Subject Assignment موضوع الدرس

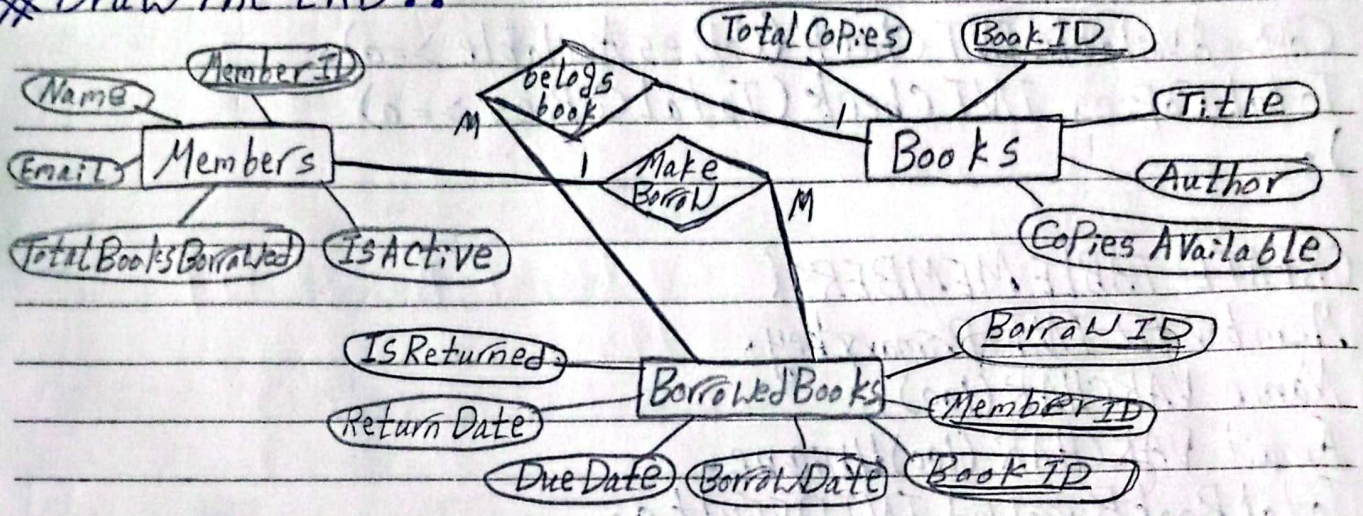
Date التاريخ

TASK 1

Part 1: Enhanced Requirements Analysis & ERD Design:-

- **Books** (BookID, Title, Author, CopiesAvailable, TotalCopies)
- **Members** (MemberID, Name, Email, TotalBooksBorrowed, isActive)
- **BorrowedBooks** (BorrowID, MemberID, BookID, BorrowDate, DueDate, ReturnDate, IsReturned)

※ Draw The ERD??



① one Member can borrow Many BorrowedBooks (1 to Many)

② one Book can appear in Many BorrowedBooks records (1 to Many)

※ All relationships are clearly connected through with Foreign key That (No Fan traps)

※ Every BorrowedBook is tied directly to a Book and a Member That (No chasm traps)

Part 2: SQL Implementation Task:-

- Stored Procedure
- Index
- Function (GetBookBorrowed)
- Triggers: PreventBorrowNoCopies
- Table Creation
- Sample Data Insertion at least
  - 3 books
  - 3 Members
  - 5 borrowing attempts

Table Creation:-

```
CREATE TABLE Books(
BookID INT Primary kEy,
Title VARCHAR(100),
Author VARCHAR(100),
CopiesAvailable INT Check (CopiesAvailable >=0),
Total Copies INT Check (TotalCopies >=0)
);

CREATE TABLE MEMBERS(
MemberID INT Primary key,
Name VARCHAR(100),
Email VARCHAR(100)Unique,
TotalBookBorroVed INTDEFult 0,
IsActive BooLEN DEFAULT TRUE,
);

CREATE TABLE BorroVedBooks(
BorroVID INT Primary key,
MemberID INT,
BookID INT,
BorroV Date ~~INT~~ Date,
Due Date ~~INT~~ Date,
ReturnDate ~~INT~~ Date,
IsReturned BooLEAN DEFAULT FALSE,
Foreign key (MemberID) REFERANCES MEmbers (MemberID),
Foreign key (BookID) REFERANCES Books (BookID)
);
```

Subject موضوع الدرس | Date التاريخ

Stored Procedure :-

```
// Stored Procedure BorrowBook
CREATE Procedure BorrowBook(
 IN P_MemberID INT,
 IN P_BookID INT,
 IN P_BorrowDate DATE,
 IN P_DueDate DATE
)
BEGIN
          // DECLare Variables For check
   DECLARE V_IsActive BooLEAN;
   DECLARE V_CopiesAvailable INT;
          // check IF The Member Is Active
SELECT IsActive INTO V_IsACtive
From Members
Where MemberID = P_MemberID;

IF V_IsACtive = FALSE THEN
  SIGNAL SolSTATE '45000'
 SET MESSAGE TEXT = 'Member is not Active';
ENDIF;
          // Check IF The Book Available Copies
SELECT Copies Available INTo V_Copies Available
From books
Where BookID = P_BookID;

IF V_CopiesAvailable = 0 THEN
  SIGNAL SolSTATE '45000'
SET MESSAGETEXT = 'Book is not Available';
ENDIF;
```

