Курсова работа

по Бази данни: Big Data

Изготвил: Петър Димитров Попов

Класно отделение: 12282

Факултетен номер: 111-7419

Вариант 02_6.

Съдържание

0. Подготвителни процедури и функции

Създаване на процедура DROP_TABLE_IF_EXISTS за по-лесно изтриване на таблици, с обработка на грешка при липса на наличието им.

```
CREATE OR REPLACE PROCEDURE DROP_TABLE_IF_EXISTS(param_table_name IN
ALL_TAB_COLUMNS.TABLE_NAME%TYPE)
```

```
    IS
BEGIN
    EXECUTE IMMEDIATE 'DROP TABLE ' || param_table_name;
EXCEPTION
    WHEN OTHERS THEN
        IF SQLCODE != -942 THEN
            RAISE;
        END IF;
END;
```

1. *Създайте функция GET_YEARS_SERVICE за извличане на брой години служба за определен служител. Функцията трябва да очаква аргумент – номер на служител и да връща брой години служба. Добавете грешка, отчитаща невалиден номер на служител. Използвайте функции MONTHS_BETWEEN и ROUND.*

```
CREATE OR REPLACE FUNCTION GET_YEARS_SERVICE(input_employee_id IN
NUMBER)
    RETURN NUMBER
    IS
    var_years_of_service NUMBER;
BEGIN
    SELECT round(months_between(sysdate, HIRE_DATE) / 12.0)
    INTO var_years_of_service
    FROM EMPLOYEES
    WHERE EMPLOYEE_ID = input_employee_id;

    RETURN var_years_of_service;

EXCEPTION
    WHEN no_data_found THEN
        DBMS_OUTPUT.PUT_LINE('Employee with id ' || input_employee_id
|| ' does not exist!');
        RETURN -1;
END;
```

Код за тестването на функцията

```
DECLARE
    var_employee_id      NUMBER := &emp_id;
    var_years_of_service NUMBER := GET_YEARS_SERVICE(var_employee_id);
BEGIN
```

```
    IF var_years_of_service <> -1 THEN
        DBMS_OUTPUT.PUT_LINE('Years of service of employee number ' ||
var_employee_id || ': ' || var_years_of_service);
    END IF;
END ;
```

- Тест със съществуващо ID - 100

```
Years of service of employee number 100: 17
```

- Тест със несъществуващо ID - 300

```
Employee with id 300 does not exist!
```

2. *Създайте PL/SQL блок за извличане на данни за служители от таблица employees с EMPLOYEE_ID между 100 и 108. Със стойностите, извлечени от таблица employees, попълнете PL/SQL таблица със стаж (брой години служба) по-голям от15. Изведете информация от PL/SQL таблицата. Използвайте функцията GET_YEARS_SERVICE.*

```
DECLARE
    TYPE EMPLOYEES_TABLE_TYPE IS TABLE OF employees%ROWTYPE INDEX BY
BINARY_INTEGER;
    employees_table             EMPLOYEES_TABLE_TYPE;
    var_employee_row            employees%ROWTYPE;
    var_employees_table_indexes NUMBER := 0;
BEGIN
    FOR counter IN 100..108
        LOOP
            SELECT *
            INTO var_employee_row
            FROM EMPLOYEES
            WHERE EMPLOYEE_ID = counter;

            -- for debugging --
            DBMS_OUTPUT.PUT_LINE(var_employee_row.HIRE_DATE);

DBMS_OUTPUT.PUT_LINE(GET_YEARS_SERVICE(var_employee_row.EMPLOYEE_ID));
            -------------------

            IF GET_YEARS_SERVICE(var_employee_row.EMPLOYEE_ID) > 15
THEN
                var_employees_table_indexes :=
var_employees_table_indexes + 1;
                employees_table(var_employees_table_indexes) :=
var_employee_row;
```

```
            END IF;
        END LOOP;

    FOR counter IN 1..var_employees_table_indexes
        LOOP
            DBMS_OUTPUT.PUT_LINE('Employee number ' ||
employees_table(counter).EMPLOYEE_ID ||
                                 ' (Name: ' ||
employees_table(counter).FIRST_NAME || ')' ||
                                 ' is working here for ' ||
GET_YEARS_SERVICE(employees_table(counter).EMPLOYEE_ID) ||
                                 ' years');
        END LOOP;
END;
```

- Проследяващи логове

```
2003-06-17
17
2005-09-21
15
2001-01-13
20
2006-01-03
15
2007-05-21
14
2005-06-25
15
2006-02-05
15
2007-02-07
14
2002-08-17
18
```

