WOLAITA SODO UNIVERSITY

SCHOOL OF : INFORMATICS

DEPARTMENT OF : COMPUTER SCIENCE

COURSE TITLE : ADVANCED DATABASE

PROJECT TITLE: LIBRARY MANAGEMENT SYSTEM

GROUP MEMBER AND ID NUMBER

	NAME	ID
1	KASU JIMA	UGR/91890/16
2	DAGMAWIT ADDISE	UGR/91594/16
3	FIKRU MATHIWOS	UGR/92629/16
4	ELSA LEUL	UGR/92608/16
5	FOZIYA ENDRIS	UGR/91779/16

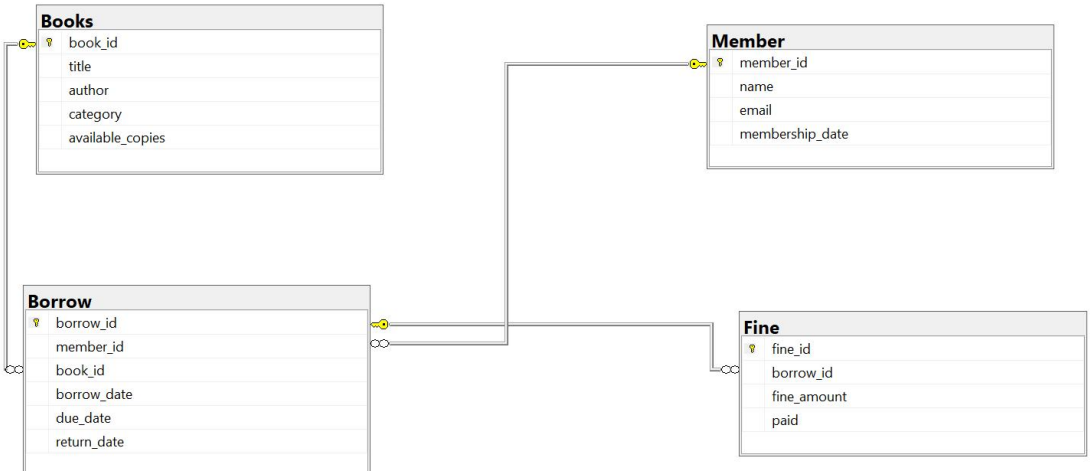
Introduction to Library Management System

A Library Management System (LMS) is a comprehensive software application designed to facilitate the efficient management of library resources. It serves as a crucial tool for librarians, enabling them to organize, track, and manage various aspects of library operations, including book acquisitions, cataloging, circulation, and member management.

In an era where information is abundant, a well-structured LMS enhances accessibility and efficiency, allowing users to easily find and borrow materials. It streamlines processes such as checking out and returning books, managing fines, and maintaining an up-to-date inventory. Additionally, an LMS often includes features for online catalog access, enabling patrons to search for resources remotely.

By automating repetitive tasks, a Library Management System reduces administrative workload, allowing librarians to focus on providing better services and support to users. The implementation of such a system not only improves operational efficiency but also enriches the overall user experience, making library services more responsive to the needs of the community.

1. Design a relational database application based on your chosen project title.



Discretionary Access Control (DAC)

means access rights are assigned by the owner of the data. Owners can grant or revoke access to other users

