Group Project

MIS 6326.001 – Data Management

A report submitted

by

Diana Francisco Almeida(Net ID: DXA180001*)*
Mary Ridhima Golamari (Net ID: mxg170031)
Sayali Paseband (Net ID: SSP180006)

Presented to

Professor Uri Smashnov

University of Texas at Dallas
1st December 2018

APPLICATION SOURCE CODE

1. BOOK SEARCH AND AVAILABILITY:

```
select a.isbn, a.title, b.author_name, c.branch_name
FROM authors b JOIN book_authors d on d.author_id = b.author_id
JOIN book a on d.isbn = a.isbn
JOIN book_copies e on e.book_id = a.isbn
JOIN library_branch c on e.branch_id = c.branch_id
where e.no_of_copies <> 0
and c.branch_name=:P2_BRANCH_NAME;
```

```
select LIBRARY_BRANCH.BRANCH_NAME D,
LIBRARY_BRANCH.BRANCH_NAME R
 from LIBRARY_BRANCH LIBRARY_BRANCH
```

CREATE 500 RECORDS IN BOOK_LOAN TABLE:

```
declare
  cursor c2 is
  SELECT ID0000ID FROM BORROWERS;
begin
  for x in c2
  loop
    insert into _BOOK_LOANS(BOOK_ID,CARD_NO,date_out,due_date,date_in)
    select *
    from(
       select t1.*,t1.date_out + 14 as due_date, t1.date_out + trunc(dbms_random.value(1,60))
as date_in
```

```
        from(
            select tb.book_id,tbs.id0000id,sysdate - trunc(dbms_random.value(1,355)) date_out
            from book_copies tb, borrowers tbs
            where tbs.id0000id = x.id0000id
            order by dbms_random.value) t1)
            where rownum < dbms_random.value(1,10);
  end loop;
  commit;
end;
```

CREATE 50 FINES RECORDS FOR 200 DIFFERENT BORROWERS:

```
select loan_id,card_no, (date_in-due_date)*0.25 as fine_amt, CASE WHEN ((date_in-due_date)*0.25) < 10.00 THEN 'Paid' WHEN ((date_in-due_date)*0.25) < 10.0 THEN 'Paid' WHEN ((date_in-due_date)*0.25) < 10 THEN 'Paid' ELSE 'Unpaid' END CASE from book_loan where date_in > due_date and date_out < due_date;
```

---

2. BOOK LOAN

CHECK IF BORROWER IS ELIGIBLE TO BORROW BOOK:

```
SELECT DISTINCT A.BOOK_ID, A.CARD_NO, A.DUE_DATE, A.DATE_IN, A.LOAN_ID, B.FINE_AMT, B.PAID
FROM BOOK_LOAN A JOIN FINES B ON A.LOAN_ID=B.LOAN_ID
WHERE A.CARD_NO IN (SELECT CARD_NO
FROM BOOK_LOAN
WHERE DATE_IN > DUE_DATE
AND DATE_IN < SYSDATE
```

```
GROUP BY CARD_NO)
AND A.DATE_IN > A.DUE_DATE
AND A.DATE_IN < SYSDATE
AND B.PAID LIKE '%Unpaid%'
ORDER BY A.DATE_IN DESC;
```

BOOK CHECK IN:

```
UPDATE BOOK_LOAN SET DATE_IN =:P14_DATEIN
WHERE LOAN_ID=:P14_LOANID AND CARD_NO=:P14_CARDNO AND
BOOK_ID=:P14_BOOKID;

UPDATE FINES SET PAID =:P14_PAID
WHERE LOAN_ID=:P14_LOANID AND CARD_NO=:P14_CARDNO;
```

PL/SQL BLOCK FOR BOOK CHECK OUT FORM:

```
INSERT INTO
BOOK_LOAN(BOOK_ID,CARD_NO,DATE_OUT,DUE_DATE,LOAN_ID)
VALUES(:P21_BOOKID,:P21_CARDNO,:P21_DATEOUT,:P21_DUEDATE,
AUTHOR_ID_SEQ.nextval);

UPDATE BOOK_COPIES SET NO_OF_COPIES = (SELECT DISTINCT
NO_OF_COPIES-1 FROM BOOK_COPIES WHERE BOOK_ID=:P21_BOOKID AND
BRANCH_ID=:P21_BRANCHID AND NO_OF_COPIES <> 0) WHERE
BOOK_ID=:P21_BOOKID AND BRANCH_ID=:P21_BRANCHID AND NO_OF_COPIES
<> 0;
```

