UNIDADE 3

MANIPULAÇÃO DE BANCOS DE DADOS

3.1 OBJETIVO GERAL

Apresentar a transformação do modelo conceitual em tabelas e possibilitar a aprendizagem e a prática de comandos SQL para criação e manipulação de bancos de dados relacionais.

3.2 OBJETIVOS ESPECÍFICOS

Espera-se que, ao final desta unidade, você seja capaz de:

a) transformar um modelo conceitual em declarações SQL;

b) compreender comandos SQL;

c) realizar a criação e a manipulação de um banco de dados utilizando comandos SQL;

d) avaliar as influências de um modelo incompleto no projeto de um banco de dados.

3.3 PRÉ-REQUISITOS

Capacidade de obter da *internet*, instalar e utilizar um programa de modelagem. Lógica básica. Compreensão de declarações com sintaxe em língua inglesa. Será necessária a utilização de uma ferramenta que permita enviar comandos a um SGBD, o qual evidentemente também deverá estar disponível. Se você não conhece isso ou não tem à sua disposição, consulte antes (ou concomitantemente a esta unidade) o Apêndice B.

3.4 INTRODUÇÃO

O MER é usualmente utilizado para a modelagem conceitual do banco de dados, o qual, posteriormente, será implementado por meio de um gerenciador. Esse modelo poderá ser hierárquico, em rede, relacional, orientado a objetos ou objeto-relacional. Nesta disciplina, vamos nos concentrar nos relacionais, portanto, nosso modelo conceitual deverá ser transformado em um modelo relacional.

O modelo relacional é a representação das entidades por meio de relações, mais conhecidas como tabelas. Os atributos das relações são chamados de campos. Finalmente, as linhas que correspondem a um conjunto de informações podem ser chamadas de tuplas. A imagem (Figura 32), a seguir, apresenta um exemplo.

Figura 32 - Exemplo de mapeamento entre os modelos ER e Relacional

Cod_Cli	Nome
001	João
002	Pedro
003	Maria
004	Vera

Cod_Cli	Num_Conta
001	900
002	902
003	900
001	903

Numero	Saldo
900	1000
902	950
903	500

Fonte: Adaptado de Silberchatz; Korth e Sudarschan (2006, p. 8, 11-12)

A representação relacional poderá ser dada na forma **Cliente {Cod_Cli, Nome}**, na qual o atributo em destaque (Cod_Cli) corresponde ao identificador ou à chave da relação. A nomenclatura Cod_Cli é uma abreviatura para Código_Do_Cliente, o código que é utilizado na organização para sua identificação (poderia ser um número de usuário em uma biblioteca, por exemplo). A grafia depende das normas de cada organização, não há uma regra para abreviar. Mas, há uma regra para o sublinhado: somente são sublinhados os identificadores, as chaves.

Atributos compostos (complexos) são separados nos seus atributos componentes, de forma que eles sejam **atômicos**. Exemplo: se o endereço é composto por rua e número, então existirão os atributos endereço-rua e endereço-número. Atômico, no caso, significa que não é mais divisível (não dá para dividir a rua ou o número em algo menor). A divisão ocorre por questões de normalização e para melhorar a consistência: por exemplo, se usarmos o número do CEP para identificar o estado, a cidade e a rua, não será necessário redigitar (talvez com erros) a rua a cada cadastro; somente será necessário o número.

Note que não é usual utilizar acentuação no banco de dados, embora neste texto, e em algumas figuras, a acentuação esteja sendo utilizada, para melhor clareza e/ou correção ortográfica.

Para representarmos atributos multivalorados, podemos utilizar duas tabelas: uma para a entidade principal; outra para seus valores de atributos (por exemplo, uma para pessoa, outra para os filhos da pessoa).

Para uma entidade E, um atributo multivalorado M é representado por uma tabela separada EM. A tabela EM possui atributos correspondentes à chave primária de E e um atributo correspondente ao valor multivalorado M. Exemplo: o atributo multivalorado nome-filho de um empregado pode ser representado pela tabela empregado-nome-filho (empregado-CPF, filho-nome). Cada valor de um atributo multivalorado estará presente em uma linha diferente da tabela EM.

Uma vez selecionada uma forma de modelagem que atenda às necessidades do negócio que você estiver modelando, e da equipe que participa de seu desenvolvimento, *Heuser* (2009) nos oferece um quadro comparativo entre as alternativas que poderão ser seguidas na implementação do modelo de tabelas a serem utilizadas. Veja a imagem (Quadro 6).

Quadro 6 - Sugestões de implementações de tabelas a partir da Modelagem ER

Tipo de relacionamento	Regra de implementação		
	Tabela própria	Adição coluna	Fusão tabelas
Relacionamentos 1:1			
(O,1) — (O,1)	±	✔	×
(O,1) — (1,1)	×	±	✔
(1,1) — (1,1)	×	±	✔
Relacionamentos 1:n			
(O,1) — (O,n)	±	✔	×
(O,1) — (1,n)	±	✔	×
(1,1) — (O,n)	×	✔	×
(1,1) — (1,n)	×	✔	×
Relacionamentos n:n			
(O,n — (O,n)	✔	×	×
(O,n — (1,n)	✔	×	×
(1,n) — (1,n)	✔	×	×

✔ Alternativa preferida ± Pode ser usada × Não usar

Fonte: Adaptado de Heuser (2009, p. 97)

Para entender as alternativas propostas na tabela, vejamos o que significam as colunas de opções de regras de implementação:

a) tabela própria é utilizada quando, em vez de criarmos mais um campo em uma tabela existente, criamos outra tabela. Por exemplo, para relacionar quais livros estão emprestados para quais usuários, criamos uma tabela de empréstimos (própria da relação empréstimo). Esse caso é especialmente indicado para a situação de relacionamentos de muitos para muitos (n:n na representação): por exemplo, muitos livros são emprestados para muitos usuários e muitos usuários podem tomar emprestados muitos livros;

b) a adição de colunas será utilizada quando precisamos inserir um atributo novo para representar uma associação com outra entidade. Por exemplo, temos uma tabela de usuários e foi decidido que passaremos a ter uma tabela de dependentes de usuários, pois foi criada uma nova funcionalidade de empréstimo aos dependentes. Um usuário pode ter de nenhum a muitos dependentes (0,n na representação). Nesse caso, colocamos na tabela de dependentes uma nova coluna indicando qual é o identificador do usuário responsável. Essa condição é tipicamente presente quando temos a relação de um para muitos (1,n na representação) entre as entidades participantes da modelagem. Por exemplo, um estado possui muitas cidades;

c) por fim, temos o caso de fusão de tabelas, a ser mais utilizada nos casos de relacionamento de um para exatamente um (1:1 na representação). Essa situação poderá surgir naturalmente na modelagem quando forem desenhadas duas entidades com características próximas em momentos diferentes ou no caso de integração de bancos de dados e sistemas, seja entre novos e antigos, seja de fornecedores diferentes. Vamos imaginar a seguinte situação: foi modelado um usuário, o qual tem relacionamento de empréstimo com as obras da biblioteca; ele foi identificado pelo CPF e foram adicionados os atributos nome, telefone de contato e e-mail. Mas, o usuário também é um aluno do colégio ao qual a biblioteca pertence, em cujo sistema são utilizados metadados semelhantes, acrescido de alguns de interesse financeiro. Nesse caso, seria melhor unir as duas tabelas em uma só.

A partir da escolha das alternativas, passa-se à implementação propriamente dita. Em nosso caso, ela será realizada por meio de comandos da linguagem de consulta estruturada SQL.

Na descrição física do banco de dados, as relações e os relacionamentos serão transformados em tabelas, para as quais é comum a elaboração de um Dicionário de Dados (DD). O objetivo do DD é o levantamento de características dos atributos, que serão transformados em campos, e sua documentação para consulta posterior e auxílio à elaboração de declarações de consulta, ou seja, os comandos em linguagem SQL, os quais serão vistos na sequência. A imagem (Quadro 7) mostra um exemplo de dicionário de dados.

No quadro em questão, temos a descrição detalhada dos atributos correspondentes a uma modelagem de cliente de uma organização qualquer (exibido parcialmente). Podemos notar, para cada campo exibido (Nome, Saldo), uma breve descrição de sua finalidade, o tipo de dados que utilizará, o tamanho, se for o caso, os valores mínimo e máximo, se

aplicáveis, se é um identifcador (chave), se pode ficar em branco ou não (pode ser nulo?), se será armazenado na tabela (A) ou se será calculado a partir de dados existentes (seria um C, mas não aparece no exemplo), qual é o proprietário dos dados (no caso representado por MKT, de departamento de *marketing*), qual transação comercial originou o dado (no caso, uma venda) e outras observações. O Dicionário de Dados, para ser útil à modelagem, deverá conter minimamente informações de nome do campo, tipo de dado, se é ou não chave, se aceita ou não nulos e se é ou não chave; outras características dependem dos modelos de documentação adotados em cada organização.

Quadro 7 - Exemplo de dicionário de dados

Campo	Descrição	Tipo de dado	Tamanho	Valor mínimo	Valor máximo	É identificador?	Pode ser nulo?	Armazenado/Gerado?	Proprietário	Origem	Observações
Nome	Nome do cliente	Texto	50	Não possui	Não possui	Não	Não	A	MKT	Vendas	...
Saldo	Saldo da conta	Valor		Zero	S/ limite	Não	Não	A	MKT	Vendas	Calcu_ lado cfme doc 2
...											

Fonte: Produção do próprio autor

O DD deverá ser criado para cada tabela e fará parte da documentação do sistema.

De onde vêm as informações? Do levantamento de requisitos, da análise de documentos e processos, das entrevistas com os usuários e outras formas comentadas na Unidade 1.

Qual é a sua utilidade? Servirá para que as definições dos tipos de dados e suas *constraints* sejam precisas e adequadas. Ademais, contém informações de segurança: quem é o proprietário dos dados e qual o local e/ou operação que o gerou. Também pode remeter a uma fórmula ou a um outro documento que especifica como deve ser tratado, calculado, gerado etc.

No exemplo mostrado de Dicionário de Dados, a organização optou por inserir informações de proprietário e de transação de origem na documentação. Isso permitirá que mais tarde sejam atribuídas autorizaçãoes para a inserção de dados na tabela somente ao pessoal do marketing, assim como seja controlado que somente poderá ocorrer uma inserção durante uma transação de vendas (eventualmente uma atualização).

Quaisquer tentativas de outros usuários inserirem dados na tabela ou sua modificação em outras transações, poderiam gerar um alerta de alteração indevida, ou poderiam ser, simplesmente, ignoradas.

3.5 INTRODUÇÃO À SQL

A SQL foi criada para a manipulação dos elementos do banco de dados. Suas características mais importantes são:

a) é padronizada para os bancos de dados relacionais;

b) cada comando é uma descrição do que se deseja obter;

c) quem executa o comando é o gerenciador do banco de dados.

Para nossos estudos, podemos dividir a SQL em três partes: *Data Definition Language* ou Linguagem de Definição de Dados (DDL), *Data Manipulation Language* (DML) e *Data Control Language* (DCL).

A DDL é composta pelo conjunto de comandos que são utilizados para a definição do esquema. Por exemplo:

```
CREATE TABLE pessoa {
        nome    char(40),
        CPF     integer);
}
```

Essa sequência de comandos deverá ser enviada ao SGBD por meio de programação ou digitada na interface de consulta ou linha de comando do aplicativo que faz acesso ao banco. Na sequência, veremos como fazê-lo passo a passo.

No exemplo (veremos detalhadamente a sintaxe dos comandos mais adiante), será criada uma tabela identificada como **pessoa**, composta por dois campos: nome e CPF. O primeiro campo aceitará até 40 dados do tipo caractere, e o segundo aceitará um número inteiro. O ; (ponto e vírgula) ao final é utilizado para terminar a declaração, faz parte da linguagem e é obrigatório.

A DML corresponde ao conjunto de comandos utilizados para acessar e manipular os dados organizados pelo modelo de dados apropriado.

Também é conhecida como ***query language***, linguagem de consulta (para um banco de dados, inserir um conteúdo ou alterá-lo é uma **consulta**, *query*, em inglês). Por exemplo, o comando ***SELECT* nome *FROM* pessoa**; retorna todos os valores do campo (ou atributo) nome da tabela (ou relação) pessoa; ou seja, retorna uma LISTA com os nomes das pessoas cadastradas na tabela pessoa. Note que foi ressaltada a palavra lista, pois esta é a forma usual de retorno do comando ***SELECT*** e dos resultados de consultas executados por meio de comandos SQL.

Por fim, a DCL agrega o conjunto de comandos utilizados para garantir ou bloquear acesso a dados ou conjuntos de dados no banco. Portanto, permite a administração da segurança de acesso aos dados.

Por exemplo, o comando ***REVOKE DELETE ON* pessoa *TO* simao**; retira (*REVOKE*) do usuário **simao** a possibilidade de apagar (*DELETE*) conteúdos na (*ON*) tabela **pessoa**.

Nesta disciplina, vamos trabalhar com alguns comandos da DDL (criação de bancos de dados e de tabelas) e da DML (inserir, alterar, consultar e apagar dados); não entraremos em detalhes da DCL.

Explicativo

Em bancos de dados, chamamos a execução de comandos de CONSULTA. Lembre-se de que estamos trabalhando com SQL, abreviatura de linguagem de **consulta** estruturada (*structured QUERY language*, sendo *QUERY* = consulta). Assim, o que faremos na prática é realizar consultas ao sistema gerenciador de bancos de dados, mesmo quando solicitando a execução de comandos para a criação de objetos, inserção, modificação ou apagamento de dados.

3.6 PRÁTICA DE DECLARAÇÕES SQL

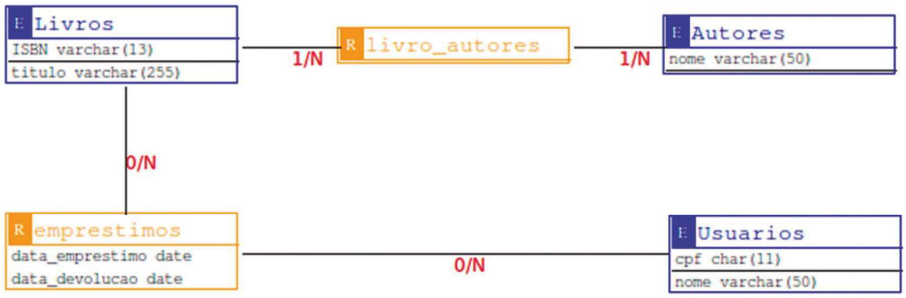
Existem muitas ferramentas para a criação e a manipulação de bancos de dados no mercado, pagas e gratuitas. Grande parte das ferramentas disponíveis são capazes de realizar a modelagem do banco, conectarem-se ao SGBD e realizarem a criação daquilo que foi criado. Há interfaces disponíveis para a inserção de dados que se assemelham a planilhas, facilitando a utilização.

Neste ponto, daremos preferência às operações manuais, de forma a introduzirmos e explicarmos os comandos necessários e permitir que você perceba a importância da sintaxe e da estrutura das declarações.

A ferramenta escolhida para este momento foi a *MySQL Workbench*, disponível para ambientes *Linux* e *MS-Windows*. Consulte o Apêndice B para verificar informações quanto à instalação e rápido guia de uso desse produto se quiser utilizá-lo.

Vamos utilizar como referência o diagrama exibido na imagem (Figura 33), a seguir. O diagrama é hipotético, está sendo apresentado aqui com finalidade didática e, em um caso real, teria sido criado a partir de uma análise de requisitos, atendendo, possivelmente, à seguinte situação: um livro pode ter de no mínimo um até vários autores, e vários autores cadastrados podem ter autorado no mínimo um até vários livros. Após existir, esse livro poderia ter sido emprestado de nenhum a muitos usuários; e, quando cadastrados, os usuários poderiam ter tomado emprestado de nenhum a muitos livros.

Figura 33 - Exemplo de MER com aspectos físicos



Fonte: Produção do próprio autor[13]

Observe que temos, no diagrama, três entidades representadas por E (Livros, Autores e Usuários) e dois relacionamentos entre elas representadas por R (empréstimos e livro_autores). É uma representação MER, porém, apresentando alguns aspectos físicos, como os tipos de dados envolvidos.

Não estamos, propositalmente, utilizando a melhor modelagem. O objetivo é que surjam novas necessidades ao longo da prática de comandos de forma a podermos ligar os aspectos práticos com a importância da boa modelagem.

Para implementar a representação da imagem (Figura 34), a seguir serão demonstrados comandos básicos para a criação e a manipulação de bancos de dados por meio de SQL.

Os exemplos foram executados em *MySQL Workbench*, conectado a um SGBD *MySQL* 5.6, em ambiente *Linux* e foram igualmente testados em ambiente *MS-Windows*. Os comandos em si funcionarão em outros SGBDs, mas as cópias de tela, obviamente, são específicas das ferramentas utilizadas.

Atenção

Considere todos os comandos do item 3.4.1 como uma grande atividade.

Você deverá executar os comandos dos exemplos em seu ambiente, de forma a treiná-los e verificar os resultados, para poder acompanhar as reflexões. Compare o resultado que você obteve com a cópia de tela que sucede o comando e, no caso de discrepância, revise cuidadosamente a linha digitada, prestando atenção, em especial, ao ponto e vírgula, às vírgulas, aspas e palavras em língua inglesa.

[13] Elaborado pelo autor na ferramenta *Ferret*.

3.6.1 Criar o banco de dados

Lembre-se: para a execução dos comandos, o servidor deve estar iniciado, e o cliente conectado (veja o apêndice B)!

Primeiramente, criaremos e colocaremos em uso um banco de dados chamado de **pratica1**.

Para isso, utilizamos os comandos *CREATE DATABASE*, que cria o banco de dados, e USE, que instrui o gerenciador a executar os próximos comandos no banco de dados informado.

A finalidade do uso do comando create database é a criação do banco de dados dentro do gerenciador. É o primeiro passo para a elaboração de um banco de dados funcional; o passo seguinte corresponderá à criação de seu metaconteúdo (as tabelas, suas descrições e relacionamentos) e, finalmente, da inclusão de seu conteúdo: os dados. Veremos tais passos na sequência de nossos estudos.

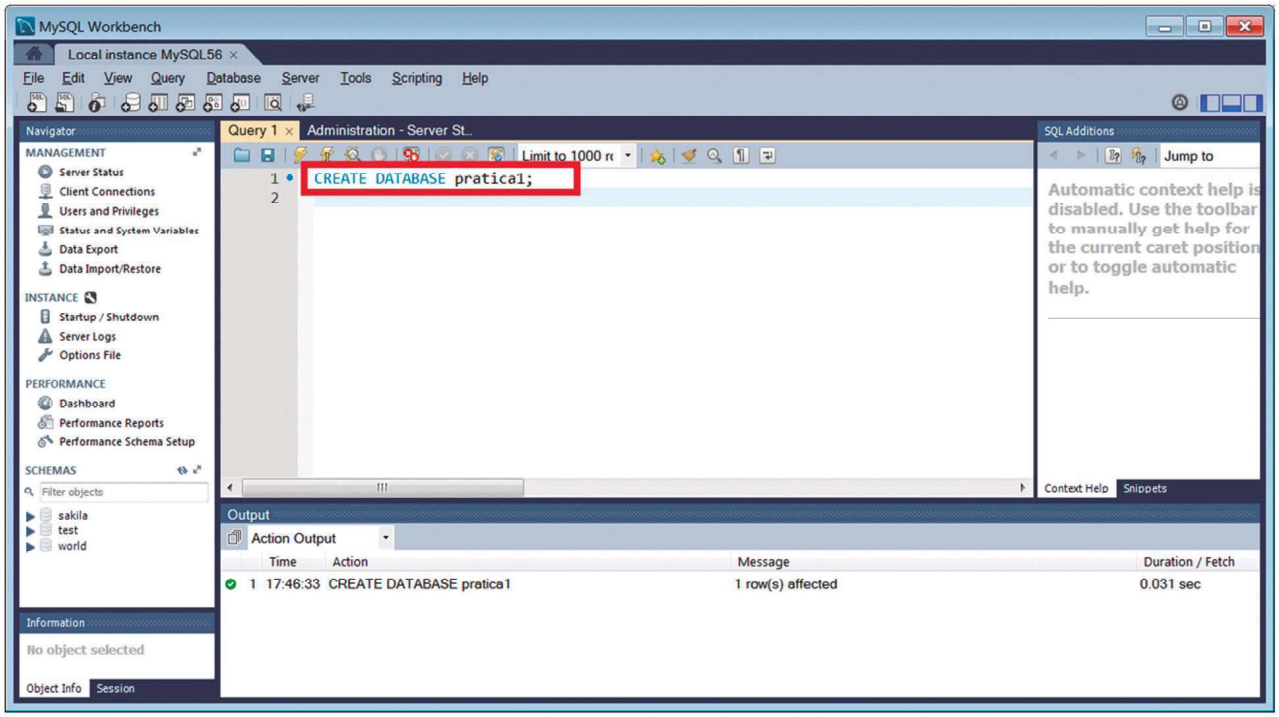
Cada SGBD possui algumas características próprias, as quais devem ser pesquisadas nos ambientes de auxílio ou na *internet*, de forma a obterem-se opções para esses comandos.

Vamos ver o passo a passo?

Passo 1: Abra o SGBD escolhido por você. Digite o comando a seguir e veja na imagem logo após um exemplo da tela com a ação.

CREATE DATABASE pratica1;

Figura 34 - Exemplo de criação de BD



Fonte: Produção do próprio autor[14]

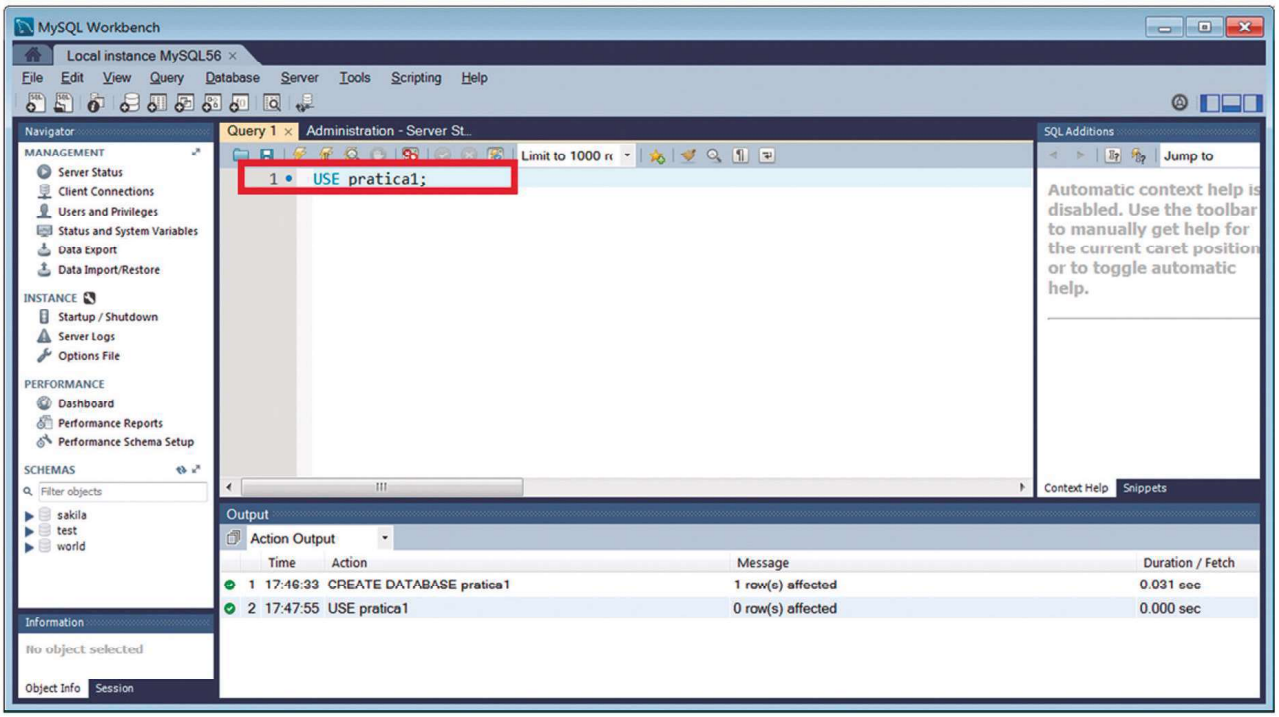
[14] Elaborado pelo autor na ferramenta *MySQL*.

Passo 2: Digite o comando a seguir e veja na imagem logo após um exemplo da tela com a ação. Observe na imagem que o comando anterior ficou registrado no painel *Action output.*

ATENÇÃO: TODOS OS COMANDOS DEVERÃO SER DIGITADOS NA ABA DE EDIÇÃO DE COMANDOS (identificada com ***Query 1*** na figura que segue).

USE pratica1;

Figura 35 - Exemplo do comando *USE* para indicar qual BD será usado para os próximos comandos



Fonte: Produção do próprio autor

Você deverá notar que solicitou a criação e o uso de um banco de dados, e todas as tarefas necessárias para isso foram efetuadas pelo SGBD. É importante ressaltar que, do ponto de vista de segurança, o usuário utilizado por você para acesso ao SGBD deverá ter permissão para criação de bancos de dados, caso contrário, o comando não será executado. Em caso de problemas, consulte a equipe de TI responsável pelo atendimento do laboratório ou seus tutores, presencialmente ou a distância.

Explicativo

Nesta disciplina, os exemplos foram executados no *MySQL*, mas o SQL é padronizado e deve funcionar em quaisquer SGBD-R, eventualmente com pequenos ajustes.

Os comandos foram digitados em maiúsculo somente para destaque, mas podem ser digitados em minúsculo sem nenhum problema.

