Designing a Comprehensive Database System for Enhanced Library Management: A Case Study of Robarts Library

INF1343 - Systems Analysis and Process Innovation
Course Project - Phase 2

March 20, 2024

Han Zheng (Student No. 1010737874)
Anna Li (Student No.1009711705)
Ang Zhao (Student No. 1010171216)
Sujaan Rajakumar (Student No. 1010745856)

University of Toronto
Faculty of Information - M.I. Program
Instructor: Professor Nada Almasri

Number of Words: 4872

Table of Contents

Executive Summary

This database will serve the Robarts Library at the University of Toronto. It will be used to support the library's core business processes and manage data for administrative purposes. This includes organizing the book catalog, tracking book check-ins and check-outs, managing user accounts, collecting fines, updating the library's collections, and collecting analytics on both books and people. The database will provide information to users looking for books and account information such as loans and overdue fines. It will also support librarians' administrative tasks and decision-making by giving them a complete record of the books and people associated with the library.

The library has several core business processes. Firstly, librarians need to catalog and organize existing books so that individual books are searchable and they know which books are in their collections. They would need to input book identification data such as title, author, genre, and ISBN. The database should be able to output a complete and organized catalog of the library inventory. Next, Robarts librarians need to oversee the check-in and check-out processes. This includes inputting which books have left and re-entered the library, how long each book has been on loan, as well as tracking which patrons the books are associated with. The collection of overdue book fines is also a core process. Patron information, such as student ID, loans and holds, and any paid or outstanding fines, should be stored in the database. The last core business process is the updating of the catalog. Librarians need to update information on the condition and age of books over time to determine when a volume should be sold, donated, discarded, and/or replaced. They also need to store information on book acquisition requests to determine which books patrons would like to see in the library. The database should be able to output statistics on book traffic and book condition to help library administrators make decisions regarding library services and inventory.

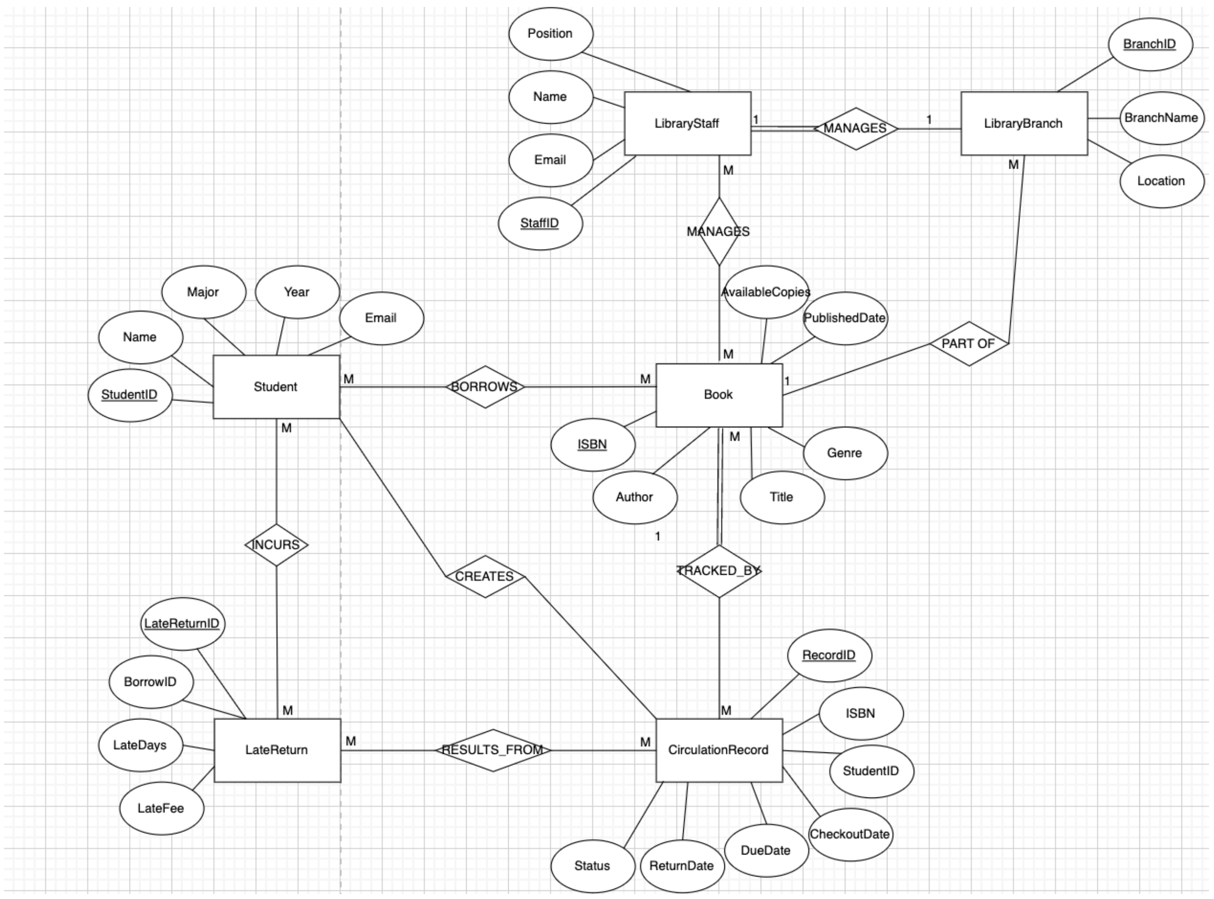
The Relational Model

ER diagram

Compared to phase 1, we updated the ER diagram for our library management system which defines the structure of the database and the relationships between the different entities that are part of the system. Entities such as "Student," "Book," "LibraryBranch," "LibraryStaff," "CirculationRecord," and "LateReturn" are connected through various relationships that indicate how they interact with each other.

