

# 25

## Acesso a bancos de dados com o JDBC



# OBJETIVOS

- Neste capítulo, você aprenderá:
- Conceitos de banco de dados relacional.
- Como utilizar Structured Query Language (SQL) para recuperar dados de um banco de dados, e manipular dados em um banco de dados.
- Como utilizar a API do JDBC™ do pacote `java.sql` para acessar bancos de dados.



- 25.1 Introdução**
- 25.2 Bancos de dados relacionais**
- 25.3 Visão geral de banco de dados relacional: O banco de dados books**
- 25.4 SQL**
  - 25.4.1 Consulta SELECT básica**
  - 25.4.2 Cláusula WHERE**
  - 25.4.3 Cláusula ORDER BY**
  - 25.4.4 Mesclando dados a partir de múltiplas tabelas: INNER JOIN**
  - 25.4.5 Instrução INSERT**
  - 25.4.6 Instrução UPDATE**
  - 25.4.7 Instrução DELETE**
- 25.5 Instruções para instalar o MySQL e o MySQL Conector/J**
- 25.6 Instruções para a configuração da conta de usuário do MySQL**
- 25.7 Criando banco de dados books no MySQL**
- 25.8 Manipulando bancos de dados com o JDBC**
  - 25.8.1 Consultando e conectando-se a um banco de dados**
  - 25.8.2 Consultando o banco de dados books**
- 25.9 Procedures armazenadas**
- 25.10 Interface RowSet**
- 25.11 Conclusão**



## 25.1 Introdução

- **Banco de dados:**
  - Coleção de dados.
- **DBMS (*database management system*):**
  - Sistema de gerenciamento de banco de dados.
  - Armazenando e organizando dados.
- **SQL:**
  - Linguagem-padrão internacional utilizada com bancos de dados relacionais.
  - Structured Query Language.



## 25.1 Introdução (*Continuação*)

- **RDBMS (Relational Database Management Systems):**
  - Sistema de gerenciamento de banco de dados relacional.
  - **MySQL:**
    - Código-fonte aberto.
    - Disponível tanto para Windows como para Linux.
    - `dev.mysql.com/downloads/mysql/4.0.html`.
- **JDBC:**
  - Estrutura de um banco de dados relacional.
  - **Driver JDBC:**
    - Permite que aplicações Java se conectem a bancos de dados.
    - Permite que programadores manipulem bancos de dados utilizando o JDBC.



# Observação de engenharia de software 25.1

---

**A separação entre API do JDBC e drivers de banco de dados particulares permite aos desenvolvedores alterar o banco de dados subjacente sem modificar o código Java que acessa o banco de dados.**

## 25.2 Bancos de dados relacionais

- **Banco de dados relacional:**
  - **Tabela:**
    - Linhas, colunas.
  - **Chave primária:**
    - Dados únicos.
- **Consultas SQL:**
  - Especificam quais dados selecionar em uma tabela.



	Number	Name	Department	Salary	Location
Linha {	23603	Jones	413	1100	New Jersey
	24568	Kerwin	413	2000	New Jersey
	34589	Larson	642	1800	Los Angeles
	35761	Myers	611	1400	Orlando
	47132	Neumann	413	9000	New Jersey
	78321	Stephens	611	8500	Orlando
	Chave primária		Coluna		

**Figura 25.1 | Dados de exemplo da tabela Employee.**



Department	Location
413	New Jersey
611	Orlando
642	Los Angeles

**Figura 25.2** | Resultado de selecionar dados distintos de Department e Location a partir da tabela Employee.



## 25.3 Visão geral de banco de dados relacional: O banco de dados books

- Banco de dados **books** de exemplo:
  - Quatro tabelas:
    - **authors**
      - `AuthorID`, `firstName`, `lastName`
    - **publishers**
      - `publisherID`, `publisherName`
    - **titles**
      - `isbn`, `title`, `editionNumber`, `copyright`, `publisherID`, `imageFile`, `price`
    - **authorISBN**
      - `authorID`, `isbn`



Coluna	Descrição
<b>authorID</b>	Número de ID do autor no banco de dados. No banco de dados <b>books</b> , essa coluna de inteiros é definida como <b>auto-incrementada</b> . Para cada linha inserida nessa tabela, o valor <b>authorID</b> é automaticamente aumentado por 1 para assegurar que cada linha tenha um <b>authorID</b> único. Essa coluna representa a chave primária da tabela.
<b>firstName</b>	Nome do autor (uma string).
<b>lastName</b>	Sobrenome do autor (uma string).

**Figura 25.3** | A tabela **authors** de **books**.



authorID	firstName	lastName
1	Harvey	Deitel
2	Paul	Deitel
3	Tem	Nieto
4	Sean	Santry

**Figura 25.4** | Dados de exemplo da tabela authors.



Coluna	Descrição
<b>publ<sub>i</sub>sherID</b>	<b>O número de ID do editor no banco de dados. Esse inteiro auto-incrementado é a chave primária da tabela.</b>
<b>publ<sub>i</sub>sherName</b>	<b>O nome do editor (uma string).</b>

**Figura 25.5** | A tabela publ<sub>i</sub>shers de books.



publ i sherID	publ i sherName
1	Prentice Hall
2	Prentice Hall PTG

**Figura 25.6** | Dados da tabela publ i shers.

Coluna	Descrição
<code>isbn</code>	<b>ISBN do livro (uma string). A chave primária da tabela. O ISBN é a abreviação de 'International Standard Book Number' — um sistema internacional de numeração padronizado que os editores utilizam para dar a todos os livros um número de identificação único.</b>
<code>title</code>	<b>Título do livro (uma string).</b>
<code>editionNumber</code>	<b>Número de edição do livro (um inteiro).</b>
<code>copyright</code>	<b>Ano de direitos autorais do livro (uma string).</b>
<code>publisherID</code>	<b>Número de ID do editor (um inteiro). Uma chave estrangeira que relaciona essa tabela com a tabela <code>publishers</code>.</b>
<code>imageFile</code>	<b>Nome do arquivo que contém a imagem da capa do livro (uma string).</b>
<code>price</code>	<b>Preço de varejo sugerido do livro (um número real). [Nota: Os preços exibidos na Figura 25.8 são apenas para fins de exemplo.]</b>

**Figura 25.7** | A tabela `titles de books`.

## 25.3 Visão geral de banco de dados relacional: Banco de dados books (*Cont.*)

- **Chave estrangeira:**
  - Uma coluna:
    - coincide com a coluna da chave primária em uma outra tabela.
  - Ajuda a manter a *Regra da Integridade Referencial*:
    - todo valor de chave estrangeira deve aparecer como o valor de chave primária da outra tabela.
- **Diagrama de relacionamento de entidade (ER — *entity-relationship*):**
  - Tabelas no banco de dados.
  - Relacionamentos entre as tabelas.





## 25.3 Visão geral de banco de dados relacional: Banco de dados books (Cont.)

- **Regra de Integridade de Entidade:**
  - A chave primária identifica unicamente cada linha.
  - Cada linha deve ter um valor para cada coluna da chave primária.
  - O valor da chave primária deve ser único na tabela.



