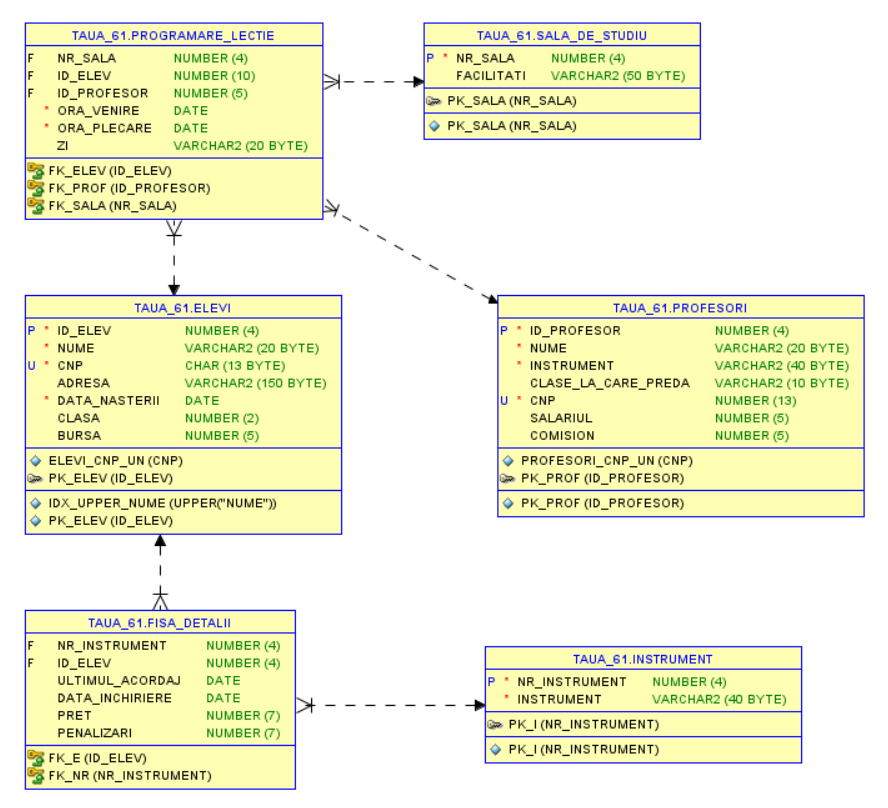
***Proiect Școala de Muzică***



**Tema proiectului**

Proiectul a cărei schemă este prezentată mai jos reprezinta baza de date a unei scoli de muzică.

**Descrierea problemei**

Proiectul are în evidență activitatea unei școli de muzică pentru a fi mai usor de ținut evidența organizării scolii. În acest fel gestinoarea timpului va fi mult mai eficientă.

Modelul e format din 6 tabele care conțin informații legate de elevi, profesor, instrumente si salile de studiu.

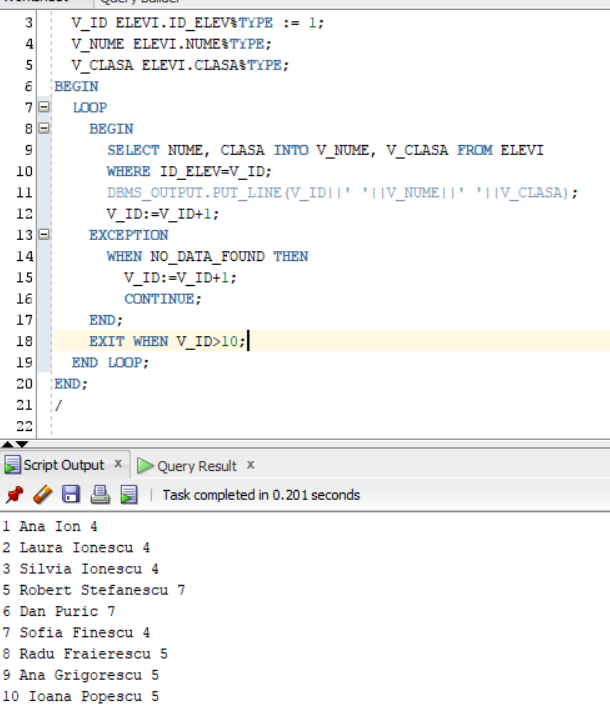
Primul tabel denumit **ELEVI** contine id-ul unic fiecarui elev, numele, CNP-ul, adresa și clasa în care se află elevii și bursa pe care o iau (dacă este cazul). Acest tabel este legat de tabelele „PROGRAMARE\_LECTIE” și de „FISA\_DETALII” prin relații de tip one to many.

Tabelul **INSTRUMENT** contine numărul intrumentului pentru a fi identificat usor, numele acestuia și este direct legat de tabelul „FISA\_DETALII” printr-o relație one to many ce conține informații despre ultimul acordaj, data la care elevul a inchiriat instrumentul, prețul pe lună pentru inchiriere și o secțiune de penalizări în cazul în care elevul nu platește la timp sau dacă deteriorează instrumentul.

Tabelul **PROFESORI** conține id-ul unic al profesorului, numele, instrumentul pe care îl predă, clasele la care predă, salariul si comisionul din salariu. Tabelul este legat printr-o relație one to many de „PROGRAMARE\_LECTII”. Acesta contine numarul sălii în care se va desfășura lecția, id-ul elevului care va veni, al profesorului, ora de venire si ora de plecare. Acest tabel este legat de „SALA\_DE\_STUDIU” printr-o relație many to one ce conține numărul sălii și facilitățile de acolo (spre exemplu: conține pian). În funcție de facilități se poate decide ce lecții se vor ține acolo și care nu.

**--Sa se afiseze toti elevi cu id ul intre 1 si 10, afisand numele si clasa acestora folosind structura repetitiva LOOP.**

SET SERVEROUTPUT ON

DECLARE

V\_ID ELEVI.ID\_ELEV%TYPE := 1;

V\_NUME ELEVI.NUME%TYPE;

V\_CLASA ELEVI.CLASA%TYPE;

BEGIN

LOOP

BEGIN

SELECT NUME, CLASA INTO V\_NUME, V\_CLASA FROM ELEVI

WHERE ID\_ELEV=V\_ID;

DBMS\_OUTPUT.PUT\_LINE(V\_ID||' '||V\_NUME||' '||V\_CLASA);

V\_ID:=V\_ID+1;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

V\_ID:=V\_ID+1;

CONTINUE;

END;

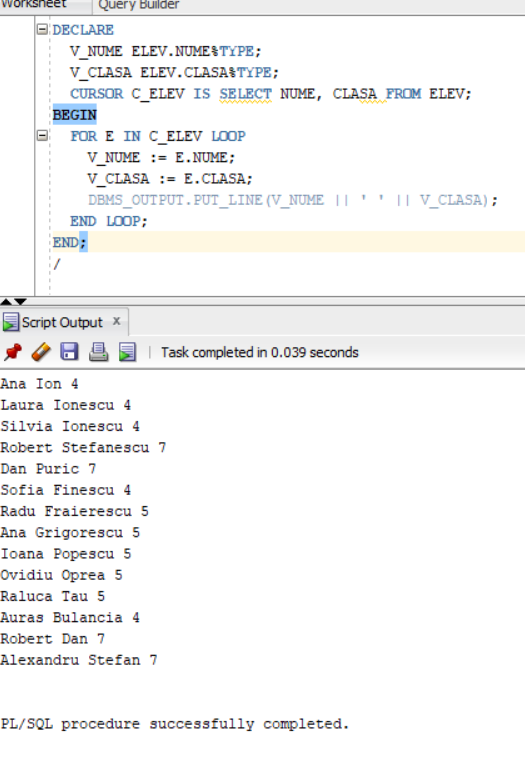
EXIT WHEN V\_ID>10;