Dica: evite utilizar acentuação nos nomes dos objetos criados (banco de dados, tabelas, campos).

A sintaxe (para *MySQL*) do comando CREATE DATABASE é a declaração do comando seguida do nome do banco de dados. Para o comando USE, igualmente é o nome do comando seguido do nome do banco de dados ao qual queremos nos conectar (se tivermos autorização para isso, como já observado).

Como a linha de comandos poderá ser maior do que a linha a ser exibida na tela, o SGBD somente vai considerar que o comando terminou quando encontrar a informação de que ele foi encerrado, quando encontrar o terminador do comando. No SQL os comandos são encerrados por meio do uso do sinal **;** (ponto e vírgula) ao final da lista de comandos a executar.

Explicativo

As ferramentas de modelagem costumam gerar o código SQL para os modelos criados. Dessa forma, se você criar um modelo conceitual na ferramenta, poderá exportar o código SQL para um arquivo e abri-lo no ambiente de trabalho. Algumas ferramentas de modelagem executam diretamente o modelo criado em um SGBD ao qual estiverem conectadas.

3.6.2 Criar as tabelas necessárias para implementar o modelo

Depois da criação do banco de dados, os comandos que seguem criarão as tabelas que receberão os dados de livros, usuários e autores de livros (lembre-se de que estamos trabalhando com o modelo representado na Figura 34).

Note que poderíamos ter criado uma tabela de pessoas, as quais poderiam ser usuários ou autores. Da forma que fizemos, se uma pessoa for ao mesmo tempo autora de um livro e usuária da biblioteca, haverá duplicidade de cadastro. Nesse caso, o que mudou foi somente o papel desempenhado pela pessoa, não a pessoa em si. Mas, por enquanto, deixaremos assim.

Observamos novamente a necessidade de autorização para a criação de tabelas por parte do usuário que você está utilizando na comunicação com o SGBD. As seguintes declarações criarão as tabelas necessárias ao exemplo:

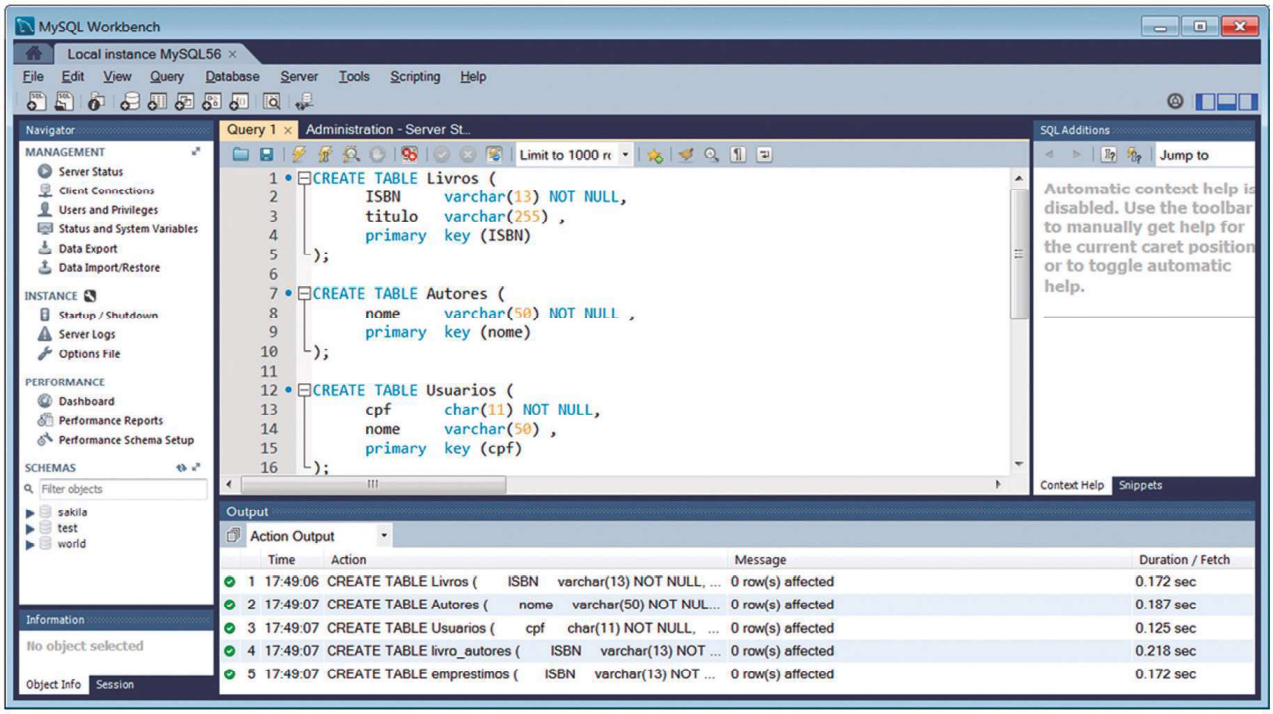
As tabelas a serem criadas, a seguir, correspondem ao modelo apresentado na imagem (Figura 34), que é um exemplo hipotético de modelagem de biblioteca utilizado nesta disciplina com finalidade didática. Serão utilizadas para a prática dos comandos de manipulação de dados descritos na sequência.

Passo 3: Digite os comandos a seguir e veja na imagem logo após um exemplo da tela com a ação realizada.

ATENÇÃO: TODOS OS COMANDOS DEVERÃO SER DIGITADOS NA ABA DE EDIÇÃO DE COMANDOS (identificada com *Query 1* na figura que segue).

```
CREATE TABLE Livros (
    ISBN    varchar(13) NOT NULL,
    titulo  varchar(255),
    primary  key (ISBN)
);

CREATE TABLE Autores (
    nome    varchar(50) NOT NULL,
    primary  key (nome)
);

CREATE TABLE Usuarios (
    CPF     char(11) NOT NULL,
    nome    varchar(50),
    primary  key (CPF)
);

CREATE TABLE livro_autores (
    ISBN    varchar(13) NOT NULL,
    nome    varchar(50) NOT NULL,
    primary  key (ISBN,nome)
);

CREATE TABLE emprestimos (
    ISBN    varchar(13) NOT NULL,
    CPF     char(11) NOT NULL,
    data_emprestimo    date,
    data_devolucao     date
);
```

Figura 36 - Exemplo do comando *CREATE TABLE* para criar as tabelas no BD em uso



Fonte: Produção do próprio autor

A Sintaxe do comando para a criação das tabelas é CREATE TABLE seguido do nome da tabela; após o nome, haverá abertura de parênteses para a criação dos campos que compõem a tabela seguida do fechamento dos parênteses e do sinal ; que encerra o comando.

Dentro da estrutura das tabelas, temos a declaração de seus campos, os quais são declarados com a sequência mínima de nome_do_campo seguido de tipo_de_dados (e tamanho, quando for o caso). Podem ocorrer mais declarações, a depender da necessidade e do tipo de SGBD adotado, por exemplo, as declarações NOT NULL em alguns casos. Note que essa declaração implementa uma regra de integridade, garantindo que certos dados não sejam nulos, ou seja, que os mesmos sejam de cadastramento obrigatório. Os campos são separados por **,** (vírgula).

Não é necessário digitar cada declaração em uma linha separada e tabular de forma a alinhar as declarações. Isso só é realizado para fins didáticos e de melhoria da visualização. Se você digitar tudo em uma única linha, o efeito será o mesmo, desde que a sintaxe esteja correta.

As tabelas também tiveram a atribuição de chaves primárias. Para isso, foi incluída a informação *primary key* em sua declaração. Relembrando, o objetivo é a criação de *constraints* para garantir a integridade de nosso banco de dados.

Algumas chaves são simples, contendo somente um campo, mas também podemos formar chaves compostas. Por exemplo, poderíamos criar uma chave formada pelo nome da pessoa, o nome da mãe e a data de nascimento. O objetivo da criação de chaves,

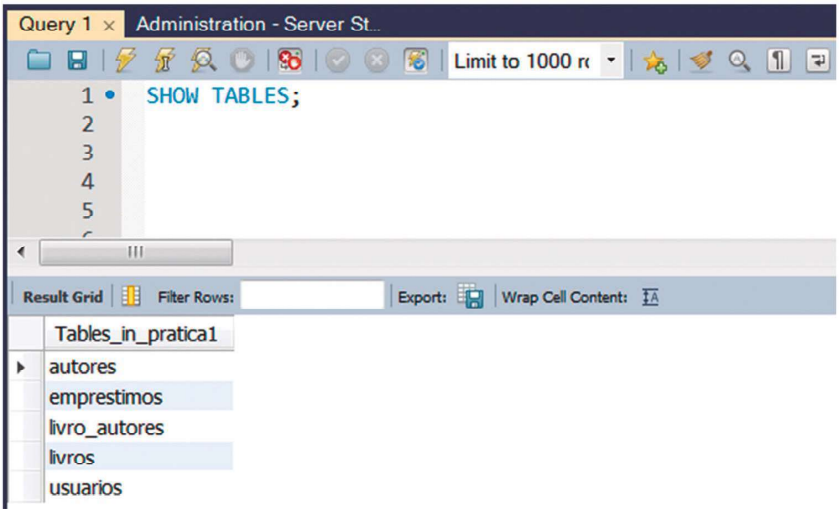
relembrando, é o de garantir unicidade, identificação única e cadastro único de certos dados (isso deve ser definido no modelo conceitual, conforme já estudado).

Quer ver as tabelas de seu banco de dados? Execute: SHOW TABLES;

Passo 4: Digite o comando a seguir para exibir as tabelas criadas por você.

SHOW TABLES;

Figura 37 - Exemplo do comando *SHOW TABLES* para mostrar as tabelas no BD em uso



Fonte: Produção do próprio autor

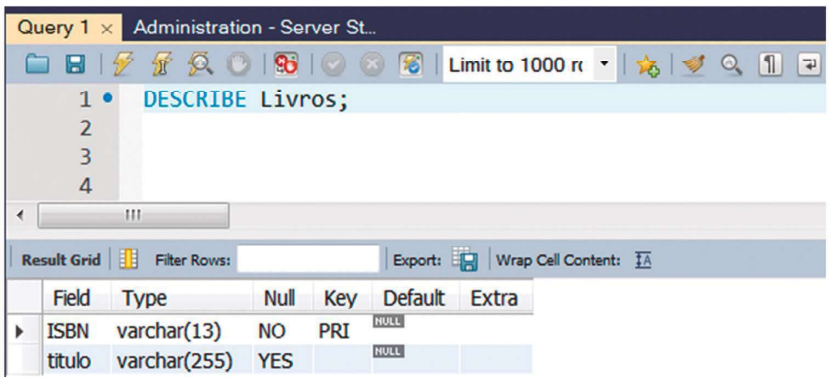
O comando *SHOW* é o comando para mostrar ou exibir; nesse caso, as tabelas.

Quer ver a estrutura de uma tabela? Execute: *DESC Livros*;

Passo 5: Digite o comando a seguir para exibir as tabelas criadas por você.

DESC Livros; (você também pode usar a forma completa, *DESCRIBE Livros*;)

Figura 38 - Exemplo do comando *DESC* para mostrar a estrututra de uma tabela no BD em uso



Fonte: Produção do próprio autor

O comando *DESC* é a abreviatura de *describe* (descreva). Ao ser executado seguido do nome da tabela desejada, ele mostra sua estrutura. Note na apresentação a informação de que o campo ISBN é chave primária (*Key* = PRI) e que o campo titulo pode ser nulo (*Null* = Yes).

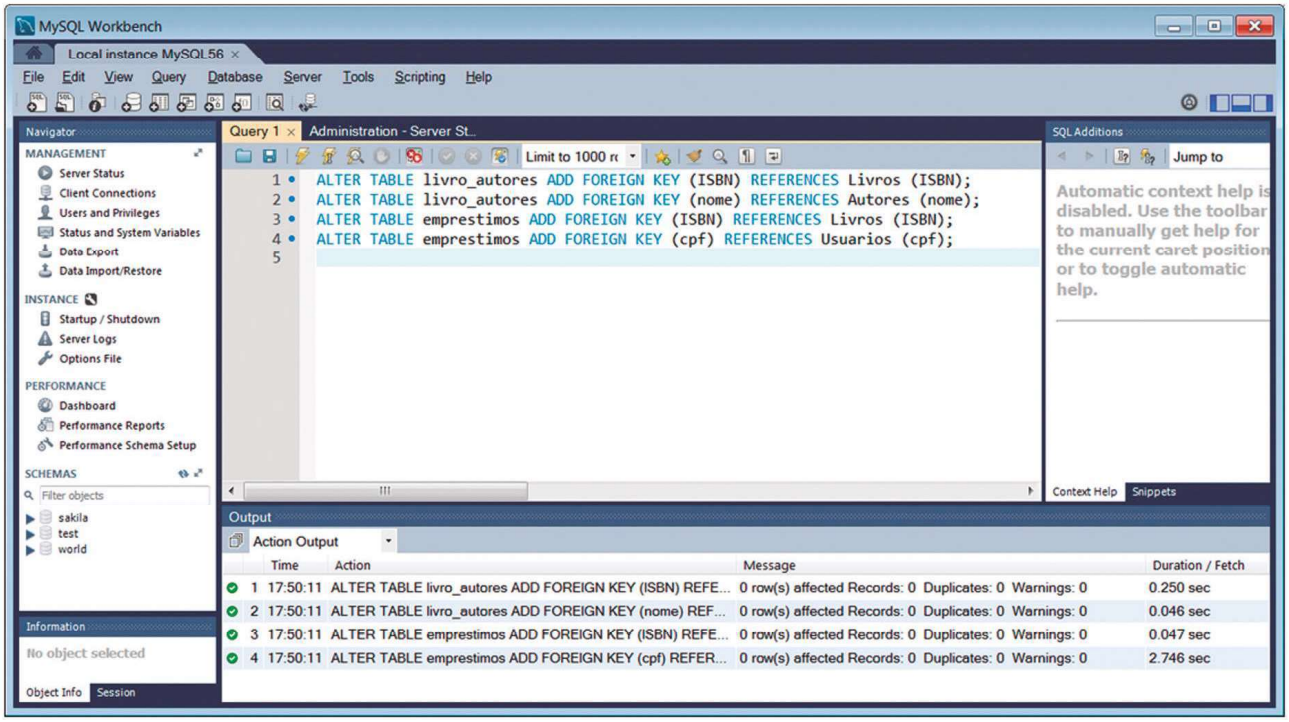
3.6.3 Criar *constraints* de integridade: chaves primárias e estrangeiras

A declaração das chaves poderá ser realizada também por meio do comando de alteração da tabela, que usaremos para criar as chaves estrangeiras:

Passo 6: Digite os comandos a seguir para fazer a declaração das chaves e veja o resultado na imagem logo após.

```
ALTER TABLE livro_autores ADD FOREIGN KEY (ISBN)
REFERENCES Livros (ISBN);

ALTER TABLE livro_autores ADD FOREIGN KEY (nome)
REFERENCES Autores (nome);

ALTER TABLE emprestimos ADD FOREIGN KEY (ISBN)
REFERENCES Livros (ISBN);

ALTER TABLE emprestimos ADD FOREIGN KEY (CPF)
REFERENCES Usuarios (CPF);
```

Figura 39 - Exemplo do comando *DESC* para mostrar a estrututra de uma tabela no BD em uso



Fonte: Produção do próprio autor

No comando ***ALTER TABLE livro_autores ADD FOREIGN KEY (ISBN) REFERENCES Livros (ISBN***); estamos alterando (*ALTER*) a tabela livro_autores, com a adição (*ADD*) ou inclusão de uma chave estrangeira

(*FOREIGN KEY*) no campo ISBN, a qual referencia a tabela Livros no campo ISBN. Com isso, estamos impedindo que seja ligado um autor a um livro inexistente (testaremos mais adiante).

3.6.4 Inserir dados de teste

A sintaxe inicial que será utilizada é *INSERT INTO nome_da_tabela VALUES (valores_dos_dados).*

Os valores serão representados entre parênteses, separados por vírgula, com caracteres entre aspas, na sequência dos campos da estrutura da tabela. Lembre-se de que o terminador de um comando é o ponto e vírgula, e ele deverá vir após o fechamento dos parênteses, antes da próxima declaração.

Atenção

NOTA: os exemplos de seleção e exibição que serão trabalhos na sequência somente apresentarão os resultados como apresentados se os dados que seguem estiverem cadastrados de acordo com o sugerido nos exemplos.

3.6.4.1 *Dados para a tabela Autores*

INSERT INTO Autores VALUES("Isaac Epstein");

Esse comando significa **insira na tabela Autores os valores indicados na cláusula *VALUES***. O conteúdo corresponde aos campos da tabela. Nossa tabela de Autores foi criada com somente um campo, o nome. Então, estamos inserindo o nome do autor na tabela correspondente. Mais adiante, trabalharemos com mais de um campo e mudando a ordem dos campos. Os comandos que seguem têm a mesma explicação, mudando somente os valores dos dados.

Passo 7: Digite os comandos a seguir para inserir os dados na tabela Autores e veja o resultado na imagem logo após.

INSERT INTO Autores VALUES("Peter Chen");

INSERT INTO Autores VALUES("Jane P. Laudon");

INSERT INTO Autores VALUES("Kenneth C. Laudon");

INSERT INTO Autores VALUES("José Simão de Paula Pinto");

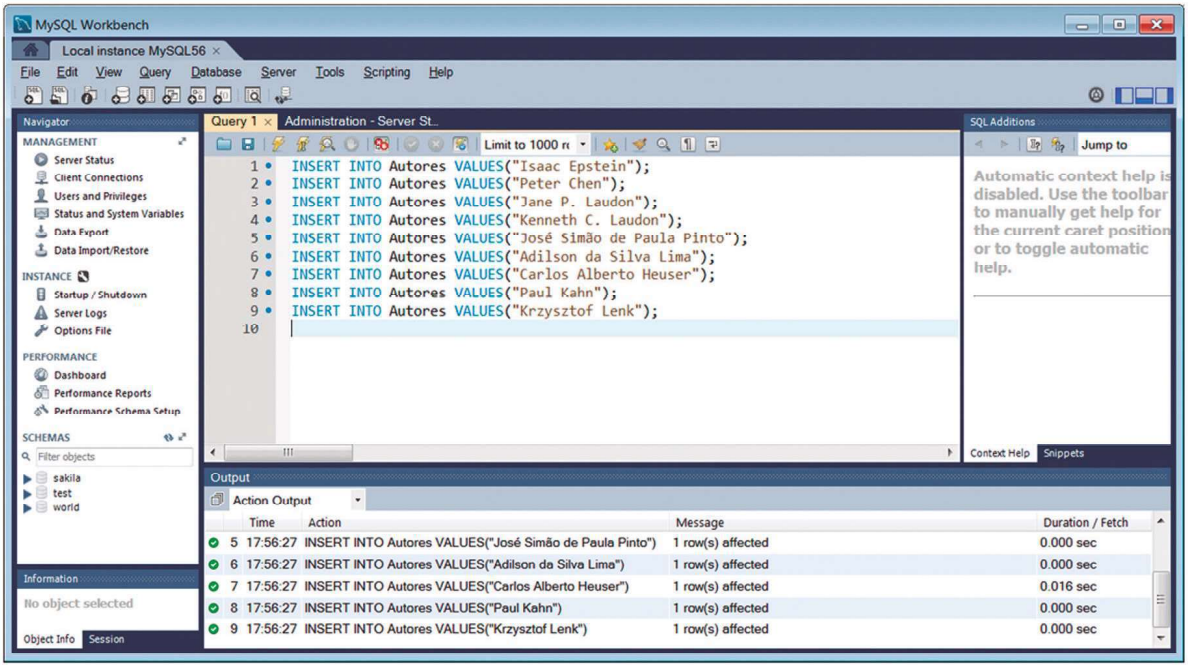
INSERT INTO Autores VALUES("Adilson da Silva Lima");

INSERT INTO Autores VALUES("Carlos Alberto Heuser");

INSERT INTO Autores VALUES("Paul Kahn");

INSERT INTO Autores VALUES("Krzysztof Lenk");

Figura 40 - Exemplo do comando *INSERT INTO* para mostrar a inserção de dados em uma tabela no BD em uso, no caso, a Tabela Autores



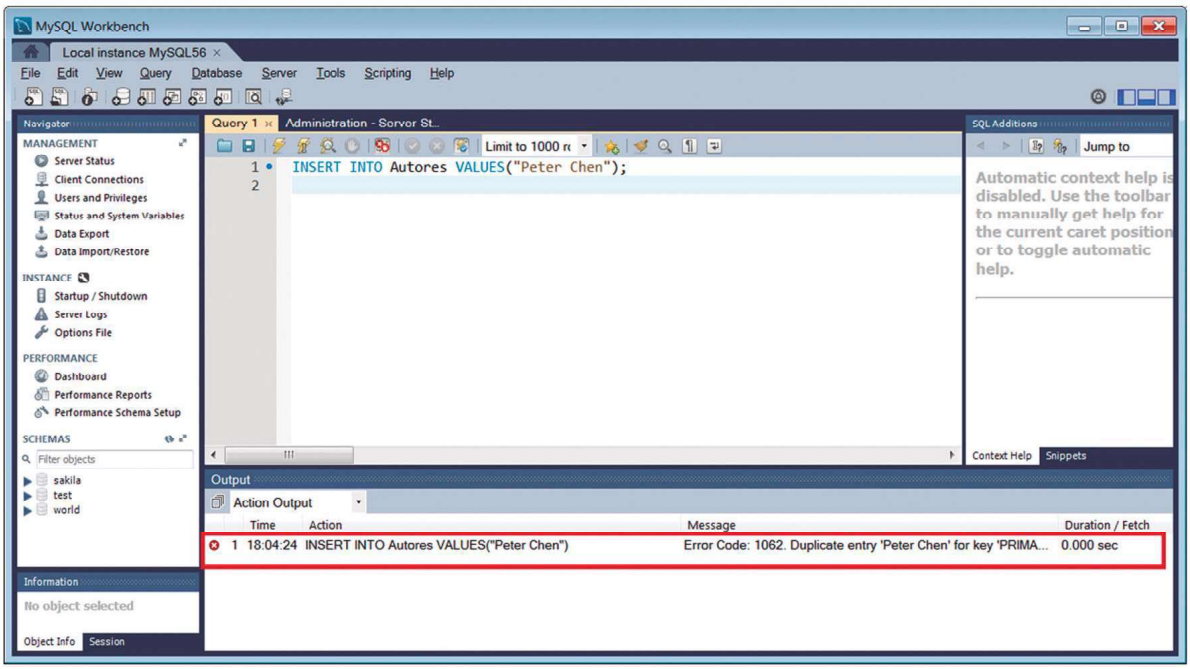
Fonte: Produção do próprio autor

Tente executar novamente o cadastro de um dos autores.

Passo 8: Digite o comando a seguir para tentar reinserir o dado *Peter Chen* na tabela Autores e veja o resultado na imagem logo após.

INSERT INTO Autores *VALUES*("*Peter Chen*");

Figura 41 - Exemplo do comando *INSERT INTO* usado para tentar inserir novamente dados já cadastrados na tabela autores no BD em uso



Fonte: Produção do próprio autor

Observe que aconteceu um erro. O campo nome foi definido como chave, e já existe um autor ***Peter Chen*** cadastrado. Dessa forma, o SGBD indica que não pode realizar a operação. Isso corresponde ao conceito anteriormente estudado de manter a integridade do banco de dados.

Agora, para pensar: e se existirem autores homônimos, como iremos diferenciá-los se estamos cadastrando somente seus nomes no banco de dados?

Resposta: o modelo é insuficiente para essa questão e teria de ser revisto. Isso remete à importância do correto levantamento de requisitos e aspectos de negócio e necessidades dos usuários.

3.6.4.2 *Dados para a tabela Usuarios*

INSERT INTO Usuarios VALUES(98302543055,"Antônio Calisto");

Esse comando significa **insira na tabela Usuarios os valores indicados na cláusula *VALUES***. O conteúdo corresponde aos campos da tabela. Nossa tabela de Usuarios foi criada com os campos CPF e nome, nessa ordem, portanto, foram informados nessa ordem.

Note que os valores correspondentes a caracteres estão entre aspas, enquanto os números não estão. Note que os valores são separados por **,** (vírgula). Os valores são fictícios.

Passo 9: Digite os comandos a seguir para inserir dados na tabela Usuarios e veja o resultado na imagem logo após.

