Tomas Santos Yciano

Professor Benjamin

CUS 510: Database Management Systems

21 December 2024

Final Project Write-Up

1. Introduction

This mini project aims to design and implement a well-structured database schema for a university information system (UIS). This project mimics the structure of student, faculty/instructor, and academic records at St. John's University. This measure makes this project a mini-world experiment of the existing St. John's UIS. The database schema designed in this project ensures data integrity by applying fundamental concepts of database management – namely constraints, normalization, and relational design. It eliminates redundancy to enhance the overall effectiveness of the database. This project offers a practical application of the concepts learned in the graduate-level Database Management Systems course, helping to reinforce the understanding of how databases are created, organized, and maintained in real-world settings. This report outlines the specific design decisions, technology stack justification, relationships between entities, and the implementation of critical constraints and features created in the database schema to best support a university information system like the one for students and faculty at St. John's University.

2. Technology Stack Justification

2.1 DBMS: MySQL

The St. John's UIS implementation leverages a carefully curated technology stack to ensure reliability, maintainability, and scalability. At its core, MySQL is the primary database management system (DBMS), chosen for its robust support of complex relationships, constraint management capabilities, and user-friendliness. In addition, its built-in functionality for triggers and stored procedures enables sophisticated data validation and automation of routine procedures.

2.2 Back End: Python Flask & SQLAlchemy

The back end of the St. John's UIS is built using Flask, a lightweight yet powerful Python web framework. This choice is driven by Flask's excellent balance of simplicity and flexibility, making it ideal for building the UIS. In addition, Flask has been implemented with SQLAlchemy, an open-source database tool that provides a SQL toolkit and an Object-Relational Mapper (ORM) for database interactions. Using Pythonic code and Pythonic objects enables efficient and flexible database access. This results in clean, maintainable code while ensuring optimal database performance. Finally, the framework's modular design allows for easy expansion of functionality and integration of additional features at St. John's needs evolve.

2.3 Front End: HTML/CSS

The front end of the St. John's UIS employs HTML and CSS to deliver a clean and responsive design. This straightforward approach prioritizes accessibility and ease of use while maintaining the university's brand identity through consistent styling and color schemes. The

interface is designed to be intuitive for both students and faculty, with a transparent navigation scheme and immediate feedback on user actions.

2.4 Web Deployment: Railway

The entire technology stack is deployed on the web using Railway, a modern cloud platform that simplifies the deployment and scaling process for applications of all types. This platform provides a simple environment variable configuration and seamless database integration. In addition, the use of environment variables enhances security by keeping sensitive information, such as database credentials, separate from the application code.

2.5 Conclusion

The above technology stack creates a cohesive system that balances modern development practices with the specific needs of St. John's UIS. The sturdy combination of MySQL's reliability, Flask's flexibility, the straightforward and intuitive styling of HTML/CSS, and Railway's robust deployment and scaling capabilities provide a solid foundation for managing the university's academic data while ensuring scalability for future growth.

3. Database Schema Design – Table Structure & Normalization Implementation

The St. John's UIS employs a meticulously structured database schema built upon five interconnected tables, each serving a specific purpose while maintaining data integrity, efficiency, maintainability, and consistency within the university's operational context. These tables are Student, Course, Instructor, Section, and Enrollment. The design process follows a

systematic approach to normalization, adhering to the First, Second, and Third Normal Forms (1NF, 2NF, 3NF) while considering the practical requirements of the UIS. In addition, the schema implements strategic cascading options and rules between related tables, allowing for automated data maintenance. The relationships between the tables are carefully thought out and structured through numerous primary and foreign key constraints, creating a sturdy network of interconnected data that mimics the relationships between students, courses, instructors, and enrollments at St. John's University.

3.1 The Student Table

Firstly, the Student table in the database schema serves as a cornerstone of the UIS, implementing robust data integrity through carefully designed constraints and relationships. It captures all the essential information St. John's requires for each student. It includes fourteen columns: Student X number, Stormcard ID, Library ID, First Name, Last Name, Date of Birth, College, Major, GPA, Gender, Email, Phone Number, Admission Year, and Graduation Year. These fourteen were chosen because they are critical data points for managing each student's interactions on campus. The table's primary key, Student X-number, follows a strict validation format, ensuring consistency in student identification across the system. This design choice supports normalization principles and practical university operations, as the X-number is a unique identifier throughout a student's academic journey at the university.