END LOOP;

END;

/

**-- Sa se afiseze toti elevi cu id ul intre 1 si 10, afisand numele si clasa acestora folosind structura repetitive FOR.**

DECLARE

V\_NUME ELEV.NUME%TYPE;

V\_CLASA ELEV.CLASA%TYPE;

CURSOR C\_ELEV IS SELECT NUME, CLASA FROM ELEV;

BEGIN

FOR E IN C\_ELEV LOOP

V\_NUME := E.NUME;

V\_CLASA := E.CLASA;

DBMS\_OUTPUT.PUT\_LINE(V\_NUME || ' ' || V\_CLASA);

END LOOP;

END;

/

**--Sa se afiseze pentru toti elevii cu id ul intre 1 si 10 numarul instrumentului pe care il au, si penalizarile (daca le au)**

DECLARE

V\_ID FISA\_DETALII.ID\_ELEV%TYPE:=1;

V\_NR FISA\_DETALII.NR\_INSTRUMENT%TYPE;

V\_PEN FISA\_DETALII.PENALIZARI%TYPE;

BEGIN

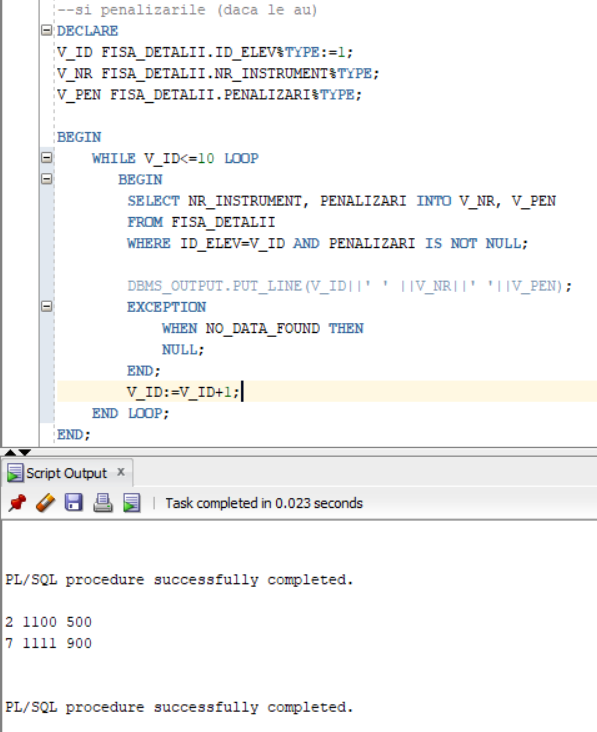
WHILE V\_ID<=10 LOOP

BEGIN

SELECT NR\_INSTRUMENT, PENALIZARI INTO V\_NR, V\_PEN

FROM FISA\_DETALII

WHERE ID\_ELEV=V\_ID AND PENALIZARI IS NOT NULL;

 DBMS\_OUTPUT.PUT\_LINE(V\_ID||' ' ||V\_NR||' '||V\_PEN);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

NULL;

END;

V\_ID:=V\_ID+1;

END LOOP;

END;

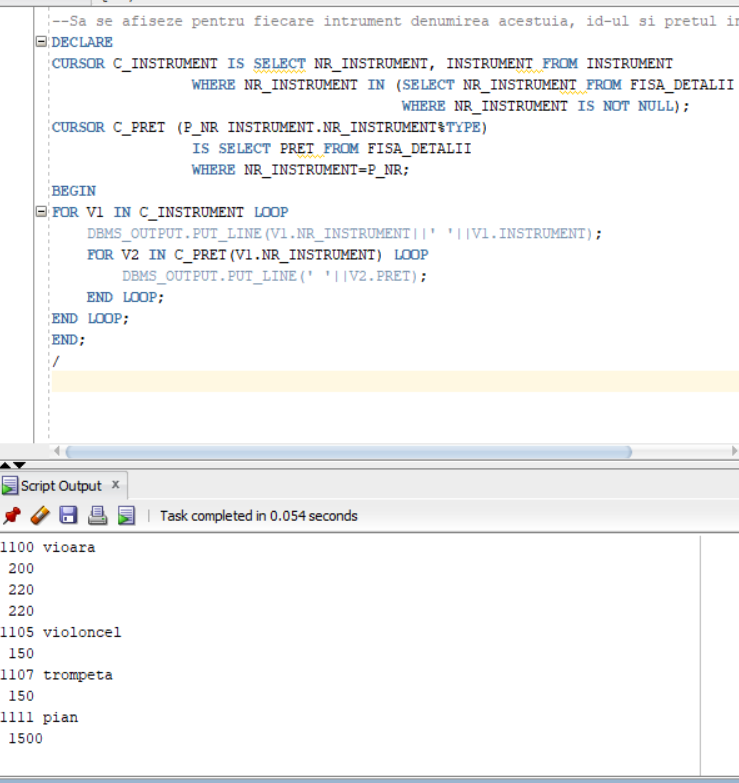
/

**--Sa se afiseze pentru fiecare intrument denumirea acestuia, id-ul si pretul inchirierii lui**

DECLARE

CURSOR C\_INSTRUMENT IS SELECT NR\_INSTRUMENT, INSTRUMENT FROM INSTRUMENT

WHERE NR\_INSTRUMENT IN (SELECT NR\_INSTRUMENT FROM FISA\_DETALII

 WHERE NR\_INSTRUMENT IS NOT NULL);

CURSOR C\_PRET (P\_NR INSTRUMENT.NR\_INSTRUMENT%TYPE)

IS SELECT PRET FROM FISA\_DETALII

WHERE NR\_INSTRUMENT=P\_NR;

BEGIN

FOR V1 IN C\_INSTRUMENT LOOP

DBMS\_OUTPUT.PUT\_LINE(V1.NR\_INSTRUMENT||' '||V1.INSTRUMENT);

FOR V2 IN C\_PRET(V1.NR\_INSTRUMENT) LOOP

DBMS\_OUTPUT.PUT\_LINE(' '||V2.PRET);

END LOOP;

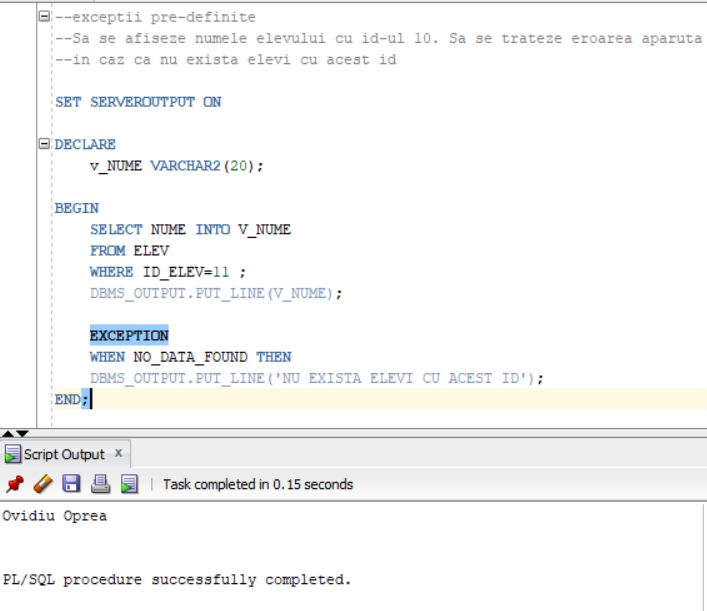
END LOOP;

END;

/

***Exceptii pre-definite***

**--Sa se afiseze numele elevului cu id-ul 10. Sa se trateze eroarea aparuta in caz ca nu exista elevi cu acest id**

SET SERVEROUTPUT ON

DECLARE

v\_NUME VARCHAR2(20);

