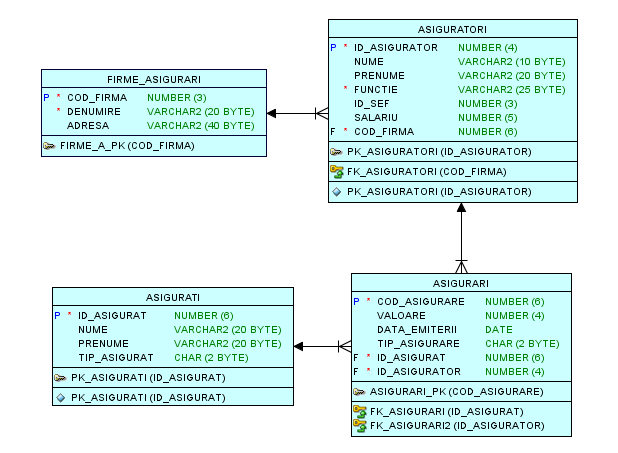
PROIECT SGBD

Proiectul prezinta baza de date a firmelor de asigurari din orasul Braila. Acesta contine patru tabele: FIRME\_ASIGURARI- unde sunt inregistrate informatii generale despre firmele din acest domeniu, ASIGURATORI-in care sunt memorati angajatii din cadrul fiecarei firme,acestia avand diverse functii (asistent, broker ,coordonator brokeri) iar rolul lor este de a emite polite de asigurare si/sau de a promova asigurarile potentialilor clienti, ASIGURATI-contine date despre clientii firmelor de asigurari si tabela ASIGURARI unde sunt stocate politele emise in cadrul fiecarei firmei, fiind de 3 tipuri: de autoturism, de sanatate, de bunuri.



1. **Sa se stearga toate asigurarile pentru un anumit id\_asigurator dat.Sa se afiseze numarul de inregistrari sterse.**

SET SERVEROUTPUT ON

BEGIN

DELETE FROM ASIGURARI WHERE ID\_ASIGURATOR=&id;

DBMS\_OUTPUT.PUT\_LINE('s-au sters ' || SQL%ROWCOUNT || 'asigurari');

END;

/

1. **Sa se determine valoarea platita de un asigurat cu un id dat pentru toate asigurarile sale.In cazul in care nu se gaseste un asigurat cu acel id, sa se trateze exceptia.**

DECLARE

V\_NUME ASIGURATI.NUME%TYPE;

V\_PRENUME ASIGURATI.PRENUME%TYPE;

V\_ID ASIGURARI.ID\_ASIGURAT%TYPE;

V\_VALOARE NUMBER;

BEGIN

SELECT A.NUME, A.PRENUME, A.ID\_ASIGURAT, SUM(B.VALOARE) INTO V\_NUME,V\_PRENUME,V\_ID,V\_VALOARE

FROM ASIGURATI A JOIN ASIGURARI B ON A.ID\_ASIGURAT=B.ID\_ASIGURAT WHERE A.ID\_ASIGURAT=&ID

GROUP BY A.NUME, A.PRENUME, A.ID\_ASIGURAT ORDER BY 1;

DBMS\_OUTPUT.PUT\_LINE ('ASIGURATORUL '|| V\_NUME || ' ' || V\_PRENUME ||'A PLATIT IN TOTAL PE ASIGURARI: '|| V\_VALOARE || ' RON');

EXCEPTION

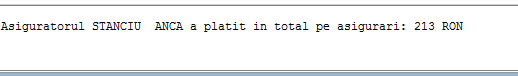
WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('NU EXSISTA ASIGURAT CU ACEL ID');

END;

/

Pentru id=456 a aparut urmatorul rezultat:



În funcție de salariul asiguratorilor avand id-ul citit de la tastatură, se va afişa un mesaj.

