DBMS PROJECT

LIBRARY MANAGEMENT SYSTEM

A PROJECT REPORT

ON

LIBRARY MANAGEMENT SYSTEM

COURSE: DATABASE MANAGEMENT SYSTEM (MCA2201)

UNDER GUIDANCE OF: PROF. ANJALI G JIVANI & PROF. MUKTI PATEL

SUBMITTED BY:

NAME	SEAT NO	PRN
DEV CHAUHAN	404	8022020805
SHUBHAM CHAUHAN	405	8022053280
ROHANSH MEHTA	410	8022054241
SINGH SHASHANK	421	8022053366

INDEX

SCOPE

A library management system is a software application that is designed to manage the operations of a library. It helps in managing the resources of the library. The system allows the librarian to keep track of all the books in the library, the members who have borrowed the books, the due dates, and the fines that are to be paid.

The above library management system consists of four tables: Member, Book, Active_records, and records_history. The Member table stores the details of the members such as member number, name, number of books issued, and total fine. The Book table stores the details of the books such as book number, name, author, price, and the number of copies available. The Transaction table stores the details of the transaction such as the book number, member number, issue date, due date, and return date. The records_history table stores the details of the complete book transaction history such as the book number, member number, issue date, due date, and return date.

The above system is better than manual record keeping because it offers several advantages. Some of the advantages are:

1. Automation: The library management system automates the process of issuing and returning books. It eliminates the need for manual record keeping, which is time-consuming and prone to errors. With the library management system, the librarian can issue and return books quickly and accurately.

2. Easy Access: The library management system provides easy access to information about the books, members, and transactions. The librarian can easily search for a particular book or member and retrieve the information quickly.

3. Real-time information: The library management system provides real-time information about the books that are available, the books that have been issued, and the books that are overdue. The librarian can use this information to make informed decisions about purchasing new books or to send reminders to members who have overdue books.

4. Better Reporting: The library management system generates reports that provide insights into the operations of the library. The reports provide information about the number of books issued, the number of books returned, the number of overdue books, and the fines that have been collected. The librarian can use this information to make informed decisions about the operations of the library.

In conclusion, the above library management system offers several advantages over manual record keeping. It provides automation, easy access, real-time information and better reporting. These features make the library management system an essential tool for managing the operations of a library efficiently.

