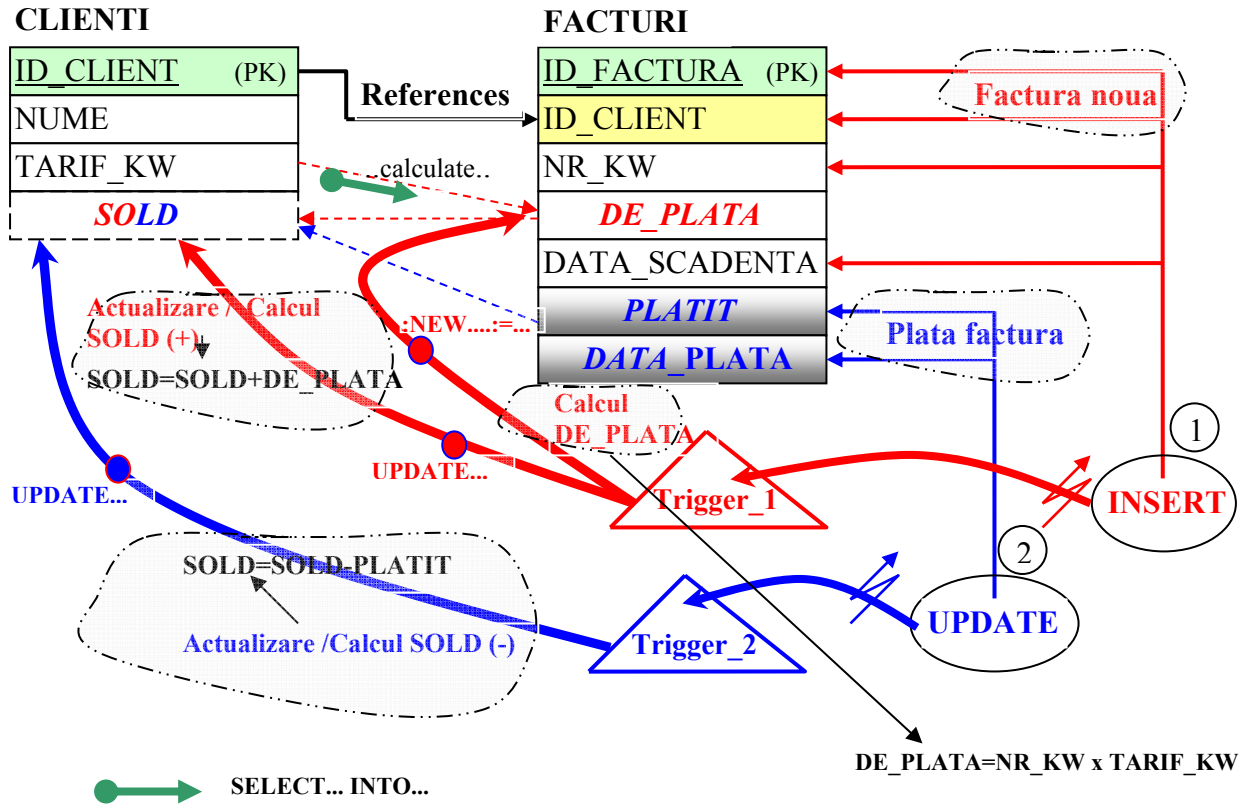
PROBLEMA EXEMPLU

1. Programare PL/SQL : 2 ore



A) Se considera o aplicatie de plata a facturilor de energie electrica, avand doua tabele:

- Tabela CLIENTI cu coloanele: ID_CLIENT (unic pentru fiecare client si nenul), NUME (numele clientului), TARIF_KW (tariful in LEI negociat cu fiecare client), respectiv SOLD (suma totala de plata, pentru toate facturile emise pentru acel client). Coloana SOLD este implicit zero la crearea tabelei. Ulterior, o valoare pozitiva in SOLD semnifica existenta unor facturi inca neplatite, iar o valoare negativa semnifica plata unor sume mai mari decat are de platit clientul (practic o plata "in avans").

