Per a cadascun d’ells, tracteu les excepcions que considereu necessàries.

1. Realitzeu un procediment anomenat MostrarAbreviatures que mostre les tres primeres lletres del cognom de cada empleat.

call MostrarAbreviatures()

CREATE OR REPLACE PROCEDURE MostrarAbreviatures()

language plpgsql

AS

$$

DECLARE

cur1 cursor FOR SELECT empname,1,3, empno FROM emp;

inf record;

BEGIN

open cur1;

loop

fetch cur1 INTO inf;

exit when not found;

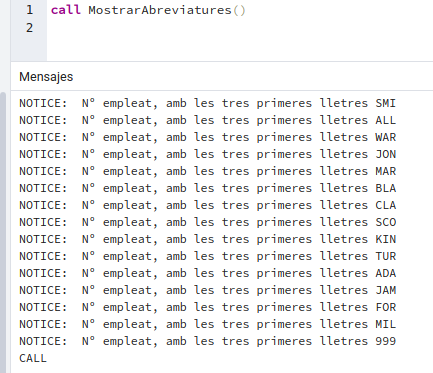
raise notice 'Nº empleat, amb les tres primeres lletres %', substring(inf.empname,1,3);

end loop;

close cur1;

end;

$$



2. Realitzeu un procediment anomenat MostrarCostosDepartament que mostre els noms de tots els departaments i el seu pressupost basat en els salaris.

Call MostrarCostosDepartament()

CREATE OR REPLACE PROCEDURE MostrarCostosDepartament()

language plpgsql

AS

$$

DECLARE

cur1 cursor for SELECT deptname, SUM(sal) AS salario FROM dept INNER JOIN emp ON emp.deptno = dept.deptno GROUP BY deptname;

BEGIN

open cur1;

loop

fetch cur1 into inf;

exit when not found;

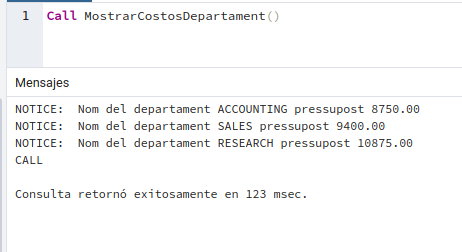
raise notice 'Nom del departament % pressupost %', inf.deptname,inf.salario;

end loop;

close cur1;

end;

$$



3. Realitzeu un procediment anomenat VeteransxDepartament que mostre el nom de cada departament i l’empleat més antic, junt amb la data de contractació.

Call VeteransxDepartament()

CREATE OR REPLACE PROCEDURE VeteransxDepartament()

language plpgsql

AS

$$

DECLARE

cur1 cursor for SELECT dept.deptname, emp.job, hiredate FROM dept INNER JOIN emp ON emp.deptno = dept.deptno group by deptname, job, hiredate LIMIT 3;

inf record;

BEGIN

open cur1;

loop

fetch cur1 into inf;

exit when not found;

raise notice 'Empleat mes antic % %', inf.deptname, MIN(inf.hiredate);

end loop;

close cur1;

end;

$$

Amb subconsulta:

CREATE OR REPLACE PROCEDURE VeteransxDepartament()

language plpgsql

AS

$$

DECLARE

cur1 cursor for SELECT dept.deptname, emp.job, hiredate FROM dept INNER JOIN emp ON emp.deptno = dept.deptno group by deptname, job, hiredate WHERE dept.deptname, emp.job, hiredate in (SELECT deptno, MIN(hiredate)) FROM emp LIMIT 3;

inf record;

BEGIN

open cur1;

loop

fetch cur1 into inf;

exit when not found;

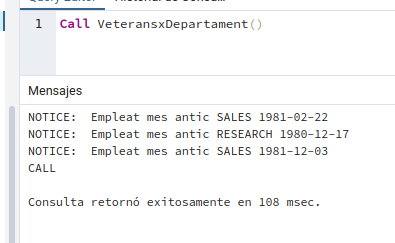
raise notice 'Empleat mes antic % %', inf.deptname, MIN(inf.hiredate);

end loop;

close cur1;

end;

$$



4. Realitzeu un procediment que donada una cadena de caràcters, mostre el nom dels departaments que coincideixen en eixa cadena i també lliste les dades dels empleats que tenen empleats al seu càrrec.

CREATE OR REPLACE PROCEDURE ex4(nomdept varchar(15))

language plpgsql

AS

$$

DECLARE

cur1 cursor FOR SELECT \* FROM emp INNER JOIN dept ON emp.deptno = dept.deptno WHERE upper(nomdept) = deptname AND deptno IN (SELECT emp.mgr, emp.empno WHERE mgr);

inf record;

BEGIN

open cur1;

loop

fetch cur1 INTO inf;

exit when not found;

raise notice 'Departaments que coincideixen % i empleats %', inf.deptname, inf.empname;

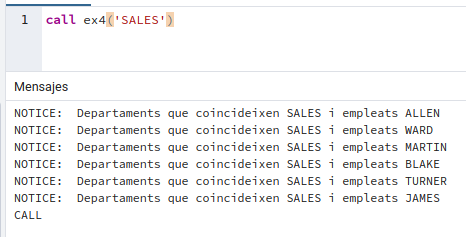
end loop;

close cur1;

end;

$$

call ex4()



