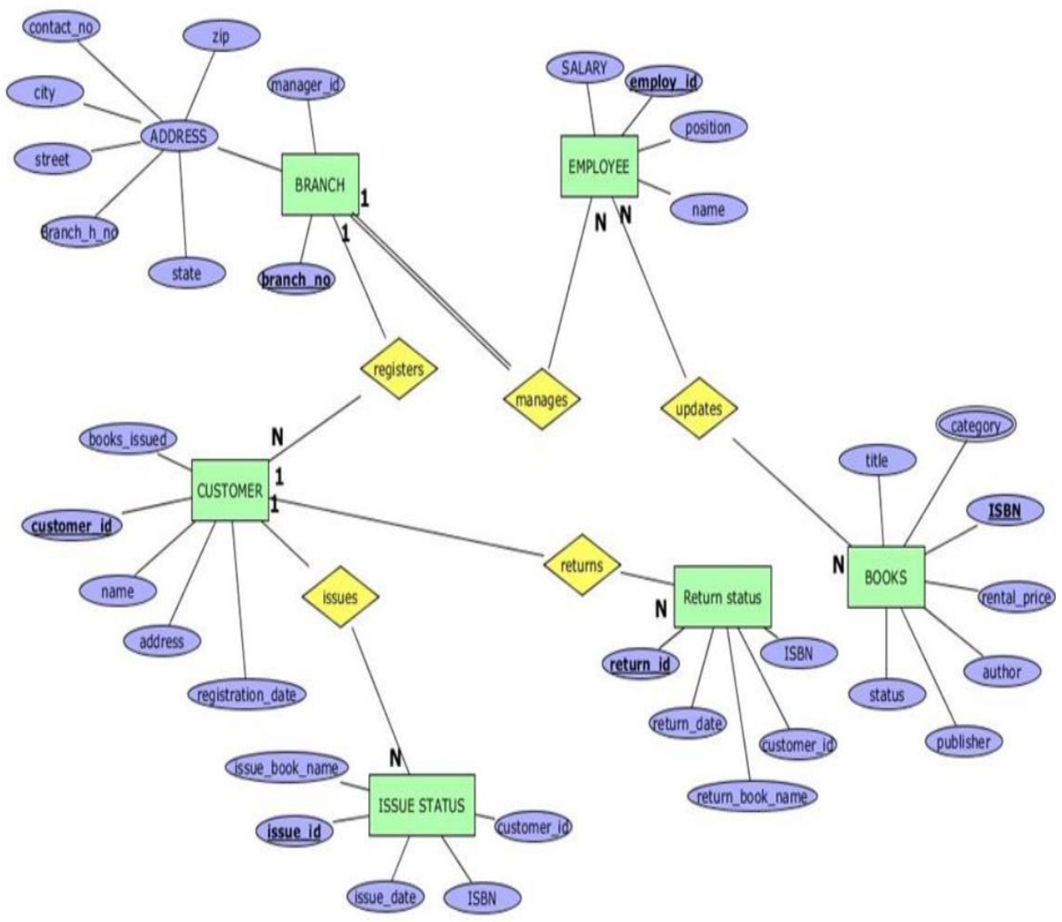