## Erro comum de programação 25.1

---

**Não fornecer um valor para cada coluna em uma chave primária quebra a Regra de Integridade de Entidade e faz com que o DBMS informe um erro.**

## Erro comum de programação 25.2

---

**Fornecer o mesmo valor para a chave primária em múltiplas linhas faz com que o DBMS informe um erro.**



isbn	title	editionNumber	copy-right	publisherID	Image File	price
0131426443	C How to Program	4	2004	1	chtp4.jpg	85.00
0130384747	C++ How to Program	4	2003	1	cpphtp4.jpg	85.00
0130461342	Java Web Services for Experienced Programmers	1	2003	1	jwsfep1.jpg	54.99
0131483986	Java How to Program	6	2005	1	jhtp6.jpg	85.00
013100252X	The Complete C++ Training Course	4	2003	2	cppctc4.jpg	109.99
0130895601	Advanced Java 2 Platform How to Program	1	2002	1	advjhtp1.jpg	69.95

**Figura 25.8** | Dados de exemplo da tabela titles de books.



Coluna	Descrição
<b>authorID</b>	<b>O número de ID do autor, uma chave estrangeira para a tabela <code>authors</code>.</b>
<b>isbn</b>	<b>O ISBN de um livro, uma chave estrangeira para a tabela <code>titles</code>.</b>

**Figura 25.9** | A tabela `authorISBN` de `books`.



authorID	isbn	authorID	isbn
1	0130895725	2	0139163050
2	0130895725	3	0130829293
2	0132261197	3	0130284173
2	0130895717	3	0130284181
2	0135289106	4	0130895601

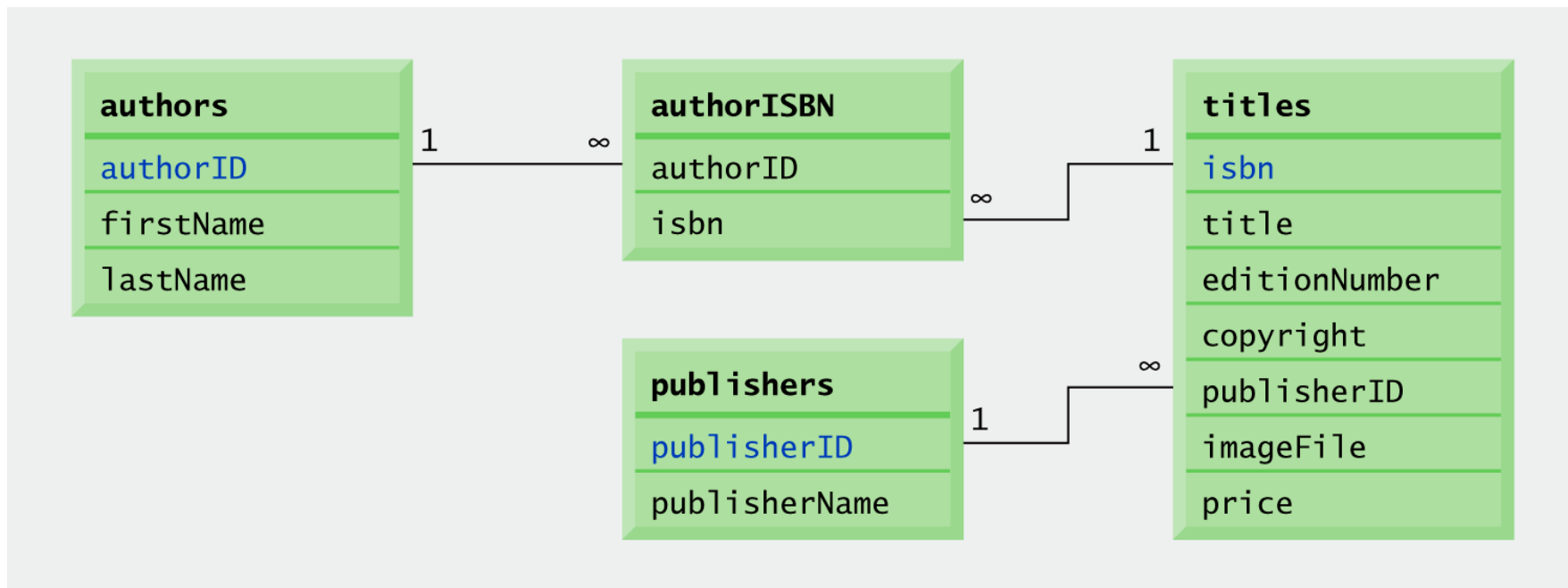
**Figura 25.10** | Dados de exemplo da tabela authorISBN de books.

## Erro comum de programação 25.3

---

**Fornecer um valor de chave estrangeira que não aparece como um valor de chave primária em outra tabela quebra a Regra de Integridade Referencial e faz com que o DBMS informe um erro.**





**Figura 25.11** | Os relacionamentos de tabela em books.



## 25.4 SQL

- **Palavras-chave de SQL.**
  - Consultas e instruções SQL.



Palavra-chave de SQL	Descrição
<b>SELECT</b>	<b>Recupera dados de uma ou mais tabelas.</b>
<b>FROM</b>	<b>Tabelas envolvidas na consulta. Requeridas em cada SELECT.</b>
<b>WHERE</b>	<b>Critérios de seleção que determinam as linhas a ser recuperadas, excluídas ou atualizadas. Opcional em uma consulta ou uma instrução SQL.</b>
<b>GROUP BY</b>	<b>Critérios para agrupar linhas. Opcional em uma consulta SELECT.</b>
<b>ORDER BY</b>	<b>Critérios para ordenar linhas. Opcional em uma consulta SELECT.</b>
<b>INNER JOIN</b>	<b>Mescla linhas de múltiplas tabelas.</b>
<b>INSERT</b>	<b>Insere linhas em uma tabela especificada.</b>
<b>UPDATE</b>	<b>Atualiza linhas em uma tabela especificada.</b>
<b>DELETE</b>	<b>Exclui linhas de uma tabela especificada.</b>

**Figura 25.12 | Palavras-chave de consulta do SQL.**



## 25.4.1 Consulta SELECT básica

- **Formato básico de uma consulta SELECT:**
  - **SELECT** \* **FROM** *nomeDaTabela*
    - **SELECT** \* **FROM** authors
- **Para selecionar campos específicos de uma tabela, como authorID e lastName:**
  - **SELECT** authorID, lastName **FROM** authors



authorID	lastName
1	Deitel
2	Deitel
3	Nieto
4	Santry

**Figura 25.13** | Dados authorID e lastName de exemplo da tabela authors.



# Observação de engenharia de software 25.2

---

**Para a maioria das consultas, o asterisco (\*) não deve ser utilizado para especificar nomes de coluna. Em geral, os programadores processam resultados sabendo antecipadamente a ordem das colunas no resultado — por exemplo, selecionar `authorID` e `lastName` da tabela `authors` assegura que as colunas aparecerão no resultado com `authorID` como a primeira coluna e `lastName` como a segunda coluna. Em geral, os programas processam colunas de resultados especificando o número de coluna no resultado (iniciando do número 1 da primeira coluna). Selecionar colunas por nome também evita retornar colunas desnecessárias e protege contra alterações na ordem real das colunas na(s) tabela(s).**

---



## Erro comum de programação 25.4

---

**Se um programador assume que as colunas são sempre retornadas na mesma ordem de uma consulta que utiliza o asterisco (\*), o programa pode processar o resultado incorretamente. Se a ordem de coluna na(s) tabela(s) mudar ou se colunas adicionais forem adicionadas posteriormente, a ordem das colunas no resultado mudaria de maneira correspondente.**



## 25.4.2 Cláusula WHERE

- **Especifica os critérios de seleção:**
  - **SELECT** *nomeDaColuna1, nomeDaColuna2, ...* **FROM** *nomeDaTabela* **WHERE** *critérios*
    - **SELECT** title, editionNumber, copyright
      - **FROM** titles
      - **WHERE** copyright > 2002



title	editionNumber	copyright
The Complete C++ Training Course	4	2003
Java How to Program	5	2003
C How to Program	4	2004
Internet and World Wide Web How to Program	3	2004
Java How to Program	6	2005
C# How to Program	1	2003

**Figura 25.14** | Amostragem de títulos com direitos autorais posteriores a 2002 da tabela `titles`.





## 25.4.2 Cláusula WHERE (*Continuação*)

- Operadores de condição da cláusula WHERE:
  - <, >, <=, >=, =, <>
  - LIKE:
    - Caracteres-curingas: porcentagem (%) e sublinhado (\_).
    - **SELECT** authorID, firstName, lastName
      - **FROM** authors
      - **WHERE** lastName **LIKE** 'D%'



authorID	firstName	lastName
1	Harvey	Deitel
2	Paul	Deitel

**Figura 25.15** | Autores cujo sobrenome inicia com D da tabela authors.

## Dica de portabilidade 25.1

---

**Consulte a documentação do sistema de banco de dados para determinar se a SQL diferencia letras maiúsculas de minúsculas no sistema e determinar a sintaxe de palavras-chave de SQL (isto é, todas devem estar em maiúsculas, todas em minúsculas ou algumas combinações das duas?).**