Triggers

1. Codifiqueu un trigger anomenat comissio\_nomes\_venedors que només permeta tindre comissions als venedors (SALESMAN).

CREATE OR REPLACE FUNCTION comissi\_venedors()

returns trigger

language plpgsql

AS

$$

BEGIN

SELECT comm FROM emp INNER JOIN dept ON emp.deptno = dept.deptno WHERE deptname = 'SALESMAN';

if (job != 'SALESMAN')

then

raise exception 'Nomes poden cobrar comissions a SALESMAN';

end if;

end;

$$

CREATE TRIGGER comisions

AFTER INSERT or UPDATE

ON emp

FOR EACH ROW

EXECUTE PROCEDURE comissi\_venedors()

update emp

set comm = 300

where empname = 'ALLEN';

2. Creeu un trigger log\_empleats que registre totes les operacions sobre la taula EMP en una taula anomenada AUDIT\_EMP on es guarde l’usuari, la data i hora i el tipus d'operació efectuada sobr la taula (INSERT, UPDATE o DELETE). Pista: TG\_OP Taula AUDIT\_EMP: id (sencer autoincremental) usuari (cadena de caràcters) fecha (data) tipus\_op (cadena de caràcters)

CREATE TABLE audit\_emp(

id serial,

usuari varchar(15),

fecha date,

tipus\_op varchar(15));

CREATE OR REPLACE FUNCTION trigger2()

returns trigger

language plpgsql

as

$$

BEGIN

if(TG\_OP = ‘DELETE’) then

INSERT INTO audit\_emp (id, usuari, fecha, tipus\_op)

VALUES(old.empno , NEW.usuari, NEW.fecha, NEW.tipus\_op);

return old;

elseif(TG\_OP = ‘UPDATE’) then

INSERT INTO audit\_emp (id, usuari, fecha, tipus\_op)

VALUES(old.empno, NEW.usuari, NEW.fecha, NEW.tipus\_op);

RETURN NEW;

elseif(TG\_OP = ‘INSERT’) THEN

INSERT INTO audit\_emp (id, usuari, fecha, tipus\_op)

VALUES(old.empno, NEW.usuari, NEW.fecha, NEW.tipus\_op);

RETURN NEW;

END;

$$

CREATE TRIGGER log\_empleats

AFTER INSERT OR UPDATE OR DELETE

ON audit\_emp

FOR EACH ROW

EXECUTE PROCEDURE trigger2()

3. Implementeu un trigger que puge un 10% el sou als empleats si canvien de localitat on treballen (la localitat del seu departament).

CREATE OR REPLACE FUNCTION trigger3()

returns trigger

language plpgsql

as

$$

begin

SELECT \* from emp INNER JOIN dept ON emp.deptno = dept.deptno;

if old.loc != new.loc

then

UPDATE emp

set new.sal = old.sal \* 1.1;

end if;

end;

$$

CREATE TRIGGER augmenta\_sou

AFTER INSERT

ON emp

FOR EACH ROW

EXECUTE PROCEDURE trigger3();

4. La crisi ha arribat a l'empresa. Realitza un trigger per forçar que els empleats que s'incorporen siguen sempre els que menys guanyen del seu departament i a més, que no cobren comissions.

CREATE OR REPLACE FUNCTION nous\_empleats()

returns trigger

language plpgsql

as

$$

BEGIN

SELECT \* FROM emp;

if(hiredate = null) then

raise exception 'No pots canviar el sou y comissions de una persona sense contracte';

end if;

while(new.sal < old.sal)

loop

if new.hiredate != old.hiredate

then

INSERT INTO emp(empno, empname, job, mgr, hiredate, sal, c omm, deptno)

VALUES (new.empno, new.empname, new.job, new.mgr, new.hiredate, new.sal, 0, new.deptno);

end if;

end loop;

RETURN NEW;

END;

$$

CREATE TRIGGER trigger4

AFTER INSERT OR UPDATE OR DELETE

ON emp

FOR EACH ROW

EXECUTE PROCEDURE nous\_empleats()

5. Fes un trigger per garantir que només hi ha empleats amb ofici 'SALESMAN' al departament 'SALES'. L'usuari ha de saber amb claredat perquè ha fallat l'operació i en quin registre ha fallat, així com si el problema és l'absència d'un departament 'SALES' o que se li ha assignat un departament erroni al comercial.

CREATE OR REPLACE FUNCTION comprova()

returns trigger

language plpgsql

as

$$

BEGIN

SELECT \* FROM emp;

if(job != 'SALESMAN')

then

raise Exception 'Nomes hi poden haver treballadors SALESMAN dintre del departament SALES';

end if;

if(job = 'SALESMAN' AND deptno = 30)

then

INSERT INTO emp(empno, empname, job, mgr, hiredate, sal, comm, deptno)

VALUES(new.empno, new.empname, old.job, new.mgr, new.hiredate, new.sal, new.comm, old.deptno);

end if;

end;

$$

6. Fes un trigger per assegurar que el total de comissions d'un departament mai supera el total dels salaris d'aquest mateix departament.

CREATE OR REPLACE FUNCTION comissio\_salari()

returns trigger

language plpgsql

as

$$

begin

SELECT \* FROM emp GROUP BY deptno;

if SUM(comm) > SUM(sal)

then

raise Exception 'Les comisions no poden superar el total de salaris del mateix departament';

else

INSERT INTO emp(empno, empname, job, mgr, hiredate, sal, comm, deptno)

VALUES(old.empno, old.empname, old.job, old.mgr, old.hiredate, old.sal, old.comm, old.deptno);

end if;

end;

$$

CREATE TRIGGER trigger6

AFTER INSERT OR UPDATE

ON emp

FOR EACH ROW

EXECUTE PROCEDURE comissio\_salari()