações.

A administração do banco de dados é realizada pelo Administrador de Banco de Dados, conhecido como DBA (DataBase Administrator). Esta é a pessoa responsável pela configuração do SGBD e pela manutenção dos dados, isto é, criação, alteração e exclusão de bancos de dados e tabelas de acordo com a necessidade de modificações do sistema. Além disto, o DBA deve criar os usuários para acessar os dados, bem como definir e garantir o respeito às regras de acesso aos mesmos. Por exemplo, pode-se limitar o acesso de um determinado usuário a ler somente uma tabela do sistema, enquanto outro usuário tem acesso a todas as tabelas. Isto ocorreria, por exemplo, em um sistema de uma empresa onde nem todos os usuários devem ter acesso aos dados sobre faturamento e folha de pagamento. Portanto esta restrição deve ser respeitada via controle de acessos do SGBD.

Por último, mas não menos importante vem a questão das cópias de segurança dos dados, conhecidas como backups. Estas cópias têm o objetivo de criar uma imagem dos dados que possa ser utilizada caso haja alguma perda de informações do sistema. Estas perdas podem ser acarretadas por falhas em equipamentos, tais como discos ou por usuários que realizam operações indevidas. Por exemplo, apagam por engano toda a base de dados de um sistema ou até mesmo uma única tabela. Nestes casos, o DBA deve utilizar esta cópia de segurança para restaurar os dados originais.

O objetivo deste capítulo é apresentar técnicas para a execução de tarefas administrativas do banco de dados, tais como as apresentadas anteriormente. Isto é, devem-se entender os papéis do DBA, e compreender as operações delegadas a ele. Para realizar a administração do MySQL, será utilizada uma ferramenta conhecida como MySQL Administrator, que fornece uma interface amigável para a criação de usuários, configuração de parâmetros do servidor, realização de cópias de segurança e restauração das mesmas, e finalmente, o monitoramento da execução do servidor. Para isto, será dada inicialmente uma visão geral desta ferramenta, e posteriormente serão discutidas as principais tarefas administrativas do sistema.

2. INTRODUÇÃO AO MySQL ADMINISTRATOR

O MySQL Administrator é uma ferramenta desenvolvida com o propósito de facilitar as tarefas de administrar o banco de dados MySQL. Esta ferramenta é distribuída gratuitamente e pode ser obtida gratuitamente a partir da internet no endereço www.mysql.com/downloads.

Assim como todos os clientes vistos até aqui, esta ferramenta requer uma conexão com o SGBD MySQL, que será feita utilizando o usuário root, que possui todos os privilégios do sistema. A **Figura 1** ilustra a tela do sistema que é utilizada para estabelecer esta conexão.

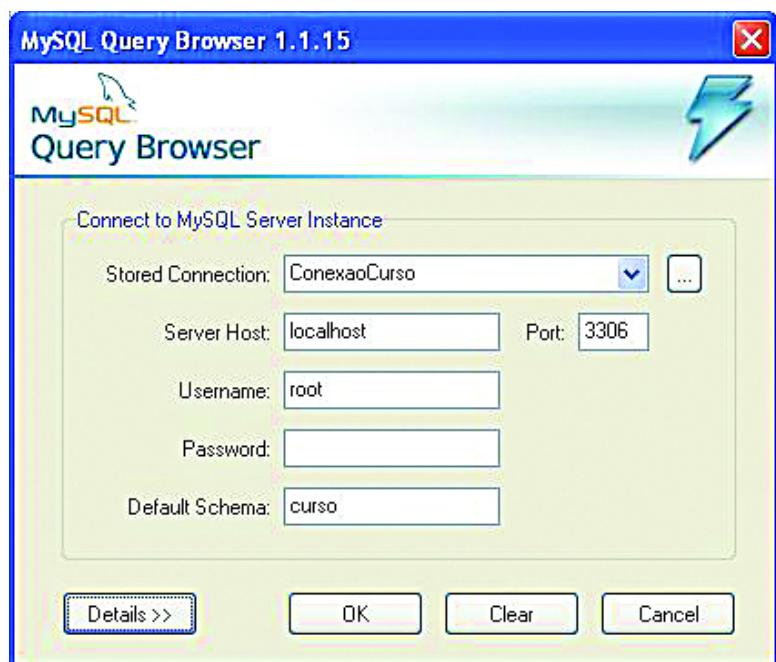


Figura 1:
Abrindo a
conexão com o
MySQL

Neste ponto é informado um nome para a conexão, o endereço do servidor MySQL, o usuário e senha. No caso, a conexão será estabelecida com um MySQL que está instalado na mesma máquina do MySQL Administrator (localhost), o usuário é o root, sem senha. Depois de informado os parâmetros deve-se acionar o botão Ok e a conexão será estabelecida, exibindo a tela ilustrada a **Figura 2**.

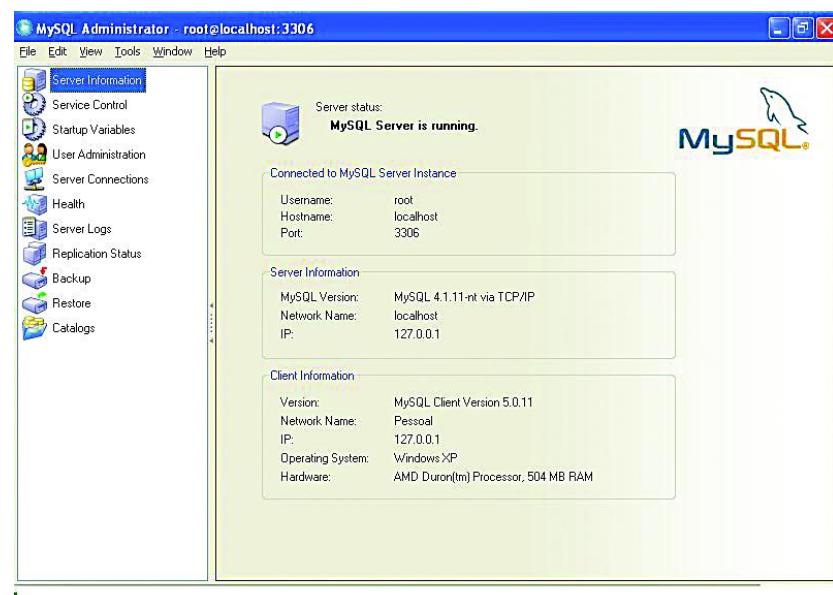


Figura 2: Tela
inicial do MySQL
Administrator

Na tela inicial são exibidas as informações sobre o servidor MySQL, que podem ser visualizadas através da opção “Server Information” (Informação do servidor), na lateral esquerda da janela. Todas as tarefas da administração do sistema podem ser desempenhadas através da utilização deste grupo de funções disponíveis à esquerda da janela. Nas seções seguintes serão discutidas cada uma das opções ilustradas na figura anterior.

3. EXPLORANDO AS OPÇÕES DA FERRAMENTA MySQL ADMINISTRATOR



Nesta seção serão discutidas as principais tarefas da administração do banco de dados, através da exploração das opções disponíveis no MySQL Administrator. O conjunto de tarefas do sistema é exibido em destaque na **Figura 3**.

As subseções a seguir discutem a utilização de cada um dos itens exibidos na figura anterior.

3.1. CONTROLE DO SERVIÇO MySQL

A opção “Service control” (Controle do serviço), exibe a situação do servidor MySQL indicando se o mesmo está em execução ou não. A **Figura 4** fornece a tela que mostra as informações contidas neste item do sistema.

