Java Database Connectivity with JDBC™

Outline

23.1 Introduction

- DataBase Management System (DBMS)
  - Mechanisms for storing and organizing data
  - Access, store, modify data without concern for internal representation (information hiding)
- Structured Query Language (SQL)
  - Standard language used with relational databases to perform queries and manipulate data
- Java DataBase Connectivity (JDBC)
  - Java programs communicate with databases using JDBC
  - JDBC driver implements interface to database

23.2 Relational-Database Model

- Relational database
  - Logical representation of data, not necessarily the way the data is stored
  - Table
    - Rows (entities), columns (attributes)
  - Primary key (column or group of columns)
    - Unique value for each row
    - Not every table has a primary key
- SQL statement
  - Query (which data to select from table or tables)

23.2 Relational-Database Model (Cont.)

Number	Name	Department	Salary	Location
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

Row

Primary key

Column

Fig. 23.1 **`Employee`** table sample data.

23.2 Relational-Database Model (Cont.)

Department	**Location**
413	New Jersey
611	Orlando
642	Los Angeles

Fig. 23.2 Result of selecting distinct **Department** and **Location** data from the **Employee** table.

23.3 Relational Database Overview: The books Database

- Sample `books` database
  - Four tables
    - `authors`, `publishers`, `authorISBN`, and `titles`
  - *Foreign key* is table entry that is a primary key in another table (enable rows from multiple tables to be joined)

23.3 Relational Database Overview: The `books` Database (Cont.)

Column	Description
`authorID`	Author's ID number in the database. In the `books` database, this integer column is defined as *autoincremented*. For each row inserted in this table, the database automatically increments the `authorID` value to ensure that each row has a unique `authorID`. This column represents the table's primary key.
`firstName`	Author's first name (a string).
`lastName`	Author's last name (a string).

Fig. 23.3 **`authors`** table from `books`.

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

Fig. 23.4 Sample data from the `authors` table.

23.3 Relational Database Overview: The `books` Database (Cont.)

Column	Description
`publisherID`	The publisher's ID number in the database. This autoincremented integer is the table's primary key.
`publisherName`	The name of the publisher (a string).

Fig. 23.5 **`publishers`** table from `books`.

`publisherID`	**`publisherName`**
1	Prentice Hall
2	Prentice Hall PTG

Fig. 23.6 Data from the `publishers` table.

23.3 Relational Database Overview: The `books` Database (Cont.)

Column	Description
`isbn`	ISBN of the book (a string). The table's primary key.
`title`	Title of the book (a string).
`editionNumber`	Edition number of the book (an integer).
`copyright`	Copyright year of the book (a string).
`publisherID`	Publisher's ID number (an integer). A foreign key to the `publishers` table.
`imageFile`	Name of the file containing the book's cover image (a string).
`price`	Suggested retail price of the book (a real number). [*Note:* The prices shown in this book are for example purposes only.]

Fig. 23.7 **`titles`** table from `books`.

23.3 Relational Database Overview: The `books` Database (Cont.)

isbn	title	edition-Number	copy-right	publish-erID	imageFile	price
0130895725	C How to Program	3	2001	1	chtp3.jpg	74.95
0130384747	C++ How to Program	4	2002	1	cpphtp4.jpg	74.95
0130461342	Java Web Services for Experienced Programmers	1	2002	1	jwsfep1.jpg	54.95
0131016210	Java How to Program	5	2003	1	jhtp5.jpg	74.95
0130852473	The Complete Java 2 Training Course	5	2002	2	javactc5.jpg	109.95
0130895601	Advanced Java 2 Platform How to Program	1	2002	1	advjhtp1.jpg	74.95

Fig. 23.8 Sample data from the `titles` table of `books`.

23.3 Relational Database Overview: The `books` Database (Cont.)

Column	Description
`authorID`	The author's ID number, a foreign key to the `authors` table.
`isbn`	The ISBN for a book, a foreign key to the `titles` table..

Fig. 23.9 **`authorISBN`** table from 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

Fig. 23.10 Sample data from the `authorISBN` table of books.