BOOK CHECKOUT

CHECK IF BOOK IS AVAILABLE TO CHECK OUT:

```
SELECT DISTINCT G.BOOK_ID, A.TITLE, A.AUTHOR, G.DATE_IN,
D.BRANCH_NAME, E.NO_OF_COPIES
```

```
FROM BOOK A JOIN BOOK_AUTHORS B ON A.ISBN=B.ISBN
JOIN AUTHORS C ON C.AUTHOR_ID = B.AUTHOR_ID
JOIN BOOK_COPIES E ON A.ISBN = E.BOOK_ID
JOIN BOOK_LOAN G ON G.BOOK_ID = E.BOOK_ID
JOIN LIBRARY_BRANCH D ON D.BRANCH_ID = E.BRANCH_ID
WHERE G.BOOK_ID IN (SELECT BOOK_ID
FROM BOOK_LOAN
WHERE DATE_IN < TRUNC(SYSDATE) AND DATE_IN <> TRUNC(SYSDATE)
GROUP BY BOOK_ID)
AND G.DATE_IN < TRUNC(SYSDATE)
AND G.DATE_IN <> TRUNC(SYSDATE)
AND E.NO_OF_COPIES <> 0
ORDER BY G.DATE_IN DESC;
```

DATE OUT IN CHECKOUT FORM

```
select * from (SELECT TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY') FROM dual AS DATE_OUT,
```

DUE DATE IN CHECK IN FORM

```
SELECT TO_CHAR(CURRENT_DATE + 14, 'DD-MON-YYYY') FROM dual AS DUE_DATE,
```

CARD NO. IN CHECK IN FORM

```
select distinct BOOK_LOAN.CARD_NO D,  BOOK_LOAN.CARD_NO R
 from BOOK_LOAN BOOK_LOAN;
```

BOOK ID IN CHECKOUT FORM

```
select BOOK.TITLE D, BOOK.TITLE R from BOOK BOOK;
```

3. BORROWERS MANAGEMENT

TRIGGER FOR SSN_DUPLICATE:

```
create or replace TRIGGER borrowers_ssn_insert
BEFORE INSERT
   ON borrowers
   FOR EACH ROW
DECLARE
   ssn_count number(10);
   pragma AUTONOMOUS_TRANSACTION;
BEGIN
select count(*)
into ssn_count
from borrowers
 where ssn = :NEW.ssn;
  if ssn_count > 0
then
  RAISE_APPLICATION_ERROR(-20000,'Duplicate SSN');
  end if;
END;
```

TO CHECK ALL PAID AND UNPAID FINES FOR A PARTICULAR BORROWER:

```
select b.book_id, b.CARD_NO, a.PAID, a.FINE_AMT
from FINES a JOIN BOOK_LOAN b
ON a.LOAN_ID = b.LOAN_ID
and b.CARD_NO=:P23_NEW;
```

We used form on table in oracle apex to insert values in borrowers table.

We also used a transaction statement to update the fines table with new records to input, records whose due_date has exceeded.

UPDATE FINES TABLE WITH FINE_AMT:

```
DECLARE
cursor c1 is
  select COUNT(*)
from BOOK_LOAN b
where b.DATE_IN IS NULL and due_date < sysdate;
BEGIN
  for x in c1
  loop
insert into FINES(LOAN_ID,CARD_NO,FINE_AMT,PAID)
SELECT * FROM (select B.LOAN_ID,b.CARD_NO, ROUND((SYSDATE-
DUE_DATE)*0.25,2) FINE_AMT,'Unpaid' Paid
from BOOK_LOAN b
where b.DATE_IN IS NULL and due_date < sysdate);
end loop;
Commit;
END;
```

BORROWER FINES INFORMATION

```
select b.CARD_NO, a.PAID
from FINES a JOIN BOOK_LOAN b
ON a.LOAN_ID = b.LOAN_ID
and b.CARD_NO=:BORROWERSEARCH;
```