The student table demonstrates comprehensive normalization implementation across the three normal forms. In accordance with the First Normal Form (1NF), all attributes are atomic, meaning each column has a unique name, and each attribute has a singular value. For instance, student names are split into first-name and last-name fields, and contact information is divided

into email and phone numbers. The email field enforces the St. John's specific format for student emails through check constraints, ensuring standardization across all student records.

Further, the table complies with the Second Normal Form (2NF) by ensuring all attributes depend entirely on the primary key of the student's X numbers. For example, each attribute, like Stormcard ID and year of graduation, directly relates to the student identified by their X-number. The implementation also includes unique constraints on the two attributes, preventing duplicate assignments of these critical identifiers. The table also provides check constraints for fields like GPA (between 0.00 and 4.00) and gender, ensuring data integrity without creating unnecessary dependencies.

Third Normal Form (3NF) compliance is maintained by eliminating transitive dependencies. In other words, all non-key attributes must be independent of each other. Each column is directly related to each student's X number as the primary key and not to any other columns in the table. For example, while the college and major fields are conceptually related, they are maintained as independent attributes directly dependent on the Student X-number. This design prevents anomalies if major information were dependent on the college an individual student is a part of.

The table's design incorporates practical considerations for university operations. Date fields in the Student table include validation rules ensuring logical chronological order and preventing inappropriate future dates. In addition, the phone number field enforces a standardized format, maintaining data quality.

The table's structure includes several key integrity features for indexing. An index is automatically created for Student X-number because it serves as the primary key. In addition, unique constraints on Stormcard ID, Library ID, and email create unique indexes for these fields. This enforces data uniqueness and optimizes queries involving these frequently accessed

identifiers.

Finally, the Student table maintains critical relationships within the UIS, primarily connecting to other entities through the Enrollment table. This is a one-to-many connection between the tables, where a single student can simultaneously enroll in multiple course sections. For example, a student can be enrolled in sections of both Software Engineering (CUS 1166) and Database Management Systems (CUS 510) during the same semester. This relationship is established through a foreign key constraint in the Enrollment table referencing the Student X-number, with cascade rules for updates and deletions. For instance, when a student withdraws from the university, and their record is deleted permanently, the cascade rules ensure that all associated enrollment records are automatically removed. Similarly, if a student's X-number needs to be updated, the change propagates to all their enrollment records. The Student table is not directly connected to the Course or Section tables. Instead, these relationships are managed through the Enrollment table, allowing for a many-to-many relationship between students and course sections. Finally, there is no direct relationship between the Student and Instructor tables, as they only interact through course sections that students are enrolled in and instructors teach.

3.2 Course Table

Secondly, the Course table in the database schema is the central repository for all academic offerings at St. John's University during any given term, managing essential course information with well-crafted relationships and constraints. The table includes seven column definitions: CRN Number (Course Reference Number), course title, course description, instructor X number, credits, department, and course number. These columns represent the primary data points defining the courses and sections students can enroll in. Each column plays a specific role: CRN (Course Reference Number) serves as the primary key as the unique identifier for each course;

the course title and course description sections provide an overview of each course; the instructor X number serves as a foreign key, linking the course to the relevant faculty member; credits outline the credit hours of each course, and department defines what department each faculty member is a part of. These attributes were selected based on critical academic and operational requirements and data management needs.

The table demonstrates compliance with the three normal forms. 1NF compliance is achieved through atomic values, with each field containing singular, indivisible values. For instance, course identifiers are split into distinct CRN number and course number fields, allowing for internal system reference and human-readable course codes. 2NF is maintained as all non-key attributes entirely depend on the CRN number as the primary key, with no partial dependencies. Finally, 3NF is accomplished by ensuring no transitive dependencies exist among non-key attributes. Each field depends directly on the primary key as opposed to other columns.

Data integrity is enforced through several crucial constraints. The CRN number is known to start with different values depending on the term. For the Spring term, the CRN number begins with ‘7’. In the Fall term, the CRN number starts with ‘1’. As a result, the CRN number is restricted to values between 10000 and 19999 for the Fall term and 70000 and 79999 for the Spring term. This ensures consistent course identification and follows real-life university operations. The credits field is constrained to values between 1 and 5, reflecting the real-life academic credit hour assignments for each course at St. John’s. The Instructor X-number field maintains referential integrity through a foreign key constraint referencing the Instructor table, with specific cascade rules that prevent instructor deletion while actively teaching a course but allow instructor information to propagate.

