Data requirements:

In library management system, the library member will have member id, name which consists of first name, last name and middle name, phone number and email.

The members borrow books where each book will have its book identification number and title. Members borrow books from library branch which have branch identification number and branch name.

The members are awarded subscription awards and the subscription award is identified by award name, award date and award identification number.

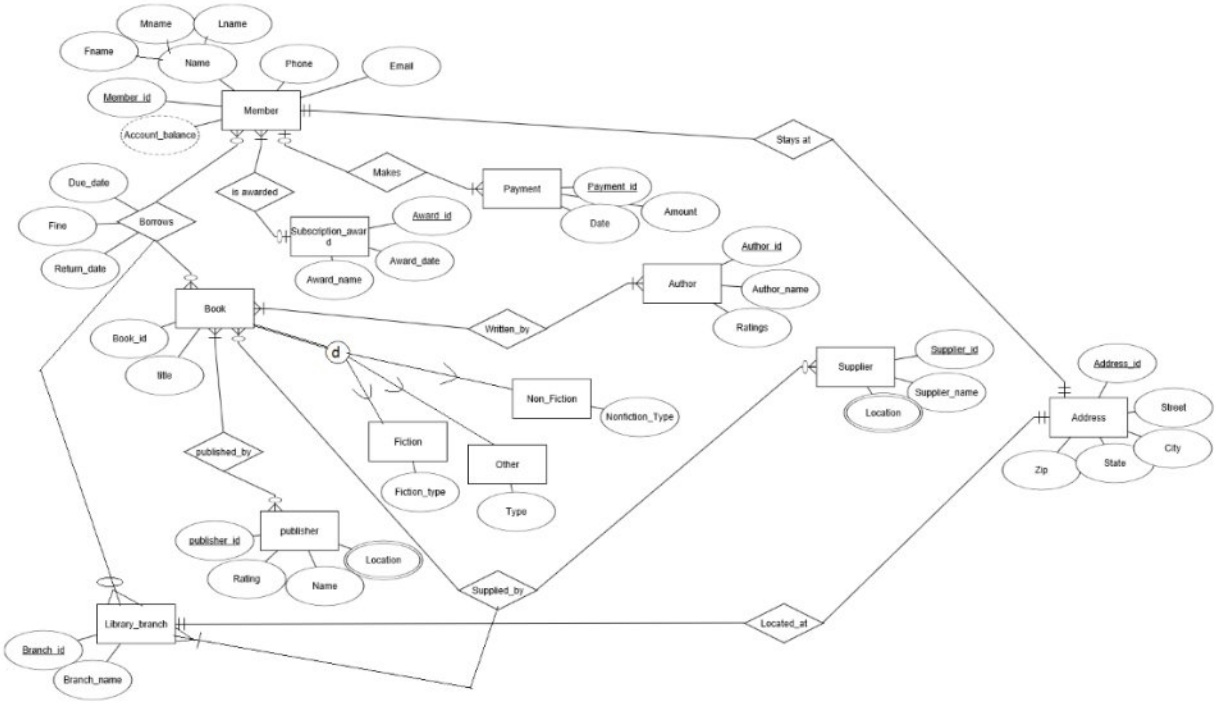
Every member must make a payment whose attributes are payment identification number, date and amount.

Every book is categorized either into fiction, non-fiction or other type and every book is written by author who is identified by author identification number. Author's attributes also include author name and ratings.

Books are published by publisher and supplied by supplier. Publisher attributes include publisher identification number, rating, name and location which is multivalued attribute. Supplier attributes include supplier identification number, supplier name and location which is multivalued.

Library branch and members are located/stays at address. Address attribute zip, state, city, street and address identification number.