## Dica de portabilidade 25.2

---

**Leia a documentação do sistema de banco de dados cuidadosamente para determinar se o sistema suporta o operador LIKE.**



## Dica de portabilidade 25.3

---

**Alguns bancos de dados utilizam o caractere  
\* em vez do caractere % em um padrão.**

## 25.4.2 Cláusula WHERE (*Continuação*)

- **SELECT** authorID, firstName, lastName
  - **FROM** authors
  - **WHERE** lastName **LIKE** 'D%'



authorID	firstName	lastName
3	Tem	Nieto

**Figura 25.16** | O único autor da tabela authors cujo sobrenome contém i como a segunda letra.



## Dica de portabilidade 25.4

---

**Alguns sistemas de banco de dados utilizam o caractere ? em vez do caractere \_ em um pattern.**



## 25.4.3 Cláusula ORDER BY

- Cláusula ORDER BY opcional:
  - **SELECT** *nomeDaColuna1, nomeDaColuna2, ...* **FROM** *nomeDaTabela* **ORDER BY** *coluna* **ASC**
    - **SELECT** authorID, firstName, lastName
      - **FROM** authors
      - **ORDER BY** lastName **ASC**
  - **SELECT** *nomeDaColuna1, nomeDaColuna2, ...* **FROM** *nomeDaTabela* **ORDER BY** *coluna* **DESC**
    - **SELECT** authorID, firstName, lastName
      - **FROM** authors
      - **ORDER BY** lastName **DESC**



authorID	firstName	lastName
1	Harvey	Deitel
2	Paul	Deitel
3	Tem	Nieto
4	Sean	Santry

**Figura 25.17** | Dados de exemplo da tabela authors em ordem crescente por lastName.



authorID	firstName	lastName
4	Sean	Santry
3	Tem	Nieto
1	Harvey	Deitel
2	Paul	Deitel

**Figura 25.18** | Dados de exemplo da tabela authors em ordem decrescente por lastName.



## 25.4.3 Cláusula ORDER BY (*Continuação*)

- **ORDER BY** múltiplos campos:
  - **ORDER BY** *coluna1 ordemDeClassificação, coluna2 ordemDaClassificação, ...*
    - **SELECT** authorID, firstName, lastName
      - **FROM** authors
      - **ORDER BY** lastName, firstName



authorID	firstName	lastName
1	Harvey	Deitel
2	Paul	Deitel
3	Tem	Nieto
4	Sean	Santry

**Figura 25.19** | Dados de exemplo de authors em ordem crescente por lastName e firstName.



## 25.4.3 Cláusula ORDER BY (*Continuação*)

- Combina as cláusulas WHERE e ORDER BY:
  - **SELECT** isbn, title, editionNumber, copyright, price
    - **FROM** titles
    - **WHERE** title **LIKE** 'How to Program'
    - **ORDER BY** title **ASC**



isbn	title	Edition -Number	copy- right	price
0130895601	Advanced Java 2 Platform How to Program	1	2002	69.95
0131426443	C How to Program	4	2004	85.00
0130384747	C++ How to Program	4	2003	85.00
013028419x	e-Business and e-Commerce How to Program	1	2001	69.95
0131450913	Internet and World Wide Web How to Program	3	2004	85.00
0130284181	Perl How to Program	1	2001	69.95
0134569555	Visual Basic 6 How to Program	1	1999	69.95
0130284173	XML How to Program	1	2001	69.95

**Figura 25.20** | Amostra de livros da tabela `titles` cujos títulos terminam com `How to Program` em ordem crescente por `title`.



## 25.4.4 Mesclando dados a partir de múltiplas tabelas: INNER JOIN

- Divide dados relacionados em tabelas separadas.
- Faz uma junção das tabelas.
  - Mescla dados de múltiplas tabelas em uma única visualização.
  - INNER JOIN:
    - **SELECT** *nomeDaColuna1, nomeDaColuna2, ...*
    - **FROM** *tabela1*
    - **INNER JOIN** *tabela2*
    - **ON** *tabela1.nomeDaColuna1 = tabela2.nomeDaColuna2*
    - **SELECT** *firstName, lastName, isbn*
    - **FROM** *authors, authorISBN*
    - **INNER JOIN** *authorISBN*
    - **ON** *authors.authorID = authorISBN.authorID*
    - **ORDER BY** *lastName, firstName*





# Observação de engenharia de software 25.3

---

**Se uma instrução de SQL incluir colunas de múltiplas tabelas que tenham o mesmo nome, a instrução deve preceder esses nomes de coluna com seus nomes de tabela e um ponto (por exemplo, `authors.authorID`).**

## Erro comum de programação 25.5

---

**Em uma consulta, a falha em qualificar nomes de colunas que tenham o mesmo nome em duas ou mais tabelas é um erro.**



firstName	lastName	isbn	first Name	lastName	isbn
Harvey	Deitel	0130895601	Paul	Deitel	0130895717
Harvey	Deitel	0130284181	Paul	Deitel	0132261197
Harvey	Deitel	0134569555	Paul	Deitel	0130895725
Harvey	Deitel	0139163050	Paul	Deitel	0130829293
Harvey	Deitel	0135289106	Paul	Deitel	0134569555
Harvey	Deitel	0130895717	Paul	Deitel	0130829277
Harvey	Deitel	0130284173	Tem	Nieto	0130161438
Harvey	Deitel	0130829293	Tem	Nieto	013028419x
Paul	Deitel	0130852473	Sean	Santry	0130895601

**Figura 25.21** | Amostragem de autores e ISBNs dos livros que eles escreveram em ordem crescente por lastName e firstName.



## 25.4.5 Instrução INSERT

- Insere uma linha em uma tabela:
  - **INSERT INTO** *nomeDaTabela* ( *nomeDaColuna1*, ... , *nomeDaColunaN* )
  - **VALUES** ( *valor1*, ... , *valorN* )
    - **INSERT INTO** authors ( firstName, lastName )
    - **VALUES** ( 'Sue', 'Smith' )



authorID	firstName	lastName
1	Harvey	Deitel
2	Paul	Deitel
3	Tem	Nieto
4	Sean	Santry
5	Sue	Smith

**Figura 25.22** | Dados de exemplo da tabela Authors depois de uma operação INSERT.

## Erro comum de programação 25.6

---

**É um erro especificar um valor para uma coluna auto-incrementada.**



## Erro comum de programação 25.7

---

**O SQL utiliza o caractere aspas simples ( ' ) como um delimitador de strings. Para especificar uma string que contém aspas simples (por exemplo, O'Malley) em uma instrução SQL, a string deve ter duas aspas simples na posição em que o caractere aspas simples aparece na string (por exemplo, 'O' 'Ma11ey'). O primeiro dos dois caracteres de aspas simples atua como um caractere de escape para o segundo. Não 'escapar' caracteres aspas simples em uma string que seja parte de uma instrução SQL é um erro de sintaxe de SQL.**

---



## 25.4.6 Instrução UPDATE

- Modifica os dados em uma tabela:

- **UPDATE** *nomeDaTabela*
- **SET** *nomeDaColuna1 = valor1, ... , nomeDaColunaN = valorN*
- **WHERE** *critérios*
  - **UPDATE** authors
  - **SET** lastName = 'Jones'
  - **WHERE** lastName = 'Smith' **AND** firstName = 'Sue'





authorID	firstName	lastName
1	Harvey	Deitel
2	Paul	Deitel
3	Tem	Nieto
4	Sean	Santry
5	Sue	Jones

**Figura 25.23** | Dados de exemplo da tabela authors depois de uma operação UPDATE.



## 25.4.7 Instrução DELETE

- Remove dados de uma tabela:
  - **DELETE FROM** *nomeDaTabela* **WHERE** *critérios*
    - **DELETE FROM** authors
    - **WHERE** lastName = 'Jones' **AND** firstName = 'Sue'



authorID	firstName	lastName
1	Harvey	Deitel
2	Paul	Deitel
3	Tem	Nieto
4	Sean	Santry

**Figura 25.24** | Dados de exemplo da tabela authors depois de uma operação DELETE.



## 25.5 Instruções para instalar o MySQL e o MySQL Conector/J

- **Instalar o MySQL:**

- Insira o CD e mude para o diretório  
`D:\software\MySQL\mysql-4.0.20c-win`
- Dê um clique duplo em `SETUP.EXE`.
- Siga as instruções.

