University of Sheffield

COM2008 : Systems Design and Security

Tutorial 04 - Combining Multiple Tables

Department of Computer Science

Contents

Chapter 1

Introduction

1.1 Learning Objectives

1. Define and structure tables in MySQL.
2. Insert data into tables.
3. Execute queries on the populated database.
4. Manage database connections.
5. Perform database operations.
6. Display data using JTables.

Chapter 2

Populating The Database

2.1 Creating the Tables

2.1.1 The BookTitle Table

The ”BookTitle” table is designed to store information about books. It has three columns: ”isbn,” ”title,” and ”author.” The ”isbn” column is the primary key, ensuring that each book is uniquely identified by its International Standard Book Number (ISBN). The ”title” column is a VARCHAR(30) that can store the title of the book, and the ”author” column is a VARCHAR(20) for the name of the book’s author. This table serves as a foundational part of a database system, enabling the organization and retrieval of book details for various purposes, such as library cataloging or book sales management.

```
-- Table to store information about books.
CREATE TABLE BookTitle (
    isbn VARCHAR(13) NOT NULL PRIMARY KEY,   -- ISBN (unique identifier)
    title VARCHAR(30),                       -- Title of the book
    author VARCHAR(20),                      -- Author of the book
    publisher VARCHAR(30),                   -- Publisher of the book
    datePublished DATE                        -- Date the book was
       published
);
```

2.1.2 The BookCopy Table

The ”BookCopy” table is designed to store information about individual copies of books. It consists of three columns: ”copyID,” ”isbn,” and a foreign key reference to the ”BookTitle” table. The ”copyID” column is the primary key and uses a CHAR(30) data type to uniquely identify each book copy. The ”isbn” column stores the International Standard Book Number (ISBN) of the book to which the copy belongs and is linked to the ”BookTitle” table’s ISBN as a foreign key constraint. This table facilitates the tracking and management of distinct physical copies of books within a library or inventory system, ensuring that each copy can

be associated with its corresponding book title.

```
-- Table to store information about individual copies of books.
CREATE TABLE BookCopy (
  copyID CHAR(30) NOT NULL PRIMARY KEY,
  isbn VARCHAR(13),
  FOREIGN KEY (isbn) REFERENCES BookTitle(isbn)
);
```

2.1.3 The Borrower Table

The ”Borrower” table serves as a repository for information related to library members or borrowers. It comprises five columns: ”memberID,” ”forename,” ”surname,” ”address,” and ”email.” The ”memberID” column, defined as an integer with a maximum length of 10 digits, functions as the primary key, ensuring each library member’s unique identification. ”Forename” and ”surname” are VARCHAR(50) columns that store the first name and last name of the borrowers, respectively. The ”address” column is capable of storing more extensive address details (up to 100 characters), and the ”email” column records the email addresses of borrowers. This table is essential for maintaining a record of library members, enabling the library to manage their personal information, track their borrowing history, and communicate with them efficiently for notifications and updates.

```
-- Table to store information about library members/borrowers.
CREATE TABLE Borrower (
  memberID INT(10) NOT NULL PRIMARY KEY,
  forename VARCHAR(50),
  surname VARCHAR(50),
  email VARCHAR(50)
);
```

2.1.4 The Loan Table

The ”Loan” table is designed to manage book loans within a library or similar lending system. It contains four main columns: ”memberID,” ”copyID,” ”issueDate,” and ”dueDate.” The ”memberID” and ”copyID” columns, both set as NOT NULL, are used to establish a unique combination as the primary key, ensuring each loan is uniquely identified by the borrower’s member ID and the book copy’s ID. The ”issueDate” and ”dueDate” columns are of the DATE data type, representing the date when a book is borrowed and the date it is due to be returned, respectively. The table enforces referential integrity through two foreign key constraints: one connecting ”memberID” to the ”Borrower” table, ensuring that the borrower exists, and the other connecting ”copyID” to the ”BookCopy” table, ensuring the book copy exists. This ”Loan” table is crucial for tracking and managing book loans, allowing the li-

brary to monitor due dates, handle returns, and ensure accountability for borrowed materials.

```
-- Table to manage book loans.
CREATE TABLE Loan (
  memberID INT(10) NOT NULL,
  copyID CHAR(30) NOT NULL,
  issueDate DATE,
  dueDate DATE,
  PRIMARY KEY (memberID, copyID),
  FOREIGN KEY (memberID) REFERENCES Borrower(memberID),
  FOREIGN KEY (copyID) REFERENCES BookCopy(copyID)
);
```

