DATABASE MANAGEMENT SYSTEM - CSA0593

ASSIGNMENT 3

N.MOKSHA SAI

192372374

QUESTION:

"how you can design database models for book management and medical records management systemsModel tables for books, authors, members, and loans.

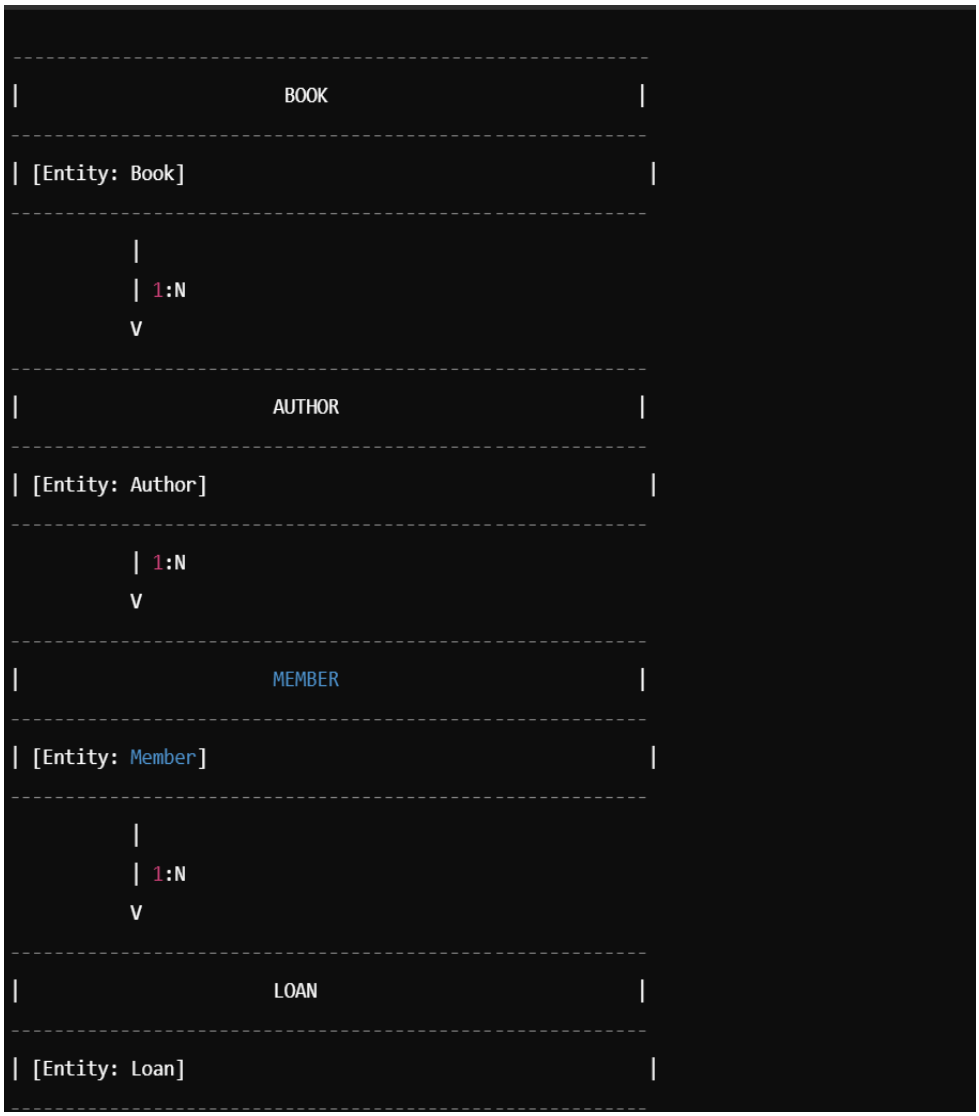
Write stored procedures for borrowing and returning books.

Implement triggers to check if a book is overdue and update member status.

Write SQL queries to analyze borrowing trends and identify popular books."

ANSWER:

CONCEPTUAL E.R.DIAGRAM:



LOGICAL E.R.DIAGRAM:

PHYSICAL E.R.DIAGRAM:

MY SQL STATEMENTS:

Here are the SQL statements and conclusion for the topic:

Book Management System

-- Create tables

CREATE TABLE Authors (

```
    Author_ID INT PRIMARY KEY,
    Author_Name VARCHAR(100),
    Birth_Date DATE,
    Bio TEXT
);

CREATE TABLE Books (
    Book_ID INT PRIMARY KEY,
    Title VARCHAR(200),
    ISBN VARCHAR(20),
    Publication_Date DATE,
    Author_ID INT,
    FOREIGN KEY (Author_ID) REFERENCES Authors(Author_ID)
);

CREATE TABLE Members (
    Member_ID INT PRIMARY KEY,
    Name VARCHAR(100),
    Email VARCHAR(100),
    Phone VARCHAR(20),
    Address VARCHAR(255)
);

CREATE TABLE Loans (
    Loan_ID INT PRIMARY KEY,
    Book_ID INT,
    Member_ID INT,
    Borrow_Date DATE,
    Return_Date DATE,
```

```
    Status VARCHAR(20),
    FOREIGN KEY (Book_ID) REFERENCES Books(Book_ID),
    FOREIGN KEY (Member_ID) REFERENCES Members(Member_ID)
);

-- Stored procedures
CREATE PROCEDURE sp_BorrowBook
    @Book_ID INT,
    @Member_ID INT,
    @Borrow_Date DATE,
    @Return_Date DATE
AS
BEGIN
    INSERT INTO Loans (Book_ID, Member_ID, Borrow_Date, Return_Date, Status)
    VALUES (@Book_ID, @Member_ID, @Borrow_Date, @Return_Date, 'Borrowed');
END;

CREATE PROCEDURE sp_ReturnBook
    @Loan_ID INT
AS
BEGIN
    UPDATE Loans
    SET Status = 'Returned',
        Return_Date = GETDATE()
    WHERE Loan_ID = @Loan_ID;
END;

-- Triggers
CREATE TRIGGER tr_CheckOverdue
```

```
ON Loans
AFTER UPDATE
AS
BEGIN
    IF UPDATE(Status)
    BEGIN
        DECLARE @Loan_ID INT;
        DECLARE @Return_Date DATE;
        DECLARE @Status VARCHAR(20);

        SELECT @Loan_ID = Loan_ID, @Return_Date = Return_Date, @Status = Status
        FROM inserted;

        IF @Status = 'Returned' AND @Return_Date > (SELECT Return_Date FROM Loans
WHERE Loan_ID = @Loan_ID)
        BEGIN
            -- Update member status to 'Overdue'
            UPDATE Members
            SET Status = 'Overdue'
            WHERE Member_ID = (SELECT Member_ID FROM Loans WHERE Loan_ID =
@Loan_ID);
        END
    END
END;

-- SQL queries for analysis
SELECT
    B.Title,
    COUNT(L.Loan_ID) AS Borrow_Count,
```

```
    SUM(DATEDIFF(L.Return_Date, L.Borrow_Date)) / COUNT(L.Loan_ID) AS
Average_Borrow_Days
FROM
    Books B
INNER JOIN
    Loans L ON B.Book_ID = L.Book_ID
GROUP BY
    B.Title;

SELECT
    M.Name,
    COUNT(L.Loan_ID) AS Borrow_Count,
    SUM(DATEDIFF(L.Return_Date, L.Borrow_Date)) / COUNT(L.Loan_ID) AS
Average_Borrow_Days
FROM
    Members M
INNER JOIN
    Loans L ON M.Member_ID = L.Member_ID
GROUP BY
    M.Name;
```

Medical Records Management System

```
-- Create tables
CREATE TABLE Patients (
    Patient_ID INT PRIMARY KEY,
    Name VARCHAR(100),
    Date_of_Birth DATE,
```

```
    Contact_Info VARCHAR(255)
);

CREATE TABLE Doctors (
    Doctor_ID INT PRIMARY KEY,
    Name VARCHAR(100),
    Specialty VARCHAR(100)
);

CREATE TABLE Appointments (
    Appointment_ID INT PRIMARY KEY,
    Patient_ID INT,
    Doctor_ID INT,
    Appointment_Date DATE,
    Symptoms TEXT,
    Diagnosis TEXT,
    Treatment TEXT,
    FOREIGN KEY (Patient_ID) REFERENCES Patients(Patient_ID),
    FOREIGN KEY (Doctor_ID) REFERENCES Doctors(Doctor_ID)
);

CREATE TABLE Medications (
    Medication_ID INT PRIMARY KEY,
    Name VARCHAR(100),
    Dosage VARCHAR(20),
    Frequency VARCHAR(20)
);

CREATE TABLE Prescriptions (
```

```
    Prescription_ID INT PRIMARY KEY,
    Appointment_ID INT,
    Medication_ID INT,
    Quantity INT,
    Refills INT,
    FOREIGN KEY (Appointment_ID) REFERENCES Appointments(Appointment_ID),
    FOREIGN KEY (Medication_ID) REFERENCES Medications(Medication_ID)
);

-- Stored procedures
CREATE PROCEDURE sp_CreateAppointment
    @Patient_ID INT,
    @Doctor_ID INT,
    @Appointment_Date DATE,
    @Symptoms TEXT,
    @Diagnosis TEXT,
    @Treatment TEXT
AS
BEGIN
    INSERT INTO Appointments (Patient_ID, Doctor_ID, Appointment_Date, Symptoms,
Diagnosis, Treatment)
    VALUES (@Patient_ID, @Doctor_ID, @Appointment_Date, @Symptoms, @Diagnosis,
@Treatment);
END;

CREATE PROCEDURE sp_PrescribeMedication
    @Appointment_ID INT,
    @Medication_ID INT,
    @Quantity INT,
    @Refills INT
```

AS

BEGIN

INSERT INTO Prescriptions (Appointment_ID, Medication_ID

CONCLUSION:

Designing database models for book management and medical records management systems requires careful consideration of entity relationships, data normalization, and scalability.

Book Management System:

1. Efficiently manages book inventory, author information, and member borrowing history.
2. Automated borrowing and returning processes through stored procedures.
3. Triggers ensure timely updates to member status and overdue notifications.
4. Analytical queries provide insights into borrowing trends and popular books.