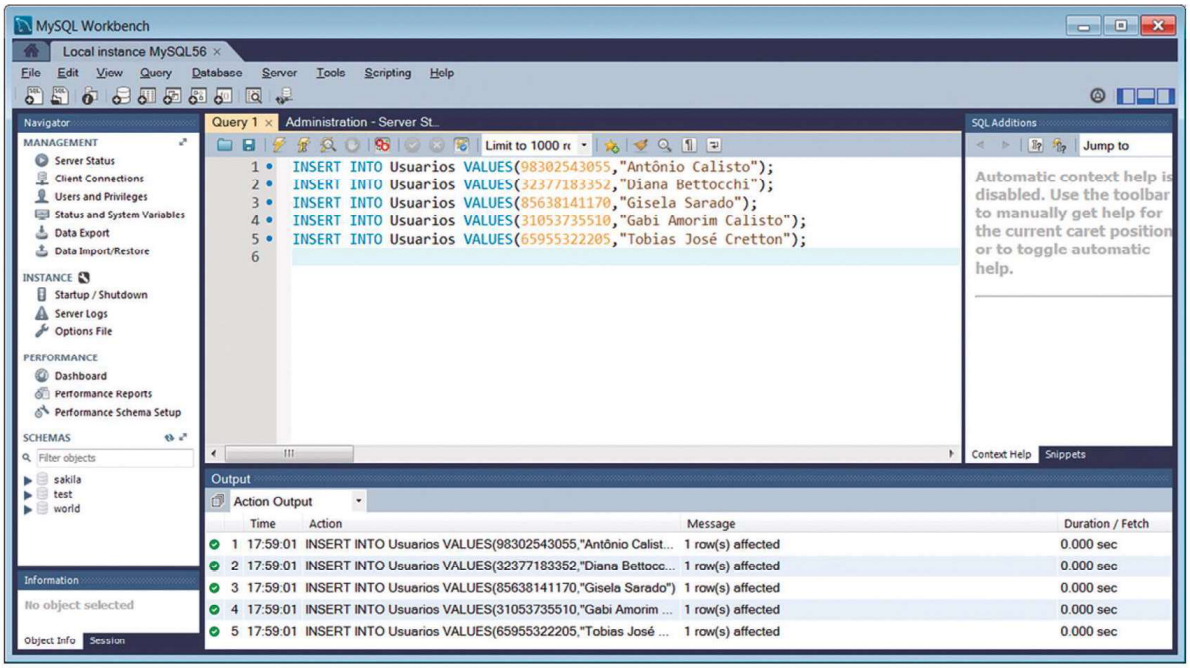
INSERT INTO Usuarios VALUES(32377183352,"Diana Bettocchi");

INSERT INTO Usuarios VALUES(85638141170,"Gisela Sarado");

INSERT INTO Usuarios VALUES(31053735510,"Gabi Amorim Calisto");

INSERT INTO Usuarios VALUES(65955322205,"Tobias José Cretton");

Figura 42 - Exemplo do comando *INSERT INTO* usado para inserir dados na tabela Usuarios no BD em uso



Fonte: Produção do próprio autor

Multimídia

Para obter CPFs e CNPJs fictícios, porém **válidos** em função do algoritmo de validação, assim como nomes de pessoas e de empresas fictícias, exclusivamente para finalidade de teste de *software*, experimente o http://www.geradorCPFcnpj.com/ (há vários outros geradores; pesquise e escolha o melhor para você).

3.6.4.3 *Dados para a tabela Livros*

INSERT INTO Livros VALUES("8508028121","Teoria da informação");

Note que agora a modelagem da tabela foi com dois campos de caracteres, motivo pelo qual os dois valores informados estão entre aspas.

Passo 10: Digite os comandos a seguir para inserir dados na tabela Livros e veja o resultado na imagem logo após.

INSERT INTO Livros VALUES("0074605755","Modelagem de dados");

INSERT INTO Livros VALUES("9788576059233","Sistemas de informação gerenciais");

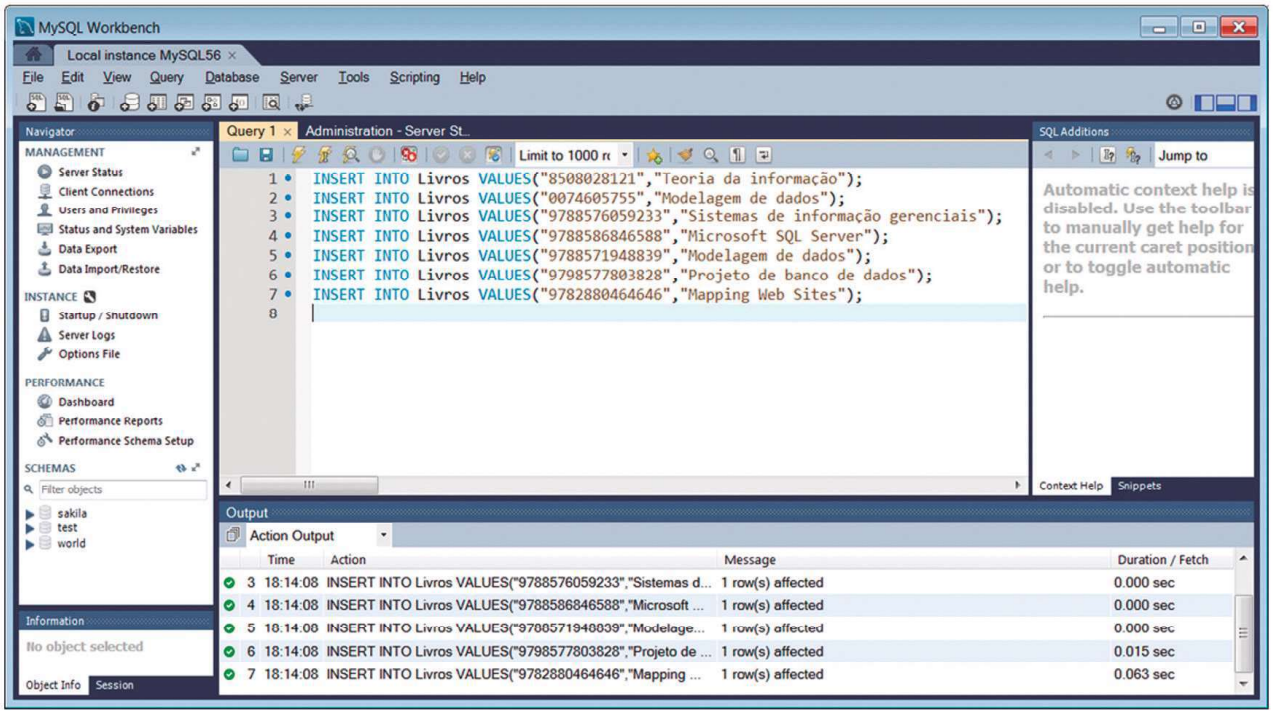
INSERT INTO Livros VALUES("9788586846588","Microsoft SQL Server");

INSERT INTO Livros VALUES("9788571948839","Modelagem de dados");

INSERT INTO Livros VALUES("9798577803828","Projeto de banco de dados");

INSERT INTO Livros VALUES("9782880464646","Mapping Web Sites");

Figura 43 - Exemplo do comando *INSERT INTO* usado para inserir dados na tabela Livros no BD em uso



Fonte: Produção do próprio autor

3.6.4.4 *Dados para a tabela livro_autores*

Passo 11: Digite os comandos a seguir para inserir dados na tabela *livro_autores* e veja o resultado na imagem logo após.

Note que o comando está inserindo dados correspondentes aos campos da tabela, na ordem: número de ISBN e autor do livro.

INSERT INTO livro_autores VALUES("8508028121","Isaac Epstein");

INSERT INTO livro_autores VALUES("0074605755","Peter Chen");

INSERT INTO livro_autores VALUES("9788576059233","Jane P. Laudon");

INSERT INTO livro_autores VALUES("9788576059233","Kenneth C. Laudon");

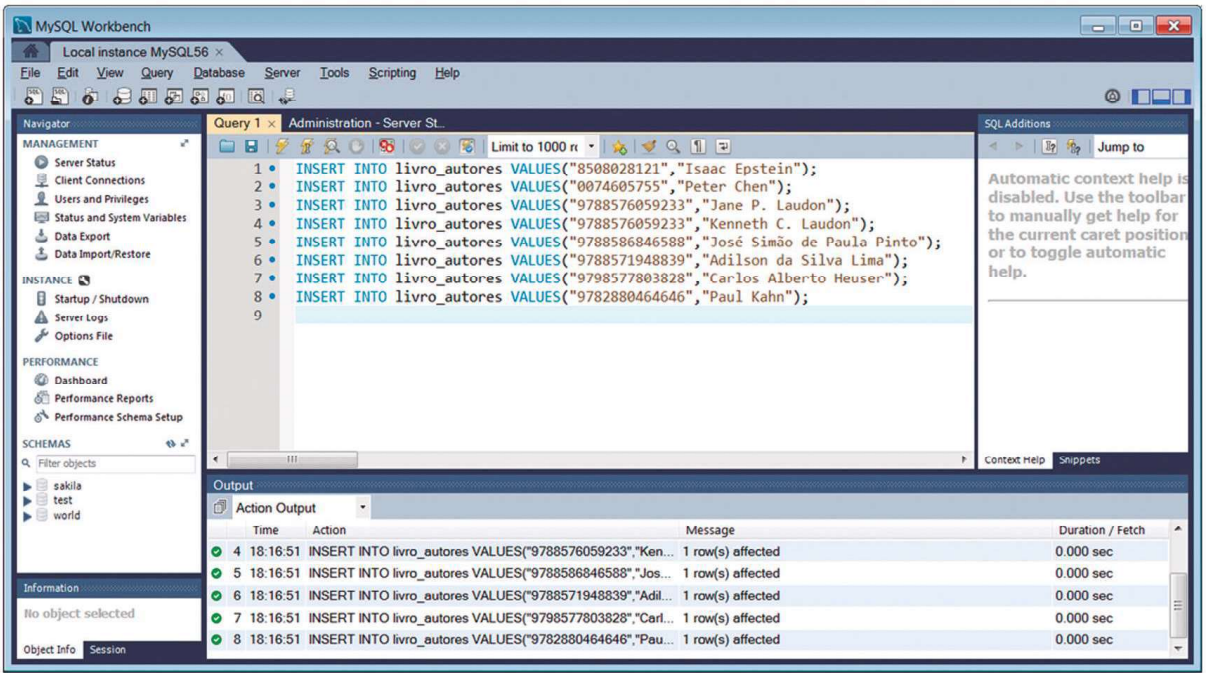
INSERT INTO livro_autores VALUES("9788586846588","José Simão de Paula Pinto");

INSERT INTO livro_autores VALUES("9788571948839","Adilson da Silva Lima");

INSERT INTO livro_autores VALUES("9798577803828","Carlos Alberto Heuser");

INSERT INTO livro_autores VALUES("9782880464646","Paul Kahn");

Figura 44 - Exemplo do comando *INSERT INTO* usado para inserir dados na tabela Livro-autores no BD em uso



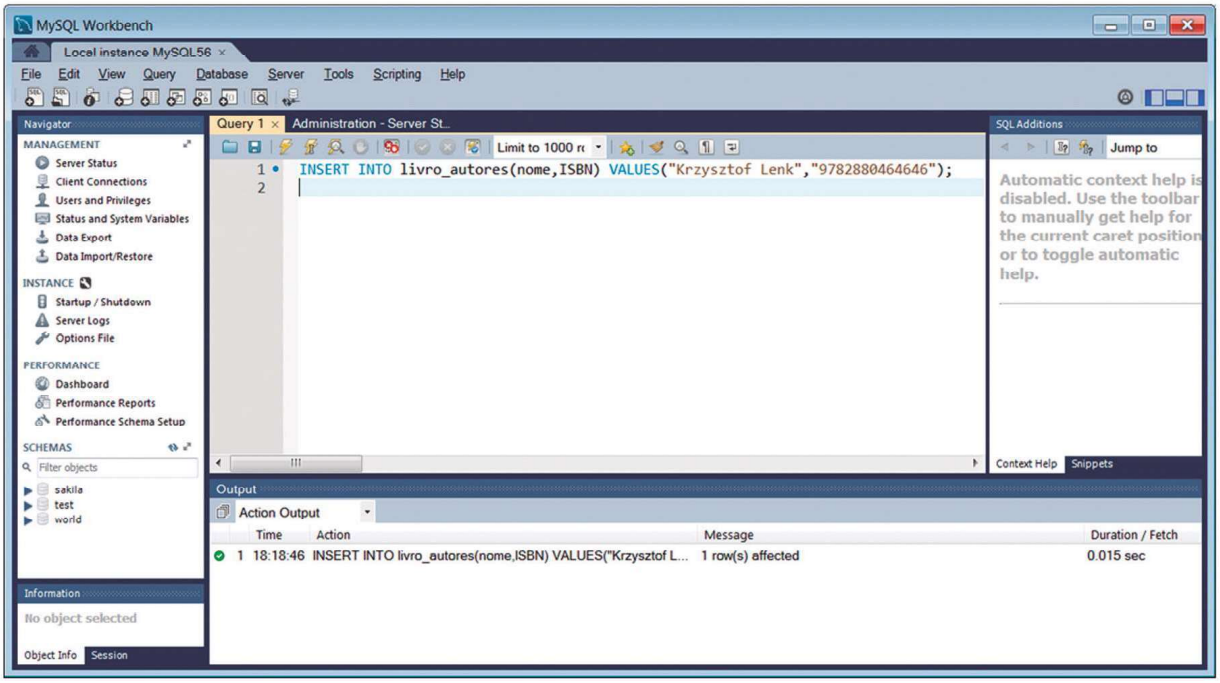
Fonte: Produção do próprio autor

Passo 12: Digite os comandos a seguir para inserir dados na tabela livro_autores e veja o resultado na imagem logo após.

Note que o comando está inserindo na tabela que relaciona livros e autores os conteúdos do nome e do ISBN.

INSERT INTO livro_autores(nome,ISBN) VALUES("Krzysztof Lenk","9782880464646");

Figura 45 - Exemplo do comando *INSERT INTO* usado para inserir dados na tabela Livro-autores no BD em uso



Fonte: Produção do próprio autor

Perceba que agora houve uma mudança no comando. Vamos analisar: até aqui, sempre que inserimos um conteúdo, informamos a tabela e os valores, na ordem exata dos campos. Neste último comando invertemos a ordem em que os dados foram informados; para isso, foi passada no comando a **ordem** em que os campos aparecem na cláusula *VALUES*, e **quais são os campos**. Isso permite que os dados sejam informados em quaisquer ordens necessárias à conveniência do usuário, por exemplo, caso venham em um arquivo e estejam em ordem diferente daquela da tabela.

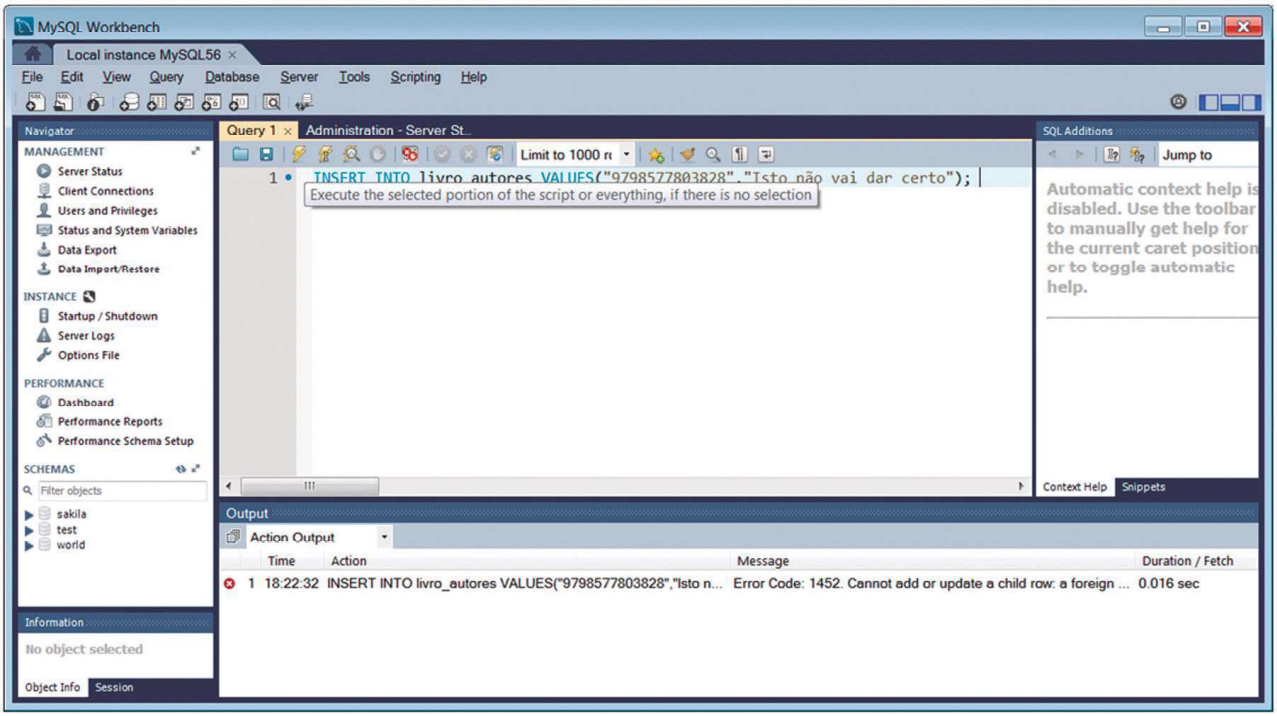
Também possibilita que sejam fornecidas informações incompletas, por exemplo, quando não temos o valor de um dos campos, bastando omiti-lo da sequência (desde que não seja uma chave) e, evidentemente, não informar um valor para ele.

Continuando com o teste de integridade, experimente agora executar outro comando.

Passo 13: Digite os comandos a seguir para inserir dados na tabela livro_autores e veja o resultado na imagem logo após.

> *INSERT INTO livro_autores VALUES("9798577803828","Isto não vai dar certo");*

Figura 46 - Exemplo do comando *INSERT INTO* usado para inserir dados na tabela Livro-autores no BD em uso



Fonte: Produção do próprio autor

O problema é de integridade referencial, ou seja, tentamos inserir um livro que existe associado a um autor que não existe. Veja os pontos grifados da mensagem de erro:

> *"Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails ('pratica1'.livro_autores', CONSTRAINT 'livro_autores_ibfk1' **FOREIGN KEY**('ISBN') **REFERENCES**...)".*

Perceba a importância da boa modelagem e da correta estruturação das tabelas e suas *constraints*: uma vez que esteja bem formulado, o SGBD cuidará para que tudo seja respeitado, garantindo a integridade dos dados e a qualidade do banco de dados.

Explicativo

Metadados

É necessário, neste ponto do curso, que você perceba a importância entre os aspectos teóricos relativos a metadados e o que está acontecendo agora no banco de dados criado. O SGBD utiliza e gerencia os metadados relativos ao banco de dados e tabelas criados, assim como os tipos de dados escolhidos, sejam estruturais, sejam de domínio, para gerenciar os dados e garantir sua integridade.

3.6.4.5 *Dados para a tabela emprestimos*

O que acontecerá se digitarmos o comando a seguir? Você tem algum palpite?

INSERT INTO emprestimos VALUES ("9782880464646", "6595532 2205","2014/11/20",NULL);

A novidade nessa linha de comando é que um dos valores passados é nulo. Nesse caso, porque a data de devolução ainda não é conhecida.

Outra forma que poderia ter sido utilizada para esse comando seria declarar ***INSERT INTO* emprestimos (ISBN,CPF, data_emprestimo) *VALUES* ("___","___","___")** e passar somente os três valores; informando explicitamente os campos a cadastrar e ignorando o outro, o SGBD assume que seu valor é nulo automaticamente (ou, se estiver programado para isso, coloca um valor padrão: *default*). Claro que, para essa opção funcionar, a estrutura da tabela deve ter sido declarada com o campo em questão aceitando valores nulos.

Note, ainda, que a data está sendo informada segundo a norma inglesa, isto é, ANO-MÊS-DIA. Mais adiante veremos como formatar sua exibição para o nosso formato usual, DIA-MÊS-ANO.

Passo 14: Digite os comandos a seguir para inserir dados na tabela emprestimos e veja o resultado na imagem logo após.

INSERT INTO emprestimos VALUES ("9798577803828", "6595532 2205","2014/11/20",NULL);

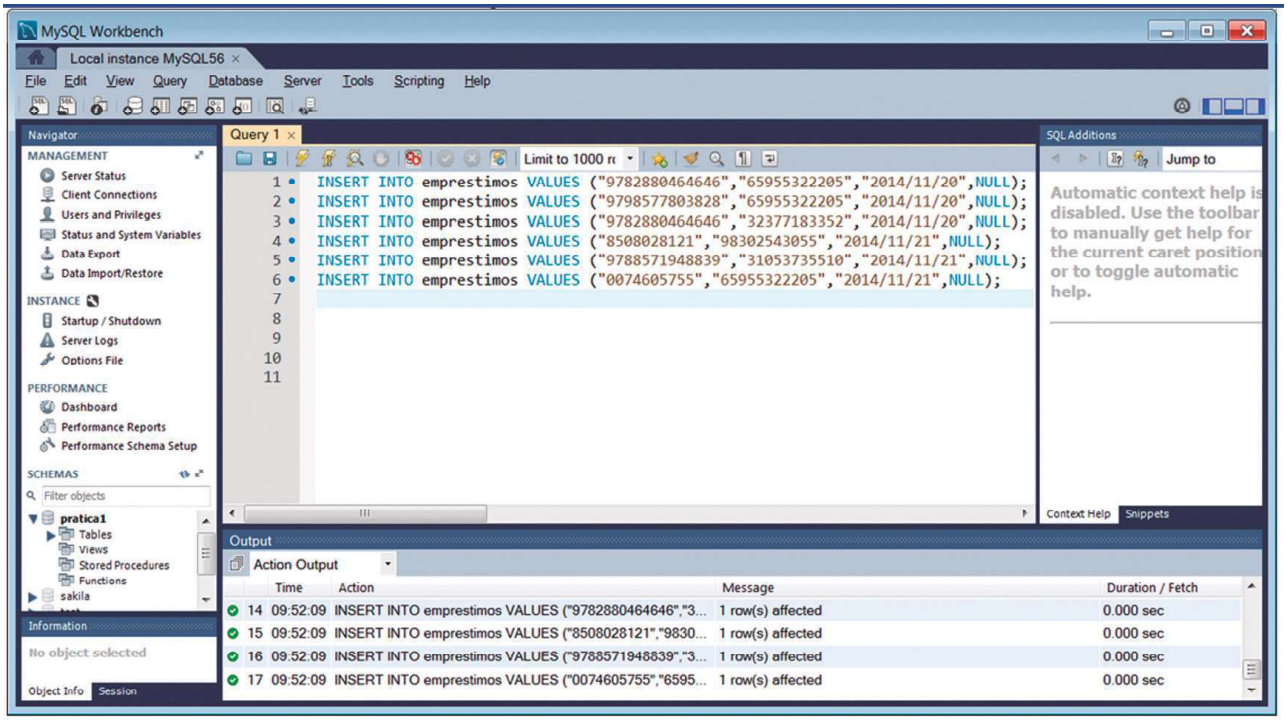
INSERT INTO emprestimos VALUES ("9782880464646", "3237718 3352","2014/11/20",NULL);

INSERT INTO emprestimos VALUES ("8508028121", "9830254305 5","2014/11/21",NULL);

INSERT INTO emprestimos VALUES ("9788571948839", "3105373 5510","2014/11/21",NULL);

INSERT INTO emprestimos VALUES ("0074605755", "6595532220 5","2014/11/21",NULL);

Figura 47 - Exemplo do comando *INSERT INTO* usado para inserir dados na tabela Empréstimos no BD em uso



Fonte: Produção do próprio autor

Note que essa tabela (emprestimos) é uma tabela de relacionamento, totalmente dependente das demais, pois ela somente pode existir se houver usuários e livros previamente cadastrados. Além disso, trata-se, para esse exemplo, de uma tabela que sofrerá inserções e atualizações constantes, pois ela implementa a regra de negócio **empréstimo e devolução de livros**.

Explicativo

É usual em SGBDs, e na informática em geral, o uso de datas no formato inglês (ANO-MÊS-DIA). Esse formato é mantido não somente por razões históricas ou de origem do sistema, mas também por ser extremamente útil para o cálculo de datas: é mais fácil para subtrair uma data de outra e ter o resultado diretamente em dias.

3.6.5 Atualizar dados

Como vimos na inserção de dados de empréstimos, será necessário, em dado momento, informar a data de devolução da obra. Para isso, se for utilizado o comando *INSERT* novamente teremos um problema, pois estaremos criando uma nova linha na tabela, e não atualizando a linha já criada. Para esse caso, devemos utilizar o comando *UPDATE*.

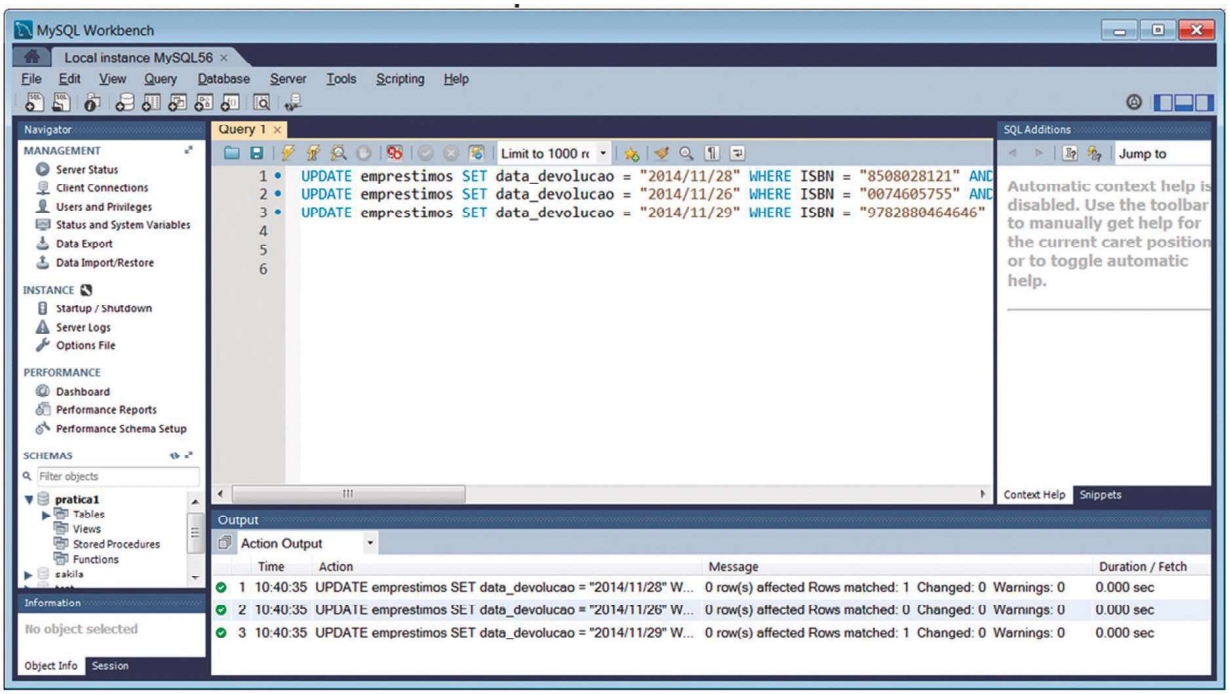
```
UPDATE emprestimos SET data_devolucao = "2014/11/28" WHERE
ISBN = "8508028121" AND CPF = "98302543055";
```

O comando passado ao SGBD aqui é: atualize (*UPDATE*) a tabela emprestimos, estabeleça (*SET*) para o campo *data_devolucao* o valor **2014/11/28** quando (*WHERE*) o valor do ISBN for **8508028121** e (*AND*) o valor do CPF for **98302543055**. Ou seja, especificamente para o usuário cujo CPF foi informado, e para o livro cujo ISBN foi fornecido, estamos informando agora a sua data de devolução. Caso não tivéssemos informado o CPF, somente o ISBN, todos os livros com aquele ISBN teriam sua data de devolução alterada. Por outro lado, se informássemos somente o CPF e não o ISBN, todos os livros emprestados para aquele usuário teriam sua data de devolução alterada. Portanto, cuidado na execução dos comandos; se ele for válido, o SGBD vai executá-lo. E não há como desfazê-lo.