- Tabele FACTURI cu coloanele: ID_FACTURA (unica pentru fiecare client si nenula), ID_CLIENT, NR_KW (consumul aferent facturii in cauza), DE_PLATA (valoarea de plata a facturii), DATA_SCADENTA (data pana la care se poate plati factura, fara a fi considerata plata intarziata), PLATIT (suma de platit, necompletata la emiterea facturii), DATA_PLATA (necompletata la emiterea facturii, respectiv data curenta in momentul efectuarii platii).

B) Emiterea unei noi facturi (evident posibila doar pentru clientii existenti in tabela CLIENTI), implica o noua linie in tabela FACTURI, cu completarea doar a coloanelor ID_FACTURA, ID_CLIENT, NR_KW, DATA_SCADENTA. Operatie necesita de asemenea urmatoarele operatiuni suplimentare:

- In momentul emiterii, pe baza valorii coloanei TARIF_KW (din tabela CLIENTI), aferenta clientului respectiv, se calculeaza valoarea corespunzatoare a coloanei DE_PLATA=NR_KW x TARIF_KW.

- In tabela CLIENTI, se va actualiza (recalcula) soldul total aferent clentului (fiind posibil ca acesta sa aiba mai multe facturi de platit): SOLD=SOLD+DE_PLATIT.

C) Plata unei facturi implica actualizarea coloanei PLATIT cu valoarea platita (si completarea coloanei DATA_PLATA cu data curenta), respectiv decrementarea SOLD-ului (din tabela CLIENTI) cu valoarea platita (SOLD=SOLD-PLATIT) . O factura poate fi platita o singura data (adica se poate face un singur UPDATE pe ea).

1.a) Sa se creeze cele doua tabele, impunand, unde este cazul, restrictiile (constrangerile) adecvate pentru fiecare coloana (chei primare, valori nenule, domenii de valori, relationare, valori implicite). Sa se insereze doi clienti in tabela clienti.

```
set autocommit on;

create table clienti
(       id_client integer primary key,
        nume varchar(30) not null,
        tarif_kw integer not null check(tarif_kw>0),
        sold integer default 0 not null);

create table facturi
(       id_factura integer primary key,
        id_client references clienti(id_client) on delete cascade,
        nr_kw integer not null check (nr_kw>0),
        de_plata integer,
        data_scadenta date not null,
        platit integer check (platit>0),
        data_plata date);

insert into clienti (id_client, nume, tarif_kw) values (1,'ana', 10);
insert into clienti (id_client, nume, tarif_kw) values (2,'vali', 20);
```

1.b) Sa se scrie codul unui trigger, declansat la emiterea unei noi facturi, care asigura executia operatiilor suplimentare necesare (calcul coloana DE_PLATA, actualizarea/recalcularea SOLD-ului, aferent clientului respectiv).

```
create or replace trigger tr1
before insert on facturi
for each row
declare
tarif integer;
begin
select tarif_kw into tarif from clienti where id_client =:NEW.id_client;
:NEW.de_plata:=tarif * :NEW.nr_kw;
update clienti set sold = sold + :NEW.de_plata where id_client=:NEW.id_client;
end;
/
```

Insert declansator:

```
insert into facturi (id_factura, id_client, nr_kw, data_scadenta)
values (1,1,150,'27-03.2017');
```

Verificare:

```
select * from clienti;
select * from facturi;
```

1.c) Sa se scrie codul unui trigger, declansat la plata unei facturi, care executa operatia de actualizare a colonei SOLD aferenta clientului in cauza (recalculare cat mai are de platit clientul).

```
create or replace trigger tr2
before update on facturi
for each row
declare
begin
update clienti set sold = sold - :NEW.platit where id_client=:NEW.id_client;
end;
/
```

Update declansator:

```
update facturi set platit=1000, data_plata =sysdate where id_factura=1;
```

