```
-- TD dénormalisation

----------------
-- Exercice 1 --
----------------
set search_path to banque ;

-- j'ai fait mes requêtes sur les données suivantes :
select * from emprunt;
/*
 nemprunt | ncompte | montant
----------+---------+---------
        1 |     978 |    1000
        2 |     145 |    1500
        3 |     978 |    2000
        4 |     302 |    4200
       11 |     978 |    1000
(5 rows)
*/

select * from compte_client;
/*
 ncompte | ncli
---------+------
     145 |   10
     978 |   12
     978 |   11
     176 |   10
     302 |   12
     529 |   14
     302 |   13
      24 |   12
(8 rows)
*/


-- Question 1.1 : Ecrire la requête SQL qui donne pour chaque compte le nombre total de propriétaires et le montant total des
emprunts.

-- cette requête n'est pas si simple à cause des doublons engendrés par une jointure.
-- première solution : on pose une requête sur compte_client et on compte en sous-requête la somme des montants des emprunts
select ncompte,
       (select sum(montant) from emprunt e where e.ncompte = c.ncompte) as montant_emprunts,
       count(ncli) as nb_proprietaires
from compte_client c
group by ncompte;
```

```
/*
 ncompte | montant_emprunts | nb_proprietaires
---------+------------------+------------------
     145 |             1500 |                1
     302 |             4200 |                2
     978 |             4000 |                2
     176 |                  |                1
      24 |                  |                1
     529 |                  |                1
(6 rows)

*/
-- seconde solution : on pose la requête sur emprunt et on compte en cous-requête le nombre de propriétaires.
-- Cette solution est partielle puisqu'elle ne permet pas de voir les comptes sans emprunt
-- mais les calculs sont corrects.
select ncompte,
       sum(montant) as montant_emprunts,
       (select count(ncli) from compte_client c where e.ncompte = c.ncompte) as nb_proprietaires
from emprunt e
group by ncompte;

/*
 ncompte | montant_emprunts | nb_proprietaires
---------+------------------+------------------
     145 |             1500 |                1
     302 |             4200 |                2
     978 |             4000 |                2
(3 rows)
*/

-- Attention, avec une jointure on démultiplie les lignes et on risque de compter plusieurs fois le même montant
-- on peut enlever les doublons pour compter les clients mais pas pour la somme des montants.
-- Donc une requête comme celle ci-dessous ne marche pas (et enlever le distinct devant ncli crée une autre erreur de calcul)
select ncompte,
       sum(distinct montant) as montant_emprunts, --> ça ne va pas si on a le meme montant dans 2 prets
       count(distinct ncli) as nb_proprietaires
from compte_client c
  join emprunt using(ncompte)
group by ncompte;
/*
 ncompte | montant_emprunts | nb_proprietaires
---------+------------------+------------------
     145 |             1500 |                1
     302 |             4200 |                2
     978 |             3000 |                2 ==> erreur
(3 rows)
*/
```

```
-- Question 1.2 :
-- https://docs.postgresql.fr/11/sql-altertable.html
alter table compte add nb_proprietaires integer not null default 0 ;
alter table compte add montant_emprunts float not null default 0 ;

-- Question 1.3 :
update compte
  set nb_proprietaires = (select count(ncli) from compte_client c where c.ncompte = compte.ncompte),
      montant_emprunts = (select coalesce(sum(montant),0) from emprunt e where e.ncompte = compte.ncompte);
/*
select * from compte;

 ncompte | nag | solde | typecpte | nb_proprietaires | montant_emprunts
---------+-----+-------+----------+------------------+------------------
     529 |   3 |  1000 | Ass. Vie |                1 |                0
     978 |   2 |  1500 | depots   |                2 |             4000
     145 |   1 |  1020 | depots   |                1 |             1500
     176 |   1 |   500 | livret A |                1 |                0
     302 |   1 |   100 | depots   |                2 |             4200
      24 |   1 |     0 | livret A |                1 |                0
(6 rows)

*/

-- Question 1.4 :
CREATE or REPLACE FUNCTION calcul_nb_proprio() RETURNS trigger AS $$
    BEGIN
        if (TG_OP = 'INSERT' or TG_OP = 'UPDATE') then