Passo 15: Digite os comandos a seguir para inserir dados de devolução na tabela emprestimos e veja o resultado na imagem logo após.

```
UPDATE emprestimos SET data_devolucao = "2014/11/26" WHERE
ISBN = "0074605755" AND CPF = "98302543055";

UPDATE emprestimos SET data_devolucao = "2014/11/29"  WHERE
ISBN = "9782880464646" AND CPF = "65955322205";
```

Figura 48 - Exemplo do comando *UPDATE* usado para inserir dados de devolução na tabela Empréstimos do BD em uso



Fonte: Produção do próprio autor

Neste momento, cabe mais uma crítica ao nosso modelo de dados: o que aconteceria se o mesmo usuário tivesse tomado emprestado o mesmo livro (e já devolvido) há um ano atrás? Resposta: o registro também teria sido alterado! Os dados ficariam com a qualidade contestável. Tal situação não foi prevista na modelagem.

Por fim, mais um pequeno teste, vamos tentar atualizar um empréstimo que não existe. Execute:

```
UPDATE emprestimos SET data_devolucao = "2014/11/28" WHERE
ISBN = "8508028121" AND CPF = "65955322205";
```

A resposta do SGBD será: *0 row(s) affected Rows matched: 0 Changed: 0.* Isso significa que nenhuma linha da tabela foi modificada *(0 row(s) affected)*, pois nenhuma corresponde aos critérios de busca dados pela cláusula *WHERE* (*Rows matched: 0*).

Note que não houve erro: o comando foi executado com perfeição, porém nada foi afetado, pois nada atendeu ao critério fornecido. Se, em vez de *AND* tivéssemos optado por *OR* (ou) na formulação do comando, duas linhas seriam, erroneamente, afetadas, o que demonstra, novamente, a necessidade de atenção à lógica da declaração formulada.

3.6.6 Praticar recuperação de dados e pesquisas no banco de dados

Agora vamos praticar comandos de seleção. O comando *SELECT* é extremamente utilizado e é capaz de realizar duas operações fundamentais conjuntamente: selecionar e exibir dados. A sintaxe completa desse comando é complexa e costuma ser enorme em todos os SGBDs, já que possui muitas variações nas declarações e comandos que podem ser agregados. Veremos alguns deles.

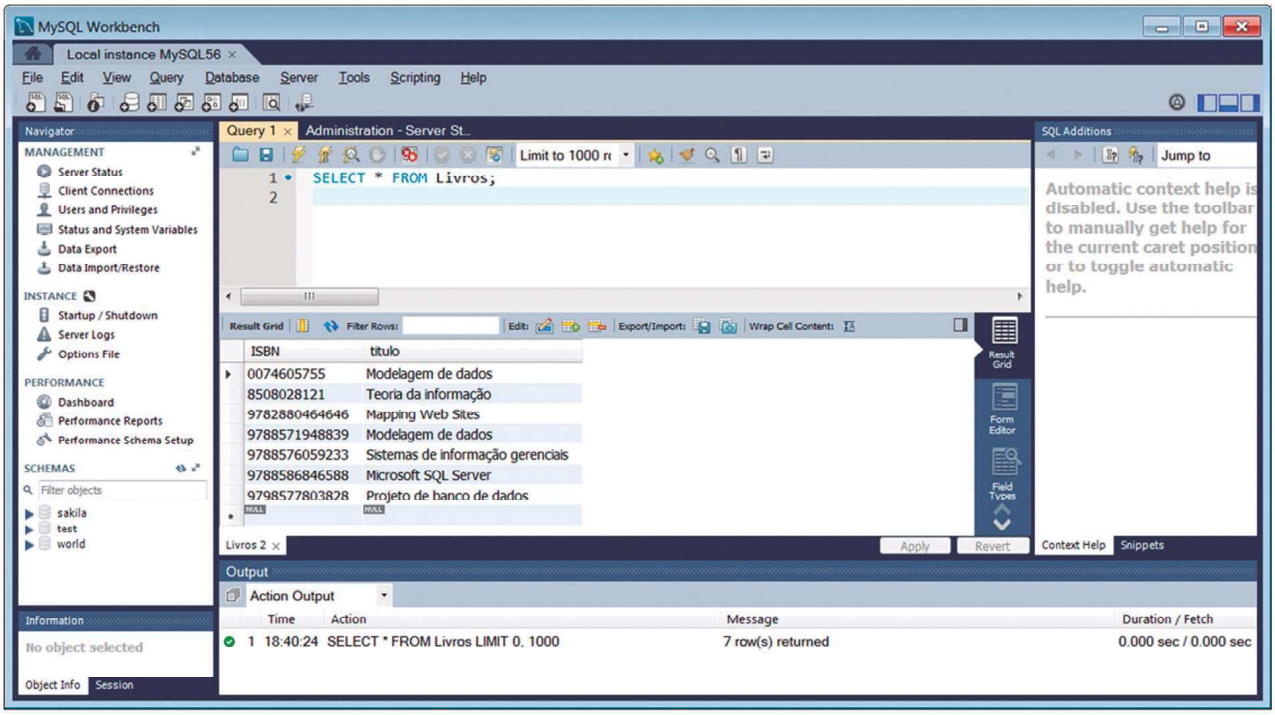
De forma geral, a sintaxe do comando é *SELECT o_que FROM de_onde*, na maioria das vezes seguido de *WHERE* condições. A cláusula *o_que* é substituída pelos nomes dos campos desejados, usando-se ***** para indicar todos, ou por uma declaração que especifica como deve ser exibido um dado. Na cláusula de_onde será informado o nome da tabela ou das tabelas das quais os dados serão recuperados. A cláusula *WHERE* especifica condições de busca, por exemplo, nomes iniciando com **A**. Vejamos alguns exemplos.

3.6.6.1 *Ver informações dos livros cadastrados*

Passo 16: Digite os comandos a seguir para selecionar e exibir dados da tabela Livros e veja o resultado na imagem logo após.

```
SELECT * FROM Livros;
```

Figura 49 - Exemplo do comando *SELECT* usado para selecionar e exibir dados da tabela Livros do BD em uso



Fonte: Produção do próprio autor

Aqui a ordem dada ao SGBD é **selecione e mostre** (***SELECT***) **todos os campos (*) da tabela Livros**. O resultado é a exibição de uma LISTA contendo no cabeçalho os nomes dos campos e nas diversas linhas seus conteúdos (perceba que surgiu na tela uma outra divisão, pois agora existem dados a serem informados, afinal o comando foi justamente para provocar isso, como já havia ocorrido com os comandos *SHOW TABLES* e *DESC* anteriormente praticados). Observe que a exibição dos nomes dos campos segue a ordem na qual foram declarados na estrutura da tabela, mas a ordem de exibição desejada (se for diferente) pode ser informada no comando; na sequência dessa prática, veremos como fazê-lo. Os dados aparecem na ordem em que foram inseridos no banco, ordem esta que também pode ser modificada, conforme critérios informados na consulta realizada (muitos programas clientes de SGBDs costuma trazer os dados já ordenados, em vez de na sequência em que foram criados, porém, isso não é padrão do SQL).

Para maior clareza, deste ponto em diante, vamos verificar somente a lista retornada no centro da tela da interface do cliente de consultas:

Figura 50 - Exemplo de resultado de lista de dados da tabela Livros do BD em uso

ISBN	titulo
0074605755	Modelagem de dados
8508028121	Teoria da informação
9782880464646	Mapping Web Sites
9788571948839	Modelagem de dados
9788576059233	Sistemas de informação gerenciais
9788586846588	Microsoft SQL Server
9798577803828	Projeto de banco de dados
NULL	NULL

Fonte: Produção do próprio autor

Note que a ordem de apresentação é a ordem da estrutura da tabela, pois não houve menção em contrário. Podemos inverter a ordem em que os campos aparecem.

Passo 17: Digite os comandos a seguir para selecionar e exibir dados da tabela Livros por ISBN e veja o resultado na imagem logo após.

SELECT titulo,ISBN FROM Livros;

Figura 51 - Exemplo de resultado de lista de dados da tabela Livros do BD em uso, selecionando por ISBN

titulo	ISBN
Modelagem de dados	0074605755
Teoria da informação	8508028121
Mapping Web Sites	9782880464646
Modelagem de dados	9788571948839
Sistemas de informação gerenciais	9788576059233
Microsoft SQL Server	9788586846588
Projeto de banco de dados	9798577803828
NULL	NULL

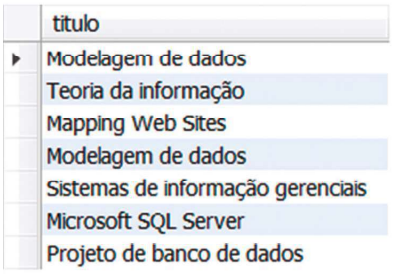
Fonte: Produção do próprio autor

Ou ainda, podemos retornar somente os campos desejados.

Passo 18: Digite os comandos a seguir para selecionar e exibir dados da tabela Livros por Título e veja o resultado na imagem logo após.

SELECT titulo FROM Livros;

Figura 52 - Exemplo de resultado de lista de dados da tabela Livros do BD em uso, selecionando por Título



titulo
Modelagem de dados
Teoria da informação
Mapping Web Sites
Modelagem de dados
Sistemas de informação gerenciais
Microsoft SQL Server
Projeto de banco de dados

Fonte: Produção do próprio autor

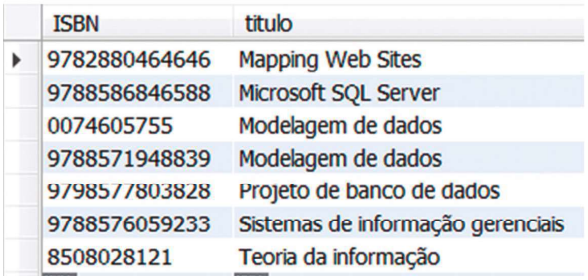
O comando *SELECT* foi executado informando o campo desejado, titulo. Dessa forma, somente o campo informado foi recuperado e exibido. Poderíamos ter solicitado uma lista de campos, ou os campos em qualquer ordem.

3.6.6.2 *Ver informações dos livros cadastrados, ordenados pelo título do livro*

Passo 19: Digite os comandos a seguir para selecionar e exibir dados da tabela Livros por Título, em ordem alfabética e veja o resultado na imagem logo após.

*SELECT * FROM Livros ORDER BY titulo;*

Figura 53 - Exemplo de resultado de lista de dados da tabela Livros do BD em uso, selecionando por Título, em ordem alfabética



ISBN	titulo
9782880464646	Mapping Web Sites
9788586846588	Microsoft SQL Server
0074605755	Modelagem de dados
9788571948839	Modelagem de dados
9798577803828	Projeto de banco de dados
9788576059233	Sistemas de informação gerenciais
8508028121	Teoria da informação

Fonte: Produção do próprio autor

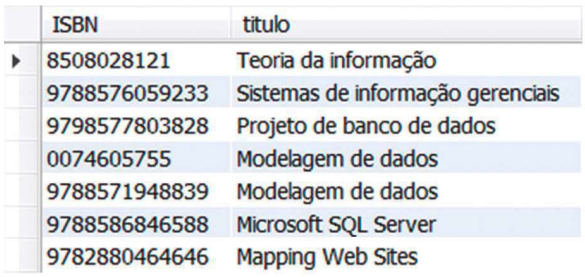
O comando *SELECT* foi executado com cláusula *ORDER BY*, seguida do nome de um campo (poderia ser mais de um). Dessa forma, o resultado foi ordenado e exibido em função do título do livro.

3.6.6.3 *Ver informações dos livros cadastrados, ordenados pelo título do livro em ordem descendente*

Passo 20: Digite os comandos a seguir para selecionar e exibir dados da tabela Livros por Título, em ordem descendente, e veja o resultado na imagem logo após.

*SELECT * FROM Livros ORDER BY titulo desc;*

Figura 54 - Exemplo de resultado de lista de dados da tabela Livros do BD em uso, selecionando por Título, em ordem descendente



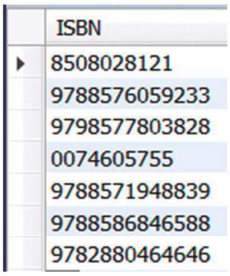
ISBN	titulo
8508028121	Teoria da informação
9788576059233	Sistemas de informação gerenciais
9798577803828	Projeto de banco de dados
0074605755	Modelagem de dados
9788571948839	Modelagem de dados
9788586846588	Microsoft SQL Server
9782880464646	Mapping Web Sites

Fonte: Produção do próprio autor

O comando *SELECT* foi executado com cláusula *ORDER BY*, seguida do nome de um campo e da cláusula *DESC*. Dessa forma, o resultado foi ordenado e exibido em função do título do livro na forma descendente (ou decrescente, termo este utilizado se estivéssemos trabalhando com números).

Note que podemos combinar a troca de ordem dos campos com a opção de ordenação. Por exemplo, que consulta foi enviada ao SGBD para produzir o seguinte resultado?

Figura 55 - Exemplo de resultado de lista de dados da tabela Livros do BD em uso, selecionando por ISBN



ISBN
8508028121
9788576059233
9798577803828
0074605755
9788571948839
9788586846588
9782880464646

Fonte: Produção do próprio autor

Se você pensou em consulta por livros em ordem descendente, acertou! Veja o comando aplicado a seguir.

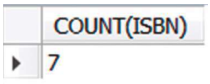
SELECT ISBN FROM Livros ORDER BY titulo desc;

3.6.6.4 *Determinar quantos livros estão cadastrados*

Passo 21: Digite os comandos a seguir para determinar a quantidade de livros cadastrados na tabela Livros e veja o resultado na imagem logo após.

SELECT COUNT(ISBN) FROM Livros;

Figura 56 - Exemplo de resultado de lista de dados da tabela Livros do BD em uso, solicitando quantidade de livros cadastrados em uma dada tabela, no caso, a de Livros



COUNT(ISBN)
7

Fonte: Produção do próprio autor

O comando *SELECT* foi executado informando-se a cláusula *COUNT*(ISBN) na qual *COUNT* vem contar e ISBN é um campo que tem informações únicas, de forma que, se contarmos o número de ISBNs diferentes, teremos a quantidade de livros cadastrados. O valor **7** exibido abaixo do cabeçalho ***COUNT*(ISBN)** corresponde à informação desejada, ou seja, são sete livros cadastrados.

Note que estamos criando um novo dado, correspondente ao total de livros cadastrados. Essa **informação** não foi cadastrada no banco de dados; ela está sendo **gerada** no momento em que for necessária. Esse procedimento também é usual, por exemplo, quando necessitamos de uma idade: calculamos a idade a partir da diferença entre a data atual e a data de nascimento.

Além de diminuir a ocupação de espaço, a vantagem de gerar informações em tempo de execução é que não precisamos, a cada momento, atualizar dados a partir da alteração de outros dados ou de um evento importante. Por exemplo, se tivéssemos cadastrado em alguma tabela a quantidade de livros existentes no banco de dados, a cada nova inserção de um livro, teríamos que atualizar o campo correspondente ao total. Da mesma forma, se for cadastrada a idade de uma pessoa no banco de dados a cada novo evento de mudança de data, ou seja, todos os dias, teríamos que atualizar as idades, verificando se mudaram ou não. E isso para todos os registros...

3.6.6.5 *Determinar quantos livros estão cadastrados, fornecendo o nome de Quantidade para a coluna que conterá a resposta*

Passo 22: Digite os comandos a seguir para determinar a quantidade de livros cadastrados na tabela Livros fornecendo o nome de quantidade e veja o resultado na imagem logo após.

SELECT COUNT(ISBN) AS Quantidade FROM Livros;

Figura 57 - Exemplo de resultado de lista de dados da tabela Livros do BD em uso, solicitando quantidade de livros cadastrados na dada tabela Livros, fornecendo o nome quantidade

	Quantidade
▶	7

Fonte: Produção do próprio autor

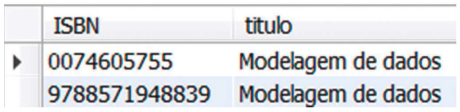
O comando *SELECT* foi executado informando-se a cláusula *COUNT*(ISBN), na qual *COUNT* vem de contar e ISBN é um campo que tem informações únicas, de forma que, se contarmos o número de ISBNs diferentes, teremos a quantidade de livros cadastrados, exatamente como no item 5.5. Porém, dessa vez, foi incluído na declaração AS Quantidade, informando ao SGBD como deverá ser exibido o cabeçalho. Note que esse cabeçalho é válido somente para essa exibição; os nomes de campos no banco de dados não são alterados. A cláusula **AS** (apresentado como) permite a modificação do nome de exibição de qualquer campo, não somente daqueles correspondentes a valores calculados.

3.6.6.6 *Listar livros cujos títulos comecem com a palavra Modelagem*

Passo 23: Digite os comandos a seguir para determinar a quantidade de livros cadastrados na tabela Livros fornecendo o título modelagem e veja o resultado na imagem logo após.

*SELECT * FROM Livros WHERE titulo LIKE 'Modelagem%';*

Figura 58 - Exemplo de resultado de lista de dados da tabela Livros do BD em uso, solicitando quantidade de livros cadastrados na dada tabela Livros, fornecendo o nome para título como Modelagem



ISBN	titulo
0074605755	Modelagem de dados
9788571948839	Modelagem de dados

Fonte: Produção do próprio autor

O comando *SELECT* foi executado em conjunto com a cláusula *WHERE*. Dessa forma, somente os campos que atenderem o que for declarado após essa cláusula serão retornados. Ou seja, a declaração *WHERE* determina condições para a seleção dos dados. Nesse caso, a cláusula utilizou o operador *LIKE*, que significa parecido.

Também foi utilizado o operador **%**, que significa que podem existir outros caracteres na sequência (equivale ao operador * nos sistemas operacionais, que em SQL tem outra função).

O que aconteceria se, em vez de *LIKE*, tivéssemos utilizado **=** ou se não tivéssemos usado o **%**? Vamos ver. Execute as seguintes variações do comando e veja os resultados.

Passo 24: Digite os comandos a seguir e veja o resultado na imagem logo após.

*SELECT * FROM Livros WHERE titulo = 'Modelagem%';*

Figura 59 - Exemplo de resultado de lista de dados da tabela Livros do BD em uso, solicitando quantidade de livros cadastrados na dada tabela Livros, fornecendo o nome para título como Modelagem

ISBN	titulo
NULL	NULL

Fonte: Produção do próprio autor

Passo 25: Digite os comandos a seguir e veja o resultado na imagem logo após.

*SELECT * FROM Livros WHERE titulo LIKE 'Modelagem';*

Figura 60 - Exemplo de resultado de lista de dados da tabela Livros do BD em uso, solicitando quantidade de livros cadastrados na dada tabela Livros, fornecendo o nome para título como Modelagem

ISBN	titulo
NULL	NULL

Fonte: Produção do próprio autor

Passo 26: Digite os comandos a seguir e veja o resultado na imagem logo após.

*SELECT * FROM Livros WHERE titulo = 'Modelagem';*

Figura 61 - Exemplo de resultado de lista de dados da tabela Livros do BD em uso, solicitando quantidade de livros cadastrados na dada tabela Livros, fornecendo o nome para título como Modelagem

Fonte: Produção do próprio autor

Note que os resultados diferem em função de a consulta solicitar dados **parecidos com** ou **exatamente igual a**. Nenhuma das três últimas consultas retornou dados (o que é indicado por *NULL* na lista). Isso ocorre pois não há um livro cadastrado exclusivamente com o nome de **Modelagem**, o que possibilitaria o uso do operador **=**, e porque não podemos utilizar o operador **%** em conjunto com **=**, somente com ***LIKE***. Novamente: tome cuidado ao formular suas consultas.

3.6.6.7 *Uso da cláusula WHERE para unir dados de tabelas diferentes*

Podemos utilizar a cláusula *WHERE* para formular uma condição ao SGBD que seja pertinente ao nosso modelo de dados, por exemplo, unir informações de tabelas diferentes, mas que pertencem ao mesmo contexto. Vamos obter uma lista com os autores e seus respectivos livros.

Passo 27: Digite os comandos a seguir aplicando o comando *WHERE* e veja o resultado na imagem logo após.

```
SELECT nome,titulo FROM livro_autores,Livros WHERE livro_autores.ISBN=Livros.ISBN;
```

Figura 62 - Exemplo de resultado usando o comando *WHERE*

nome	titulo
Peter Chen	Modelagem de dados
Isaac Epstein	Teoria da informação
Krzysztof Lenk	Mapping Web Sites
Paul Kahn	Mapping Web Sites
Adilson da Silva Lima	Modelagem de dados
Jane P. Laudon	Sistemas de informação gerenciais
Kenneth C. Laudon	Sistemas de informação gerenciais
José Simão de Paula Pinto	Microsoft SQL Server
Carlos Alberto Heuser	Projeto de banco de dados

Fonte: Produção do próprio autor

Nesse comando, produzimos uma lista de autores e livros, ligando-os por meio da chave previamente definida, o ISBN, que é chave primária na tabela Livros e estrangeira na livro_autores. Isso torna o comando extremamente poderoso, e corresponde ao início das operações conhecidas como JUNÇÃO (em inglês, *JOIN*, mas não serão exploradas nesta disciplina). O resultado da execução dessa consulta é chamado de **produto cartesiano** e, tecnicamente, tem a ver com a álgebra relacional e operações com conjuntos.

E se a cláusula *WHERE* não fosse especificada?

Passo 28: Digite os comandos a seguir sem aplicar o comando *WHERE* e veja o resultado na imagem logo após.

SELECT nome,titulo FROM livro_autores,Livros;

Figura 63 - Exemplo de resultado sem usar o comando *WHERE*

nome	titulo
Adilson da Silva Lima	Modelagem de dados
Adilson da Silva Lima	Teoria da informação
Adilson da Silva Lima	Mapping Web Sites
Adilson da Silva Lima	Modelagem de dados
Adilson da Silva Lima	Sistemas de informação gerenciais
Adilson da Silva Lima	Microsoft SQL Server
Adilson da Silva Lima	Projeto de banco de dados
Carlos Alberto Heuser	Modelagem de dados
Carlos Alberto Heuser	Teoria da informação
Carlos Alberto Heuser	Mapping Web Sites
Carlos Alberto Heuser	Modelagem de dados
Carlos Alberto Heuser	Sistemas de informação gerenciais
Carlos Alberto Heuser	Microsoft SQL Server
Carlos Alberto Heuser	Projeto de banco de dados
Isaac Epstein	Modelagem de dados
Isaac Epstein	Teoria da informação
Isaac Epstein	Mapping Web Sites
Isaac Epstein	Modelagem de dados
Isaac Epstein	Sistemas de informação gerenciais
Isaac Epstein	Microsoft SQL Server
Isaac Epstein	Projeto de banco de dados
Jane P. Laudon	Modelagem de dados

...

...

Fonte: Produção do próprio autor

Como nenhuma cláusula *WHERE* foi especificada, o SGBD entendeu que deveria combinar todos os dados da primeira tabela com todos os dados da segunda tabela. E produziu um produto de 9 autores x 7 livros = 63 resultados. Imagine a execução de um comando como esse em um banco de dados com milhares de linhas em cada tabela...

3.6.6.7.1 Atividade

Pesquise na *internet* o comando *JOIN* da linguagem SQL. Para que ele serve?

Resposta comentada

Ele serve para fazer a junção ou a ligação entre tabelas do banco de dados. Baseado em operações da teoria dos conjuntos, é capaz de fornecer como resultado o conjunto união, intersecção ou disjunção de duas ou mais tabelas.

3.6.6.8 *Aninhamento de consultas*

Sempre que trabalhamos com o comando *SELECT*, o SGBD retorna uma lista (esta também é uma relação, ou uma pequena tabela). Essa lista pode ser utilizada como entrada para outro comando de seleção. Chamamos a isso aninhamento.

Vamos ver uma situação de uso dessa característica.

Primeiramente, vamos retornar os CPFs das pessoas que tomaram livros emprestados.

Passo 29: Digite os comandos a seguir aplicando o comando *SELECT* e veja o resultado na imagem logo após.

SELECT CPF FROM emprestimos;

Figura 64 - Exemplo de resultado usando o comando *Select* para filtrar por CPF

cpf
31053735510
32377183352
65955322205
65955322205
65955322205
98302543055

Fonte: Produção do próprio autor

Agora, vamos utilizar o comando anterior como entrada em um comando para determinar os nomes dos usuários.