Figura 3: Menu de tarefas do MySQL Administrator

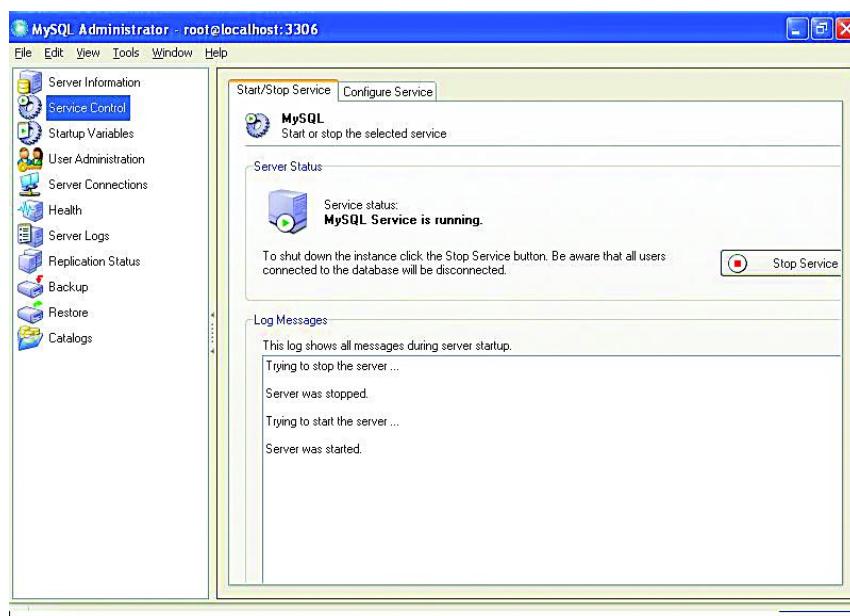


Figura 4: Explorando a opção Controle do serviço

Caso seja necessário interromper a execução do servidor MySQL, basta acionar o botão “Stop service” (Parar o serviço). Neste momento aparecerá na janela “Log messages” (Registro de mensagens), a indicação de que o servidor foi parado. Além disto, o texto do botão mudará para “Start service” (Iniciar o serviço), permitindo assim reiniciar a execução do mesmo. Para colocar o MySQL em funcionamento novamente, basta acionar este botão e a mensagem de que o servidor está no ar será exibida no registro de mensagens.

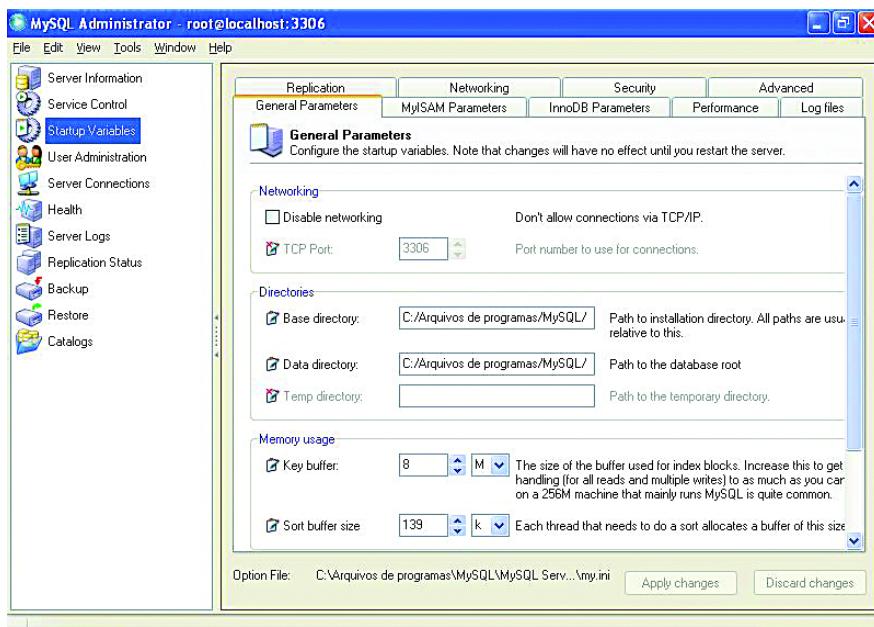
Com este recurso é possível controlar a execução do SGBD, e verificar se o mesmo se encontra em operação, e caso esteja parado, é possível determinar o motivo da interrupção via as mensagens exibidas na janela “Log messages” (Registro de mensagens).

3.2. CONFIGURANDO OS PARÂMETROS DO SERVIDOR MySQL

Existe um vasto conjunto de parâmetros relacionados ao servidor de banco de dados, responsáveis pelo funcionamento do mesmo. Por exemplo, existem configurações para controlar o número máximo de conexões simultâneas ao SGBD, ou seja, o número máximo de usuários fazendo uso do sistema. Vale lembrar que em um sistema de banco de dados podem existir várias pessoas fazendo uso dele ao mesmo tempo. Para ilustrar, basta observar um sistema bancário, onde vários clientes utilizam seus caixas eletrônicos ao mesmo tempo.

Além disto, existem variáveis que controlam a utilização de memória e disco pelo sistema, controlando assim o uso de recursos da máquina e do sistema operacional. Todas as configurações do MySQL estão disponíveis a partir do item “Startup variables” (Variáveis de inicialização), que são popularmente conhecidas como variáveis de configuração. A **Figura 5** fornece uma visão geral de todas as variáveis do sistema.

Figura 5:
Explorando as
configurações
do sistema



É possível notar que existe um conjunto considerável de parâmetros, que estão organizados de acordo com os seus propósitos. A distribuição destes parâmetros é feita em oito grupos: 1. General parameters (Parâmetros gerais), 2. Security (Segurança), 3. Networking (Configurações de rede), 4. Advanced (Avançado), 5. Performance (Desempenho), 6. Log files (Arquivos de monitoramento), 7. Replication (Replicação de dados) e 8. InnoDB, MyISAM parameters (Parâmetros para MyISAM e InnoDB). Este livro não abordará todos os conjuntos de parâmetros disponíveis, ficando restrito apenas a exemplificar como alterar uma determinada configuração. O ajuste avançado das configurações do servidor fica fora do escopo deste livro.

Portanto para alterar um parâmetro qualquer, por exemplo, “Key buffer” (Área de memória para chaves ou índices), basta digitar o novo valor desejado e então acionar o botão “Apply changes” (Confirmar alterações), para confirmar a alteração. Assim, o novo valor será assumido pelo sistema imediatamente. Caso a alteração deva ser desfeita, basta acionar o botão “Discard changes” (Descartar alterações). Desta forma podem ser alterados quaisquer parâmetros do servidor de forma a adequá-los à aplicação que o utiliza, obtendo assim um melhor desempenho da aplicação como um todo.

3.3. ADMINISTRAÇÃO DOS USUÁRIOS DO SISTEMA

No decorrer deste livro foram utilizadas diversas ferramentas para acessar o MySQL, sendo que em todos os casos foi necessário estabelecer uma conexão com o mesmo informando um usuário e senha. Um sistema de banco de dados contém mecanismos para controlar o acesso dos seus usuários, disponibilizando recursos para adicionar um novo usuário e para conceder privilégios. Desta forma, é possível definir quais ações estes usuários poderão executar dentro do SGBD.

O MySQL contém uma abordagem para controle de acessos onde é permitido controlar quais os comandos poderão ser executados, por exemplo, SELECT, INSERT, UPDATE, dentre outros. Portanto é possível criar um usuário que possa somente ler os dados do SGBD (privilegio SELECT), mas que não possa remover ou alterá-los. Além de controlar quais comandos poderão ser submetidos ao banco, pode-se ainda controlar o nível ao qual este privilégio se aplica. Por exemplo, pode-se limitar o acesso a apenas um banco, tabela ou até mesmo uma coluna. Desta forma seria possível ter um usuário que pode ler (SELECT) o nome de todos os funcionários, mas não a data de nascimento dos mesmos.

Assim, para a criação de um novo usuário deverão ser informados o seu nome, senha e os privilégios que lhe cabem. Esta tarefa é executada a partir da opção “User administration” (Administração de usuários), conforme ilustra a **Figura 6**. Na parte inferior à esquerda da janela, são exibidos todos os usuários cadastrados. Percebe-se a existência do usuário root, que foi utilizado durante todos os acessos ao servidor no decorrer deste livro.

