Project 2
CSE4110 and AIE4055
Database Systems
Spring 2025

Database Implementation and Application Development

In this course project, you will transform your conceptual E-R design from Project 1 into a fully functional database system. This project involves logical schema design, normalization, physical implementation, and application development using MySQL and C/C++.

Submission Rules

- Late submissions will not be accepted.
- Due date: **Monday, June 9, 2025**
- Submit all required files as specified in the deliverables section
- Upload your submission to the assignment section on Cybercampus

1 Goal

The goal of this project is to provide comprehensive experience in the complete database development lifecycle, from conceptual design to implementation and application development. You will work individually to:

1. Transform your E-R diagram into a logical relational schema
2. Normalize the schema to BCNF (Boyce-Codd Normal Form)
3. Design and implement a physical database schema
4. Create and populate a MySQL database
5. Develop a C/C++ application using ODBC and MySQL C API
6. Implement all sample queries from Project 1 as functional database operations

2 Project Requirements

2.1 Logical Schema Design

Transform your E-R diagram from Project 1 into a relational schema following standard reduction rules:

- Convert entities to relations with appropriate primary keys
- Handle different relationship types (1:1, 1:N, M:N) according to standard mapping rules
- Resolve multi-valued attributes and composite attributes
- Document foreign key relationships and referential integrity constraints

2.2 BCNF Normalization

Analyze your logical schema and decompose relations that are not in BCNF:

- Identify functional dependencies in each relation
- Check for BCNF violations (non-trivial functional dependencies where the left side is not a superkey)
- Decompose violating relations using the BCNF decomposition algorithm
- Ensure the decomposition preserves dependencies where possible
- Document the normalization process with detailed explanations

2.3 Physical Schema Design

Design the physical implementation of your normalized logical schema:

- Define appropriate data types for all attributes (VARCHAR, INT, DECIMAL, DATE, etc.)
- Specify constraints (NOT NULL, UNIQUE, CHECK constraints)
- Design indexes for performance optimization
- Define triggers if necessary for business rule enforcement
- Create the schema using MySQL Workbench with proper documentation

2.4 Database Implementation

Create and populate your database in MySQL:

- Use MySQL Workbench to implement your physical schema
- Create comprehensive sample data that reflects real-world scenarios
- Ensure data consistency and referential integrity
- Include sufficient data volume to demonstrate query functionality
- Export your database structure and data as SQL scripts

2.5 Application Development

Develop a C/C++ application that interfaces with your MySQL database:

- Use both ODBC and MySQL C API for database connectivity
- Implement a menu-driven interface for query execution
- Handle all sample queries from Project 1 as functional operations
- Include proper error handling and user input validation
- Provide clear output formatting for query results

2.6 Query Implementation

Implement all sample queries from Project 1 as working SQL queries in your application:

1. **Product Availability**: Find stores carrying specific products with inventory levels
2. **Top-Selling Items**: Identify highest sales volume products per store
3. **Store Performance**: Determine highest revenue-generating stores
4. **Vendor Statistics**: Analyze vendor product supply and sales data
5. **Inventory Alerts**: Identify products below reorder thresholds
6. **Customer Patterns**: Analyze customer purchase behaviors (if implemented)
7. **Franchise Comparison**: Compare product variety between store types

3 Technical Specifications

3.1 Development Environment

- **Database**: MySQL 8.0 or later
- **Design Tool**: MySQL Workbench
- **Programming Language**: C/C++
- **Compiler**: GCC or Visual Studio
- **Database Connectivity**: ODBC and MySQL C API

3.2 Application Requirements

Your C/C++ application must include:

- Menu-driven interface for query selection
- Support for both ODBC and MySQL C API connections
- Parameterized queries to prevent SQL injection

- Proper memory management and connection handling
- Clear error messages and user feedback
- Well-commented code with proper documentation

3.3 Code Structure Example

```
// Main menu structure
int main() {
    while (true) {
        displayMenu();
        int choice = getUserChoice();

        switch (choice) {
            case 1: executeProductAvailabilityQuery(); break;
            case 2: executeTopSellingItemsQuery(); break;
            // ... other cases
            case 0: exitProgram(); break;
        }
    }
    return 0;
}
```

4 Sample Data Guidelines

To ensure consistency and realism in your database implementation:

4.1 Data Requirements

- **Minimum Volume:** Each table in your database must contain at least 10 records
- **Data Quality:** Sample data should be realistic and consistent with real-world convenience store operations
- **Referential Integrity:** All foreign key relationships must be properly maintained across tables

4.2 Data Source Reference

- Use the sample data available at `https://retaildb.or.kr/data/sample` as a reference
- Adapt the provided retail data to match your specific convenience store schema
- You may modify product names, prices, and other attributes to fit your design
- Ensure that your sample data covers various scenarios needed to test all implemented queries

4.3 Data Creation Strategy

- Create diverse product categories (snacks, beverages, household items, etc.)
- Include multiple store locations with different characteristics
- Generate realistic sales transactions across different time periods
- Ensure vendor-product relationships reflect real supply chain scenarios
- Include both franchise and corporate store examples if your design supports this distinction

5 Deliverables

Submit the following files by the due date:

5.1 Database Design Documents

- **Logical Schema Diagram** (.png): Visual representation of your relational schema
- **Physical Schema Diagram** (.png): Complete MySQL Workbench export
- **Database Creation Script** (.sql): Script to recreate your database with sample data

5.2 Source Code

- **Application Source** (.cpp or .c): Complete C/C++ application code
- **Header Files** (.h): Any custom header files
- **README** (.md): Documentation for using your app.

5.3 Documentation

- **Technical Report** (.pdf): Comprehensive project documentation. Should include an explanation of schema design and SQL implementation.

6 Report Specification

Your technical report must include the following sections:

6.1 Logical Schema Design (25%)

- Detailed explanation of ERD to relational schema transformation
- Justification for design decisions

6.2 Normalization Analysis (20%)

- Functional dependency analysis for each relation
- Step-by-step BCNF decomposition process
- **Proof that final schema satisfies BCNF**

6.3 Physical Implementation (15%)

- Data type selection rationale
- Constraint implementation and business rule enforcement
- **sample data description**

6.4 Application Development (30%)

- Database connectivity implementation details
- Query implementation
- Error handling and user interface design

6.5 Testing and Validation (10%)

- Test case descriptions and results
- Validation of business rule enforcement

7 Grading Criteria

Your submission will be evaluated based on:

7.1 Database Design and Normalization (40%)

- Correct logical schema transformation (15%)
- Proper BCNF normalization with clear documentation (15%)
- Appropriate physical schema design (10%)

7.2 Implementation Quality (30%)

- Functional database with consistent sample data (20%)
- Working C/C++ application with proper connectivity (10%)

7.3 Query Functionality (20%)

- All sample queries implemented and working correctly
- Proper SQL query optimization and performance

7.4 Documentation and Presentation (10%)

- Clear, comprehensive technical documentation
- Well-commented, readable code
- Professional report formatting and organization

8 Important Notes

- **Academic Integrity**: **All work must be your own**. Collaboration on concepts is allowed, but code and documentation must be individual work.
- **Testing**: Ensure your application compiles and runs correctly before submission.
- **Help**: Please post all project-related inquiries on the Cybercampus QA board.

9 Submission Format

Create a compressed archive (.zip) named `StudentID_Project2` containing:

```
StudentID_Project2/
 docs/
    logical_schema.png
    physical_schema.png
    project_report.pdf
 database/
    schema.sql
    sample_data.sql
 src/
     main.cpp (or main.c)
     database.h
     README.md
```

Upload this archive to Cybercampus before the deadline.