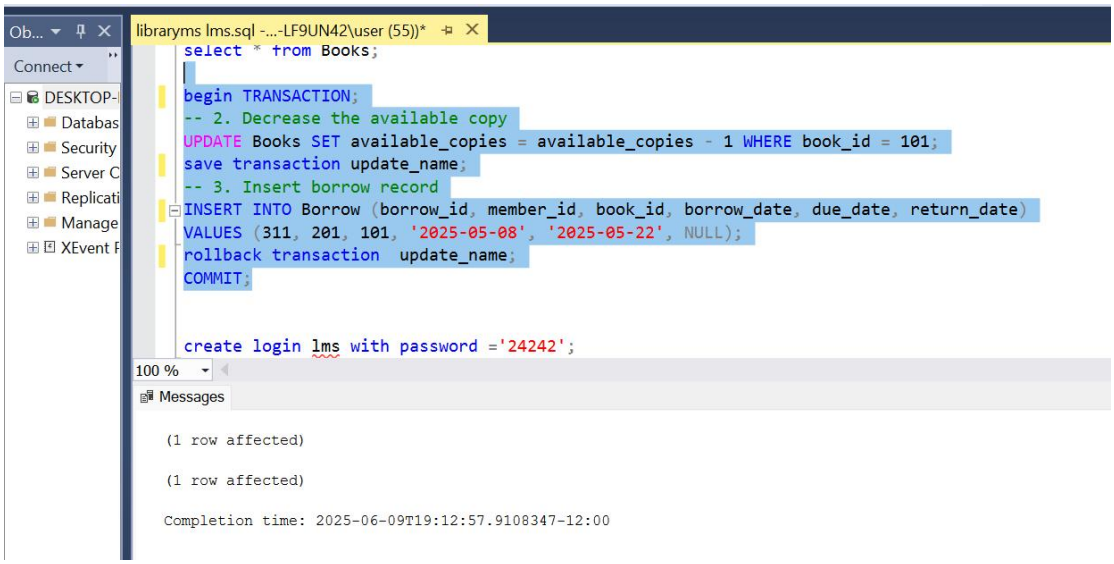
Statistical databases

provide statistical information (like averages, sums) about a population without revealing individual records. The challenge is to prevent inference attacks, where a user deduces individual data by repeated queries.

```
create database librarymanagementsystem
use librarymanagementsystem;

CREATE TABLE Books (
    book_id INT PRIMARY KEY,
    title VARCHAR(100),
    author VARCHAR(100),
    category VARCHAR(50),
    available_copies INT
);
CREATE TABLE Member (
    member_id INT PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100),
    membership_date DATE
);
CREATE TABLE Borrow (
    borrow_id INT PRIMARY KEY,
    member_id INT,
    book_id INT,
    borrow_date DATE,
    due_date DATE,
    return_date DATE,
    FOREIGN KEY (member_id) REFERENCES Member(member_id),
    FOREIGN KEY (book_id) REFERENCES Books(book_id)
);
CREATE TABLE Fine (
    fine_id INT PRIMARY KEY,
    borrow_id INT,
    fine_amount DECIMAL(8, 2),
    paid varchar (100),
    FOREIGN KEY (borrow_id) REFERENCES Borrow(borrow_id)
);
INSERT INTO Books (book_id, title, author, category, available_copies) VALUES
(101, '1984', 'George Orwell', 'Dystopian', 4),
(102, 'To Kill a Mockingbird', 'Harper Lee', 'Classic', 2),
(103, 'The Great Gatsby', 'F. Scott Fitzgerald', 'Classic', 3),
(104, 'Moby Dick', 'Herman Melville', 'Adventure', 1),
(105, 'Pride and Prejudice', 'Jane Austen', 'Romance', 5),
```

```
(106, 'The Hobbit', 'J.R.R. Tolkien', 'Fantasy', 3),
(107, 'War and Peace', 'Leo Tolstoy', 'Historical', 2),
(108, 'Hamlet', 'William Shakespeare', 'Drama', 4),
(109, 'The Catcher in the Rye', 'J.D. Salinger', 'Classic', 2),
(110, 'Brave New World', 'Aldous Huxley', 'Dystopian', 3);
INSERT INTO Member (member_id, name, email, membership_date) VALUES
(201, 'Kasu J ', 'KASS@gmail.com', '2021-01-15'),
(202, 'Denbobe D', 'done@gmail.com', '2022-03-22'),
(203, 'Mercy G', 'mercy@gmail.com', '2023-05-10'),
(204, 'Dagi A', 'dagi@gmail.com', '2022-07-08'),
(205, 'Aklilu A', 'akililu@gmail.com', '2021-09-12'),
(206, 'Foziya E', 'fozi@gmail.com', '2023-02-03'),
(207, 'Grace M', 'grace@gmail.com', '2021-06-20'),
(208, 'Fikru M', 'fikru@gmail.com', '2022-11-01'),
(209, 'Elsa L', 'elsa@gmail.com', '2023-04-18'),
(210, 'Jack A', 'jack@gmail.com', '2022-12-25');
INSERT INTO Borrow (borrow_id, member_id, book_id, borrow_date, due_date, return_date) VALUES
(301, 201, 101, '2023-01-01', '2023-01-15', '2023-01-16'),
(302, 202, 102, '2023-01-10', '2023-01-24', '2023-01-22'),
(303, 203, 103, '2023-01-12', '2023-01-26', '2023-01-30'),
(304, 204, 104, '2023-02-01', '2023-02-15', '2023-02-16'),
(305, 205, 105, '2023-02-10', '2023-02-24', NULL),
(306, 206, 106, '2023-03-01', '2023-03-15', '2023-03-14'),
(307, 207, 107, '2023-03-10', '2023-03-24', '2023-03-26'),
(308, 208, 108, '2023-04-01', '2023-04-15', '2023-04-12'),
(309, 209, 109, '2023-04-10', '2023-04-24', '2023-05-01'),
(310, 210, 110, '2023-05-01', '2023-05-15', NULL);
INSERT INTO Fine (fine_id, borrow_id, fine_amount, paid) VALUES
(401, 301, 2.00, 'TRUE'),
(402, 302, 0.00, 'TRUE'),
(403, 303, 5.00, 'FALSE'),
(404, 304, 1.00, 'TRUE'),
(405, 305, 0.00, 'FALSE'),
(406, 306, 0.00, 'TRUE'),
(407, 307, 3.00, 'FALSE'),
(408, 308, 0.00, 'TRUE'),
(409, 309, 7.50, 'FALSE'),
(410, 310, 0.00, 'FALSE');
select * from Books;
begin TRANSACTION;
-- 2. Decrease the available copy
UPDATE Books SET available_copies = available_copies - 1 WHERE book_id = 101;
save transaction update_name;
-- 3. Insert borrow record
INSERT INTO Borrow (borrow_id, member_id, book_id, borrow_date, due_date, return_date)
VALUES (311, 201, 101, '2025-05-08', '2025-05-22', NULL);
rollback transaction  update_name;
COMMIT;
create login lms with password ='24242';
Go
create user kasu for login lms;
GO
CREATE ROLE lms_G12;
GO
SP_ADDROLEMEMBER 'lms_G12', 'USER12';
GO
grant update, select, insert on books to kasu with grant option;
GO
revoke grant option for update ,select,insert on books from kasu ;
go
CREATE VIEW secure_fine_stats
AS
SELECT
    COUNT(*) AS fine_count,
    CASE
        WHEN COUNT(*) >= 3 THEN AVG(fine_amount)
        ELSE NULL
    END AS avg_fine
FROM Fine;
```

