Library Management System Final

Library Management System - Database Project

Project Overview

This document contains all the required components for the Database Project (Spring 2025). The project implements a Library Management System with four entities (Books, Members, Loans, and Staff) and demonstrates various database concepts including ERD design, table creation, data insertion, SQL queries, and advanced SQL features.

Table of Contents

Entity-Relationship Diagram

The Library Management System consists of four main entities with the following relationships: - Books (1) to Loans (N): One-to-Many - Members (1) to Loans (N): One-to-Many - Staff (1) to Loans (N): One-to-Many

Entities and Attributes

1. Books

- **book_id** (Primary Key): Unique identifier for each book
- **isbn**: International Standard Book Number

- **title**: Title of the book
- **author**: Author of the book
- **publisher**: Publisher of the book
- **publication_year**: Year the book was published
- **category**: Category/genre of the book
- **total_copies**: Total number of copies owned by the library
- **available_copies**: Number of copies currently available for borrowing

2. Members

- **member_id** (Primary Key): Unique identifier for each member
- **first_name**: Member's first name
- **last_name**: Member's last name
- **email**: Member's email address
- **phone**: Member's phone number
- **address**: Member's physical address
- **join_date**: Date when the member joined the library
- **membership_status**: Current status of membership (active, expired, suspended)

3. Loans

- **loan_id** (Primary Key): Unique identifier for each loan
- **book_id** (Foreign Key): Reference to the borrowed book
- **member_id** (Foreign Key): Reference to the member who borrowed the book
- **staff_id** (Foreign Key): Reference to the staff who processed the loan
- **loan_date**: Date when the book was borrowed
- **due_date**: Date when the book is due to be returned
- **return_date**: Actual date when the book was returned (NULL if not yet returned)
- **fine_amount**: Amount of fine for late returns (if applicable)

4. Staff

- **staff_id** (Primary Key): Unique identifier for each staff member
- **first_name**: Staff's first name
- **last_name**: Staff's last name
- **position**: Staff's position/role in the library
- **email**: Staff's email address
- **phone**: Staff's phone number
- **hire_date**: Date when the staff was hired

ERD Representation

```
+------------+      +------------+      +------------+
|   Books    |      |   Loans    |      |  Members   |
+------------+       +------------+       +------------+
| PK book_id |<----->| FK book_id |       | PK member_id|
|    isbn    |       | FK member_id|<---->|  first_name |
|    title   |  1:N  | FK staff_id |       |  last_name  |
|   author   |       |  loan_date  |       |    email    |
| publisher  |       |  due_date   |  N:1  |    phone    |
| pub_year   |       | return_date |       |   address   |
|  category  |       | fine_amount |       |  join_date  |
|total_copies|       +------------+       |member_status|
|avail_copies|                            +------------+
+------------+                                  ^
                                                |
                                                | N:1
                                                |
                                          +------------+
                                          |   Staff    |
                                          +------------+
                                          | PK staff_id |
                                          |  first_name |
                                          |  last_name  |
                                          |  position   |
                                          |    email    |
                                          |    phone    |
                                          |  hire_date  |
                                          +------------+
```

