Islamic University of
Technology

Lab-4 Report: PL/SQL (Functions, Procedures, Cursors)

Submitted by:

Name : Mashrur Ahsan

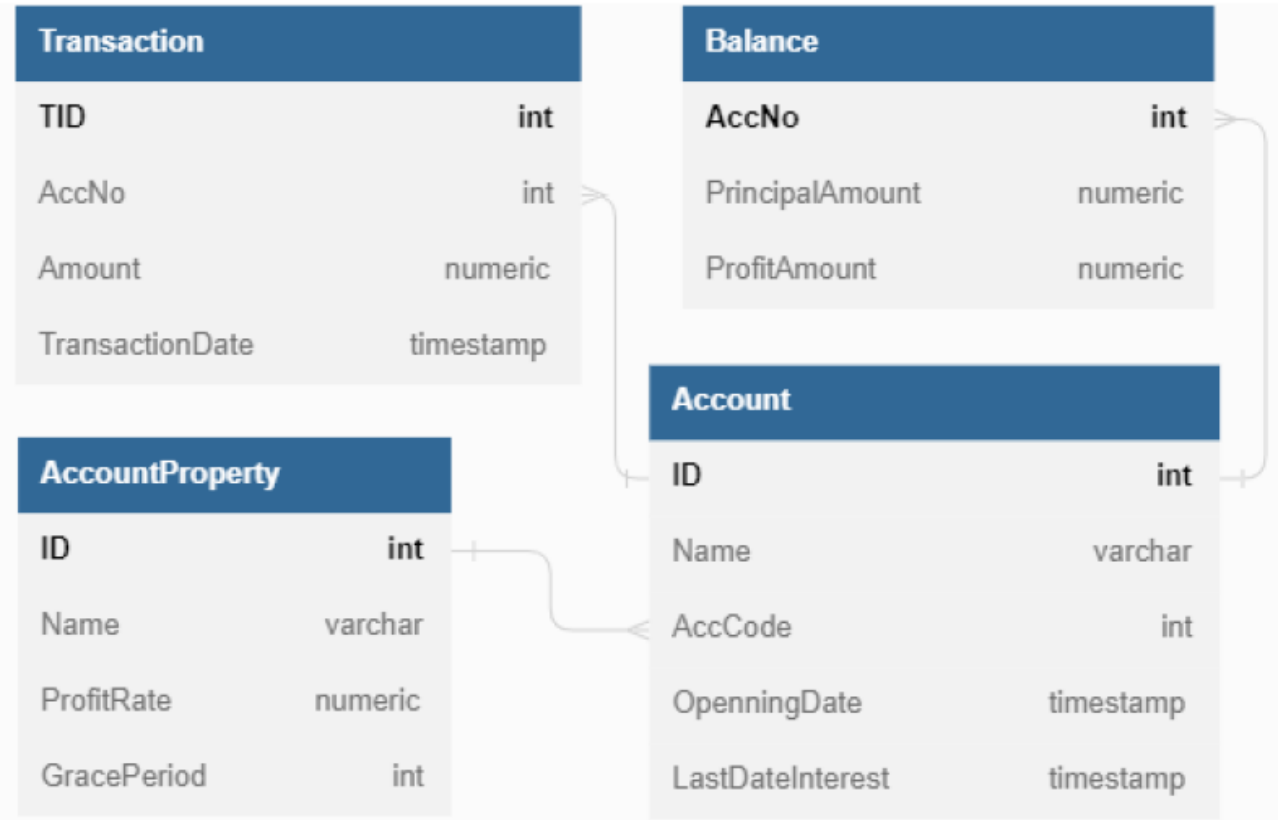
ID : 200042115

Program : SWE

Department : CSE

Course Code : CSE 4410

The Schema (Banking Database):



Default Insertions

Table 1: Account Property.

ID	Name	ProfitRate(per month)	GracePeriod (month)
2002	monthly	2.8	1
3003	quarterly	4.2	4
4004	biyearly	6.8	6
5005	yearly	8	12

Problem Statements

1. You have to write a function to calculate the current balance from the transactions.

The Solution:

```
4   -- 1 : Calculating the current balance from the transactions
5
6   CREATE OR REPLACE FUNCTION get_current_balance (id number)
7   RETURN number
8   AS
9   |  | v_principal_balance number;
10  |  | v_change number;
11  |  | v_profit number;
12  Begin
13  |
14  |  select PrincipalAmount into v_principal_balance from BALANCE where AccNo = id;
15  |
16  |  /* A transaction can be negative (debit) or positive (credit) */
17  |  SELECT sum(Amount) as Change   /* That's why we're summing the amount to get the net change */
18  |  into v_change
19  |  from Transaction
20  |  where AccNo = id
21  |  Group by AccNo;
22  |
23  |  return v_principal_balance + v_change;
24  END;
25  /
26
27  SELECT get_current_balance(1) FROM DUAL;
28  SELECT get_current_balance(2) FROM DUAL;
29
30  DECLARE
31  |  | ID number;
32  BEGIN
33  |  | ID := '&ID';
34  |  DBMS_OUTPUT.PUT_LINE(get_current_balance(ID)) ;
35  END;
36  /
37
```

Explanation & Analysis:

Line 17: A transaction can be negative (debit) or positive (credit). That's why we're summing the amount to get the net change

Line 27-28: Without user input.