2. Implement transaction management for critical operations. For example, in a bank management system, this would include fund transfers.




```
select * from Books;

begin TRANSACTION;
-- 2. Decrease the available copy
UPDATE Books SET available_copies = available_copies - 1 WHERE book_id = 101;
save transaction update_name;

-- 3. Insert borrow record
INSERT INTO Borrow (borrow_id, member_id, book_id, borrow_date, due_date, return_date)

VALUES (311, 201, 101, '2025-05-08', '2025-05-22', NULL);
rollback transaction  update_name;
```

.00 %

Results | Messages

	book_id	title	author	category	available_copies
1	101	1984	George Orwell	Dystopian	3
2	102	To Kill a Mockingbird	Harper Lee	Classic	2
3	103	The Great Gatsby	F. Scott Fitzgerald	Classic	3
4	104	Moby Dick	Herman Melville	Adventure	1
5	105	Pride and Prejudice	Jane Austen	Romance	5
6	106	The Hobbit	J.R.R. Tolkien	Fantasy	3
7	107	War and Peace	Leo Tolstoy	Historical	2
8	108	Hamlet	William Shakespeare	Drama	4
9	109	The Catcher in the Rye	J.D. Salinger	Classic	2
10	110	Brave New World	Aldous Huxley	Dystopian	3

3. Demonstrate the use of commit, rollback and savepoints.

Save point

```sql
select * from Books;

begin TRANSACTION;
-- 2. Decrease the available copy
UPDATE Books SET available_copies = available_copies + 1 WHERE book_id = 101;
save transaction update_available_copies;

-- 3. Insert borrow record
INSERT INTO Borrow (borrow_id, member_id, book_id, borrow_date, due_date, return_date)

VALUES (311, 201, 101, '2025-05-08', '2025-05-22', NULL);
rollback transaction  update_available_copies;
COMMIT;
```