2.2 Populating the Tables

2.2.1 Populating the BookTitle table

This SQL query is used to insert records into the ”BookTitle” table. It’s inserting four distinct rows into the table, with each row representing a book’s information. The columns being populated are ”isbn,” ”title,” and ”author.” The values include the International Standard Book Number (ISBN), the title of the book, and the name of the author. This query efficiently adds books, such as ”To Kill a Mockingbird” by Harper Lee, ”1984” by George Orwell, ”The Great Gatsby” by F. Scott Fitzgerald, and ”Animal Farm” by George Orwell, into the database, making them available for retrieval and management within the system.

```
-- Insert records into the BookTitle table
INSERT INTO BookTitle (isbn, title, author, publisher, datePublished)
   VALUES
('9780451524935', 'To Kill a Mockingbird', 'Harper Lee', 'PublisherA', '
   1960-07-11'),
('9780061120084', '1984', 'George Orwell', 'PublisherB', '1949-06-08'),
('9780062390747', 'The Great Gatsby', 'F. Scott Fitzgerald', 'PublisherC'
   , '1925-04-10'),
('9780141983767', 'Animal Farm', 'George Orwell', 'PublisherB', '
   1945-08-17');
```

2.2.2 Populating the BookCopy table

This SQL query is used to insert records into the ”BookCopy” table, which is intended to manage individual copies of books within a database. The query inserts six rows into the table, with each row representing a unique book copy. Two columns are populated: ”copyID” and

”isbn.” The ”copyID” serves as a unique identifier for each book copy, and the ”isbn” specifies the International Standard Book Number (ISBN) of the corresponding book title, forming a connection to the ”BookTitle” table. The query efficiently records book copy information, such as unique identification codes (’ABC123456789’, ’DEF987654321’, etc.) and their associated book ISBNs, enabling the system to keep track of specific physical copies of books and link them to their respective titles for effective inventory management and tracking.

```
-- Insert records into the BookCopy table
INSERT INTO BookCopy (copyID, isbn) VALUES
('ABC123456789', 9780451524935),
('DEF987654321', 9780451524935),
('GHI123987654', 9780061120084),
('JKL456123789', 9780061120084),
('MNO789654321', 9780141983767),
('PQR654987321', 9780141983767);
```

2.2.3 Populating the Borrower table

This SQL query is used to insert records into the ”Borrower” table, which is intended to store information about library members or borrowers. The query inserts four distinct rows into the table, with each row representing a unique borrower. The columns being populated are ”memberID,” ”forename,” ”surname,” ”address,” and ”email.” The ”memberID” column acts as a unique identifier for each borrower, ensuring their individual recognition within the system. The ”forename” and ”surname” columns store the borrower’s first name and last name, respectively. The ”address” column accommodates a more comprehensive address, and the ”email” column records the borrower’s email address. This query efficiently captures borrower details, such as names, contact information, and unique identifiers, allowing the system to manage and communicate with library members effectively, while also facilitating record-keeping and tracking of borrowing history.

```
INSERT INTO Borrower (memberID, forename, surname, email) VALUES
(1, 'John', 'Doe', 'john.doe@email.com'),  -- John Doe
(2, 'Jane', 'Smith', 'jane.smith@email.com'),  -- Jane Smith
(3, 'Bob', 'Johnson', 'bob.johnson@email.com'),  -- Bob Johnson
(4, 'Alice', 'Brown', 'alice.brown@email.com');  -- Alice Brown
```

