**Exercice 1**

1-

create trigger T1 after insert

on Achat for each row

update Produit

set Qut\_st = Qut\_st + new.Qut\_Ent

where Ref = new.Ref;

2-

create trigger T2 after insert

on Vente for each row

update Produit

set Qut\_st = Qut\_st - new.Qut\_Sortie

where Ref = new.Ref;

Variante avec test et message d’erreur

**CREATE** **TRIGGER** `t2` **AFTER** **INSERT** **ON** `vente` **FOR** **EACH** **ROW** **begin**

**DECLARE** v\_val **INT**;

**DECLARE** v\_prod **VARCHAR**(50);

**DECLARE** v\_msg **VARCHAR**(255);

**SELECT** qut\_st **into** v\_val **from** produit **where** ref=**NEW**.ref;

**IF** **NEW**.qut\_sortie <= v\_val **THEN**

**update** produit **set** qut\_st = qut\_st-**NEW**.qut\_sortie **where** ref=**NEW**.ref;

**ELSE**

**SELECT** desig **into** v\_prod **from** produit **where** ref=**NEW**.ref;

**SET** v\_msg = **concat**('Pas assez de produits de type ',v\_prod);

**signal** **sqlstate** '45012' **set** message\_text = v\_msg;

**END** **IF**;

**end**

3-

create trigger T3 after delete

on Achat for each row

update Produit

set Qut\_st = Qut\_st - old.Qut\_Ent

where Ref = old.Ref;

4-

create trigger T4 after delete

on Vente for each row

update Produit

set Qut\_st = Qut\_st + old.Qut\_Sortie

where Ref = old.Ref;

**5-**

create trigger T6 before update

on Achat for each row

update Produit

set Qut\_st = Qut\_st – old.Qut\_Ent + new.Qut\_Ent

where Ref = new.Ref;

**6-**

create trigger T6 before update

on Vente for each row

update Produit

set Qut\_st = Qut\_st + old.Qut\_Sortie - new.Qut\_ Sortie

where Ref = new.Ref;

**Correction Exercice 2**

**Solution formateur :**

delimiter |  
create trigger before\_insert\_client before insert

on client for each row

begin

if substring(new.cp, 1, 2) not in (‘01’, ‘07’, ‘26’, ‘38’, ‘42’, ‘69’, ‘73’, ‘74’) then

-- signal permet de générer une erreur.

-- Le code 45000 signifie ‘unhandled user-defined exception’

signal sqlstate '45000' set message\_text = ‘County not accepted.’;

end if;

-- Je declare une variable locale

-- Cette variable va contenir le nom du conjoint, s’il existe

declare v\_conjoint\_name varchar(50);

if (new.conjoint is not null) then

select conjoint.nom into v\_conjoint\_name

from client conjoint

where new.conjoint = conjoint.numcl;

end if;

if new.nom != v\_conjoint\_name then

signal ‘45000’ set message\_text = ‘Conjoints\’ names are different.’;

end if;

end|

delimiter ;

**Solution internet :**

CREATE TRIGGER INS\_CLIENT  
         BEFORE INSERT ON CLIENT  
         FOR EACH ROW  
  
DECLARE  
  
nom\_conjoint CLIENT.NOM%TYPE ;  
compteur CLIENT.NUMCL%TYPE ;  
pb\_dept EXCEPTION ;  
pb\_conjoint1 EXCEPTION ;  
pb\_conjoint2 EXCEPTION ;  
  
BEGIN  
  
-- Contrainte sur le département  
  
IF TRUNC(:NEW.CP/1000) NOT IN (01, 07, 26, 38, 42, 69, 73, 74) THEN  
    RAISE pb\_dept ;  
END IF ;  
  
-- Contrainte sur le nom du conjoint (+ test d’existence du conjoint)  
  
IF NEW.CONJOINT IS NOT NULL THEN  
SELECT COUNT(\*), NOM  
INTO compteur, nom\_conjoint  
FROM CLIENT  
WHERE NUMCL = :NEW.CONJOINT  
GROUP BY NOM ;  
  
IF compteur = 0 THEN -- Pas de conjoint  
     RAISE pb\_conjoint1 ;  
END IF ;  
  
IF nom\_conjoint <> :NEW.NOM THEN  
    RAISE pb\_conjoint2 ;  
END IF ;  
END IF ;  
  
EXCEPTION  
  
WHEN pb\_dept THEN RAISE\_APPLICATION\_ERROR (-20501,  
‘Insertion impossible : le client n’habite pas en région Rhône-Alpes !’) ;  
  
WHEN pb\_conjoint1 THEN RAISE\_APPLICATION\_ERROR (-20502,  
‘Insertion impossible : le conjoint du client n’existe pas !’) ;  
  
WHEN pb\_conjoint2 THEN RAISE\_APPLICATION\_ERROR (-20503,  
‘Insertion impossible : le nom du conjoint est différent de celui du client !’) ;  
  
END ;

**Correction Exercice 3**  
  
**Solution formateur :**

delimiter |

create trigger before\_insert\_tabl before insert

on tabl for each row

begin

-- Variable qui va contenir le nombre de lignes presents dans la table

declare v\_number\_of\_rows;

-- Variable qui va contenir la valeur max de clenum

declare v\_max\_key\_value;

select count(\*) into v\_number\_of\_rows

from tabl;

if v\_number\_of\_rows = 0 then

set new.clenum := 1;

else

select max(clenum) into v\_max\_key\_value

from tabl;

set new.clenum := v\_max\_key\_value + 1;

end if;

end|

delimiter ;

**Solution internet :**

create trigger cleauto  
before insert on tabl  
for each row  
  
declare  
  
begin  
  
n integer;  
newkey integer;  
preums exception;  
  
-- Recherche s'il existe des tuples dans la table  
select count(\*) into n from tabl;  
if n=0 then  
     raise preums; -- Premiere insertion  
end if;  
  
-- Recherche la valeur de cle C la plus elevee  
-- et affecte C+1 a la nouvelle cle  
select max(clenum) into newkey from tabl;  
:new.clenum := newkey + 1;  
  
exception  
  
-- Premier numero = 1  
when preums then :new.clenum := 1;  
  
end;