For example, the "Student" entity, which includes attributes like "StudentID," "Name," "Email," "Major," and "Year," has a many-to-many relationship with the "Book" entity, meaning that a student can borrow multiple books and a book can be borrowed by multiple students. Similarly, the "LibraryStaff" entity, which holds "StaffID," "Name," "Email," and "Position," is in charge of managing one or more "LibraryBranch" entities, as indicated by the "MANAGES" relationship. Besides, the "Book" entity contains crucial information such as "ISBN," "Title," "Author," "Genre," "PublishedDate," and "AvailableCopies," and it is part of a "LibraryBranch" through the "PART OF" relationship, suggesting that different branches hold different copies of books. Moreover, the diagram incorporates a "LateReturn" entity, which is related to the "CirculationRecord" entity and holds information on overdue books, such as "LateDays" and "LateFee."

Overall, this ER diagram is a comprehensive representation of a Robarts Library's operational data model, reflecting the various processes and interactions like book lending, staff management, and branch operations within the library system.



Relational diagram

Mapping the conceptual model into a relational model involves a careful process where entities are transformed into tables, and relationships are delineated through primary and foreign keys. Here's a detailed look at this process:

Specifying Primary Keys

- **Primary keys** are unique identifiers for each table, derived from the attributes of the conceptual entities.
  - **LibraryBranch**: Uses **BranchID** as the primary key.
  - **LibraryStaff**: Uses **StaffID** as the primary key.
  - **Book**: Uses **ISBN** as the primary key.
  - **Student**: Uses **StudentID** as the primary key.
  - **CirculationRecord**: Uses **RecordID** as the primary key.
  - **LateReturn**: Uses **LateReturnID** as the primary key.

Specifying Foreign Keys

- **Foreign keys** link tables together, establishing relational pathways.
  - **LibraryStaff.BranchID** references **LibraryBranch.BranchID** to connect staff with branches.
  - **Book.BranchID** references **LibraryBranch.BranchID**, associating books with branches.

- **Borrows.StudentID** references **Student.StudentID**, and **Borrows.ISBN** references **Book.ISBN**, linking students and books.
- **CirculationRecord.ISBN** references **Book.ISBN**, and **CirculationRecord.StudentID** references **Student.StudentID**, connecting circulation records with books and students.
- **LateReturn.RecordID** references **CirculationRecord.RecordID**, relating late returns with circulation records.

Establishing Relationships

- The relational model's relationships depict the interactions between entities.
  - One-to-One: Enforced by unique constraints, such as a **LibraryStaff** member assigned to one **LibraryBranch**.
  - One-to-Many: Illustrated by foreign keys in the 'many' side table pointing to the 'one' side, like multiple **Books** linked to a single **LibraryBranch**.
  - Many-to-Many: Facilitated using junction tables, for instance:
    - **Manages** bridges **LibraryStaff** and **Book**, with **StaffID** and **ISBN** foreign keys, allowing multiple staff to manage multiple books.
    - **Borrows** connects **Student** and **Book**, enabling many students to borrow many books.
    - **Incurs** associates **Student** and **LateReturn**, indicating students can have numerous late returns.

Normalization

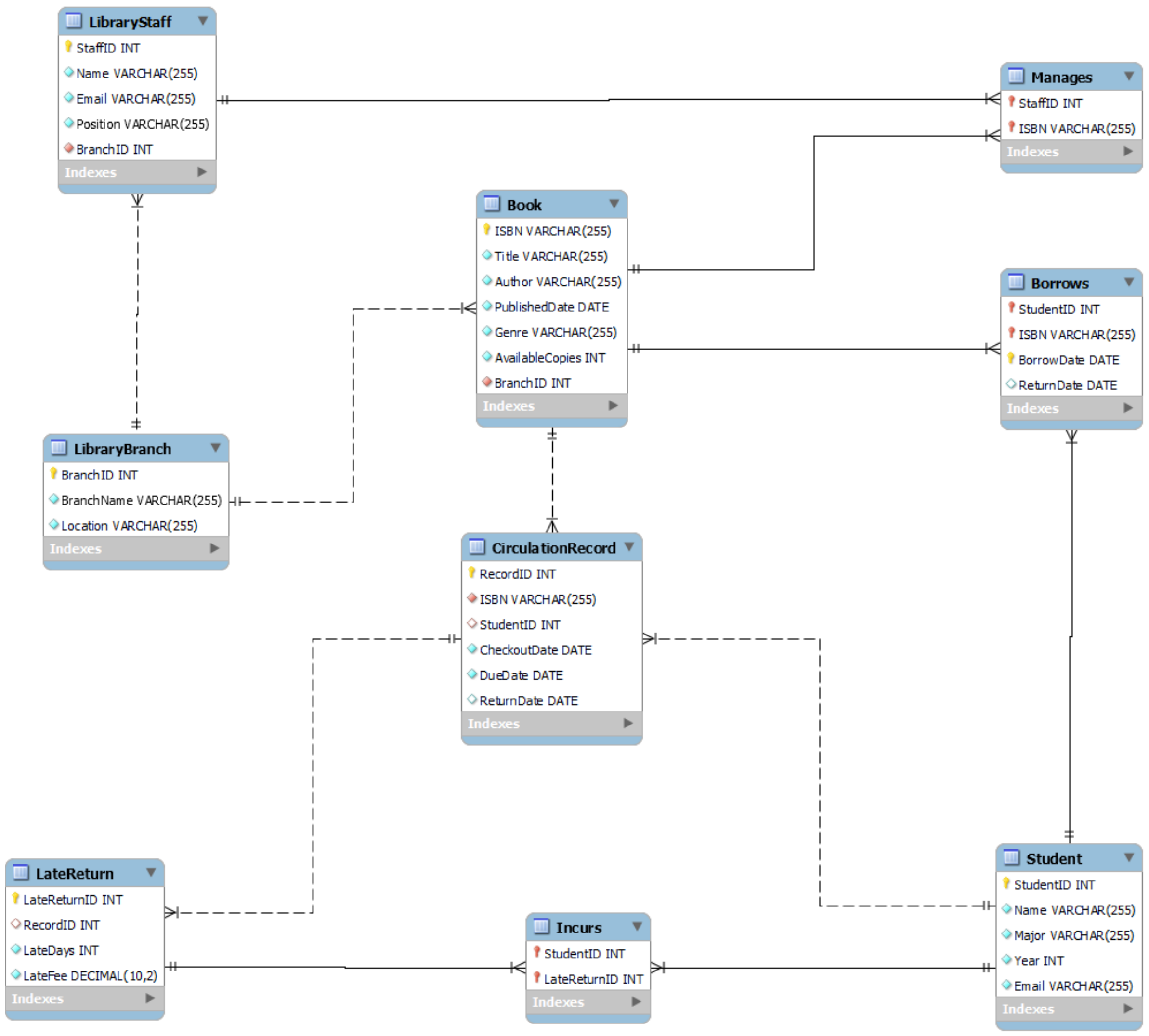
- Ensures data is structured efficiently to reduce redundancy and improve integrity.
  - Tables are normalized typically to Third Normal Form (3NF), meaning:
    - All attributes depend on the primary key (**1NF**).
    - All attributes are independent of each other, with no subsets of data relying on other subsets (**2NF**).
    - All attributes are directly dependent on the primary key, not on other non-primary attributes (**3NF**).