- Резултат

```
Employee number 100 (Name: Steven) is working here for 17 years
Employee number 102 (Name: Lex) is working here for 20 years
Employee number 108 (Name: Nancy) is working here for 18 years
```

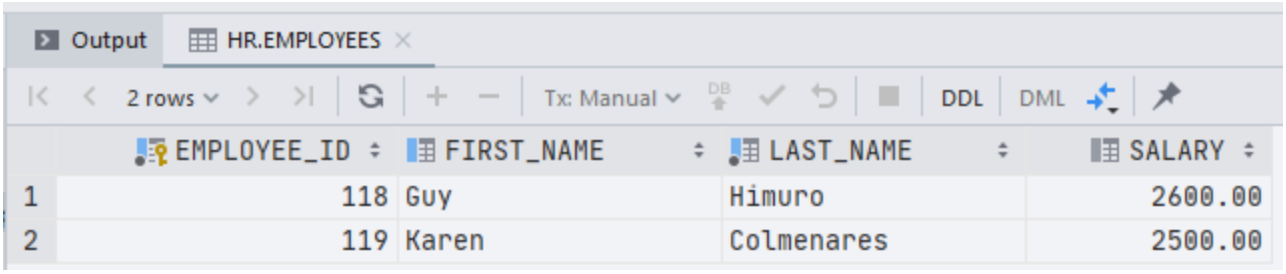
3. *Напишете анонимен PL/SQL блок с параметричен курсор (параметър - номер на отдел) за номер на служител, номер на отдел за служители със заплата по-малка от 2700 от таблица emp (копие на таблица employees). Като стойност на параметъра използвайте отдел 30 и променете номера на отдела с нов номер 20 и увеличете заплатите с 5%.*

```
BEGIN
    DROP_TABLE_IF_EXISTS('emp');
END;

CREATE TABLE emp AS (SELECT *
                     FROM EMPLOYEES);

DECLARE
    CURSOR cursor_c (param_department_id emp.department_id%TYPE) IS
        SELECT EMPLOYEE_ID, DEPARTMENT_ID
        FROM emp
        WHERE DEPARTMENT_ID = param_department_id
          AND SALARY < 2700
            FOR UPDATE OF SALARY, DEPARTMENT_ID;
    var_result_row cursor_c%ROWTYPE;
BEGIN
    OPEN cursor_c(30);
    LOOP
        FETCH cursor_c INTO var_result_row;
        EXIT WHEN cursor_c%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE(var_result_row.EMPLOYEE_ID);

        UPDATE emp
        SET SALARY        = SALARY * 1.05,
            DEPARTMENT_ID = 20
        WHERE CURRENT OF cursor_c;
    END LOOP;
    CLOSE cursor_c;
    COMMIT;
END;
```

Променени са заплатите на служители с ID 118 и 119:

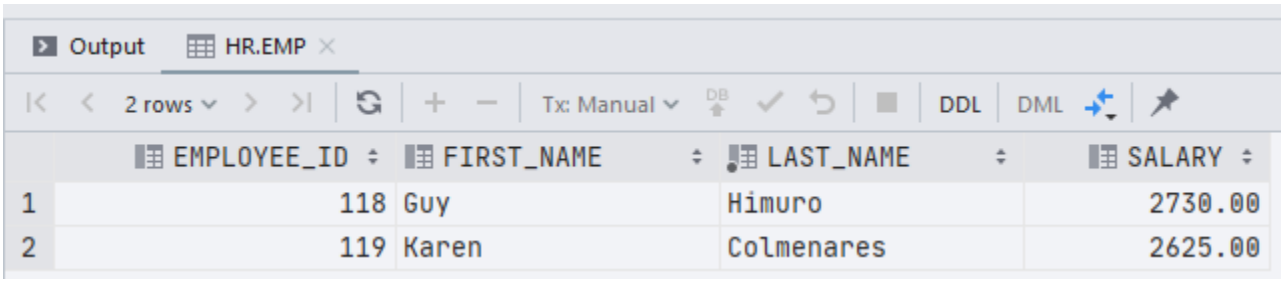
```
118
119
```

- HR.EMPLOYEES



	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY
1	118	Guy	Himuro	2600.00
2	119	Karen	Colmenares	2500.00

- HR.EMP



	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY
1	118	Guy	Himuro	2730.00
2	119	Karen	Colmenares	2625.00