Passo 30: Digite os comandos a seguir aplicando o comando *SELECT DISTINCT* e veja o resultado na imagem logo após.

SELECT DISTINCT(nome) FROM Usuarios WHERE CPF IN (SELECT CPF FROM emprestimos) ORDER BY nome;

Figura 65 - Exemplo de resultado sem usar o comando *WHERE* buscando o nome dos usuários

nome
Antônio Calisto
Diana Bettocchi
Gabi Amorim Calisto
Tobias José Cretton

Fonte: Produção do próprio autor

O operador *IN* (em) determina que o comando deverá utilizar a lista fornecida como dados de entada (e será executado para cada elemento presente na lista). O *ORDER BY* determina a ordem de classificação. Como um usuário pode fazer mais de um empréstimo, há duplicação de nomes. Para evitar a duplicação, utilizamos o *DISTINCT*().

Assim, obtivemos uma lista dos usuários que tomaram livros emprestados. E os que não o fizeram? Poderíamos obter isso informando ao SGBD que queremos os nomes de usuários que não estão na lista de empréstimo, desta forma:

Passo 32: Digite os comandos a seguir aplicando o comando *SELECT DISTINCT* e *WHERE* aliado a *NOTbIN* e veja o resultado na imagem logo após.

```
SELECT DISTINCT(nome) FROM Usuarios WHERE CPF NOT IN
(SELECT CPF FROM emprestimos) ORDER BY nome;
```

Figura 66 - Exemplo de resultado usando o comando *SELECT DISTINCT* e *WHERE* buscando o nome dos usuários com base no CPF aplicando *NOT IN*

nome
Gisela Sarado

Fonte: Produção do próprio autor

3.6.6.9 *Trabalhar formatação de datas no comando select*

As datas de empréstimo e devolução que cadastramos em nossa tabela de empréstimos ficaram com formatação inglesa. Vamos verificar:

Passo 33: Digite os comandos a seguir aplicando o comando *SELECT DISTINCT* e veja o resultado na imagem logo após.

```
SELECT ISBN, CPF, data_emprestimo,data_devolucao FROM
emprestimos;
```

Figura 67 - Exemplo de resultado usando o comando *SELECT DISTINCT* e *WHERE* buscando o nome dos usuários com base no CPF aplicando *NOT IN*

ISBN	cpf	data_emprestimo	data_devolucao
9782880464646	65955322205	2014-11-20	2014-11-29
9798577803828	65955322205	2014-11-20	NULL
9782880464646	32377183352	2014-11-20	NULL
8508028121	98302543055	2014-11-21	2014-11-28
9788571948839	31053735510	2014-11-21	NULL
0074605755	65955322205	2014-11-21	NULL

Fonte: Produção do próprio autor

Para modificar essa apresentação, podemos fazer uso da função da função ***DATE_FORMAT()***. Essa função recebe dois parâmetros (duas variáveis): o primeiro é a data que desejamos formatar; e, o segundo, o formato desejado. Vamos utilizá-la:

Passo 34: Digite os comandos a seguir aplicando o comando *DATE_FORMAT* e veja o resultado na imagem logo após.

```
SELECT ISBN, CPF, DATE_FORMAT(data_emprestimo, '%d/%m/%Y')
AS Emprestimo, DATE_FORMAT(data_devolucao, '%d/%m/%Y')
AS Devolucao FROM emprestimos;
```

Figura 68 – Exemplo de resultado usando o comando *DATE_FORMAT*

ISBN	cpf	Emprestimo	Devolucao
9782880464646	65955322205	20/11/2014	29/11/2014
9798577803828	65955322205	20/11/2014	NULL
9782880464646	32377183352	20/11/2014	NULL
8508028121	98302543055	21/11/2014	28/11/2014
9788571948839	31053735510	21/11/2014	NULL
0074605755	65955322205	21/11/2014	NULL

Fonte: Produção do próprio autor

Já a função ***DATEDIFF()*** retorna a diferença entre duas datas fornecidas. Por exemplo, podemos verificar qual o período de devolução.

Passo 35: Digite os comandos a seguir aplicando o comando *DATEDIFF* e veja o resultado na imagem logo após.

```
SELECT nome, DATEDIFF(data_devolucao, data_emprestimo) AS
'Período em dias' FROM Usuarios, emprestimos WHERE Usuarios.
CPF = emprestimos.CPF;
```

Figura 69 - Exemplo de resultado usando o comando *DATEDIFF*

nome	Periodo em dias
Gabi Amorim Calisto	NULL
Diana Bettocchi	NULL
Tobias José Cretton	9
Tobias José Cretton	NULL
Tobias José Cretton	NULL
Antônio Calisto	7

Fonte: Produção do próprio autor

Também podemos utilizar a função ***CURDATE()***, que retorna a data corrente (ou atual) do sistema e combiná-la com outras datas. Vamos ver.

Passo 36: Digite os comandos a seguir aplicando o comando *CURDATE* e veja o resultado na imagem logo após.

```
SELECT CURDATE();
```

Figura 70 - Exemplo de resultado usando o comando *CURDATE*

CURDATE()
2014-12-02

Fonte: Produção do próprio autor

Passo 37: Digite os comandos a seguir aplicando o comando *CURDATE* para tornar a data corrente na tabela empréstimos e veja o resultado na imagem logo após.

```
SELECT nome, DATEDIFF(CURDATE(), data_emprestimo) AS 'Emprestado há... dias' FROM Usuarios, emprestimos WHERE Usuarios.CPF = emprestimos.CPF;
```

Figura 71 - Exemplo de resultado usando o comando *CURDATE* na tabela Empréstimos

nome	Emprestado há... dias
Gabi Amorim Calisto	11
Diana Bettocchi	12
Tobias José Cretton	12
Tobias José Cretton	12
Tobias José Cretton	11
Antônio Calisto	11

Fonte: Produção do próprio autor

Nota: o número de dias calculado nessa última execução é em função da data corrente fornecida, **02/12/2014**. Dessa forma, leve em conta a data de execução desse comando antes de se perguntar o porquê de os valores serem diferentes quando for testá-lo com os dados cadastrados em nossos exemplos. Nomes que aparecem mais de uma vez são referentes a usuários que possuem mais de um livro emprestado. Por fim, note que esses comandos utilizam funções para trabalhar com as datas, as quais poderão não estar disponíveis ou serem ligeiramente diferentes de um SGBD para outro.

Multimídia

Pesquise outras funções com datas e aprenda muito mais a respeito de SQL em: http://downloads.mysql.com/docs/refman-4.1-pt.a4.pdf

3.6.7 Campos de numeração automática

Como regra geral, a modelagem deverá ater-se às características, aos atributos dos objetos/entidades do mundo real. Dessa forma, não devemos, em princípio, criar algo que não exista na realidade, por exemplo, para identificar uma pessoa criar um código qualquer. Na prática, isso acaba ocorrendo, por facilidade ou desconhecimento.

Voltando ao caso da identificação da pessoa, em um banco de dados, podemos ter como chave o CPF, que é um documento que a pessoa possui e é universal, serve para integração com outros sistemas, por exemplo.

Mas, não é incomum que seja criado um identificador **código** e que, na documentação de sistemas, por exemplo, encontremos um campo cod_pessoa. Se seu sistema necessita da criação de um identificador, analise primeiro o contexto e verifique se não há como combinar dados para criar uma chave, o que poderia ser feito combinando-se nome+data de nascimento ou outra combinação única.

Se, mesmo assim, você precisar criar um identificador, pode usar campos de autonumeração. Veja como:

Passo 38: Digite os comandos a seguir aplicando o comando *CREATE TABLE* COMPONENTES para criar um identificador e veja o resultado na imagem logo após.

```
CREATE TABLE Componentes (
sequencia INT UNSIGNED NOT NULL AUTO_INCREMENT,
descricao CHAR(50) NOT NULL,
PRIMARY KEY (sequencia)
);
```

O campo **sequencia** será do tipo inteiro sem sinal, não admitirá nulos e será com autoincremento.

```
INSERT INTO Componentes(descricao) VALUES ('Resistor');
SELECT * FROM COMPONENTES;
```

Figura 72 - Exemplo de resultado usando o comando Componentes

sequencia	descricao
1	Resistor

Fonte: Produção do próprio autor

Adicione mais alguns itens.

Passo 39: Digite os comandos a seguir e veja o resultado na imagem logo após.

```
INSERT INTO Componentes(descricao) VALUES ('Capacitor'),
('Transistor'), ('Circuito integrado'), ('Diodo');
```

Note que, após a cláusula *VALUES*, colocamos vários valores, economizando a digitação do comando *INSERT*.

Veja como ficou (use o comando *SELECT*, que você já conhece) na imagem (Figura 73) a seguir.

Figura 73 - Exemplo de resultado usando o comando *SELECT*

	sequencia	descricao
▶	1	Resistor
	2	Capacitor
	3	Transistor
	4	Circuito integrado
	5	Diodo

Fonte: Produção do próprio autor

Desafio: procure no manual ou na *internet* uma solução para que os próximos valores de autoincremento iniciem em 100; execute a solução e insira alguns valores para testar.

Uma possibilidade seria a seguinte:

ALTER TABLE Componentes AUTO_INCREMENT = 100;

INSERT INTO Componentes(descricao) VALUES ('Varistor'), ('LED');

*SELECT * FROM Componentes;*

Figura 74 - Exemplo de resultado usando o comando *ALTER TABLE*

	sequencia	descricao
▶	1	Resistor
	2	Capacitor
	3	Transistor
	4	Circuito integrado
	5	Diodo
	100	Varistor
	101	LED

Fonte: Produção do próprio autor

Essa tabela de componentes eletrônicos foi criada somente para ilustrar os campos de autonumeração e, evidentemente, não faz parte da modelagem vista até aqui (Figura 34). Mas não se preocupe, pois na sequência ela será eliminada do banco de dados.

3.6.8 Apagar dados e destruir objetos

Para apagar o conteúdo de uma tabela, utilizaremos o comando *DELETE*. Após utilizado, o conteúdo da tabela será apagado, mas não sua estrutura.

Para destruir um objeto, ou seja, apagá-lo definitivamente do banco de dados, utilizamos o comando *DROP*. Após utilizá-lo, o objeto será totalmente eliminado; se efetuarmos o *DROP* em uma tabela, seu conteúdo e estrutura serão eliminados.

Atenção

CUIDADO! Os SGBDs não possuem um comando de *undelete*, de desfazer o que foi feito, como nos editores de texto e planilhas. Uma vez executada a transação com êxito, ela não poderá ser desfeita, ou seja, se você apagar os dados de uma tabela, ou a própria tabela, não há como recuperá-los com um comando. A única forma é o par *backup/restore*, **se o backup tiver sido executado antes** (que nem sempre fornecerá a última versão).

3.6.8.1 *Apagar dados da tabela Componentes*

Vamos apagar da tabela componentes todos os dados referentes a componentes cujo código seja menor do que 100.

Passo 40: Digite os comandos a seguir usando o comando *DELETE* para valores menores que 100 e veja o resultado na imagem logo após.

```
DELETE FROM Componentes WHERE sequencia < 100;
SELECT * FROM Componentes;
```

Figura 75 - Exemplo de resultado usando o comando *DELETE* para dados menores que 100

	sequencia	descricao
▶	100	Varistor
	101	LED

Fonte: Produção do próprio autor

Agora vamos apagar os dados restantes.

Passo 41: Digite os comandos a seguir usando o comando *DELETE* para valores maiores que 99 e veja o resultado na imagem logo após.

```
DELETE FROM Componentes WHERE sequencia > 99;
```

Observação: se você não estiver rodando no modo de segurança, pode executar ***DELETE FROM* Componentes**; para apagar todos os dados da tabela. Se estiver, o comando será bloqueado.

```
SELECT * FROM Componentes;
```

Figura 76 - Exemplo de resultado usando o comando *DELETE* para dados maiores que 99

	sequencia	descricao
*	NULL	NULL

Fonte: Produção do próprio autor

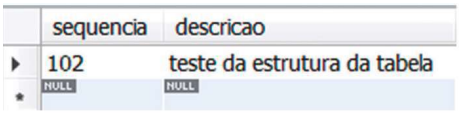
A tabela está vazia, porém, continua no banco de dados. Vamos verificar se a tabela Componentes ainda é funcional, apesar de ter tido todo o conteúdo apagado.

Passo 42: Digite os comandos a seguir e veja o resultado na imagem logo após.

INSERT INTO Componentes(descricao) VALUES ('teste da estrutura da tabela');

*SELECT * FROM Componentes;*

Figura 77 - Exemplo de resultado usando o comando *INSERT INTO* para verificar se os dados foram totalmente apagados do BD



Fonte: Produção do próprio autor

Comprovamos, então, que a estrutura da tabela está intacta. Somente seu conteúdo foi apagado ao utilizarmos o comando *DELETE*.

3.6.8.2 *Apagar um objeto do banco de dados*

Para retirar um objeto do banco de dados, utilizamos o comando *DROP* (que também pode ser utilizado para apagar o próprio banco de dados). Se utilizado para apagar um objeto do tipo tabela, por exemplo, ele apagará não somente seu conteúdo mas também sua estrutura; a tabela deixará de existir. Vamos retirar de nosso banco de dados a tabela Componentes.

Passo 43: Digite os comandos a seguir e veja o resultado.

DROP TABLE Componentes;

*SELECT * FROM Componentes;*

Retornará:

Error Code: 1146. Table 'pratica1.componentes' doesn't exist

Ou seja, a tabela Componentes não existe mais, desaparecendo com ela todo o seu conteúdo!

COMENTÁRIOS

Agora que já possuímos algum conhecimento básico da manipulação de bancos de dados utilizando a linguagem SQL, poderá surgir a pergunta: todos os bancos de dados exigem a execução de comandos SQL e, portanto, seu conhecimento? A resposta é sim e não. Se forem bancos de dados baseados no modelo relacional, e que implementem os comandos SQL, sim, será sempre necessário executar comandos nessa linguagem. Mas, não, não é essencial conhecer SQL sempre. Vamos tomar por exem-

plo o difundido *MS-Access*, que executa comandos SQL mas que permite ao usuário, por meio de interface gráfica de consulta, gerar os comandos e executá-los simplesmente selecionando ícones ou relações entre o conteúdo existente. Embora o usuário possa ver e alterar os comandos, pode trabalhar sem utilizá-los de forma direta (o SGBD sempre os utilizará).

Então, por que não estamos utilizando um gerenciador desse tipo? Porque o conhecimento da linguagem auxilia na ampliação do pensamento lógico, na busca por um melhor modelo e na utilização dos comandos que deverão ser colocados em outras aplicações. Por exemplo, páginas da *internet* escritas em linguagem PHP ou JAVA embutem comandos SQL, exatamente os praticados aqui, para retornar listas de dados que serão posteriormente formatadas e exibidas. Além disso, este material é voltado a um curso de bacharelado, o qual é voltado para o aprendizado de conceitos e a possibilidade de crescimento de aprendizado e pesquisas futuras.

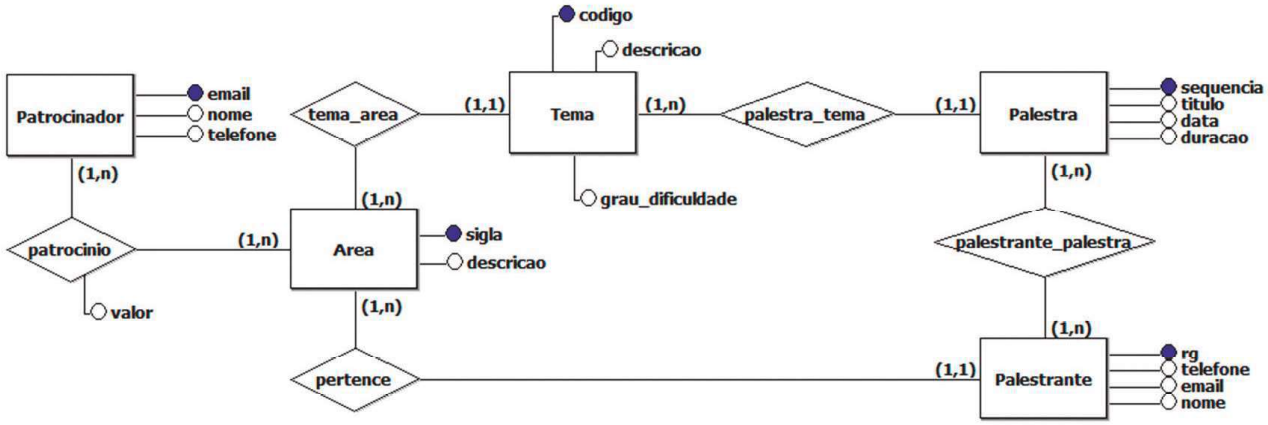
Como curiosidade, note que executamos o que é popularmente conhecido como CRUD, que é um acrônimo para as palavras da língua inglesa *Create, Read, Update e Delete* (ou *Destroy*), correspondendo, respectivamente à criação, leitura/recuperação, atualização e destruição de um dado; no SQL utilizamos respectivamente os comandos *INSERT, SELECT, UPDATE* e *DELETE* para realizar essas operações. Observe que essa sequência de operações (PINTO, 2000) implementa o ciclo de vida do dado ou da informação a ele vinculada.

3.6.8.3 Atividade

A situação referente ao exercício final da unidade anterior foi retomada em sua empresa (retorne àquela unidade e revise-a). Após algumas alterações, decidiu-se que, para idealizar o congresso desejado, cada área temática teria um ou mais patrocinadores, e que cada patrocinador teria a possibilidade de patrocinar mais de uma área, sempre implicando um determinado valor de auxílio. E que cada área poderia ter mais de um palestrante.

Também ficou estabelecido que uma palestra poderia contar com mais de um palestrante, e que um mesmo palestrante poderia realizar mais de uma palestra.

O MER foi refeito, ficando com o aspecto apresentado a seguir.



Os tipos de dados foram melhor definidos a partir da análise da documentação e de conversas com os interessados. Foram elaborados dicionários de dados, representados pelos quadros anteriores.

TABELA: Patrocinador					
CAMPO	IDENTIFICADOR	TIPO DE DADOS	TAMANHO	Valor mínimo	Valor máximo
Email	SIM	VARCHAR	50		
Nome	NÃO	VARCHAR	50		
Telefone	NÃO	VARCHAR	20		

TABELA: Area					
CAMPO	IDENTIFICADOR	TIPO DE DADOS	TAMANHO	Valor mínimo	Valor máximo
Sigla	SIM	VARCHAR	10		
Descricao	NÃO	VARCHAR	50		

TABELA: Tema					
CAMPO	IDENTIFICADOR	TIPO DE DADOS	TAMANHO	Valor mínimo	Valor máximo
Codigo	SIM	CHAR	15		
Descricao	NÃO	VARCHAR	50		
grau_dificuldade	NÃO	INT		0	10

TABELA: Palestra					
CAMPO	IDENTIFICADOR	TIPO DE DADOS	TAMANHO	Valor mínimo	Valor máximo
Sequencia	SIM	INT		1	
Titulo	NÃO	VARCHAR	50		
Data	NÃO	DATE			
Duracao	NÃO	INT		1	2

TABELA: Palestrante					
CAMPO	IDENTIFICADOR	TIPO DE DADOS	TAMANHO	Valor mínimo	Valor máximo
Rg	SIM	INT			
Telefone	NÃO	VARCHAR	20		
Email	NÃO	VARCHAR	50		
Nome	NÃO	VARCHAR	50		

Note que os tamanhos de campos e tipos de dados não são homogêneos; preste atenção à modelagem.

Definiram-se também quais são as consultas necessárias, chegando-se à seguinte lista:

a) todos os patrocinadores de cada área, ordenados por valor de contribuição;

b) todas as palestras de todos os palestrantes, ordenadas pelo nome do palestrante;

c) relação das palestras em ordem sequencial, informando-se o palestrante que a realizou. A data deverá ser no formato brasileiro e o cabeçalho deverá mostrar a palavra **Palestrante** para indicar seu nome;

d) uma relação de todas as palestras que foram realizadas a trinta dias ou menos a partir da data atual.

A partir dessas informações:

a) crie todos os comandos necessários à criação das tabelas;

b) apresente as declarações SQL referentes às operações de consulta necessárias.

Dica: lembre-se de que as ferramentas de modelagem podem gerar o código estrutural, de forma que você pode contar com o auxílio delas para criar as tabelas, restando apenas as consultas.

Resposta comentada

A solução que segue apresenta consultas que respondem às questões formuladas contra as tabelas criadas a partir das declarações que seguem. Se você criou tabelas de forma diferente, as consultas deverão ser adaptadas, evidentemente.

Para ter certeza de que seu modelo funciona, cadastre dados e teste-o.

Criação das tabelas:

SCRIPTS para criação das tabelas

```
CREATE TABLE Patrocinador (
email VARCHAR(50) PRIMARY KEY,
nome VARCHAR(50),
telefone VARCHAR(50)
);

CREATE TABLE Area (
sigla VARCHAR(10) PRIMARY KEY,
descricao VARCHAR(50)
);

CREATE TABLE Tema (
codigo CHAR(15) PRIMARY KEY,
descricao VARCHAR(50),
grau_dificuldade int
);

CREATE TABLE Palestrante (
rg int PRIMARY KEY,
telefone VARCHAR(20),
email VARCHAR(50),
nome VARCHAR(50)
);

CREATE TABLE Palestra (
sequencia INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
titulo char(50),
data DATE,
duracao int
);

CREATE TABLE patrocinio (
valor DECIMAL(8,2);
sigla VARCHAR(10),
email VARCHAR(50)
);

CREATE TABLE palestrante_palestra (
rg int,
sequencia int
);
```

a) Consultas

#1. todos os patrocinadores de cada área, ordenados por valor de contribuição;

SELECT descricao, nome, valor FROM Area,Patrocinador, patrocinio WHERE patrocinio.sigla = Area.sigla AND patrocinio.email = Patrocinador.email ORDER BY valor;

Explicação: seleciona (*SELECT*) as colunas descricao, nome e valor, a partir das tabelas (*FROM*) Area,Patrocinador, patrocínio, nas quais (*WHERE*) exista ligação entre a sigla do patrocínio e da área (patrocinio.sigla = Area.sigla) e (*AND*) do email (patrocinio.email = Patrocinador.email), ordenando (classificando) pelo valor (*ORDER BY* valor).

#2. todas as palestras de todos os palestrantes, ordenadas pelo nome do palestrante;

SELECT nome, titulo FROM Palestrante,Palestra, palestrante_palestra WHERE palestrante_palestra.rg = Palestrante.rg AND palestrante_palestra.sequencia = Palestra.sequencia ORDER BY nome;

Explicação: seleciona (*SELECT*) os campos nome e titulo a partir (*FROM*) das tabelas Palestrante,Palestra e palestrante_palestra, sempre que (*WHERE*) os números de RG forem iguais (palestrante_palestra.rg = Palestrante.rg) e (*AND*) as sequências também (palestrante_palestra. sequencia = Palestra.sequencia), ordenado posteriormente pelo nome do palestrante (*ORDER BY* nome);

#3. relação das palestras em ordem sequencial, informando-se o palestrante que a realizou. A data deverá ser no formato brasileiro e o cabeçalho deverá mostrar a palavra **Palestrante** para indicar seu nome

SELECT Palestra.sequencia, titulo, DATE_FORMAT(Palestra.data, '%d/%m/%Y') As 'Data', duracao, nome AS 'Palestrante' FROM Palestrante, Palestra, palestrante_palestra WHERE palestrante_palestra.rg = Palestrante.rg AND palestrante_palestra.sequencia = Palestra. sequencia ORDER BY Palestra.sequencia;

Explicação: seleciona (*SELECT*) os campos sequência, da tabela palestra (Palestra.sequencia) e, nesse caso, estamos informando a tabela, pois pode existir outro campo com o mesmo nome em outra tabela, o campo titulo, a data de realização, formatada pelo comando *DATE_ FORMAT*(Palestra.data, '%d/%m/%Y') e com o cabeçalho de **Data** (*AS* 'Data'), o campo duração, o campo nome com o cabeçalho de **Palestrante** (*AS* 'Palestrante') a partir das tabelas (*FROM*) Palestrante, Palestra e palestrante_palestra, limitando o resultado para quando os números de RG e sequência forem idênticos nas tabelas requeridas (*WHERE* palestrante_palestra.rg = Palestrante. rg *AND* palestrante_palestra.sequencia = Palestra.sequencia) e fornecendo o resultado por ordem de sequência cadastrada (*ORDER BY* Palestra.sequencia;)

#4. uma relação de todas as palestras que foram realizadas a 30 dias ou menos a partir da data atual.

SELECT Palestra.sequencia AS '#', Palestra.titulo AS 'Título', DATE_FORMAT(Palestra. data, '%d/%m/%Y') AS 'Data', Palestra.duracao AS 'Duração' FROM Palestra WHERE DATEDIFF(CURDATE(), Palestra.data) <31;

Explicação: seleciona o número da palestra com o cabeçalho de **#** no resultado (*SELECT* Palestra. sequencia *AS* '#'), o título da palestra com o cabeçalho **Título** (Palestra.titulo *AS* 'Título'), a data de realização no formato dia-mês-ano, com cabeçalho **Data** (*DATE_FORMAT*(Palestra.data, '%d/%m/%Y') *AS* 'Data'), e a duração da palestra com o cabeçalho **Duração** (Palestra.duracao *AS* 'Duração'), a partir da tabela Palestra (*FROM* Palestra), somente quando for a atendida a situação de que a diferença entre a data atual (*CURDATE*()) e a data da palestra for menor do que 31 (*WHERE DATEDIFF*(*CURDATE*(), Palestra.data) <31;), ou seja, para palestras ocorridas há 30 dias ou menos. Você deve notar aqui que, na execução de seu comando, serão consideradas a data atual do seu sistema e as datas cadastradas no seu banco de dados.