1. **Se afişează în ordine asiguratorii cu codurile în intervalul 100-140 atât timp cât salariul acestora este mai mic decât media:**

DECLARE

V\_SAL ASIGURATORI.SALARIU%TYPE;

V\_SALMEDIU V\_SAL%TYPE;

I NUMBER(4):=100;

BEGIN

SELECT AVG(SALARIU) INTO V\_SALMEDIU FROM ASIGURATORI;

DBMS\_OUTPUT.PUT\_LINE('SALARIUL MEDIU ESTE: '||V\_SALMEDIU);

WHILE I<=140 LOOP

SELECT SALARIU INTO V\_SAL FROM ASIGURATORI WHERE ID\_ASIGURATOR=I;

DBMS\_OUTPUT.PUT\_LINE('ASIGURATORUL CU CODUL '||I||' ARE SALARIUL: '||V\_SAL);

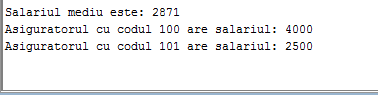
I:=I+1;

EXIT WHEN V\_SAL<V\_SALMEDIU;

END LOOP;

END;

/



1. **S a se afiseze utilizand un cursor explicit si o variabila de tip record( definita de utilizator) toti angajatii din firma cu codul 356.**

DECLARE

CURSOR C IS SELECT ID\_ASIGURATOR, NUME,PRENUME, SALARIU FROM ASIGURATORI WHERE COD\_FIRMA=356;

TYPE X IS RECORD(

ID ASIGURATORI.ID\_ASIGURATOR%TYPE,

N ASIGURATORI.NUME%TYPE,

P ASIGURATORI.PRENUME%TYPE,

S ASIGURATORI.SALARIU%TYPE

);

R X;

BEGIN

IF NOT C%ISOPEN THEN OPEN C;

END IF;

DBMS\_OUTPUT.PUT\_LINE('LISTA ASIGURATORILOR DE LA FIRMA CU CODUL 356');

LOOP

FETCH C INTO R;

EXIT WHEN C%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('ASIGURATORUL ' ||R.N||' ' ||R.P || ' CU ID-UL '|| R.ID|| ' SI SALARIUL: '||R.S );

END LOOP;

CLOSE C;

END;

/

1. **S a se afiseze suma salariilor aferenta fiecarei firme utilizand un cursor inline.**

DECLARE

BEGIN

DBMS\_OUTPUT.PUT\_LINE('TOTAL SALARII PE FIECARE FIRMA:');

FOR C IN

(SELECT F.COD\_FIRMA X, SUM(A.SALARIU) Y

FROM ASIGURATORI A, FIRME\_ASIGURARI F

WHERE A.COD\_FIRMA=F.COD\_FIRMA

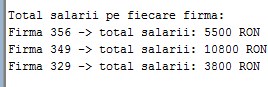
GROUP BY A.COD\_FIRMA)

LOOP

DBMS\_OUTPUT.PUT\_LINE('FIRMA '||C.X||' -> TOTAL SALARII: '|| C.Y||' RON');

END LOOP;

END;

****

1. **Utilizand un cursor explicit si o structura alternativa stabiliti unde lucreaza fiecare asigurator.**

DECLARE

A ASIGURATORI%ROWTYPE;

CURSOR C IS SELECT \* FROM ASIGURATORI;

BEGIN

FOR A IN C LOOP

IF A.COD\_FIRMA=356 THEN DBMS\_OUTPUT.PUT\_LINE('ASIGURATORUL ' || A.NUME ||' ' || A.PRENUME || ' LUCREAZA LA FIRMA ' || 'AXASIG BROKER.');

ELSIF A.COD\_FIRMA=349 THEN DBMS\_OUTPUT.PUT\_LINE('ASIGURATORUL ' || A.NUME ||' ' || A.PRENUME || ' LUCREAZA LA FIRMA ' || 'ROMASIG.');

ELSE DBMS\_OUTPUT.PUT\_LINE('ASIGURATORUL ' || A.NUME ||' ' || A.PRENUME || ' LUCREAZA LA FIRMA ' || 'AIR BROKER.');

END IF;

EXIT WHEN C%NOTFOUND;

END LOOP;

END;

/

1. **Utilizand un case si un cursor stabiliti comisionul pentru fiecare asigurare realizata.**

DECLARE

A ASIGURARI%ROWTYPE;

COMISION NUMBER;

CURSOR C IS SELECT \* INTO A FROM ASIGURARI;

BEGIN

FOR R IN C LOOP

CASE WHEN R.VALOARE<100 THEN COMISION:=R.VALOARE\*0.07;

WHEN R.VALOARE BETWEEN 100 AND 150 THEN COMISION:=R.VALOARE\*0.1;

ELSE COMISION:=R.VALOARE\*0.13;

END CASE;