23.3 Relational Database Overview: The `books` Database (Cont.)

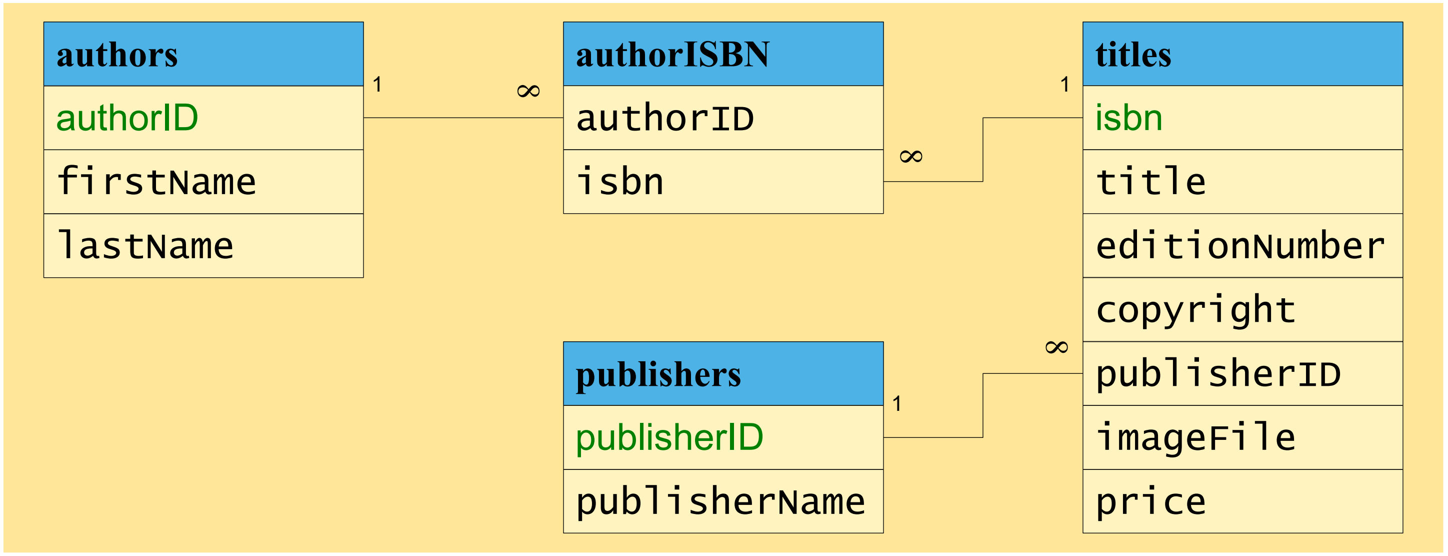


Fig. 23.11 Table relationships in **books**. Entity-relationship (ER) diagram.

23.4 SQL

• SQL keywords

SQL keyword	Description
`SELECT`	Retrieves data from one or more tables.
`FROM`	Tables involved in the query. Required in every `SELECT`.
`WHERE`	Criteria for selection that determine the rows to be retrieved, deleted or updated.
`GROUP BY`	Criteria for grouping rows.
`ORDER BY`	Criteria for ordering rows.
`INNER JOIN`	Merge rows from multiple tables.
`INSERT`	Insert rows into a specified table.
`UPDATE`	Update rows in a specified table.
`DELETE`	Delete rows from a specified table.

Fig. 23.12 SQL query keywords.

23.4.1 Basic SELECT Query

- Simplest form of a SELECT query
  - `SELECT * FROM` *tableName*
    - `SELECT * FROM authors`
    - * means all columns (not recommended)
- Select specific fields from a table
  - `SELECT authorID, lastName FROM authors`

authorID	lastName
1	Deitel
2	Deitel
3	Nieto
4	Santry

Fig. 23.13 Sample `authorID` and `lastName` data from the `authors` table.

23.4.2 WHERE Clause

- Specify the selection criteria (predicates)
  - **SELECT** *columnName1*, *columnName2*, … **FROM** *tableName* **WHERE** *criteria*
    - **SELECT** title, editionNumber, copyright
      FROM titles
      WHERE copyright > 2000

23.4.2 WHERE Clause (Cont.)

title	editionNumber	copyright
C How to Program	3	2001
C++ How to Program	4	2002
The Complete C++ Training Course	4	2002
Internet and World Wide Web How to Program	2	2002
Java How to Program	5	2003
XML How to Program	1	2001
Perl How to Program	1	2001
Advanced Java 2 Platform How to Program	1	2002

Fig. 23.14 Sampling of titles with copyrights after 2000 from table `titles`.

23.4.2 WHERE Clause

- WHERE clause condition operators
  - <, >, <=, >=, =, <>, LIKE
- LIKE (pattern matching)
  - wildcard characters **%** and _
    - % or * (zero or more characters no matter what they are)
    - _ or ? (single character no matter what it is)
    - wildcard string surrounded by single quotes

23.4.2 WHERE Clause (Cont.)

- **SELECT** authorID, firstName, lastName

  FROM authors

  WHERE lastName **LIKE** ‘D%’

authorID	firstName	lastName
1	Harvey	Deitel
2	Paul	Deitel

Fig. 23.15 Authors whose last name starts with `D` from the `authors` table.