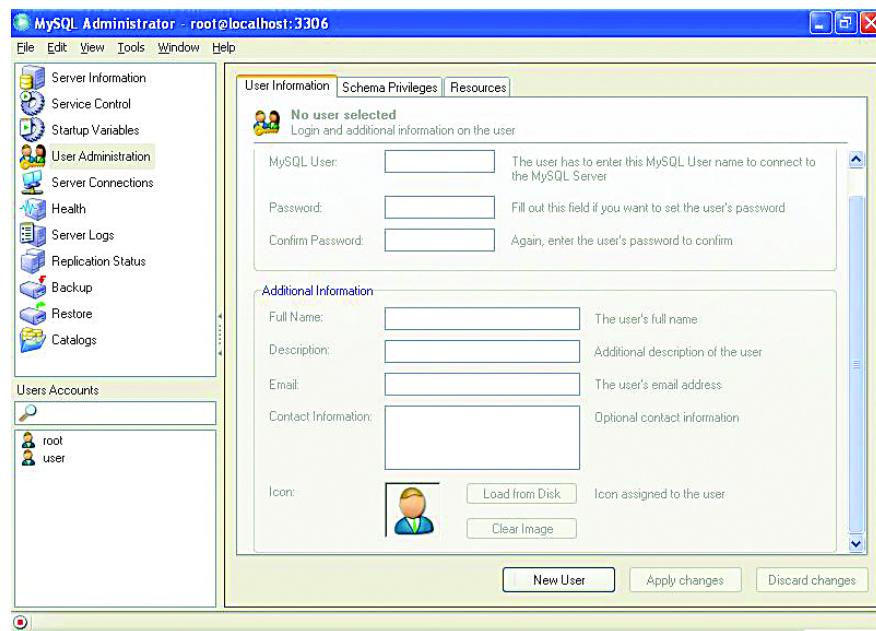


Figura 6:
Explorando a
opção de
administração
de usuários

Para criar um novo usuário basta acionar o botão “New user” (Novo usuário), e então preencher as informações do usuário, tais como nome, senha e até mesmo incluir uma imagem ou foto do mesmo. A **Figura 7** ilustra a criação do usuário “UsuarioCurso”, com a senha “abacaxi”. Percebe-se que o sistema coloca um asterisco (*) para cada letra digitada no campo senha, isto previne que pessoas próximas à máquina vejam a senha de outros usuários.

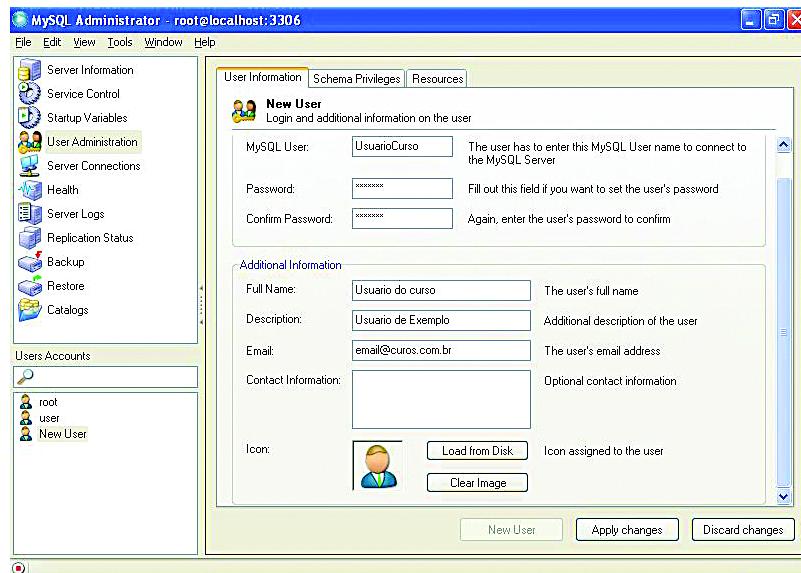


Figura 7:
Criando um
novo
usuário
chamado
UsuarioCurso

Definida as informações do usuário é necessário informar os seus privilégiosacionando a aba contendo “Schema privileges” (Privilégios do esquema). Vale ressaltar que um esquemade banco de dados é um conjunto de elementos que constituem o banco, tais como tabelas, colunas, visões, procedimentos, dentre outros. Neste momento devem-se selecionar o banco de dados que o usuário terá acesso e definir quais os comandos poderão ser executados por ele. Para atribuir um privilégio, selecione-o na janela “Available privileges” (Privilégios disponíveis), e então clique na seta para a esquerda “<”, e para excluir o privilégio marque-o na janela “Assigned privileges” (Privilégios atribuídos), e então açãone a seta para direita “>”. As setas duplas “<<” e “>>” podem ser empregadas para atribuir ou excluir todos os privilégios de uma só vez. A **Figura 8** mostra a atribuição do privilégio para executar os comandos SELECT e UPDATE em todas as tabelas do banco de dados curso.

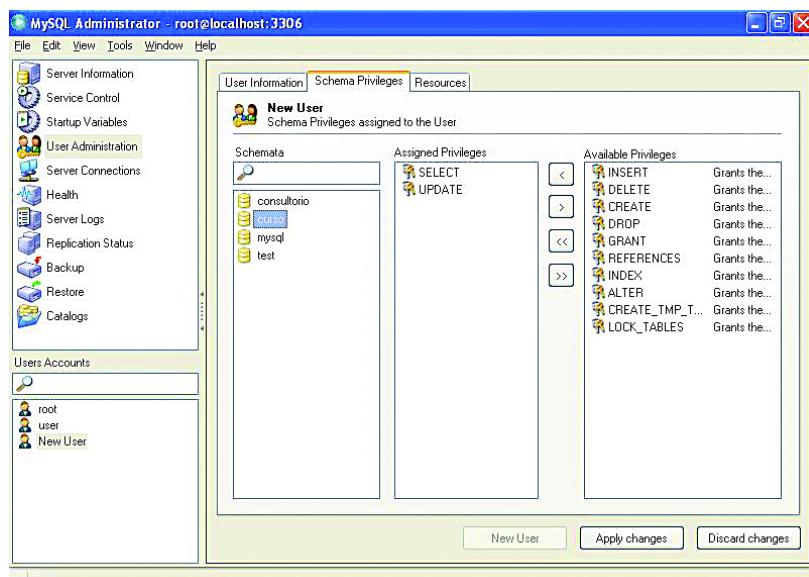


Figura 8:
Definindo o
esquema de
privilegios do
usuário
UsuarioCurso

Por último é possível controlar a utilização de recursos do sistema através do item “Resources” (Recursos), tais como o número de atualizações e conexões por hora. Concluída a entrada dos dados do usuário açãone o botão “Apply changes” (Confirmar alterações) para confirmar a criação do usuário. Para descartar a inclusão, o botão

“Discard changes” (Descartar alterações) deverá ser utilizado. A **Figura 9** ilustra o sistema após a confirmação da inserção, percebe-se que o usuário **UsuarioCurso** está mostrado na parte inferior da tela, ou seja, é um usuário válido no sistema.

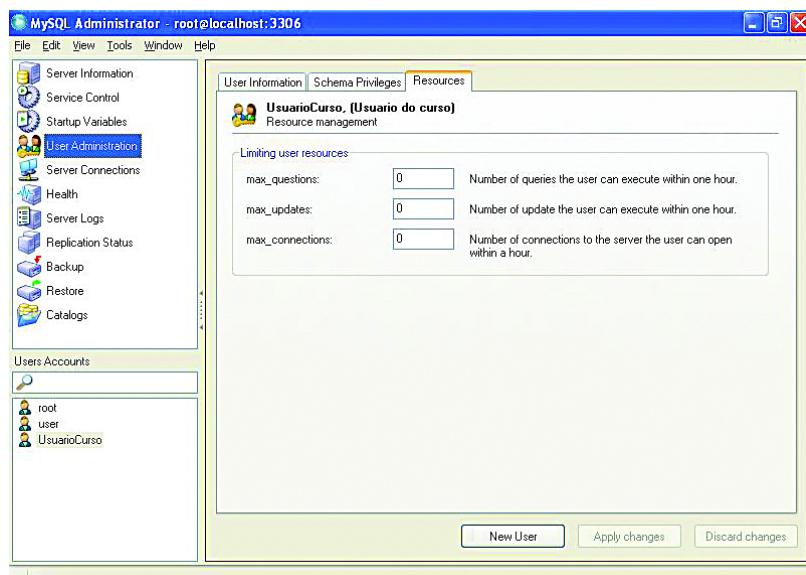


Figura 9:
Concluindo a
inserção do
usuário
UsuarioCurso

Para exibir os privilégios do usuário basta utilizar um clique duplo sobre o seu nome que está exibido na parte inferior da tela. Caso seja necessário fazer alterações neste usuário, basta utilizar a abordagem apresentada para a criação de um novo usuário, vista anteriormente.

3.4. MONITORANDO A ATIVIDADE DO SERVIDOR

O MySQL administrator apresenta quatro opções para realizar o monitoramento do servidor de banco de dados. A primeira delas é o item “Server connections” (Conexões ao servidor), que permite visualizar todos os usuários conectados ao sistema. A **Figura 10** ilustra o cenário onde apenas o usuário root está conectado ao sistema. Vale lembrar que o usuário root possui todos os privilégios do sistema, portanto é considerado o administrador.