DBMS\_OUTPUT.PUT\_LINE('PENTRU ASIGURAREA CU CODUL: ' || R.COD\_ASIGURARE || ' SI VALOAREA: '|| R.VALOARE || ' RON '|| 'SE PRIMESTE UN COMISION DE ' || COMISION || ' RON.' );

EXIT WHEN C%NOTFOUND;

END LOOP;

END;

/

1. **Mariti salariile cu 100 RON asiguratorilor care au salariul <2000 si apoi determinate numarul de salarii modificate.**

BEGIN

UPDATE ASIGURATORI SET SALARIU=SALARIU+100

WHERE SALARIU<2000;

IF SQL%FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('NUMAR DE SALARII MODIFICATE: ' || SQL%ROWCOUNT);

ELSE

DBMS\_OUTPUT.PUT\_LINE('NU S-A MODIFICAT NICIUN SALARIU');

END IF;

END;

/

1. **S a se afiseze asiguratorii din firma cu cei mai multi angajati.**

SET SERVEROUTPUT ON

DECLARE

CURSOR C(COD NUMBER)

IS SELECT NUME,PRENUME FROM ASIGURATORI

WHERE COD\_FIRMA=COD;

CURSOR D IS

SELECT COD\_FIRMA FROM ASIGURATORI

GROUP BY COD\_FIRMA ORDER BY COUNT(\*) DESC;

N ASIGURATORI.COD\_FIRMA%TYPE;

BEGIN

OPEN D;

FETCH D INTO N;

CLOSE D;

DBMS\_OUTPUT.PUT\_LINE(N);

FOR R IN C(N) LOOP

DBMS\_OUTPUT.PUT\_LINE(R.NUME ||' ' || R.PRENUME);

END LOOP;

END;

/

1. **Sa se afiseze denumirea firmei si apoi numele si prenumele fiecarui angajat din cadrul firmei.**

DECLARE

CURSOR C IS SELECT DENUMIRE,COD\_FIRMA FROM FIRME\_ASIGURARI;

CURSOR D(COD NUMBER) IS SELECT NUME,PRENUME FROM ASIGURATORI WHERE COD\_FIRMA=COD;

BEGIN

FOR F IN C LOOP

DBMS\_OUTPUT.PUT\_LINE(F.DENUMIRE);

FOR A IN D(F.COD\_FIRMA) LOOP

DBMS\_OUTPUT.PUT\_LINE(' '|| A.NUME || ' ' || A.PRENUME);

END LOOP;

END LOOP;

END;

/

1. **Utilizand un cursor si for update, mariti cu 100 RON salariul asiguratorilor cu salariul <2500 RON.**

DECLARE

CURSOR C IS SELECT ID\_ASIGURATOR,NUME,PRENUME,SALARIU FROM ASIGURATORI WHERE SALARIU<2500 FOR UPDATE;

BEGIN

FOR R IN C LOOP

UPDATE ASIGURATORI SET SALARIU=SALARIU+100 WHERE CURRENT OF C;

DBMS\_OUTPUT.PUT\_LINE(R.ID\_ASIGURATOR || ' ' || R.NUME || ' '|| R.PRENUME || ' ' || R.SALARIU);

END LOOP;

END;

1. **Tratati excepita ce apare atunci cand se incearca sa se stearga toate inregistrarile dintr-o tabela.**

DECLARE

DELETE\_EXCEPTION EXCEPTION;

PRAGMA EXCEPTION\_INIT(DELETE\_EXCEPTION, -2292);

BEGIN

DELETE FROM ASIGURATORI;

EXCEPTION

WHEN DELETE\_EXCEPTION THEN

DBMS\_OUTPUT.PUT\_LINE('NU PUTETI STERGE!');

END;

/

1. **Tratati printr-un exemplu exceptia care apare atunci cand se incearca sa se deschisa un cursor deja deschis.**

DECLARE

CURSOR C IS SELECT DENUMIRE,COD\_FIRMA FROM FIRME\_ASIGURARI;

CURSOR D(COD NUMBER) IS SELECT NUME,PRENUME FROM ASIGURATORI WHERE COD\_FIRMA=COD;

BEGIN

open c;

FOR F IN C LOOP

DBMS\_OUTPUT.PUT\_LINE(F.DENUMIRE);

