Islamic University of Technology

Lab 10

CSE 4308 - DBMS Lab

Submitted To :

Zannatun Naim Sristy
Lecturer, CSE Department
Islamic University of Technology

Submitted By :

Rhidwan Rashid
ID : 200042149
Prog. : SWE
Dept. : CSE

Task 1:

Analysis: The problem requires providing a 10% increase to the instructors that get a salary less than 75000 and also show the number of instructors whose salary got the increase. The problem needs to be solved by cursor.

Working code:

```
declare
rows number(3);
begin
  UPDATE instructor
  SET salary = salary + salary * 0.1
  WHERE salary < 75000;

  if SQL%FOUND then
    rows := SQL%ROWCOUNT;
    DBMS_OUTPUT.PUT_LINE('Rows affected: ' || rows);
  end if;
end;
/
```

Explanation: Here an implicit cursor is used which is created by oracle when a sql statement has been executed. The code blocks are the same as normal pl blocks. Cursor is used in the "SQL%FOUND" statement. The "%FOUND" statement returns true if the query has affected one or more rows. "SQL%ROWCOUNT" statement returns the number of rows that have been affected.

Task 2:

Analysis: The problem requires writing a procedure that will print time slots for every teacher.

Working code:

```
CREATE OR REPLACE
PROCEDURE PRINT_TIMESLOT
AS
begin
```

```
    FOR i IN (SELECT T.TIME_SLOT_ID, T.DAY, T.start_hr, T.start_min
,T.end_hr, T.end_min FROM INSTRUCTOR I, TEACHES E, SECTION S,TIME_SLOT T
WHERE I.ID = E.ID AND
                    E.COURSE_ID = S.COURSE_ID AND E.SEC_ID = S.SEC_ID AND
E.SEMESTER = S.SEMESTER AND E.YEAR = S.YEAR AND
                    S.TIME_SLOT_ID = T.TIME_SLOT_ID) LOOP
        DBMS_OUTPUT . PUT_LINE (i.TIME_SLOT_ID || ' ' || i.DAY || ' ' ||
i.start_hr || ' ' || i.end_hr);
    END LOOP;

end;
/

begin

  PRINT_TIMESLOT;

end;
/
```

Explanation: First a procedure is created, then inside the "begin" block, we iterate through the query that finds the time slots of teachers and print them.

Task 3:

Analysis: The problem requires writing a procedure that finds N advisors and their details who has the highest number of students under their advising.

Working code:

```
CREATE OR REPLACE
PROCEDURE ADVISOR_INFO(N in number)
AS
row number(5);
instructor_name varchar2(20);
dept varchar2(20);
```

```
begin
  SELECT MAX(ROWNUM) INTO row
  FROM(SELECT i_id, COUNT(s_id) AS stdnts FROM advisor GROUP BY i_id ORDER
BY stdnts DESC);

  if (N>row) then
    DBMS_OUTPUT.PUT_LINE('out of bounds');
    RETURN;
  end if;

  FOR I IN(SELECT * FROM(SELECT i_id, COUNT(s_id) AS stdnts FROM advisor
GROUP BY i_id ORDER BY stdnts DESC) WHERE ROWNUM<=N) loop
    SELECT name INTO instructor_name
    FROM instructor
    WHERE I.i_id = instructor.ID;

    SELECT dept_name INTO dept
    from instructor
    WHERE I.i_id = instructor.ID;

    DBMS_OUTPUT.PUT_LINE(instructor_name || ' ' || dept || ' ' ||
i.stdnts);
  end loop;
end;
/
```

Explanation: First procedure and variables are declared. The procedure takes N as input. Then inside the "begin" block we run a query to find the number of instructors and store it inside the row variable. Then we iterate through a query that gives id of instructors who have highest students in descending order. Inside the loop, we conduct two different queries to store instructor name and dept in variables. Then we print them.

Task 4:

Analysis: The problem requires creating a trigger that automatically generates IDs for student when data is inserted in the student table.

Working code:

```
CREATE SEQUENCE STUDENT_SEQ
MINVALUE 10000
MAXVALUE 99999
START WITH 10000
INCREMENT BY 1
CACHE 20;
CREATE OR REPLACE
TRIGGER STUDENT_ID_GENERATOR
BEFORE INSERT ON student
FOR EACH ROW
DECLARE
    NEW_ID student.ID% TYPE ;
BEGIN
    SELECT STUDENT_SEQ . NEXTVAL INTO NEW_ID
    FROM DUAL ;
    :NEW.ID := NEW_ID ;
END ;
/
```

Explanation: We know that code blocks inside triggers are executed when a particular DDL or DML action has been done. To create a trigger for the given problem, first we need to initiate a sequence. The sequence has a minimum value of 10000 and maximum value of 99999. It starts from minimum value and increments by 1 after every trigger executes. Then we create the trigger like given in the code. As it is a row level trigger, we have to define it for each row before the declare block. We also have to declare whether we want the trigger to be executed before the operation or after the operation. Here we need the trigger to execute before the insert operation. Inside the begin block, we select the sequence value and assign it to the id variable. Here, "NEW.ID" is the id value that will be inserted into the student table.

Task 5:

Analysis: The problem requires creating a trigger that will automatically assign an advisor to a newly admitted student of his/her own department.

Working code:

```
CREATE OR REPLACE
    TRIGGER ASSIGN_ADVISOR
    AFTER INSERT ON STUDENT
    FOR EACH ROW
declare
    INS_ID INSTRUCTOR.ID% TYPE ;
begin
    SELECT ID INTO INS_ID
    FROM(
        SELECT ID
        FROM INSTRUCTOR I
        WHERE I.DEPT_NAME = :NEW.DEPT_NAME
    )
    WHERE ROWNUM<=1;
    INSERT INTO ADVISOR VALUES(INS_ID, :NEW.ID);

end ;
/
```

Explanation: First a trigger is declared which will be executed after insert operation on the student table. Inside the begin block, we execute a query that selects instructor id based on the department of the newly admitted student and store it inside the INS_ID variable. Then we insert the instructor id and newly admitted student id inside the advisor table.

Overall problems: Understanding how trigger and sequence work was the hardest part. I had difficulties understanding how new values and old values work. Using dual instead of a table is still unclear to me. Other concepts were easier to understand, so no problem occurred there.