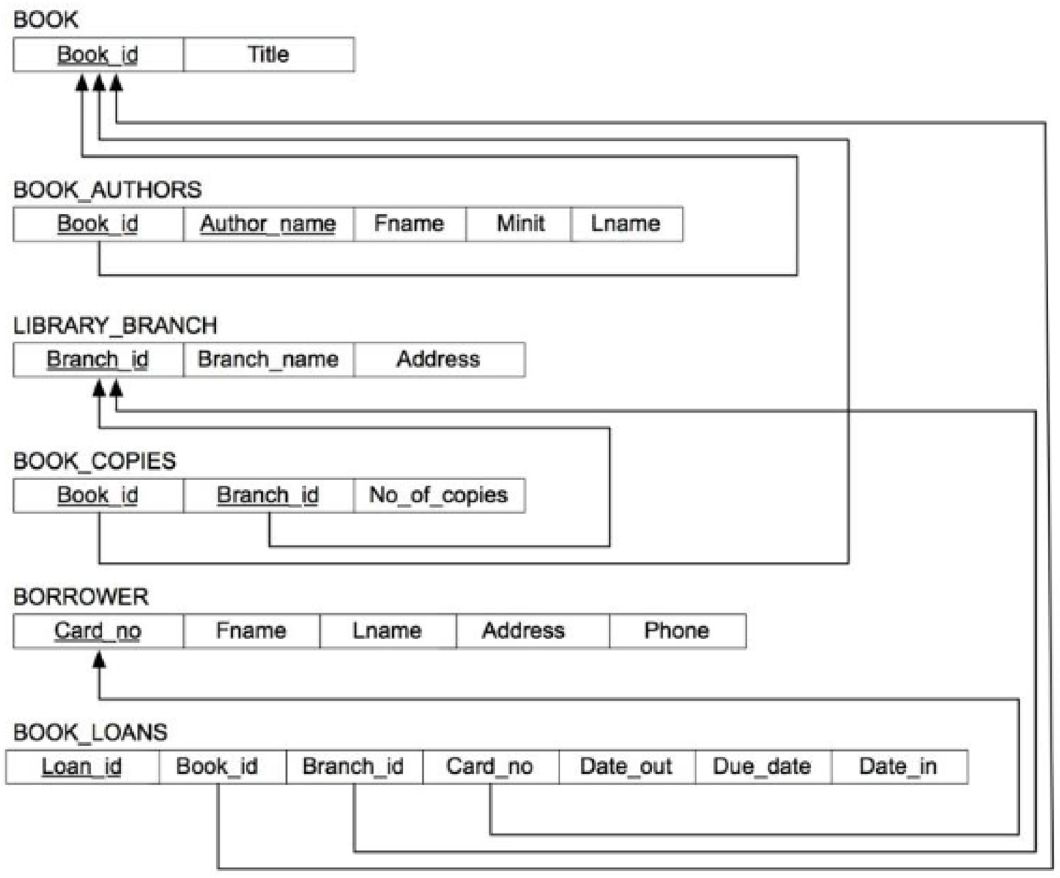
ER Model