4. *Създайте съхранена процедура ADD_DEPARTMENT за добавяне на ред в таблица DEPT (копие на DEPARTMENTS), която да има 4 параметъра. Първият параметър е за department_id, а следващите са за department_title, локацията и номера на началника на отдела. Да се прави проверка за наличието на такъв номер на отдел, за наличието на такова име на отдел, за наличието на локацията в таблица locations и наличието на номер на служител в таблица employees както и служителят да не е началник на друг отдел. Ако изискванията не са изпълнени, да се вдигнат съответни изключения и за тези данни да се създаде запис в помощна таблица ERROR_DEPART със структурата на таблица DEPARTMENTS.*

а) създаване на таблица DEPT и ERROR_DEPART

```
BEGIN
    DROP_TABLE_IF_EXISTS('DEPT');
END;

CREATE TABLE DEPT AS (SELECT *
```

```
                        FROM DEPARTMENTS);

BEGIN
    DROP_TABLE_IF_EXISTS('ERROR_DEPART');
END;

CREATE TABLE ERROR_DEPART AS (SELECT *
                              FROM DEPARTMENTS
                              WHERE 1 = 2);
```

б) създаване на функциите за проверка на наличност на записи в отделните таблици

```
CREATE OR REPLACE FUNCTION DEPARTMENT_EXISTS(input_department_id IN
DEPT.DEPARTMENT_ID%TYPE)
    RETURN BOOLEAN
    IS
    var_department_id DEPT.DEPARTMENT_ID%TYPE;
BEGIN
    SELECT DEPARTMENT_ID
    INTO var_department_id
    FROM DEPT
    WHERE DEPARTMENT_ID = input_department_id;

    RETURN TRUE;

EXCEPTION
    WHEN no_data_found THEN
        RETURN FALSE;
END;

CREATE OR REPLACE FUNCTION
IS_DEPARTMENT_NAME_AVAILABLE(input_department_name IN
DEPT.DEPARTMENT_NAME%TYPE)
    RETURN BOOLEAN
    IS
    var_department_name DEPT.DEPARTMENT_NAME%TYPE;
BEGIN
    SELECT DISTINCT DEPARTMENT_NAME
    INTO var_department_name
    FROM DEPT
    WHERE DEPARTMENT_NAME = input_department_name;

    RETURN FALSE;
```

```
EXCEPTION
    WHEN no_data_found THEN
        RETURN TRUE;
END;

CREATE OR REPLACE FUNCTION LOCATION_EXISTS(input_location_id IN
LOCATIONS.LOCATION_ID%TYPE)
    RETURN BOOLEAN
    IS
    var_location_id LOCATIONS.LOCATION_ID%TYPE;
BEGIN
    SELECT LOCATION_ID
    INTO var_location_id
    FROM LOCATIONS
    WHERE LOCATION_ID = input_location_id;

    RETURN TRUE;

EXCEPTION
    WHEN no_data_found THEN
        RETURN FALSE;
END;

CREATE OR REPLACE FUNCTION EMPLOYEE_EXISTS(input_employee_id IN
EMPLOYEES.EMPLOYEE_ID%TYPE)
    RETURN BOOLEAN
    IS
    var_employee_id EMPLOYEES.EMPLOYEE_ID%TYPE;
BEGIN
    SELECT EMPLOYEE_ID
    INTO var_employee_id
    FROM EMPLOYEES
    WHERE EMPLOYEE_ID = input_employee_id;

    RETURN TRUE;

EXCEPTION
    WHEN no_data_found THEN
        RETURN FALSE;
END;

CREATE OR REPLACE FUNCTION IS_EMPLOYEE_MANAGER(input_employee_id IN
EMPLOYEES.EMPLOYEE_ID%TYPE)
    RETURN BOOLEAN
    IS
    var_employee_id EMPLOYEES.EMPLOYEE_ID%TYPE;
```

```
BEGIN
    SELECT DISTINCT MANAGER_ID
    INTO var_employee_id
    FROM EMPLOYEES
    WHERE MANAGER_ID = input_employee_id;

    RETURN TRUE;

EXCEPTION
    WHEN no_data_found THEN
        RETURN FALSE;
END;
```