NORMALISED TABLES

MEMBER	
COLUMN	CONSTRAINT
MEMBER_ID	PK
M_NAME	NOT NULL
M_TYPE	NOT NULL
NO_OF_BOOKS	
TOT_FINE	

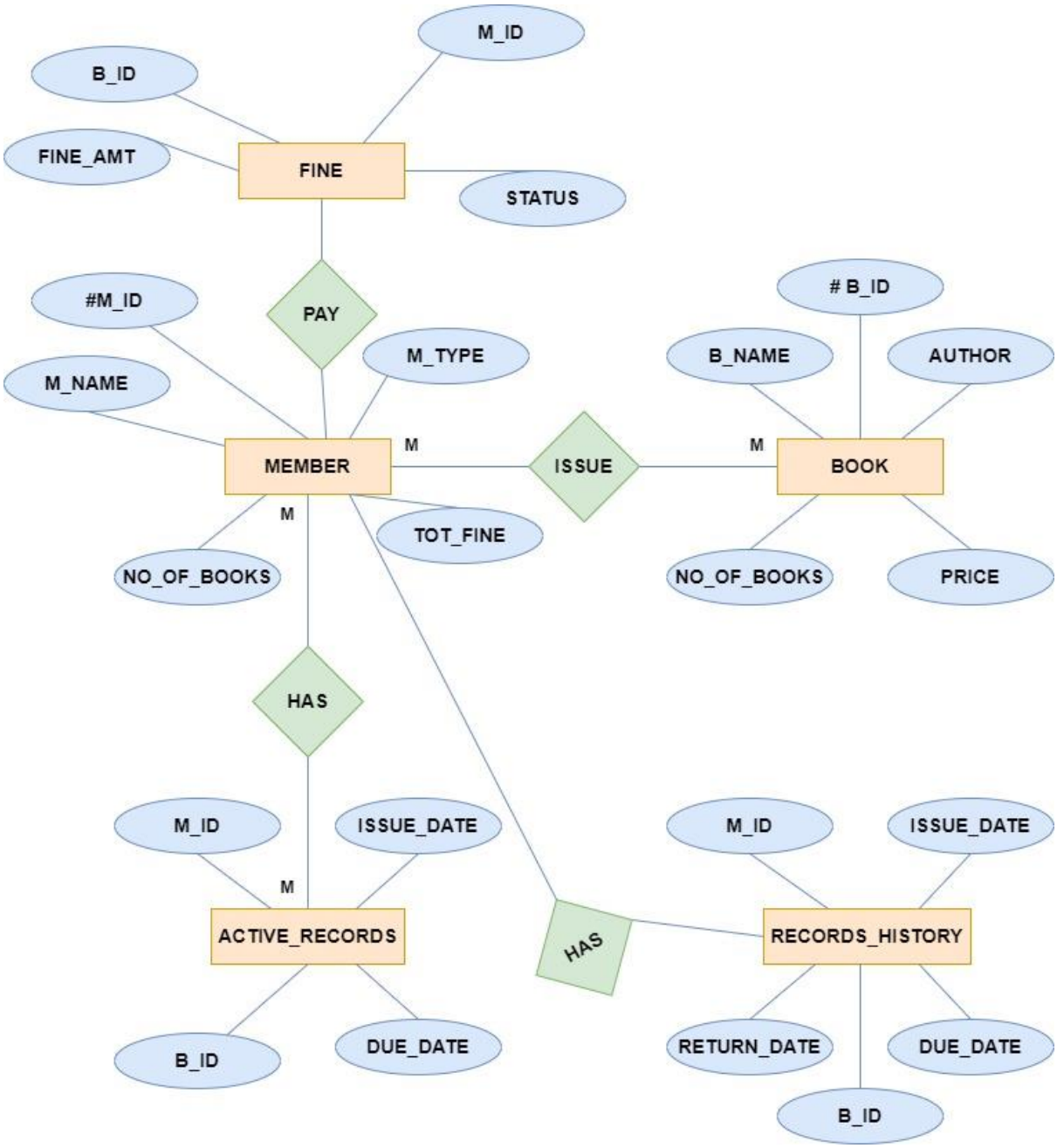
BOOK	
COLUMN	CONSTRAINT
BOOK_ID	PK
B_NAME	NOT NULL
AUTHOR	NOT NULL
PRICE	NOT NULL
NO_OF_BOOK	CHECK (PRICE>0)

ACTIVE RECORDS	
COLUMN	CONSTRAINT
BOOK_ID, MEMBER_ID	PK
BOOK_ID	FK
MEMBER_ID	FK
ISSUE_DATE	NOT NULL
DUE_DATE	NOT NULL
RETURN_DATE	

RECORDS _HISTORY	
COLUMN	CONSTRAINT
BOOK_ID, MEMBER_ID	PK
BOOK_ID	FK
MEMBER_ID	FK
ISSUE_DATE	NOT NULL
DUE_DATE	NOT NULL
RETURN_DATE	NOT NULL

FINE	
COLUMN	CONSTRAINT
BOOK_ID, MEMBER_ID	PK
BOOK_ID	FK
MEMBER_ID	FK
FINE_AMT	CHECK(FINE_AMT>0)
STATUS	CHECK(STATUS IN (‘UNPAID’,’PAID’))

ER DIAGRAM


M_ID
B_ID
FINE_AMT
FINE
STATUS
PAY
#M_ID
M_TYPE
M_NAME
MEMBER
M
ISSUE
M
BOOK
B_ID
B_NAME
AUTHOR
M
TOT_FINE
NO_OF_BOOKS
NO_OF_BOOKS
PRICE
HAS
M_ID
ISSUE_DATE
M
ACTIVE_RECORDS
HAS
M_ID
ISSUE_DATE
RECORDS_HISTORY
B_ID
DUE_DATE
RETURN_DATE
DUE_DATE
B_ID

