University Library System

The following assumptions have been made: -

- The library staff would be the only ones using the library system.
- A new member start date would be given to an existing member after their membership renewal.
- It is assumed that the member cannot rent the same book that they have not returned.
- When a new member joins, the number on the member card would be the primary way to identify the member. The member no is going to be the primary key in members table.

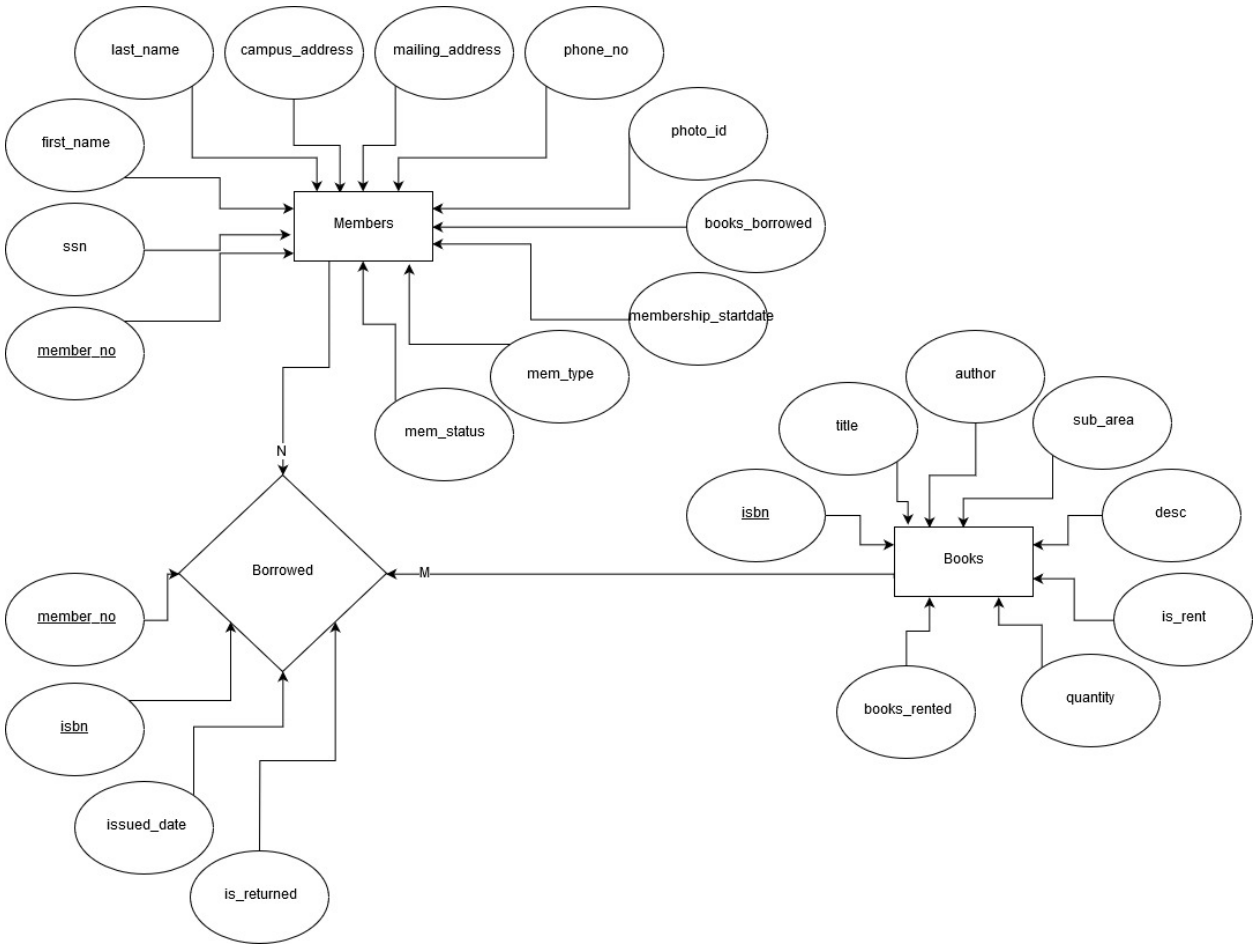
Based on the above assumptions and the requirements the following entities and their relationships have been identified: -