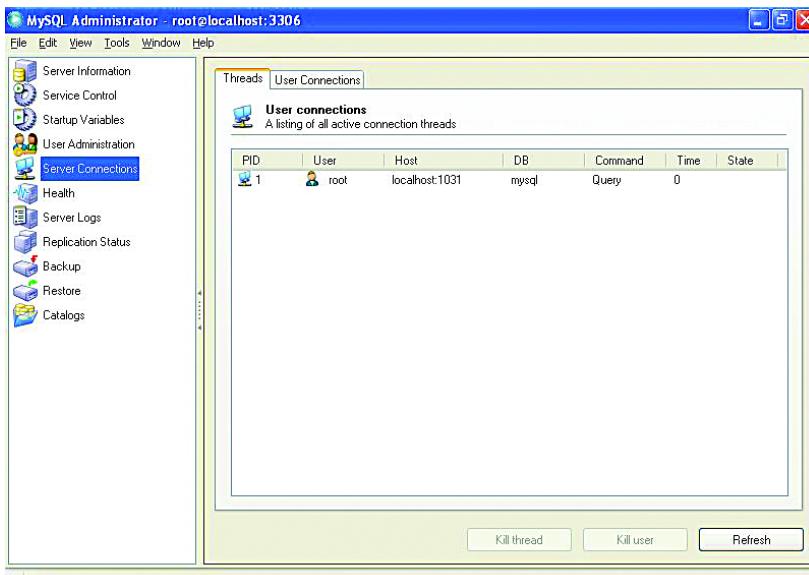


Figura 10:
Exibindo as
conexões de
usuários

Além de monitorar as conexões ou processos em execução no SGBD, a segunda opção fornecida para o monitoramento é a visualização da utilização dos recursos do sistema, tais como memória e conexões disponíveis. A **Figura 11** exibe os gráficos de utilização de memória, chamado “Memory Health” (Saúde da memória), vistos a partir da opção “Health” (Saúde). Entende-se por saúde da memória o modo como o MySQL está utilizando este recurso. Por exemplo, em uma residência o hidrômetro é responsável pelo acompanhamento do consumo de água. Neste caso, a saúde da memória fornece como está sendo consumida a memória da máquina ao logo da execução do SGBD.

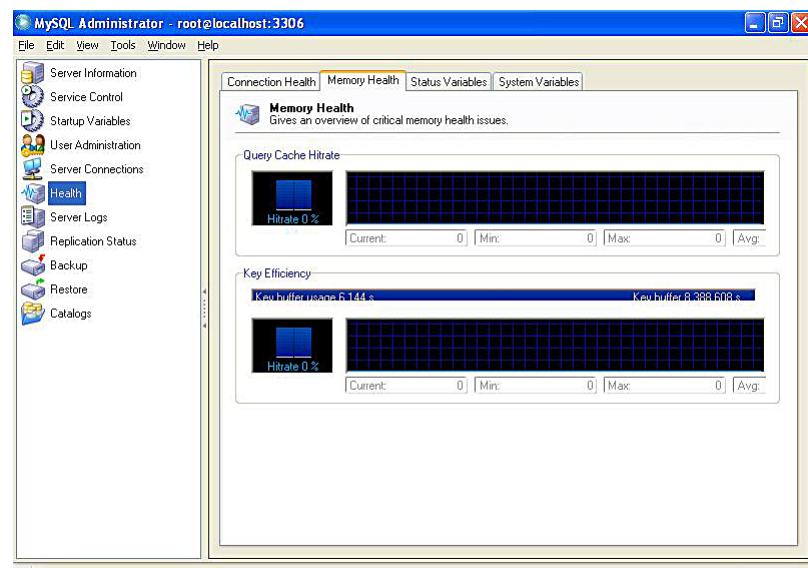


Figura 11:
Monitorando a
utilização de
memória

A terceira e última opção para monitorar a execução do MySQL é feita através dos arquivos de registros de operações, conhecidos como Log files. Estes arquivos são responsáveis por armazenar as atividades realizadas pelo servidor, tais como a hora em que este foi iniciado e parado, quais os comandos foram executados, e por qual (is) usuário (s). O conteúdo destes arquivos com os registros das atividades do banco podem ser acessados por meio da opção “Server logs” (Arquivos de monitoramento do servidor), conforme ilustra a **Figura 12**.

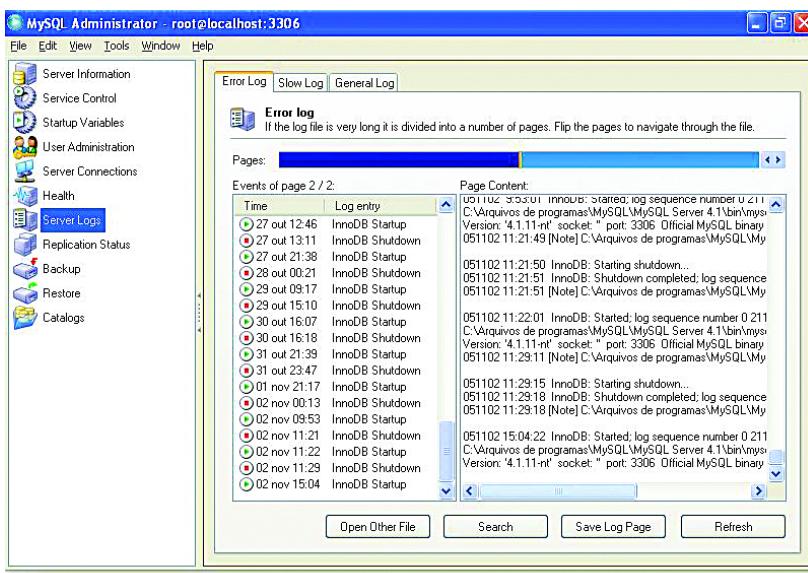


Figura 12:
Acessando os
arquivos de log
do sistema

Finalmente, esta ferramenta permite visualizar os bancos de dados do sistema, bem como manipula-los. Ou seja, é possível criar tabelas, alterar suas estruturas ou até mesmo remover tabelas e bancos, assim como era feito no DBDesigner4 e MySQL Query Browser. Para acessar a estrutura de dados do sistema ação a opção “Catalogs” (Catálogos), e então selecione o banco de dados desejado. Um catálogo nada mais é que o conjunto de todos os esquemas de banco de dados gerenciados pelo servidor MySQL. Assim, ao selecionar um banco de dados todas as tabelas que compõem este banco serão exibidas. A **Figura 13** ilustra a estrutura do banco de dados curso criado nos capítulos anteriores.