в) създаване на процедурата ADD_DEPARTMENT

```
CREATE OR REPLACE PROCEDURE ADD_DEPARTMENT(
        param_department_id IN DEPT.department_id%TYPE,
        param_department_title IN DEPT.department_name%TYPE,
        param_location_id IN DEPT.location_id%TYPE,
        param_manager_id IN DEPT.manager_id%TYPE)
    IS
    department_exists_exc EXCEPTION;
    department_name_not_avail_exc EXCEPTION;
    location_does_not_exist_exc EXCEPTION;
    employee_does_not_exist_exc EXCEPTION;
    employee_is_manager_exc EXCEPTION;

    PRAGMA EXCEPTION_INIT ( department_exists_exc, -20001 );
    PRAGMA EXCEPTION_INIT ( department_name_not_avail_exc, -20002 );
    PRAGMA EXCEPTION_INIT ( location_does_not_exist_exc, -20003 );
    PRAGMA EXCEPTION_INIT ( employee_does_not_exist_exc, -20004 );
    PRAGMA EXCEPTION_INIT ( employee_is_manager_exc, -20005 );
BEGIN

    IF DEPARTMENT_EXISTS(param_department_id)
    THEN
        raise_application_error(-20001, 'Duplicate of PK
(Department_Id) ' || param_department_id);
    END IF;

    IF IS_DEPARTMENT_NAME_AVAILABLE(param_department_title) = FALSE
    THEN
        raise_application_error(-20002, 'Duplicate of unique column
(Department_Name) ' || param_department_title);
    END IF;
```

```
    IF LOCATION_EXISTS(param_location_id) = FALSE
    THEN
        raise_application_error(-20003, 'No corresponding FK exist
(Location_Id) ' || param_location_id);
    END IF;

    IF EMPLOYEE_EXISTS(param_manager_id) = FALSE
    THEN
        raise_application_error(-20004, 'No corresponding FK exist
(Employee_Id) ' || param_manager_id);
    END IF;

    IF IS_EMPLOYEE_MANAGER(param_manager_id)
    THEN
        raise_application_error(-20005, 'Employee is already manager
(Employee_Id) ' || param_manager_id);
    END IF;

    INSERT INTO
        DEPT (department_id, department_name, manager_id, location_id)
    VALUES (param_department_id, param_department_title,
param_manager_id, param_location_id);

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(sqlerrm);
        INSERT INTO
            ERROR_DEPART(department_id, department_name, manager_id,
location_id)
        VALUES (param_department_id, param_department_title,
param_manager_id, param_location_id);

        COMMIT;
END;
```

г) тестване на процедурата ADD_DEPARTMENT

- процедурата трябва да работи както е очаквано

```
     BEGIN
    ADD_DEPARTMENT(
            280,
            'Politics',
```

```
            1000,
            104);
    -- All correct
END;
```

```
HR> BEGIN
        ADD_DEPARTMENT(
                280,
                'Politics',
                1000,
                104);
        -- All correct
    END;
[2020-12-09 21:02:19] completed in 106 ms
```

- Процедурата трябва да добавя запис в ERROR_DEPART, с вдигане на съобщение за грешка при дублиран primary key (дублаж на ID на съществуващ отдел)

```
BEGIN
    ADD_DEPARTMENT(
            100,
            'Politics2',
            1000,
            105);
    -- Existing department id
END;
```

```
ORA-20001: Duplicate of PK (Department_Id) 100
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
100	Politics2	105	1000

- Процедурата трябва да добавя запис в ERROR_DEPART, с вдигане на съобщение за грешка при дублирана уникална колона (дублаж на име на съществуващ отдел)

```
BEGIN
    ADD_DEPARTMENT(
            290,
```

```
            'Executive',
            1000,
            106);
    -- Duplicate name
END;
```

```
ORA-20002: Duplicate of unique column (Department_Name) Executive
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
290	Executive	106	1000

- Процедурата трябва да добавя запис в ERROR_DEPART, с вдигане на съобщение за грешка липса на съответстващ foreign key (несъществуващо местоположение)

```
BEGIN
    ADD_DEPARTMENT(
            300,
            'Politics3',
            3300,
            107);
    -- Not existing location
END;
```

```
ORA-20003: No corresponding FK exist (Location_Id) 3300
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
300	Politics3	107	3300

- Процедурата трябва да добавя запис в ERROR_DEPART, с вдигане на съобщение за грешка липса на съответстващ foreign key (несъществуващ служител)

```
BEGIN
    ADD_DEPARTMENT(
            310,
            'Politics4',
            1000,
```

```
            207);
    -- Not existing employee
END;
```

```
ORA-20004: No corresponding FK exist (Employee_Id) 207
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
310	Politics4	207	1000

- Процедурата трябва да добавя запис в ERROR_DEPART, с вдигане на съобщение за грешка нарушение на логиката (служителят е вече мениджър на друг отдел)

```
BEGIN
    ADD_DEPARTMENT(
            320,
            'Politics5',
            1000,
            100);
    -- Employee is manager
END;
```

```
ORA-20005: Employee is already manager (Employee_Id) 100
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
320	Politics5	100	1000

Резултатите в таблица ERROR_DEPART след изпълнението на всички тестове

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
100	Politics2	105	1000
290	Executive	106	1000
300	Politics3	107	3300
310	Politics4	207	1000
320	Politics5	100	1000