BEGIN

SELECT NUME INTO V\_NUME

FROM ELEVI

WHERE ID\_ELEV=11 ;

DBMS\_OUTPUT.PUT\_LINE(V\_NUME);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('NU EXISTA ELEVI CU ACEST ID');

END;

**--Afisati pentru toti elevi care sunt nascuti dupa un an ce va fi citit de la tastatura, id-ul, numele si clasa in care se afla. Daca nu exista niciun elev nascut dupa anul respectiv se va afisa un mesaj corespunzator, tratati si restul erorilor ce mai pot aparea.**

DECLARE

V\_AN NUMBER;

V\_ID ELEVI.ID\_ELEV%TYPE;

V\_NUME ELEVI.NUME%TYPE;

V\_CLS ELEVI.CLASA%TYPE;

BEGIN

V\_AN := &AN\_NASTERE;

SELECT COUNT(\*)

INTO V\_ID

FROM ELEVI

WHERE EXTRACT(YEAR FROM DATA\_NASTERII) > V\_AN;

IF V\_ID = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('NU EXISTA NICIUN ELEV NASCUT DUPA ANUL ' || V\_AN);

ELSE

FOR E IN (

SELECT ID\_ELEV, NUME, CLASA

FROM ELEVI

WHERE EXTRACT(YEAR FROM DATA\_NASTERII) > V\_AN

) LOOP

V\_ID := E.ID\_ELEV;

V\_NUME := E.NUME;

V\_CLS := E.CLASA;

DBMS\_OUTPUT.PUT\_LINE(V\_ID || ' ' || V\_NUME || ' ' || V\_CLS);

END LOOP;

END IF;

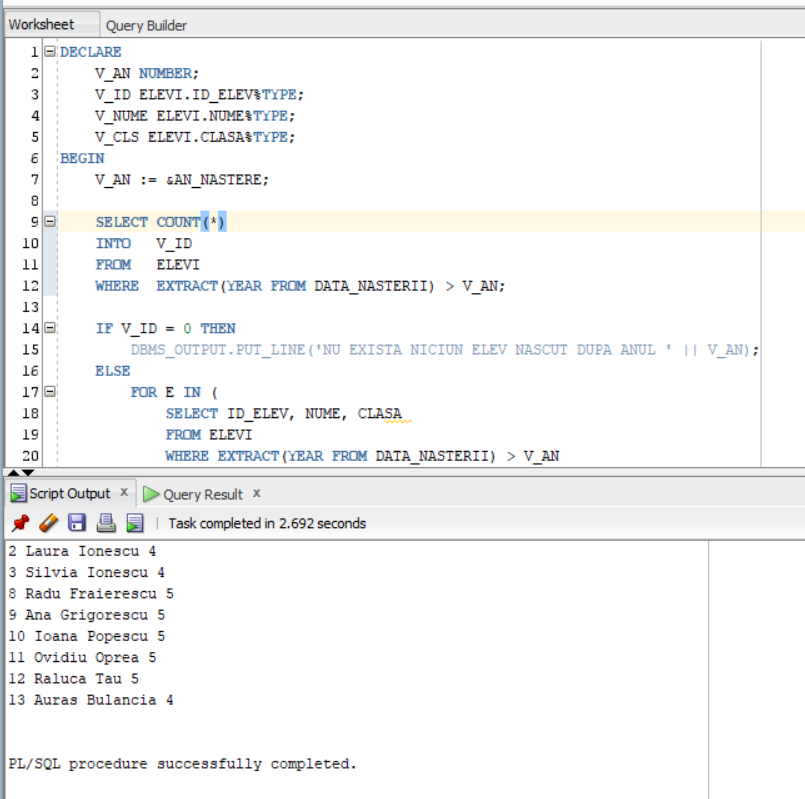
EXCEPTION

WHEN OTHERS THEN

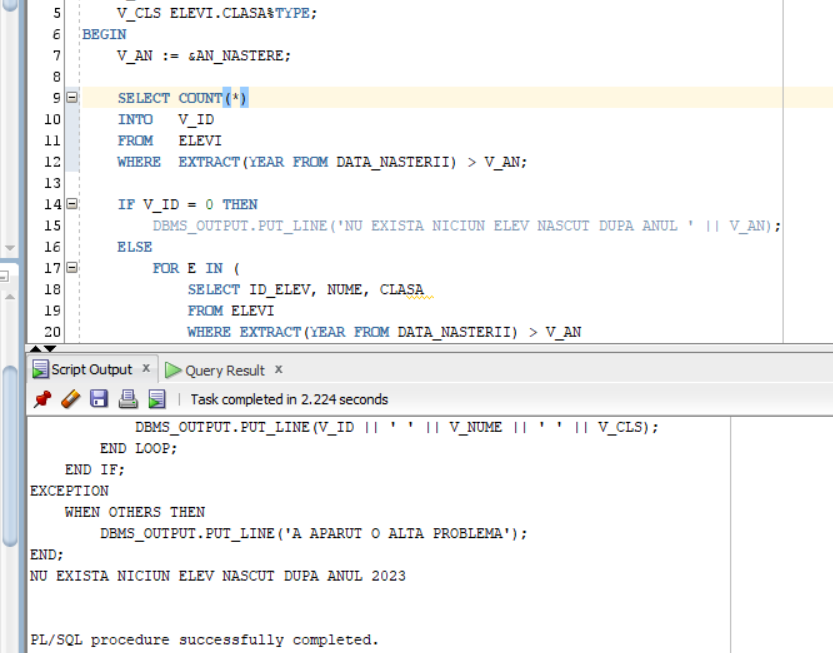
DBMS\_OUTPUT.PUT\_LINE('A APARUT O ALTA PROBLEMA');

END;

Cand introducem 2011:



Cand introducem 2023:



***Exceptii definite de utilizator***

**--Sa se actualizeze numele elevului cu id ul 23. Dacă nu se produce nicio actualizare (valoarea atributului SQL%ROWCOUNT este 0) sau dacă apare o altă excepție (clauza OTHERS) atunci să se declanşeze o excepţie prin care să fie avertizat utilizatorul.**

DECLARE

INVALID\_NUME EXCEPTION;

BEGIN

UPDATE ELEV

SET NUME='DAN'

WHERE ID\_ELEV=23;

IF SQL%NOTFOUND THEN

RAISE INVALID\_NUME;

END IF;

EXCEPTION

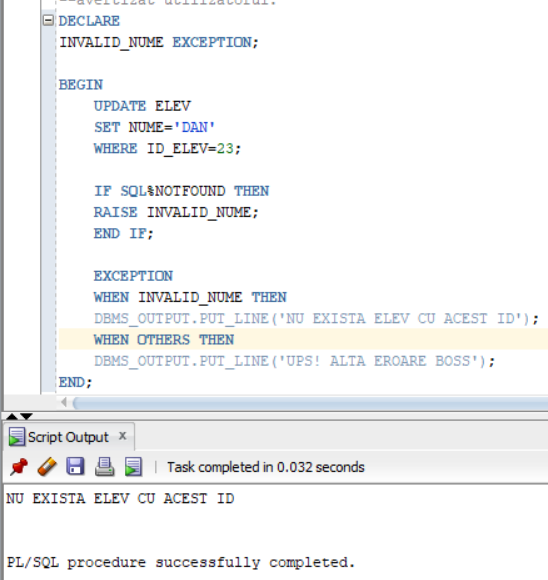
WHEN INVALID\_NUME THEN

DBMS\_OUTPUT.PUT\_LINE('NU EXISTA ELEV CU ACEST ID');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('UPS! ALTA EROARE BOSS');

END;