Constraints

- Constraints enforce data rules and business logic within the database.
  - **CHECK** constraints, like ensuring **ISBN** is 13 digits and **LateFee** is non-negative.
  - **NOT NULL** constraints ensure mandatory fields like **BranchID** in **LibraryStaff** and **ISBN** in **CirculationRecord** are always populated.
  - Unique constraints prevent duplicate entries in certain fields like **BranchID** in **LibraryStaff** for one-to-one relationships.

Incorporating normalization and constraints, the relational model not only structurally represents the conceptual model but also enforces the business rules and data quality through these database mechanisms. This ensures the integrity, accuracy, and reliability of the data in the database system.

Below is our visualized diagram generated by MYSQL workbench, showing our relational model:



- SQL

```
-- Create table for Library Branches
CREATE TABLE LibraryBranch (
    BranchID INT PRIMARY KEY,
    BranchName VARCHAR(255) NOT NULL,
    Location VARCHAR(255) NOT NULL
);

-- Create table for Library Staff
-- Each staff member is associated with exactly one branch (BranchID is NOT NULL)
CREATE TABLE LibraryStaff (
    StaffID INT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Email VARCHAR(255) NOT NULL,
    Position VARCHAR(255) NOT NULL,
    BranchID INT NOT NULL,
    FOREIGN KEY (BranchID) REFERENCES LibraryBranch(BranchID)
);

-- Create table for Books
-- Each book must be associated with a branch (BranchID is NOT NULL)
-- CHECK constraint ensures ISBN is 13 digits long
CREATE TABLE Book (
    ISBN VARCHAR(255) PRIMARY KEY CHECK (LENGTH(ISBN) = 13),
    Title VARCHAR(255) NOT NULL,
    Author VARCHAR(255) NOT NULL,
    PublishedDate DATE NOT NULL,
    Genre VARCHAR(255) NOT NULL,
    AvailableCopies INT NOT NULL,
    BranchID INT NOT NULL,
    FOREIGN KEY (BranchID) REFERENCES LibraryBranch(BranchID)
);

-- Create table for Students
CREATE TABLE Student (
    StudentID INT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Major VARCHAR(255) NOT NULL,
    Year INT NOT NULL,
    Email VARCHAR(255) NOT NULL
);

-- Create a junction table for many-to-many relationship between Library Staff and Books
CREATE TABLE Manages (
    StaffID INT,
    ISBN VARCHAR(255),
    PRIMARY KEY (StaffID, ISBN),
    FOREIGN KEY (StaffID) REFERENCES LibraryStaff(StaffID),
    FOREIGN KEY (ISBN) REFERENCES Book(ISBN)
);

-- Create a junction table for many-to-many relationship between Students and Books
-- BorrowDate included to track individual borrowing instances
CREATE TABLE Borrows (
    StudentID INT,
    ISBN VARCHAR(255),
    BorrowDate DATE NOT NULL,
    ReturnDate DATE,
    PRIMARY KEY (StudentID, ISBN, BorrowDate),
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID),
    FOREIGN KEY (ISBN) REFERENCES Book(ISBN)
);

-- Create table for Circulation Records
-- Links books and students, with NOT NULL constraints to ensure each book has a circulation record
CREATE TABLE CirculationRecord (
    RecordID INT PRIMARY KEY,
    ISBN VARCHAR(255) NOT NULL,
    StudentID INT,
    CheckoutDate DATE NOT NULL,
    DueDate DATE NOT NULL,
    ReturnDate DATE,
    FOREIGN KEY (ISBN) REFERENCES Book(ISBN),
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID)
);
```

```
-- Create table for tracking Late Returns
-- Linked to CirculationRecord via RecordID
-- CHECK constraint ensures LateFee is not negative
CREATE TABLE LateReturn (
    LateReturnID INT PRIMARY KEY,
    RecordID INT,
    LateDays INT NOT NULL,
    LateFee DECIMAL(10,2) NOT NULL CHECK (LateFee >= 0),
    FOREIGN KEY (RecordID) REFERENCES CirculationRecord(RecordID)
);

-- Create a junction table for many-to-many relationship between Students and Late Returns
CREATE TABLE Incurs (
    StudentID INT,
    LateReturnID INT,
    PRIMARY KEY (StudentID, LateReturnID),
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID),
    FOREIGN KEY (LateReturnID) REFERENCES LateReturn(LateReturnID)
);
```

Detailed Data Dictionary

Based on the relational model, we create a data dictionary for all attributes listed above.