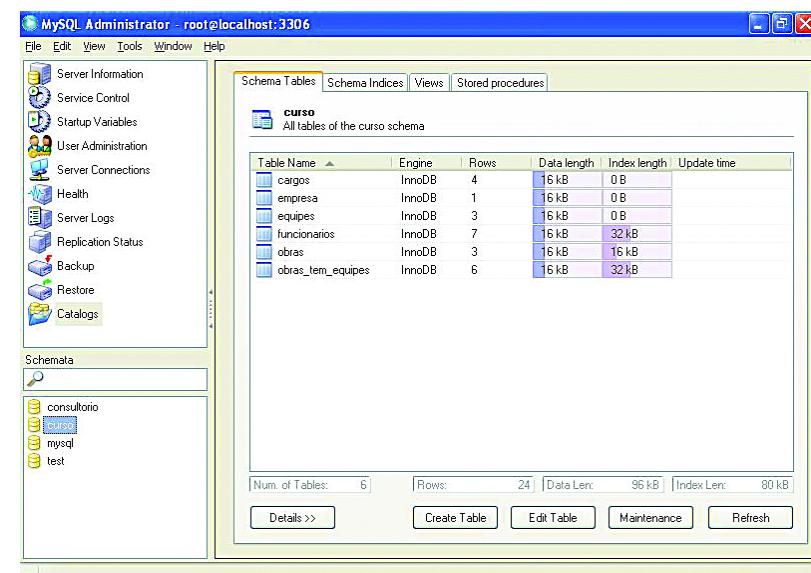


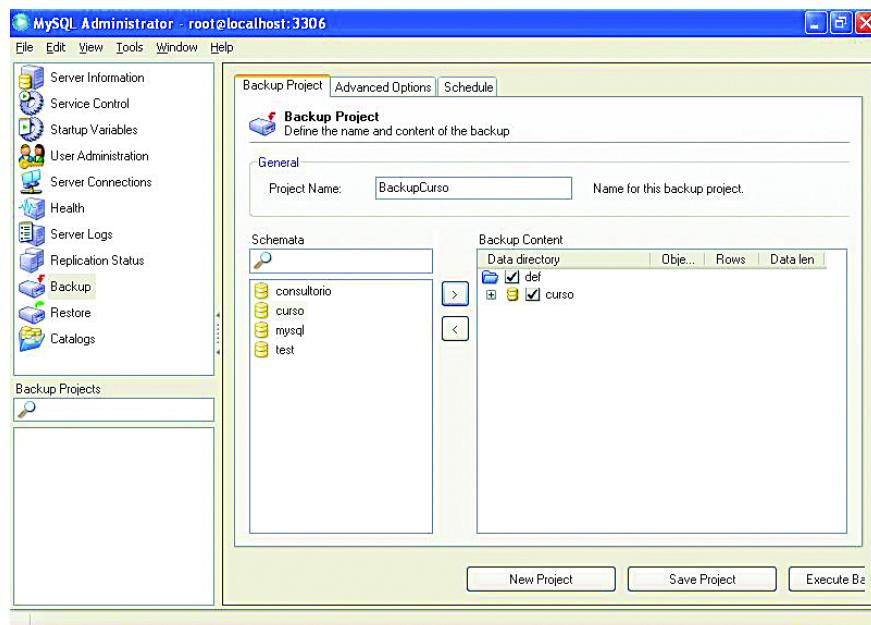
Figura 13:
Acessando a estrutura do banco de dados curso

3.5. REALIZANDO CÓPIAS DE SEGURANÇA DO BANCO DE DADOS

Uma cópia de segurança consiste em armazenar as informações do banco de dados em um local diferente de onde se encontra o SGBD. Isto é feito com o intuito de proteger os dados contra falhas de equipamentos ou de usuários. No primeiro caso, se a máquina ou o disco onde estão os dados forem danificados, toda a informação contida nele será perdida. Desta forma, tendo a cópia de segurança, conhecida como backup é possível fazer a restauração dos dados e eliminar a perda de informações. No caso de falhas de usuários, se o mesmo remover por engano uma base de dados qualquer, esta pode ser recuperada através do backup.

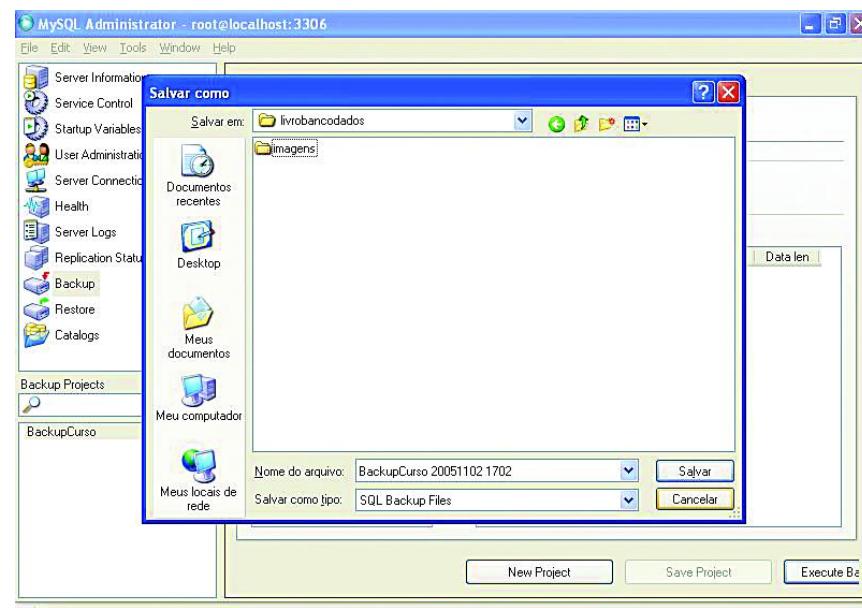
As cópias de segurança devem ser feitas periodicamente garantindo que haverá disponível uma cópia atual dos dados, minimizando as perdas de dados. No MySQL Administrator esta tarefa é realizada a partir da opção “Backup”, onde será criado um projeto para a cópia dos dados desejados. Para isto, ação-se botão “New project” (Novo projeto), definem-se o nome do projeto, selecionam-se os bancos de dados a serem copiados, e finalmente armazena o projeto açãoando-se o botão “Save project” (Salvar projeto), conforme ilustra a **Figura 14**.

Figura 14:
Criando um projeto para realizar backup do banco curso



Neste momento o projeto salvo aparecerá na parte inferior da janela e ficará disponível para utilização a qualquer momento em que se desejar realizar uma cópia do banco de dados curso. Para realizar esta operação, basta selecionar o projeto e então acionar o botão “Execute backup” (Executar cópia de segurança). Então o sistema abrirá uma janela para informar o local e o nome do arquivo de backup, conforme ilustra a **Figura 15**. Ao acionar o botão “Salvar”, a cópia será armazenada no local indicado com o nome previamente especificado.

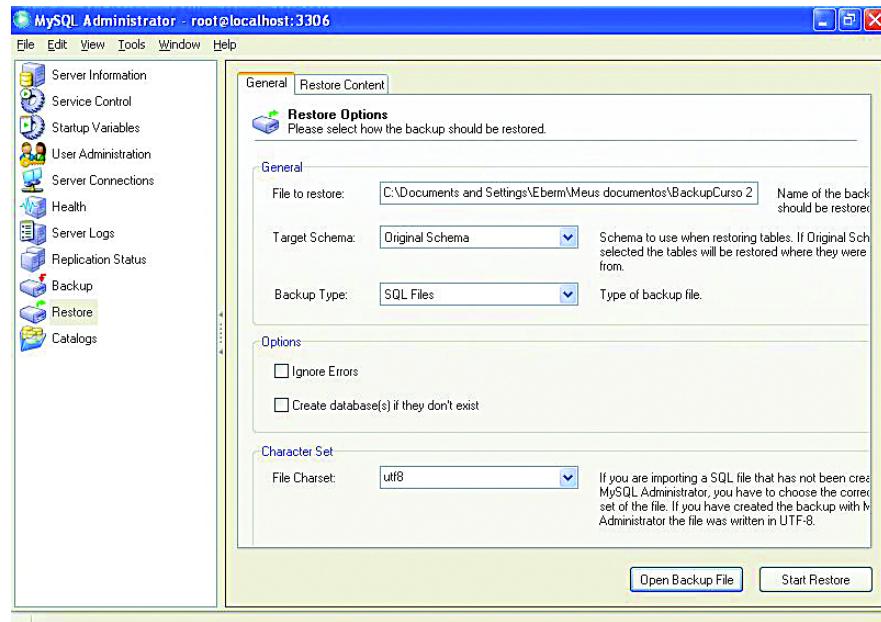
Figura 15:
Especificando o local e o arquivo de backup



Caso haja um problema com os dados, para restaurar a cópia de segurança deve-se acionar a opção “Restore” (Restaurar), e acionar o botão “Open backup file” (Abrir arquivo de backup). Neste momento abrirá uma janela para que seja selecionado o

arquivo de backup desejado, isto é, a cópia realizada no passo anterior. A **Figura 16** ilustra esta situação.

Figura 16:
Restauração
de uma cópia
de segurança



Finalmente, acionando o botão “Start restore” (Iniciar restauração), o banco de dados será restaurado para a posição em que se encontrava no momento em que o backup foi realizado. Desta forma, toda a informação perdida é devolvida para o sistema.

4. CONCLUSÕES

Este capítulo ilustrou as principais rotinas para administração de um sistema de banco de dados, salientando as principais tarefas realizadas pelo DBA. O objetivo é introduzir a ferramenta MySQL Administrator como um sistema auxiliar, bem como formar conhecimento básico sobre este procedimento para a manutenção do seu SGBD. É importante ressaltar, que o objetivo deste livro não é formar um DBA em MySQL, mas sim apresentar as rotinas pertinentes à manutenção de um sistema de banco de dados, construindo uma visão geral dos mesmo.

No capítulo seguinte serão abordados recursos avançados de um sistema de banco de dados, tais como o controle de transação e a replicação de dados. Desta forma, estará encerrada a discussão de todos os aspectos de um sistema de banco de dados real.