PROCEDURE

- **PROCEDURE TO ISSUE BOOK**

```
CREATE OR REPLACE PROCEDURE borrow_book(
  p_member_no IN Member.member_id%TYPE,
  p_book_no IN Book.book_id%TYPE
)
IS
  v_member_name Member.m_name%TYPE;
  v_no_of_books Member.no_of_books%TYPE;
  v_total_fine Member.tot_fine%TYPE;
  v_book_name Book.b_name%TYPE;
  v_author Book.author%TYPE;
  v_price Book.price%TYPE;
  v_no_of_books_available Book.no_of_books%TYPE;
     v_issue_date  ACTIVE_RECORDS.issue_date%TYPE  :=
SYSDATE;
  v_due_date ACTIVE_RECORDS.due_date%TYPE := SYSDATE +
14;
BEGIN
  -- Check if member exists
  SELECT m_name, no_of_books, tot_fine
  INTO v_member_name, v_no_of_books, v_total_fine
  FROM Member
  WHERE member_id = p_member_no;

  -- Check if book exists
  SELECT b_name, author, price, no_of_books
        INTO   v_book_name,   v_author,   v_price,
v_no_of_books_available
  FROM Book
  WHERE book_id = p_book_no;

  -- Check if member has already borrowed the maximum
number of books allowed
  IF v_no_of_books >= 5 THEN
```

```
      DBMS_OUTPUT.PUT_LINE('This member has already
borrowed the maximum number of books allowed.');
    RETURN;
  END IF;

  -- Check if book is available for borrowing
  IF v_no_of_books_available <= 0 THEN
     DBMS_OUTPUT.PUT_LINE('This book is not available
for borrowing.');
    RETURN;
  END IF;

  -- Insert new ACTIVE_RECORDS record
    INSERT INTO ACTIVE_RECORDS(book_id, member_id,
issue_date, due_date)
      VALUES(p_book_no, p_member_no, v_issue_date,
v_due_date);


  DBMS_OUTPUT.PUT_LINE('Book ' || v_book_name || ' by
' || v_author || ' has been borrowed by ' ||
v_member_name || ' and is due on ' || v_due_date);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Member or book not found.');
  WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred: ' ||
SQLERRM);
END borrow_book;
```

```
Statement processed.
Book The Color Purple by Alice Walker has been borrowed by Shubham and is due on 10-MAY-23
```