Verificare:

```
select * from clienti;
select * from facturi;
```

1.d) Sa se scrie codul unui trigger care cumuleaza ambele functionalitati ale triggerelor anterioare intr-unul **singur**. Tratati exceptiile care pot sa apara, afisand mesajul ‘Eroare’.

```
create or replace trigger tr3
before insert or update on facturi
for each row
declare
tarif integer;
begin
if (:OLD.id_factura is null) then
        select tarif_kw into tarif from clienti where id_client =:NEW.id_client;
        :NEW.de_plata:=tarif * :NEW.nr_kw;
        update clienti set sold = sold + :NEW.de_plata where id_client=:NEW.id_client;
else
        update clienti set sold = sold - :NEW.platit where id_client=:NEW.id_client;
end if;

exception
when others then
raise_application_error(-20000, 'Eroare!');
end;
/
```

Comenzi declansatoare:

```
insert into facturi (id_factura, id_client, nr_kw, data_scadenta)
values(2,2,150,'12-06-2017');
update facturi set platit=2000, data_plata =sysdate where id_factura=2;
```

Verificare:

```
select * from clienti;
select * from facturi;
```

1.e) Sa se scrie o procedura stocata al carei apel permite emiterea unei noi facturi (cu toate operatiile necesare aferente - vezi punctul 1.b), evident substituind functionalitatea trigger-ului aferent (inainte de executia procedurii, se vor dezactiva trigger-ele anterioare: **alter trigger *nume_trigger* disable;**).

```
create or replace procedure p1(id_factura_p integer, id_client_p integer, nr_kw_p
integer, data_scadenta_p date)
as
tarif_p integer;
de_plata_p integer;
begin
select tarif_kw into tarif_p from clienti where id_client =id_client_p;
de_plata_p:=tarif_p * nr_kw_p;
insert into facturi (id_factura,id_client, nr_kw, de_plata, data_scadenta)
values(id_factura_p, id_client_p, nr_kw_p, de_plata_p, data_scadenta_p);
update clienti set sold=sold+de_plata_p where id_client=id_client_p;
end;
/
```

Rulare:

```
exec p1(1,1,10,'29-08-2017');
```

Verificare:

```
select * from clienti;
select * from facturi;
```

1.f) Sa se scrie o procedura stocata al carei apel permite plata unei facturi (cu toate operatiile necesare aferente - vezi punctul 1.c), evident substituind functionalitatea trigger-ului aferent (inainte de executia procedurii, se vor dezactiva trigger-ele anterioare : ***alter trigger nume_trigger disable;***).

```
create or replace procedure p2(id_factura_p integer,  platit_p integer)
as
id_client_p integer;
begin
select  id_client into id_client_p from facturi where id_factura=id_factura_p;
update facturi set platit=platit_p, data_plata=sysdate where id_factura=id_factura_p;
update clienti set sold=sold-platit_p where id_client=id_client_p;
end;
/
```

Rulare:

```
exec p2(1,1000);
```

Verificare:

```
select * from clienti;
select * from facturi;
```

2. Programare SQL: 2 ore

2. Presupunand ca, coloana SOLD din tabela CLIENTI nu exista, sa se scrie comenzile care permit interogarile urmatoare:

2.a) Comanda SELECT utilizata pentru afisarea clientilor (ID_CLIENT, NUME) cu toate facturile emise platite, ordonati alfabetic (se va lua in considerare posibilitatea ca suma platita in contul unei facturi sa fie diferita de suma de plata, fiind posibile si plati in avans).