5. EXERCÍCIOS DE FIXAÇÃO

- 1- Descreva as 11 tarefas disponíveis no MySQL Administrator.
- 2- O que é um DBA?
- 3- Quais as tarefas atribuídas ao DBA?
- 4- Quantos grupos de configuração existem no MySQL Adminstrator? Cite-os.

- 5- Execute o MySQL Administrator e liste todos os privilégios disponíveis nesta ferramenta.
- 6- O que é um esquema de banco de dados? E um catálogo?
- 7- O que você entende por backup? Dê exemplos de sua utilização.
- 8- O que é a restauração de uma cópia de segurança? Em quais situações esta operação seria realizada?

6. REFERÊNCIAS BIBLIOGRÁFICAS

- MySQL AB: MySQL 5.0 Reference Manual. Disponível em: <<http://www.mysql.com/documentation>>. Acesso em: 20 dez. 2005.
- MySQL AB: MySQL Administrator. Disponível em: <<http://dev.mysql.com/doc/administrator/en/index.html>>. Acesso em: 20 dez. 2005.

CAPÍTULO 10

1. INTRODUÇÃO

No decorrer deste livro foram apresentados diversos conceitos acerca de um sistema de banco de dados relacional. Para a elaboração de um sistema baseado em um banco de dados relacional é preciso passar por várias etapas a fim de se determinar qual a estrutura final do mesmo. Inicialmente foram apresentadas as questões ligadas à modelagem de dados. Nesta etapa foram definidas as entidades que compõem o sistema, bem como os seus atributos e as regras de integridade que se aplicam aos dados. O produto final da etapa de modelagem é o modelo lógico do banco de dados conhecido como modelo Entidade-Relacionamento, ou somente modelo ER. Esta é uma forma de representar o banco de dados graficamente permitindo a documentação do sistema.

A segunda etapa do processo se refere à construção do banco de dados utilizando um Sistema Gerenciador de Banco de Dados (SGBD), neste livro foi adotado o MySQL como exemplo. Esta definição do banco de dados foi elaborada a partir do modelo ER, e foi empregada a ferramenta DBDesigner⁴ para esta tarefa.

Uma vez construído um banco de dados exemplo, denominado curso, foram apresentadas as técnicas de consulta às informações. Para efeito prático a linguagem SQL foi introduzida, sendo exibida a sua Linguagem de Definição de Dados (DDL), e a Linguagem para Manipulação de Dados (DML). Sobre a linguagem de consulta, esta é uma ferramenta que permite manipular as informações contidas no banco de dados. Deste modo, vários exemplos de relatórios básicos e avançados foram ilustrados, permitindo o entendimento da mesma. Vale lembrar, que o MySQL Query Browser foi introduzido possibilitando a sua utilização para a extração de dados do MySQL, sem a utilização do SQL explicitamente.

Um sistema de banco de dados é mantido pelo SGBD, e algumas tarefas têm que ser executadas periodicamente a fim de se garantir o funcionamento adequado do sistema. Para isto, foram discutidas as principais técnicas de administração de um sistema desta natureza, além de apresentar a ferramenta MySQL Administrator como um mecanismo eficiente para a execução destas operações.

Neste ponto todos os aspectos importantes relativos à utilização de um sistema de banco de dados já foram conceituados e exemplificados, possibilitando a sua aplicação em um sistema real. No entanto as aplicações reais operam em um contexto onde há uma grande complexidade nos dados, bem como nas regras de utilização dos mesmos. Além disto, em um ambiente prático, ao contrário do que ocorreu nos exemplos deste livro, o banco de dados é utilizado por diversos usuários simultaneamente. Isto introduz dificuldades e requer mecanismos para realizar o controle de concorrência sobre os dados. Exemplificando, não seria impossível que em um sistema dois usuários tentassem alterar um mesmo registro ao mesmo tempo, e neste caso, deve-se garantir a integridade da informação.

Outro problema comum são as falhas que podem ocorrer durante uma operação, isto é, pode haver uma queda de energia durante a manipulação das informações e isto poderia levar a uma inconsistência nos dados. No entanto, para contornar estas dificuldades um SGBD provê métodos para controlar o acesso simultâneo aos dados, bem como eliminar inconsistências de informações devido às falhas de sistema ou até mesmo de usuários. Este recurso é conhecido como controle de transações, e será apresentado

posteriormente neste capítulo.

Existem ainda as aplicações que exigem que o banco de dados esteja disponível o tempo todo, ou seja, 24 horas por dia e sete dias por semana. Este é o caso de um sistema bancário, que deve permitir que os seus correntistas tenham acesso às suas movimentações ou recursos financeiros a qualquer hora do dia. Estes tipos de aplicação exigem uma alta disponibilidade dos dados, e para isto existem alguns recursos presentes no SGBD para fornecer este nível de confiabilidade. Neste capítulo será conceituada e exemplificada a replicação de dados que é utilizada com o propósito de manter várias cópias dos dados, permitindo que mesmo em caso de falhas do sistema, o dado ainda esteja disponível para uso.

Finalmente, a maioria dos SGBD permite a definição de rotinas que ficam armazenadas dentro deles e que podem ser acessadas a qualquer momento pelos usuários do sistema. Estas rotinas, conhecidas como Stored Procedure (Procedimentos armazenados), são como pequenos programas que podem ser construídos de forma a realizar uma manipulação complexa sobre os dados, e garantir a obediência às regras de negócios da aplicação. Para ilustrar, na inserção de um funcionário no banco curso, é preciso verificar se o cargo informado está cadastrado na tabela cargos, caso contrário, a inserção não pode ocorrer. Uma forma de se fazer esta operação é através de uma rotina que ficaria guardada no SGBD e que poderia ser invocada pelo usuário que desejasse fazer a inclusão de um novo funcionário, garantindo assim a não violação da regra anterior. Na verdade esta é uma rotina dispensável, pois como mostrado anteriormente as restrições de chave estrangeira já garantem que não haverá um funcionário com um cargo inexistente, mas vale para ilustrar o uso das rotinas armazenadas.

Assim, este capítulo tem como objetivo discutir os recursos avançados de um sistema de banco de dados, sendo que estes são de extrema importância em sistemas que realizam tarefas corriqueiras ou comuns no dia a dia das pessoas. Todas as características anunciadas nesta seção serão conceituadas e exemplificadas no decorrer das próximas seções.

2. ENTENDENDO O CONTROLE DE TRANSAÇÕES

Um SGBD transacional é aquele que suporta uma transação ACID, ou seja, Atomicidade, Consistência, Isolamento e Durabilidade. Atomicidade consiste em executar um grupo de comandos como se fosse único, sendo que caso ocorra uma falha em um destes comandos, toda a transação será descartada levando a base de dados à situação inicial. O conceito de consistência é consequência da atomicidade, já que em caso de falhas do sistema os dados serão restaurados para a situação inicial, eliminando assim as perdas de informações. O isolamento consiste em permitir que várias transações ocorram sobre o mesmo dado e estas não interfiram umas nas outras, isto é, garante o controle de concorrência sobre os dados. Finalmente, a durabilidade está associada ao fato de que uma vez encerrada a sua transação os efeitos dela persistirão mesmo que haja uma falha do sistema.

Para ilustrar a utilização do controle de transação será considerada uma tabela que armazena os saldos de todos os correntistas de um determinado banco. Este cadastro de clientes é ilustrado pela **Tabela 1**, lembrando que esta tabela será reproduzida no MySQL, dentro do banco curso.

CÓDIGO	NOME	SALDO
1	JOSÉ DA SILVA	2.000
2	MARIA APARECIDA	4.000


```

CREATE TABLE clientes (
    codigo INTEGER NOT NULL,
    nome CHAR(30) NOT NULL,
    saldo DECIMAL(10, 2) NOT NULL,
    PRIMARY KEY(codigo)
) ENGINE=InnoDB;

INSERT INTO clientes (codigo, nome, saldo)
VALUES (1, 'José da Silva', 2000.00);

INSERT INTO clientes (codigo, nome, saldo)
VALUES (2, 'Maria Aparecida', 4000.00);

```

Tabela 1: Tabela de saldos dos correntistas de uma agência bancária

A **Figura 1**, ilustra a tabela de clientes após a execução dos comandos exibidos na **Tabela 1**, através do MySQL Query Browser.