contact_no
zip
city
ADDRESS
street
Branch_h_no
state
manager_id
BRANCH
branch_no
SALARY
employ_id
position
EMPLOYEE
name
registers
manages
updates
books_issued
CUSTOMER
customer_id
name
address
registration_date
issues
returns
Return status
return_id
return_date
customer_id
ISBN
return_book_name
category
title
ISBN
BOOKS
rental_price
author
status
publisher
issue_book_name
ISSUE STATUS
issue_id
customer_id
issue_date
ISBN
1
N

Relational Mapping



Creating Table statements

```
-- Step 1: Create ADDRESS table
CREATE TABLE ADDRESS (
  contact_no VARCHAR(15),
  city VARCHAR(50),
  street VARCHAR(100),
  branch_h_no VARCHAR(10),
  state VARCHAR(50),
  zip VARCHAR(10)
);

-- Step 2: Add a unique index on branch_h_no in ADDRESS table
CREATE UNIQUE INDEX idx_branch_h_no ON ADDRESS(branch_h_no);

-- Step 3: Create BRANCH table with foreign key constraint
CREATE TABLE BRANCH (
  branch_no INT PRIMARY KEY,
  manager_id INT,
  branch_h_no VARCHAR(10),
  FOREIGN KEY (branch_h_no) REFERENCES ADDRESS(branch_h_no)
);

-- Step 4: Create the remaining tables
CREATE TABLE EMPLOYEE (
  employ_id INT PRIMARY KEY,
  name VARCHAR(100),
  position VARCHAR(50),
  SALARY DECIMAL(10, 2)
);
```

```
CREATE TABLE CUSTOMER (

  customer_id INT PRIMARY KEY,

  name VARCHAR(100),

  address VARCHAR(200),

  registration_date DATE,

  books_issued INT

);


CREATE TABLE BOOKS (

  ISBN VARCHAR(13) PRIMARY KEY,

  title VARCHAR(200),

  author VARCHAR(100),

  publisher VARCHAR(100),

  category VARCHAR(50),

  rental_price DECIMAL(10, 2),

  status VARCHAR(20)

);


CREATE TABLE ISSUE_STATUS (

  issue_id INT PRIMARY KEY,

  issue_date DATE,

  issue_book_name VARCHAR(200),

  customer_id INT,

  ISBN VARCHAR(13),

  FOREIGN KEY (customer_id) REFERENCES CUSTOMER(customer_id),

  FOREIGN KEY (ISBN) REFERENCES BOOKS(ISBN)

);


CREATE TABLE RETURN_STATUS (

  return_id INT PRIMARY KEY,

  return_date DATE,
  return_book_name VARCHAR(200),
  customer_id INT,
  ISBN VARCHAR(13),
  FOREIGN KEY (customer_id) REFERENCES CUSTOMER(customer_id),
  FOREIGN KEY (ISBN) REFERENCES BOOKS(ISBN)
);

-- Relationship Tables

CREATE TABLE REGISTERS (
  customer_id INT,
  branch_no INT,
  PRIMARY KEY (customer_id, branch_no),
  FOREIGN KEY (customer_id) REFERENCES CUSTOMER(customer_id),
  FOREIGN KEY (branch_no) REFERENCES BRANCH(branch_no)
);

CREATE TABLE MANAGES (
  employ_id INT,
  branch_no INT,
  PRIMARY KEY (employ_id, branch_no),
  FOREIGN KEY (employ_id) REFERENCES EMPLOYEE(employ_id),
  FOREIGN KEY (branch_no) REFERENCES BRANCH(branch_no)
);

CREATE TABLE UPDATES (
  employ_id INT,
  ISBN VARCHAR(13),
  PRIMARY KEY (employ_id, ISBN),
  FOREIGN KEY (employ_id) REFERENCES EMPLOYEE(employ_id),
  FOREIGN KEY (ISBN) REFERENCES BOOKS(ISBN)
```

```
);

CREATE TABLE ISSUES (
   customer_id INT,
   issue_id INT,
   PRIMARY KEY (customer_id, issue_id),
   FOREIGN KEY (customer_id) REFERENCES CUSTOMER(customer_id),
   FOREIGN KEY (issue_id) REFERENCES ISSUE_STATUS(issue_id)
);

CREATE TABLE RETURNS (
   customer_id INT,
   return_id INT,
   PRIMARY KEY (customer_id, return_id),
   FOREIGN KEY (customer_id) REFERENCES CUSTOMER(customer_id),
   FOREIGN KEY (return_id) REFERENCES RETURN_STATUS(return_id)
);
```