2.2.4 Populating the Loan table

This SQL query is used to insert records into the ”Loan” table, which is responsible for managing book loans within a library or lending system. It inserts five distinct rows into the table, each representing a unique loan transaction. The columns being populated are ”memberID,” ”copyID,” ”issueDate,” and ”dueDate.” The ”memberID” and ”copyID” columns

serve as references to borrowers and specific book copies, ensuring that each loan is associated with a library member and a book copy. The ”issueDate” records the date when the book is borrowed, and the ”dueDate” is calculated using the DATE_ADD function, which adds 14 days to the issue date, representing the date by which the book should be returned. This query efficiently captures loan details, facilitating the tracking of borrowing and return dates, helping libraries manage their inventory, and ensuring that borrowed materials are returned promptly.

```
  -- Insert records into the Loan table
INSERT INTO Loan (memberID, copyID, issueDate, dueDate) VALUES
(1, 'MNO789654321', '2023-10-01', DATE_ADD('2023-10-01', INTERVAL 14 DAY)
   ),
(1, 'ABC123456789', '2023-10-15', DATE_ADD('2023-10-15', INTERVAL 14 DAY)
   ),
(2, 'DEF987654321', '2023-10-20', DATE_ADD('2023-10-20', INTERVAL 14 DAY)
   ),
(3, 'GHI123987654', '2023-10-25', DATE_ADD('2023-10-25', INTERVAL 14 DAY)
   ),
(4, 'JKL456123789', '2023-10-30', DATE_ADD('2023-10-30', INTERVAL 14 DAY)
   );
```

2.3 Trialing the Query

```
-- Select information about overdue books and their borrowers
SELECT b.memberID, b.forename, b.surname, t.title, t.isbn, c.copyID, m.
   dueDate
FROM Borrower b, BookTitle t, BookCopy c, Loan m
WHERE m.dueDate < CURRENT_DATE
  AND m.copyID = c.copyID
  AND c.isbn = t.isbn
  AND m.memberID = b.memberID;
```

This SQL query retrieves information about overdue books and their respective borrowers by querying multiple related tables in the database.

Let’s break down the query step by step:

- `SELECT` Clause: The `SELECT` clause specifies the columns that should be included in the query result. It selects several columns: memberID, forename, surname, title, isbn, copyID, and dueDate. These columns represent borrower information (first name, last name, and member ID), book title information (title and ISBN), and loan information (copy ID and due date).

- `FROM` Clause: The `FROM` clause specifies the tables involved in the query. It lists four tables: Borrower (aliased as 'b'), BookTitle (aliased as 't'), BookCopy (aliased as 'c'), and Loan (aliased as 'm'). These tables are joined together to retrieve the required information.
- `WHERE` Clause: The `WHERE` clause sets conditions for selecting rows from the tables. The conditions are as follows:
  - `m.dueDate < CURRENT_DATE`: This condition checks whether the due date for a loan is earlier than the current date, which identifies loans that are overdue.
  - `m.copyID = c.copyID`: This condition ensures that the copyID in the Loan table matches the copyID in the BookCopy table, connecting loans to specific book copies.
  - `c.isbn = t.isbn`: This condition links the BookCopy table to the BookTitle table through the ISBN, allowing the retrieval of book title information.
  - `m.memberID = b.memberID`: This condition associates loans with borrowers by matching the memberID in the Loan table to the memberID in the Borrower table.

In summary, this query retrieves information about overdue books by joining the `Borrower`, `BookTitle`, `BookCopy`, and `Loan` tables. It identifies overdue loans by comparing the due date to the current date and returns details about borrowers (forename, surname, and member ID), the titles of overdue books (title and ISBN), the specific book copies (copyID), and the due dates. This information can be used to manage overdue books, notify borrowers, and take necessary actions to ensure the return of the borrowed materials.

Chapter 3

Using JTables to Display Data

3.1 The Main Class

The `Main` class is the entry point of a Java application. It serves as the starting point for the program's execution. Let's break down the key aspects of the Main class:

1. **Package and Imports**: The `Main` class is part of the `com.sheffield` package. It imports two important classes: `DatabaseConnectionHandler` and `LoanTableDisplay` from other packages. These imports are necessary to access these classes and their functionalities within the Main class.