Oracle Database Tables

The following SQL script creates the tables for the Library Management System:

```
-- Create Books table
CREATE TABLE Books (
    book_id NUMBER PRIMARY KEY,
    isbn VARCHAR2(20) UNIQUE,
    title VARCHAR2(100) NOT NULL,
    author VARCHAR2(100) NOT NULL,
    publisher VARCHAR2(100),
    publication_year NUMBER(4),
    category VARCHAR2(50),
    total_copies NUMBER DEFAULT 0,
    available_copies NUMBER DEFAULT 0,
    CONSTRAINT check_copies CHECK (available_copies <= total_copies),
    CONSTRAINT check_pub_year CHECK (publication_year > 0)
);
```

```
-- Create Members table
CREATE TABLE Members (
    member_id NUMBER PRIMARY KEY,
    first_name VARCHAR2(50) NOT NULL,
    last_name VARCHAR2(50) NOT NULL,
    email VARCHAR2(100) UNIQUE,
    phone VARCHAR2(20),
    address VARCHAR2(200),
    join_date DATE DEFAULT SYSDATE,
    membership_status VARCHAR2(20) DEFAULT 'active',
    CONSTRAINT check_status CHECK (membership_status IN ('active', 'expired',
'suspended'))
);

-- Create Staff table
CREATE TABLE Staff (
    staff_id NUMBER PRIMARY KEY,
    first_name VARCHAR2(50) NOT NULL,
    last_name VARCHAR2(50) NOT NULL,
    position VARCHAR2(50) NOT NULL,
    email VARCHAR2(100) UNIQUE,
    phone VARCHAR2(20),
    hire_date DATE DEFAULT SYSDATE
);

-- Create Loans table
CREATE TABLE Loans (
    loan_id NUMBER PRIMARY KEY,
    book_id NUMBER NOT NULL,
    member_id NUMBER NOT NULL,
    staff_id NUMBER NOT NULL,
    loan_date DATE DEFAULT SYSDATE,
    due_date DATE NOT NULL,
    return_date DATE,
    fine_amount NUMBER(10,2) DEFAULT 0,
    CONSTRAINT fk_book FOREIGN KEY (book_id) REFERENCES Books(book_id),
    CONSTRAINT fk_member FOREIGN KEY (member_id) REFERENCES
Members(member_id),
    CONSTRAINT fk_staff FOREIGN KEY (staff_id) REFERENCES Staff(staff_id),
    CONSTRAINT check_dates CHECK (return_date IS NULL OR return_date >=
loan_date),
    CONSTRAINT check_due_date CHECK (due_date >= loan_date),
    CONSTRAINT check_fine CHECK (fine_amount >= 0)
);
```