100 %

Results | Messages

	book_id	title	author	category	available_copies
1	101	1984	George Orwell	Dystopian	3
2	102	To Kill a Mockingbird	Harper Lee	Classic	2
3	103	The Great Gatsby	F. Scott Fitzgerald	Classic	3
4	104	Moby Dick	Herman Melville	Adventure	1
5	105	Pride and Prejudice	Jane Austen	Romance	5
6	106	The Hobbit	J.R.R. Tolkien	Fantasy	3
7	107	War and Peace	Leo Tolstoy	Historical	2
8	108	Hamlet	William Shakespeare	Drama	4
9	109	The Catcher in the Rye	J.D. Salinger	Classic	2
10	110	Brave New World	Aldous Huxley	Dystopian	3

rollback

libraryms lms.sql -...-LF9UN42\user (55))*

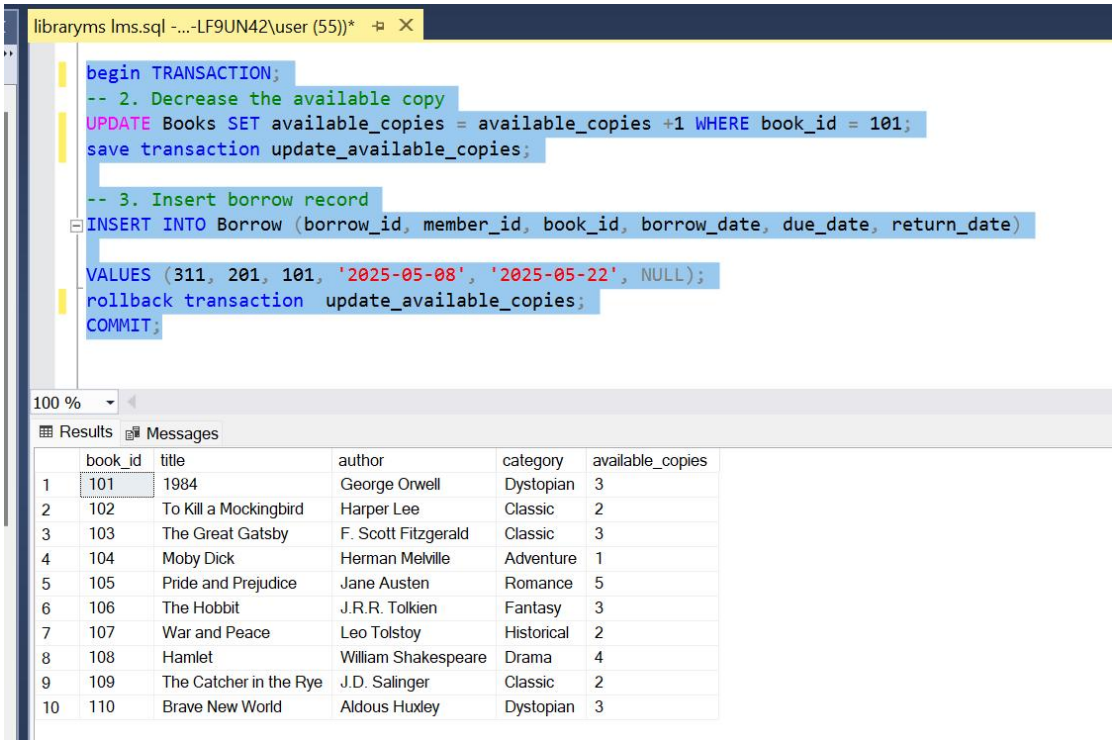
```sql
select * from Books;

begin TRANSACTION;
-- 2. Decrease the available copy
UPDATE Books SET available_copies = available_copies +1 WHERE book_id = 101;
save transaction update_available_copies;

-- 3. Insert borrow record
INSERT INTO Borrow (borrow_id, member_id, book_id, borrow_date, due_date, return_date)

VALUES (311, 201, 101, '2025-05-08', '2025-05-22', NULL);
rollback transaction  update_available_copies;
COMMIT;
```

100 %

Results | Messages

	book_id	title	author	category	available_copies
1	101	1984	George Orwell	Dystopian	2
2	102	To Kill a Mockingbird	Harper Lee	Classic	2
3	103	The Great Gatsby	F. Scott Fitzgerald	Classic	3
4	104	Moby Dick	Herman Melville	Adventure	1
5	105	Pride and Prejudice	Jane Austen	Romance	5
6	106	The Hobbit	J.R.R. Tolkien	Fantasy	3
7	107	War and Peace	Leo Tolstoy	Historical	2
8	108	Hamlet	William Shakespeare	Drama	4
9	109	The Catcher in the Rye	J.D. Salinger	Classic	2
10	110	Brave New World	Aldous Huxley	Dystopian	3