**--Actualizati facilitatile salii de studiu cu numarul 111, iar daca nu exista, sa se afiseze un mesaj corespunzator.**

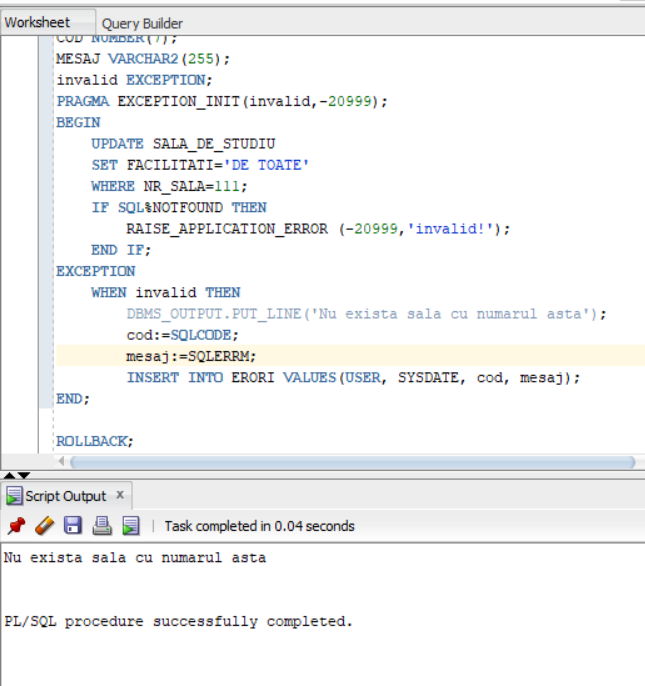
CREATE TABLE ERORI(

utilizator varchar2(32),

data\_exc date,

cod\_exc number(7),

mesaj\_exc varchar2(128));



DECLARE

COD NUMBER(7);

MESAJ VARCHAR2(255);

invalid EXCEPTION;

PRAGMA EXCEPTION\_INIT(invalid,-20999);

BEGIN

UPDATE SALA\_DE\_STUDIU

SET FACILITATI='DE TOATE'

WHERE NR\_SALA=111;

IF SQL%NOTFOUND THEN

RAISE\_APPLICATION\_ERROR (-20999,'invalid!');

END IF;

EXCEPTION

WHEN invalid THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista sala cu numarul asta');

cod:=SQLCODE;

mesaj:=SQLERRM;

INSERT INTO ERORI VALUES(USER, SYSDATE, cod, mesaj);

END;

**Tratarea excepţiilor non-predefinite oracle server**

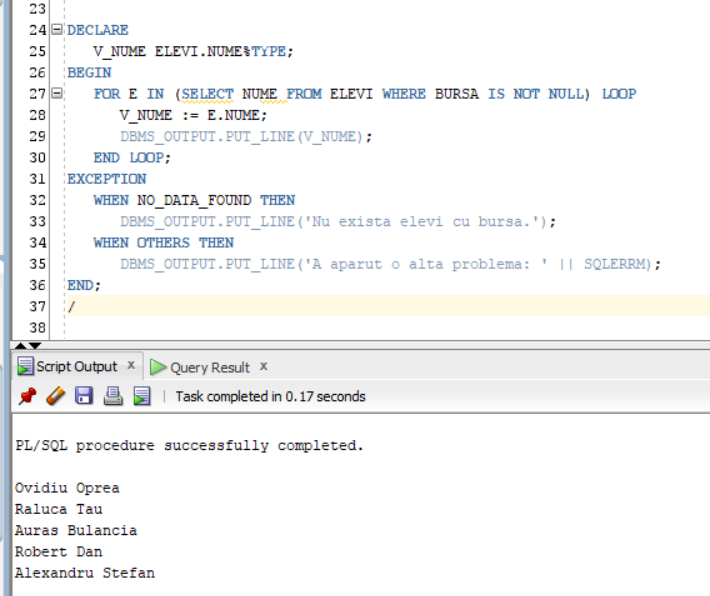
**--Sa se afiseze numele tuturor elevilor care beneficiaza de bursa. Tratati exceptiile ce pot aparea.**

DECLARE

V\_NUME ELEVI.NUME%TYPE;

BEGIN

FOR E IN (SELECT NUME FROM ELEVI WHERE BURSA IS NOT NULL) LOOP

 V\_NUME := E.NUME;

DBMS\_OUTPUT.PUT\_LINE(V\_NUME);

END LOOP;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista elevi cu bursa.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o alta problema: ' || SQLERRM);

END;

/

**Cursor implicit**

**Sa se mareasca cu 10% bursa elevilor care beneficiaza de ea, din tabela ELEVI si sa se afiseze cate randuri au fost afectate.**

DECLARE

v\_Modificari INTEGER := 0;

BEGIN

FOR elev IN (SELECT \* FROM ELEVI WHERE Bursa IS NOT NULL) LOOP

UPDATE ELEVI

SET BURSA = BURSA \* 1.1 -- Mărirea valorii bursei cu 10%

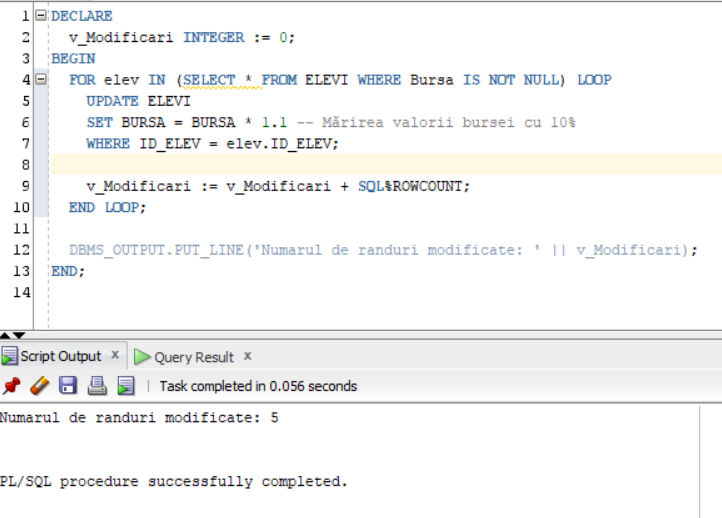
WHERE ID\_ELEV = elev.ID\_ELEV;

v\_Modificari := v\_Modificari + SQL%ROWCOUNT;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Numarul de randuri modificate: ' || v\_Modificari);

END;



**Sa se afiseze toti elevii care nu beneficiaza de bursa.**

DECLARE

v\_IDElev ELEVI.ID\_Elev%TYPE;

v\_NumeElev ELEVI.Nume%TYPE;

v\_Cursor SYS\_REFCURSOR;

BEGIN

OPEN v\_Cursor FOR

SELECT ID\_Elev, Nume

FROM ELEVI

WHERE Bursa IS NULL;

LOOP

FETCH v\_Cursor INTO v\_IDElev, v\_NumeElev;

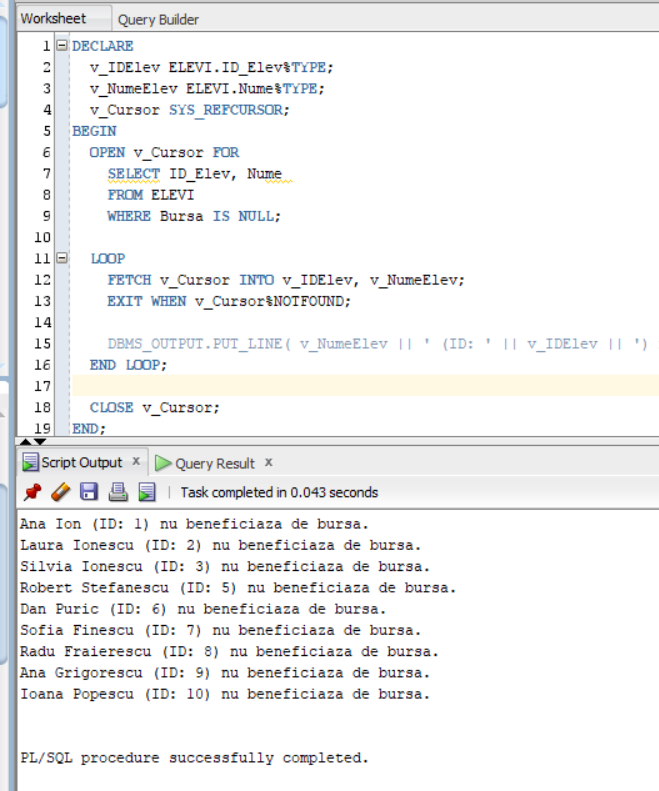
EXIT WHEN v\_Cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE( v\_NumeElev || ' (ID: ' || v\_IDElev || ') nu beneficiază de bursă.');