23.4.2 WHERE Clause (Cont.)

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

authorID	firstName	lastName
3	Tem	Nieto

Fig. 23.16 The only author from the `authors` table whose last name contains `i` as the second letter.

23.4.3 ORDER BY Clause

- Optional **`ORDER BY`** clause
  - **`SELECT`** *columnName1*, *columnName2*, … **`FROM`** *tableName* **`ORDER BY`** *column* [**`ASC`**]
  - **`SELECT`** *columnName1*, *columnName2*, … **`FROM`** *tableName* **`ORDER BY`** *column* **`DESC`**
- Note that ASC is default (thus optional)
- **`ORDER BY`** multiple fields
  - **`ORDER BY`** *column1 sortingOrder*, *column2 sortingOrder*, …
- Combine the **`WHERE`** and **`ORDER BY`** clauses

23.4.3 ORDER BY Clause (Cont.)

- **SELECT** authorID, firstName, lastName
  FROM authors
  ORDER BY lastName **ASC**

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

Fig. 23.17 Sample data from table `authors` in ascending order by `lastName`.

23.4.3 ORDER BY Clause (Cont.)

- **SELECT** authorID, firstName, lastName

  FROM authors

  ORDER BY lastName **DESC**

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

Fig. 23.18 Sample data from table `authors` in descending order by `lastName`.

23.4.3 ORDER BY Clause (Cont.)

- **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

Fig. 23.19 Sample author data from table `authors` in ascending order by `lastName` and by `firstName`.

23.4.3 ORDER BY Clause (Cont.)

- **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	74.95
0130895725	C How to Program	3	2001	74.95
0130384747	C++ How to Program	4	2002	74.95
0130308978	Internet and World Wide Web How to Program	2	2002	74.95
0130284181	Perl How to Program	1	2001	74.95
0134569555	Visual Basic 6 How to Program	1	1999	74.95
0130284173	XML How to Program	1	2001	74.95
013028419x	e-Business and e-Commerce How to Program	1	2001	74.95

Fig. 23.20 Sampling of books from table `titles` whose titles end with `How to Program` in ascending order by `title`.

23.4.4 Merging Data from Multiple Tables: Joining

- Split related data into separate tables to avoid redundancy
- Join the tables
  - Merge data from multiple tables into a single view
  - INNER JOIN
    - **SELECT** *columnName1*, *columnName2*, …
      FROM *table1*
      INNER JOIN *table2*
      ON *table1.columnName* = *table2.column2Name*
    - **SELECT** firstName, lastName, isbn
      FROM authors
      INNER JOIN authorISBN
      ON authors.authorID = authorISBN.authorID
      ORDER BY lastName, firstName

23.4.4 Merging Data from Multiple Tables: Joining (Cont.)

firstName	lastName	isbn	firstName	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

Fig. 23.21 Sampling of authors and ISBNs for the books they have written in ascending order by `lastName` and `firstName`.

23.4.5 INSERT Statement

- Insert a row into a table
  - **INSERT INTO** *tableName* (*columnName1*, … , *columnNameN*)
    VALUES (*value1*, … , *valueN*)
    - **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

Fig. 23.22 Sample data from table `Authors` after an `INSERT` operation.

23.4.6 UPDATE Statement

- Modify data in a table
  - **UPDATE** *tableName*

    SET *columnName1* = *value1*, … , *columnNameN* = *valueN*

    WHERE *criteria*

    - **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

Fig. 23.23 Sample data from table `authors` after an `UPDATE` operation.

23.4.7 DELETE Statement

- Remove data from a table (row or rows)
  - **DELETE FROM** *tableName* **WHERE** *criteria*
    - **DELETE FROM** authors
      WHERE lastName = 'Jones' **AND** firstName = 'Sue'

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

Fig. 23.24 Sample data from table `authors` after a `DELETE` operation.

23.6 Manipulating Databases with JDBC

- Connect to a database
- Query the database
- Display the results of the query

23.6.1 Connecting to and Querying a Database

- DisplayAuthors
  - Retrieves the entire **`authors`** table
  - Displays the data in a **`JTextArea`**
  - **`Connection`** object manages connection between Java program and database
  - `connection = DriverManager.getConnection (DATABASE_URL);`
  - URL `jdbc:db2j:books` specifies communication protocol (jdbc), subprotocol (db2j), name of database (books)
  - `getConnection` overloaded (one version can be used to supply account and password)

Outline

```
1   // Fig. 23.26: DisplayAuthors.java
2   // Displaying the contents of the authors table.
3
4   import java.awt.*;
5   import java.sql.*;
6   import java.util.*;
7   import javax.swing.*;
8
9   public class DisplayAuthors extends JFrame {
10
11     // JDBC driver name and database URL
12     static final String JDBC_DRIVER = "com.ibm.db2j.jdbc.DB2jDriver";
13     static final String DATABASE_URL = "jdbc:db2j:books";
14
15     // declare Connection and Statement for accessing
16     // and querying database
17     private Connection connection;
18     private Statement statement;
19
20     // constructor connects to database, queries database, processes
21     // results and displays results in window
22     public DisplayAuthors()
23     {
24        super( "Authors Table of Books Database" );
25
```

Imports package `java.sql`, which contains classes and interfaces for the JDBC API.