FOR A IN D(F.COD\_FIRMA) LOOP

DBMS\_OUTPUT.PUT\_LINE(' '|| A.NUME || ' ' || A.PRENUME);

END LOOP;

END LOOP;

exception

when cursor\_already\_open then

dbms\_output.put\_line('Cursorul este deja deschis!');

END;

/

1. **Definiti si tratati o exceptie printr-un mesaj afisat ,care sa apara atunci cand nu se gasesc datele cerute.**

DECLARE NU\_EXISTA\_ID EXCEPTION;

PRAGMA EXCEPTION\_INIT(NU\_EXISTA\_ID,-20100);

BEGIN

UPDATE ASIGURATORI SET SALARIU=SALARIU+100

WHERE ID\_ASIGURATOR=1200;

IF SQL%NOTFOUND THEN RAISE\_APPLICATION\_ERROR(-20100, 'Nu exista asigurator cu acel id');

END IF;

EXCEPTION

WHEN NU\_EXISTA\_ID THEN

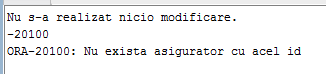
DBMS\_OUTPUT.PUT\_LINE('Nu s-a realizat nicio modificare.');

DBMS\_OUTPUT.PUT\_LINE(SQLCODE);

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

/



1. Modificat tipul asigurarii undei asigurari cu un cod dat de la tastatura. In cazul in care cod-ul e invalid definiti o exceptie.

DECLARE

ASIG\_INVALIDA EXCEPTION;

BEGIN

UPDATE ASIGURARI

SET TIP\_ASIGURARE='A'

WHERE COD\_ASIGURARE=&x;

IF SQL%NOTFOUND THEN

RAISE ASIG\_INVALIDA;

END IF;

EXCEPTION

WHEN ASIG\_INVALIDA THEN

DBMS\_OUTPUT.PUT\_LINE('NU EXISTA ASIGURARE CU ACEST COD');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A APARUT O EROARE! NU SE POATE ACTUALIZA TIPUL ASIGURARII!');

END;

/

1. **Creati o functie care sa calculeze valoarea totala a asigurarilor realizate de un asigurator.**

CREATE OR REPLACE FUNCTION

VAL\_TOTALA(ID\_A NUMBER)

RETURN NUMBER

IS

TOTAL NUMBER(10,2);

BEGIN

SELECT SUM(A.VALOARE) INTO TOTAL FROM ASIGURARI A

WHERE A.ID\_ASIGURATOR=ID\_A;

RETURN TOTAL;

END;

Test: select val\_totala(101) total, nume,prenume from asiguratori where id\_asigurator=101;

1. **Creati o procedura care afla media salariilor asiguratorilor.**

CREATE OR REPLACE PROCEDURE SAL\_MED

(A\_SAL\_MED OUT NUMBER)

IS

BEGIN

SELECT AVG(SALARIU) INTO A\_SAL\_MED FROM ASIGURATORI;

END;

/

1. **Creati o functie care primeste ca parametrii id-ul unui asigurator si salariu mediu .Cu acest salariu mediu va fi comparat salariul asiguratorului. Daca salariul asiguratorului este mai mare decat cel introdus se returneaza TRUE, altfel FALSE.**

CREATE OR REPLACE FUNCTION VERIFICA\_SALARIUL (A\_ID ASIGURATORI.ID\_ASIGURATOR%TYPE, SAL NUMBER)

RETURN BOOLEAN

IS

V\_SALARIU ASIGURATORI.SALARIU%TYPE;

BEGIN

SELECT SALARIU INTO V\_SALARIU FROM ASIGURATORI WHERE ID\_ASIGURATOR=A\_ID;

IF V\_SALARIU > SAL THEN

RETURN TRUE;

ELSE

RETURN FALSE;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN NULL;

END;

/

Apel:

DECLARE

SAL\_M NUMBER;

BEGIN

SAL\_MEDIU (SAL\_M);

IF (VERIFICA\_SALARIUL(101, SAL\_M) IS NULL) THEN

DBMS\_OUTPUT.PUT\_LINE('ASIGURATOR INEXISTENT !');

ELSIF (VERIFICA\_SALARIUL(101, SAL\_M)) THEN

DBMS\_OUTPUT.PUT\_LINE('ASIGURATORUL ARE SALARIUL MAI MARE DECAT SALARIUL MEDIU!');