- **Instalar o MySQL Conector/J:**

- Copie  
`mysql-connector-java-3.0.14-production.zip`
- Abra  
`mysql-connector-java-3.0.14-production.zip`
  - Extraia seu conteúdo para a unidade C:\.



## 25.6 Instruções para a configuração da conta de usuário do MySQL

- **Configure uma conta de usuário:**
  - **Inicie o servidor do banco de dados executando o script:**  
**C:\mysql\bin\mysqld**
  - **Inicie o monitor MySQL executando o comando:**  
**C:\mysql\bin>mysql -h localhost -u root**
  - **Crie uma conta:**  
**mysql> USE mysql;**  
**mysql> INSERT INTO user SET Host='localhost',**  
**User='jhtp6',**  
**Password=PASSWORD('jhtp6'), select\_priv='Y',**  
**Insert\_priv='Y', Update\_priv='Y',**  
**Delete\_priv='Y',**  
**Create\_priv='Y', Drop\_priv='Y',**  
**References\_priv='Y',**  
**Execute\_priv='Y';**  
**mysql> FLUSH PRIVILEGES;**  
**mysql> exit;**



## 25.7 Criando banco de dados books no MySQL

- **Crie o banco de dados books:**
  - Abra o prompt de comando e mude para o diretório
    - `C:\mysql\bin`
  - Inicie o banco de dados executando o comando
    - `C:\mysql\bin\mysqld`
  - Copie o script de SQL `books.sql` para o diretório
    - `C:\mysql\bin`
  - Abra um outro prompt de comando e mude para o diretório `C:\mysql\bin`
  - **5. Crie o banco de dados books executando o comando**
    - `C:\mysql\bin>mysql -h localhost -u jhttp6 -p < books.sql`



## 25.8 Manipulando bancos de dados com o JDBC

- **Conecte-se a um banco de dados.**
- **Consulte o banco de dados.**
- **Exiba os resultados da consulta na `Jtable`.**



## 25.8.1 Consultando e conectando-se a um banco de dados

- **DisplayAuthors:**

- Recupera toda a tabela **authors**.
- Exibe os dados no fluxo de saída-padrão.
- O exemplo ilustra como:
  - conectar-se ao banco de dados;
  - consultar o banco de dados; e
  - processar o resultado.





# Resumo

DisplayAuthors.java

(1 de 3)

```

1 // Fig. 25.25: DisplayAuthors.java
2 // Exibindo o conteúdo da tabela authors.
3 import java.sql.Connection;
4 import java.sql.Statement;
5 import java.sql.DriverManager;
6 import java.sql.ResultSet;
7 import java.sql.ResultSetMetaData;
8 import java.sql.SQLException;
9

```

Importa as classes e interfaces JDBC do pacote `java.sql`

```

10 public class DisplayAuthors
11 {

```

```

12 // nome do driver JDBC e URL do banco de dados

```

```

13 static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";

```

```

14 static final String DATABASE_URL = "jdbc:mysql://localhost/books";

```

```

15 // carrega o aplicativo

```

```

16 public static void main( String args[] )
17 {

```

```

18     Connection connection = null; // gerencia a conexão

```

```

19     Statement statement = null; // instrução de consulta

```

```

20 // conecta-se ao banco de dados books e o consulta

```

```

21 try

```

```

22 {

```

```

23     Class.forName( JDBC_DRIVER ); // carrega classe de driver do banco de dados

```

```

24 // estabelece conexão com o banco de dados

```

```

25 connection =

```

```

26     DriverManager.getConnection( DATABASE_URL, "jhtp6", "jhtp6" );

```

```

27

```

Declara uma constante **String** que especifica o nome da classe do driver JDBC

Declara uma constante **String** que especifica o URL do banco de dados

Carrega a definição de classe para o driver do banco de dados

Declara e inicializa uma referência **Connection** chamada **connection**.

```

31 // cria Statement para consultar banco de dados
32 statement = connection.createStatement();
33
34 // consulta banco de dados
35 ResultSet resultSet = statement.executeQuery(
36     "SELECT authorID, firstName, lastName FROM author");
37
38 // processa resultados da consulta
39 ResultSetMetaData metaData = resultSet.getMetaData();
40 int numberOfColumns = metaData.getColumnCount();
41 System.out.println( "Authors Table of Books Database:" );
42
43 for ( int i = 1; i <= numberOfColumns; i++ )
44     System.out.printf( "%-8s\t", metaData.getColumnName( i ) );
45 System.out.println();
46
47 while ( resultSet.next() )
48 {
49     for ( int i = 1; i <= numberOfColumns; i++ )
50         System.out.printf( "%-8s\t", resultSet.getObject( i ) );
51     System.out.println();
52 } // fim do while
53 } // fim do try
54 catch ( SQLException sqlException )
55 {
56     sqlException.printStackTrace();
57     System.exit( 1 );
58 } // fim do catch

```

Invoca o método **Connection**  
**createStatement** para obter um  
objeto **Statement**

Utiliza o método  
**executeQuery** do  
objeto **Statement** para  
obter um objeto **ResultSet**

Obtém os metadados que  
descrevem o **ResultSet**

Utiliza o método  
**ResultSetMetaData**  
**getColumnCount** para  
recuperar o  
número de  
colunas no **ResultSet**

Obtém o nome de  
coluna utilizando o  
método **getColumnName**

Posiciona o cursor do **ResultSet** na  
primeira linha no **ResultSet** com método  
**next**

Extraí o conteúdo de  
uma das colunas na  
linha atual

Captura a **SQLException**,  
que é lançada se a execução da  
consulta ou o processo do  
**ResultSet** falhar



```

59 catch ( ClassNotFoundException classNotFound )
60 {
61     classNotFound.printStackTrace();
62     System.exit( 1 );
63 } // fim do catch
64 finally // assegura que a instrução e conexão são fechada
65 {
66     try
67     {
68         statement.close();
69         connection.close();
70     } // fim do try
71     catch ( Exception exception )
72     {
73         exception.printStackTrace();
74         system.exit( 1 );
75     } // fim do catch
76 } // fim do finally
77 } // fim do main
78 } // fim da classe DisplayAuthors

```

A **ClassNotFoundException** é lançada se o carregador de classe não puder localizar a classe driver

rs.jav

Fecha a **Statement** e a **Connection** de banco de dados.