1. MEMBERS:
    - The primary key for the table is the 'member_No'.
    - The SSN is going to be unique for every row. Though the ssn would not be used as a primary key for security reasons.
    - The status column would indicate if the member is active or not. Only 'Active' members can rent a book.
    - To avoid increasing the database size the 'photo_id' column would store the path to the actual member photo rather than storing the actual photo.
    - If a library employee no longer works at the library the 'mem_type' would be changed and their 'Status' would become inactive.
    - The 'books_borrowed' will indicate the no of books rented and would be updated every time the member rents or returns books.
    - For ease of use this table would consider the library staff and professors as members of the library. All the members are differentiated using the 'mem_type' column. This is design choice so as avoid creating a separate table for staff as it does not fit in the given scenario. Additionally, this will facilitate the renting of books by library staff without having to create separate rows in the Members table, thus avoiding redundant data.

2. BOOKS
    - The primary key for the table would be the ISBN of the book as it is unique.
    - User can check if a book is available for rent by checking the quantity, while the books already given on rent can be checked from the 'books_rented' column.
    - The quantity and books_rented columns would be updated everytime a book is rented or returned.
    - It is considered that if a book needs to be acquired the Quantity and 'Books_rented' columns would show the amount as 0.
    - The column 'is_rent' specifies if a book can be rented or not.

3. Borrowed
    - The borrowed table uses composite keys which a combination of the ISBN of the book and the member's Member_No along with if the book is returned or not.
    - The 'is_returned' attribute will indicate if the book is returned or not.

1. EER Diagram: -

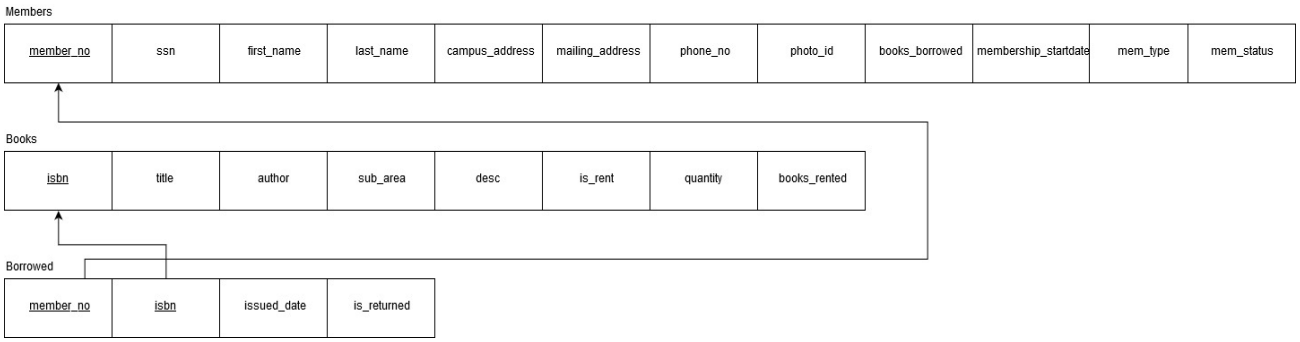


2. Relational Database Schema:

As per the EER Diagram, the database would consider of three tables: - Books, Members and Borrowed. The borrowed table store the information of the books that are currently borrowed by the members. The borrowed table has a composite key consisting of Book isbn, member_no and is_returned indicating if the book is returned or not. This combination will ensure that a book that has been already by a member and returned, cannot be rented by the same member.

The primary key for the members table would be member_no, this can be used to look up a member instead of looking up using the ssn, this way the ssn would remain private.

The primary key for the books table would be the isbn as it is unique and as mentioned in the requirements changes for every book even with a new edition.



