Library Management System – PL/SQL

The various functionality associated with it is depicted as below:

Consider the following:

1. The library has 3 kinds of members
   a. Monthly – this member can borrow 4 books
   b. Yearly – this member can borrow 2 books
   c. Lifetime – This member can be borrow 6 books

2. The same kind of a book cannot be borrowed by a member at one instance.

3. The fine amount should be calculated basing on the issuedate ,returndate and duedate.

4. The fine amount can be 5/- per day.

5. When a book is issued automatically it should reflect in the book table.

<table>
<thead>
<tr>
<th>Table name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mem_no</td>
<td>Varchar2(20)</td>
<td>Primary Key</td>
</tr>
<tr>
<td>Mem_name</td>
<td>Varchar2(20)</td>
<td>Not Null</td>
</tr>
<tr>
<td>Mem_type</td>
<td>Varchar2(20)</td>
<td>(M,Y,L)</td>
</tr>
<tr>
<td>No_of_books</td>
<td>Number(4)</td>
<td></td>
</tr>
<tr>
<td>Total_fine</td>
<td>Number(4)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Book</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Book_no</td>
<td>Varchar2(20)</td>
<td>Primary Key</td>
</tr>
<tr>
<td>Book_name</td>
<td>Varchar2(20)</td>
<td>Not Null</td>
</tr>
<tr>
<td>Author</td>
<td>Varchar2(20)</td>
<td></td>
</tr>
<tr>
<td>Price</td>
<td>Varchar2(20)</td>
<td></td>
</tr>
<tr>
<td>No_of_books</td>
<td>Number(4)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trans</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Book_no</td>
<td>Varchar2(20)</td>
<td>Foreign key of books</td>
</tr>
<tr>
<td>Mem_no</td>
<td>Varchar2(20)</td>
<td>Foreign key of member</td>
</tr>
<tr>
<td>Issue_date</td>
<td>Date</td>
<td>Sysdate</td>
</tr>
</tbody>
</table>
Due_date | Date | Sysdate+7
---|---|---
Return_date | Date | 

-- table member
create table member(mem_no varchar2(20) primary key, mem_name varchar2(20) not null, mem_type varchar2(20), no_of_books number(4), total_fine number(4));

-- table book
create table book(book_no varchar2(20) primary key, book_name varchar2(20) not null, author varchar2(20), price varchar2(20), no_of_books number(4));

-- table trans
create table trans(book_no varchar2(20), mem_no varchar2(20), issue_date date, due_date date, return_date date, constraint bid_fkey FOREIGN KEY (book_no) REFERENCES book(book_no), constraint mid_fkey FOREIGN KEY (mem_no) REFERENCES member(mem_no));

-- table transaction history
create table transaction_history(book_no varchar2(20), mem_no varchar2(20), issue_date date, due_date date, return_date date, constraint bid_fkey1 FOREIGN KEY (book_no) REFERENCES book(book_no), constraint mid_fkey1 FOREIGN KEY (mem_no) REFERENCES member(mem_no))
/

--For adding member

declare

member_name varchar2(15);
member_type varchar2(20);
no varchar2(20);
id varchar2(20);

begin

member_name:='&name';
member_type:='&type';

select MAX(mem_no) into no from member;

if no is not null then
    id:=no+1;
else
id:=1;
end if;

insert into member values(id,member_name,member_type,0,0);
dbms_output.put_line('Mr/Mrs/Miss. ||member_name||', your membership id is ||id);
end;
/

-- for adding book
declare
book_name varchar2(50);
author varchar2(20);
price varchar2(20);
no_of_books number(5);
book_id varchar2(20);
begin
book_id:='&bookno';
book_name:='&bknam';
**1. Write a procedure to issue the book to the member.**

**Proc Name : Issue**

**Parameters :** book_no, mem_no

create or replace procedure insert1(book_id varchar2, mem_id number)

is

 a boolean default false;

 b boolean default false;

 c boolean default false;

 d boolean default false;

 mep number(4);
tep number(4);  
sep number(4);  
bep number(4);  
mb number(4);  
dat varchar2(10);  
typ varchar2(10);  
expiry_date date;  
ddate date;  

**Consideration for issue of book**

**a. Book No. must be a valid book no. from the book table or handle exception.**

```sql
select count(*) into tep from book where book_no = book_id;
if tep = 1
then
    dbms_output.put_line('The book '||book_id||' exist in the library');
else
    dbms_output.put_line('The book '||book_id||' does not exist in the library');
end if;
```

**b. Mem no. Must be a valid Mem no. From member table.**

```sql
select count(*) into mep from member where mem_no = mem_id;
if mep = 1
then
    dbms_output.put_line('The user '||mem_id||' is a member of club');
else
    dbms_output.put_line('The user '||mem_id||' is not a member of club');
end if;
```

**c. The same memno. Cannot borrow the same book without returning the book.**

```sql
select count(*) into sep from trans where book_no = book_id and mem_no = mem_id and return_date is null;
if sep = 1
then
    dbms_output.put_line('The user already have this book');
end if;
```

**d. If the due date is crossing the expiry date of the member, doesn’t issue the book.**

```sql
select mem_type into typ from member where mem_no=mem_id;
expiry_date := add_months(ROUND(SYSDATE,'MONTH'),1);
```
ddate := ROUND(SYSDATE,'YEAR');
if typ='M'
then
    if expiry_date < SYSDATE+7
    then
        a:=true;
        dbms_output.put_line('Your membership expiry date ' || expiry_date || ' is before due date ' || SYSDATE+7);
    end if;
elsif typ='Y'
then
    if ddate < SYSDATE+7
    then
        a:=true;
        dbms_output.put_line('Your membership expiry date ' || ddate || ' is before due date ' || SYSDATE+7);
    end if;
elsif typ='L'
then
    dbms_output.put_line('You have lifetime membership');
end if;