Commit



```
libraryms lms.sql -...-LF9UN42\user (55))*
begin TRANSACTION;
-- 2. Decrease the available copy
UPDATE Books SET available_copies = available_copies +1 WHERE book_id = 101;
save transaction update_available_copies;

-- 3. Insert borrow record
INSERT INTO Borrow (borrow_id, member_id, book_id, borrow_date, due_date, return_date)

VALUES (311, 201, 101, '2025-05-08', '2025-05-22', NULL);
rollback transaction  update_available_copies;
COMMIT;
```

100 %

Results | Messages

	book_id	title	author	category	available_copies
1	101	1984	George Orwell	Dystopian	3
2	102	To Kill a Mockingbird	Harper Lee	Classic	2
3	103	The Great Gatsby	F. Scott Fitzgerald	Classic	3
4	104	Moby Dick	Herman Melville	Adventure	1
5	105	Pride and Prejudice	Jane Austen	Romance	5
6	106	The Hobbit	J.R.R. Tolkien	Fantasy	3
7	107	War and Peace	Leo Tolstoy	Historical	2
8	108	Hamlet	William Shakespeare	Drama	4
9	109	The Catcher in the Rye	J.D. Salinger	Classic	2
10	110	Brave New World	Aldous Huxley	Dystopian	3

4. Simulate concurrent transactions and analyze the impact of concurrency control techniques

1. Dirty Read:

A dirty read occurs when a transaction reads data that has been modified by another transaction that has not yet been committed. This can lead to inconsistencies because if the other transaction is rolled back, the data read by the first transaction may no longer be valid.

solution

Setting isolation level read committed but in our project case the dbms by default set it.

syntax (set transaction isolation level read committed)

2. Incorrect Summary:

This situation arises when a transaction reads intermediate results from other transactions that are still in progress. If the transactions are rolled back or modified after the summary is computed, the summary may be incorrect or misleading. This can happen in scenarios where aggregate functions like SUM, AVG, etc., are used on data that is being concurrently modified.

solution

Setting isolation level repeatable read

syntax (set transaction isolation level repeatable read)

3. Lost Update:

A lost update occurs when two transactions read the same data and then both make updates based on that data. If one transaction commits its changes after the other has already read the original data, the first transaction's update can be lost because it was overwritten by the second transaction's update.

solution

set transaction isolation level repeatable read

5.Implement mechanisms to handle deadlocks and ensure the serializability of transactions.

A deadlock is a situation where two or more transactions are waiting for each other to release locks on resources they need to proceed. For example, Transaction A holds a lock on Resource 1 and is waiting for Resource 2, while Transaction B holds a lock on Resource 2 and is waiting for Resource 1. Neither transaction can proceed, leading to a standstill. Most database management systems have deadlock detection mechanisms to resolve this issue by terminating one of the transactions.

Transction 1

```
use librarymanagementsystem;
--transaction 1
--drtiy read
select *from books
BEGIN TRANSACTION;

UPDATE Books SET available_copies = available_copies + 1 WHERE Book_ID = 101; -- Returning BookID 1
WAITFOR DELAY '00:00:15'; -- Simulate some processing time
rollback transaction
--non reapatable
set transaction isolation level repeatable read--solition
begin transaction
select available_copies from books where book_id=105
```

```
WAITFOR DELAY '00:00:15';
select available_copies from books where book_id=105
commit transaction
--lost update
set transaction isolation level repeatable read
begin transaction
declare @itmcopy int
select @itmcopy=available_copies from books where book_id=105
WAITFOR DELAY '00:00:10';
set @itmcopy=@itmcopy-1
update Books
set available_copies=@itmcopy where book_id=105
print @itmcopy
commit transaction
--deadluck
begin transaction
update books set title= 'the devil' where book_id =101
update books set title= 'war' where book_id =102
commit transaction
```

Transactin 2

```
use librarymanagementsystem;
select* from books
--transaction 2
--drtiy read
set transaction isolation level read uncommitted--solition
select * from books where book_id=101 ;


--non reapatable
update Books set  available_copies=15 where book_id=105
--lost update
begin transaction
declare @itmcopy int
select @itmcopy=available_copies from books where book_id=105
WAITFOR DELAY '00:00:5';
set @itmcopy=@itmcopy-3
update books
set available_copies=@itmcopy where book_id=105
print @itmcopy
commit transaction
--deadluck
begin transaction
update books set title= 'the devil2' where book_id =102
update books set title= 'war2' where book_id =101
commit transaction
```