ELSE

DBMS\_OUTPUT.PUT\_LINE(' ASIGURATORUL ARE SALARIUL MAI MIC DECAT SALARIUL MEDIU!);

END IF;

END;

/

1. **Sa se afiseze printr-o procedura informatiile despre o asigurare utilizand ca data de intrare codul asigurarii ,si ca date de iesire valoarea si tipul.**

CREATE OR REPLACE PROCEDURE CAUTA\_ASIGURARE

(V\_COD ASIGURARI.COD\_ASIGURARE%TYPE,

V\_VALOARE OUT ASIGURARI.VALOARE%TYPE,

V\_TIP OUT ASIGURARI.TIP\_ASIGURARE%TYPE)

IS

BEGIN

SELECT VALOARE,TIP\_ASIGURARE INTO V\_VALOARE, V\_TIP FROM ASIGURARI WHERE COD\_ASIGURARE=V\_COD;

DBMS\_OUTPUT.PUT\_LINE('ASIGURAREA CU CODUL ' || V\_COD || ' ARE VALOAREA ' || V\_VALOARE||' SI E PENTRU ');

IF V\_TIP='A' THEN DBMS\_OUTPUT.PUT\_LINE(' AUTOTURISM');

ELSIF V\_TIP='B' THEN DBMS\_OUTPUT.PUT\_LINE(' BUNURI');

ELSE DBMS\_OUTPUT.PUT\_LINE(' SANATATE ');

END IF;

END;

/

Apel:

DECLARE

VAL ASIGURARI.VALOARE%TYPE;

TIP ASIGURARI.TIP\_ASIGURARE%TYPE;

BEGIN

CAUTA\_ASIGURARE(1135, VAL, TIP);

END;

/

1. **Sa se creeze o functie prin care sa returneze impozitul unui salariu.**

CREATE OR REPLACE FUNCTION

GET\_IMPOZIT(SAL ASIGURATORI.SALARIU%TYPE)

RETURN NUMBER

IS

IMPOZIT NUMBER;

BEGIN

IF SAL<2000 THEN IMPOZIT:=SAL \*0.16;

ELSIF SAL BETWEEN 2000 AND 3000 THEN IMPOZIT:=SAL \*0.2;

ELSE IMPOZIT:=SAL \*0.3;

END IF;

RETURN IMPOZIT;

END;

DECLARE

N NUMBER;

BEGIN

N:=GET\_IMPOZIT(2001);

DBMS\_OUTPUT.PUT\_LINE(N);

END;

1. **Creati un pachet utilizand o functie si o procedura definite mai sus.**

CREATE OR REPLACE PACKAGE P1 IS

PROCEDURE SAL\_MED(A\_SAL\_MED OUT NUMBER);

FUNCTION VERIFICA\_SALARIUL (A\_ID ASIGURATORI.ID\_ASIGURATOR%TYPE, SAL NUMBER);

RETURN BOOLEAN

END; /

CREATE OR REPLACE PACKAGE BODY P1

IS

PROCEDURE SAL\_MED

(A\_SAL\_MED OUT NUMBER)

IS

BEGIN

SELECT AVG(SALARIU) INTO A\_SAL\_MED FROM ASIGURATORI;

END;

FUNCTION VERIFICA\_SALARIUL (A\_ID ASIGURATORI.ID\_ASIGURATOR%TYPE, SAL NUMBER)

RETURN BOOLEAN

IS

V\_SALARIU ASIGURATORI.SALARIU%TYPE;

BEGIN

SELECT SALARIU INTO V\_SALARIU FROM ASIGURATORI WHERE ID\_ASIGURATOR=A\_ID;

IF V\_SALARIU > SAL THEN

RETURN TRUE;

ELSE

RETURN FALSE;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN NULL;

END;

END;

/

1. **Creati un trigger care se va declansa atunci cand incearca sa se stearga un rand din tabela asigurari.**

CREATE OR REPLACE TRIGGER VERIFICA\_ID BEFORE DELETE ON ASIGURARI

BEGIN

RAISE\_APPLICATION\_ERROR(-20000,'NU SE POT STERGE ASIGURARILE, ODATA CE AU FOST EMISE');

END;

/

Test: DELETE FROM ASIGURARI WHERE COD\_ASIGURARE=1785;