2. **`main` Method**: The main method is the primary method for executing Java applications. It's where the program begins its execution. In this case, it's the starting point for the Swing-based GUI application. Within the main method, the following steps are taken:

    - An instance of `DatabaseConnectionHandler` is created. This class is presumably responsible for managing database connections and interactions.
    - The `SwingUtilities.invokeLater()` method is used to ensure that the GUI-related code is executed on the Event Dispatch Thread (EDT). The EDT is responsible for handling Swing components and ensures that the GUI remains responsive.
    - Inside the `invokeLater()` block, several actions are performed:
        - The `LoanTableDisplay` view, which displays information about loans, is created and initialized. This view likely interacts with the database via the `DatabaseConnectionHandler`.
        - A try-catch block handles potential exceptions during the process, with any errors resulting in a runtime exception.
        - Finally, the database connection is closed to release resources, and the `LoanTableDisplay` is made visible to the user.

The Main class essentially coordinates the setup of a Swing-based GUI application and handles the opening and closing of a database connection through the use of the `DatabaseConnectionHandler`. It adheres to best practices by running the GUI code on the

EDT, ensuring a responsive user interface. This class's main purpose is to kickstart the application and set the stage for user interaction with the loan display system.

3.2 The DatabaseConnectionHandler Class

The `DatabaseConnectionHandler` class is responsible for managing and handling database connections within a Java application. Here's an explanation of its key components and purpose:

- Import Statements: This class imports the necessary Java libraries to work with databases, including `java.sql.Connection`, `java.sql.DriverManager`, and `java.sql.SQLException`. These classes are essential for establishing and managing database connections.
- Class Fields: The class defines several fields:
  - `DB_URL`, `DB_USER`, and `DB_PASSWORD`: These are constants representing the URL of the MySQL database, the database username, and the password. They are used to configure the database connection.
  - `connection`: This field represents the database connection itself and is kept as a class member so that it can be shared and reused across the application.
- `openConnection` Method: This method is used to establish a connection to the database. Within this method:
  - The JDBC (Java Database Connectivity) driver is loaded, which is necessary for database communication.
  - The `DriverManager.getConnection` method is used to connect to the database using the provided URL, username, and password. If the connection is successful, the connection field is set to the established connection.
- `closeConnection` Method: This method is responsible for closing the database connection. It ensures proper resource management and prevents resource leaks. If the connection is not null, it is closed using the connection.close() method.
- `getConnection` Method: This method is a getter that allows other parts of the application to obtain the database connection. It returns the connection field, which can be used to execute SQL queries and interact with the database.

The `DatabaseConnectionHandler` class encapsulates the logic for establishing, managing, and closing database connections. It follows best practices for resource management and provides a straightforward interface for other parts of the application to access the database connection when needed. This class can be a crucial component in applications that interact with databases.

3.3 The DatabaseOperations Class

The `DatabaseOperations` class is responsible for executing SQL queries related to retrieving information about books that are overdue in a library database. This class provides a method, `getBooksOverDueDate`, which takes a database connection as a parameter and executes a SQL query to fetch data on borrowers who have overdue books. The query involves joining several tables, including `Borrower`, `BookTitle`, `BookCopy`, and `Loan`, to retrieve details such as member ID, forename, surname, book title, ISBN, copy ID, and due date for books that have exceeded their due date. The method encapsulates database interaction, error handling, and the execution of the SQL query. It returns a `ResultSet` containing the queried data, which can be processed further by the calling code. This class aids in managing database operations and extracting relevant information for library management or similar applications.

This method has been left for you to implement. Please use the screenshot below.

```
11      public ResultSet getBooksOverDueDate(Connection connection) throws SQLException {
12          ResultSet resultSet = null;
13          try {
14              // Establish a database connection
15
16              // Execute the SQL query
17              String sqlQuery = "SELECT b.memberID, b.forename, b.surname, t.title, t.isbn, c.copyID, m.dueDate " +
18                      "FROM Borrower b, BookTitle t, BookCopy c, Loan m " +
19                      "WHERE m.dueDate < CURRENT_DATE " +
20                      "AND m.copyID = c.copyID " +
21                      "AND c.isbn = t.isbn " +
22                      "AND m.memberID = b.memberID";
23              PreparedStatement statement = connection.prepareStatement(sqlQuery);
24              resultSet = statement.executeQuery(sqlQuery);
25          } catch (Exception e) {
26              e.printStackTrace();
27          }
28          return resultSet;
29      }
```