The Course table implements strategic indexing to optimize query performance and maintain data integrity across the table and the entirety of the UIS. An index is automatically created for the CRN number as the primary key, ensuring efficient retrieval of course records by their unique identifier. Additionally, the table includes several secondary indexes to enhance query performance for everyday operations. This includes indexes for the course title, Instructor X-number, department, and course number fields. These indexes facilitate quick course searches, efficient lookups of courses by an instructor, rapid filtering and grouping of courses by department, and fast retrieval of courses using their human-readable course numbers. These indexes were selected based on common query patterns in university operations, and they benefit students and faculty alike during peak registration periods toward the end of each term when the system experiences heavy read-and-write operations.

Finally, the Course table maintains critical relationships within the schema. Its primary relationship is with the Instructor table, where it establishes a many-to-one connection. This means that each course has exactly one instructor while enabling instructors to teach multiple courses simultaneously. Additionally, it maintains a one-to-many relationship with the Section table, where each course can have various sections at a time. This relationship implements cascade deletion, ensuring that all sections and their enrollment records are automatically removed when a course is discontinued to maintain consistency within the database. For example, when a course like Software Engineering (CUS 1166) is created, it receives a unique CRN, is assigned to a qualified instructor, and can be divided into multiple sections if needed. If the instructor's information needs updating, the cascade rules ensure that changes are reflected in all course records. However, if an attempt is made to delete an instructor's record while they are

still actively teaching, the restrict constraint will prevent this operation. Ultimately, this maintains data integrity and reflects real-world academic operations.

3.3 Instructor Table

Thirdly, the Instructor table is a comprehensive gateway for all faculty information within the UIS, implementing robust data management with various constraints and relationships. It contains twelve essential columns: Instructor X-number, Stormcard ID, First Name, Last Name, Date of Birth, College, Department, Email, Gender, Phone Number, Date of Hire, and Salary. These attributes were selected as they represent the critical data points for managing faculty information and academic responsibilities. Additionally, Instructors are similar to students in that they get X-numbers and Stormcard IDs and are part of a specific college. The differences here are that instructors are part of departments instead of majors, their emails follow a different naming convention than students, there is a hire date instead of an enrollment date, and instructors make a salary while actively teaching at the university. The Instructor X-number serves as the table's primary key, with a strict format validation like that of students, ensuring consistency in faculty identification across the system.

The Instructor table demonstrates a thorough implementation of the three normal forms. In compliance with the 1NF, all attributes maintain atomic values, with each column having a unique name and a singular value. For example, instructor names are separated into first and last name fields, and contact information is divided between email and phone number columns. The email field enforces the St. John's specific format for faculty through check constraints, ensuring standardization across all instructor records.

The table achieves 2NF by ensuring all attributes depend wholly on the primary key of the instructor's X-number. Each attribute directly relates to the individual instructor identified by their X-number, just like it is for students. This implementation includes unique constraints on Stormcard ID, email, and phone number, which prevents duplicate assignments of these identifiers. The table also provides check constraints for salary (between $0 and $500,000) and gender, ensuring data integrity.

3NF is maintained by eliminating transitive dependencies among non-key attributes. Each column is directly related to the instructor's X-number as the primary key and not other columns in the table. For example, while the college and department fields are related, they are independent attributes dependent on the instructor's X-number. This design prevents anomalies that could arise if department information were dependent on the college each professor is a part of.

Next, the table's design incorporates stringent constraints to maintain efficient university operations. The Date columns include validation rules ensuring logical chronological order and preventing future dates. For instance, an instructor's date of birth cannot be in the future and also cannot be later than their date of hire. An instructor's date of hire cannot be in the future either. The salary column maintains accuracy with decimal precision, and the phone number field enforces a standardized format like in the other tables, ensuring consistency throughout the schema.

Furthermore, the table includes several key integrity features for indexing. An index is automatically created for the instructor X-number as it serves as the primary key. Additional indexes include the instructor's name to optimize faculty searches based on first and last name, department for efficient department-based queries, and college for college-based filtering. These

indexes enhance query performance for everyday administrative operations while maintaining the integrity of the schema's structure. This would be helpful during active registration when students are actively searching for the instructors to whom their course sections will be taught.