END LOOP;

CLOSE v\_Cursor;

END;



--Sa se afiseze numarul de profesori din tabela PROFESORI care predau 'vioara' si sa li se mareasca salariul cu 10%

DECLARE

v\_NumarProfesori INTEGER;

BEGIN

UPDATE PROFESORI

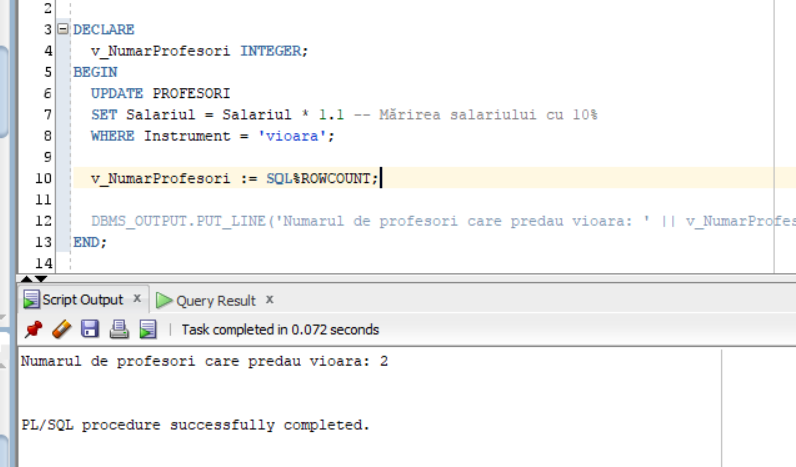
SET Salariul = Salariul \* 1.1 -- Mărirea salariului cu 10%

WHERE Instrument = 'vioara';

v\_NumarProfesori := SQL%ROWCOUNT;

DBMS\_OUTPUT.PUT\_LINE('Numarul de profesori care predau vioara: ' || v\_NumarProfesori);

END;



**Cursori expliciti**

**--Sa se creeze un bloc PL/SQL prin care sa se afiseze pentru fiecare profesor (id, nume, instrumentul pe care il preda) detalii cu privire la zilele in care preda (numarul salii in care va avea lectia + zilele)**

DECLARE

CURSOR C\_PROF IS

SELECT ID\_PROFESOR, NUME, INSTRUMENT

FROM PROFESORI

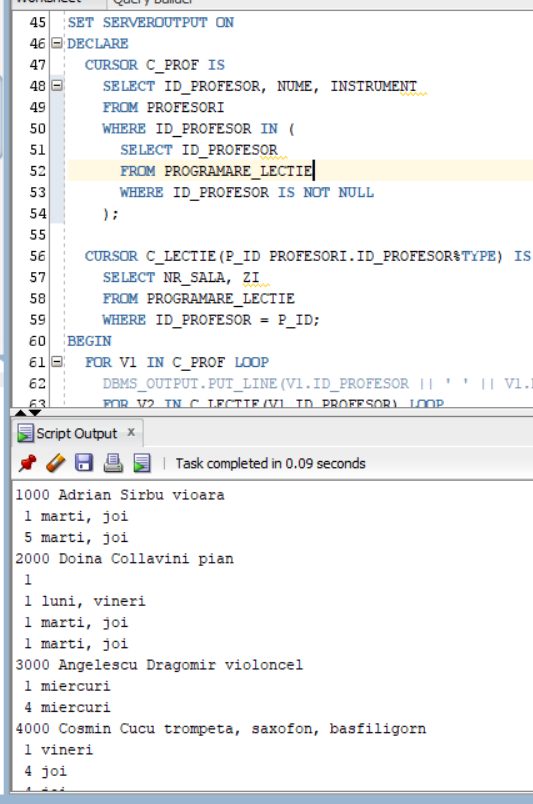
WHERE ID\_PROFESOR IN (

SELECT ID\_PROFESOR

FROM PROGRAMARE\_LECTIE

WHERE ID\_PROFESOR IS NOT NULL

);

 CURSOR C\_LECTIE(P\_ID PROFESORI.ID\_PROFESOR%TYPE) IS

SELECT NR\_SALA, ZI

FROM PROGRAMARE\_LECTIE

WHERE ID\_PROFESOR = P\_ID;

BEGIN

FOR V1 IN C\_PROF LOOP

DBMS\_OUTPUT.PUT\_LINE(V1.ID\_PROFESOR || ' ' || V1.NUME || ' ' || V1.INSTRUMENT);

FOR V2 IN C\_LECTIE(V1.ID\_PROFESOR) LOOP

DBMS\_OUTPUT.PUT\_LINE(' ' || V2.NR\_SALA || ' ' || V2.ZI);

END LOOP;

END LOOP;

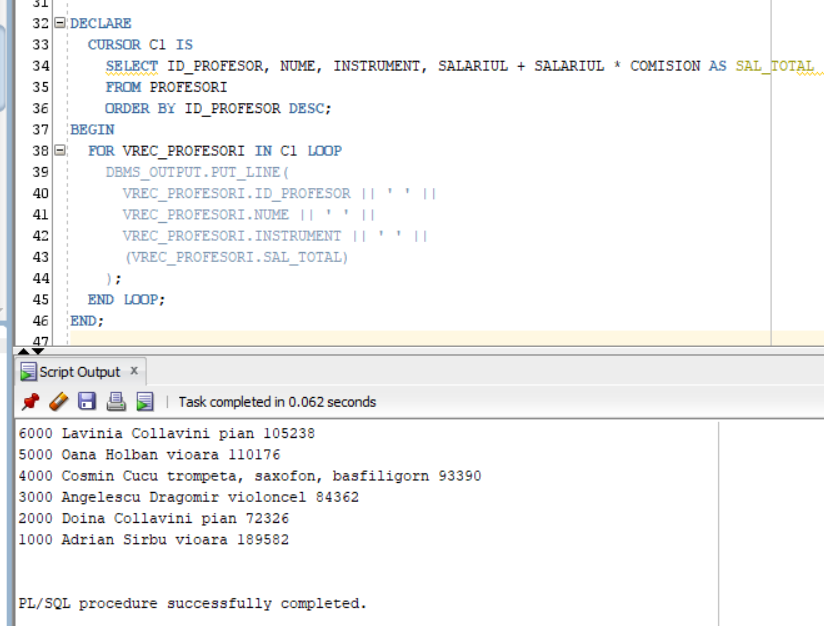
END;

**Afisati pentru fiecare profesor valoarea totala a salariului**

DECLARE

CURSOR C1 IS

SELECT ID\_PROFESOR, NUME, INSTRUMENT, SALARIUL + SALARIUL \* COMISION AS SAL\_TOTAL

 FROM PROFESORI

ORDER BY ID\_PROFESOR DESC;

BEGIN

FOR VREC\_PROFESORI IN C1 LOOP

DBMS\_OUTPUT.PUT\_LINE(

VREC\_PROFESORI.ID\_PROFESOR || ' ' ||

VREC\_PROFESORI.NUME || ' ' ||

VREC\_PROFESORI.INSTRUMENT || ' ' ||

(VREC\_PROFESORI.SAL\_TOTAL)