Insert values statements

```
INSERT INTO ADDRESS (contact_no, city, street, branch_h_no, state, zip) VALUES
('1234567890', 'New York', '5th Avenue', 'A1', 'NY', '10001'),
('0987654321', 'Los Angeles', 'Sunset Blvd', 'B2', 'CA', '90001'),
('5555555555', 'Chicago', 'Michigan Ave', 'C3', 'IL', '60601'),
('4444444444', 'Houston', 'Main St', 'D4', 'TX', '77001'),
('3333333333', 'Phoenix', 'Central Ave', 'E5', 'AZ', '85001');

INSERT INTO BRANCH (branch_no, manager_id, branch_h_no) VALUES
(1, 101, 'A1'),
(2, 102, 'B2'),
(3, 103, 'C3'),
(4, 104, 'D4'),
(5, 105, 'E5');

INSERT INTO EMPLOYEE (employ_id, name, position, salary) VALUES
(101, 'Alice Johnson', 'Manager', 75000.00),
(102, 'Bob Smith', 'Assistant Manager', 65000.00),
(103, 'Charlie Brown', 'Manager', 80000.00),
(104, 'Diana Ross', 'Assistant Manager', 70000.00),
(105, 'Edward Johnson', 'Manager', 85000.00);

INSERT INTO CUSTOMER (customer_id, name, address, registration_date, books_issued)
VALUES
(201, 'David Miller', '123 Maple St, New York, NY', '2022-01-01', 3),
(202, 'Eve Adams', '456 Oak St, Los Angeles, CA', '2022-02-15', 1),
(203, 'Frank White', '789 Pine St, Chicago, IL', '2022-03-20', 5),
```

```sql
(204, 'Grace Green', '101 Palm St, Houston, TX', '2022-04-10', 2),
(205, 'Hank Black', '202 Birch St, Phoenix, AZ', '2022-05-25', 4);

INSERT INTO BOOKS (ISBN, title, author, publisher, category, rental_price, status) VALUES
('9783161484100', 'Book Title 1', 'Author A', 'Publisher X', 'Fiction', 9.99, 'Available'),
('9781402894626', 'Book Title 2', 'Author B', 'Publisher Y', 'Non-Fiction', 14.99, 'Issued'),
('9780545010221', 'Book Title 3', 'Author C', 'Publisher Z', 'Science', 19.99, 'Available'),
('9780141034354', 'Book Title 4', 'Author D', 'Publisher W', 'History', 12.99, 'Available'),
('9781250076536', 'Book Title 5', 'Author E', 'Publisher V', 'Biography', 10.99, 'Issued');

INSERT INTO ISSUE_STATUS (issue_id, issue_date, issue_book_name, customer_id, ISBN)
VALUES
(301, '2023-01-10', 'Book Title 2', 201, '9781402894626'),
(302, '2023-02-20', 'Book Title 1', 202, '9783161484100'),
(303, '2023-03-15', 'Book Title 3', 203, '9780545010221'),
(304, '2023-04-05', 'Book Title 4', 204, '9780141034354'),
(305, '2023-05-25', 'Book Title 5', 205, '9781250076536');

INSERT INTO RETURN_STATUS (return_id, return_date, return_book_name, customer_id, ISBN)
VALUES
(401, '2023-01-20', 'Book Title 2', 201, '9781402894626'),
(402, '2023-02-28', 'Book Title 1', 202, '9783161484100'),
(403, '2023-03-25', 'Book Title 3', 203, '9780545010221'),
(404, '2023-04-15', 'Book Title 4', 204, '9780141034354'),
(405, '2023-06-05', 'Book Title 5', 205, '9781250076536');

INSERT INTO REGISTERS (customer_id, branch_no) VALUES
(201, 1),
(202, 2),
(203, 3),
(204, 4),
(205, 5);


INSERT INTO MANAGES (employ_id, branch_no) VALUES
(101, 1),
(102, 2),
(103, 3),
(104, 4),
(105, 5);


INSERT INTO UPDATES (employ_id, ISBN) VALUES
(101, '9783161484100'),
(102, '9781402894626'),
(103, '9780545010221'),
(104, '9780141034354'),
(105, '9781250076536');


INSERT INTO ISSUES (customer_id, issue_id) VALUES
(201, 301),
(202, 302),
(203, 303),
(204, 304),
(205, 305);


INSERT INTO RETURNS (customer_id, return_id) VALUES
```

```
(201, 401),

(202, 402),

(203, 403),

(204, 404),

(205, 405);
```