4. FINES

TRIGGER ON BOOK_LOAN FOR PAID CHECK CONDITION:

```
create or replace TRIGGER PaidCheckConditn
BEFORE INSERT
ON book_loan
FOR EACH ROW
DECLARE
paid_status_count number(10);
PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
select count(*) into paid_status_count from fines a join book_loans b on a.loan_id =
b.loan_id
where card_no=:NEW.card_no and paid='Unpaid';
if paid_status_count <> 0
   then
       raise_application_error(-20000,'Fine Due');

end if;
Commit;
END;
```

TO CHECK ALL PAID AND UNPAID FINES FOR A PARTICULAR BORROWER:

```
select b.book_id, b.CARD_NO, a.PAID, a.FINE_AMT
from FINES a JOIN BOOK_LOAN b
ON a.LOAN_ID = b.LOAN_ID
and b.CARD_NO=:P23_NEW;
```

OVERDUE FINES QUERY:

```
select b.CARD_NO, b.BOOK_ID, b.DUE_DATE, b.DATE_IN
from BOOK_LOAN b
where b.DATE_IN IS NULL and due_date < sysdate and b.CARD_NO=:P11_NEW1;
```

UPDATE FINES TABLE WITH FINE_AMT:

```
DECLARE
cursor c1 is
  select COUNT(*)
from BOOK_LOAN b
where b.DATE_IN IS NULL and due_date < sysdate;
BEGIN
  for x in c1
  loop
insert into FINES(LOAN_ID,CARD_NO,FINE_AMT,PAID)
SELECT * FROM (select B.LOAN_ID,b.CARD_NO, ROUND((SYSDATE-
DUE_DATE)*0.25,2) FINE_AMT,'Unpaid' Paid
from BOOK_LOAN b
where b.DATE_IN IS NULL and due_date < sysdate);
end loop;
Commit;
END;
```

UPDATE QUERY ON FINES TABLE :

```
UPDATE FINES SET PAID = 'PAID'
WHERE LOAN_ID=:P33_LOAN_ID AND CARD_NO=:P33_CARD_NO;
```

PAYMENT OF FINES:

```
UPDATE FINES SET PAID = 'PAID'
WHERE CARD_NO=:P35_NEW AND LOAN_ID=:P35_NEW_1;
```

---

5. REPORTS:

1. TOP 10 CHECKED IN LATE

```
SELECT DISTINCT E.BOOK_ID, A.TITLE, A.AUTHOR, G.DATE_OUT, G.DUE_DATE,
G.DATE_IN

FROM BOOK A JOIN BOOK_AUTHORS B ON A.ISBN=B.ISBN

JOIN AUTHORS C ON C.AUTHOR_ID = B.AUTHOR_ID

JOIN BOOK_COPIES E ON A.ISBN = E.BOOK_ID

JOIN BOOK_LOAN G ON G.BOOK_ID = E.BOOK_ID

JOIN LIBRARY_BRANCH D ON D.BRANCH_ID = E.BRANCH_ID

AND CARD_NO IN (SELECT CARD_NO

FROM BOOK_LOAN

WHERE DATE_IN > DUE_DATE

GROUP BY CARD_NO)

ORDER BY G.DATE_OUT DESC

FETCH FIRST 10 ROWS ONLY;
```

---

2. TOP 10 LEAST POPULAR BOOKS

```
SELECT DISTINCT E.BOOK_ID, A.TITLE, A.AUTHOR, G.DATE_OUT, G.DATE_IN

FROM BOOK A JOIN BOOK_AUTHORS B ON A.ISBN=B.ISBN

JOIN AUTHORS C ON C.AUTHOR_ID = B.AUTHOR_ID

JOIN BOOK_COPIES E ON A.ISBN = E.BOOK_ID

JOIN BOOK_LOAN G ON G.BOOK_ID = E.BOOK_ID

JOIN LIBRARY_BRANCH D ON D.BRANCH_ID = E.BRANCH_ID

WHERE G.DATE_OUT BETWEEN SYSDATE-365 AND SYSDATE

AND CARD_NO IN (SELECT CARD_NO
```

```
FROM BOOK_LOAN
WHERE DATE_IN BETWEEN SYSDATE-365 AND SYSDATE
GROUP BY CARD_NO)
ORDER BY G.DATE_OUT ASC
FETCH FIRST 10 ROWS ONLY;
```