BOOK_ID	MEMBER_ID	ISSUE_DATE	DUE_DATE
B1	1	26-APR-23	10-MAY-23

- **PROCEDURE TO RETURN BOOK**

```
CREATE OR REPLACE PROCEDURE return_book(
  p_member_no IN ACTIVE_RECORDS.member_id%TYPE,
  p_book_no IN ACTIVE_RECORDS.book_id%TYPE
)
IS
  v_member_name Member.m_name%TYPE;
  v_no_of_books Member.no_of_books%TYPE;
  v_total_fine Member.tot_fine%TYPE;
  v_book_name Book.b_name%TYPE;
  v_author Book.author%TYPE;
  v_price Book.price%TYPE;
  v_no_of_books_available Book.no_of_books%TYPE;
  v_issue_date ACTIVE_RECORDS.issue_date%TYPE;
  v_due_date ACTIVE_RECORDS.due_date%TYPE;
  v_return_date RECORDS_HISTORY.return_date%TYPE :=
SYSDATE;
  v_days_late INTEGER;
  v_fine_amount INTEGER;
BEGIN
  -- Check if ACTIVE_RECORDS exists
  SELECT m_name, no_of_books, tot_fine
  INTO v_member_name, v_no_of_books, v_total_fine
  FROM Member
  WHERE member_id = p_member_no;

  -- Check if book exists
  SELECT b_name, author, price, no_of_books
  INTO v_book_name, v_author, v_price,
v_no_of_books_available
  FROM Book
  WHERE book_id = p_book_no;

  -- Get ACTIVE_RECORDS details
  SELECT issue_date, due_date
  INTO v_issue_date, v_due_date
```

```
  FROM ACTIVE_RECORDS
  WHERE book_id = p_book_no AND member_id =
p_member_no;

  -- Calculate fine amount if book is returned late
  IF v_return_date > v_due_date THEN
    v_days_late := v_return_date - v_due_date;
    v_fine_amount := v_days_late * 5; -- Assuming a
fine of Rs.5 per day late
    v_total_fine := v_total_fine + v_fine_amount;
  END IF;

    UPDATE MEMBER SET tot_fine = v_total_fine WHERE
member_id = p_member_no;

--Insert returned book record into RECORDS_HISTORY
table
  INSERT INTO RECORDS_HISTORY (book_id, member_id,
issue_date, due_date, return_date)
  SELECT book_id, member_id, issue_date, due_date,
v_return_date
  FROM ACTIVE_RECORDS
  WHERE book_id = p_book_no AND member_id =
p_member_no;

-- Insert into fine table
    IF v_fine_amount > 0 THEN
    INSERT INTO FINE(book_id, member_id,fine_amt)
    SELECT book_id, member_id, v_fine_amount
    FROM ACTIVE_RECORDS
    WHERE book_id = p_book_no AND member_id =
p_member_no;
    END IF;

  -- Delete returned book record from ACTIVE_RECORDS
table
  DELETE FROM ACTIVE_RECORDS
```

```
  WHERE book_id = p_book_no AND member_id =
p_member_no;

  DBMS_OUTPUT.PUT_LINE('Book ' || v_book_name || ' by
' || v_author || ' has been returned by ' ||
v_member_name);
  IF v_fine_amount > 0 THEN
    DBMS_OUTPUT.PUT_LINE('Fine amount: Rs.' ||
v_fine_amount);
  END IF;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Record not found.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred: ' ||
SQLERRM);
END return_book;
```

```
Statement processed.
Book Beloved by Toni Morrison has been returned by Shiuli
Fine amount: Rs.53
```

- **PROCEDURE TO PAY FINE**

```
CREATE OR REPLACE PROCEDURE PAY_FINE (
   p_member_id IN FINE.MEMBER_ID%TYPE,
   p_book_id IN FINE.BOOK_ID%TYPE,
   p_fine_amt IN FINE.FINE_AMT%TYPE
)
AS
   v_status FINE.STATUS%TYPE;
BEGIN
   -- Check if the fine record exists for the specified member_id and book_id
   SELECT status INTO v_status
   FROM fine
   WHERE member_id = p_member_id
   AND book_id = p_book_id;

   IF v_status = 'UNPAID' THEN
      -- Update the status to paid
      UPDATE fine
      SET status = 'PAID'
      WHERE member_id = p_member_id
      AND book_id = p_book_id;

      -- Deduct the fine amount from the member's account
      UPDATE member
      SET tot_fine = tot_fine - p_fine_amt
      WHERE member_id = p_member_id;

      DBMS_OUTPUT.PUT_LINE('Fine of Rs.' || p_fine_amt || ' for book ' || p_book_id || ' has been paid by member ' || p_member_id || '.');
   ELSE
```

```
      DBMS_OUTPUT.PUT_LINE('Fine for book ' ||
p_book_id || ' has already been paid by member ' ||
p_member_id || '.');
   END IF;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('No fine record found for
member ' || p_member_id || ' and book ' || p_book_id
|| '.');
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred: ' ||
SQLERRM);
END PAY_FINE;

declare
    MEMBER_NAME MEMBER.M_NAME%TYPE;
begin
    -- borrow_book(1, 'B4');
    -- return_book(7, 'B4');
    -- return_book(7,'B4');
    -- pay_fine(7,'B4',37);
    -- MEMBER_NAME:=max_books_borrowed();
    --   DBMS_OUTPUT.PUT_LINE('The names of the
member(s) who have borrowed the maximum books :
'||MEMBER_NAME);
    -- MEMBER_NAME:= max_fine_to_pay();
    --   DBMS_OUTPUT.PUT_LINE('The names of the
member(s) who have to pay the maximum fine :
'||MEMBER_NAME);
end;
```

```
Statement processed.
Fine of Rs.53 for book B3 has been paid by member 4.
```

