```
-------------------------------------------------------------------------------
-- corr_contraintes.sql
-- tourne le 23 janvier 2018
-------------------------------------------------------------------------------
/* Contraintes SQL et non SQL.

Contraintes SQL (bonus en partiel/examen si presentation comme suit) :

client :
    idc pk
    nom not null
    age not null check 16<=age<=120
    avoir not null check 0<=avoir<=2000

village :
    idv pk
    ville not null
    activite -- null possible
    prix not null check 0<prix<=2000
    capacite not null check 1<=capacite<=1000

sejour :
    ids pk
    idc fk client not null -- rappel : fk n'implique pas not null
    idv fk village not null
    jour not null check 1<=jour<=365
    (idc, jour) unique

creer sequences pour client, village, sejour

Contraintes non SQL :

1. le nombre de sejours pour un centre pour un jour ne peut pas
depasser sa capacite :

  pour tout idv i de capacite n dans village,
  il y a au plus n lignes avec idv i pour chaque jour j dans sejour

2. l'avoir d'un client plus la somme des prix de ses sejours ne peut
exceder 2000 :

  pour tout idc i dans client : avoir + S <= 2000
  en effet, un client part de 2000 et achete des sejours, donc :
  avoir + somme des prix de ses sejours presents + somme des prix de
  ses sejours detruits = 2000 ;
  donc avoir + somme des prix de ses sejours presents <= 2000 ;
  comment obtient-on S : considerons toutes les lignes d'idc i dans
  sejour, et pour chacune, a partir de sa colonne idv dans sejour, son
  prix dans village identifie par idv ; on note S la somme de tous ces
  prix
*/

-------------------------------------------------------------------------------
/* Rappel :
Null :
Peuvent etre "null" les valeurs intuitivement non
indispensable au fonctionnement de l'application. De plus, par
convention dans le module les colonnes utilisees en parametres ou a
l'interieur d'un traitement ne peuvent etre "null", mais il n'est pas
impose que celles en sortie d'un traitement ne le soient pas.
Rappel : fk et check n'impliquent pas not null.

Contraintes non SQL :
en general, il s'agit de compter un ensemble de lignes d'une table, ou de
comparer des valeurs dans deux tables, ou des variantes de ces situations
(ex : somme d'un ensemble de lignes). */

-------------------------------------------------------------------------------
-- Ordres SQL et modeles d'ordre : completer avec nouveaux outils TD precedent
-------------------------------------------------------------------------------
-- Ordres tapes par le programmeur :
-- traitement 2 :

create or replace procedure traitement2(
    l_idc client.idc%type,
    la_ville village.ville%type,
    le_jour sejour.jour%type,
    l_idv out village.idv%type,
    l_ids out sejour.ids%type,
    l_activite out village.activite%type)
is
    cursor c is
        select idv, prix, activite
            from village
            where ville = la_ville
            order by prix desc;
    le_prix village.prix%type;
begin
    open c;
    fetch c into l_idv, le_prix, l_activite;
    if c%found then
        l_ids := seq_sejour.nextval;
        insert into sejour
            values(l_ids, l_idc, l_idv, le_jour);
        update client
            set avoir = avoir - le_prix
            where idc = l_idc;
    else
        l_idv := -1;
        l_ids := -1;
        l_activite := 'neant';
    end if;
end;
/

select * from client;
select * from village;

select * from sejour;

declare
    iv village.idv%type;
    l_ids sejour.ids%type;
    a village.activite%type;
begin
    traitement2(1, 'Chatelaillon', 361, iv, l_ids, a);
    dbms_output.put_line('idv '||iv||', ids '||l_ids||', activite '||a);
    traitement2(1, 'Chatelaillon', 360, iv, l_ids, a);
    dbms_output.put_line('idv '||iv||', ids '||l_ids||', activite '||a);
end;
/
select * from sejour;

-------------------------------------------------------------------------------
```