CONCLUSÃO

O trabalho com bancos de dados pode ser fascinante, com oportunidades de desenvolvimento profissional e da lógica necessária à elaboração de boas consultas. Nesta disciplina, tivemos a oportunidade de verificar o papel da modelagem conjunta com o usuário, realizar experimentos com diagramas MER e, especificamente nesta unidade, colocar em prática os conceitos para ver um banco de dados funcionar. Implica conhecer ferramentas de *software* e técnicas, aliar conceitos a atividades práticas, e praticar. Praticar muito.

Embora tenhamos passado nesta unidade pelos comandos básicos, os quais constituem grande parte do trabalho com bancos de dados, as variações possibilitadas pela SQL e a complexidade que alguns tipos de consulta e junções podem atingir nem de longe foi alcançada.

O que foi visto permite que você participe de discussões a repeito do planejamento e implementação de bancos de dados com conceitos suficientes para compreender o andamento do projeto e, ainda, poderá participar de projetos mais simples ou acompanhar os muitos exemplos existentes na imensa *internet*.

Cumprimos o papel de introduzir os conceitos e as práticas iniciais. Agora, cabe a você fazer mais por si e aprofundar seus conhecimentos. Para quem realmente deseja saber um pouco mais a respeito de bancos de dados, sugiro de imediato prosseguir nos estudos com normalização, criação de visões e o uso do comando *JOIN*.

RESUMO

A modelagem física é a ligação do modelo conceitual com a identificação de tipos de dados e chaves necessárias à implementação do banco de dados.

Para a implementação, por meio dos comandos de criação de dados (DDL, abreviatura inglesa para a linguagem de definição de dados) da linguagem de consulta estruturada, ou SQL, iremos enviar as declarações para processamento no SGBD.

Os comandos utilizados na criação de dados são o *CREATE DATABASE* e o *CREATE TABLE*. O primeiro é capaz de criar uma estrutura de banco de dados em um arquivo em disco, a ser gerenciado pelo SGBD. O segundo, cria as tabelas dentro do banco de dados. Para este último será necessário termos previamente uma boa definição dos metadados, tipos de dados, o que será considerado chave, o que poderá e o que não poderá ser nulo e domínios de valores a armazenar. Isso porque, se a tabela for definida de forma incorreta, posteriormente o armazenamento dos dados reais não será possível.

O conteúdo das tabelas é definido por declarações entre chaves ({ e }) e serão utilizados os tipos de dados compatíveis com o que se deseja armazenar e disponíveis no SGBD, por exemplo *INT* para números

inteiros e *CHAR* e *VARCHAR* para caracteres. Os tipos *CHAR* e *VARCHAR* deverão ter o tamanho máximo informado (quantidade de caracteres disponíveis para armazenamento), o que é realizado na forma *VARCHAR* (50), que possibilita cinquenta caracteres.

Após a criação dos objetos básicos no SGBD (banco e suas tabelas) o trabalho passará a ser realizado por meio dos comandos do grupo de manipulação de dados (DML na sigla inglesa).

São quatro os comandos básicos para trabalhar com a manipulação dos dados: *INSERT*, para inserir os dados no banco, *SELECT*, para recuperar os dados na forma de listas, *UPDATE*, para atualizar os dados após cadastrados, e *DELETE*, para apagar dados que não são mais necessários. Esses comandos são auxiliados por uma série de cláusulas e de funções, tais como para manipulação de datas ou geração de números sequenciais.

A utilização dos comandos *SELECT, UPDATE* e *DELETE* quase sempre vem acompanhada da cláusula *WHERE*, que permite especificar exatamente qual a linha ou conjuntos de linhas da tabela deverão ser recuperadas, atualizadas ou apagadas, respectivamente. A cláusula *WHERE* está fortemente ligada às chaves e aos identificadores utilizados na modelagem das tabelas do banco.

O comando *INSERT* possui em sua sintaxe a sequência *INSERT INTO* nome-da-tabela *VALUES* seguido dos valores dos dados a cadastrar.

O comando *SELECT* tem sintaxe básica *SELECT* quais-campos-selecionar *FROM* nome-da-tabela *WHERE* cláusulas-condicionais.

Para o comando *UPDATE* a sintaxe é *UPDATE* nome-da-tabela *SET* nome-do-campo = valor *WHERE* condição.

O comando *DELETE* possui a seguinte sintaxe: *DELETE FROM* nome-da-tabela *WHERE* condições.

Embora exista um comando, *ALTER*, que permite realizar algumas alterações nas tabelas, e mesmo sabendo-se que as estruturas criadas podem ser eliminadas, com o comando *DROP*, e depois recriadas, fica claro o papel e a importância da modelagem prévia, por exemplo, por meio de um MER, para o trabalho produtivo e eficiente com bancos de dados relacionais.

Além das questões conceituais, lembre-se de praticar os comandos. É a melhor forma de aprendê-los, pois o que se pode resumir é somente a sintaxe básica; a prática é que trará a aprendizagem.

Finalmente, note que os objetos, tais como o banco de dados ou as tabelas, são criados. Já os dados são manipulados. Os dados só são criados quando utilizamos operações de agrupamento ou matemáticas. Um exemplo é contar o número de livros cadastrados em uma tabela, por exemplo, por meio do comando *COUNT*. Porém, se estamos criando esse dado, na verdade, estamos criando uma informação, ou seja, estamos processando os dados (contando os livros) para criar uma informação que corresponda a esse número desejado (quantidade de livros). Note a ligação com seus conhecimentos teóricos prévios.

Sugestão de Leitura

CELKO, J. **SQL for smarties**: advanced SQL programming. 4. ed. [s.l.]: Morgan Kaufmann, 2010.

DATE, C. J. **Introdução a sistemas de banco de dados**. Rio de Janeiro: Campus, 2001.

HEUSER, C. A. **Projeto de banco de dados**. 6. ed. São Paulo: Bookman, 2009.

LARMAN, C. **Applying UML and patterns**: an introduction to object- oriented analysis and design and the unified process. Prentice-Hall: New Jersey, 2004.

PINTO, J. S. de P. Bancos de dados distribuídos, *middleware* e integração de aplicações. *In*: SOUZA, L. de; QUINÁIA, M. A.; VENSKE, S. M. S. (orgs.). **ERI 2007**. XIV Escola Regional de Informática SBC-PR. Guarapuava, PR: Gráfica Universitária Unicentro, 2007. v. 1, p. 199-224.

ULLMAN, J. D.; WIDOM, J. **A first course in database systems**. New Jersey: Prentice Hall, 1997.

REFERÊNCIAS

ELMASRI, R.; NAVATHE, S. B. **Sistemas de bancos de dados**. 6. ed. São Paulo: Pearson, 2011.

LAUDON, K. C.; LAUDON, J. P. **Sistemas de informação com internet**. 4. ed. Rio de Janeiro: LTC, 1999.

MACHADO, F.; ABREU, M. **Projeto de banco de dados**: uma visão prática. 11. ed. São Paulo: Érica, 2004.

MARTIN, J.; FILKELSTEIN, C. Engenharia da Informação: técnicas e abordagens de implementação. **Cadernos de Informática**, São Paulo, [s.n.], 1984. (Série James Martin)

NASCIMENTO, J. B. do. **Metodologias de desenvolvimento de sistemas**. São Paulo: Érica, 1993.

PINTO, J. S. de P. **SQL**: guia de consulta e aprendizagem. Rio de Janeiro: BookExpress, 2000.

PINTO, J. S. de P. **CGI**: guia de consulta e aprendizagem. Rio de Janeiro: BookExpress, 2001.

SERVA, M.; JAIME JR, P. Observação participante e pesquisa em administração: uma postura antropológica. **Revista de Administração de Empresas**, São Paulo, v. 35, n. 1, p, 64-79, maio/jun. 1995.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de banco de dados**. 5. ed. São Paulo: Elsevier, 2006.

STAIR, R. M. **Princípios de Sistemas de Informação**: uma abordagem gerencial. 2. ed. Rio de Janeiro: LTC, 1998.

SUSMAN, G. I.; EVERED, R. D. An assessment of the scientific merits of action research. **Administrative Science Quarterly**, v. 23, p, 582-603, Dec. 1978.

APÊNDICE A

APRESENTAÇÃO DA FERRAMENTA BR MODELO PARA MODELAGEM ENTIDADE *VERSUS* RELACIONAMENTO

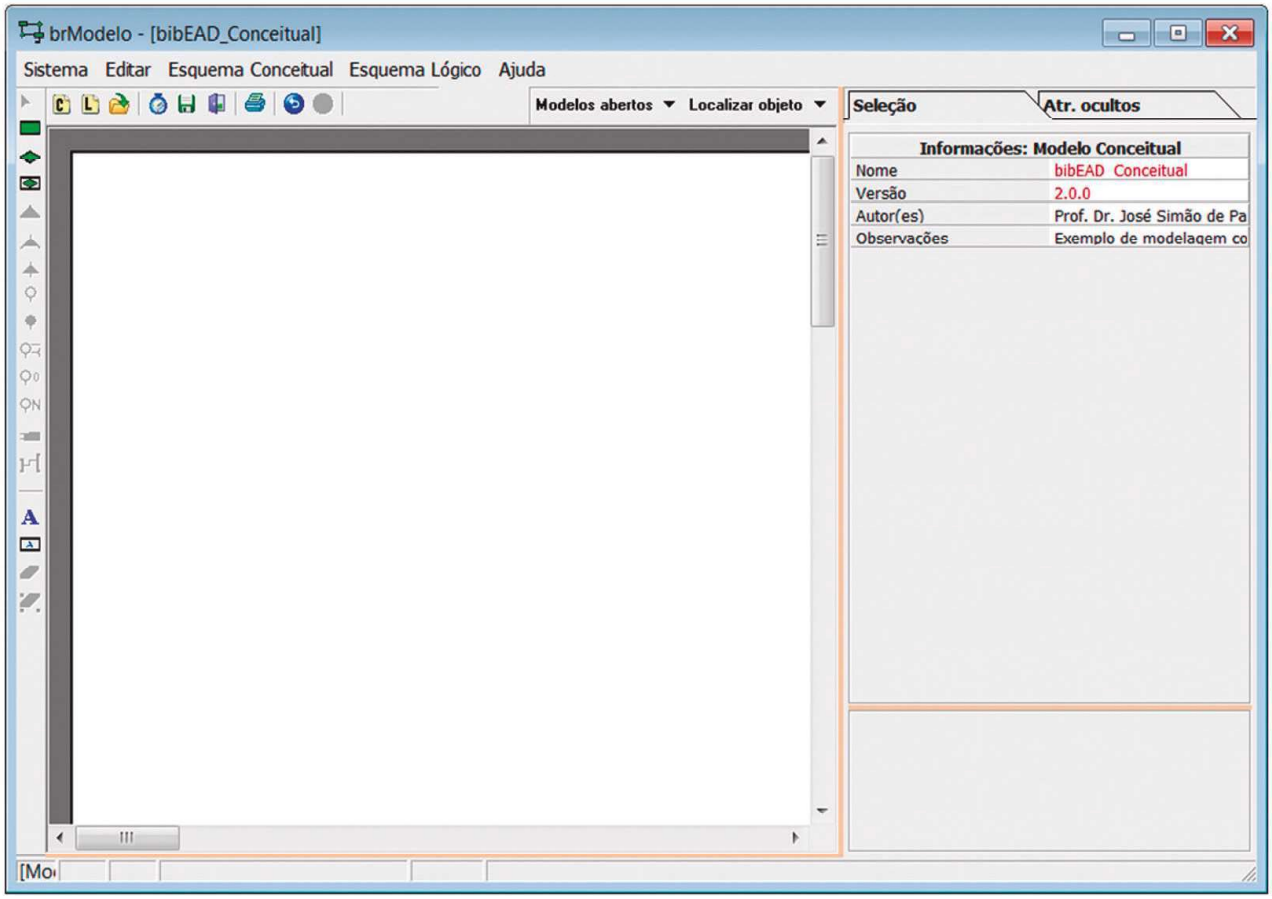
Existem várias ferramentas de *software*, pagas e gratuitas, disponíveis na *internet*, as quais são capazes de apoiar o desenvolvimento de diagramas que correspondem aos modelos físico e lógico de bancos de dados. Nesta disciplina, optou-se pela brModelo, que é disponível (infelizmente) somente para ambiente *Microsoft Windows*, o que não impede que seja utilizada diretamente no *Linux*, por exemplo, por meio do *Wine* ou de máquinas virtuais.

Não é o objetivo desta disciplina um curso da ferramenta, mas sim da modelagem. Porém, para facilitar seu trabalho de aprendizagem, nas páginas seguintes veremos uma rápida apresentação da ferramenta escolhida e de sua utilização no apoio à modelagem de um banco de dados. Alternativamente você poderá utilizar qualquer outra ferramenta que tenha à sua disposição, nesse caso, passando diretamente à modelagem.

A ferramenta em questão pode ser obtida no seguinte *link*: http://sis4.com/brModelo/. Acesso em 20 ago. 2014.

A imagem (Figura A.1) mostra a tela inicial da ferramenta.

Figura A.1 - Tela inicial da ferramenta brMODELO



Fonte: Produção do próprio autor[15]

Para trabalhar com essa ferramenta, dispomos de três grandes conjuntos de ícones.

Os Quadros A.1, A.2 e A.3, ao final desse texto, explicam suas funcionalidades.

Após baixar e instalar a ferramenta, para começarmos os trabalhos com ela, a partir da tela inicial (Figura A.1), vamos utilizar inicialmente as ferramentas de modelagem conceitual (as funções estão no Quadro A.2).

Veja uma sequência de trabalho (acompanhe pela Figura A.2):

a) clicar no ícone de criação de nova entidade;

b) 

;

c) posicionar o cursor no ponto desejado do diagrama e clicar; será criado um retângulo com o nome de Entidade_1;

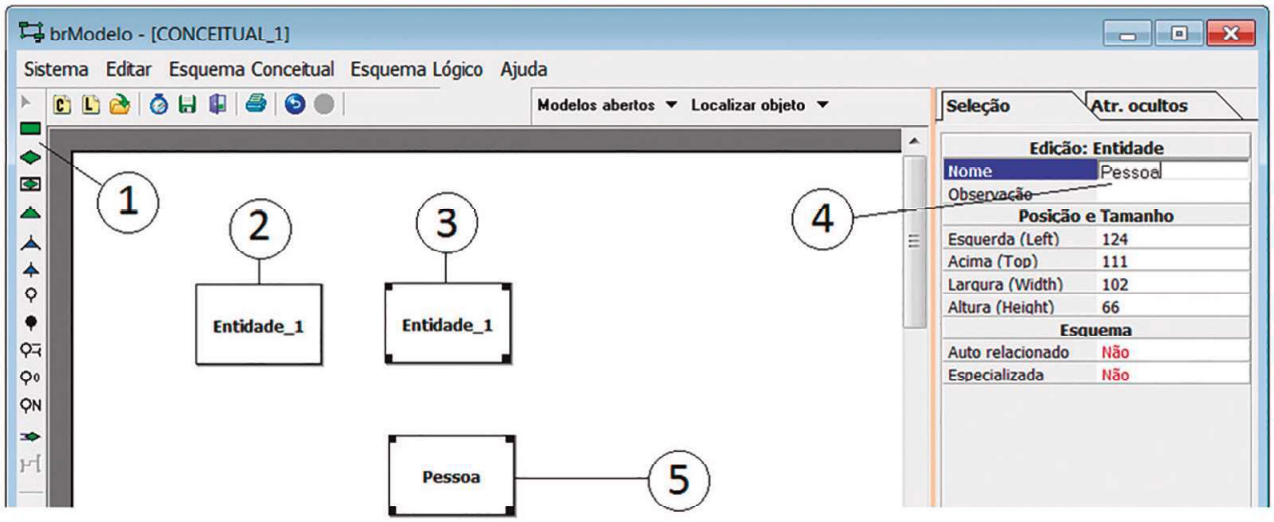
d) clique na Entidade_1, ela vai ficar selecionada;

[15] Cópia de tela da execução do programa brMODELO.

e) na guia de seleção, na edição dos atributos da entidade, clique sobre o nome e renomeie-o para Pessoa;

f) a entidade está criada e com o novo nome.

Figura A.2 - Utilização da ferramenta brMODELO para criação de uma nova entidade pessoa



Fonte: Produção do próprio autor[16]

Vamos agora criar atributos em nossa entidade Pessoa. O processo é parecido com o anterior, bem fácil de realizar. Vamos fazer assim (acompanhe pela Figura A.3):

a) clicar no ícone de atributo identificador (Atributo identificador);

b) clicar na entidade Pessoa;

c) selecionar o atributo, clicar no aba de mudança de edição e trocar o nome para CPF;

d) repetir os passos anteriores, porém clicando no ícone de criação de atributo (Criação de atributo), e criar e nomear o atributo nome;

e) clicar no ícone de atributo multivalorado (Atributo multivalorado) e criar o atributo telefone, conforme os passos anteriores;

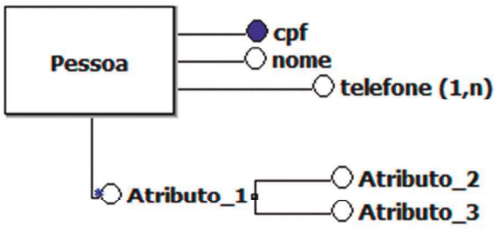
f) clique na entidade pessoa e arraste-a pela tela. Você verá que é possível posicioná-la onde quiser. Depois, pode ajustar os atributos (para ajustá-los em conjunto basta selecioná-los com o mouse);

g) clicar no ícone de atributo composto (Atributo composto) e criá-lo na entidade Pessoa.

Neste ponto, seu diagrama deverá ter uma entidade correspondente à da imagem (Figura A.3).

[16] Cópias de telas da ferramenta brMODELO editadas pelo autor.

Figura A.3 - Utilização da ferramenta brMODELO para criação de atributos na entidade pessoa



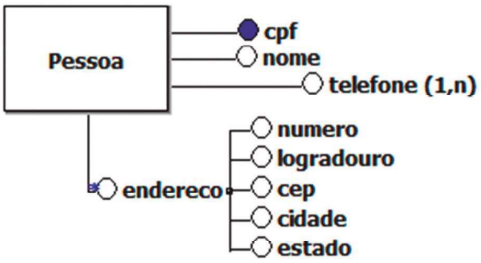
Fonte: Produção do próprio autor[17]

Para ajustar o atributo composto (Figura A.4):

a) clique primeiramente sobre o nome Atributo_1 e mude-o para endereco (acostume-se a evitar cedilhas e outras formas gráficas de acentuação nos modelos); na sequência ajuste o Atributo_2 para logradouro e o Atributo_3 para numero;

b) clique no ícone de criação de atributo e depois clique no atributo endereco; faça isso três vezes, para criar três novos atributos, renomeando-os depois para CEP, cidade e estado.

Figura A.4 - Utilização da ferramenta brMODELO para ajuste do atributo composto endereço na entidade pessoa



Fonte: Produção do próprio autor[18]

Vamos agora trabalhar com os atributos em função do dicionário de dados (Figura A.5):

Nosso dicionário de dados simplificado será o seguinte:

Quadro A.1 - Dicionário de dados

Campo	Identificador	Multivalorado	Opcional	Composto	Tipo de dado
CPF	SIM	NÃO	NÃO	NÃO	*int*
nome	NÃO	NÃO	NÃO	NÃO	*char(50)*
telefone	NÃO	SIM, máximo 3	SIM	NÃO	*char(20)*
endereco	NÃO	NÃO	NÃO	SIM	*char(148)*
endereco.logradouro	NÃO	NÃO	NÃO	NÃO	*char(50)*
endereco.numero	NÃO	NÃO	NÃO	NÃO	*char(20)*
endereco.cep	NÃO	NÃO	NÃO	NÃO	*int*
endereco.cidade	NÃO	NÃO	NÃO	NÃO	*char(50)*
endereco.estado	NÃO	NÃO	NÃO	NÃO	*char(20)*

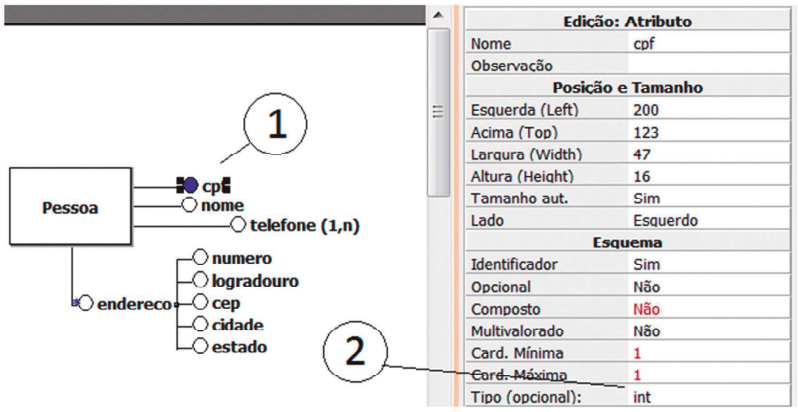
[17] Cópias de telas da ferramenta brMODELO editadas pelo autor.
[18] Cópias de telas da ferramenta brMODELO editadas pelo autor.

a) clique no atributo CPF;

b) clique na aba de edição de atributo;

c) informe o tipo de dados ***int*** no campo nomeado Tipo (opcional). Note as outras informações pertinentes, tais como se é ou não composto ou se é multivalorado. Experimente trocar a opção Identificador para Não e veja o que acontece no diagrama, depois retorne para Sim.

Figura A.5 - Utilização da ferramenta brMODELO para ajuste de características dos atributos conforme o dicionário de dados do modelo



Fonte: Produção do próprio autor[19]

Continue a edição para os demais atributos, seguindo o especificado no dicionário de dados passado. Para o caso do telefone, informe 3 na cardinalidade máxima.

Agora vamos criar um modelo lógico:

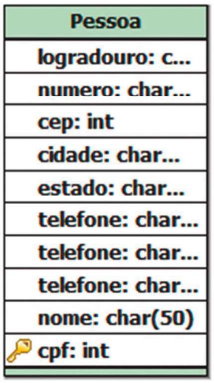
a) clique sobre a entidade Pessoa e, com o botão direito do mouse, selecione Gerar Esquema Lógico;

b) surgirá uma janela com três opções. Selecione a opção b) e clique em OK;

c) outra janela do assistente surgirá. Selecione novamente a opção b);

d) você criou o modelo lógico. Observe-o, explore-o, salve-o. Para voltar à janela do modelo conceitual use Modelos abertos.

Figura A.6 - Criação do modelo lógico a partir do modelo conceitual

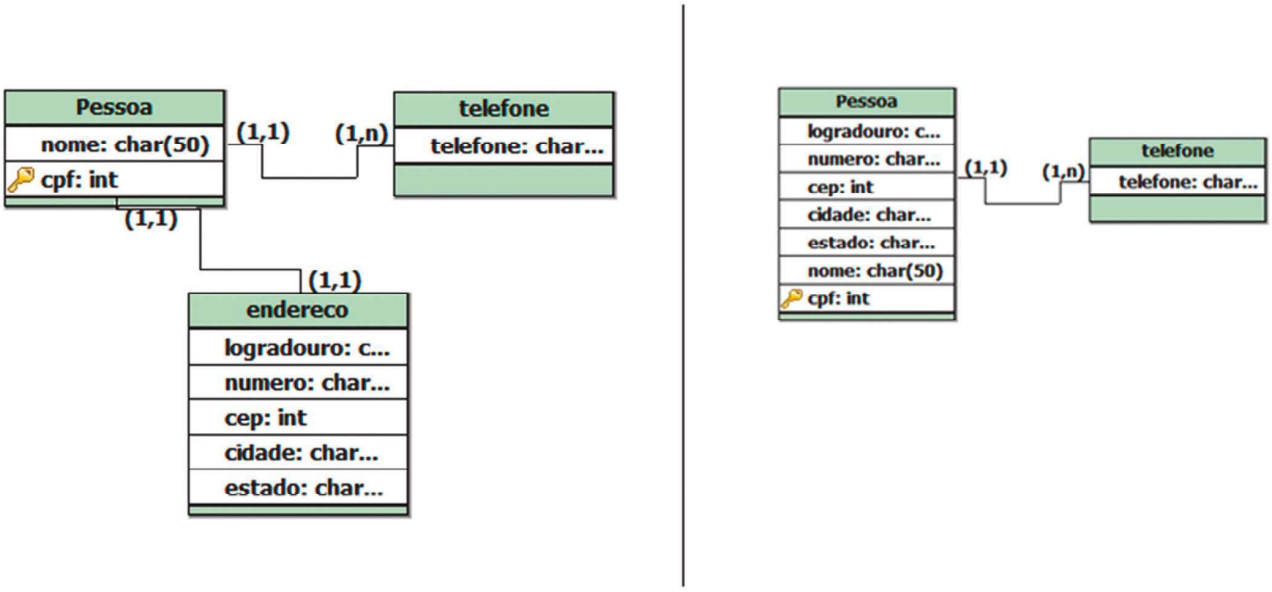


Fonte: Produção do próprio autor[20]

[19] Cópias de telas da ferramenta brMODELO editadas pelo autor.
[20] Cópias de telas da ferramenta brMODELO editadas pelo autor.

A escolha de outras opções durante a geração do modelo trará, evidentemente, resultados diferentes. Observe outras possibilidades na imagem (Figura A.7).