          update compte set nb_proprietaires = nb_proprietaires+1
          where ncompte = NEW.ncompte;
        end if ;
        if (TG_OP = 'DELETE' or TG_OP = 'UPDATE') then
          update compte set nb_proprietaires = nb_proprietaires-1
          where ncompte = OLD.ncompte;
        end if ;
        if TG_OP = 'DELETE'
        then
          return OLD; -- ne surtout pas renvoyer NEW qui vaut NULL,
          -- ça annulerait le delete tout en conservant l'effet du trigger
        else return NEW;
        end if ;
        -- pour insert et delete sur compte_client => 1 update sur compte
        -- pour update de compte_client.ncompte => update de 2 lignes sur compte
    END;
$$ LANGUAGE plpgsql;

-- drop trigger trig_calcul_nb_proprio on compte_client;
CREATE TRIGGER trig_calcul_nb_proprio
```

```
BEFORE INSERT OR DELETE OR UPDATE OF ncompte ON compte_client
FOR EACH ROW EXECUTE FUNCTION calcul_nb_proprio();

insert into compte_client(ncompte,ncli) values (978,13);
/*
select * from compte;
 ncompte | nag | solde |   typecpte   | nb_proprietaires | montant_emprunts
---------+-----+-------+--------------+------------------+------------------
     145 |   1 |  1020 | cpte courant |                1 |             1500
     176 |   1 |   500 | livret A     |                1 |                0
     302 |   1 |   100 | cpte courant |                2 |             4200
     529 |   3 |  1000 | Ass. Vie     |                1 |                0
      24 |   1 |     0 | livret A     |                1 |                0
     978 |   2 |  1500 | cpte courant |                3 |             4000 => on est bien à 3 proprio
(6 rows)
*/

delete from compte_client where ncli=13 and ncompte=978;
/*
 ncompte | nag | solde |   typecpte   | nb_proprietaires | montant_emprunts
---------+-----+-------+--------------+------------------+------------------
     145 |   1 |  1020 | cpte courant |                1 |             1500
     176 |   1 |   500 | livret A     |                1 |                0
     302 |   1 |   100 | cpte courant |                2 |             4200
     529 |   3 |  1000 | Ass. Vie     |                1 |                0
      24 |   1 |     0 | livret A     |                1 |                0
     978 |   2 |  1500 | cpte courant |                2 |             4000 => on est revenu à 2 proprio
(6 rows)
*/

update compte_client set ncompte=978 where ncompte=302 and ncli=13;

-- et pour montant_emprunts
-- c'est quasiment la même fonction.
CREATE or REPLACE FUNCTION calcul_montant_emprunts() RETURNS trigger AS $$
    BEGIN
        if (TG_OP = 'INSERT' or TG_OP = 'UPDATE') then
          update compte set montant_emprunts = montant_emprunts + NEW.montant
          where ncompte = NEW.ncompte;
        end if ;
        if (TG_OP = 'DELETE' or TG_OP = 'UPDATE') then
          update compte set montant_emprunts = montant_emprunts - OLD.montant
          where ncompte = OLD.ncompte;
        end if ;
        if TG_OP = 'DELETE'
        then
          return OLD;
        else return NEW;
```

```
        end if ;
        -- à améliorer : si on modifie juste le montant, on pourrait faire 1 seul update au lieu de 2 ici
    END;
$$ LANGUAGE plpgsql;

-- drop trigger trig_calcul_montant_emprunts on emprunt;
CREATE TRIGGER trig_calcul_montant_emprunts
BEFORE INSERT OR DELETE OR UPDATE OF ncompte, montant ON emprunt
FOR EACH ROW EXECUTE FUNCTION calcul_montant_emprunts();

/*
select * from emprunt;
 nemprunt | ncompte | montant
----------+---------+---------
        1 |     978 |    1000
        2 |     145 |    1500
        3 |     978 |    2000
        4 |     302 |    4200
       11 |     978 |    1000
*/

delete from emprunt where nemprunt = 11;
/*
 ncompte | nag | solde |   typecpte   | nb_proprietaires | montant_emprunts
---------+-----+-------+--------------+------------------+------------------
     176 |   1 |   500 | livret A     |                1 |                0
     529 |   3 |  1000 | Ass. Vie     |                1 |                0
      24 |   1 |     0 | livret A     |                1 |                0
     302 |   1 |   100 | cpte courant |                1 |             4200
     145 |   1 |  1020 | cpte courant |                1 |             1500