---

3. TOP 10 MOST POPULAR BOOKS

```
SELECT DISTINCT E.BOOK_ID, A.TITLE, A.AUTHOR, G.DATE_OUT, G.DATE_IN
FROM BOOK A JOIN BOOK_AUTHORS B ON A.ISBN=B.ISBN
JOIN AUTHORS C ON C.AUTHOR_ID = B.AUTHOR_ID
JOIN BOOK_COPIES E ON A.ISBN = E.BOOK_ID
JOIN BOOK_LOAN G ON G.BOOK_ID = E.BOOK_ID
JOIN LIBRARY_BRANCH D ON D.BRANCH_ID = E.BRANCH_ID
WHERE G.DATE_OUT BETWEEN SYSDATE-30 AND SYSDATE
AND CARD_NO IN (SELECT CARD_NO
FROM BOOK_LOAN
WHERE DATE_IN BETWEEN SYSDATE-30 AND SYSDATE
GROUP BY CARD_NO)
ORDER BY G.DATE_OUT DESC
FETCH FIRST 10 ROWS ONLY;
```

EXTRA QUERIES IN APEX REGIONS/PAGES

==========

```
SELECT DATE_OUT, DUE_DATE, CARD_NO
FROM BOOK_LOAN
WHERE DATE_OUT=:12_DATE_OUT AND
DUE_DATE=:12_DUE_DATE AND
CARD_NO=:12_CARD_NO;
```

```
select a.isbn, a.title, b.author_name, c.branch_name
FROM authors b JOIN book_authors d on d.author_id = b.author_id
JOIN book a on d.isbn = a.isbn
JOIN book_copies e on e.book_id = a.isbn
JOIN library_branch c on e.branch_id = c.branch_id
where e.no_of_copies <> 0
and c.branch_name=:P2_BRANCH_NAME;
```

```
select LIBRARY_BRANCH.BRANCH_NAME D,
LIBRARY_BRANCH.BRANCH_NAME R
 from LIBRARY_BRANCH LIBRARY_BRANCH
```

```
select a.card_no, b.paid
from fines a JOIN book_loan b
where a.loan_id = b.loan_id
and a.card_no=:BORROWERSEARCH;
```

```
select b.CARD_NO, a.PAID
from FINES a JOIN BOOK_LOAN b
ON a.LOAN_ID = b.LOAN_ID
and b.CARD_NO=:BORROWERSEARCH;
```

FINES----------

```
select book_loan.card_no d, book_loan.card_no r
from book_loan book_loan
```

```
select b.CARD_NO, b.BOOK_ID, b.DUE_DATE, b.DATE_IN
from FINES a JOIN BOOK_LOAN b
ON a.LOAN_ID = b.LOAN_ID
and b.CARD_NO=:BORROWERSEARCH
where b.DATE_IN = NULL and DUE_DATE < SYSDATE;

select b.book_id, b.CARD_NO, a.PAID, a.FINE_AMT
from FINES a JOIN BOOK_LOAN b
ON a.LOAN_ID = b.LOAN_ID
and b.CARD_NO=:P23_NEW;

BEGIN
INSERT INTO BORROWERS VALUES(CARD_ID_SEQ.NEXTVAL);
END;


UPDATE FINES SET PAID = 'PAID'
WHERE LOAN_ID=:P33_LOAN_ID AND CARD_NO=:P33_CARD_NO;



BEGIN
 INSERT INTO
BOOK_LOAN(BOOK_ID,CARD_NO,DATE_OUT,DUE_DATE,LOAN_ID)
VALUES(:12_BOOK,:12_CARD_NO,:12_DATE_OUT,:12_DUE_DATE,
AUTHOR_ID_SEQ.nextval);
 UPDATE BOOK_COPIES SET NO_OF_COPIES = (SELECT DISTINCT
NO_OF_COPIES-1 FROM BOOK_COPIES WHERE BOOK_ID=BOOK_ID AND
BRANCH_ID=:P12_BRANCHNO_NEW AND NO_OF_COPIES <> 0) WHERE
```

```
BOOK_ID=BOOK_ID AND BRANCH_ID=:P12_BRANCHNO_NEW AND
NO_OF_COPIES <> 0;

 END;
```