Fig. 23.26 DisplayAuthors.java Displaying the authors table from the books database.

Line 5

rs.
java
Displaying the
authors table

```
         // connect to database books and query database
         try {

            // specify location of database on filesystem
            System.setProperty( "db2j.system.home", "C:/Cloudscape_5.0" );

            // load database driver class
            Class.forName( JDBC_DRIVER );

            // establish connection to database
            connection = DriverManager.getConnection( DATABAS

            // create Statement for querying database
            statement = connection.createStatement();

            // query database
            ResultSet resultSet =
               statement.executeQuery( "SELECT * FROM authors" );

            // process query results
            StringBuffer results = new StringBuffer();
            ResultSetMetaData metaData = resultSet.getMetaData();
            int numberOfColumns = metaData.getColumnCount();

```

Specify location of database

Loads the class definition for the database driv

ze a
rence
on.

Invokes `Connection` method `createStatement` to obtain an object that implements interface `Statement`.

Use the `Statement` object's `executeQuery` method to execute a query

Obtains the metadata

Uses `ResultSetMetaData` method `getColumnCount` to retrieve the number of columns in the `ResultSet`.

```
50          for ( int i = 1; i <= numberOfColumns; i++ )
51             results.append( metaData.getColumnName( i ) + "\t" );
52
53          results.append( "\n" );
54
55          while ( resultSet.next() ) {
56
57             for ( int i = 1; i <= numberOfColumns; i++ )
58                results.append( resultSet.getObject( i ) + "\t" );
59
60             results.append( "\n" );
61          }
62
63          // set up GUI and display window
64          JTextArea textArea = new JTextArea( results.toString() );
65          Container container = getContentPane();
66
67          container.add( new JScrollPane( textArea ) );
68
69          setSize( 300, 100 );  // set window size
70          setVisible( true );   // display window
71
72       }  // end try
73
```

Append the column names to the `StringBuffer` `results`.

java
Displaying the

Append the data in each `ResultSet` row to the `StringBuffer` `results`.

Line 58

Create the GUI that displays the `StringBuffer` `results`, set the size of the application window and show the application window.

```
74          // detect problems interacting with the database
75          catch ( SQLException sqlException ) {
76             JOptionPane.showMessageDialog( null, sqlException.getMessage(),
77                "Database Error", JOptionPane.ERROR_MESSAGE );
78
79             System.exit( 1 );
80          }
81
82          // detect problems loading database driver
83          catch ( ClassNotFoundException classNotFound ) {
84             JOptionPane.showMessageDialog( null, classNotFound.getMessage(),
85                "Driver Not Found", JOptionPane.ERROR_MESSAGE );
86
87             System.exit( 1 );
88          }
89
90          // ensure statement and connection are closed properly
91          finally {
92
93             try {
94                statement.close();
95                connection.close();
96             }
97
```

Close the `Statement` and the database `Connection`.

Outline

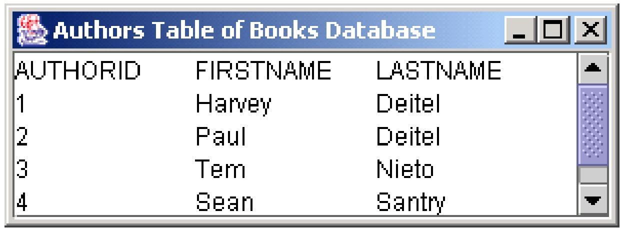
Fig. 23.26 DisplayAuthors.java Displaying the authors table from the books database.

Lines 94-95

```
         // handle exceptions closing statement and connection
         catch ( SQLException sqlException ) {
            JOptionPane.showMessageDialog( null,
               sqlException.getMessage(), "Database Error",
               JOptionPane.ERROR_MESSAGE );

            System.exit( 1 );
         }
      }

   }  // end DisplayAuthors constructor

   // launch the application
   public static void main( String args[] )
   {
      DisplayAuthors window = new DisplayAuthors();
      window.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
   }

}  // end class DisplayAuthors
```

Fig. 23.26 DisplayAuthors.java Displaying the authors table from the books database.