Sample Data

The following SQL script inserts sample data into the tables:

```
-- Insert sample data into Books table
INSERT INTO Books (isbn, title, author, publisher, publication_year, category,
total_copies, available_copies)
VALUES ('978-0451524935', 'Nineteen Eighty-Four', 'George Orwell', 'Signet
Classics', 1949, 'Fiction', 10, 8);

INSERT INTO Books (isbn, title, author, publisher, publication_year, category,
total_copies, available_copies)
VALUES ('978-0061120084', 'To Kill a Mockingbird', 'Harper Lee', 'Harper
Perennial', 1960, 'Fiction', 15, 12);

INSERT INTO Books (isbn, title, author, publisher, publication_year, category,
total_copies, available_copies)
VALUES ('978-0307474278', 'The Da Vinci Code', 'Dan Brown', 'Anchor', 2003,
'Mystery', 8, 5);

INSERT INTO Books (isbn, title, author, publisher, publication_year, category,
total_copies, available_copies)
VALUES ('978-0547928227', 'The Hobbit', 'J.R.R. Tolkien', 'Houghton Mifflin',
1937, 'Fantasy', 12, 10);

INSERT INTO Books (isbn, title, author, publisher, publication_year, category,
total_copies, available_copies)
VALUES ('978-0553593716', 'A Game of Thrones', 'George R.R. Martin', 'Bantam',
1996, 'Fantasy', 20, 15);

-- Insert sample data into Members table
INSERT INTO Members (first_name, last_name, email, phone, address, join_date,
membership_status)
VALUES ('John', 'Smith', 'john.smith@email.com', '555-123-4567', '123 Main St,
Anytown', TO_DATE('2023-01-15', 'YYYY-MM-DD'), 'active');

INSERT INTO Members (first_name, last_name, email, phone, address, join_date,
membership_status)
VALUES ('Sarah', 'Johnson', 'sarah.j@email.com', '555-234-5678', '456 Oak Ave,
Somewhere', TO_DATE('2023-03-22', 'YYYY-MM-DD'), 'active');

INSERT INTO Members (first_name, last_name, email, phone, address, join_date,
membership_status)
VALUES ('Michael', 'Williams', 'michael.w@email.com', '555-345-6789', '789
Pine Rd, Nowhere', TO_DATE('2022-11-05', 'YYYY-MM-DD'), 'suspended');
```

```
INSERT INTO Members (first_name, last_name, email, phone, address, join_date,
membership_status)
VALUES ('Emily', 'Brown', 'emily.b@email.com', '555-456-7890', '101 Cedar Ln,
Elsewhere', TO_DATE('2023-05-10', 'YYYY-MM-DD'), 'active');

INSERT INTO Members (first_name, last_name, email, phone, address, join_date,
membership_status)
VALUES ('David', 'Jones', 'david.j@email.com', '555-567-8901', '202 Maple Dr,
Anywhere', TO_DATE('2022-08-30', 'YYYY-MM-DD'), 'expired');

-- Insert sample data into Staff table
INSERT INTO Staff (first_name, last_name, position, email, phone, hire_date)
VALUES ('Robert', 'Anderson', 'Head Librarian', 'robert.a@library.com',
'555-678-9012', TO_DATE('2020-06-15', 'YYYY-MM-DD'));

INSERT INTO Staff (first_name, last_name, position, email, phone, hire_date)
VALUES ('Jennifer', 'Martinez', 'Librarian', 'jennifer.m@library.com',
'555-789-0123', TO_DATE('2021-03-10', 'YYYY-MM-DD'));

INSERT INTO Staff (first_name, last_name, position, email, phone, hire_date)
VALUES ('Thomas', 'Taylor', 'Assistant Librarian', 'thomas.t@library.com',
'555-890-1234', TO_DATE('2022-01-20', 'YYYY-MM-DD'));

INSERT INTO Staff (first_name, last_name, position, email, phone, hire_date)
VALUES ('Lisa', 'Garcia', 'IT Specialist', 'lisa.g@library.com',
'555-901-2345', TO_DATE('2021-09-05', 'YYYY-MM-DD'));

INSERT INTO Staff (first_name, last_name, position, email, phone, hire_date)
VALUES ('Kevin', 'Wilson', 'Library Assistant', 'kevin.w@library.com',
'555-012-3456', TO_DATE('2022-07-12', 'YYYY-MM-DD'));

-- Insert sample data into Loans table
INSERT INTO Loans (book_id, member_id, staff_id, loan_date, due_date,
return_date, fine_amount)
VALUES (1, 1, 1, TO_DATE('2023-06-01', 'YYYY-MM-DD'), TO_DATE('2023-06-15',
'YYYY-MM-DD'), TO_DATE('2023-06-14', 'YYYY-MM-DD'), 0);

INSERT INTO Loans (book_id, member_id, staff_id, loan_date, due_date,
return_date, fine_amount)
VALUES (2, 2, 2, TO_DATE('2023-06-05', 'YYYY-MM-DD'), TO_DATE('2023-06-19',
'YYYY-MM-DD'), TO_DATE('2023-06-25', 'YYYY-MM-DD'), 3.00);

INSERT INTO Loans (book_id, member_id, staff_id, loan_date, due_date,
return_date, fine_amount)
VALUES (3, 3, 1, TO_DATE('2023-06-10', 'YYYY-MM-DD'), TO_DATE('2023-06-24',
'YYYY-MM-DD'), NULL, 0);
```

```
INSERT INTO Loans (book_id, member_id, staff_id, loan_date, due_date,
return_date, fine_amount)
VALUES (4, 4, 3, TO_DATE('2023-06-12', 'YYYY-MM-DD'), TO_DATE('2023-06-26',
'YYYY-MM-DD'), TO_DATE('2023-06-20', 'YYYY-MM-DD'), 0);

INSERT INTO Loans (book_id, member_id, staff_id, loan_date, due_date,
return_date, fine_amount)
VALUES (5, 5, 2, TO_DATE('2023-06-15', 'YYYY-MM-DD'), TO_DATE('2023-06-29',
'YYYY-MM-DD'), NULL, 0);
```

SQL Queries

The following SQL queries demonstrate various SQL clauses and concepts:

Query 1: List all books with their availability status (WHERE clause)

```
-- Description: This query retrieves all books in the library along with their
title, author,
-- and availability status, filtered to show only books with at least 5
available copies.
SELECT
    book_id,
    title,
    author,
    category,
    total_copies,
    available_copies,
    CASE
        WHEN available_copies = 0 THEN 'Not Available'
        WHEN available_copies < 5 THEN 'Limited Availability'
        ELSE 'Available'
    END AS availability_status
FROM
    Books
WHERE
    available_copies >= 5
ORDER BY
    title;
```

Query 2: Find all active members and their current loans (INNER JOIN)

```
-- Description: This query uses an INNER JOIN to retrieve all active members
and the books
-- they currently have on loan (where return_date is NULL).
SELECT
    m.member_id,
    m.first_name || ' ' || m.last_name AS member_name,
    b.title AS book_title,
    l.loan_date,
    l.due_date,
    SYSDATE - l.due_date AS days_overdue
FROM
    Members m
INNER JOIN
    Loans l ON m.member_id = l.member_id
INNER JOIN
    Books b ON l.book_id = b.book_id
WHERE
    m.membership_status = 'active'
    AND l.return_date IS NULL
ORDER BY
    m.last_name, m.first_name;
```