ER diagram:



Mapping ER diagram into relational schema:

Id	Fname	Mname	Lname	Phone	Email	Accountbalance	Address

Member(Address-FK to PK Address table)

Address_id	Street	City	State	Zip

Address-Violates 3NF because City and State can be derived from ZIP

Publisher_id	Rating	Name	Location

Publisher-Violates 1NF because multiple locations of publishers

Book_id	Publisher_id

Book_publishers

Payment_id	Date	Amount	Member_id

Payments(FK Member_id to PK of Member table id)

Book_id	Title	Book_type	Fiction_type	Nonfiction_type	Other_type

Book

Author_id	Author_name	Ratings

Author

Book_id	Author_id

Book_authors(FK Book_id to PK book_id of Book table, FK Author_id to Author_id of Author table)

Award_id	Award_date	Award_name	Member_id

Subscription_award-Violates 2NF because award_name is derived from award_id(FK Member_id to PK of Member table id)

Supplier_id	Supplier_name	Location

Supplier-Violates 1NF because multiple locations for one supplier

Book_id	supplier_id	branch_id

Book_supplier(FK Book_id to PK id of Member table, FK Supplier_id to PK of Supplier_id of supplier table, Branch_id Fk to PK of Library_branch table Branch-id)

Branch_id	Branch_name	Address

Library_branch

Member_id	Book_id	Branch_id	Due_date	Fine	Return_date

Book_borrowers(FK Member_id to PK if od Member table, FK Book_id to PK book_id of Book table, FK Branch_id to PK Branch_id of Library_branch table)

Database normalization rules:

1. Address table violates 3NF. Therefore, it is split into two table Address and Zip.

Address id	Street	Zip

Zip	City	State

2. Publisher table violates 1NF. Therefore, primary key is now the combination of Publisher_id and location

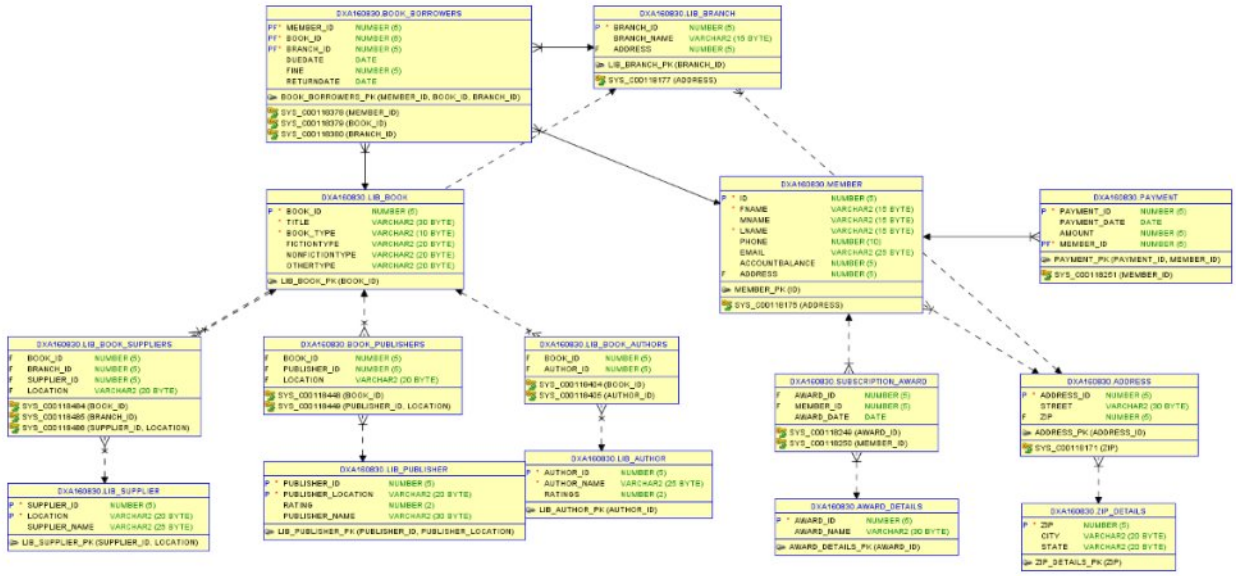
3. Supplier table violates 1NF. Therefore, primary key is now the combination of Supplier_id and location