1. **Creati un trigger care sa nu permita inserari sau update-uri cand ora este 00:00.**

CREATE OR REPLACE TRIGGER TRG\_ORA AFTER INSERT OR UPDATE OR DELETE ON

ASIGURARI

BEGIN

IF UPDATING AND TO\_CHAR(SYSDATE,'HH24:MI') = '00:00' THEN RAISE\_APPLICATION\_ERROR(-20900,'NU SE PERMITE UPDATAREA LA ACEASTA ORA');

ELSIF INSERTING AND TO\_CHAR(SYSDATE,'HH24:MI')= '00:00' THEN RAISE\_APPLICATION\_ERROR(-20900,'NU SE PERMITE INSERAREA LA ACEASTA ORA');

END IF;

END;

/

1. **Creati un trigger care sa nu permita scaderea salariului.**

CREATE OR REPLACE TRIGGER VERIF\_SALARIU

AFTER UPDATE OF SALARIU ON ASIGURATORI

FOR EACH ROW

DECLARE

BEGIN

IF :NEW.SALARIU<:OLD.SALARIU THEN RAISE\_APPLICATION\_ERROR(-20000,'SALARIUL INTRODUS E MAI MIC CA CEL VECHI');

END IF;

END;

/

Test: UPDATE ASIGURATORI SET SALARIU=20 WHERE ID\_ASIGURATOR=101;

1. **Creati un trigger care sa se declanseze atunci cand se introduce un angajat cu salariul in afara intervalului [100,10000].**

CREATE OR REPLACE TRIGGER VERIF\_SALARIU2

AFTER INSERT ON ASIGURATORI

FOR EACH ROW

DECLARE

BEGIN

IF :NEW.SALARIU NOT BETWEEN 100 AND 10000 THEN RAISE\_APPLICATION\_ERROR(-20010,'SALARIUL INTRODUS ESTE IN AFARA INTERVALULUI');

END IF;

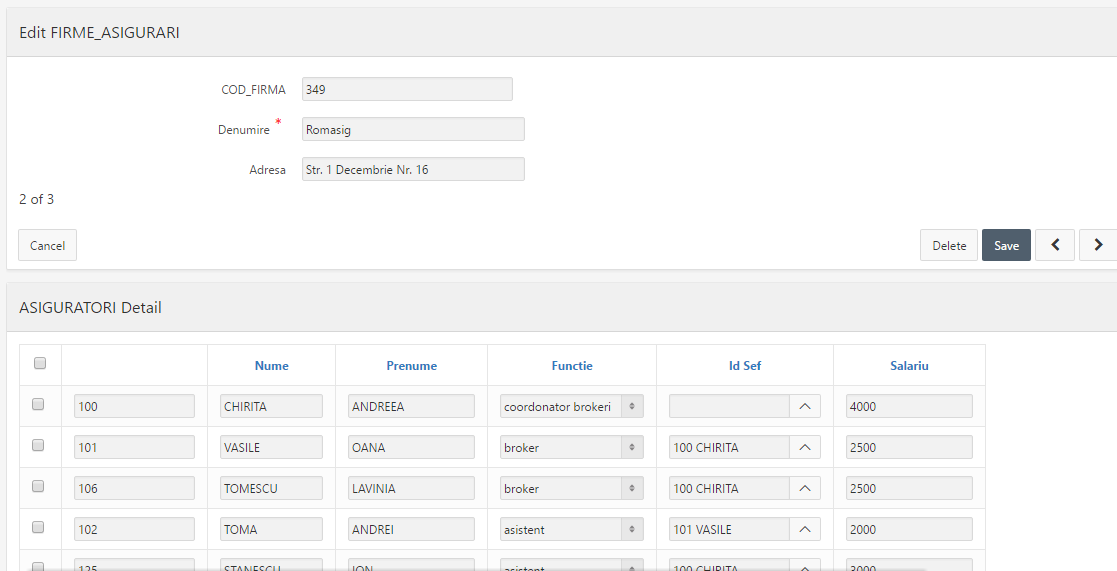
END;