If the number of book is already borrowed by the member without returning the book exceeds the membership limit then handle error.
select no_of_books into mb from member where mem_no=mem_id;
if typ='M'
then
    if mb >= 4
    then
        b:=true;
        dbms_output.put_line('You have reached monthly borrow limit of 4 books');
    end if;
elsif typ='Y'
then
    if mb >= 2
    then
        b:=true;
        dbms_output.put_line('You have reached yearly borrow limit of 2 books');
    end if;
else if typ='L'
then

end if;
then
    if mb >= 6
    then
        b:=true;
        dbms_output.put_line('Your have reached lifetime borrow limit of 6 books');
    end if;
end if;

f. **If the stock of the book is not available then trap the error.**

select no_of_books into bep from book where book_no=book_id;
if bep >= 1
then
    d:= true;
    dbms_output.put_line(' The book is available in the library ');
end if;

g. **If all validations are fulfilled, then enter into transaction table bookno. Memno. Issue will by sysdate and due_date is sysdate+7,return date is null and fine is null.**

if (tep is not null and mep is not null and b is not null and a is not null and  d is not null and c is not null)
then
    insert into trans values(book_id,mem_id,SYSDATE,SYSDATE+7,NULL);
    dbms_output.put_line('its working');
end if;

h. **On Saturday or Sunday no issue of the books.**

select to_char(SYSDATE,'DY') into dat from dual;
if dat = 'SUN'
then
    dbms_output.put_line('It is '||to_char(SYSDATE,'DAY')||' so cannot issue book.');
elsif dat = 'SAT'
then
    dbms_output.put_line('It is '||to_char(SYSDATE,'DAY')||' so cannot issue book.');
else
    c:=true;
    dbms_output.put_line('It is '||to_char(SYSDATE,'DAY')||' so can issue book.');
end if;

Output:

-- simple running for new user
if user already has issued the book

-- if user already has issued the book
-- if user can borrow more book or not

SQL> execute insert1('B2',4);
The book B2 exist in the library
The user 4 is a member of club
You have reached monthly borrow limit of 4 books
PL/SQL procedure successfully completed.
SQL>

-- if Sunday or Saturday no return of book

SQL> execute insert1('B4',5);
The book B4 exist in the library
The user 5 is a member of club
You have lifetime membership
It is SUNDAY  so cannot issue book.
PL/SQL procedure successfully completed.
SQL>
2. Write a procedure to return the book.

Proc name : Return

Parameter : book_no, mem_no

create OR REPLACE procedure retrn(book_id varchar2, mem_id number)

is
fine number(20);
memid number(20);
retrn_date date not null := '30-AUG-16';
dat varchar2(5);

dd date;

begin

Consideration for return book

a. Return of the book is possible only if the member has borrowed the book, check for existence of record in the transaction table.

select mem_no into memid from trans where mem_no = mem_id and book_no=book_id;

b. Update return_date with the current date and calculate the fine amount by finding the difference between due_date and return_date.

update trans set return_date='30-AUG-16' where book_no=book_id and mem_no=memid;

select due_date into dd from trans where book_no=book_id and mem_no=memid;

fine := (retrn_date - dd)*5;
dbms_output.put_line('Fine is '||fine);

c. Update the total_fine of that member by add this fine amount with the existing total_fine in the member table.

update member set total_fine = fine where mem_no=memid;

d. On Saturday or Sunday no return of book.

select to_char(SYSDATE,'DY') into dat from dual;

if dat = 'SUN'

then

dbms_output.put_line('It is '||to_char(SYSDATE,'DAY')||' so you cannot return book.');
end if;
if dat = 'SAT'
then
dbms_output.put_line('It is '||to_char(SYSDATE,'DAY')||' so you cannot return book.');
end if;
e. Upon returning the book delete the information from the transaction table and move the data to transaction_history.
-- used using trigger
f. Create transaction_history as that of transaction table to record old data.
-- table created transaction history
EXCEPTION
WHEN NO_DATA_FOUND THEN
dbms_output.put_line('There is no book issued to this member.');
end;
/
-- return date is set as 30 August for now to calculate fine
3. Write a trigger to automatically increment and decrement the no_of_books from the and member table upon issue and return.

CREATE OR REPLACE TRIGGER incr_trigger
AFTER INSERT OR UPDATE ON trans
FOR EACH ROW
BEGIN
  IF INSERTING THEN
    UPDATE book
    SET no_of_books = no_of_books - 1
    WHERE book_no = :NEW.book_no;
    UPDATE MEMBER
    SET no_of_books = no_of_books + 1
    WHERE mem_no = :NEW.mem_no;
  ELSIF UPDATING THEN
    UPDATE book
SET no_of_books = no_of_books+1
WHERE book_no = :old.book_no;
UPDATE MEMBER
SET no_of_books = no_of_books-1
WHERE mem_no = :NEW.mem_no;
END IF;

END;
/

-- on issue of book

-- before issue of book

-- B4 has 5 no of book and priyansh has 0 no_of_books
-- after issuing a book B4 to priyansh member_id 2
4. Write a trigger to move the data from the transaction to transaction_history table upon deletion.

create or replace trigger move_trigger
before delete on trans
for each row
begin
insert into transaction_history
values(:old.book_no,:old.mem_no,:old.issue_date,:old.due_date,:old.return_date);
end;
/

-- hence the built library management system meets all the requirements specified.