Program output



Authors Table of Books Database

```
AUTHORID    FIRSTNAME   LASTNAME
1           Harvey      Deitel
2           Paul        Deitel
3           Tem         Nieto
4           Sean        Santry
```

23.6.2 Querying the books Database

- Allow the user to enter any query into the program
- Display the results of a query in a `JTable` (GUI component that looks like a table)
- `ResultSetTableModel` constructor throws any exceptions back to the application that created the `ResultSetTableModel` object

```java
// Fig. 23.27: ResultSetTableModel.java
// A TableModel that supplies ResultSet data to a JTable.

import java.sql.*;
import java.util.*;
import javax.swing.table.*;

// ResultSet rows and columns are counted from 1 and JTable
// rows and columns are counted from 0. When processing
// ResultSet rows or columns for use in a JTable, it is
// necessary to add 1 to the row or column number to manipulate
// the appropriate ResultSet column (i.e., JTable column 0 is
// ResultSet column 1 and JTable row 0 is ResultSet row 1).
public class ResultSetTableModel extends AbstractTableModel {
   private Connection connection;
   private Statement statement;
   private ResultSet resultSet;
   private ResultSetMetaData metaData;
   private int numberOfRows;

   // keep track of database connection status
   private boolean connectedToDatabase = false;

```

ResultSetTableModel enables a Jtable to display the contents of a ResultSet.

Outline

```
24     // initialize resultSet and obtain its meta data object;
25     // determine number of rows
26     public ResultSetTableModel( String driver, String url,
27        String query ) throws SQLException, ClassNotFoundException
28     {
29        // load database driver class
30        Class.forName( driver );
31
32        // connect to database
33        connection = DriverManager.getConnection( url );
34
35        // create Statement to query database
36        statement = connection.createStatement(
37           ResultSet.TYPE_SCROLL_INSENSITIVE,
38           ResultSet.CONCUR_READ_ONLY );
39
40        // update database connection status
41        connectedToDatabase = true;
42
43        // set query and execute it
44        setQuery( query );
45     }
46
```

Establishes a `connection` to the database.

Invokes `Connection` method `createStatement` to create a `Statement` object.

Invokes `ResultSetTableModel` method `setQuery` to perform the default query.

ResultSetTableModel enables a Jtable to display the

Line 33

-38

Outline

```
47       // get class that represents column type
48       public Class getColumnClass( int column ) throws IllegalStateException
49       {
50          // ensure database connection is available
51          if ( !connectedToDatabase )
52             throw new IllegalStateException( "Not Connected to Database" );
53
54          // determine Java class of column
55          try {
56             String className = metaData.getColumnClassName( column + 1 );
57
58             // return Class object that represents cla
59             return Class.forName( className );
60          }
61
62          // catch SQLExceptions and ClassNotFoundExceptions
63          catch ( Exception exception ) {
64             exception.printStackTrace();
65          }
66
67          // if problems occur above,                    ject
68          return Object.class;
69       }
70
```

Obtains the fully qualified class name for the specified column.

Loads the class definition for the class and returns the corresponding `Class` object.

Returns the default type.

ResultSetTableModel enables a
to
the
contents of a
ResultSet.

Line 59

Line 68

Outline

```
71        // get number of columns in ResultSet
72        public int getColumnCount() throws IllegalStateException
73        {
74           // ensure database connection is available
75           if ( !connectedToDatabase )
76              throw new IllegalStateException( "Not Connected to Database" );
77
78           // determine number of columns
79           try {
80              return metaData.getColumnCount();
81           }
82
83           // catch SQLExceptions and print error message
84           catch ( SQLException sqlException ) {
85              sqlException.printStackTrace();
86           }
87
88           // if problems occur above, return 0 for number of columns
89           return 0;
90        }
91
92        // get name of a particular column in ResultSet
93        public String getColumnName( int column ) throws IllegalStateException
94        {
95           // ensure database connection is available
96           if ( !connectedToDatabase )
97              throw new IllegalStateException( "Not Connected to Database" );
```

Obtains the number of columns in the `ResultSet`.

`ResultSetTableModel enables a Jtable to display the`
`tents of a`
`ltSet.`

`Line 80`

Outline

```
98
99        // determine column name
100       try {
101          return metaData.getColumnName( column + 1 );
102       }
103
104       // catch SQLExceptions and print error message
105       catch ( SQLException sqlException ) {
106          sqlException.printStackTrace();
107       }
108
109       // if problems, return empty string for column name
110       return "";
111    }
112
113    // return number of rows in ResultSet
114    public int getRowCount() throws IllegalStateException
115    {
116       // ensure database connection is available
117       if ( !connectedToDatabase )
118          throw new IllegalStateException( "Not Connected to Database" );
119
120       return numberOfRows;
121    }
122
```

Obtains the column name from the `ResultSet`.