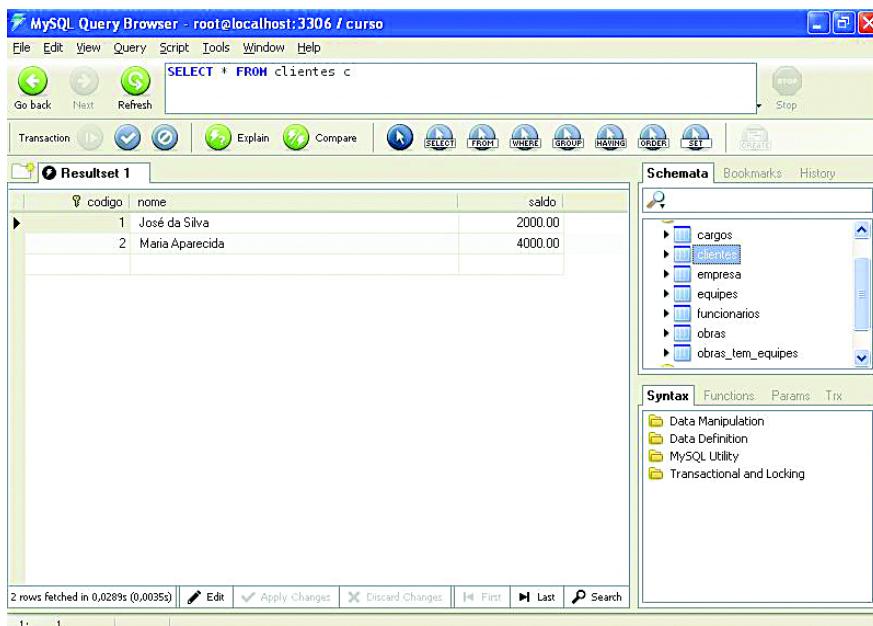


Figura 1: Tela com a tabela de saldos dos correntistas

Percebe-se que a tabela possui o código do cliente, seu nome e o seu saldo em reais, e cada registro desta tabela representa um correntista. O objetivo deste cenário é ilustrar a aplicação do controle de transação, e para isto, será realizada uma transferência de valores entre contas de clientes, assim como ocorre em um caixa eletrônico de uma agência bancária. O exemplo consiste em transferir a quantia de R\$100,00 da conta do cliente com o código 1 para a conta do cliente com o código 2.

Figura 2:
Falha em uma
transferência
de fundos
entre contas

A operação de transferência, no ponto de vista do SGBD, consiste em executar dois comandos UPDATE de forma a retirar a quantia de uma conta e depositar na outra. Ou seja, um comando UPDATE será responsável pela redução do saldo da conta do cliente 1 em R\$100,00, e o segundo comando UPDATE deve fazer um acréscimo da mesma quantia no saldo do cliente 2. Desta forma, o total em reais existentes nas duas contas não se alterará ao final da operação, já que R\$100,00 foram sacados da primeira conta e depositado na segunda.



Figura 3:
Painel para o
controle de
transação no
MySQL Query
Browser



preciso indicar para o SGBD o início e o término da sua transação. No MySQL Query Browser existem as teclas para o controle de transação, conforme ilustra a **Figura 3**.

Existem três comandos básicos que são o START TRANSACTION, COMMIT e ROLLBACK. O primeiro marca o início da transação, o segundo confirma todos os comandos executados até o momento, ou seja, é a confirmação da transação. Caso queira desfazer todos os comandos de uma transação basta enviar um ROLLBACK, que todos os comandos executados até este momento serão desfeitos automaticamente. Assim, quando ocorrer uma falha no meio da transação o SGBD não receberá a confirmação (COMMIT), e então desfará todo o processo retornando a base de dados para a situação em que se encontrava no início da operação. A **Figura 4** ilustra a mesma operação de transferência utilizando o controle de transação.

O cenário descrito anteriormente seria perfeito desde que houvesse garantia de que os dois comandos seriam executados sem problemas. Na verdade, em um sistema real não há como garantir que não haverá uma queda de energia durante a operação de transferência. Neste caso, pode ocorrer de o primeiro comando ser executado com sucesso, realizando o saque na conta cliente 1, e devido à esta falta de energia, o segundo comando não ser executado. Deste modo, o depósito da mesma quantia não seria feito na conta 2, o que levaria a base de dados à uma situação inconsistente, seja, desapareceriam R\$100,00. A **Figura 2** ilustra a execução desta operação e a situação final em que se contraram os saldos das contas. Vale destacar que para imitar a queda de energia, a ferramenta é encerrada meio da transação acionando o "X" no canto superiore-direito da tela. Desta forma, impõe-se uma interrupção abrupta no processo durante a sua execução, similar àquela que ocorreria durante a falha de energia.

Para resolver este problema deve-se entender a operação de transferência como uma única transação, composta de mais de um comando. Desta forma, a transação só seria processada se todos os comandos que a constituí forem executados com sucesso. Para isto, é

CURSO DE INTRODUÇÃO AOS BANCOS DE DADOS

Secretaria de Estado de Educação MG



Figura 4: Transferência bancária com controle de transação

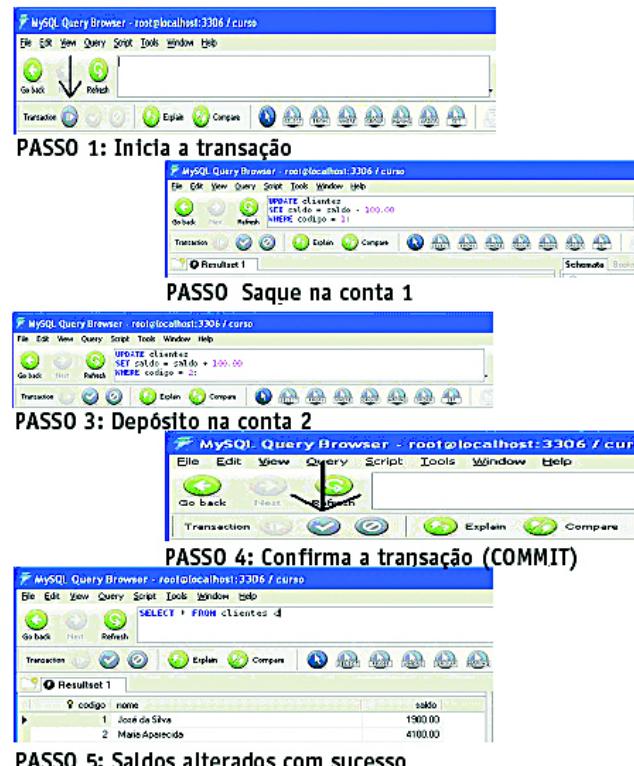


Figura 5:
Confirmando a
transferência
bancária com o
COMMIT

Neste caso, ao encerrar o MySQL Query Browser no meio da transação o primeiro comando UPDATE que havia sido executado foi desfeito pelo SGBD, devolvendo a quantia para a conta 1, eliminando assim a perda de dados ocorrida no caso onde não foi utilizado este controle.

A **Figura 5** ilustra a transação completa após a execução do comando COMMIT, neste caso todas as alterações foram aplicadas com sucesso, e a quantia foi movimentada entre contas sem perdas de informações, ou seja, de modo consistente.

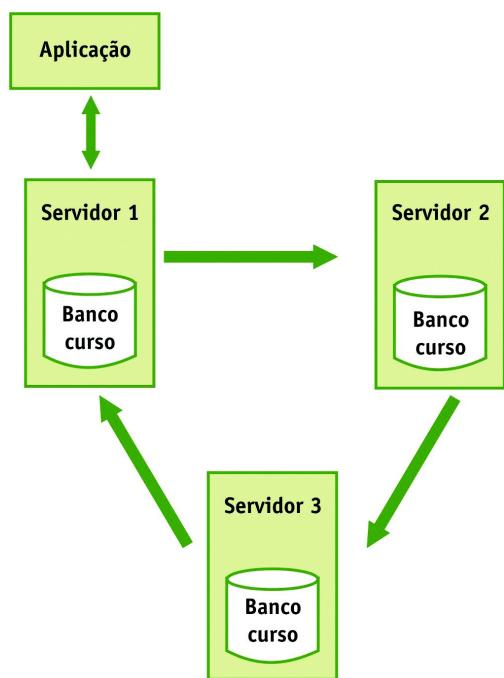
Pode ocorrer ainda de após terem sido executados os dois comandos para realizar a transferência de fundos, haja a necessidade de desistir da operação. Neste caso, utilizando o controle de transação isto é bem simples de ser feito, basta acionar o comando ROLLBACK. Desta forma tudo que tenha sido realizado até o momento será desfeito. Assim, o saque e o depósito serão desfeitos e o saldo inicial será restabelecido, da mesma forma como ocorre no exemplo da **Figura 4**.

