Qt Database Integration for Qt Developers

Exam Structure

- 5 progressive exercises (30 minutes each)
- Starts with basic concepts then practical implementation
- Focuses on SQLite (simplest database for beginners)
- Solutions involve both code and conceptual understanding

1 Exercise 1: Database Fundamentals (30 minutes)

1. What is a database? Give 3 examples of applications that use databases.
2. Which Qt module must be added to the .pro file for database support?
3. What makes SQLite different from other database systems?
4. Write Qt code to:
   (a) Create a SQLite database connection named "my_first_db.db"
   (b) Open the connection and check if it succeeded
   (c) Close the connection properly
5. Where in a Qt application's lifecycle should database connections be established?

2 Exercise 2: Your First Database Table (30 minutes)

We'll create a "Tasks" table with:

- id (integer, primary key)
- description (text)
- due_date (text)
- completed (boolean)

1. Write the Qt code to create this table using QSqlQuery
2. Explain the difference between these two approaches:

```
query.exec("INSERT INTO Tasks VALUES(1, 'Learn Qt', '2023-12-01', 0)");
```

vs.

```
query.prepare("INSERT INTO Tasks VALUES(?, ?, ?)");
query.bindValue(0, 1);
// ... etc ...
query.exec();
```

3. Add 3 sample tasks to the table using the safer approach
4. Write a query to count all incomplete tasks
5. What does QSqlQuery::lastError() do and when should you use it?

3 Exercise 3: Displaying Data in Qt (30 minutes)

1. Create a basic Qt Widgets application with:

- A QTableView
- A QPushButton labeled "Refresh Data"

2. Connect to your database and display the Tasks table using QSqlTableModel
3. Implement the refresh button to reload data from the database
4. Add a QCheckBox to filter and show only incomplete tasks
5. What are the advantages of using QSqlTableModel versus direct SQL queries?

4 Exercise 4: Building a Task Manager (30 minutes)

Enhance your application:

1. Add UI elements for:
   - Adding new tasks (description and due date)
   - Marking tasks as complete
   - Deleting tasks
2. Implement the "Add Task" functionality with input validation
3. Implement task completion/deletion for selected tasks
4. Add a status bar message showing the count of pending tasks
5. How would you modify this to support editing existing tasks?

5 Exercise 5: Database Best Practices (30 minutes)

1. Why is proper database connection cleanup important?
2. What are database transactions and how would you use them in Qt?

3. Explain how to prevent SQL injection attacks in Qt applications
4. Where should database connection code be placed in a MVC Qt application?
5. Propose a project structure for a medium-sized Qt database application

Learning Outcomes

After completing this exam, students should be able to:

- Understand basic database concepts
- Set up and manage database connections in Qt
- Perform CRUD operations safely
- Display database content in Qt views
- Apply basic database security practices