(3 de 3)

Linha 69

Linhas 68-69

Saída do programa

```

Authors Table of Books Database:
authorID  firstName  lastName
1         Harvey  Deitel
2         Paul   Deitel
3         Tem    Nieto
4         Sean   Santry

```



Tipo	Descrição
1	<p>O driver de ponte JDBC-para-ODBC conecta programas Java a origens de dados Microsoft ODBC (Open Database Connectivity). O Java 2 Software Development Kit da Sun Microsystems, Inc. inclui o driver de ponte JDBC-para-ODBC (<code>sun.jdbc.odbc.JdbcOdbcDriver</code>). Em geral, esse driver requer o driver ODBC no computador cliente e normalmente requer a configuração de origens de dados ODBC. O driver de ponte (Bridge) foi introduzido principalmente para propósitos de desenvolvimento, antes que outros tipos de drivers fossem disponibilizados e não deve ser utilizado em aplicativos de produção.</p>
2	<p>Os drivers parcialmente Java para API nativa, permitem que os programas JDBC utilizem APIs específicas de banco de dados (normalmente escritas em C ou C++) que permitem que os programas clientes acessem bancos de dados via a <b>Java Native Interface (JNI)</b>. A JNI é uma ponte entre uma JVM e o código escrito e compilado em uma linguagem específica de plataforma como C ou C++. Esse código é conhecido como <b>código nativo</b>. A JNI permite que os aplicativos Java interajam com o código nativo. Um driver do Tipo 2 traduz JDBC em chamadas específicas do banco de dados. Os drivers do Tipo 2 foram introduzidos por razões semelhantes aos do driver de ponte ODBC do Tipo 1.</p>
3	<p>Os drivers cliente para servidor Pure Java aceitam solicitações JDBC e as traduzem em um protocolo de rede que não é específico ao banco de dados. Essas solicitações são enviadas para um servidor, que converte as solicitações de banco de dados em um protocolo específico do banco de dados.</p>
4	<p>Os drivers Pure Java implementam protocolos de rede específicos ao banco de dados, para que os programas Java possam conectar-se diretamente a um banco de dados.</p>

**Figura 25.26 | Tipos de driver JDBC.**



# Observação de engenharia de software 25.4

---

**A maioria dos fornecedores de banco de dados importantes fornece seus próprios drivers de banco de dados de JDBC e muitos fornecedores independentes também fornecem drivers JDBC. Para obter informações adicionais sobre drivers JDBC, visite o site Web JDBC da Sun Microsystems, [servlet.java.sun.com/products/jdbc/drivers](http://servlet.java.sun.com/products/jdbc/drivers).**

# Observação de engenharia de software 25.5

---

**Na plataforma Microsoft Windows, a maioria dos bancos de dados suporta acesso via Open Database Connectivity (ODBC). ODBC é uma tecnologia desenvolvida pela Microsoft para permitir acesso genérico a diferentes sistemas de banco de dados na plataforma Windows (e algumas plataformas UNIX). O JDBC-to-ODBC Bridge permite que qualquer programa Java acesse qualquer origem de dados ODBC. O driver é a classe `JdbcOdbcDriver` no pacote `sun.jdbc.odbc`.**

---



RDBMS	Nome de driver JDBC	Formato de URL de banco de dados
MySQL	<code>com.mysql.jdbc.Driver</code>	<code>jdbc:mysql://nomeDoHost/ nomeDoBancoDeDados</code>
ORACLE	<code>oracle.jdbc.driver.OracleDriver</code>	<code>jdbc:oracle:thin:@nomeDoHost: númeroDaPorta:nomeDoBancoDeDados</code>
DB2	<code>COM.ibm.db2.jdbc.net.DB2Driver</code>	<code>jdbc:db2: nomeoHost:númeroDaPorta/ nomeDoBancoDeDados</code>
Sybase	<code>com.sybase.jdbc.SybDriver</code>	<code>jdbc:sybase:Tds: nomeDoHost: númeroDaPorta/nomeDoBancoDeDados</code>

**Figura 25.27** | Nomes de driver JDBC e URL de bancos de dados populares.



## Observação de engenharia de software 25.6

---

**A maioria dos sistemas de gerenciamento de bancos de dados requer que o usuário efetue logon antes de acessar o conteúdo do banco de dados. O método `DriverManager.getConnection` é sobrecarregado com as versões que permitem ao programa fornecer o nome de usuário e a senha para ganhar acesso.**



# Observação de engenharia de software 25.7

---

**Os metadados permitem aos programas processar o conteúdo de `ResultSet` dinamicamente quando as informações detalhadas sobre o `ResultSet` não forem conhecidas com antecedência.**

## Erro comum de programação 25.8

---

**Inicialmente, um cursor `ResultSet` é posicionado antes da primeira linha. Tentar acessar o conteúdo de um `ResultSet` antes de posicionar o cursor `ResultSet` na primeira linha com o método `next` produz uma `SQLException`.**

## Dica de desempenho 25.1

---

**Se uma consulta especificar as colunas exatas a selecionar do banco de dados, o `ResultSet` conterá as colunas na ordem especificada. Nesse caso, utilizar o número de coluna para obter o valor da coluna é mais eficiente que utilizar o nome de coluna. O número de coluna fornece acesso direto à coluna especificada. Utilizar o nome de coluna requer uma pesquisa linear dos nomes de coluna para localizar a coluna apropriada.**

---



## Erro comum de programação 25.9

---

**Especificar número de coluna 0 ao obter valores de um `ResultSet` produz uma `SQLException`.**



## Erro comum de programação 25.10

---

**Tentar manipular um `ResultSet` depois de fechar `Statement` que criou o `ResultSet` causa uma `SQLException`. O programa descarta `ResultSet` quando `Statement` correspondente é fechado.**



## Observação de engenharia de software 25.8

---

**Todo objeto `Statement` pode abrir apenas um objeto `ResultSet` por vez. Quando um `Statement` retorna um novo `ResultSet`, `Statement` fecha o `ResultSet` anterior. Para utilizar múltiplos `ResultSet`s paralelamente, objetos `Statement` separados devem retornar `ResultSet`s.**

## 25.8.2 Consultando o banco de dados books

- **Permite que o usuário insira qualquer consulta no programa.**
- **Exibe os resultados de uma consulta em uma `JTable`.**

# Resumo

## ResultSetTableModel.java

(1 de 8)

Linha 17

Linha 26

```
1 // Fig. 25.28: ResultSetTableModel.java
2 // Um TableModel que fornece dados ResultSet a uma JTable.
3 import java.sql.Connection;
4 import java.sql.Statement;
5 import java.sql.DriverManager;
6 import java.sql.ResultSet;
7 import java.sql.ResultSetMetaData;
8 import java.sql.SQLException;
9 import javax.swing.table.AbstractTableModel;
10
11 // Linhas e colunas do ResultSet são contadas a partir de 1 e linhas e
12 // colunas JTable são contadas a partir de 0. Ao processar
13 // linhas ou colunas de ResultSet para utilização em uma JTable, é
14 // necessário adicionar 1 ao número de linha ou coluna para manipular
15 // a coluna apropriada de ResultSet (isto é, coluna 0 de JTable é a
16 // coluna de ResultSet 1 e a linha de JTable 0 é a linha de ResultSet 1).
17 public class ResultSetTableModel extends AbstractTableModel
18 {
19     private Connection connection;
20     private Statement statement;
21     private ResultSet resultSet;
22     private ResultSetMetaData metaData;
23     private int numberOfRows;
24
25     // monitora o status da conexão de banco de dados
26     private boolean connectedToDatabase = false;
27
```

A classe **ResultSetTableModel** estende a classe **AbstractTableModel**, que implementa a interface **TableModel**.

A variável de instância **connectedToDatabase** monitora o status da conexão do banco de dados





28 // construtor inicializa resultSet e obtém seu objeto de metadados;

29 // determina número de linhas

```
30 public ResultSetTableModel( String driver, String url,
31     String username, String password, String query )
32     throws SQLException, ClassNotFoundException
33 {
```

34 // carrega classe de driver do banco de dados

35 Class.forName( driver );

37 // conecta-se ao banco de dados

38 connection = DriverManager.getConnection( url, username, password );

40 // cria Statement para consultar banco de dados

41 statement = connection.createStatement(

42 ResultSet.TYPE\_SCROLL\_INSENSITIVE,

43 ResultSet.CONCUR\_READ\_ONLY );

45 // atualiza status de conexão de banco de dados

46 connectedToDatabase = true;

48 // configura consulta e a executa

49 setQuery( query );

50 } // fim do construtor ResultSetTableModel

51

O construtor aceita cinco argumentos **String** — o nome da classe do driver, o URL de banco de dados, o nome do usuário, a senha e a consulta-padrão a realizar

(2 de 8)

Invoca o método **Connection.createStatement** para criar um objeto

linhas 30-31

linha 38

Indica que a conexão com o banco de dados

linhas 41-43

Invoca o método **ResultSetTableModel.setQuery** para realizar a consulta-padrão.