Query 3: List all books that have never been borrowed (LEFT JOIN)

```
-- Description: This query uses a LEFT JOIN to find books that have never been
borrowed
-- (no matching records in the Loans table).
SELECT
    b.book_id,
    b.title,
    b.author,
    b.category
FROM
    Books b
LEFT JOIN
    Loans l ON b.book_id = l.book_id
WHERE
    l.loan_id IS NULL
ORDER BY
    b.title;
```

Query 4: Find the number of loans processed by each staff member (GROUP BY with HAVING)

```
-- Description: This query uses GROUP BY with HAVING to count the number of loans processed
-- by each staff member, showing only those who have processed more than 1 loan.
SELECT
    s.staff_id,
    s.first_name || ' ' || s.last_name AS staff_name,
    s.position,
    COUNT(l.loan_id) AS loans_processed
FROM
    Staff s
JOIN
    Loans l ON s.staff_id = l.staff_id
GROUP BY
    s.staff_id, s.first_name, s.last_name, s.position
HAVING
    COUNT(l.loan_id) > 1
ORDER BY
    loans_processed DESC;
```

Query 5: Find members who have borrowed books from the same category more than once (Subquery)

```
-- Description: This query uses a subquery to identify members who have borrowed books
-- from the same category multiple times, showing their reading preferences.
SELECT
    m.member_id,
    m.first_name || ' ' || m.last_name AS member_name,
    b.category,
    COUNT(*) AS category_borrow_count
FROM
    Members m
JOIN
    Loans l ON m.member_id = l.member_id
JOIN
    Books b ON l.book_id = b.book_id
GROUP BY
    m.member_id, m.first_name, m.last_name, b.category
HAVING
    COUNT(*) > 1
```

```
ORDER BY
    m.member_id, category_borrow_count DESC;
```

Advanced SQL Features

The following SQL script demonstrates advanced SQL features including views, sequences, functions, and procedures:

View: CurrentLoans

```
-- Create a view to show all currently borrowed books with borrower
information
CREATE OR REPLACE VIEW CurrentLoans AS
SELECT
    l.loan_id,
    b.title AS book_title,
    b.author AS book_author,
    m.first_name || ' ' || m.last_name AS borrower_name,
    m.email AS borrower_email,
    m.phone AS borrower_phone,
    l.loan_date,
    l.due_date,
    CASE
        WHEN SYSDATE > l.due_date THEN 'Overdue'
        WHEN SYSDATE > l.due_date - 2 THEN 'Due Soon'
        ELSE 'On Time'
    END AS status,
    ROUND(SYSDATE - l.due_date) AS days_overdue
FROM
    Loans l
JOIN
    Books b ON l.book_id = b.book_id
JOIN
    Members m ON l.member_id = m.member_id
WHERE
    l.return_date IS NULL
ORDER BY
    status DESC, days_overdue DESC;
```

Sequence: invoice_seq

```
-- Create a sequence for generating invoice numbers for fine payments
CREATE SEQUENCE invoice_seq
    START WITH 1000
    INCREMENT BY 1
    NOCACHE
    NOCYCLE;
```

Function: calculate_fine

```
-- Create a function to calculate fine for overdue books
CREATE OR REPLACE FUNCTION calculate_fine(
    p_loan_id IN NUMBER
) RETURN NUMBER IS
    v_due_date DATE;
    v_return_date DATE;
    v_days_overdue NUMBER;
    v_fine_amount NUMBER := 0;
    v_daily_rate NUMBER := 0.50; -- $0.50 per day
BEGIN
    -- Get the due date and return date for the loan
    SELECT due_date, return_date
    INTO v_due_date, v_return_date
    FROM Loans
    WHERE loan_id = p_loan_id;

    -- If the book has been returned
    IF v_return_date IS NOT NULL THEN
        -- Calculate days overdue
        v_days_overdue := v_return_date - v_due_date;

        -- Calculate fine if overdue
        IF v_days_overdue > 0 THEN
            v_fine_amount := v_days_overdue * v_daily_rate;
        END IF;
    -- If the book has not been returned yet
    ELSE
        -- Calculate days overdue based on current date
        v_days_overdue := SYSDATE - v_due_date;

        -- Calculate fine if overdue
        IF v_days_overdue > 0 THEN
            v_fine_amount := v_days_overdue * v_daily_rate;
        END IF;
    END IF;

    RETURN v_fine_amount;
END;
```