BOOK_ID	MEMBER_ID	FINE_AMT	STATUS
B4	7	38	UNPAID
B1	1	43	PAID
B3	4	53	PAID

Download CSV

TRIGGER

- **TRIGGER TO UPDATE THE NUMBER OF BOOKS AVAILABLE ON EVERY RETURN AND BORROW OF BOOK**

```
CREATE OR REPLACE TRIGGER UPDATE_BOOK
AFTER INSERT OR UPDATE ON ACTIVE_RECORDS
FOR EACH ROW
BEGIN
 IF INSERTING THEN
 UPDATE BOOK
 SET NO_OF_BOOKS=NO_OF_BOOKS-1
 WHERE book_id=:NEW.book_id;
 ELSIF UPDATING THEN
 UPDATE BOOK
 SET NO_OF_BOOKS=NO_OF_BOOKS+1
 WHERE book_id=:OLD.book_id;
 END IF;
END;
```

NO. OF BOOKS BEFORE BORROWING

BOOK_ID	B_NAME	AUTHOR	PRICE	NO_OF_BOOKS
B1	Pride and Prejudice	Jane Austen	200	4
B2	The Hobbit	J.R.R. Tolkien	100	4

AFTER BOTH BOOKS ARE BORROWED

BOOK_ID	B_NAME	AUTHOR	PRICE	NO_OF_BOOKS
B1	Pride and Prejudice	Jane Austen	200	3
B2	The Hobbit	J.R.R. Tolkien	100	3

- **TRIGGER TO UPDATE THE NUMBER OF BOOKS MEMBERS CAN ISSUE ON EVERY RETURN AND BORROW OF BOOK**

```
CREATE OR REPLACE TRIGGER UPDATE_MEMBER
AFTER INSERT OR UPDATE ON ACTIVE_RECORDS
FOR EACH ROW
BEGIN
 IF INSERTING THEN
UPDATE MEMBER
 SET NO_OF_BOOKS=NO_OF_BOOKS+1
 WHERE member_id=:NEW.member_id;
 ELSIF UPDATING THEN
 UPDATE MEMBER
 SET NO_OF_BOOKS=NO_OF_BOOKS-1
 WHERE member_id=:OLD.member_id;
 END IF;
END;
```

BEFORE BOOKS ARE BORROWED

MEMBER_ID	M_NAME	M_TYPE	NO_OF_BOOKS	TOT_FINE
1	Shubham	M	1	0
2	Vaibhav	Q	0	0

AFTER BOOKS ARE BORROWED

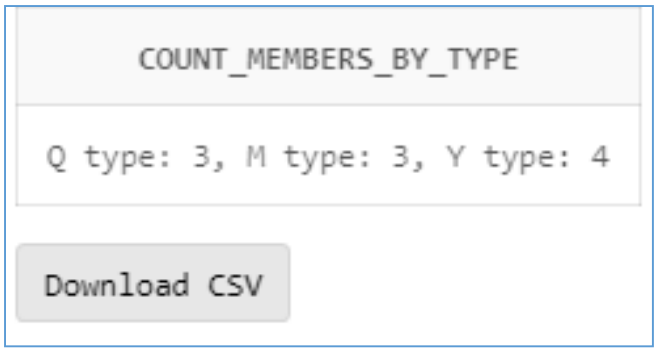
MEMBER_ID	M_NAME	M_TYPE	NO_OF_BOOKS	TOT_FINE
1	Shubham	M	2	0
2	Vaibhav	Q	1	0