     978 |   2 |  1500 | cpte courant |                3 |             3000 => on a baissé de 1000€
*/

insert into emprunt values(5,145,1000);
-- NOTICE:  montant total avant cet emprunt : 1500 ; et après : 2500

/*
ncompte | nag | solde |   typecpte   | nb_proprietaires | montant_emprunts
---------+-----+-------+--------------+------------------+------------------
     176 |   1 |   500 | livret A     |                1 |                0
     529 |   3 |  1000 | Ass. Vie     |                1 |                0
      24 |   1 |     0 | livret A     |                1 |                0
     302 |   1 |   100 | cpte courant |                1 |             4200
     978 |   2 |  1500 | cpte courant |                3 |             3000
     145 |   1 |  1020 | cpte courant |                1 |             2500 => on est passé à 2500
*/

update emprunt set ncompte=978 where nemprunt=5;
```

```
/*
 ncompte | nag | solde |   typecpte   | nb_proprietaires | montant_emprunts
---------+-----+-------+--------------+------------------+------------------
     176 |   1 |   500 | livret A     |                1 |                0
     529 |   3 |  1000 | Ass. Vie     |                1 |                0
      24 |   1 |     0 | livret A     |                1 |                0
     302 |   1 |   100 | cpte courant |                1 |             4200
     978 |   2 |  1500 | cpte courant |                3 |             4000 => on augmente de 1000
     145 |   1 |  1020 | cpte courant |                1 |             1500 => on diminue de 1000
*/

update emprunt set montant=500 where nemprunt=5; -- au lieu de 1000
/*
 ncompte | nag | solde |   typecpte   | nb_proprietaires | montant_emprunts
---------+-----+-------+--------------+------------------+------------------
     176 |   1 |   500 | livret A     |                1 |                0
     529 |   3 |  1000 | Ass. Vie     |                1 |                0
      24 |   1 |     0 | livret A     |                1 |                0
     302 |   1 |   100 | cpte courant |                1 |             4200
     145 |   1 |  1020 | cpte courant |                1 |             1500
     978 |   2 |  1500 | cpte courant |                3 |             3500 => la somme a baissé
*/

-- Question 1.5 :
-- En quoi la définition de ces nouveaux attributs peut modifier la solution
-- implémentée au TD précédent pour vérifier que seuls les comptes courants peuvent être partagés.

-- on peut supprimer le trigger trig_verif_compte et ajouter une contrainte
drop trigger trig_verif_compte on compte_client ;
alter table compte add constraint verif_compte check (typecpte = 'cpte courant' or nb_proprietaires < 2);

select nouveauCompte(100,1,13,'livret A',100.0);
--> ok
insert into compte_client(ncompte,ncli) values (100,10);
-- ERROR:  new row for relation "compte" violates check constraint "verif_compte"

update compte set typecpte='cpte courant' where ncompte=100;
insert into compte_client(ncompte,ncli) values (100,10);
--> ok

-- Question 1.6 :
-- De même, en quoi la définition de ces nouveaux attributs simplifie la vérification
-- du montant total des emprunts liés à un compte ?

-- On peut supprimer le trigger verif_endettement et ajouter une contrainte
drop trigger verif_endettement on emprunt;

alter table compte add constraint verif_endettement check (montant_emprunts <= 3*solde);
```

```
--> ERROR:  check constraint "verif_endettement" is violated by some row
-- on a une ancienne ligne qui ne satisfait pas la contrainte
-- on supprime la ligne qui pose problème :
delete from emprunt where ncompte=302 ;
-- Remarque : les triggers ne fonctionnent que pour les futures modif, mais ne vérifient pas les données déjà en base.
-- alors que lorsqu'on crée une contrainte, elle doit être vérifiée sur les données présentes dans la table.

alter table compte add constraint verif_endettement check (montant_emprunts <= 3*solde);

-- Conclusion : on a remplacé des triggers qui faisaient des requêtes globales sur compte et emprunt
-- en des triggers qui font des requêtes unitaires (la clause where utilise la clé primaire)
-- et des contraintes que l'on vérifie aussi sur la ligne courante de COMPTE.
-- Le calcul est moins couteux (il est incrémental, ne nécessite pas la lecture de plusieurs lignes)
```