SetTableM odel enables a Jtable to display the contents of a ResultSet.

Line 101

```
   // obtain value in particular row and column
   public Object getValueAt( int row, int column )
      throws IllegalStateException
   {
      // ensure database connection is available
      if ( !connectedToDatabase )
         throw new IllegalStateException( "Not Connected to Database" );

      // obtain a value at specified ResultSet row and column
      try {
         resultSet.absolute( row + 1 );

         return resultSet.getObject( column + 1 );
      }

      // catch SQLExceptions and print error message
      catch ( SQLException sqlException ) {
         sqlException.printStackTrace();
      }

      // if problems, return empty string object
      return "";
   }

```

ResultSetTableModel enables a Jtable to display the contents of a

Uses ResultSet method absolute to position the ResultSet cursor at a specific row.

Uses ResultSet method getObject to obtain the Object in a specific column of the current row.

Outline

```
147    // set new database query string
148    public void setQuery( String query )
149       throws SQLException, IllegalStateException
150    {
151       // ensure database connection is available
152       if ( !connectedToDatabase )
153          throw new IllegalStateException( "Not Connected to Database" );
154
155       // specify query and execute it
156       resultSet = statement.executeQuery( query );
157
158       // obtain meta data for ResultSet
159       metaData = resultSet.getMetaData();
160
161       // determine number of rows in Result
162       resultSet.last();                     /
163       numberOfRows = resultSet.getRow();    /
164
165       // notify JTable that model has chang
166       fireTableStructureChanged();
167    }
168
```

Executes the query to obtain a new `ResultSet`.

Uses `ResultSet` method `last` `Resu` row

Uses `ResultSet` method `getRow` to obtain the row number for the

Invokes method `fireTableStructureChanged` to notify any `JTable` using this `ResultSetTableModel` object as its model that the structure of the model has changed.

ResultSetTableModel enables a Jtable to display the ts of a Set.

Line 156

Line 162

63

Outline

```
169    // close Statement and Connection
170    public void disconnectFromDatabase()
171    {
172       // close Statement and Connection
173       try {
174          statement.close();
175          connection.close();
176       }
177
178       // catch SQLExceptions and print error message
179       catch ( SQLException sqlException ) {
180          sqlException.printStackTrace();
181       }
182
183       // update database connection status
184       finally {
185          connectedToDatabase = false;
186       }
187    }
188
189 }  // end class ResultSetTableModel
```

Close the `Statement` and `Connection` if a `ResultSetTableModel` object is garbage collected.

ResultSetTableM **ables a** **to** **the** **s of a** **ResultSet.**

Lines 174-175

23.6.2 Querying the books Database (Cont.)

ResultSet static type constant	Description
`TYPE_FORWARD_ONLY`	
	Specifies that a `ResultSet`'s cursor can move only in the forward direction (i.e., from the first row to the last row in the `ResultSet`).
`TYPE_SCROLL_INSENSITIVE`	
	Specifies that a `ResultSet`'s cursor can scroll in either direction and that the changes made to the `ResultSet` during `ResultSet` processing are not reflected in the `ResultSet` unless the program queries the database again.
`TYPE_SCROLL_SENSITIVE`	
	Specifies that a `ResultSet`'s cursor can scroll in either direction and that the changes made to the `ResultSet` during `ResultSet` processing are reflected immediately in the `ResultSet`.

Fig. 23.28 **`ResultSet`** constants for specifying `ResultSet` type.

23.6.2 Querying the books Database (Cont.)

`ResultSet static` concurrency constant	Description
`CONCUR_READ_ONLY`	Specifies that a `ResultSet` cannot be updated (i.e., changes to the `ResultSet` contents cannot be reflected in the database with `ResultSet`'s *update* methods).
`CONCUR_UPDATABLE`	Specifies that a `ResultSet` can be updated (i.e., changes to the `ResultSet` contents can be reflected in the database with `ResultSet`'s *update* methods).

Fig. 23.29 `ResultSet` constants for specifying result properties.

Outline

```
1   // Fig. 23.30: DisplayQueryResults.java
2   // Display the contents of the Authors table in the
3   // Books database.
4
5   import java.awt.*;
6   import java.awt.event.*;
7   import java.sql.*;
8   import java.util.*;
9   import javax.swing.*;
10  import javax.swing.table.*;
11
12  public class DisplayQueryResults extends JFrame {
13
14     // JDBC driver and database URL
15     static final String JDBC_DRIVER = "com.ibm.db2j.jdbc.DB2jDriver";
16     static final String DATABASE_URL = "jdbc:db2j:books";
17
18     // default query selects all rows from authors table
19     static final String DEFAULT_QUERY = "SELECT * FROM authors";
20
21     private ResultSetTableModel tableModel;
22     private JTextArea queryArea;
23
24       // create ResultSetTableModel and GUI
25       public DisplayQueryResults()
26       {
27          super( "Displaying Query Results" );
```

DisplayQueryResults for querying database books.