A Stored procedure to retrieve the details of employee and customer

```
DELIMITER //

CREATE PROCEDURE GetEmployeeAndCustomerDetails()
BEGIN
   -- Declare variables to hold employee details
   DECLARE v_employ_id INT;
   DECLARE v_name VARCHAR(100);
   DECLARE v_position VARCHAR(50);
   DECLARE v_salary DECIMAL(10, 2);

   -- Declare variables to hold customer details
   DECLARE v_customer_id INT;
   DECLARE v_customer_name VARCHAR(100);
   DECLARE v_address VARCHAR(200);
   DECLARE v_registration_date DATE;
   DECLARE v_books_issued INT;

   -- Declare cursors for employee and customer
   DECLARE employee_cursor CURSOR FOR SELECT employ_id, name, position, salary FROM
EMPLOYEE;
   DECLARE customer_cursor CURSOR FOR SELECT customer_id, name, address,
registration_date, books_issued FROM CUSTOMER;

   -- Declare 'not found' handlers
   DECLARE CONTINUE HANDLER FOR NOT FOUND SET @not_found = 1;

   -- Process employee details
   OPEN employee_cursor;
```

```
  employee_loop: LOOP
    FETCH employee_cursor INTO v_employ_id, v_name, v_position, v_salary;
    IF @not_found THEN
      SET @not_found = 0;
      LEAVE employee_loop;
    END IF;
    -- Here you can process each employee record
    SELECT v_employ_id, v_name, v_position, v_salary;
  END LOOP;
  CLOSE employee_cursor;

  -- Process customer details
  OPEN customer_cursor;
  customer_loop: LOOP
    FETCH customer_cursor INTO v_customer_id, v_customer_name, v_address,
v_registration_date, v_books_issued;
    IF @not_found THEN
      SET @not_found = 0;
      LEAVE customer_loop;
    END IF;
    -- Here you can process each customer record
    SELECT v_customer_id, v_customer_name, v_address, v_registration_date, v_books_issued;
  END LOOP;
  CLOSE customer_cursor;
END //

DELIMITER ;
```

A warning trigger to alert such that customer cannot lend more than 5 books per month

```
DELIMITER //

CREATE TRIGGER CheckBookLimitBeforeInsert
BEFORE INSERT ON ISSUE_STATUS
FOR EACH ROW
BEGIN
  DECLARE book_count INT;

  -- Calculate the number of books issued to the customer in the current month
  SELECT COUNT(*)
  INTO book_count
  FROM ISSUE_STATUS
  WHERE customer_id = NEW.customer_id
   AND MONTH(issue_date) = MONTH(NEW.issue_date)
   AND YEAR(issue_date) = YEAR(NEW.issue_date);

  -- If the count is 5 or more, signal an error
  IF book_count >= 5 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Customer cannot borrow more than 5 books in a month';
  END IF;
END //

DELIMITER ;
```

Lending Fee Calculation Query:

To calculate the lending fee based on the rental price of the book and the duration it is issued:

```
SELECT
    IS.issue_id,
    IS.ISBN,
    IS.issue_date,
    IS.customer_id,
    B.rental_price,
    DATEDIFF(RS.return_date, IS.issue_date) AS days_issued,
    (B.rental_price * DATEDIFF(RS.return_date, IS.issue_date)) AS lending_fee
FROM
    ISSUE_STATUS IS
JOIN
    BOOKS B ON IS.ISBN = B.ISBN
JOIN
    RETURN_STATUS RS ON IS.issue_id = RS.issue_id;
```

Fine Calculation Query:

To calculate fines based on the return date and a fixed fine rate per day overdue:

```
SELECT

   RS.return_id,

   IS.ISBN,

   IS.issue_date,

   RS.return_date,

   (CASE WHEN RS.return_date > DATE_ADD(IS.issue_date, INTERVAL 30 DAY)

         THEN DATEDIFF(RS.return_date, DATE_ADD(IS.issue_date, INTERVAL 30
DAY))

         ELSE 0

    END) AS days_overdue,

   (CASE WHEN RS.return_date > DATE_ADD(IS.issue_date, INTERVAL 30 DAY)

         THEN DATEDIFF(RS.return_date, DATE_ADD(IS.issue_date, INTERVAL 30
DAY)) * 1.00 -- Fine rate $1 per day

         ELSE 0.00

    END) AS fine_amount

FROM

   ISSUE_STATUS IS

JOIN

   RETURN_STATUS RS ON IS.issue_id = RS.issue_id;
```

Automated Employee Salary Increment Query:

```
UPDATE EMPLOYEE

SET salary = salary * 1.05 -- Increase salary by 5%

WHERE years_of_service >= 5; -- Increment salary for employees with 5 or more years of
service
```