Database Management Systems Lab
Lab 10
CSE 4308

A Z Hasnain Kabir
200042102
Software Engineering

17 November 2022

Introduction

We are given a database with already inserted values where we will execute some PL/SQL queries.

Task 1

Decrease the budget of the departments having a budget of more than 99999 by 10% Show the number of departments that did not get affected.

Analysis of the problem

It is easy to update department budgets through PL/SQL queries and loop through for keeping a count of departments that did not get updated;

Code

```
1 CREATE OR REPLACE PROCEDURE BUDGET_INC
2 AS
3     X NUMBER DEFAULT 0;
4 BEGIN
5     UPDATE DEPARTMENT SET BUDGET = BUDGET - 0.10*BUDGET
        WHERE BUDGET > 99999;
```

```
6
7       FOR I IN (SELECT * FROM DEPARTMENT) LOOP
8           IF I.BUDGET <= 99999 THEN
9               X := X + 1;
10          END IF;
11      END LOOP;
12
13      DBMS_OUTPUT.PUT_LINE(X);
14  END;
15  /
16
17  BEGIN
18      BUDGET_INC;
19  END;
20  /
```

Explanation of the solution

We have used update statement to make changes to the department attribute and used for loop to keep track of the unchanged department number.

Findings

We have found out that the value of X is slightly off due to the fact that we are working with the changed values of department budget. We had to work with unchanged values of department budget to keep perfect track of everything.

Task 2

Write a procedure that takes the day of the week and starting hour and ending hour as input and prints which instructors should be in the class during that time.

Analysis of the problem

We are to create a procedure with one input and three outputs and then use DBMS output to print our desired values.

Code

```
1  CREATE OR REPLACE PROCEDURE INSTRUCTOR_TIME(WEEK_DAY IN
      varchar2, START_TIME IN NUMBER, END_TIME IN NUMBER)
2  AS
3  BEGIN
4
5      FOR ROW IN (
6      SELECT NAME FROM instructor WHERE ID IN
7      (SELECT ID FROM teaches WHERE course_id IN
8      (SELECT course_id FROM SECTION WHERE time_slot_id =
9      (SELECT time_slot_id FROM time_slot T WHERE day =
          SUBSTR(WEEK_DAY, 1,2) AND
10     T.start_hr = START_TIME
11     AND T.end_hr = END_TIME)))) LOOP
12
13         DBMS_OUTPUT.PUT_LINE(ROW.NAME);
14
15     END LOOP;
16
17 END;
18 /
```

Explanation of the solution

We used nested queries to get the names of the instructor then use a loop to print out the names.

Findings

We can use SUBSTR() function to extract the first character of the name of the week days as the first characters are stored as weekday in time_slot table.

Task 3

Write a procedure to find the top N students based on the number of courses they are enrolled in. The procedure should take N as input and print the ID, name, department name, and the number of courses taken by the student.

Analysis of the problem

We can use simultaneous group by, order by, rownum to find the number of top N students.

Code

```
CREATE OR REPLACE PROCEDURE TOP_N_STUDENTS(N IN NUMBER)
AS
BEGIN
    FOR ROW IN (
        SELECT ID, max(NAME)AS ST_NAME, max(DEPT_NAME) AS
           ST_DEPT
        , count(course_id) AS COURSES
        FROM (SELECT * FROM student NATURAL JOIN takes)
           WHERE ROWNUM <=N GROUP BY
        ID ORDER BY COURSES DESC
    ) LOOP

        DBMS_OUTPUT.PUT_LINE(ROW.ID || ' ' || ROW.ST_NAME
           || ' '|| ROW.ST_DEPT ||' '|| ROW.COURSES);
    END LOOP;

END;
/
```

Explanation of the solution

We have used natural join to join takes and student table then select our desired attributes after subsequent group by, order by and rownum.

Findings

In case of printing many attribute side by side, we can use —— to separate variables from characters.

1 Task 4

Create a trigger that automatically generates IDs for instructors when we insert data into INSTRUCTOR table.

Analysis of the problem

We have to create a sequence first which will increment ID by 1.

Code

```
CREATE SEQUENCE SEQ
MINVALUE 0001
MAXVALUE 9999
START WITH 1
INCREMENT BY 1
CACHE 20;

CREATE OR REPLACE TRIGGER ST_ID_GENERATOR
BEFORE INSERT ON student
FOR EACH ROW
BEGIN
    SELECT SEQ.NEXTVAL INTO :NEW.ID FROM DUAL;
END;
/
```

Explanation of the solution

We have created a sequence to increment values of ID by 1 then used that sequence on our trigger which works before each row is inserted into student.

Task 5

Create a trigger that will automatically assign an advisor to a newly admitted student of his/her own department. In case there are multiple teachers, the advisor should be selected based on the least number of students advised.

Analysis of the problem

Contrary to Task 4, we do not have to create anything before the trigger and can proceed to the trigger immediately.

Code

```
1  CREATE OR REPLACE TRIGGER ADVISORY
2  BEFORE INSERT ON
3      student
4      FOR EACH ROW
5  DECLARE
6      INSTRUCTOR_ID advisor.i_id%TYPE;
7  BEGIN
8      SELECT i_id INTO INSTRUCTOR_ID FROM (SELECT i_id FROM
           advisor WHERE s_id<>:NEW.ID)
9      WHERE ROWNUM<=1;
10     INSERT INTO advisor(i_ID, s_id) values(INSTRUCTOR_ID, :
           NEW.ID);
11 END;
12 /
```

Explanation of the solution

We have created a trigger before insertion on student table where we have selected instructor to insert into advisor table whenever a new student is inserted.