Attribute Name	Domain	Meaning	Example Values
StaffID	Integer	Unique identifier for each staff member	4511237417, 4511239432
Name	Text/String	Full name of the staff member	"John Doe", "Jane Smith"
Position	Text/String	Job position of the staff member within the library	"Librarian", "Administrative Assistant"
Email	Text/String	Email address of the staff member	"j.doe@library.com", "j.smith@library.com"
BranchID	Integer	Identifier for the branch where the book copy is located	3497316053, 3497316112
BranchName	Text/String	Name of the library branch	"Main Branch", "Downtown Branch"
Location	Text/String	Name of the location	510 yonge sterrt
StudentID	Integer	Unique identifier for each student	1010773490, 1009457014
Name	Text/String	Full name of the student	"Alice Brown", "Bob Johnson"
Major	Text/String	Academic major of the	"Literature", "Computer

		student	Science"
Year	Integer	Academic year of the student	1, 2, 3, 4
Email	Text/String	Email address of the student	"a.brown@mail.utoronto.ca", "b.johnson@mail.utoronto.ca"
ISBN	Text/String	Unique identifier for each book	"978-0-545-01022-1", "978-3-16-148410-0"
Author	Text/String	Name of the book's author	"Margaret Atwood", "Gabriel Garcia Marquez"
Title	Text/String	Title of the book	"One Hundred Years of Solitude"
PublishedDate	Date	Date when the book was published	"1985-03-26", "1967-06-05"
Genre	Text/String	Genre or category of the book	"Fiction", "Science Fiction"
AvailableCopies	Integer	Number of available copies of the book	1, 2, 3, 4
LateReturnID	Integer	Unique identifier for each late return event	1000000987, 1000000988
BorrowID	Integer	Unique identifier for each borrowing	8361215660, 8361215661
LateDays	Integer	Number of late days	9, 0
LateFee	Decimal	Amount of late fees	10.5, 16.89
RecordID	Integer	Unique identifier for each book were borrowed each time	5632983410, 5632983411
CheckoutDate	Date	Date when the book was borrowed	"2024-03-03", "2024-01-27"
DueDate	Date	Date when the book should be returned	"2024-03-10", "2024-02-03"
ReturnDate	Date	Date when the book be returned exactly	"2024-03-19", "2024-01-31"
Status	Text/String	Status of the book currently	"Borrowed", "Available"