Obs: Ce este pus intre paranteze drepte este optional!

```
select a. id_client, nume, [sum(de_plata), sum(platit)]
from clienti a, facturi.b
where   a,id_client=b.id_client
group by a.id_client, nume
having sum(de_plata)>=sum(platit)
order by nume;
```

Sau:

```
select id_client,nume
from clienti
where id_client in (select id_client
                    from facturi
                    group by id_client
                    having sum(de_plata)=sum(platit))
order by nume;
```

2.b) Comanda SELECT care permite afisarea clientilor (ID_CLIENT, NUME) pentru care nu s-a emis inca nici o factura, ordonati alfabetic.

```
select id_client, nume
from clienti
where id_client not in (select id_client from facturi)
order by nume;
```

Sau:

```
select a.id_client, nume
from clienti a, facturi b
where a.id_client=b.id_client(+) and b.id_client is null;
```

2.c) Comanda SELECT utilizata pentru afisarea clientilor (ID_CLIENT, NUME) care au platit facturi cu intarziere (deci cu plati efectuate dupa DATA_SCADENTA).

```
select a, id_client, nume
from from clienti a, facturi b
where a.id_client=b.id_client and data_plata>data_scadenta;
```

Sau:

```
select id_client, nume
from clienti
where id_client in (select id_client from facturi where data_plata>data_scadenta);
```

2.d) Comanda SELECT utilizata pentru afisarea clientilor (ID_CLIENT, NUME) cu facturi neplatite, respectiv a soldului pe care il mai au de platit (totalul sumelor neplatite), ordonati in ordine descrescatoare a soldului. Se va avea in vedere posibilitatea ca suma platita in contul unei facturi sa fie diferita de suma de plata.

```
select A.id_client,nume, sum(de_plata)- sum(nvl(platit,0)) as sold
from clienti A, facturi B
where A.id_client=B.id_client
group by A.id_client, nume
having sum(de_plata)- sum(nvl(platit,0))>0
order by sold desc;
```

2.e) Plecand de la cerinta anteriora si tinand cont ca unii clienti ar putea avea inca facturi neplatite fara ca acestea sa fi ajuns la scadenta (termenul de plata inca nu este depasit), sa se scrie comanda SELECT care permite afisarea doar a clientilor cu restante reale, deci cu facturi neplatite si cu termenul de plata depasit (se va afisata ID_CLIENT, NUME, totalul sumelor restante cu termenul de plata depasit, ordonat descrescator dupa aceste sume).

```
select A.id_client,nume, sum(de_plata)- sum(nvl(platit,0)) as sold
from clienti A, facturi B
where A.id_client=B.id_client and data_scadenta<sysdate
group by A.id_client, nume
having sum(de_plata)- sum(nvl(platit,0))>0
order by sold desc;
```

2.f) Sa se particularizeze comanda anterioara doar pentru afisarea celui mai mare platitor/ restantier (ID_CLIENT, NUME, totalul de plata restanta).

```
select A.id_client,nume, sum(de_plata) as sold
from clienti A, facturi B
where A.id_client=B.id_client and data_scadenta<sysdate
group by A.id_client, nume
having sum(de_plata)= (select max(sum(de_plata))
                       from facturi
                       where data_scadenta<sysdate
                       group by id_client);
```

2.g) Comanda SELECT utilizata pentru afisarea clientilor (ID_CLIENT, NUME) care au efectuat plati (una sau mai multe) in ziua curenta, in valoare mai mare de 100 lei, respectiv a totalului platit de acesti clienti in ziua curenta, ordonat descrescator dupa acest total.

```
select A.id_client, nume, sum(platit) as total_platit
where A.id_client=B.id_client and trunc(data_plata)=trunc(sysdate)
group by A.id_client, nume,
having sum(platit) >100
order by total_platit desc;
```

2.h) Comanda SELECT utilizata pentru afisarea **primilor 3 clienti** (ID_CLIENT, NUME, TARIF_KW, SOLD) care au cel mai mare SOLD de platit (cele mai mari sume restante de plata). Se presupune existenta coloanei SOLD (calculata anterior cu triggere sau proceduri stocate).

```
select * from (select * from clienti order by sold desc)
   where rownum<=3;

sau

select rownum NR, A.* from (select * from clienti order by sold desc) A
     where rownum<=3;
```