Lines 15, 16, 19

Define the database driver class name, database URL and default query.

Outline

DisplayQueryResults for querying database books.

```

         // create ResultSetTableModel and display database table
         try {

            // specify location of database on filesystem
            System.setProperty( "db2j.system.home", "C:/Cloudscape_5.0" );

            // create TableModel for results of query SELECT * FROM authors
            tableModel = new ResultSetTableModel( JDBC_DRIVER, DATABASE_URL,
               DEFAULT_QUERY );

            // set up JTextArea in which user types queries
            queryArea = new JTextArea( DEFAULT_QUERY, 3, 100 );
            queryArea.setWrapStyleWord( true );
            queryArea.setLineWrap( true );

            JScrollPane scrollPane = new JScrollPane( queryArea,
               ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED,
               ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER );

            // set up JButton for submitting queries
            JButton submitButton = new JButton( "Submit Query" );

            // create Box to manage placement of queryArea and
            // submitButton in GUI
            Box box = Box.createHorizontalBox();
            box.add( scrollPane );
            box.add( submitButton );

```

Create `TableModel` for results of query `SELECT * FROM authors`

Outline

```
57          // create JTable delegate for tableModel
58          JTable resultTable = new JTable( tableModel );
59
60          // place GUI components on content pane
61          Container c = getContentPane();
62          c.add( box, BorderLayout.NORTH );
63          c.add( new JScrollPane( resultTable ), BorderLayout.CENTER );
64
65          // create event listener for submitButton
66          submitButton.addActionListener(
67
68             new ActionListener() {
69
70                // pass query to table model
71                public void actionPerformed( ActionEvent event )
72                {
73                   // perform a new query
74                   try {
75                      tableModel.setQuery( queryArea.getText() );
76                   }
77
78                   // catch SQLExceptions when performing a n
79                   catch ( SQLException sqlException ) {
80                      JOptionPane.showMessageDialog( null,
81                         sqlException.getMessage(), "Database error",
82                         JOptionPane.ERROR_MESSAGE );
83
```

Create `JTable` delegate for `tableModel`

Register an event handler for the `submitButton` that the user clicks to submit a query to the database.

Invokes `ResultSetTableModel` method `setQuery` to execute the new query.

DisplayQueryResults for querying database books.

e 58

es 66-110

Line 75

```
                           // try to recover from invalid user query
                           // by executing default query
                           try {
                              tableModel.setQuery( DEFAULT_QUERY );
                              queryArea.setText( DEFAULT_QUERY );
                           }

                           // catch SQLException when performing default query
                           catch ( SQLException sqlException2 ) {
                              JOptionPane.showMessageDialog( null,
                                 sqlException2.getMessage(), "Database error",
                                 JOptionPane.ERROR_MESSAGE );

                              // ensure database connection is closed
                              tableModel.disconnectFromDatabase();

                             System.exit( 1 );   // terminate application

                          }  // end inner catch

                       } // end outer catch

                    }  // end actionPerformed

                 }  // end ActionListener inner class

              ); // end call to addActionListener

```

DisplayQueryResults for querying database books.

```
         // set window size and display window
         setSize( 500, 250 );
         setVisible( true );

      }  // end try

      // catch ClassNotFoundException thrown by
      // ResultSetTableModel if database driver not found
      catch ( ClassNotFoundException classNotFound ) {
         JOptionPane.showMessageDialog( null,
            "Cloudscape driver not found", "Driver not found",
            JOptionPane.ERROR_MESSAGE );

         System.exit( 1 );   // terminate application
      } // end catch

      // catch SQLException thrown by ResultSetTableModel
      // if problems occur while setting up database
      // connection and querying database
      catch ( SQLException sqlException ) {
         JOptionPane.showMessageDialog( null, sqlException.getMessage(),
            "Database error", JOptionPane.ERROR_MESSAGE );

         // ensure database connection is closed
         tableModel.disconnectFromDatabase();

         System.exit( 1 );   // terminate application
      }

```

DisplayQueryResults for querying database books.