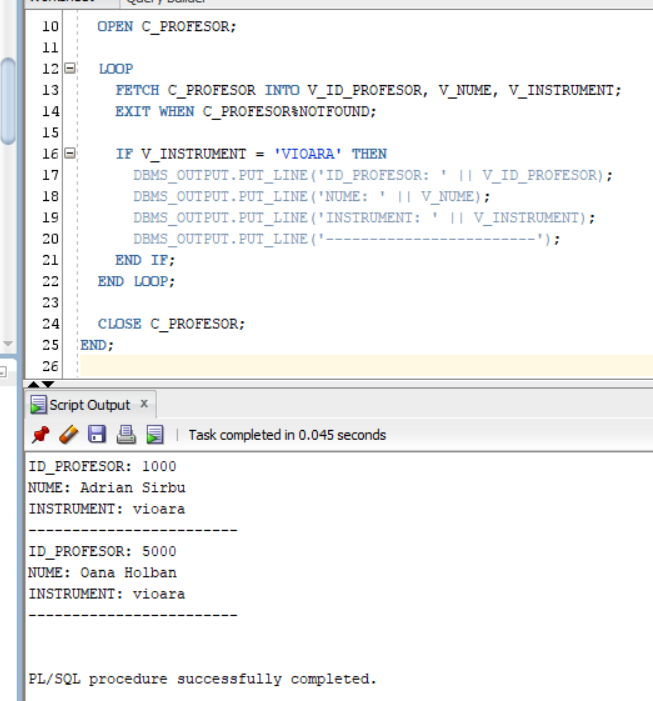
);

END LOOP;

END;

**Sa se afiseze toti profesorii care canta la vioara.**

DECLARE

 CURSOR C\_PROFESOR IS

SELECT ID\_PROFESOR, NUME, INSTRUMENT

FROM PROFESORI;

V\_ID\_PROFESOR PROFESORI.ID\_PROFESOR%TYPE;

V\_NUME PROFESORI.NUME%TYPE;

V\_INSTRUMENT PROFESORI.INSTRUMENT%TYPE;

BEGIN

OPEN C\_PROFESOR;

LOOP

FETCH C\_PROFESOR INTO V\_ID\_PROFESOR, V\_NUME, V\_INSTRUMENT;

EXIT WHEN C\_PROFESOR%NOTFOUND;

IF V\_INSTRUMENT = 'VIOARA' THEN

DBMS\_OUTPUT.PUT\_LINE('ID\_PROFESOR: ' || V\_ID\_PROFESOR);

DBMS\_OUTPUT.PUT\_LINE('NUME: ' || V\_NUME);

DBMS\_OUTPUT.PUT\_LINE('INSTRUMENT: ' || V\_INSTRUMENT);

DBMS\_OUTPUT.PUT\_LINE('------------------------');

END IF;

END LOOP;

CLOSE C\_PROFESOR;

END;

**Pachete**

-- Realizati un pachet de subprograme care sa contina:

- o functie care sa verifice daca un anumit id al unei functii exista deja in tabela Profesori si returneaza true atunci cand exista si false cand nu

- o procedura care sa angajeze un nou profesor prin a se adauga acesta in tabela Profesori. Parametrii procedurii se refera la informatiile ce trebuie adaugate

Se trateaza cazul in care exista deja un profesor cu acelasi id, folosind functia de verificare

- o procedura care sa modifice clasele la care preda un profesor. Folosim ca parametrii id-ul profesorului pentru care actualizam si clase\_la\_care\_preda

Se trateaza cazul in care nu se realizeaza modificarea din cauza faptului ca id-ul precizat nu se regaseste in tabela ( folosind functia)

- o procedura care sa concedieze un profesor prin eliminarea acestuia din tabel pe baza id-ului primit. Se va trata cazul in care id-ul furnizat nu exista

CREATE OR REPLACE PACKAGE PACHET\_SCOALA

IS

FUNCTION EXISTA\_PROF(P\_ID NUMBER) RETURN BOOLEAN;

PROCEDURE ANGAJEAZA\_PROF(P\_ID NUMBER, P\_NUME VARCHAR2, P\_INSTRUMENT VARCHAR2, P\_CNP NUMBER, P\_SALARIU NUMBER);

PROCEDURE MODIFICA\_CLS(P\_ID NUMBER, P\_NEW\_CLS VARCHAR2);

PROCEDURE CONCEDIERE\_PROF(P\_ID NUMBER);

END PACHET\_SCOALA;

/

CREATE OR REPLACE PACKAGE BODY PACHET\_SCOALA AS

FUNCTION EXISTA\_PROF(P\_ID NUMBER) RETURN BOOLEAN IS

V\_COUNT NUMBER;

BEGIN

SELECT COUNT(\*) INTO V\_COUNT FROM PROFESORI WHERE ID\_PROFESOR = P\_ID;

IF V\_COUNT > 0 THEN

RETURN TRUE;

ELSE

RETURN FALSE;

END IF;

END EXISTA\_PROF;

PROCEDURE ANGAJEAZA\_PROF(P\_ID NUMBER, P\_NUME VARCHAR2, P\_INSTRUMENT VARCHAR2, P\_CNP NUMBER, P\_SALARIU NUMBER) IS

BEGIN

IF EXISTA\_PROF(P\_ID) THEN

RAISE\_APPLICATION\_ERROR(-20001, 'EXISTĂ DEJA UN PROFESOR CU ACEST ID');

ELSE

INSERT INTO PROFESORI(ID\_PROFESOR, NUME, INSTRUMENT, CNP, SALARIU)

VALUES (P\_ID, P\_NUME, P\_INSTRUMENT, P\_CNP, P\_SALARIU);

COMMIT;

END IF;

END ANGAJEAZA\_PROF;

PROCEDURE MODIFICA\_CLS(P\_ID NUMBER, P\_NEW\_CLS VARCHAR2) IS

BEGIN

IF EXISTA\_PROF(P\_ID) THEN

UPDATE PROGRAMARE\_LECTIE

SET CLASE\_LA\_CARE\_PREDA = P\_NEW\_CLS

WHERE ID\_PROFESOR = P\_ID;

COMMIT;

ELSE

RAISE\_APPLICATION\_ERROR(-20002, 'ID PROFESOR NEVALID');

END IF;

END MODIFICA\_CLS;

PROCEDURE CONCEDIERE\_PROF(P\_ID NUMBER) IS

BEGIN

IF EXISTA\_PROF(P\_ID) THEN

DELETE FROM PROFESORI

WHERE ID\_PROFESOR = P\_ID;

COMMIT;

ELSE

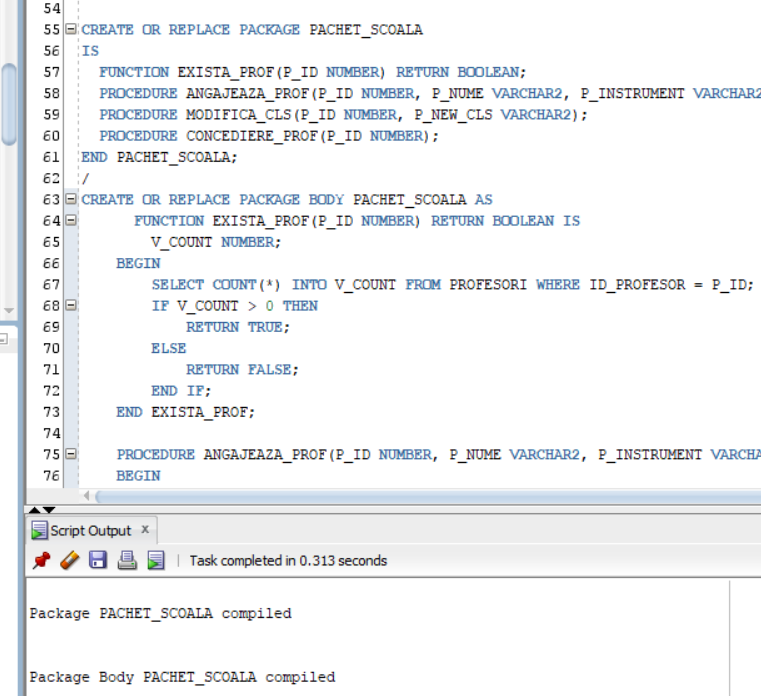
RAISE\_APPLICATION\_ERROR(-20003, 'ID PROFESOR NEVALID');