Lastly, the Instructor table maintains significant relationships with the other tables in the database schema, mainly through its connections to the Course and Section tables. These relationships are established through the instructor X-number, which serves as a foreign key in both tables, referencing back to the Instructor table. The relationship with the Course table is one-to-many, where one instructor can simultaneously be assigned as the primary instructor for multiple courses. However, each instructor must have precisely one instructor. Similarly, the relationship with the Section table is also one-to-many, where one instructor can teach multiple course sections simultaneously, while precisely one instructor must teach each section. Both relationships establish cascading options and rules, such as restrict rules for deletion and cascade rules for updates. These rules prevent the deletion of instructor information while they are actively teaching at the university and ensures the propagation of new instructor information if and when changes are made. For example, if an instructor changes departments or changes colleges, the change would automatically update instructor records. This ensures that the instructor table effectively manages faculty data while maintaining the integrity and accuracy of teaching assignments and responsibilities throughout the university's academic operations.

3.4 Section Table

Fourth, the Section table in the database schema functions as a crucial component in the UIS, managing the specific instances of courses offered each term. It includes eleven column definitions: Section ID, CRN Number, Instructor X-number, start date, term, building name, room number, max capacity, current enrollment, and online status. These attributes were selected

to capture all the necessary information for managing course offerings, classroom assignments, and enrollment tracking. Unlike other tables, this table employs a composite primary key consisting of Section ID and CRN number, ensuring unique identification of each course section while maintaining its relationship with the parent course offering. The table demonstrates a thorough compliance with all three normal forms.

By the 1NF, all attributes maintain atomic values, with each column containing individual data points. For instance, the location information for each section is divided into separate building name, room number, and online status fields. Temporal data is split into start date and end date columns.

2NF compliance is achieved through all non-key attributes entirely dependent on the composite primary key of Section ID and CRN number. Each attribute depends on the specific section instance identified by this composite key. The implementation includes appropriate constraints, such as maximum capacity validation and building name verification, ensuring data consistency and integrity while properly maintaining these functional dependencies.

3NF is maintained by eliminating transitive dependencies among all non-key attributes. Each column depends directly on the composite primary key and not other columns. For example, while building and room assignments are directly related, they are independent attributes dependent on the composite key. This prevents update anomalies if room assignments depend on building assignments and vice versa.

This table's design incorporates various constraints relevant to the university's operating context for practical purposes. Building names are restricted to valid locations on the flagship Queens Campus, room numbers must follow a standardized format, and the online status flag

determines whether a class is in-person or asynchronous and is used to determine whether physical campus location information is required. Date constraints ensure logical chronological order between start and end dates, while enrollment constraints ensure that current enrollment in a course cannot exceed the established maximum capacity. These constraints work together to support the university's academic operations.

In addition, the Section table includes critical indexing for query optimization. Indexes are automatically created for the composite primary key tandem of Section ID and CRN number. Secondary indexes include instructor X-number and term to enhance query performance for everyday operations like looking up all sections for a specific course offering in a particular term or finding an instructor's teaching schedule. Combining these indexes creates a sturdy foundation for effectively managing course sections.

Finally, the Section table maintains critical relationships with other tables in the database schema through its connections. It establishes many-to-one relationships with the Course and Instructor tables, implemented through foreign key constraints on CRN number and instructor X-number, respectively. This means multiple sections can be created for a single course, and numerous sections can be assigned to one instructor. For example, CUS 510 (Database Management Systems) can have various sections taught by the same instructor or distributed multiple sections among different instructors. Numerous cascading rules are established between these two relationships. In the Course table, there are cascade rules for updates and deletions; in the Instructor table, there is a restrict option for instructor deletions. This ensures that a section is removed when a course offering is deleted and the prevention of instructor removal with active teaching assignments. Additionally, the Section table shares a one-to-many relationship with the

Enrollment table, where each section can have multiple student enrollments, managed through cascade rules that automatically remove enrollment records when a section is deleted.

3.5 Enrollment Table

Finally, the Enrollment table in the database schema contains critical details concerning a student's enrollment at the university. The table includes five column definitions: Student X-number, Section ID, CRN Number, Grade, and Enrollment date. Each column does the following: like the Section table, it employs a composite primary key, consisting of student X-number, Section ID, and CRN number, along with additional attributes: grade and enrollment date. The composite key structure ensures entity integrity while facilitating the table's role as a junction between students and course sections. These attributes were chosen to capture the essential information needed for student registration, enrollment in course sections, and academic record keeping. This table exemplifies the thorough implementation of database normalization principles across the three normal forms.

Under 1NF, all attributes are atomic, with each column containing singular data points. For example, the grade field stores a single letter grade value, while the enrollment date captures the precise timestamp of a student's registration into a course.