Test Cases

1. Testing Schema Constraints

```
use ver5;

# Testing domains with INSERT----------------------------------------------------------------------------

# insert tuple with wrong attribute value type (number instead of name characters)
insert into LibraryStaff
values (2347910277, 123, "jjones@utoronto.ca", "Head Librarian", 001);

# insert tuple in wrong format (ISBN longer than 13 characters)
insert into Book
values (123456789101112, "Philosophiæ Naturalis Principia Mathematica", "Isaac Newton",
1687-00-00,"Nonfiction", 3, 002);

# insert tuple with out-of-range value (negative fine)
insert into LateReturn
values (0982734123, 3759023894, 3, -6.15);

# Testing key constraints with INSERT---------------------------------------------------------------------

# insert tuples with non-unique primary key values (StaffID)
insert into LibraryStaff
values (2347910277, "John Jones", "jjones@utoronto.ca", "Director", 001);
insert into LibraryStaff
values (2347910277, "Hank Hoover", "hhoover@utoronto.ca", "Senior Preservation Associate", 002);

# insert tuples with non-unique primary key values (ISBN)
insert into Book
values (1234567891011, "Philosophiæ Naturalis Principia Mathematica", "Isaac Newton",
1687-00-00, "Science", 3, 002);
insert into Book
values (1234567891011, "Relativity : the Special and General Theory", "Albert Einstein",
1916-00-00, "Science", 5, 002);

# insert tuples with non-unique primary key values (BranchID)
insert into LibraryBranch
values (2940582749, "Natural Sciences", "2nd floor");
insert into LibraryBranch
values (2940582749, "Foreign Language", "4th floor");

# insert tuples with non-unique primary key values (StudentID)
insert into Student
values (0987654321, "Andrew Arlott", "Finance", 2026, "aarlott@utoronto.ca");
insert into Student
values (0987654321, "Beatrice Bennet", "Chemistry", 2025, "bbennet@utoronto.ca");

# Testing Entity Constraints------------------------------------------------------------------------------
# insert tuple with null primary key value (StaffID)
insert into LibraryStaff
values (NULL, "Harry Hayes", "hhayes@utoronto.ca", "Director", 001);

# insert tuple with null primary key value (ISBN)
insert into Book
values (NULL, "Radiology in War", "Marie Curie", 1687-00-00, "Biography", 1, 003);

# insert tuple with null primary key value (BranchID)
insert into LibraryBranch
values (NULL, "History", "6th floor");

# insert tuple with null primary key value (StudentID)
insert into Student
values (NULL, "Benjamin Buckley", "History", 2024, "bbuckley@utoronto.ca");

# Testing Referential Integrity Constraints-------------------------------------------------------------------

# insert tuple that references nonexistent foreign key (LibraryStaff references nonexistent
LibraryBranchID)
insert into LibraryStaff
values (2347910277, "John Jones", "jjones@utoronto.ca", "Director", 300);

# insert tuple that references nonexistent foreign key (CirculationRecord references nonexistent
ISBN)
insert into CirculationRecord
values (1029384756, 0000000000000, 1029743865, 2024-4-20, 2024-5-20, NULL);

# insert tuple that references nonexistent foreign key (LateReturn references nonexistent RecordID)
insert into LateReturn
values (0989878765, 0000000000, 4, 8.20);

# Testing minimum participation constraints------------------------------------------------------------------

# Violates requirement that "Every librarian must be in a branch"
insert into LibraryStaff
values (2347910277, "John Jones", "jjones@utoronto.ca", "Director", NULL);

# Violates requirement that "Every book must be in a branch"
insert into Book
values (123456789101112, "Philosophiæ Naturalis Principia Mathematica", "Isaac Newton",
1687-00-00,"Nonfiction", 3, NULL);

# Violates requirement that "All circulation records must be associated with a book"
insert into CirculationRecord
values (0987987687, NULL, 0987987687, 2024-4-20, 2024-5-20, NULL);
```

```
# Testing domains with UPDATE--------------------------------------------------------------------------------

# update tuple with wrong attribute value type (number instead of name characters)
update LibraryStaff
set Name = 123
where StaffID = 2347910277;

# update tuple to wrong format (ISBN longer than 13 characters)
update Book
set ISBN = 123456789101112
where Title =  "Philosophiæ Naturalis Principia Mathematica";

# insert tuple with out-of-range value (negative fine)
update LateReturn
set LateFee = -6.15
where LateReturnId = 0982734123;

# Testing key constraints with UPDATE----------------------------------------------------------------------

# update tuples with non-unique primary key values (StaffID)
update LibraryStaff
set StaffID = 2347910277;

# update tuples with non-unique primary key values (ISBN)
update Book
set ISBN = 1234567891011;

# update tuples with non-unique primary key values (BranchID)
update LibraryBranch
set BranchID = 2940582749;

# update tuples with non-unique primary key values (StudentID)
update Student
set StudentID = 0987654321;

# Testing Entity Constraints with UPDATE-------------------------------------------------------------------

# update tuples with null primary key value (StaffID)
update LibraryStaff
set StaffID = NULL;

# update tuples with null primary key value (ISBN)
update Book
set ISBN = NULL;

# update tuple with null primary key value (BranchID)
update LibraryBranch
```

```
set BranchID = NULL;

# update tuple with null primary key value (StudentID)
update Student
set StudentID = NULL;

# Testing Referential Integrity Constraints with UPDATE---------------------------------------------

# update tuple that references nonexistent foreign key (LibraryStaff references nonexistent
LibraryBranchID)
update LibraryStaff
set LibraryBranchID = 300
where Name = "John Jones";

# update tuple that references nonexistent foreign key (CirculationRecord references nonexistent
ISBN)
update CirculationRecord
set ISBN = 0000000000000
where RecordID = 1029384756;

# update tuple that references nonexistent foreign key (LateReturn references nonexistent RecordID)
update LateReturn
set RecordID = 0000000000
Where LateReturnID = 0989878765;

# Testing minimum participation constraints with UPDATE---------------------------------------------

# Violates requirement that "Every librarian must be in a branch"
update LibraryStaff
set BranchID = NULL
where StaffID = 2347910277;

# Violates requirement that "Every book must be in a branch"
update Book
set BranchID = NULL
where ISBN = 123456789101112;

# Violates requirement that "All circulation records must be associated with a book"
update CirculationRecord
Set ISBN = NULL
where RecordID = 0987987687;



# Testing Referential Integrity with DELETE------------------------------------------------------------------

# delete tuple that is referenced through foreign key (LibraryStaff references LibraryBranchID)
```

```
delete from LibraryBranch
where LibraryBranchID = 001;

# delete tuple that is referenced through foreign key (CirculationRecord references ISBN)
delete from Book
where ISBN = 1234567891011;

# update tuple that is references through foreign key (LateReturn references RecordID)
delete from CirculationRecord
Where RecordID = 1029384756;
```

2. Defining and Testing Complex Constraints

Based on our business processes, we determined that some more complex constraints naturally arise. For example, books cannot be allowed to be checked out by more than one person at a time because that is not allowed in reality, late fees and fines should not exceed a certain amount–such as $50–before a patron is barred from borrowing more books to encourage people to pay their dues, and students should have an upper limit to the number of books they can check out at once–such as 10 or 15–to allow others access to materials.

In order to check that the first constraint is being followed, one could use SELECT queries to look at the Borrows table and compare the number of associated students per ISBN to the number AvailableCopies in the table Book. If the number of students that have borrowed the book(s) is/are higher than the number of copies, then this is a data anomaly that should be investigated.

For the second constraint, it is necessary to use SELECT to look at multiple tuples within the LateReturn table. All fees needed to be added together for each separate student and if the total is, for example, over $50, then an administrator can check that they have not been associated with more books in the Borrows table.

Finally, for the last constraint, an administrator could query the Borrows table again and count the number of ISBNs from the Book table associated with each StudentID to check that no students are checking out egregious numbers of books.

Key Reports

1. Business Process Review

The Robarts Library is the largest library in the UofT Libraries system. Its main patrons are the students and staff of the university. As a library affiliated with a university, it is often used for research, especially for its particularly strong collections in the Arts, Humanities & Social Sciences. The goals of the library are to support students and professors by maintaining and archiving vast collections of information and making these collections available to inquiring readers.