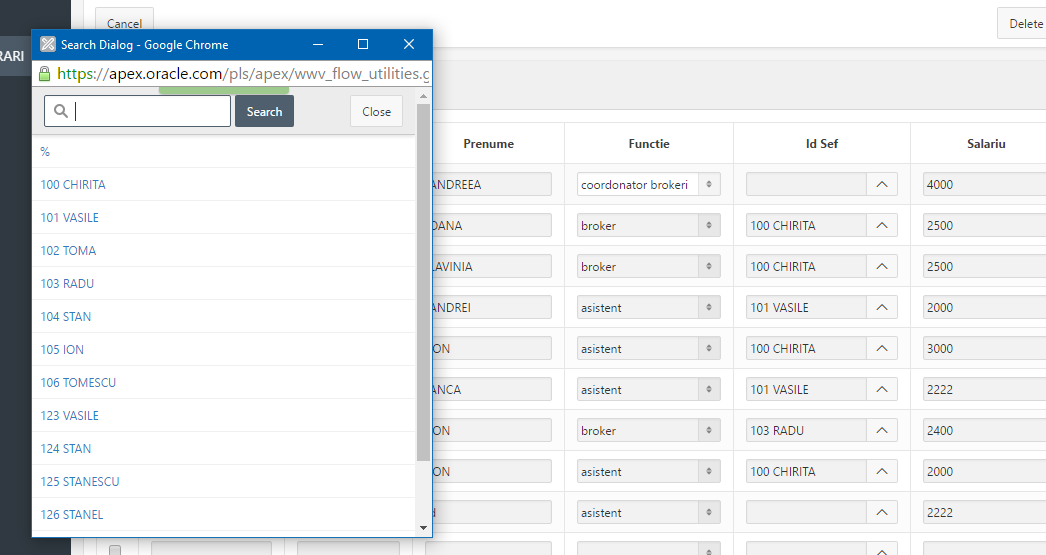
/

Test: INSERT INTO ASIGURATORI VALUES (243,'VASILE','ION','broker',100,10,349);

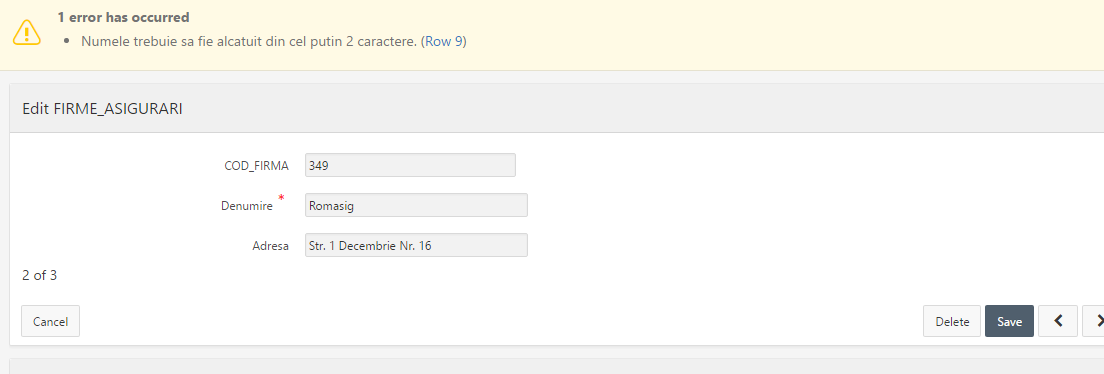
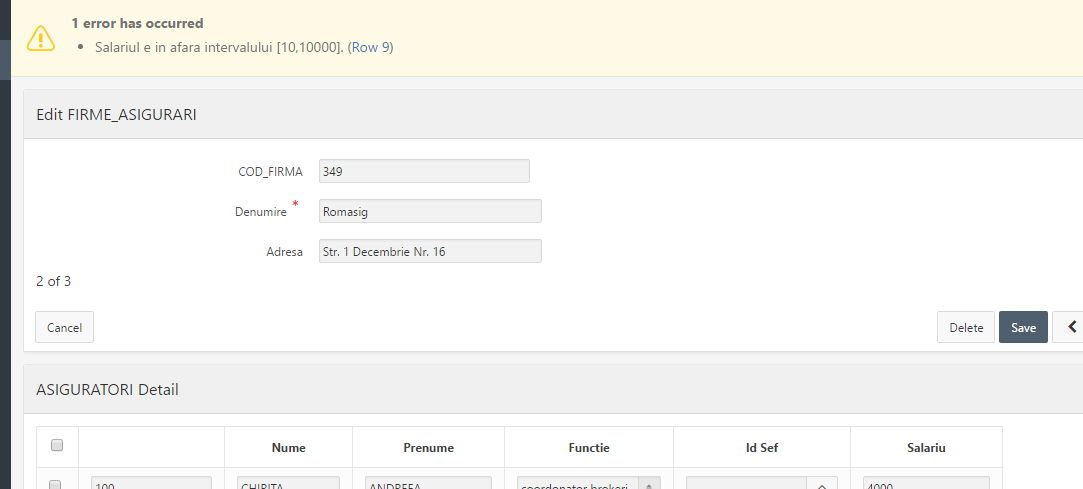
**Form-> Master Detail**