FUNCTIONS

- **FUNCTION TO COUNT NO OF MEMBERS OF EACH TYPE :**

```
CREATE OR REPLACE FUNCTION count_members_by_type
RETURN VARCHAR2 IS
   q_count NUMBER := 0;
   m_count NUMBER := 0;
   y_count NUMBER := 0;
   CURSOR c_members IS SELECT m_type FROM member;
BEGIN
   FOR member_rec IN c_members LOOP
      IF member_rec.m_type = 'Q' THEN
         q_count := q_count + 1;
      ELSIF member_rec.m_type = 'M' THEN
         m_count := m_count + 1;
      ELSIF member_rec.m_type = 'Y' THEN
         y_count := y_count + 1;
      END IF;
   END LOOP;

   RETURN 'Q type: ' || q_count || ', M type: ' ||
m_count || ', Y type: ' || y_count;
END;
/
```



COUNT_MEMBERS_BY_TYPE
Q type: 3, M type: 3, Y type: 4

Download CSV

- **FUNCTION TO CALCULATE WHICH MEMBER HAS TO PAY THE MAXIMUM FINE CURRENTLY:**

```
CREATE OR REPLACE FUNCTION max_fine_to_pay
RETURN VARCHAR2
AS
  v_max_fine NUMBER;
  v_member_name VARCHAR2(50);
  CURSOR c_members IS
    SELECT m_name
    FROM Member
    WHERE tot_fine = v_max_fine;
BEGIN
  -- Get the maximum fine to be paid
  SELECT MAX(tot_fine)
  INTO v_max_fine
  FROM Member;

  -- Get the member names who have to pay the maximum fine
  FOR member IN c_members LOOP
      v_member_name := v_member_name || ', ' || member.m_name;
  END LOOP;

  -- Remove the leading comma and space from the string
  v_member_name := SUBSTR(v_member_name, 3);

  RETURN v_member_name
```

```
Statement processed.
The names of the member(s) who have to pay the maximum fine : Rohansh
```

MEMBER_ID	M_NAME	M_TYPE	NO_OF_BOOKS	TOT_FINE
1	Shubham	M	2	0
2	Vaibhav	Q	1	0
3	Hitesh	Q	1	0
4	Shiuli	Y	2	0
5	Shambhavi	Q	2	0
6	Shashank	Y	0	0
7	Rohansh	M	1	38
8	Dev	Y	0	0

- **FUNCTION TO CALCULATE WHICH MEMBER HAS BORROWED THE MAXIMUM NO. OF BOOKS AT A TIME CURRENTLY:**

```
CREATE OR REPLACE FUNCTION max_books_borrowed
RETURN VARCHAR2
AS
  v_max_books NUMBER;
  v_member_names VARCHAR2(4000);
BEGIN
  -- Get the maximum number of books borrowed
  SELECT MAX(no_of_books)
  INTO v_max_books
  FROM Member;

  -- Get the member names who have borrowed the maximum number of books
  FOR rec IN (SELECT m_name
              FROM Member
              WHERE no_of_books = v_max_books)
  LOOP
    v_member_names := v_member_names || rec.m_name || ', ';
  END LOOP;

  -- Remove the trailing comma and space
     v_member_names   :=   SUBSTR(v_member_names,   1, LENGTH(v_member_names) - 2);

  RETURN v_member_names;
END;
/
```

```
Statement processed.
The names of the member(s) who have borrowed the maximum books : Shubham, Shiuli, Shambhavi, Nikunj
```

MEMBER_ID	M_NAME	M_TYPE	NO_OF_BOOKS	TOT_FINE
1	Shubham	M	2	0
2	Vaibhav	Q	1	0
3	Hitesh	Q	1	0
4	Shiuli	Y	2	0
5	Shambhavi	Q	2	0
6	Shashank	Y	0	0
7	Rohansh	M	1	0
8	Dev	Y	0	0
9	Nikunj	M	2	0