Line 30-36: Taking user input (the Account ID).

Line 23: Returning the sum of principal amount and the transaction change amount.

Problems Faced:

- Nothing too significant.

The queries I needed to solve the problem:

```
40
41   select AccNo, PrincipalAmount from BALANCE;
42
43   SELECT AccNo, sum(Amount) as Change from Transaction
44   where AccNo = 1
45   Group by AccNo;
46
47   SELECT accno, ProfitAmount from Balance;
48
```

2. Write another function to calculate the profit based on profitRate, amount and duration. Take account id as input and return profit, balance before profit, and balance after profit.

The Solution:

```
48  -- 2 : Calculating the profit based on profit rate, grace period and amount
49
50  CREATE OR REPLACE FUNCTION calculate_profit(p_acc_id NUMBER)
51  RETURN VARCHAR2
52  AS
53    v_profit_rate NUMBER;
54    v_principal_amount NUMBER;
55    v_profit NUMBER(12,3);
56    v_balance_before NUMBER(12,3);
57    v_balance_after NUMBER(12,3);
58    v_days NUMBER;
59    v_last_date_interest DATE;
60    v_grace_period NUMBER;
61    v_times_balance_increase NUMBER;
62  BEGIN
63
64    SELECT (a.profitrate/100) INTO v_profit_rate  /* Fetching Profit Rate */
65    FROM AccountProperty a, ACCOUNT_ ac
66    WHERE ac.AccCode = a.ID AND ac.ID = p_acc_id;
67
68    SELECT b.PrincipalAmount INTO v_principal_amount  /* Fetching Principal Amount */
69    FROM Balance b, ACCOUNT_ ac
70    WHERE b.AccNo = ac.ID AND ac.ID = p_acc_id;
71
72    SELECT ac.LastDateInterest INTO v_last_date_interest  /* Fetching Last Date Inerest */
73    FROM ACCOUNT_ ac WHERE ac.ID = p_acc_id;
74
75    SELECT a.GracePeriod into v_grace_period   /* Fetching Grace Period */
76    FROM AccountProperty a, ACCOUNT_ ac
77    where a.ID = ac.AccCode AND ac.ID = p_acc_id;
78
```

```
79      v_days := TRUNC(SYSDATE) - TRUNC(v_last_date_interest);  /* Difference Between the Dates in days */
80
81      /* After every grace period, Profit amount increases. Here we're calculating the # of increases */
82      v_times_balance_increase := floor((floor(v_days / 30))/v_grace_period);
83
84      v_balance_after := v_principal_amount;
85
86      FOR i in 1..v_times_balance_increase
87      LOOP                        /* The initial balance increases after every grace peiod */
88      |  v_balance_after := v_balance_after + v_profit_rate*v_balance_after;
89      END LOOP;
90
91      v_balance_before := v_principal_amount;
92
93      v_profit := v_balance_after - v_balance_before;
94
95      RETURN 'Profit: ' || v_profit || '   Before: ' || v_balance_before || '   After:' || v_balance_after;
96  END;
97  /
98
99  SELECT calculate_profit(1) FROM DUAL;
100 SELECT calculate_profit(2) FROM DUAL;
101
102 DECLARE
103 |  |  ID number;
104 BEGIN
105 |  |  ID := '&ID';
106 |  DBMS_OUTPUT.PUT_LINE(calculate_profit(ID)) ;
107 END;
108 /
109
```

Explanation & Analysis:

Line 79: We are getting the difference between the dates in days with the help of the TRUNC() function. Another example showing the use of TRUNC() functions: The answer is 2714.

```
BEGIN
|  |  DBMS_OUTPUT.PUT_LINE(TRUNC(SYSDATE) - TRUNC(TO_DATE('01-SEP-15'))) ;
END;
/
```

Line 86-89: Increasing the initial amount every time there's a completion of a grace period.

Line 99-109: Providing the ID without user input and with user input.

Problems Faced:

- Nothing too significant.

The queries I needed to solve the problem:

```
110
111   SELECT floor((floor((TRUNC(SYSDATE) - TRUNC(ACCOUNT_.LastDateInterest))/30))/GracePeriod)
112   from Account_, AccountProperty
113   where Account_.AccCode = AccountProperty.ID AND Account_.AccCode = 4004;
114
115
116   BEGIN
117   |  |  DBMS_OUTPUT.PUT_LINE(TRUNC(SYSDATE) - TRUNC(TO_DATE('01-SEP-15'))) ;
118   END;
119   /
120
```

3. Write a procedure to calculate all accounts' profit (i.e. profit will be calculated if it satisfies conditions). Use the cursor for loop for this problem. The procedure will insert the appropriate record in its Amounts table.