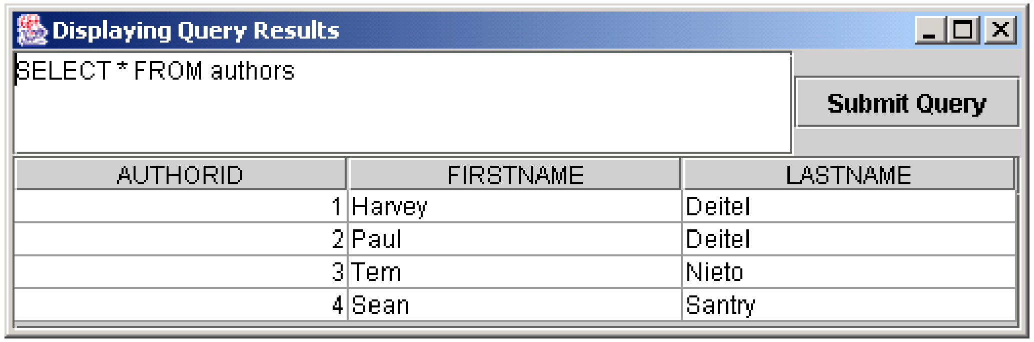
```
      // dispose of window when user quits application (this overrides
      // the default of HIDE_ON_CLOSE)
      setDefaultCloseOperation( DISPOSE_ON_CLOSE );

      // ensure database connection is closed when user quits application
      addWindowListener(

         new WindowAdapter() {

            // disconnect from database and exit when window has closed
            public void windowClosed( WindowEvent event )
            {
               tableModel.disconnectFromDatabase();
               System.exit( 0 );
            }
         }
      );

   }  // end DisplayQueryResults constructor

   // execute application
   public static void main( String args[] )
   {
      new DisplayQueryResults();
   }

}  // end class DisplayQueryResults
```

DisplayQueryResults for querying database books.

Outline

DisplayQueryResults for querying database books.

Program output

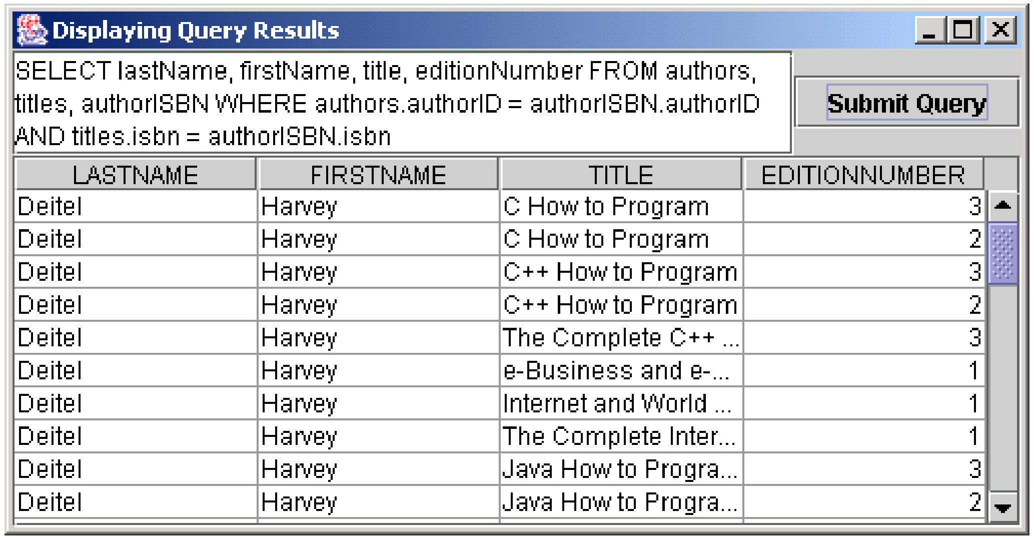


Displaying Query Results

```
SELECT * FROM authors
```

Submit Query

AUTHORID	FIRSTNAME	LASTNAME
1	Harvey	Deitel
2	Paul	Deitel
3	Tem	Nieto
4	Sean	Santry



Displaying Query Results

```
SELECT lastName, firstName, title, editionNumber FROM authors,
titles, authorISBN WHERE authors.authorID = authorISBN.authorID
AND titles.isbn = authorISBN.isbn
```

Submit Query

LASTNAME	FIRSTNAME	TITLE	EDITIONNUMBER
Deitel	Harvey	C How to Program	3
Deitel	Harvey	C How to Program	2
Deitel	Harvey	C++ How to Program	3
Deitel	Harvey	C++ How to Program	2
Deitel	Harvey	The Complete C++ ...	3
Deitel	Harvey	e-Business and e-...	1
Deitel	Harvey	Internet and World ...	1
Deitel	Harvey	The Complete Inter...	1
Deitel	Harvey	Java How to Progra...	3
Deitel	Harvey	Java How to Progra...	2