4. Subscription_award violates 2NF. Therefore, it is split into 2 tables: subscription award and award details

Award_id	Member_id	Award_date

Award_id	Award_name

Final Relational schema:



Create table commands:

create table Zip_details (Zip number (5) primary key, City varchar (20) not null, State varchar(20) not null);

create table Address (Address_id number (5) check(Address_id>0) primary key, Street varchar (20) not null, Zip number (5) references Zip_details(Zip));

create table member(id number (5) primary key, fname varchar (15) not null, mname varchar(15), lname varchar(15) not null, Phone number(10), email varchar(20),accountbalance number(5), Address number(5) references Address(Address_id));

create table Lib_branch (Branch_id number (5) check(branch_id>0) primary key, Branch_name varchar (20) not null,Address references Address(Address_id));

create table Lib_publisher (publisher_id number (5) check(publisher_id>0) primary key, publisher_name varchar (25) not null, Location varchar(20) not null, Primary key(publisher_id, Location));

create table Lib_supplier (Supplier_id number (5) check(Supplier_id>0) primary key, Supplier_name varchar (25) not null, Location varchar(20) not null,Primary key(Supplier_id, Location));

create table award_details(award_id number(5) primary key check(award_id>0), award_name varchar(20) not null);

create table subscription_award(award_id number(5) check(award_id>0) not null, member_id number(5) check(member_id>0) not null references member(id),award_date date, primary key(award_id, member_id));

create table payment(payment_id number(5) check(payment_id>0) primary key, payment_date date not null,amount number(5) not null check(amount>0),member_id number(5) references member(id),primary key(payment_id,member_id));

create table lib_book(book_id number95) primary key check(book_id>0), title varchar(20) not null, book_type varchar(10) not null, fictiontype varchar(20), nonfictiontype varchar(20), othertype varchar(20));

create table lib_Book_publishers (Book_id number (5) references Book(book_id), Publisher_id number (5) references Lib_publisher(publisher_id), Location varchar(20) references Lib_publisher(location),Primary key(Book_id, Publihser_id, Location));

create table Lib_author (Author_id number (5) primary key, Author_name varchar (20) not null,Ratings number(2) not null);

create table lib_book_authors(book_id references book(book_id), author_id references author(author_id), primary key(book_id, author_id));

create table lib_Book_suppliers (Book_id number (5) references Book(book_id), Supplier_id number (5) references Lib_supplier(supplier_id), Location varchar(20) references Lib_supplier(location),Primary key(Book_id, supplier_id, Location));

create table book_borrowers(member_id number(5) not null check(member_id>0) references member(id), book_id number(5) not null check(book_id>0) references book(book_id), branch_id number(5) not null check(branch_id>0) references lib_branch(branch_id), duedate date not null, fine number(5) check(fine>=0), returndate date, primary key(member_id, book_id, branch_id));

Procedures:

1. Procedure for updating account balance of a given member from fine and payments:

```
CREATE OR REPLACE PROCEDURE FINE_PAY(MID BOOK_BORROWERS.MEMBER_ID%TYPE) AS
 PAY1 PAYMENT.AMOUNT%TYPE;
 PAY2 BOOK_BORROWERS.FINE%TYPE;
 CURSOR C1 IS SELECT ACCOUNTBALANCE FROM MEMBER FOR UPDATE;
 CURSOR CUR1 IS SELECT SUM(AMOUNT) FROM PAYMENT WHERE MEMBER_ID=MID;
 CURSOR CUR2 IS SELECT FINE FROM BOOK_BORROWERS WHERE MEMBER_ID=MID;
 BEGIN
 OPEN C1;
 OPEN CUR1;
 OPEN CUR2;
 LOOP
 FETCH CUR1 INTO PAY1;
 FETCH CUR2 INTO PAY2;
```

```
UPDATE MEMBER SET ACCOUNTBALANCE=PAY1-PAY2 where ID=MID;
END LOOP;
CLOSE CUR1;
CLOSE CUR2;
NULL;
END FINE_PAY;
```