Figura A.7 - Criação do modelo lógico a partir do modelo conceitual: outras opções



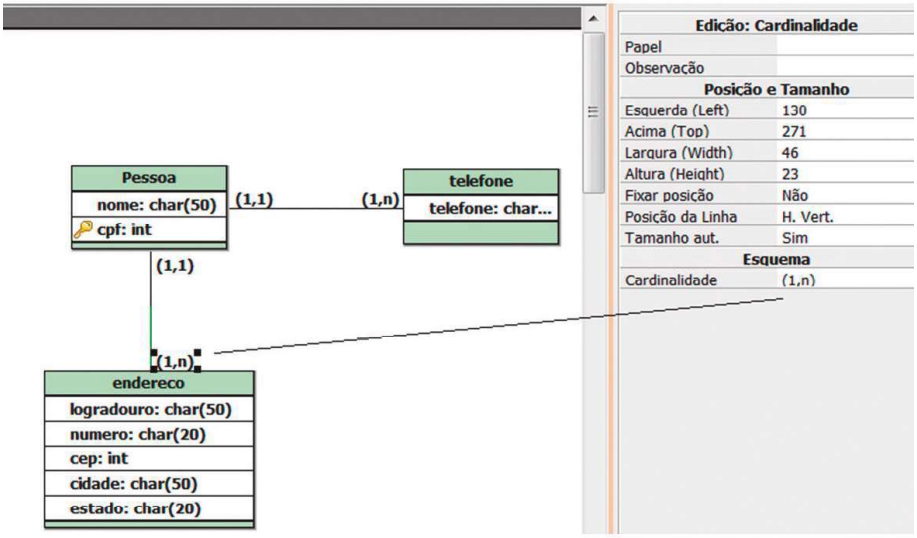
Fonte: Produção do próprio autor[21]

Rápida discussão: comparando-se as Figuras A.6 e A.7, vemos que o modelo lógico gerado em cada caso implicará termos uma única tabela (A.6), duas tabelas (A.7 direito) ou três tabelas (A.7 esquerdo) em nosso banco de dados. A vantagem de dividir é que podemos reaproveitar melhor os dados e criar possibilidades de conjuntos (conjuntos de endereço, de telefone) para cada pessoa. A desvantagem é que serão mais tabelas a unir no momento da consulta. É uma decisão de projeto; todas são funcionais.

Volte ao modelo conceitual (Modelos abertos) e gere um novo modelo lógico (clique na entidade Pessoa e clique o botão direito do mouse, selecionando Gerar Esquema Lógico). Responda **a)** e **a)** para as opções apresentadas pelos assistentes (criar tabelas separadas), de forma a gerar um esquema equivalente ao da Figura A.7, lado esquerdo. Ajuste a cardinalidade da tabela endereço para (1,n). Acompanhe o resultado pela imagem (Figura A.8). Note que, se você não conseguiu o resultado esperado na primeira interação, sempre poderá realizar novamente os passos, até aprender. O objetivo é esse. Não tenha medo de errar nem se preocupe com o fato de ter de fazer mais de uma vez, cada pessoa tem um ritmo e uma concentração.

[21] Cópias de telas da ferramenta brMODELO editadas pelo autor.

Figura A.8 - Utilização da ferramenta brMODELO para gerar modelo lógico: cardinalidade ajustada

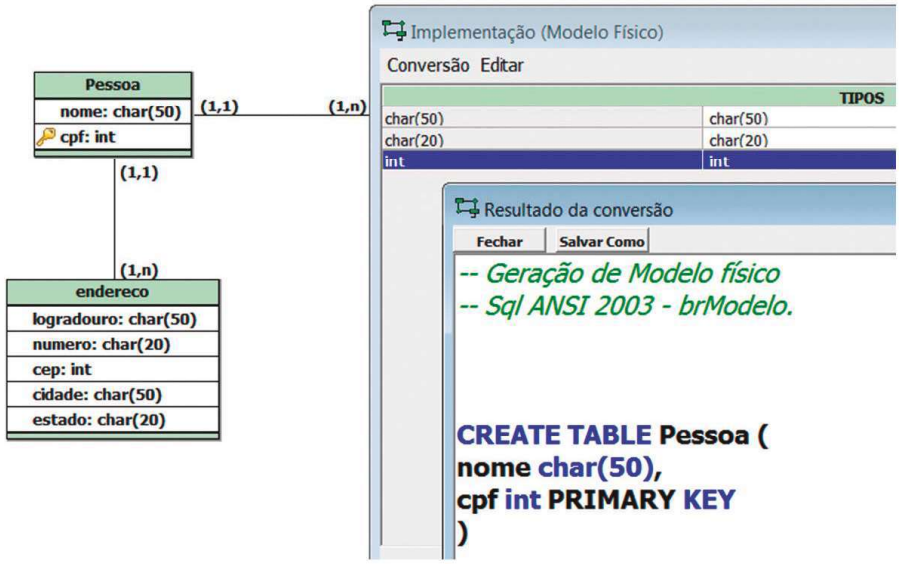


Fonte: Produção do próprio autor[22]

Assim, teremos agora a possibilidade de que a pessoa possua mais de um endereço, digamos um residencial e um comercial (e deveríamos ter um identificador de qual é, mas deixaremos para depois).

Agora vamos gerar o esquema físico. Clique com o botão direito do mouse sobre qualquer ponto do diagrama e selecione Gerar Esquema Físico. Clique em Conversão e Converter para confirmar a conversão de dados solicitada e você terá o esquema físico gerado. O arquivo com as declarações pode ser salvo e posteriormente aberto dentro de sua ferramenta de trabalho com o SGBD. Acompanhe o resultado dessas operações pela imagem (Figura A.9).

Figura A.9 - Utilização da ferramenta brMODELO para gerar o esquema físico



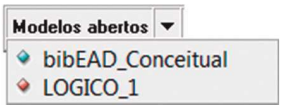
Fonte: Produção do próprio autor[23]

[22] Cópias de telas da ferramenta brMODELO editadas pelo autor.
[23] Cópias de telas da ferramenta brMODELO editadas pelo autor.

O esquema físico será utilizado na Unidade 3 desta disciplina.

Agora que você já possui o básico, pratique mais um pouco e volte para a Unidade 2 da disciplina, para aprender mais a respeito de modelagem conceitual.

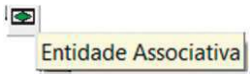
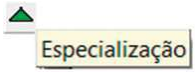
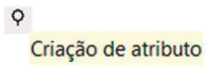
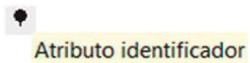
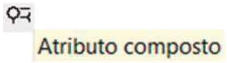
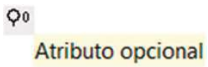
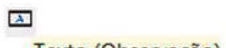
Quadro A.2 - Funções dos ícones da barra de ferramentas básicas do *software* brMODELO

Ícone	Finalidade
 Novo modelo conceitual (Ctrl+N)	Cria um novo diagrama de modelagem conceitual. Não salva o anterior automaticamente, caso tenha sido realizada alguma modificação nele.
 Novo modelo lógico (Ctrl+L)	Cria um novo diagrama de modelagem lógica. Não salva o anterior automaticamente, caso tenha sido realizada alguma modificação nele.
 Abrir modelo (Lógico ou Conceitual)	Carrega para edição os arquivos de modelos conceituais e lógicos anteriormente salvos.
 Salvar o modelo automaticamente em tempos pré configurados	Permite que os modelos sob edição sejam automaticamente salvos em disco em intervalos de tempo configuráveis pelo usuário. Sistema/Configurações do sistema/Aba de autossalvamento. O tempo deverá ser informado, em minutos. O padrão, caso não seja alterado, é de salvamento a cada 5 minutos.
 Salvar modelo	Guarda em arquivo digital o diagrama exibido na tela.
 Fechar modelo	Fecha a exibição e a edição do arquivo atual, perguntando antes se é necessário salvar alterações, caso tenham ocorrido.
 Imprimir modelo	Abre a tela do gerenciador de impressão, que permite configurar impressoras e a forma de impressão, assim como comandar a impressão do diagrama exibido.
 Desfazer [3] (Ctrl+Z)	Desfaz a última alteração realizada no diagrama em edição. O atalho para a mesma função, padrão no *Windows*, é a combinação das teclas CTRL + Z.
 Refazer [1] (Ctrl+R)	Permite desfazer o trabalho da função desfazer, caso essa função tenha sido clicada por engano. O atalho para a mesma função, padrão no *Windows*, é a combinação das teclas CTRL + Z. ATENÇÃO: essa tecla pode trazer a mensagem "*Stream read error*" e não funcionar, tratando-se de um *bug* do programa.
	Permite alternar a tela de edição entre os diagramas abertos.
	Permite a localização rápida dos objetos adicionados no diagrama aberto para edição (se houver mais de um diagrama aberto, localiza os objetos no diagrama que está em exibição).

Fonte: Produção do próprio autor[24]

[24] Elaboração do autor a partir das características e cópias de telas da ferramenta brMODELO.

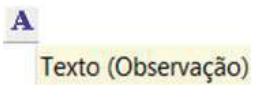
Quadro A.3 - Funções dos ícones da barra de ferramentas para modelagem conceitual do *software* brMODELO

Ícone	Finalidade
	Cancela a seleção de operação (ícones a seguir). Somente fica habilitado se for clicado um dos ícones da barra, comentados na sequência.
	Criar uma nova entidade no diagrama. Por exemplo, criar a entidade pessoa ou a entidade livro.
	Criar um relacionamento entre entidades do diagrama. Por exemplo, criar o relacionamento de empréstimo entre uma entidade pessoa e uma entidade livro.
	Criar uma entidade associativa no diagrama. Por exemplo, criar uma entidade empréstimo, relacionando pessoas com livros. A diferença desta para a anterior (criar relação) é que, na associação, será criada uma tabela que existe sempre, enquanto, na relação, poderá ou não existir uma tabela.
	Criar uma especialização de uma entidade no diagrama. Por exemplo, a partir de uma entidade pessoa, criar as especializações pessoa física e pessoa jurídica. Esse conceito amplia as possibilidades de modelagem e torna os SGDBs mais próximos à modelagem orientada a objetos, comumente utilizada hoje no desenvolvimento de sistemas de informação.
	Permite a utilização da modelagem estendida, com a criação de conceitos de herança advindos da orientação a objetos.
	Permite a utilização da modelagem estendida, com a criação de conceitos de herança advindos da orientação a objetos.
	Cria um atributo na entidade selecionada. Permite atribuir as características da entidade. Por exemplo, criar os atributos nome e gênero para uma entidade pessoa.
	Permite a criação de um atributo que será o identificador ou a chave da tabela.
	Permite a criação de um atributo que, na verdade, é formado por um conjunto de outros atributos. Endereço é um caso típico.
	Atributo que será posteriormente marcado como sendo um campo que pode ser nulo, ou seja, que não deverá estar sempre preenchido/sendo informado.
	Um atributo que poderá assumir vários valores, como o telefone (pode haver mais de um). Na transformação do diagrama para tabelas poderá ser representado por uma tabela à parte.
	Permite a criação de um relacionamento da tabela com ela mesma. Por exemplo, no caso de uma tabela empregado, podemos ter na modelagem a especificação de que **empregado é chefe de empregado**. O relacionamento **é chefe de** representa a ligação da tabela empregado com ela mesma.
	Efetua a ligação dos objetos selecionados no diagrama em exibição.
	Permite colocar um texto com observações ou destaques no diagrama em edição. O texto poderá ter uma moldura configurável, veja o ícone seguinte.
	Permite colocar um texto com observações ou destaques no diagrama em edição. O texto será destacado por um retângulo configurável, que poderá ser retirado, ficando o texto com a aparência obtida por meio do ícone anterior.
	Permite apagar objetos a serem selecionados, inclusive de forma parcial quando se tratar de um conjunto. Por exemplo, podemos apagar com essa ferramenta um campo de uma tabela.
	Apaga o objeto que estiver selecionado, seja conjunto ou único. A diferença em relação ao ícone anterior é que. para que essa ferramenta funcione, o objeto a ser apagado já deverá ter sido previamente selecionado.

Fonte: Produção do próprio autor[25]

[25] Elaboração do autor a partir das características e cópias de telas da ferramenta brMODELO.

Quadro A.4 - Funções dos ícones da barra de ferramentas para modelagem lógica da ferramenta brMODELO

Ícone	Finalidade
 Cancelar	Cancela a seleção de operação (ícones a seguir). Somente fica habilitado se for clicado um dos ícones da barra, comentados na sequência.
 Modelo lógico: Criar Tabela	Permite a criação de tabelas no diagrama correspondente ao modelo físico (as tabelas representarão as entidades, associações e especializações do modelo conceitual).
 Modelo lógico: Criar Relacionamento	Cria relacionamentos entre tabelas, utilizando campos destas (chaves) para garantir sua integridade (os campos correspondem aos atributos do modelo conceitual). Permite especificar a cardinalidade da relação (a quantidade que participa da relação, por exemplo, um livro possui um ou mais autores: cardinalidade de 1 para n).
 Criar Campo	Cria um campo de dados (atributo, no modelo conceitual) na tabela selecionada. Permite que seja especificado/modificado, se é chave primária ou estrangeira e o tipo de dados correspondente (número, caractere).
 Criar Campo Chave Estrangeira	Cria um campo que é identificador único de outra tabela. Por exemplo, em uma tabela de autores, uma chave estrangeira poderia ser a que corresponde ao identificador de uma obra, digamos o ISBN de um livro. As tabelas podem ter mais de uma chave estrangeira.
 Criar Campo Chave Primária	Cria um campo em uma tabela cujo valor é capaz de identificar unicamente um elemento de sua composição. Por exemplo, em uma tabela de livros, um campo identificador pode ser o ISBN. Quando esse campo é referenciado por outra tabela (por exemplo, uma de autores de livros), nessa outra tabela, ele é chamado de chave estrangeira (veja o ícone anterior). Somente pode haver uma chave primária por tabela, embora ela possa ser composta (por exemplo, nome + nome da mãe + data de nascimento para uma identificação de pessoa).
 Criar um separador de campos	Cria um separador gráfico no desenho da tabela. Só tem utilidade na melhoria da visibilidade dos campos, não influencia o modelo lógico do banco de dados em si.
 Texto (Observação)	Permite colocar um texto com observações ou destaques no diagrama em edição. O texto poderá ter uma moldura configurável. Veja o ícone seguinte.
 Texto (Observação)	Permite colocar um texto com observações ou destaques no diagrama em edição. O texto será destacado por um retângulo configurável, que poderá ser retirado, ficando o texto com a aparência obtida por meio do ícone anterior.
 Apagar	Permite apagar objetos a serem selecionados, inclusive de forma parcial, quando se tratar de um conjunto. Por exemplo, podemos apagar com essa ferramenta um campo de uma tabela.
 Excluir o objeto selecionado	Apaga o objeto que estiver selecionado, seja conjunto ou único. A diferença em relação ao ícone anterior é que, para que essa ferramenta funcione, o objeto a ser apagado já deverá ter sido previamente selecionado.

Fonte: Produção do próprio autor[26]

[26] Elaboração do autor a partir das características e cópias de telas da ferramenta brMODELO.

APÊNDICE B

APRESENTAÇÃO DA FERRAMENTA MYSQL WORKSHOP PARA TRABALHO COM SQL E BANCOS DE DADOS RELACIONAIS MYSQL

Existe grande variedade de produtos de *software* que podem ser utilizados para a prática de bancos de dados. Entre eles encontramos opções pagas e gratuitas. Escolhemos para esta disciplina um gerenciador que possui versão livre para estudantes e práticas acadêmicas (e paga para o mercado): o *MySQL*. Esse SGBD foi escolhido por sua simplicidade de utilização e grande popularidade no meio acadêmico e comercial, mas, caso você possua acesso a *Oracle*, *DB2*, *PostgreSQL*, *SQLite*, *MS-SQL Server*, *Microsoft Access* ou outro banco de dados que aceite comandos SQL poderá utilizá-lo para as práticas que seguem. Eventualmente, pequenos ajustes de sintaxe/digitação serão necessários.

Se quiser utilizar o banco aqui sugerido, para baixar o SGBD foi a acessada a página: http://dev.mysql.com/downloads/repo/apt/. O último acesso foi realizado em agosto de 2014, de forma que, quando você estiver acessando, poderá existir uma versão mais nova; em caso de problemas, consulte a seção de *downloads* a partir da página inicial do *MySQL*.

Também foi instalado o *MySQL Workbench*, para facilidade de acesso ao servidor por meio de uma interface gráfica, disponível na página indicada.

Este material foi desenvolvido em *Linux*, distribuição *Ubuntu 14.04 LTS*, mas a sintaxe aqui utilizada funciona igualmente em ambiente *MS-Windows* (em outros SGBDs, como já comentado, poderá haver mínimas diferenças nos comandos e, certamente, haverá nas telas).

- **INSTALAÇÃO DO *MySQL***

  Se o *MySQL* não estiver instalado, poderá sê-lo a partir do gerenciador de programas ou a partir dos comandos usuais que, no Ubuntu, por exemplo, são (lembre-se de que você precisa da senha de *root*, use sudo na frente do apt-get se não estiver logado como *root*):

  # apt-get install mysql-server

A configuração e o estabelecimento do serviço podem ser realizadas com:

a) # mysql_install_db

b) *# service mysqld start*

c) *# chkconfig mysqld on*

d) *# mysql_secure_installation*

e) *# service mysql restart*

E, finalmente, atribuir uma senha para o usuário root do SGBD:

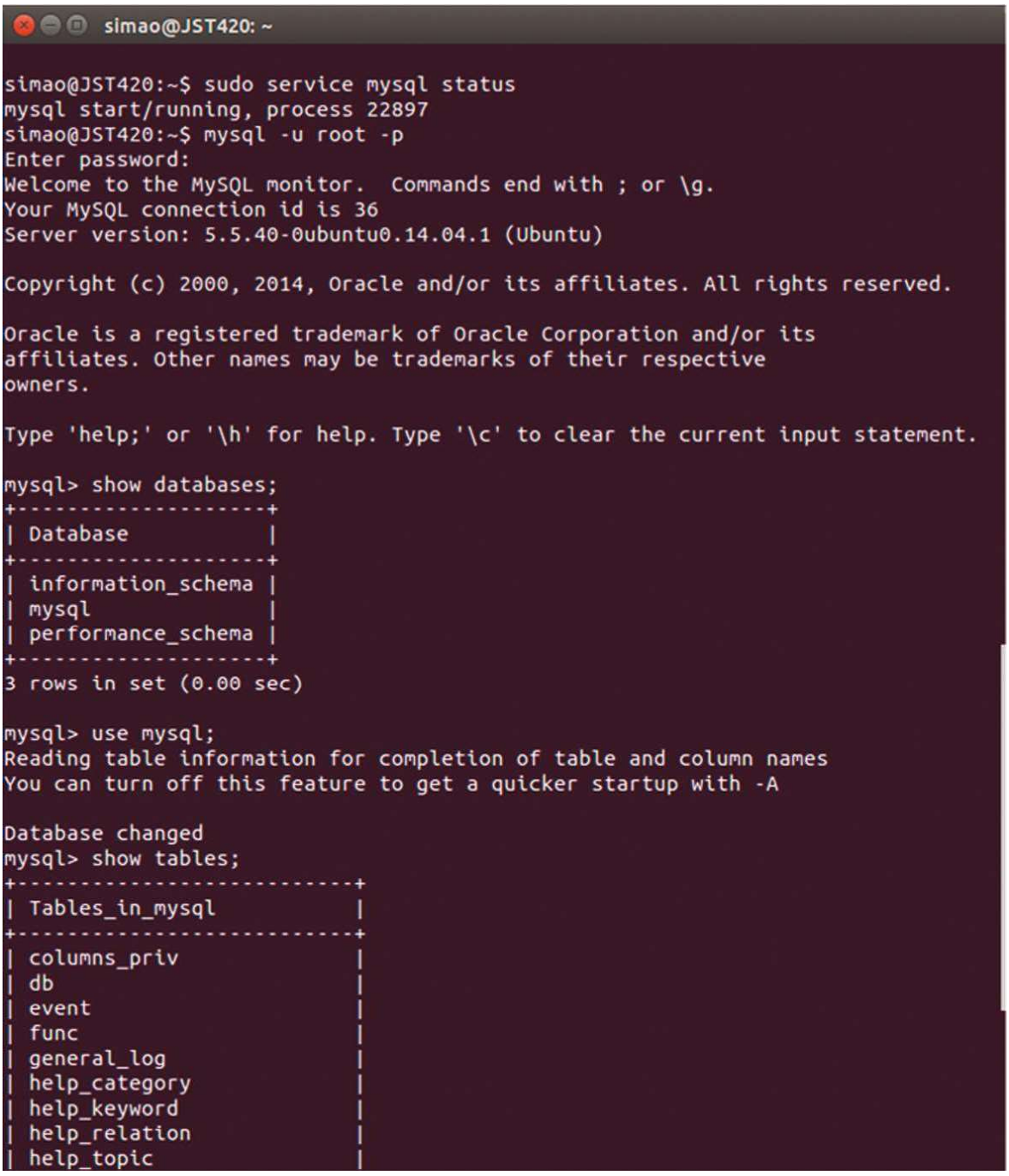
 # mysqladmin -u root password **coloque aqui sua senha**

No *MS-Windows* a instalação poderá ser realizada a partir do programa baixado na seguinte página: https://dev.mysql.com/downloads/workbench/ (acesso em: dez. 2014).

Após baixá-lo, basta iniciá-lo e responder às questões do assistente.

Executados esses passos, o SGBD estará instalado e pronto para funcionar. A partir de agora, você já pode utilizar seu *MySQL*, basta digitar os comandos no terminal, no caso do *Linux* (veja Figura B.1), ou na janela de comandos, se for *MS-Windows*.

Figura B.1 - Execução de comandos no *MySQL* em interface texto por meio do terminal no sistema *Ubuntu*



Fonte: Produção do próprio autor[27]

- **INSTALAÇÃO DO *MySQL WORKSHOP***

Para melhorar a usabilidade, podemos instalar o ambiente gráfico *MySQL Server Workshop*. Vejamos como.

No *Ubuntu*, ele pode ser obtido em http://dev.mysql.com/downloads/repo/apt/ (acesso em: ago. 2014) e executado com o gerenciador de pacotes, por exemplo.

No *MS-Windows*, acesse: http://dev.mysql.com/downloads/installer/ (acesso em: dez. 2014), baixe o programa e depois execute-o. Bastará seguir as instruções do assistente.

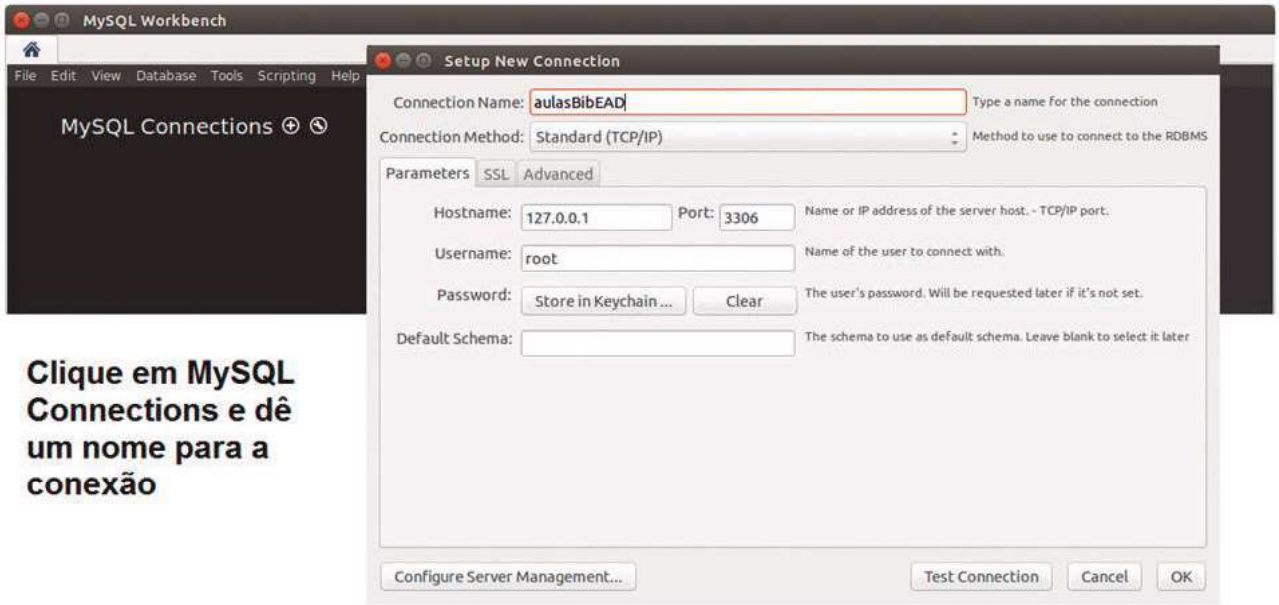
[27] Cópia de tela durante a execução do terminal no equipamento do autor.

A seguir, estão cópias de telas de algumas operações a serem realizadas no *Workbench*. Antes, uma observação. Os programas evoluem. E muito, muito rapidamente. Dessa forma, não é improvável que a interface do programa que você está utilizando seja diferente; com certeza será se a versão for antiga. Se isso acontecer, verifique o *help*, a ajuda do programa, ou seu manual (por exemplo, http://dev.mysql.com/doc/index.html), para obter um auxílio inicial. De qualquer forma, é improvável que os comandos mudem, somente a forma de obtê-los, eventualmente, deverá mudar.

2.1 CRIAR CONEXÃO NO *MYSQL WORKSHOP*

Para que possamos enviar as declarações de comandos ao SGBD, deverá haver uma conexão de rede. Ela deverá ser criada para cada SGBD desejado e, uma vez criada, ficará disponível na tela inicial do programa. Para efetuá-lo, basta clicar em ***MySQL Connections***, fornecer um nome e confirmar. Acompanhe pela imagem (Figura B.2).