Revisiting the major business processes defined in Phase 1 reveals that each process requires unique data to properly support decision-making, monitoring, and performance evaluation. For starters, classifying and organizing books necessitates the collection of detailed information such as title, author, genre, ISBN, and availability status. This information allows librarians to efficiently manage the library's inventory, track book usage patterns, and make educated decisions about collection development and resource allocation. Furthermore, data on the state and age of books over time is critical for deciding when volumes should be replaced or removed from circulation, which improves the library's collection management strategy.

Second, the check-in and check-out operations rely on financial data such as book movements, loan lengths, patron information, and overdue fees. This data helps to track book circulation trends, enforce lending policies, and collect overdue fines. Analyzing this data allows library administrators to identify areas for improvement in service delivery, undertake targeted interventions to prevent late returns, and optimize resource allocation.

Third, maintaining the catalog necessitates information on book acquisition requests, circulation statistics, and user input. This information allows librarians to stay current on patron preferences, anticipate demand for specific titles, and make educated judgments about new acquisitions and collection management. By exploiting this data, the library can ensure that its collection remains relevant and responsive to its users' requirements, resulting in increased user satisfaction and engagement. Overall, efficient management of these essential business activities is dependent on the availability and use of accurate and complete data. Robarts Library can improve its decision-making abilities by recognizing and addressing the individual data requirements of each operation.

2. Key Reports Identification

Report 1: Inventory Management Summary

The Inventory Management Summary Report is designed to provide librarians and management with detailed insights into the status and flow of books within the library system. The report's primary objective is to track book availability, understand demand patterns, and monitor the financial aspects of book procurement. By detailing the available copies of each book and highlighting demand patterns, the report enables librarians to make informed decisions about collection development, such as identifying which books may need additional copies or which underutilized genres might require scaling back. Additionally, understanding the financial aspects of book procurement helps the library to steward its budget wisely, allocating funds to areas of the collection that will offer the most value to patrons. It answers crucial questions such as:

- How many copies of each book are available at each branch of the library?
- Which books are being checked out most frequently and have the highest demand?
- What are the costs associated with newly acquired books for each branch within a specified time frame?

Data Elements
- Tables:
  - Book
  - BookCopy
  - Borrowing
  - LibraryBranch
- Attributes:
  - Book.Genre
  - Book.ISBN
  - Book.Title
  - BookCopy.ISBN
  - Borrowing.BorrowID
  - Book.AvailableCopies

- LibraryBranch.BranchName

Layout

- Executive Summary:
  - Overview of inventory management objectives and key findings.
- Introduction:
  - Explanation of the report's scope and methodology.
- Inventory Summary Table:
  - Columns: Title, Genre, Available Copies, Status, Branch Name
- Charts:
  - Pie Chart: Distribution of genres in the library's collection.
  - Bar Chart: Number of copies across different branches.
  - Trend Line: Variation in book availability over time.
- Summary Section:
  - Total number of books in inventory.
  - Average books per genre.
  - Branch with the highest number of available copies.
  - Recommendations for inventory optimization.

Inventory Management Summary Report SQL Queries

```
-- Books by Genre
SELECT Genre, COUNT(*) AS NumberOfBooks
FROM Book
GROUP BY Genre;

-- Books with Highest Borrowing Frequency
SELECT b.Title, b.ISBN, COUNT(br.BorrowDate) AS TimesBorrowed
FROM Book b
JOIN Borrows br ON b.ISBN = br.ISBN
GROUP BY b.Title, b.ISBN
ORDER BY TimesBorrowed DESC
LIMIT 10;

-- Book Availability Across Branches
SELECT b.Title, b.ISBN, b.AvailableCopies, lb.BranchName
FROM Book b
JOIN LibraryBranch lb ON b.BranchID = lb.BranchID
ORDER BY lb.BranchName, b.Title;
```

Report 2: User Services Metrics

The User Services Metrics Report aims to measure and analyze the library's service quality and efficiency from the perspective of user interactions and satisfaction. The report’s insights into loan durations and late returns are invaluable for refining lending policies and identifying potential issues in service delivery. For instance, if certain subjects show higher instances of late returns, this could indicate a need for more copies or a review of borrowing terms. This report seeks to answer questions about the usage patterns of the library's student population:

- What is the average borrowing rate among students from different academic majors?
- How long, on average, are books checked out?
- What trends can be identified in the occurrence of late returns?