Subject موضوع الدرس | Date التاريخ

```
    // Insert a new record into BorrowedBooks
INSERT INTO BorrowedBooks (BookID, MemberID, BorrowDate,
    DueDate, IsReturned)
Values (p_BookID, p_MemberID, p_DueDate, p_BorrowDate
    , FALSE);
    // Decrease Copies In Books
UPDATE Books          Available
SET CopiesAvailable = CopiesAvailable - 1
Where BookID = p_BookID;
    // Increase TotalBooksBorrowed In Members
UPDATE Members
SET TotalBooksBorrowed = TotalBooksBorrowed + 1
Where MemberID = p_MemberID;

END IF;
DELIMITER;
```

* Index Creation:

```
CREATE INDEX idx_BookID ON Books (BookID);
CREATE INDEX idx_MemberID ON Members (MemberID);
```

* Function (GetBooksBorrowed) :-

```
CREATE FUNCTION GETBorrowedBooks(p_MemberID INT)
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE Count_borrowed INT;
SELECT Count(*) INTO Count_borrow FROM BorrowBooks
Where MemberID = p_MemberID AND IsReturned = FALSE;
RETURN Count_Borrow;
END;
    SELECT GetBooksBorrowed(1);
```

Trigger (Prevent Borrow If No Copies) :-

```
CREATE Trigger PreventBorrowIfNoCopies
BEFORE INSERT ON BorrowedBooks
FOR EACH ROW
BEGIN
  //DECleare Values
 DECLARE V-CopiesAvailable INT;
 DECLARE V-IsActive BOOLEAN;
  //GET Copies Available
SELECT CopiesAvailable INTO V-CopiesAvailable
From Books
Where BookID=NEW.BookID;
  //GET ISActive
SELECT ISActive INTO V-ISActive
From Members
Where MemberID=NEW.MemberID;

IF V-CopiesAvailable = 0 THEN
 SIGNAL SQLState '45000'
 SET MESSAGE TEXT = 'No Copies Available'
END IF;

IF V-IsActive = FALSE THEN
 SIGNAL SQLstate '45000'
 SET MESSAGE TEXT='MemberIS Not Active'

ENDIF;
END;
```

Subject موضوع الدرس | Date التاريخ

Sample Data Insertion:-

```
// Insert Books
INSERT INTO Books (Title, Author, CopiesAvailable, BookID)
VALUES
('The Lord of Rings', 'J.R.R. Tolkien', 3, 1),
('Pride and Prejudice', 'Jane Austen', 1, 2),
('Brave New World', 'Aldous Huxley', 1, 3);

// Insert Members
INSERT INTO Members VALUES
(1, 'Alice', 'alice@example.com', 0, True),
(2, 'Bob', 'bob@example.com', 0, False),
(3, 'Charlie', 'charlie@example.com', 0, True);

// Insert Borrowing Attempts
CALL BorrowBook(1, 1, '2024-09-15');
CALL BorrowBook(2, 2, '2024-09-16');
CALL BorrowBook(3, 3, '2024-09-17');
CALL BorrowBook(4, 2, '2024-09-18');
CALL BorrowBook(5, 3, '2024-09-19');
```

Subject موضوع الدرس Date التاريخ

TASK 2

The Relational Algebra (before optimized):-

Π {S.Name, C.Title, E.Grade} (
σ {S.StudentID = E.StudentID AND C.CourseID = E.CourseID AND S.Major = 'ComputerScience' AND (Student x Course x ENROLLMENT)
)

The Query Tree (before optimized):

Π (S.name, C.Titles, E.Grade)
|
σ (Student = E.StudentID AND C.CourseID = E.CourseID AND Major = 'ComputerScience' AND C.Credits >= 3)
|
(Student x Course x ENRollments)

optimization Relational Algebra using Heuristics:-

steps:- // Applay selection

① σ {S.Major = 'ComputerScience'} (Student)
σ (C.Credits >= 3) (course)

// Replace CardisenProdect with join

② (Student $\bowtie$ {S.StudentID = E.StudentID} EnRollment
$\bowtie$ {C.CourseID = E.CourseID} Course

// APPLy all Relationel Algebra

∴ Optimized Relational Algebra:-
Π (S.name, C.title, E.Grade) (
(σ {S.Major = 'ComputerScience'} (Student)
$\bowtie$ (S.StudentID = E.StudentID} Enrollemed)
$\bowtie$ (C.CourseID = E.CourseID)
σ (C.Credits >= 3} (course)
)

26/4/2025 11:18 REDMI 12

Subject موضوع الدرس Date التاريخ

The optimized Query Tree :-