Linha 46

Linha 49



# Resumo

```

52 // obtém a classe que representa o tipo de coluna
53 public Class getColumnClass( int column ) throws IllegalStateException
54 {
55     // assegura que o banco de dados está conectado
56     if ( !connectedToDatabase )
57         throw new IllegalStateException( "Banco de dados não conectado." );
58
59     // determina a classe Java de coluna
60     try
61     {
62         String className = metaData.getColumnClassName( column );
63
64         // retorna objeto Class que representa classe
65         return Class.forName( className );
66     } // fim do try
67     catch ( Exception exception )
68     {
69         exception.printStackTrace();
70     } // fim do catch
71
72     return Object.class; // se ocorrerem os problemas acima, assume tipo Object
73 } // fim do método getColumnClass
74

```

Verifica o status da conexão de banco de dados

descreve o método **getColumnClass** para obter um objeto **Class** que representa a classe de uma coluna particular.

Obtém o nome de classe completamente qualificado para a coluna especificada.

Carrega a definição de classe para a classe e retorna o objeto **Class** correspondente.

Linha 56

Linha 62

Retorna o tipo-padrão.

Linha 72



# Resumo

```

75 // obtém número de colunas em ResultSet
76 public int getColumnCount() throws IllegalStateException
77 {
78     // assegura que o banco de dados conexão está disponível
79     if ( !connectedToDatabase )
80         throw new IllegalStateException( "Not Connected to Database" );
81
82     // determina número de colunas
83     try
84     {
85         return metaData.getColumnCount();
86     } // fim do try
87     catch ( SQLException sqlException )
88     {
89         sqlException.printStackTrace();
90     } // fim do catch
91
92     return 0; // se ocorrerem os problemas acima, retorna 0 p/ o número de colunas
93 } // fim do método getColumnCount
94
95 // obtém nome de uma coluna particular em ResultSet
96 public String getColumnName( int column ) throws IllegalStateException
97 {
98     // assegura que o banco de dados conexão está disponível
99     if ( !connectedToDatabase )
100         throw new IllegalStateException( "Not Connected to Database" );
101

```

Sobrescreve o método **getColumnCount** para obter o número de colunas no **ResultSet** subjacente do modelo

Obtém o número de colunas no **ResultSet**.

Sobrescreve o método **getColumnName** para obter o nome da coluna no **ResultSet** subjacente do modelo

ResultSetTableModel  
java

(4 de 8)

Linhas 76-93

Linha 85

Linhas 96-113

## Resumo

ResultSetTableModel  
1.java

(5 de 8)

Linha 105

Linhas 116-123

```
102 // determina o nome de coluna
103 try
104 {
105     return metaData.getColumnName( column + 1 );
106 } // fim do try
107 catch ( SQLException sqlException )
108 {
109     sqlException.printStackTrace();
110 } // fim do catch
111
112 return ""; // se ocorrerem problemas, retorna string vazia p/ nome de coluna
113 } // fim do método getColumnName
114
115 // retorna número de linhas em ResultSet
116 public int getRowCount() throws IllegalStateException
117 {
118     // assegura que o banco de dados conexão está disponível
119     if ( !connectedToDatabase )
120         throw new IllegalStateException(
121
122     return numberOfRows;
123 } // fim do método getRowCount
124
```

Obtém o nome de coluna  
no **ResultSet**.

Sobrescreve o método  
**getColumnCount** para obter o  
número de linhas no **ResultSet**  
subjacente do modelo



# Resumo

ResultSetTableModel  
1.java

(6 de 8)

45

```

125 // obtém valor na linha e coluna particular
126 public Object getValueAt( int row, int column )
127     throws IllegalStateException
128 {
129     // assegura que o banco de
130     if ( !connectedToDatabase )
131         throw new IllegalStateException
132
133     // obtém um valor na linha
134     try
135     {
136         resultSet.absolute( row + 1 );
137         return resultSet.getObject( column + 1 );
138     } // fim do try
139     catch ( SQLException sqlException )
140     {
141         sqlException.printStackTrace();
142     } // fim do catch
143
144     return ""; // se ocorrerem problemas, retorna objeto string vazio
145 } // fim do método getValueAt
146
147 // configura nova string de consulta de banco de dados
148 public void setQuery( String query )
149     throws SQLException, IllegalStateException
150 {
151     // assegura que o banco de dados conexão está disponível
152     if ( !connectedToDatabase )
153         throw new IllegalStateException( "Not Connected to Database" );
154

```

Sobrescreve o método **getValueAt** para obter o **Object** em uma linha e coluna particulares do **ResultSet** subjacente do modelo

Utiliza o método **ResultSet absolute** para posicionar o cursor de **ResultSet** em uma linha específica

Utiliza método **ResultSet getObject** para obter o **Object** em uma coluna específica da linha atual.



# Resumo

ResultSetTableModel  
1.java

(7 de 8)

Executa a consulta para  
obter um novo  
**ResultSet**

Utiliza o método **ResultSet**  
**last** para posicionar o cursor  
de **ResultSet** na última linha  
do **ResultSet**.

Utiliza o método **ResultSet**  
**getRow** para obter o número da  
linha atual no **ResultSet**.

Linha 162

Invoca o método  
**fireTableAStructureChanged** para  
notificar qualquer **JTable** que utiliza esse  
objeto **ResultSetTableModel** como seu  
modelo que a estrutura do modelo mudou

```

155 // especifica consulta e a executa
156 resultSet = statement.executeQuery( query );
157
158 // obtém metadados para ResultSet
159 metaData = resultSet.getMetaData();
160
161 // determina o número de linhas em Res
162 resultSet.last();
163 numberOfRows = resultSet.getRow();
164
165 // notifica a JTable de que modelo foi alterado
166 fireTableStructureChanged();
167 } // fim do método setQuery
168

```



```

169 // fecha Statement e Connection
170 public void disconnectFromDatabase()
171 {
172     if ( !connectedToDatabase )
173         return;
174
175     // fecha Statement e Connection
176     try
177     {
178         statement.close();
179         connection.close();
180     } // fim do try
181     catch ( SQLException sqlException )
182     {
183         sqlException.printStackTrace();
184     } // fim do catch
185     finally // atualiza status de conexão de banco de dados
186     {
187         connectedToDatabase = false;
188     } // fim do finally
189 } // fim do método disconnectFromDatabase
190 } // fim da classe ResultSetTableModel

```

Verifica se a conexão já foi terminada

O método **disconnectFromDatabase** implementa um método de terminação apropriado para a classe **ResultSetTableModel**

Fecha **Statement** e **Connection** se um objeto **ResultSetTableModel** tiver sofrido coleta de lixo.

Linhas 170-189

Linha 172

Configura **ConnectedToDatabase** como **false** para assegurar que os clientes não utilizem uma instância de **ResultSetTableModel** depois que essa instância já tenha sido terminada



## Dica de portabilidade 25.5

---

**Alguns drivers de JDBC não suportam `ResultSet`s roláveis. Nesses casos, o driver, em geral, retorna um `ResultSet` em que o cursor só pode se mover para frente. Para informações adicionais, consulte sua documentação de driver de banco de dados.**



## Dica de portabilidade 25.6

---

**Alguns drivers JDBC não suportam `ResultSet`s atualizáveis. Nesses casos, o driver, em geral, retorna um `ResultSet` de leitura. Para informações adicionais, consulte sua documentação de driver de banco de dados.**

# Erro comum de programação 25.11

---

**Tentar atualizar um `ResultSet` quando o driver de banco de dados não suporta `ResultSet`s atualizáveis causa `SQLException`s.**

Constante de tipo static ResultSet	Descrição
TYPE_FORWARD_ONLY	Especifica que o cursor de um <code>ResultSet</code> pode mover apenas para frente (isto é, da primeira para a última linha no <code>ResultSet</code> ).
TYPE_SCROLL_INSENSITIVE	Especifica que cursor de um <code>ResultSet</code> pode rolar em qualquer direção e que as alterações feitas no <code>ResultSet</code> durante o processamento do <code>ResultSet</code> não são refletidas no <code>ResultSet</code> a menos que o programa consulte novamente o banco de dados.
TYPE_SCROLL_SENSITIVE	Especifica que cursor de um <code>ResultSet</code> pode rolar em qualquer direção e que as alterações feitas no <code>ResultSet</code> durante o processamento do <code>ResultSet</code> são refletidas imediatamente no <code>ResultSet</code> .