END IF;

END CONCEDIERE\_PROF;

END PACHET\_SCOALA;

/



**Apelul functiei de cautare a profesorului:**

DECLARE

V\_ID NUMBER := 701;

V\_EXISTS BOOLEAN;

BEGIN

V\_EXISTS := PACHET\_SCOALA.EXISTA\_PROF(V\_ID);

IF V\_EXISTS THEN

DBMS\_OUTPUT.PUT\_LINE('PROFESORUL EXISTA ÎN BAZA DE DATE.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('PROFESORUL NU EXISTA ÎN BAZA DE DATE.');

END IF;

END;

/

**Apelul functiei care adauga un nou profesor in baza de date:**

BEGIN

PACHET\_SCOALA.ANGAJEAZA\_PROF(7000, 'Alex Stefanescu', 'Chitara', '1234567890', 2500);

DBMS\_OUTPUT.PUT\_LINE('PROFESORUL A FOST ANGAJAT CU SUCCES.');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('EROARE: ' || SQLERRM);

END;

/

**Apelul functiei ce modifica clasele la care preda un profesor:**

BEGIN

PACHET\_SCOALA.MODIFICA\_CLS(123, '5');

DBMS\_OUTPUT.PUT\_LINE('CLASELE PROFESORULUI AU FOST ACTUALIZATE CU SUCCES.');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('EROARE: ' || SQLERRM);

END;

/

**Apelul functiei ce realizeaza concedierea unui profesor:**

BEGIN

PACHET\_SCOALA.CONCEDIERE\_PROF(456);

DBMS\_OUTPUT.PUT\_LINE('PROFESORUL A FOST CONCEDIAT CU SUCCES.');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('EROARE: ' || SQLERRM);

END;

/

**-- Contruiti un pachet care sa contina:**

**-- - o procedura care returneaza numele si venitul total pentru profesorul al carui id este dat ca parametru**

**-- - o procedura care scade cu 20% salariile profesorului al carui id este dat**

**--In ambele prodeuri de mai sus sa se verifice situatia in care angajatul indicat nu exista**

**-- (invocand o exceptie din cadrul pachetului) prin apelul unei functii private, creata in acest scop**

**--Sa se apeleze procedurile din cadrul pachetului**

CREATE OR REPLACE PACKAGE SCOALA1 AS

FUNCTION EXISTA(ID\_P PROFESORI.ID\_PROFESOR%TYPE) RETURN NUMBER;

PROCEDURE PROC\_VENIT(ID\_P PROFESORI.ID\_PROFESOR%TYPE);

PROCEDURE PROC\_SAL(ID\_P PROFESORI.ID\_PROFESOR%TYPE);

END SCOALA1;

/

CREATE OR REPLACE PACKAGE BODY SCOALA1 AS

FUNCTION EXISTA(ID\_P PROFESORI.ID\_PROFESOR%TYPE) RETURN NUMBER AS

v\_check NUMBER := 0;

BEGIN

SELECT COUNT(1) INTO v\_check FROM PROFESORI WHERE ID\_PROFESOR = ID\_P;

RETURN v\_check;

END EXISTA;

PROCEDURE PROC\_VENIT(ID\_P PROFESORI.ID\_PROFESOR%TYPE) AS

TYPE TIPRECORD IS RECORD (

NUME PROFESORI.NUME%TYPE,

VENIT\_TOTAL NUMBER(10, 2)

);

REC TIPRECORD;

NO\_DATA EXCEPTION;

BEGIN

IF SCOALA1.EXISTA(ID\_P) = 0 THEN

RAISE NO\_DATA;

END IF;

SELECT NUME, NVL(SALARIUL + SALARIUL \* COMISION, SALARIUL)

INTO REC

FROM PROFESORI

WHERE ID\_PROFESOR = ID\_P;

DBMS\_OUTPUT.PUT\_LINE(REC.NUME || ' ' || REC.VENIT\_TOTAL);

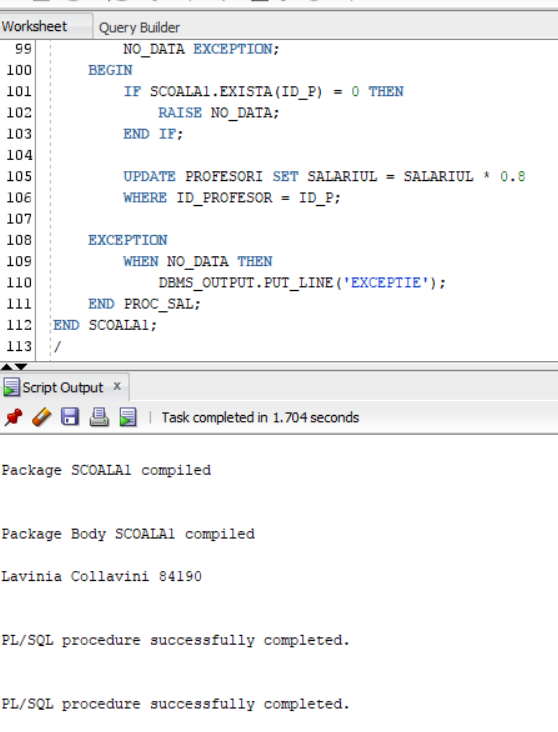
EXCEPTION

WHEN NO\_DATA THEN

DBMS\_OUTPUT.PUT\_LINE('EXCEPTIE');

END PROC\_VENIT;

PROCEDURE PROC\_SAL(ID\_P PROFESORI.ID\_PROFESOR%TYPE) AS

 NO\_DATA EXCEPTION;

BEGIN

IF SCOALA1.EXISTA(ID\_P) = 0 THEN

RAISE NO\_DATA;

END IF;

UPDATE PROFESORI SET SALARIUL = SALARIUL \* 0.8

WHERE ID\_PROFESOR = ID\_P;

EXCEPTION

WHEN NO\_DATA THEN

DBMS\_OUTPUT.PUT\_LINE('EXCEPTIE');

END PROC\_SAL;

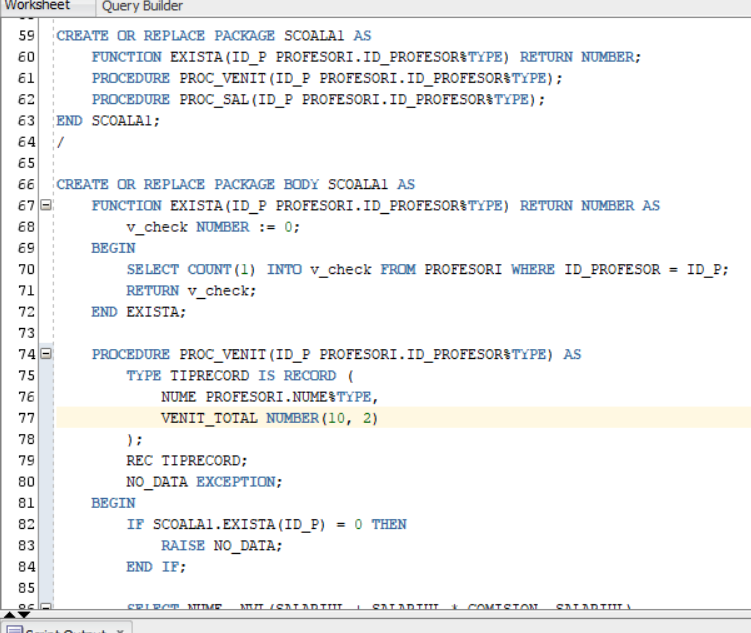
END SCOALA1;