The Solution:

```
122  -- 3 : Calculate profit for all accounts
123
124  CREATE OR REPLACE PROCEDURE calculate_profit_all_accounts
125  AS
126      v_acc_id number;
127      output varchar2(1000);
128      profit number(12,3);
129
130      cursor get_accounts is
131      SELECT ac.ID FROM Account_ ac;
132
133  BEGIN
134    OPEN get_accounts;
135    LOOP
136      FETCH get_accounts into v_acc_id;   /* Fetching every account from ACCOUNT_ */
137      EXIT WHEN get_accounts%notfound;
138      output := calculate_profit(v_acc_id);   /* getting all the relevant info */
139      DBMS_OUTPUT.PUT_LINE('Account ID: ' || v_acc_id || '    ' || output );
140
141      /* TRIM() function is helping us extract the profit from the output */
142      SELECT TRIM( ':' FROM REGEXP_SUBSTR( output, '\:[^B]+' )) INTO profit FROM DUAL;
143
144      /* Updating the profit amount */
145      UPDATE BALANCE SET ProfitAmount = profit WHERE AccNo =  v_acc_id;
146
147      /* Updating the Principal amount (which will increase because of profit) */
148      UPDATE BALANCE SET PrincipalAmount = PrincipalAmount + profit WHERE AccNo =  v_acc_id;
149
150      /* Updating Last Date of Interest to SYSDATE */
151      UPDATE ACCOUNT_ SET LastDateInterest = SYSDATE WHERE ID = v_acc_id;
152
153    END LOOP;
154    CLOSE get_accounts;
155
156  END;
157  /
158
159  BEGIN
160      calculate_profit_all_accounts();
161  END;
162  /
163
```

Explanation & Analysis:

Line 130-131: Explicit cursor which helps us fetch every single account ID from the ACCOUNT_ table.

Line 142: The TRIM() function is being used to extract variable (in this case: the profit amount) from a string (in this case: from the string that was returned by the previous function). Extracting the characters from ":" to "B".

Line 148: The new principal amount will be the addition of the previous amount and the profit amount.

Problems Faced:

- Had some difficulty figuring out on how to extract variables from a string.

The DDLs and Insertions:

```
1   DROP table Transaction;
2   DROP table BALANCE;
3   DROP table ACCOUNT_;
4   DROP table AccountProperty;
```

```
7   CREATE TABLE AccountProperty
8   (
9       ID number,
10      name VARCHAR(255),
11      ProfitRate number,
12      GracePeriod number,
13      CONSTRAINT PK_AccountProperty PRIMARY KEY(ID)
14  );
15
16  CREATE TABLE ACCOUNT_
17  (
18      ID number,
19      name VARCHAR(255),
20      AccCode number,
21      OpeningDate Date,
22      LastDateInterest Date,
23      CONSTRAINT PK_ACCOUNT PRIMARY KEY (ID),
24      CONSTRAINT FK_ACCOUNT FOREIGN KEY (AccCode) REFERENCES AccountProperty (ID)
25  );
26
```

```
26
27  CREATE TABLE Balance
28  (
29      AccNo number,
30      PrincipalAmount number,
31      ProfitAmount number,
32      CONSTRAINT PK_BALANCE PRIMARY KEY (AccNo),
33      CONSTRAINT FK_BALANCE FOREIGN KEY (AccNo) REFERENCES ACCOUNT_(ID)
34  );
35
36  Create Table Transaction
37  (
38      TID number,
39      AccNo number,
40      Amount number,
41      TransactionDate Date,
42      CONSTRAINT PK_TRANSACTION PRIMARY KEY (TID),
43      CONSTRAINT FK_TRANSACTION FOREIGN KEY (AccNo) REFERENCES ACCOUNT_(ID)
44  );
45
```

```
46
47  insert into accountProperty values(2002, 'monthly', 2.8, 1);
48  insert into accountProperty values(3003, 'quaterly', 4.2, 4);
49  insert into accountProperty values(4004, 'biyearly', 6.8, 6);
50  insert into accountProperty values(5005, 'yearly', 8, 12);
51
52  insert into Account_ values(1, 'MAK', 3003, '31-DEC-2021' , '10-JAN-2023');
53  insert into Account_ values(2, 'SAM', 4004, '31-DEC-2020' , '10-JAN-2021');
54
55
56  insert into Balance values(1,  10000, 0);
57  insert into Balance values(2,  20000, 0);
58
59
60  insert into Transaction values(1111, 1, 500, '02-FEB-2022');
61  insert into Transaction values(1112, 1, -2000, '02-FEB-2022');
62  insert into Transaction values(1113, 2, 200, '12-SEP-2022');
63
```

Explanation & Analysis:

AccountProperty table:

- ID is the primary key and has a one-to-one relationship with AccCode in the ACCOUNT_ table.

ACCOUNT_ table:

- ID is the primary key and has a one-to-many relationship with AccNo in the Balance table.
- AccCode has a one-to-one relationship with ID in the AccountProperty table.

Balance table:

- AccNo is the primary key and has a one-to-many relationship with TID in the Transaction table.

Transaction table:

- TID is the primary key and has a one-to-one relationship with AccNo in the Balance table.