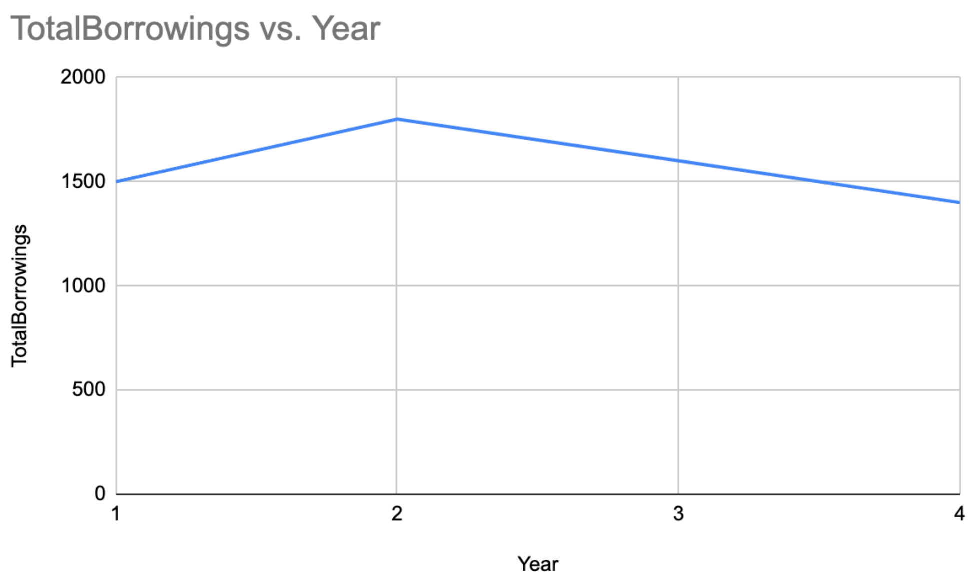
Data Elements

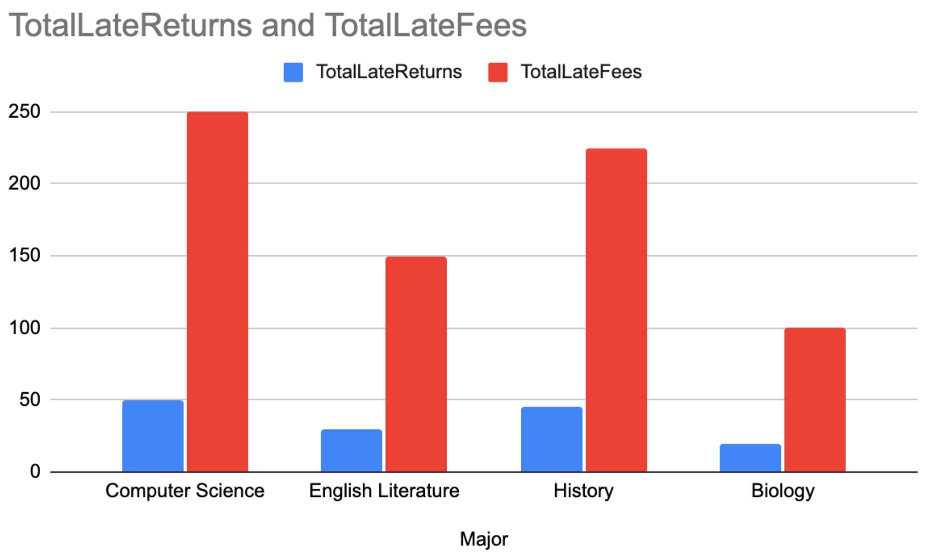
- Tables:
  - Borrows
  - Student
  - LateReturn
  - Incurs
- Attributes:
  - Borrows.StudentID
  - Student.Year
  - Student.Major
  - Borrows.ReturnDate
  - Borrows.BorrowDate
  - LateReturn.LateReturnID
  - LateReturn.LateFee
  - Incurs.StudentID

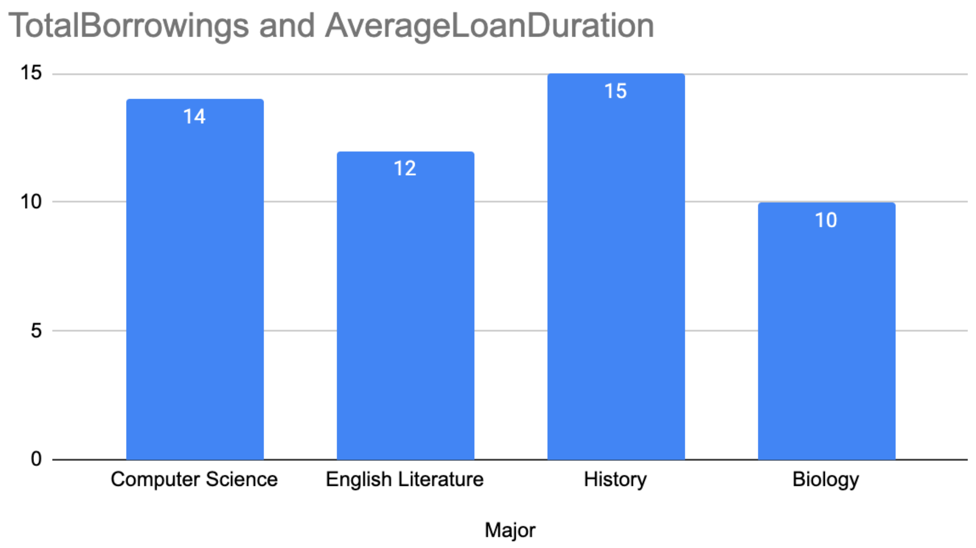
Layout

- Executive Summary:
  - Overview of user services objectives and key findings.
- Introduction:
  - Explanation of the report's purpose and methodology.
- Charts:
  - Line Graph: Borrowing trends over time segmented by student year.
  - Stacked Bar Chart: Comparison of borrowing rates across different majors.
  - Heat Map: Visualization of late returns frequency by subject area.
  - Histogram: Distribution of loan durations.
- Summary Section:
  - Analysis of borrowing patterns and trends.
  - Assessment of late return incidents.
  - Recommendations for enhancing user services based on findings.