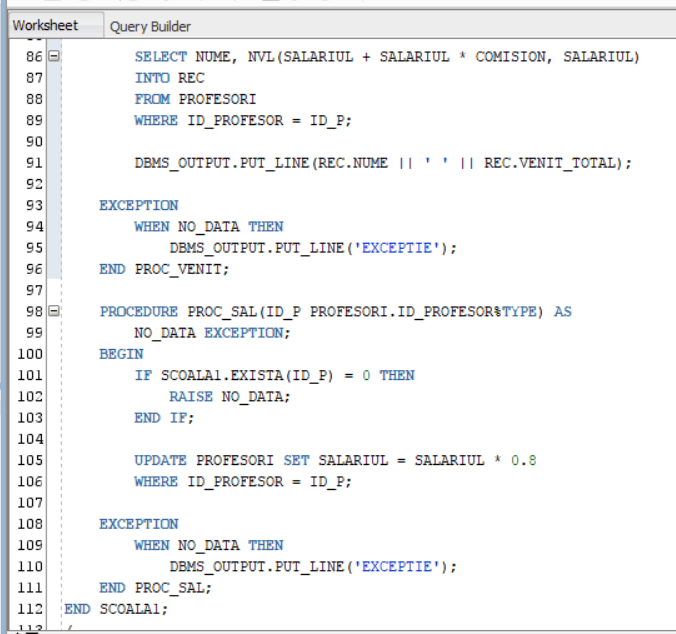
/

set serveroutput on

EXECUTE SCOALA1.PROC\_VENIT(6000);

EXECUTE SCOALA1.PROC\_SAL(6000);





**Triggeri**

**--1.Sa se realizeze un trigger care sa nu permita introducerea de valori negative ale bursei elevilor.**

SET SERVEROUTPUT ON

CREATE OR REPLACE TRIGGER BURSA\_NEGATIVA

BEFORE

INSERT OR UPDATE ON ELEVI

FOR EACH ROW

BEGIN

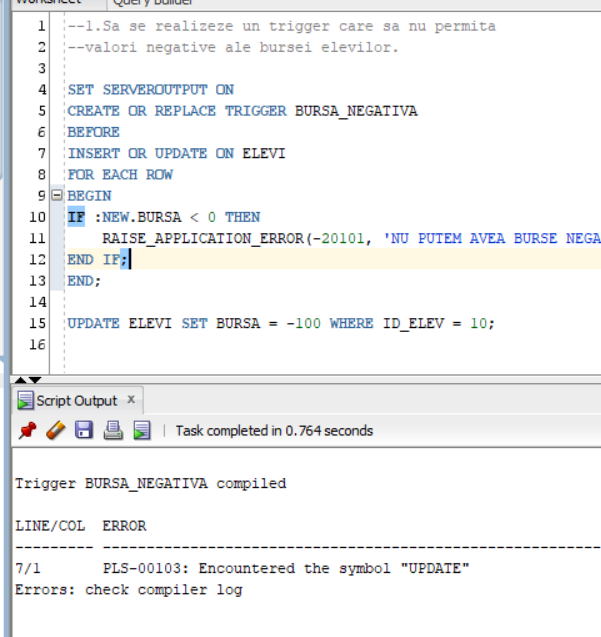
IF :NEW.BURSA < 0 THEN

RAISE\_APPLICATION\_ERROR(-20101, 'NU PUTEM AVEA BURSE NEGATIVE');

END IF;

END;

UPDATE ELEVI SET BURSA = -100 WHERE ID\_ELEV = 10;



**--Sa se creeze o tabela T\_LOGIN care sa fie populata la operatiile de INSERT, DELETE sau UPDATE pe tabela INSTRUMENT, cu tipul operatiei, utilizatorul care a executat-o, data curenta.**

CREATE OR REPLACE TRIGGER LOG\_INSTRUMENT

BEFORE INSERT OR UPDATE OR DELETE ON INSTRUMENT

FOR EACH ROW

BEGIN

IF INSERTING THEN

INSERT INTO T\_LOGIN VALUES ('I', USER, SYSDATE);

ELSIF UPDATING THEN

INSERT INTO T\_LOGIN VALUES ('U', USER, SYSDATE);

ELSIF DELETING THEN

INSERT INTO T\_LOGIN VALUES ('D', USER, SYSDATE);

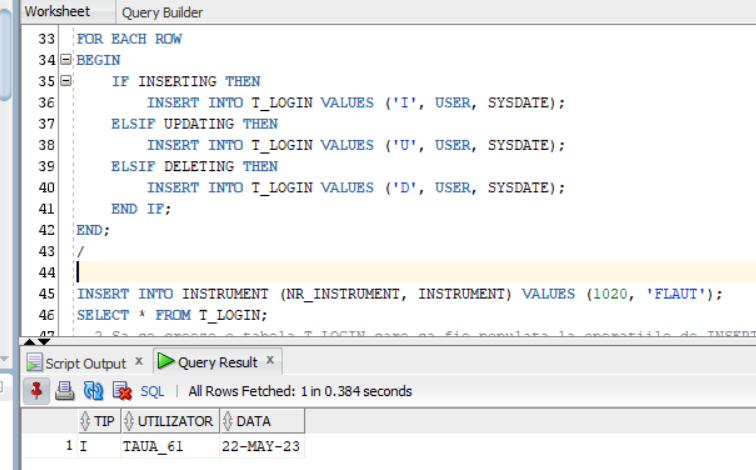
END IF;

END;

/

INSERT INTO INSTRUMENT (NR\_INSTRUMENT, INSTRUMENT) VALUES (1020, 'FLAUT');

SELECT \* FROM T\_LOGIN;



**--Sa se adauge in tabela ELEVI coloana BUGET, reprezentant bugetul maxim al scolii pentru fiecare elev.**

**--Sa se introduca valoarea 1500 in coloana nou adaugata**

**--Sa se creeze un trigger care sa nu permita acordarea unei burse mai mare decat este disponibil bugetul.**

**--De asemenea, pentru bursele curente, prin trigger se va reduce bugetul cu valoarea bursei curente. Sa se verifice declansatorul.**

ALTER TABLE ELEVI ADD BUGET NUMBER DEFAULT 1500;

CREATE OR REPLACE TRIGGER BUGET

BEFORE INSERT OR UPDATE OF BURSA

ON ELEVI

FOR EACH ROW

DECLARE

V\_BUGET NUMBER;

BEGIN

SELECT BUGET INTO V\_BUGET FROM ELEVI WHERE ID\_ELEV = :NEW.ID\_ELEV;

IF V\_BUGET <: NEW.BURSA THEN

RAISE\_APPLICATION\_ERROR(-21010, 'NU SE POATE ACORDA O BURSA ATAT DE MARE');

ELSIF INSERTING THEN

UPDATE ELEVI

SET BUGET= BUGET- :NEW.BURSA

WHERE ID\_ELEV = :NEW.ID\_ELEV;

ELSIF UPDATING THEN

UPDATE ELEVI

SET BUGET = BUGET + :OLD.BURSA - :NEW.BUGET

WHERE ID\_ELEV = :NEW.ID\_ELEV;

END IF;

END;

UPDATE ELEVI

SET BURSA=200

WHERE ID\_ELEV = 14;