2. Procedure to increase ratings of the authors of a particular type of book

```
CREATE OR REPLACE PROCEDURE FICTION_AUTHORS(BTYPE LIB_BOOK.BOOK_TYPE%TYPE,FNAME
LIB_BOOK.FICTIONTYPE%TYPE) AS
AUTHOR LIB_AUTHOR.AUTHOR_ID%TYPE;
CURSOR C2 IS SELECT RATINGS FROM LIB_AUTHOR FOR UPDATE;
CURSOR C1 IS SELECT LA.AUTHOR_ID FROM LIB_AUTHOR LA, LIB_BOOK_AUTHORS LBA, LIB_BOOK LB
WHERE LB.BOOK_ID=LBA.BOOK_ID AND
LA.AUTHOR_ID=LBA.AUTHOR_ID AND LB.BOOK_ID IN(SELECT BOOK_ID FROM LIB_BOOK WHERE
BOOK_TYPE=BTYPE AND FICTIONTYPE=FNAME);
NAME LIB_AUTHOR.AUTHOR_NAME%TYPE;
BEGIN
OPEN C1;
OPEN C2;
LOOP
FETCH C1 INTO AUTHOR;
EXIT WHEN C1%NOTFOUND;
UPDATE LIB_AUTHOR SET RATINGS=RATINGS+1 WHERE AUTHOR_ID=AUTHOR;
SELECT AUTHOR_NAME INTO NAME FROM LIB_AUTHOR WHERE AUTHOR_ID=AUTHOR;
DBMS_OUTPUT.PUT_LINE('RATING OF AUTHOR '|| NAME || ' IS INCREMENTED BY 1');
END LOOP;
CLOSE C1;
CLOSE C2;
NULL;
```

```
END FICTION_AUTHORS;
```

Triggers:

1. Update fine after insert or update on book_borrowers

```
CREATE OR REPLACE TRIGGER TRIGGER2
AFTER INSERT OR UPDATE
ON BOOK_BORROWERS
FOR EACH ROW
BEGIN
IF(:NEW.RETURNDATE-:NEW.DUEDATE>=10) THEN
UPDATE BOOK_BORROWERS SET FINE=FINE+50;
END IF;
  NULL;
END;
```

2. Trigger for updating awards based on book lending:

```
CREATE OR REPLACE TRIGGER TRIGGER3
AFTER INSERT ON BOOK_BORROWERS
FOR EACH ROW
DECLARE
COU NUMBER(5);
BEGIN
SELECT COUNT(BOOK_ID) INTO COU FROM BOOK_BORROWERS WHERE
MEMBER_ID=:NEW.MEMBER_ID;
IF(COU>=3 AND COU<=7)THEN
INSERT INTO SUBSCRIPTION_AWARD VALUES('603',:NEW.MEMBER_ID,TO_DATE(SYSDATE,'YYYY-MM-
DD'));
ELSE IF(COU>0 AND COU<3) THEN
INSERT INTO SUBSCRIPTION_AWARD VALUES('604',:NEW.MEMBER_ID,TO_DATE(SYSDATE,'YYYY-MM-
DD'));
```

```
ELSE IF(COU>7 AND COU<=10)THEN

INSERT INTO SUBSCRIPTION_AWARD VALUES('606',:NEW.MEMBER_ID,TO_DATE(SYSDATE,'YYYY-MM-DD'));

ELSE IF(COU=0)THEN

INSERT INTO SUBSCRIPTION_AWARD VALUES('605',:NEW.MEMBER_ID,TO_DATE(SYSDATE,'YYYY-MM-DD'));

ELSE

INSERT INTO SUBSCRIPTION_AWARD VALUES('607',:NEW.MEMBER_ID,TO_DATE(SYSDATE,'YYYY-MM-DD'));

END IF;

END IF;

END IF;

END IF;

END;
```

Business Rules and their Implementation:

1. All the amount attributes in the payment table must be greater than 0 but if the amount=0 then it means that payment is invalid. This was implemented in our system by putting a check constraint on amount attribute in payment table.
2. All the ID's are considered to be greater than 0 which are checked by check constraint.