CS-499: Computer Science Capstone

5-2 Milestone Four: Enhancement Three: Databases

Emily Black

November 28, 2025

INTRODUCTION & ARTIFACT SELECTION

For the databases category, I selected my Course Planner application, originally developed in CS-300: Data Structures and Algorithms in October 2024. In its original version, the program loaded course information only from a CSV file and stored data in a hash map for quick in-memory lookups. While this met the CS-300 course specifications, it did not demonstrate the use of relational databases, parameterized SQL queries, or scalable data retrieval. Because I am pursuing a career in data analytics and business intelligence, where structured data and relational querying are essential, this artifact was an ideal candidate for expansion.

OVERVIEW OF THE ENHANCEMENT

For Milestone Four, I enhanced the Course Planner to load and store course data using a MySQL relational database, which required the introduction of a dedicated `DatabaseConfig` module for connection management using MySQL Connector/C++. Within `CourseRepository`, I implemented two new functions, `loadFromDatabase()` and `saveCourseToDatabase(const Course&)`, which allow the system to retrieve and write course records using secure, parameterized SQL statements. The original CSV functionality remains available so that the user may choose either data source at runtime. This enhancement represents a substantial shift toward a more scalable and industry-relevant data architecture.

TECHNICAL IMPLEMENTATION DETAILS

The enhanced system uses a relational courses table that includes `course_id`, `title`, and `prereqs`, with `course_id` designated as the primary key. The new `saveCourseToDatabase()` function uses a parameterized SQL statement that performs an insert-or-update operation:

```
INSERT INTO courses (course_id, title, prereqs)
VALUES (?, ?, ?)
ON DUPLICATE KEY UPDATE title = VALUES(title), prereqs = VALUES(prereqs);
```

Parameterized queries help prevent SQL injection and follow secure coding guidance recommended by OWASP (OWASP Foundation, 2021). The `loadFromDatabase()` function performs a sorted database query using `ORDER BY course_id`, then reconstructs `Course` objects by splitting comma-separated prerequisites, which preserves the logical behavior of the program while shifting storage and retrieval to a relational database.

The design retains the multi-layer structure created in Milestone Two. The UI continues to route user commands, the service layer handles formatting and domain logic, and the repository layer manages all data access. This alignment with multi-tier architecture supports maintainability and scalability, which are important in real analytical systems where data sources often evolve (Fowler, 2022).

ALIGNMENT WITH DATABASE CATEGORY COMPETENCIES

This enhancement satisfies the database-aligned outcomes by demonstrating practical skills in relational schema design, secure parameter binding, database connectivity, and structured data querying. The implementation reflects best practices for integrating application-level logic with relational data stores, which is central to database-driven systems used in analytics, reporting, and business intelligence environments (Coronel & Morris, 2020).

The ability to select, retrieve, transform, and persist structured data directly supports the database competency expectations within CS-499. The system demonstrates the full cycle of ETL-like operations on a small scale: extraction from SQL, transformation into object form, and loading back into persistent storage.

RELEVANCE TO PROFESSIONAL GOALS

Because Data Analysts and Business Intelligence Analysts frequently work with relational databases in environments such as MySQL, SQL Server, Snowflake, and PostgreSQL, this enhancement directly strengthens the skills required in those fields. The work in this milestone demonstrates proficiency in SQL, secure query execution, relational modeling, and structured data retrieval. These are fundamental capabilities for analysts responsible for generating insights, supporting decision-making, and working with production datasets in enterprise contexts. Adding a database component to the Course Planner not only improves the artifact but also gives me hands-on experience with skills I will draw upon in my career.

HOW THIS ENHANCEMENT BUILDS TOWARD THE FINAL PROJECT

This enhancement meaningfully contributes to the final project by transforming the Course Planner into a more complete, professional-quality artifact suitable for inclusion in my capstone portfolio. The final project requires demonstrating mastery across software design, algorithms, data structures, and databases. By adding MySQL support, I now have an artifact that showcases all three categories. The database integration complements the earlier structural refinements from Milestone Two and aligns with my long-term goal of presenting a polished, data-focused project that reflects industry practices. This enhancement also positions the Course Planner as a stronger portfolio example because it incorporates both data engineering fundamentals and analytical data retrieval, which are central to business intelligence and data analyst roles.

REFERENCES

Coronel, C., & Morris, S. (2020). Database systems: Design, implementation, and management (13th ed.). Cengage Learning.

Fowler, M. (2022). Patterns of enterprise application architecture. Addison-Wesley Professional.

OWASP Foundation. (2021). OWASP code review guide: Secure coding and review practices. https://owasp.org/www-project-code-review-guide/

Oracle. (2024). MySQL Connector/C++ developer guide. https://dev.mysql.com/doc/