Este é um pequeno exemplo do uso do controle de transações em um sistema real. Este é um mecanismo complexo e indispensável em determinados sistemas, visto que ele fornece uma garantia de consistência das informações contidas no banco de dados, mesmo em situações de falhas, muito corriqueiras na prática.

É importante ressaltar que existem outros aspectos a respeito do controle de transações que serão omitidos neste livro, já que o objetivo aqui é apenas fornecer os conceitos básicos acerca deste recurso.

3. UTILIZANDO A REPLICACÃO PARA OBTER ALTA DISPONIBILIDADE

Aplicações críticas exigem, em geral, que as informações estejam sempre disponíveis para o usuário, independentemente de quaisquer problemas que possam ocorrer com a máquina ou o SGBD que mantêm estes dados. Para garantir este nível de segurança é preciso manter uma redundância destas informações, permitindo assim que caso um servidor falhe, as informações contidas nele estejam disponíveis para uso em algum outro local.



Uma das técnicas mais difundidas entre os SGBD disponíveis atualmente é a replicação de dados. Este mecanismo consiste em manter uma cópia idêntica do banco de dados em diversas máquinas, sendo que qualquer alteração ocorrida em um destes servidores é propagada imediatamente para os demais participantes da replicação. A **Figura 6** fornece uma visão esquemática de uma replicação de dados onde participam três máquinas.

No cenário descrito pela figura anterior, existem três cópias idênticas dos dados e a aplicação acessa o servidor de número 1. Os demais SGBD presentes nos outros locais fazem a leitura das modificações ocorridas no primeiro servidor, garantindo que todas as máquinas estarão sincronizadas, isto é, idênticas. Esta cópia de dados é realizada constantemente, de forma a garantir a menor diferença possível de informações. Vale ressaltar que cada SGBD disponibiliza o recurso de replicação de uma forma diferente, mas o objetivo maior desta seção é apresentar a idéia básica deste artifício.

Figura 6:
Esquema de um sistema de replicação de dados

O grande benefício desta abordagem é que como existem diversas cópias idênticas dos dados, em máquinas distintas, a aplicação não fica vulnerável a um problema localizado no servidor que é acessado primariamente pela aplicação. Considerando que caso ocorra um dano no equipamento, a aplicação pode ser redirecionada para um dos demais servidores e continuar operando como se nada tivesse ocorrido ao sistema de banco de dados.

Além desta segurança um sistema de replicação permite que mais usuários acessem as informações simultaneamente, já que estes acessos podem ser distribuídos entre os vários servidores existentes. Dado que a informação contida nestas máquinas é igual, é permitida a distribuição de carga, aumentando consideravelmente o desempenho do sistema como um todo.

4. TRABALHANDO COM ROTINAS ARMAZENADAS NO SGBD

Uma rotina pode ser entendida como um conjunto de instruções que devem ser executadas com um propósito de realizar uma tarefa qualquer. No contexto de banco de dados, estas tarefas são em geral relacionadas à manipulação das informações armazenadas por ele. As instruções contidas em uma rotina são definidas em cima da linguagem SQL, e estas rotinas podem ser armazenadas no SGBD e serem utilizadas posteriormente por qualquer usuário do sistema.

O objetivo deste recurso é permitir uma padronização do acesso aos dados e garantir que a base de dados não será levada para uma situação de inconsistência. Para ilustrar este recurso, será considerada a tabela de clientes criada na seção anterior. Tomando como exemplo a transação de transferência bancária, existem algumas regras que devem ser cumpridas durante este procedimento. Por exemplo, não se deve permitir que

um cliente faça uma transferência de uma quantia superior ao saldo de sua conta.

Assim, para garantir o cumprimento deste requisito, é interessante criar uma rotina que antes de realizar a operação, verifica se há saldo suficiente. A execução do procedimento só prossegue se houver uma quantia suficiente de dinheiro na conta de onde será feito o saque. Desta forma, todo usuário do sistema que venha a realizar esta operação não precisará conhecer os detalhes que cercam esta operação, bastaria apenas invocar esta rotina que já está escrita e armazenada no SGBD. A **Figura 7** ilustra a criação e utilização de uma rotina que transfere uma quantia qualquer entre duas contas quaisquer.

A rotina chamada `transfere_quantia` recebe três argumentos como parâmetros, que são uma conta de origem (`conta_origem`), a conta de destino (`conta_destino`), e a quantia a ser transferida entre

elas (`quantia`). A rotina executa uma consulta para verificar se o saldo da conta de origem é superior à quantia a ser transferida. Caso não tenha saldo suficiente, a mensagem “Saldo insuficiente” será impressa. Caso contrário, a transferência será realizada. A **Figura 7** ilustra a chamada desta rotina exemplificando o resultado nas duas situações ilustradas anteriormente. Os comandos executados são `CALL transfere_quantia(1, 2, 300000)`, `CALL transfere_quantia(1, 2, 100)` e `SELECT * from clientes c`. Perceba que para executar um procedimento basta utilizar o comando `call` (Chamar).

Portanto, com a utilização de rotinas armazenadas é possível esconder a complexidade dos dados dos usuários, e padronizar o acesso à base de dados. O exemplo serve apenas como uma motivação para a utilização deste recurso. Vale ressaltar que existem detalhes sobre esta abordagem que foram omitidos por estar fora do escopo deste livro.

Com isto, tem-se uma visão geral do que é e qual a aplicação prática deste recurso, possibilitando um entendimento básico deste mecanismo que é amplamente utilizado em aplicações reais.

Figura 7:
Rotina para
transferência
bancária



5. CONCLUSÕES

Neste capítulo foi apresentada uma visão geral dos principais recursos avançados em um sistema de banco de dados relacional. Todos os aspectos gerais ligados à construção e utilização de um sistema de banco de dados foram discutidos ao longo deste livro com o intuito de promover o entendimento básico destes sistemas. Vale ressaltar que o objetivo deste livro é possibilitar o entendimento do que é um banco de dados, bem como tornar viável a construção de um banco de dados para resolver problemas reais

com uma baixa complexidade. Entretanto, a meta deste livro não é formar um DBA (Administrador de Banco de Dados), nem tampouco discutir todos os aspectos de um SGBD.

A partir deste ponto, todo conhecimento básico é familiar, o que torna factível o emprego desta ferramenta para modelar e representar sistemas reais dentro das necessidades específicas de cada um. Além disto, com esta base teórica é mais fácil avançar no conhecimento de banco de dados através de um estudo de aplicações específicas, bem como de outros SGBD disponíveis no mercado.

6. EXERCÍCIOS DE FIXAÇÃO

- 1- O que é uma transação ACID? Descreva o que representa cada letra desta sigla.
- 2- Qual o principal objetivo do controle de transações?
- 3- Para que servem os comandos START TRANSACTION, COMMIT e ROLLBACK?
- 4- O que é a replicação de banco de dados?
- 5- Para qual propósito é utilizada a replicação de dados?
- 6- O que é um procedimento armazenado?
- 7- Qual o propósito das stored procedures?
- 8- O que acontece ao tentar transferir uma quantia superior ao saldo da conta de origem, utilizando a procedure transfere_quantia?

7. REFERÊNCIAS BIBLIOGRÁFICAS

- MySQL AB: MySQL 5.0 Reference Manual. Disponível em: <<http://www.mysql.com/documentation>>. Acesso em: 20 dez. 2005.
- Gulutzan, Peter: MySQL 5.0 Stored Procedures. MySQL AB, march 2005. Disponível em: <<http://dev.mysql.com/tech-resources/articles/mysql-storedprocedures.html>>. Acesso em: 20 dez. 2005.
- Gulutzan, Peter: MySQL 5.0 Triggers. MySQL AB, march 2005. Disponível em: <<http://dev.mysql.com/tech-resources/articles/mysql-triggers.html>>. Acesso em: 20 dez. 2005.
- Pelzer, Trudy: MySQL 5.0 Views. MySQL AB, march 2005. Disponível em: <<http://dev.mysql.com/tech-resources/articles/mysql-views.html>>. Acesso em: 20 dez. 2005.