Sample Layout of Charts:







User Services Metrics Report SQL Queries

```
-- Borrowing Trends by Student Year
SELECT s.Year, COUNT(*) AS TotalBorrowings
FROM Borrows b
JOIN Student s ON b.StudentID = s.StudentID
GROUP BY s.Year;

-- Average Number of Borrowings per Major
SELECT s.Major, COUNT(*) AS TotalBorrowings, AVG(DATEDIFF(b.ReturnDate,
b.BorrowDate)) AS AverageLoanDuration
FROM Borrows b
JOIN Student s ON b.StudentID = s.StudentID
GROUP BY s.Major;

-- Late Return Frequency by Subject Area
```

```
SELECT s.Major, COUNT(l.LateReturnID) AS TotalLateReturns, SUM(l.LateFee) AS TotalLateFees
FROM Incurs i
JOIN Student s ON i.StudentID = s.StudentID
JOIN LateReturn l ON i.LateReturnID = l.LateReturnID
GROUP BY s.Major;
```

Report 3: Financial Overview Report

The Financial Overview Report serves to deliver a comprehensive analysis of the library's financial activities related to acquisitions, fines, and user transactions. The report not only helps track the efficacy of fine policies but also provides insights into patron behavior regarding late returns. Understanding these financial patterns helps the library to forecast budgets more accurately and develop strategies to minimize losses through overdue books. Additionally, by comparing the expenditure on new acquisitions against the allocated budget, the library can ensure fiscal responsibility, making adjustments to stay within financial constraints while still meeting the informational needs of its users. This report addresses pivotal financial questions such as:

- How much revenue is generated from late return fines?
- What is the expenditure on new book acquisitions, and how does it compare with the allocated budget for such purchases?
- What are the financial implications of book reservations for the library's overall financial health?

Data Elements
- Tables:
    - LateReturn
    - CirculationRecord
    - Book
    - Student
- Attributes:
    - LateReturn.LateFee
    - LateReturn.RecordID
    - CirculationRecord.DueDate
    - CirculationRecord.RecordID
    - Book.Title
    - Book.ISBN
    - Student.StudentID
    - Student.Year

Layout
- Executive Summary:
    - Overview of financial objectives and key findings.
- Introduction:
    - Explanation of the report's focus and methodology.
- Tables:
    - Late returns: book title, ISBN, late days, late fees.
    - New book acquisitions: title, ISBN, purchase date, cost.
- Charts:
    - Bar Chart: Accumulation of fines over months.
    - Pie Chart: Distribution of spending across genres.
    - Scatter Plot: Relationship between late fees and loan durations.
- Summary Section:
    - Total revenue from late return fines.

- Expenditure on new book acquisitions.
- Average investment per book copy.
- Recommendations for financial management and optimization.

Financial Overview Report SQL Queries

```
-- Late Fee Revenue
SELECT MONTH(c.DueDate) AS DueMonth, SUM(l.LateFee) AS TotalLateFees
FROM LateReturn l
JOIN CirculationRecord c ON l.RecordID = c.RecordID
GROUP BY DueMonth
ORDER BY DueMonth;

-- Late Returns and Fees by Book
SELECT b.Title, b.ISBN, COUNT(l.LateReturnID) AS TotalLateReturns, SUM(l.LateFee) AS
TotalFees
FROM Book b
JOIN CirculationRecord c ON b.ISBN = c.ISBN
JOIN LateReturn l ON c.RecordID = l.RecordID
GROUP BY b.Title, b.ISBN
ORDER BY TotalFees DESC;

-- Late Fee Incidence by Student Year
SELECT s.Year, COUNT(l.LateReturnID) AS TotalLateReturnsByYear,
    SUM(l.LateFee) AS TotalFeesByYear,
    AVG(l.LateFee) AS AverageLateFeeByYear
FROM Student s
JOIN CirculationRecord c ON s.StudentID = c.StudentID
JOIN LateReturn l ON c.RecordID = l.RecordID
GROUP BY s.Year
ORDER BY s.Year;
```

Access Control

1. IT Administrators

IT administrators at the highest levels of responsibility might need a root account that has access to everything so that if there is a difficult problem anywhere, they can access and fix it. Also, if there are any major information architecture changes, then someone needs the access to implement these changes across the entire database. However, for day-to-day maintenance and for IT workers with lower responsibilities, they probably just need access to attributes in the Book and CirculationRecord tables and less personal identifiers like StudentID and StaffID from the Student and LibraryStaff tables, since that's where most of the daily updates happen and where data anomalies are therefore most likely to arise. There is generally no need for the IT professional to have access to private details like names, emails, and salaries because they are not authorized to change these anyway. Their job is to facilitate the upkeep of the database so that business processes like data storage, organization, and updates can all occur.

2. Librarians

Librarians interact most directly with most business processes so they need access to most tables and attributes. They catalog, organize, and track the conditions of existing books, as well as processing books for disposal or acquisition, so they need access to all the attributes in the Book table. They might need to update book attributes like Title, Author, Genre, and ISBN, and they might also need to search the catalog to help patrons find books. Librarians can also help patrons check books in and out, as well as process and assist with/adjust late returns and fees, so they need access to the Borrows, CirculationRecord, and LateReturns tables. Finally, librarians need access to attributes from the Student table major and year for analytics purposes. Also, they need access to Email to send important communications with patrons. They do not need write access to most of the LibraryStaff table because they should not be able to change their own position, branch, ID, salary, or even name without management oversight. Email might be the one attribute that they should have easy access to change.

3. Library Patrons

Finally, patrons, mostly expected to be students, need read access to tables like Book to search the catalog for the books they need. They also need to be able to update attributes like AvailableCopies in the Book table and StudentID in the Borrows table so that they can self-check out books. They also need read access to some LibraryStaff table attributes like Name and Email so they can find who to contact for assistance with their particular needs. They do not need direct access to other parts of the database because they should not be allowed to directly change most data about books, personnel, and fees without oversight from librarians. They would also not be involved in collecting analytics and deciding on which books should be discarded or acquired, so they don't need write access to the rest of Book and CirculationRecord.

Statement of Individual Contributions

Anna Li: Cover page & Executive Summary, Test cases (w/ SQL Test Cases Script), Access control
Sujaan Rajakumar: Key Reports and SQL code for reports.
Ang Zhao: Write SQL code to create the database basis of updated ERD and data requirement. Draw and improve the relational diagram based on mapping choices & ERD
Han Zheng: Fixed ER diagram and data dictionary from phase 1 and created a rough version of relational model