3. Create Queries:

a. Database:

CREATE SCHEMA `universitylibrarysystem` ;

b. Tables:

Members: -

```
CREATE TABLE `librarysystem`.`members` (
        `member_No` INT NOT NULL,
        `ssn` INT NOT NULL,
        `first_name` VARCHAR(45) NOT NULL,
        `last_name` VARCHAR(45) NOT NULL,
        `campus_address` VARCHAR(45) NOT NULL,
        `mailing_address` VARCHAR(45) NOT NULL,
        `phone_no` VARCHAR(10) NOT NULL,
        `photo_id` VARCHAR(45) NOT NULL,
        `books_borrowed` INT NOT NULL DEFAULT 0,
        `membership_startdate` DATE NOT NULL,
        `mem_type` VARCHAR(45) NOT NULL,
        `mem_status` VARCHAR(45) NOT NULL,
        PRIMARY KEY (`Member_No`),
        UNIQUE INDEX `SSN_UNIQUE` (`SSN` ASC) VISIBLE);
```

Books: -

```
CREATE TABLE `librarysystem`.`books` (
        `isbn` INT NOT NULL,
        `title` VARCHAR(45) NOT NULL,
        `author` VARCHAR(45) NOT NULL,
        `sub_area` VARCHAR(45) NOT NULL,
        `desc` LONGTEXT NULL,
        `is_rent` TINYINT NOT NULL DEFAULT 0,
        `quantity` INT NOT NULL,
        `books_rented` INT NOT NULL,
        UNIQUE INDEX `ISBN_UNIQUE` (`ISBN` ASC) VISIBLE,
```

PRIMARY KEY (`ISBN`));

Borrowed: -

```
CREATE TABLE `librarysystem`.`borrowed` (
  `member_No` INT NOT NULL,
  `isbn` INT NOT NULL,
  `issue_date` DATE NOT NULL,
  `is_returned` TINYINT NOT NULL DEFAULT 0,
  PRIMARY KEY (`Member_No`, `ISBN`, `is_returned`),
  INDEX `ISBN_FK_idx` (`ISBN` ASC) VISIBLE,
  CONSTRAINT `Member_no_FK`
  FOREIGN KEY (`Member_No`)
 REFERENCES `librarysystem`.`members` (`Member_No`)
 ON DELETE NO ACTION
 ON UPDATE NO ACTION,
 CONSTRAINT `ISBN_FK`
FOREIGN KEY (`ISBN`)
REFERENCES `librarysystem`.`books` (`ISBN`)
ON DELETE NO ACTION
ON UPDATE NO ACTION);
```

c. Triggers: -

```
DROP TRIGGER IF EXISTS `librarysystem`.`borrowed_AFTER_INSERT`;

DELIMITER $$
USE `librarysystem`$$
CREATE DEFINER=`root`@`localhost` TRIGGER `borrowed_AFTER_INSERT` AFTER INSERT ON `borrowed` FOR EACH ROW BEGIN
	SET @COUNT=(SELECT COUNT(*) FROM borrowed WHERE member_no=New.member_no);
	SET @COUNTBK=(SELECT COUNT(*) FROM borrowed WHERE ISBN=New.ISBN);
  IF @COUNT!=0 THEN
		UPDATE members set books_borrowed = @Count where member_no = New.member_no;
	end if;
	IF @COUNTBK!=0 THEN
		UPDATE books set books_rented = @Count where ISBN=New.ISBN;
		UPDATE books set Quantity = Quantity-1 where ISBN=New.ISBN;
	end if;
END$$
DELIMITER ;

DROP TRIGGER IF EXISTS `librarysystem`.`borrowed_After_UPDATE`;

DELIMITER $$
USE `librarysystem`$$
```

```
CREATE DEFINER=`root`@`localhost` TRIGGER `borrowed_After_UPDATE` AFTER Update ON `borrowed` FOR EACH ROW BEGIN

	UPDATE members set books_borrowed = books_borrowed-1 where member_no = New.member_no;
	UPDATE books set books_rented = books_rented-1 where ISBN=New.ISBN;
	UPDATE books set Quantity = Quantity+1 where ISBN=New.ISBN;

END$$
DELIMITER ;
```