Figure 3.1: getBooksOverDueDate function

3.4 The LoanTableDisplay Class

The `LoanTableDisplay` class is a Swing-based Java GUI application that displays information about overdue loans from a database. It extends the JFrame class to create a window for presenting the data. Inside the window, a JTable is used to present the loan information, and a `DefaultTableModel` is used to define the table's structure. The class interacts with the database through the `DatabaseOperations` class, specifically the `getBooksOverDueDate` method, which retrieves a `ResultSet` containing data on overdue books. It iterates through the ResultSet and populates the JTable with information such as member ID, forename, surname, book title, ISBN, copy ID, and due date.

This constructor has been left for you to implement. Please use the screenshot below.

```
1 usage
15  public LoanTableDisplay(Connection connection) throws SQLException {
16      setTitle("Loan Information");
17      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
18      setSize( width: 800,  height: 400);
```

Figure 3.2: LoanTableDisplay lines 15-18

```
19
20      // Create a DefaultTableModel for the JTable
21      DefaultTableModel tableModel = new DefaultTableModel();
22      loanTable = new JTable(tableModel);
23      tableModel.addColumn( columnName: "Member ID");
24      tableModel.addColumn( columnName: "Forename");
25      tableModel.addColumn( columnName: "Surname");
26      tableModel.addColumn( columnName: "Title");
27      tableModel.addColumn( columnName: "ISBN");
28      tableModel.addColumn( columnName: "Copy ID");
29      tableModel.addColumn( columnName: "Due Date");
30
```

Figure 3.3: LoanTableDisplay lines 19-29

```
31      DatabaseOperations databaseOperations = new DatabaseOperations();
32      ResultSet resultSet = databaseOperations.getBooksOverDueDate(connection);
33      // Populate the JTable with the query results
34      while (resultSet.next()) {
35          tableModel.addRow(new Object[]{
36                  resultSet.getInt( columnLabel: "memberID"),
37                  resultSet.getString( columnLabel: "forename"),
38                  resultSet.getString( columnLabel: "surname"),
39                  resultSet.getString( columnLabel: "title"),
40                  resultSet.getString( columnLabel: "isbn"),
41                  resultSet.getString( columnLabel: "copyID"),
42                  resultSet.getDate( columnLabel: "dueDate")
43          });
44      }
```

Figure 3.4: LoanTableDisplay lines 31-44

```
45
46          resultSet.close();
47          connection.close();
48          // Create a JScrollPane to display the table
49          JScrollPane scrollPane = new JScrollPane(loanTable);
50          getContentPane().add(scrollPane, BorderLayout.CENTER);
51      }
```

Figure 3.5: LoanTableDisplay lines 45-51

Chapter 4

Conclusion

In this tutorial, we have covered a comprehensive set of topics related to the creation and management of a database, as well as the display of data using Java Swing components. The learning journey was structured into the following key sections:

We set the stage by introducing the tutorial's purpose and objectives. This prepared us for the content that follows. We delved into the fundamental aspects of database management, focusing on creating tables for different entities such as books, copies, borrowers, and loans. We provided insight into the structure and organization of these tables. Building upon our understanding of table creation, we explored how to populate these tables with relevant data. We looked into each table individually and discussed the process of inserting data. To ensure the effectiveness of our database operations, we learned how to execute queries, particularly focusing on querying for data based on due dates.

Moving from the backend, we shifted our attention to the user interface. We explored how to create a user-friendly display of data using JTables in Java Swing. This section included discussions on the Main Class, DatabaseConnectionHandler, DatabaseOperations, and the LoanTableDisplay class.

By the end of this tutorial, you should have gained a solid foundation in creating and managing a database for a library system. You have learned how to query the database and display the information in a user-friendly manner. These skills are valuable not only for library management systems but also for various other applications that involve databases and user interfaces. We encourage you to apply these concepts and adapt them to your specific projects and requirements. With this knowledge, you are well-equipped to develop effective and efficient database-driven applications, providing users with a seamless experience.