2NF compliance is achieved through all non-key attributes utterly dependent on the composite primary key trio of Student X-number, Section ID, and CRN number. Each attribute depends on the specific student-section enrollment instance this composite key setup identifies. This implementation includes appropriate constraints to ensure data integrity while maintaining the functional dependencies on the composite key, including grades and enrollment dates.

3NF is maintained because all non-key attributes are independent of each other and directly dependent on the composite key. For example, a student's grade in one section is wholly independent of their enrollment date or grades in different sections.

The table's design incorporates essential constraints applicable within the university's operational context for practical purposes. For instance, the grade field is restricted to valid letter grades handed out by the university (A, A-, B+, etc.). It is worth noting that St. John's does not give out 'A+' letter grades to any student, regardless of course load. Enrollment dates must be valid timestamps. Additional rules are enforced through the application layer of this project, such as preventing enrollment in sections that have reached their maximum capacity. These constraints work to ensure proper support of the university's academic operations.

Next, this table's structure includes strategic indexing for query optimization, like every other table in this schema. Indexes of the composite key trio are created automatically. Meanwhile, secondary indexes include student X-number and the section ID/CRN number combination to enhance query performance for everyday operations such as retrieving a student's course schedule during a particular term or generating class rosters for each section.

Finally, the Enrollment table maintains key relationships with other tables within the schema. This table shares many-to-one relationships with both the Student and Section tables. The relationship with the Student table, implemented through the student X-number foreign key constraint, allows each student to enroll in multiple course sections while ensuring each enrollment record is associated with precisely one student. Similarly, the relationship with the Section table, implemented through the composite foreign key of Section ID and CRN number, enables each section to have multiple student enrollments while ensuring each enrollment record corresponds to precisely one section. These relationships implement cascade rules for updates

and deletions, ensuring that all associated enrollment records are automatically deleted to maintain database consistency if a student record is removed or a section is canceled.

4. Future Improvements

The St. John's UIS's current implementation, while sturdy and functional, has several potential enhancements that could further improve its functionality, efficiency, and potential for future growth.

From a database schema perspective, implementing several new tables, such as Term, would better organize academic periods and reduce reliance on the existing CRN patterns for term identification of course sections. The Course table could also benefit from a prerequisite system (like the real-world St. John's already has). At the same time, the Section table could incorporate waitlist management and cross-listed course support (like the real-world St. John's also has). Adding a grade audit for potential grade changes in the Enrollment table would vastly improve the student experience and enhance academic record-keeping. Finally, enhancing the authentication and authorization system through dedicated user management tables would strengthen security and enable role-based access control. A Users table with encrypted credentials and a Roles table for defining access levels (student, faculty, administrator) would allow for permissions for each role and personalized application views. This would ensure that students can only view and modify their enrollment records, faculty can manage their course sections, and administrators have appropriate oversight capabilities while maintaining data privacy and security best practices. The database itself could be migrated from MySQL to a service like AWS RDS or Google Cloud SQL, allowing for improved monitoring and performance.

From a technical standpoint, the application currently operates with a single administrative view that provides unrestricted access to all database operations. While this implementation demonstrates the full functionality of the database schema, it presents security concerns in a real-world environment. Future improvements include implementing a strong authentication system and role-based access control to create distinct views and permissions for students, faculty, and administrators. Finally, the current technological stack could be modernized by implementing a React.js or Next.js framework for dynamic component rendering and improved state management. Styling could be enhanced by applying the Tailwind CSS framework or Material-UI components, providing a more enjoyable and responsive user interface. Migrating from Railway as a cloud platform to a more robust platform such as AWS, Google Cloud, or Azure would provide better scalability, reliability, and cost-effectiveness.

These improvements would maintain the current schema's strong normalization principles across the three normal forms while enhancing performance, scalability, and user experience for everyone. These changes address the specific needs of the university's academic operations.

5. Conclusion

In conclusion, the St. John's University Mini-World Database Schema representing the St. John's University Information System (UIS) demonstrates a robust implementation of database management principles through its carefully curated design. The interconnected Student, Course, Instructor, Section, and Enrollment tables form a cohesive system that effectively models the real-life academic operations of the university. These tables facilitate seamless data management, support university initiatives, and provide a sturdy foundation for

managing student and faculty information across various colleges, departments, and majors. While this project is currently deployed as a prototype on Railway, the system's present architecture opens the door for future enhancements, including role-based access control, improved cloud infrastructure, modern frontend frameworks, and more. This mini-project demonstrates the practical application and mastery of the concepts learned in the graduate-level Database Management Systems course and provides a sturdy foundation for educational informational systems.