**Figura 25.29** | Constantes `ResultSet` para especificar o tipo `ResultSet`.

Constante ResultSet static de concorrência	Descrição
CONCUR_READ_ONLY	Especifica que um <code>ResultSet</code> pode ser atualizado (isto é, alterações no conteúdo do <code>ResultSet</code> não podem ser refletidas no banco de dados com os métodos de <i>atualização</i> do <code>ResultSet</code> ).
CONCUR_UPDATABLE	Especifica que um <code>ResultSet</code> pode ser atualizado (isto é, as alterações no conteúdo do <code>ResultSet</code> podem ser refletidas no banco de dados com os métodos de <i>atualização</i> do <code>ResultSet</code> ).

**Figura 25.30** | Constantes `ResultSet` para especificar as propriedades do resultado.



## Erro comum de programação 25.12

---

**Tentar mover o cursor para trás por um `ResultSet` quando o driver de banco de dados não suportar rolagem para trás causa uma `SQLException`.**



# Resumo

## DisplayQueryResults.java

(1 de 7)

Linhas 22-25

```
1 // Fig. 25.31: DisplayQueryResults.java
2 // Exibe o conteúdo da tabela Authors no
3 // banco de dados Books.
4 import java.awt.BorderLayout;
5 import java.awt.event.ActionListener;
6 import java.awt.event.ActionEvent;
7 import java.awt.event.WindowAdapter;
8 import java.awt.event.WindowEvent;
9 import java.sql.SQLException;
10 import javax.swing.JFrame;
11 import javax.swing.JTextArea;
12 import javax.swing.JScrollPane;
13 import javax.swing.ScrollPaneConstants;
14 import javax.swing.JTable;
15 import javax.swing.JOptionPane;
16 import javax.swing.JButton;
17 import javax.swing.Box;
18
19 public class DisplayQueryResults extends JFrame
20 {
21     // driver JDBC e URL de banco de dados
22     static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
23     static final String DATABASE_URL = "jdbc:mysql://localhost/books";
24     static final String USERNAME = "jhttp6";
25     static final String PASSWORD = "jhttp6";
26
```

Declara o nome da classe do driver de banco de dados, o URL do banco de dados, o nome e a senha do usuário para acessar o banco de dados

27 // consulta padrão seleciona todas as linhas de tabela authors

28 static final String DEFAULT\_QUERY = "SELECT \* FROM authors";

Declara a consulta-padrão

30 private ResultSetTableModel tableModel;

31 private JTextArea queryArea;

Declara o **tableModel** como uma referência ao **ResultSetTableModel**

33 // cria o ResultSetTableModel e GUI

34 public DisplayQueryResults()

35 {

36 super( "Displaying Query Results" );

38 // cria o ResultSetTableModel e exibe tabela de banco de dados

39 try

40 {

41 // cria o TableModel para resultados da consulta SELECT \* FROM authors

42 tableModel = new ResultSetTableModel( JDBC\_DRIVER, DATABASE\_URL,  
43 USERNAME, PASSWORD, DEFAULT\_QUERY );

45 // configura JTextArea em que o usuário digita consultas

46 queryArea = new JTextArea( DEFAULT\_QUERY, 3, 100 );

47 queryArea.setWrapStyleWord( true );

48 queryArea.setLineWrap( true );

50 JScrollPane scrollPane = new JScrollPane( queryArea,

51 JScrollPaneConstants.VERTICAL\_SCROLLBAR\_AS\_NEEDED,

52 JScrollPaneConstants.HORIZONTAL\_SCROLLBAR\_NEVER );

54 // configura o JButton para enviar consulta

55 JButton submitButton = new JButton( "Submit Query" );

DisplayQueryResults  
.java

(2 de 7)

Linha 28

Cria o **TableModel**  
para resultados da  
consulta-padrão  
"SELECT \* FROM  
authors"



# Resumo

```

57 // cria o Box para gerenciar o posicionamento da queryArea e do
58 // submitButton na GUI
59 Box box = Box.createHorizontalBox();
60 box.add( scrollPane );
61 box.add( submitButton );

```

```

62 // cria o delegado JTable para tableModel
63 JTable resultTable = new JTable( tableModel );

```

Cria o delegado **JTable** para **tableModel**

```

64 // posiciona os componentes GUI no painel de conteúdo
65 add( box, BorderLayout.NORTH );
66 add( new JScrollPane( resultTable ), BorderLayout.CENTER );

```

(3 de 7)

```

67 // cria evento ouvinte para submitButton
68 submitButton.addActionListener(

```

Registra um handler de evento para **submitButton** em que o usuário clica para submeter uma consulta para o banco de dados.

```

69 new ActionListener()
70 {

```

```

71 // passa consulta para modelo de tabela
72 public void actionPerformed((ActionEvent event)
73 {

```

```

74 // realiza uma nova consulta
75 try
76 {

```

```

77 tableModel.setQuery( queryArea.getText() );

```

```

78 } // fim do try

```

```

79 catch (SQLException sqlException)
80 {

```

```

81 JOptionPane.showMessageDialog( null,
82 sqlException.getMessage(), "Database e
83 JOptionPane.ERROR_MESSAGE );
84
85
86
87
88

```

Invoca o método **ResultSetTableModel setQuery** para executar a nova consulta





# Resumo

DisplayQueryResult  
s.java

(4 de 7)

Linha 103

```

89 // tenta recuperar a partir da consulta de usuário inválida
90 // executando consulta padrão
91 try
92 {
93     tableModel.setQuery( DEFAULT_QUERY );
94     queryArea.setText( DEFAULT_QUERY );
95 } // fim do try
96 catch ( SQLException sqlException2 )
97 {
98     JOptionPane.showMessageDialog( null,
99         sqlException2.getMessage(), "Database error",
100         JOptionPane.ERROR_MESSAGE );
101
102 // assegura que a conexão de banco de dados
103 tableModel.disconnectFromDatabase();
104
105     System.exit( 1 ); // termina o aplicativo
106 } // fim do catch interno
107 } // fim do catch externo
108 } // fim do actionPerformed
109 } // fim da classe interna ActionListener
110 ); // fim da chamada para addActionListener
111
112 setSize( 500, 250 ); // configura tamanho da janela
113 setVisible( true ); // exhibe a janela
114 } // fim do try
115 catch ( ClassNotFoundException classNotFound )

```

Assegura que a conexão  
de banco de dados está  
fechada



# Resumo

DisplayQueryResult  
s.java

(5 de 7)

Linha 129

```
116 {
117     JOptionPane.showMessageDialog( null,
118         "MySQL driver not found", "Driver not found",
119         JOptionPane.ERROR_MESSAGE );
120
121     System.exit( 1 ); // termina o aplicativo
122 } // fim do catch
123 catch ( SQLException sqlException )
124 {
125     JOptionPane.showMessageDialog( null, sqlException.getMessage(),
126         "Database error", JOptionPane.ERROR_MESSAGE );
127
128     // assegura que a conexão de banco de dados
129     tableModel.disconnectFromDatabase(); ←
130
131     System.exit( 1 ); // termina o aplicativo
132 } // fim do catch
133
134 // dispõe da janela quando o usuário fecha o aplicativo (isso sobrescreve
135 // o padrão de HIDE_ON_CLOSE)
136 setDefaultCloseOperation( DISPOSE_ON_CLOSE );
137
138 // assegura que a conexão de banco de dados é fechada quando usuário fecha o aplicativo
139 addWindowListener(
140
```

Assegura que a conexão  
de banco de dados seja  
fechada



# Resumo

Assegura que a conexão de banco de dados esteja fechada quando a janela for fechada

DisplayQueryResults

DisplayQueryResults.java

(6 de 7)