Figura B.2 - Criação de conexão com o *MySQL* dentro do *workshop*

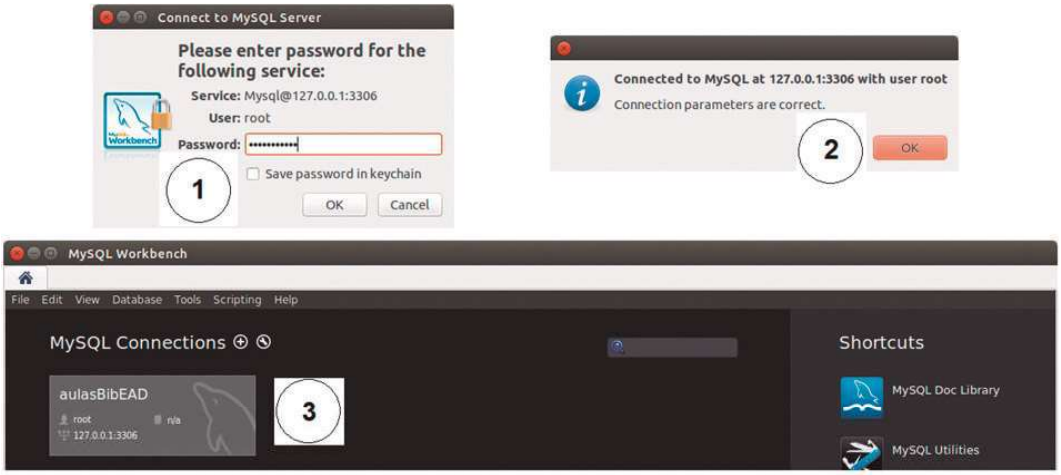


Fonte: Produção do próprio autor[28]

Após nomear a conexão, forneça a senha (1) e crie a conexão de trabalho confirmando (2); ela ficará disponível na tela inicial (3) do programa (os números correspondem à sequência marcada na Figura B.3).

[28] Cópia de tela durante a execução do terminal no equipamento do autor.

Figura B.3 - Fornecer senha e confirmar conexão com o *MySQL*



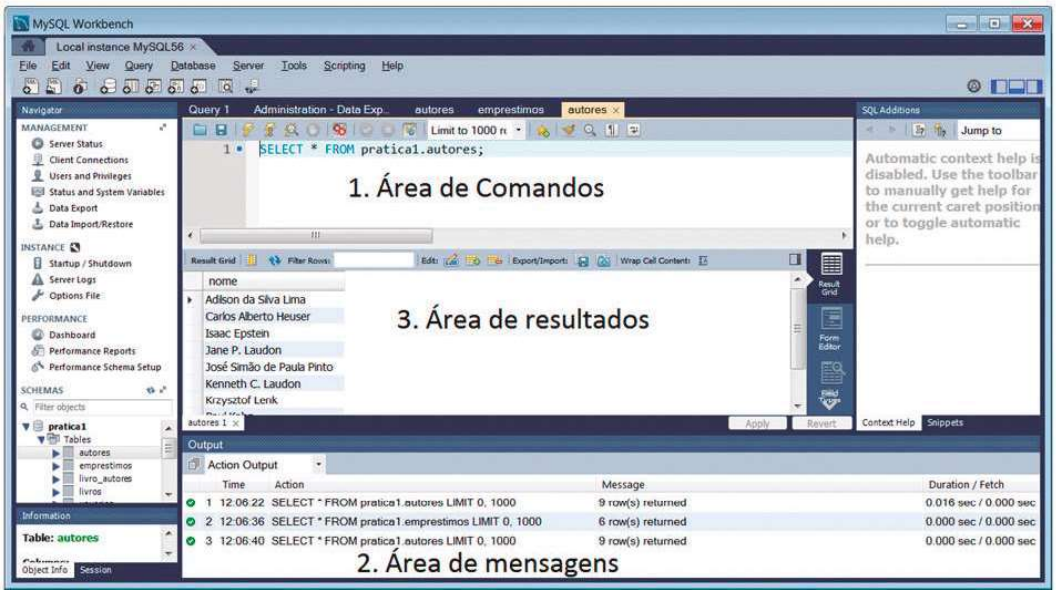
Fonte: Produção do próprio autor[29]

2.2 TRABALHAR NO *MYSQL WORKBENCH*

Para trabalhar com o programa, após logados, teremos uma tela com várias opções, úteis à edição de comandos e gerenciamento do SGBD e seus bancos de dados.

Uma visão geral pode ser vista na imagem (Figura B.4). Somente os principais pontos de interesse serão explorados neste apêndice; consulte o manual do produto para mais informações (disponível em: http://downloads.mysql.com/docs/workbench-en.a4.pdf).

Figura B.4 - Visão da tela do *MySQL Worbench* após logado



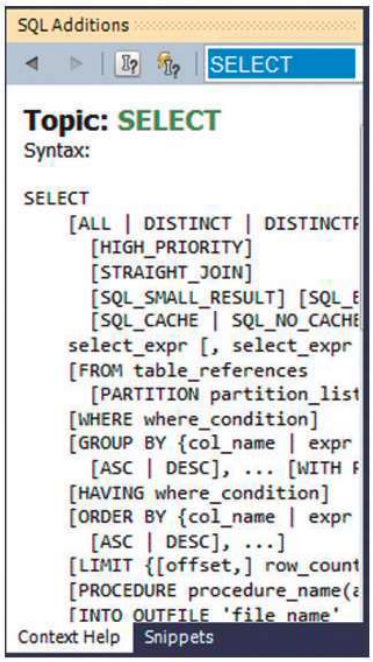
Fonte: Produção do próprio autor[29]

[29] Montagem do autor a partir de cópias de telas do programa.

As três áreas de interesse inicial visíveis na Figura B.4 são as áreas nas quais serão executados os comandos (1), a de mensagens do sistema (2) e aquela na qual serão apresentadas as listas de respostas aos comandos (3).

Além das três áreas comentadas, localiza-se ao lado delas uma tela com auxílio sensível ao contexto (Figura B.5), útil para lembrar a sintaxe dos comandos e muito adequada a iniciantes (toda a sintaxe é exibida em inglês).

Figura B.5 - Auxílio sensível ao contexto



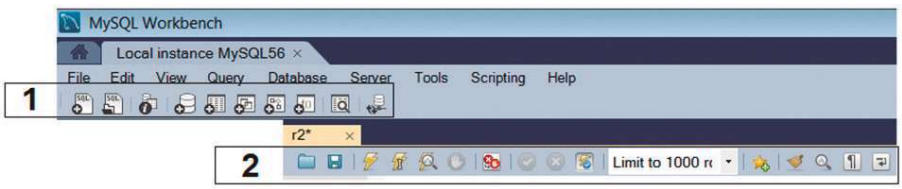
Fonte: Produção do próprio autor[30]

Para executar um comando, ele pode ser digitado (ou colado, ou obtido de um arquivo previamente salvo) na área 1 da Figura B.4. Também poderá vir de um arquivo.

No momento da execução, o comando deverá estar SELECIONADO. Se não houver um comando selecionado, TODOS os comandos de todas as linhas presentes na área de comandos serão executados.

Para auxiliar a execução das atividades, temos duas barras de ferramentas contendo vários ícones (1 e 2 na Figura B.6). Vamos ver suas funções.

Figura B.6 - Barras de ferramentas do *MySQL Workbench* e seus ícones

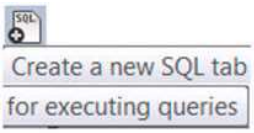
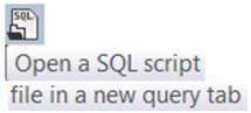
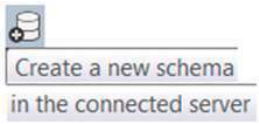
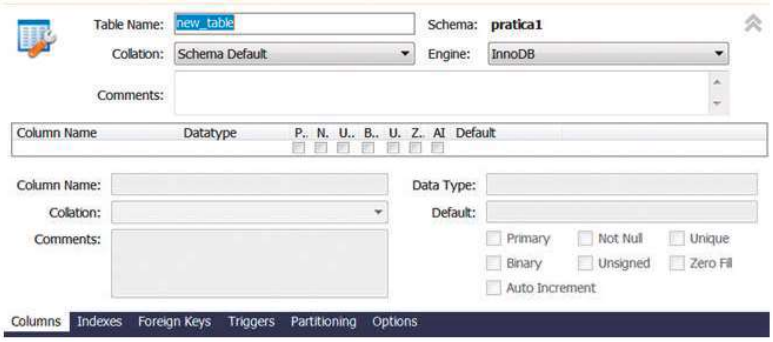
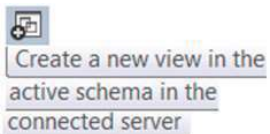
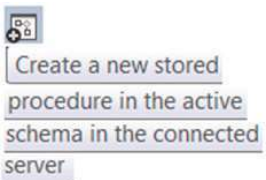


Fonte: Produção do próprio autor[31]

[30] Montagem do autor a partir de cópias de telas do programa.
[31] Montagem do autor a partir de cópias de telas do programa.

A primeira barra (barra 1 na Figura B.6) fornece acesso a assistentes e controle; novas abas de edição e suas funções podem ser vistas de forma comentada a seguir .

Quadro B.1 - Funções dos ícones da barra de ferramentas de assistentes

Ícone	Utilidade
	Cria uma nova aba para a execução de consultas na área de comandos.
	Abre um arquivo com *scripts* previamente salvos em outra aba para edição/uso.
	Abre a janela de inspeção de objetos em uma nova aba. Permite verificar várias características dos objetos, por exemplo, uma tabela. 
	Abre uma aba para a criação de um novo banco de dados (*SCHEMA*). É o mesmo que executar o comando *CREATE SCHEMA* ou *CREATE DATABASE*.
	Abre o assistente para a criação de tabelas e permite criar uma nova tabela no banco de dados atual do servidor a que estiver conectado. É o mesmo que *CREATE TABLE*. 
	Abre o editor para a criação de visões e permite criar uma nova visão no banco de dados atual do servidor a que estiver conectado. É o mesmo que *CREATE VIEW*.
	Abre o editor para com modelo para criação de procedimento armazenado e permite criar uma *STORED PROCEDURE* no banco.

Create a new function in the active schema in the connected server	Abre o editor para com modelo para criação de função e permite criar uma *FUNCTION* no banco.
Search table data for text in objects selected in the sidebar schema tree	Dá acesso ao mecanismo de pesquisa: 
Reconnect to DBMS	Efetua a reconexão ao banco de dados, utilizando o mesmo usuário e senha.

Fonte: Produção do próprio autor[33]

Vamos ver agora, no Quadro B.2, a barra de ferramentas do editor de comandos, localizada acima da área de comandos (barra 2 na Figura B.6), que será a mais utilizada para a realização dos exercícios descritos nesta disciplina.

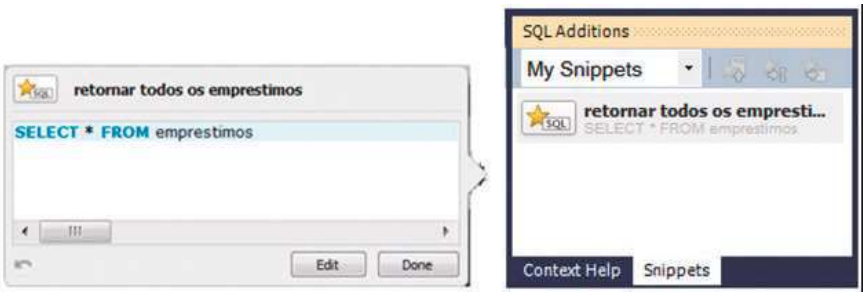
O termo usual que aparecerá a seguir como referência a uma sequência de comandos é *script* e será mantido devido à sua popularidade. Seu significado é um conjunto de instruções que deve ser executado para a realização de uma tarefa por um aplicativo.

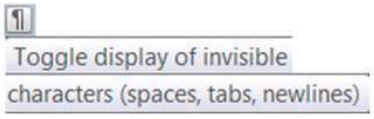
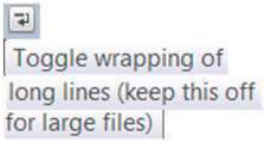
Poderia ser traduzido como roteiro, mas isso não é usual e pode confundi-lo quando estiver lendo material técnico ou interagindo com pessoal técnico da área, motivo pelo qual foi mantido o termo em inglês.

Quadro B.2 - Funções dos ícones da barra de ferramentas do editor de comandos

Ícone	Utilidade
Open a script file in this editor	Abre um arquivo previamente salvo que contém comandos SQL. Eles são chamados de *scripts*.
Save the script to a file.	Salva em um arquivo o *script* atual da área de comandos.
Execute the selected portion of the script or everything, if there is no selection	Permite a execução dos comandos de *script* digitados na área de comandos. Conforme já alertado, executa todos, caso não esteja selecionada somente uma parte dos comandos.
Execute the statement under the keyboard cursor	Executa somente o comando da linha atual.

[32] Montagem do autor a partir de cópias de telas do programa.

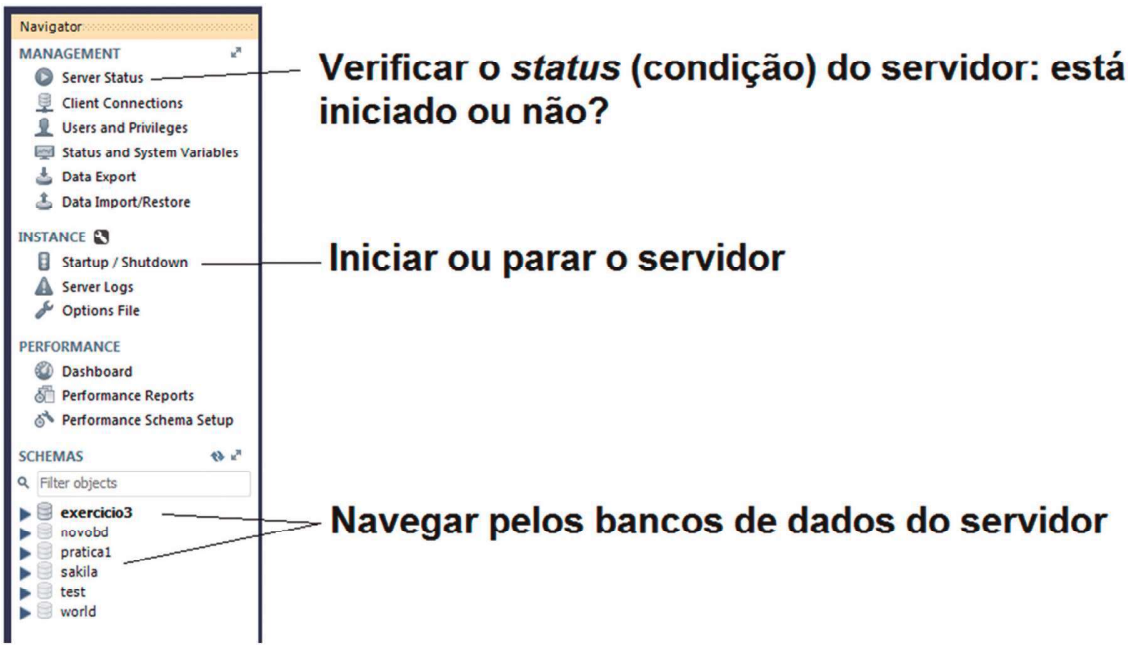
Stop the query being executed (the connection to the DB server will not be restarted and any open transactions will remain open)	Termina (aborta) a execução da consulta atual. Utilizado quando uma consulta for muito demorada ou não houver resposta para novos comandos.
Toggle whether execution of SQL script should continue after failed statements	Determina se a execução dos próximos comandos SQL continuará ou não após falha em alguma declaração.
Commit the current transaction.	Grava definitivamente a transação atual. Só funciona se o *autocommit* estiver desligado (veja o ícone , a seguir).
Rollback the current transaction.	Desfaz a gravação da transação atual. Só funciona se o *autocommit* estiver desligado (veja o ícone , a seguir).
Toggle autocommit mode.	Estabelece se será ou não realizado o *autocommit* das transações. Em bancos de dados, *COMMIT* significa finalizar uma transação, aceitando como corretos seus resultados e salvando todos em disco (uma sequência de atualizações ou inserções, por exemplo). Se a transação falha, pode ser realizado um *ROLLBACK*, que desfaz os comandos já executados. Porém, isso somente funciona dentro de transações, as quais terão a forma *BEGIN TRANSACTION ... END TRANSACTION*. As declarações a executar (*STATMENTS*) estarão incluídas entre o início e o final, assim como a decisão de executar o *commit* ou o *rollback*. Para efeitos dos conteúdos desta disciplina, deixe ligado o *autocommit* (ícone na forma pressionada,).
Limit to 1000 r Set limit for number of rows returned by queries.	Limita o número de linhas retornadas em uma consulta. Serve para evitar que uma tabela com muitas linhas (por exemplo milhares, milhões) congestione a rede e o *buffer* do cliente.
Save current statement or selection to the snippet list.	Salva a declaração ou seleção atual para uma lista de transações mais usadas, permitindo dar-lhe um nome. Essa declaração ficará disponível na subdivisão ***SQL Additions***, que fica ao lado da área de comandos, na aba *Snippets*: 
Beautify/reformat the SQL script	Melhora a formatação do *script* da área de comandos para melhorar sua compreensão. Exemplo: `SELECT nome, DATEDIFF(data_devolucao, data_emprestimo) AS 'Periodo em dias' FROM Usuarios, emprestimos WHERE Usuarios.cpf = emprestimos.cpf;`
Show the Find panel for the editor	Exibe o painel de buscas: 

	Apresenta ou esconde os caracteres invisíveis, de formatação. É semelhante ao dos editores de texto.
	Quebra linhas de comando longas em linhas que caibam na área de visualização dos comandos. Não é aconselhada para arquivos muito extensos.

Fonte: Produção do próprio autor[34]

O mínimo a utilizar do gerenciamento do servidor está no quadro de navegação, que pode ser visto na imagem (Figura B.7). Para efeitos desta disciplina são de interesse verificar se o servidor *MySQL* está ativo, iniciar ou para o serviço do servidor *MySQL*, e navegar pelos esquemas (ou bancos de dados).

Figura B.7 - Principais comandos presentes no quadro de navegação e gerenciamento



Fonte: Produção do próprio autor[35]

2.3 CRIAR DIAGRAMAS NO *MYSQL WORKBENCH*

O *Workbench* possui um poderoso editor de diagramas para modelagem de bancos de dados, que podem posteriormente criar objetos no banco de dados gerenciado pelo *MySQL* ao qual estiver conectado.

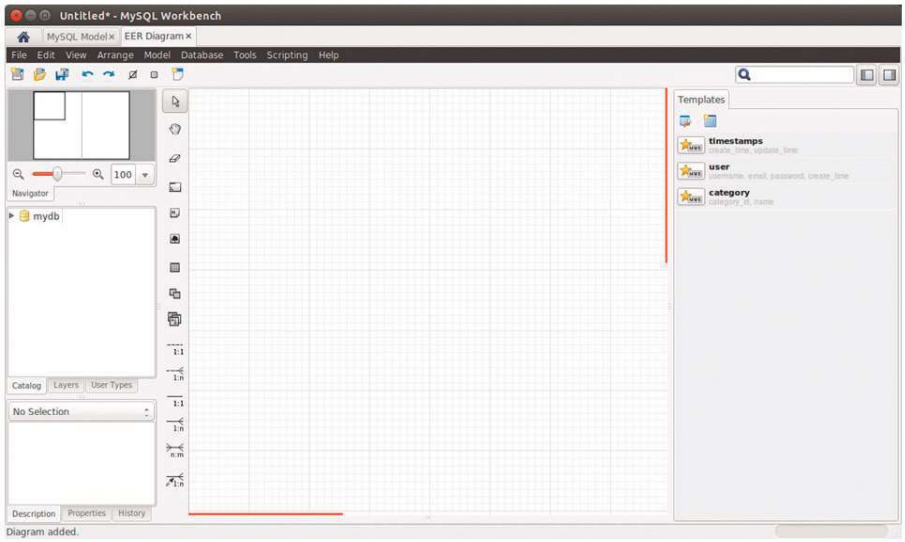
[33] Montagem do autor a partir de cópias de telas do programa.
[34] Montagem do autor a partir de cópias de telas do programa.

Figura B.8 - Edição de diagramas no Workbench

Fonte: Produção do próprio autor[35]

Você verá a tela de um novo diagrama em branco, na qual poderá editar seu diagrama (Figura B.9).

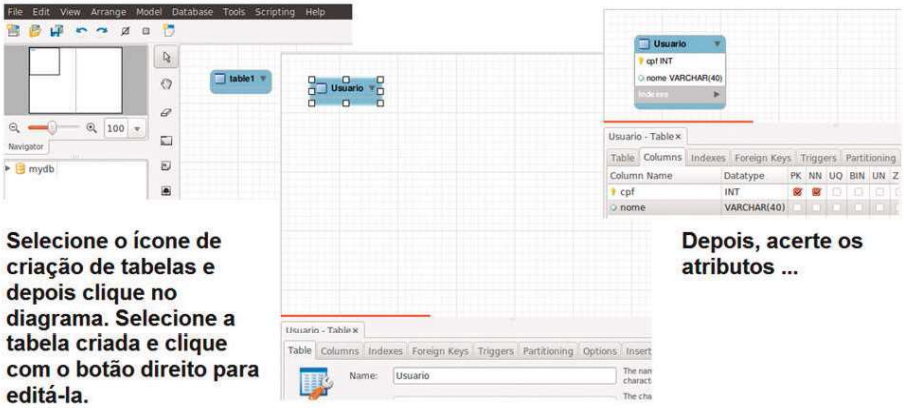
Figura B.9 - Aspecto da tela de edição de diagramas



Fonte: Produção do próprio autor[36]

Podemos então iniciar a criação das tabelas, o que é realizado clicando-se no ícone de criação de tabelas e depois selecionando-a, para possibilitar os ajustes de seus metadados e atributos (Figura B.10).

Figura B.10 - Criação de tabelas

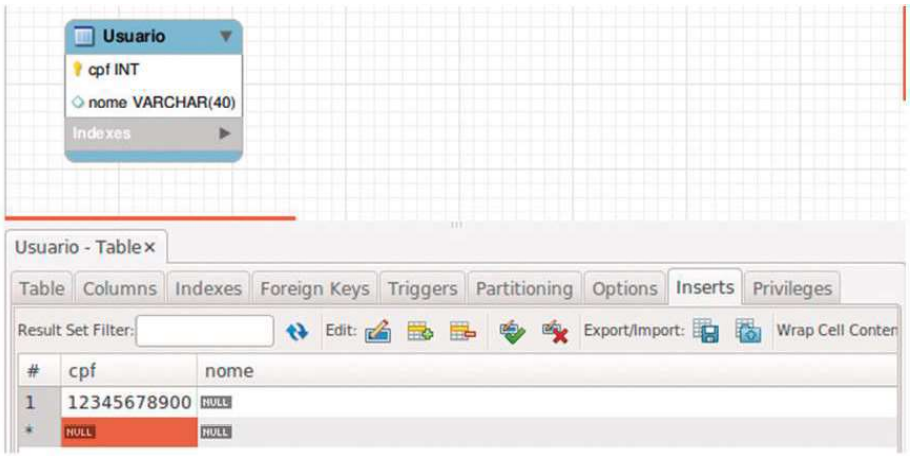


Fonte: Produção do próprio autor[37]

[35] Montagem do autor a partir de cópias de telas do programa.
[36] Montagem do autor a partir de cópias de telas do programa.
[37] Montagem do autor a partir de cópias de telas do programa.

Uma facilidade extra proporcionada pela ferramenta é a de que você poderá inserir dados para testar a tabela (Figura B.11).

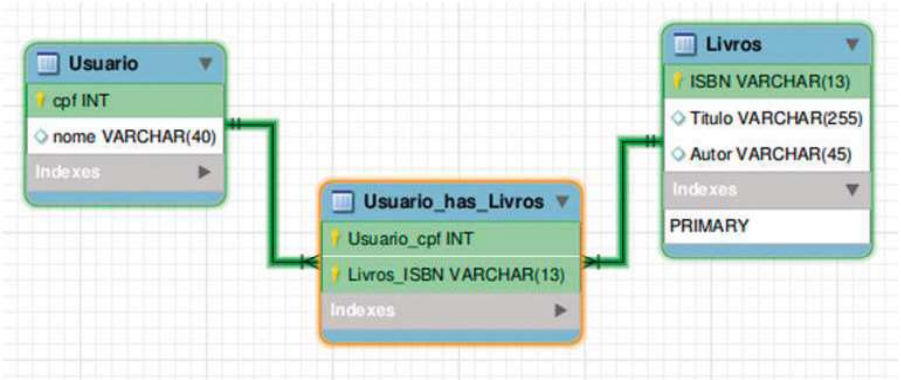
Figura B.11 - Tela de inserção de dados na tabela criada



Fonte: Produção do próprio autor[38]

Por fim, também será possível criar relacionamentos, conforme exibido na imagem (Figura B.12).

Figura B.12 - Criação de relacionamentos entre as tabelas no editor de diagramas



Fonte: Produção do próprio autor[39]

Evidentemente há muito mais a explorar, porém, isso não está no escopo desta disciplina, que é introdutória. Sugerimos que você explore melhor a ferramenta, em especial as funções de conexão e geração de esquemas e de *backup/restore*.

[38] Montagem do autor a partir de cópias de telas do programa.
[39] Montagem do autor a partir de cópias de telas do programa.