**LOV pe atributul id\_sef:**

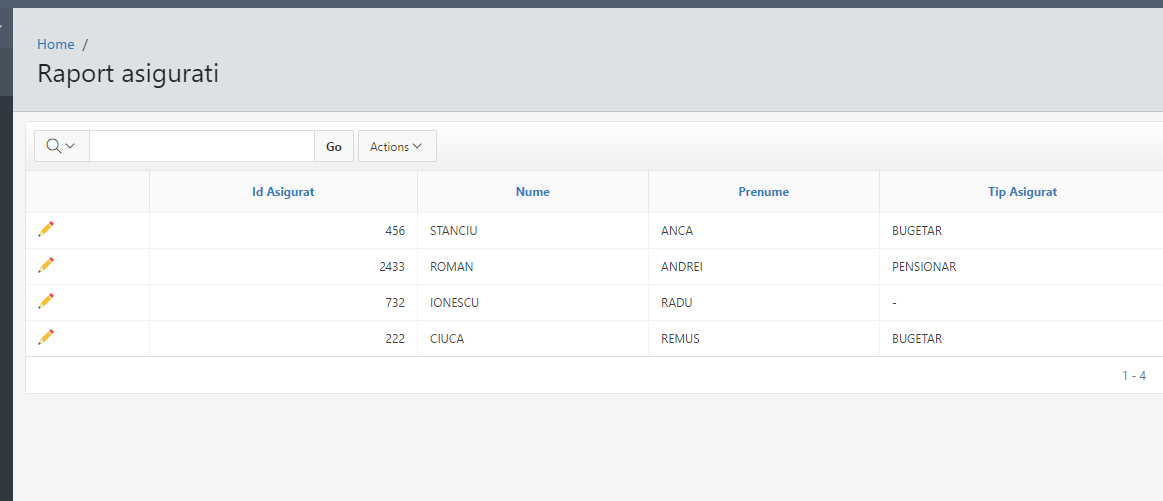
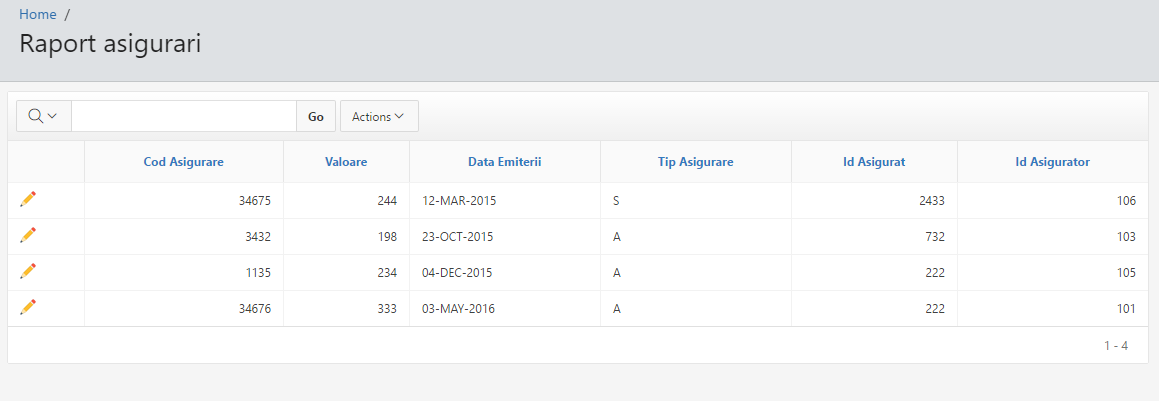




**Validari:**



**Report-Interactive Report**



Link: https://apex.oracle.com/pls/apex/f?p=72614:1:1174501327025::NO:::