Linhas 144-148

```
141 new WindowAdapter()
142 {
143     // desconecta-se do banco de dados e sai quando a janela
144     public void windowClosed( WindowEvent event )
145     {
146         tableModel.disconnectFromDatabase();
147         System.exit( 0 );
148     } // fim do método windowClosed
149 } // fim da classe WindowAdapter interna
150 ); // fim da chamada para addWindowListener
151 } // fim do construtor DisplayQueryResults
152
153 // executa o aplicativo
154 public static void main( String args[] )
155 {
156     new DisplayQueryResults();
157 } // fim do main
158 } // fim da classe DisplayQueryResults
```



# Resumo

DisplayQueryResults  
.java

(7 de 7)

Saída do programa

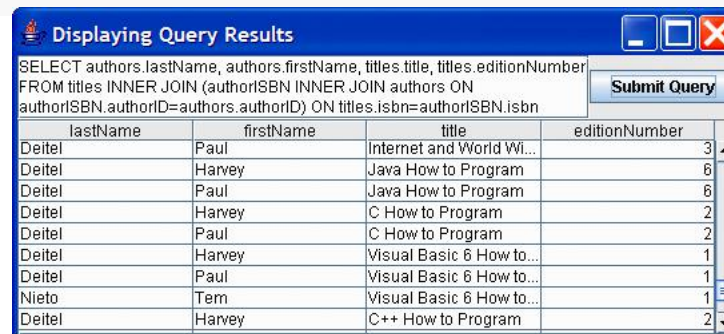


Displaying Query Results

SELECT \* FROM authors

Submit Query

authorID	firstName	lastName
1	Harvey	Deitel
2	Paul	Deitel
3	Teri	Nieto
4	Sean	Santry



Displaying Query Results

SELECT authors.lastName, authors.firstName, titles.title, titles.editionNumber  
FROM titles INNER JOIN (authorISBN INNER JOIN authors ON  
authorISBN.authorID=authors.authorID) ON titles.isbn=authorISBN.isbn

Submit Query

lastName	firstName	title	editionNumber
Deitel	Paul	Internet and World Wi...	3
Deitel	Harvey	Java How to Program	6
Deitel	Paul	Java How to Program	6
Deitel	Harvey	C How to Program	2
Deitel	Paul	C How to Program	2
Deitel	Harvey	Visual Basic 6 How to...	1
Deitel	Paul	Visual Basic 6 How to...	1
Nieto	Teri	Visual Basic 6 How to...	1
Deitel	Harvey	C++ How to Program	2



## 25.9 Procedures armazenadas

- **Procedures armazenadas:**
  - Armazenam instruções SQL em um banco de dados.
  - Invocam instruções SQL por programas que acessam o banco de dados.
- **Interface CallableStatement:**
  - Recebe argumentos.
  - Gera saída de parâmetros.



## Dica de portabilidade 25.7

---

**Embora a sintaxe para criar procedures armazenadas seja diferente entre sistemas de gerenciamento de bancos de dados, a interface CallableStatement fornece uma interface uniforme para especificar os parâmetros de entrada e saída para procedures armazenadas e para invocar procedures armazenadas.**

## Dica de portabilidade 25.8

---

**De acordo com a documentação da API do Java para a interface `CallableStatement`, para a máxima portabilidade entre sistemas de banco de dados, os programas devem processar as contagens de atualização ou os `ResultSet`s retornados de um `CallableStatement` antes de obter os valores de qualquer parâmetro de saída.**

## 15.10 Interface RowSet

- **Interface RowSet:**

- Configura a conexão de banco de dados automaticamente.
- Prepara as instruções de consulta automaticamente.
- Fornece os métodos *set* para especificar as propriedades necessárias para estabelecer uma conexão.
- Parte do pacote `javax.sql`.

- **Dois tipos de RowSet:**

- **RowSet conectado:** conecta a banco de dados uma vez e permanece conectado.
- **RowSet desconectado:** conecta a banco de dados, executa uma consulta e, então, fecha a conexão.





## 15.10 Interface RowSet (*Continuação*)

- Pacote `javax.sql.rowset`:
  - `JdbcRowSet`:
    - `RowSet` conectado.
    - Empacotador de um `ResultSet`.
    - Rolável e atualizável por padrão.
  - `CachedRowSet`:
    - `RowSet` desconectado.
    - Armazena os dados em cache do `ResultSet` na memória.
    - Rolável e atualizável por padrão.
    - Serializável:
      - Pode ser passada entre aplicações Java.
    - Limitação:
      - A quantidade de dados que pode ser armazenada na memória é limitada.



# Resumo

## JdbcRowSetTest.java

(1 de 3)

Linha 27

Linha 17

Linha 28

Linha 29

```

1 // Fig. 25.32: JdbcRowSetTest.java
2 // Exibindo o conteúdo da tabela authors com JdbcRowSet.
3 import java.sql.ResultSetMetaData;
4 import java.sql.SQLException;
5 import javax.sql.rowset.JdbcRowSet;
6 import com.sun.rowset.JdbcRowSetImpl; // Implementação JdbcRowSet da Sun
7
8 public class JdbcRowSetTest
9 {
10     // nome do driver JDBC e URL do banco de dados
11     static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
12     static final String DATABASE_URL = "jdbc:mysql://localhost/books";
13     static final String USERNAME = "jhtp6";
14     static final String PASSWORD = "jhtp6";
15
16     // construtor conecta-se ao banco de dados, consulta banco de dados, processa
17     // resultados e os exibe na janela
18     public JdbcRowSetTest()
19     {
20         // conecta-se ao banco de dados books e o consulta
21         try
22         {
23             Class.forName( JDBC_DRIVER ); // carrega classe de driver do banco de dados
24
25             // especifica propriedades de JdbcRowSet
26             JdbcRowSet rowSet = new JdbcRowSetImpl();
27             rowSet.setUrl( DATABASE_URL ); // configura URL do banco de dados
28             rowSet.setUsername( USERNAME ); // configura nome de usuário
29             rowSet.setPassword( PASSWORD ); // configura senha
30             rowSet.setCommand( "SELECT * FROM authors" ); // configura comando da consulta
31             rowSet.execute(); // executa a consulta
32

```

Utiliza a implementação de referência

Invoca o método JdbcRowSet setUsername

Invoca o método JdbcRowSet

Invoca o método JdbcRowSet execute para executar a consulta



# Resumo

jdbcRowSetTest.java

(2 de 3)

```
33 // processa resultados da consulta
34 ResultSetMetaData metaData = rowSet.getMetaData();
35 int numberOfColumns = metaData.getColumnCount();
36 System.out.println( "Authors Table of Books Database:" );
37
38 // exibe o cabeçalho rowset
39 for ( int i = 1; i <= numberOfColumns; i++ )
40     System.out.printf( "%-8s\t", metaData.getColumnName( i ) );
41 System.out.println();
42
43 // exibe cada linha
44 while ( rowSet.next() )
45 {
46     for ( int i = 1; i <= numberOfColumns; i++ )
47         System.out.printf( "%-8s\t", rowSet.getObject( i ) );
48     System.out.println();
49 } // fim do while
50 } // fim do try
51 catch ( SQLException sqlException )
52 {
53     sqlException.printStackTrace();
54     System.exit( 1 );
55 } // fim do catch
56 catch ( ClassNotFoundException classNotFound )
57 {
58     classNotFound.printStackTrace();
59     System.exit( 1 );
60 } // fim do catch
61 } // fim do construtor DisplayAuthors
62
```



# Resumo

```
63 // carrega o aplicativo
64 public static void main( String args[] )
65 {
66     JdbcRowSetTest window = new JdbcRowSetTest();
67 } // fim do main
68 } // fim da classe JdbcRowSetTest
```

JdbcRowSetTest.java

(3 de 3)

Saída do  
programa

```
Authors Table of Books Database:
authorID      firstName      lastName
1             Harvey        Deitel
2             Paul         Deitel
3             Tom          Nieto
4             Sean         Santry
```

