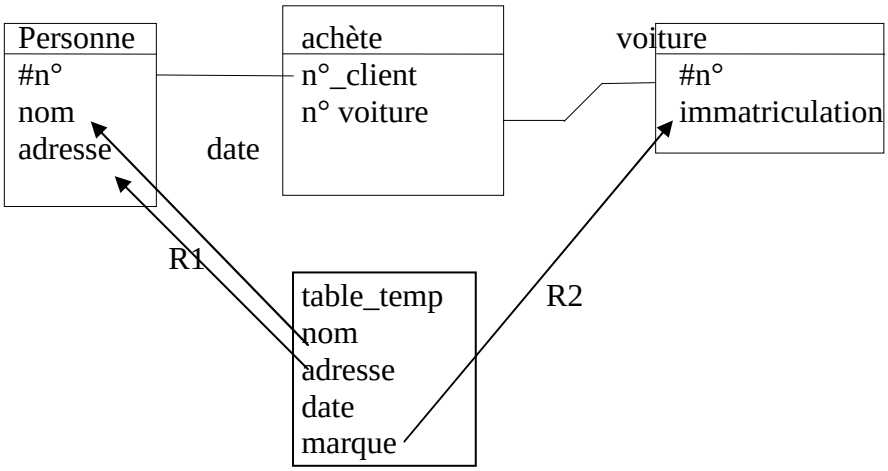
TD SQL



pour saisir les achats chez un concessionnaire

1) on saisit dans une table temporaire (plusieurs ventes d'un coup)

2) on écrit deux requêtes R1 et R2 qui ajoutent à personne et voiture les informations saisies si elles n'y sont pas déjà (requêtes ajout; on peut faire une seule requête)
nota: ces info seront automatiquement numérotées

1.1.)
```
Insert into voiture(immatriculation)
select table_temp.immatriculation   from table_temp
where table_temp.immatriculation   not in
        (select immatriculation from voiture)
```

nota:

a) bien montrer qu'on ne peut pas écrire
where table_temp.immatriculation   not in voiture
puisque voiture n'est pas un ensemble d'immatriculations

b) montrer qu'il ne faut surtout pas écrire
```
Insert into voiture(immatriculation)
select table_temp.immatriculation   from table_temp, voiture
where table_temp.immatriculation   not in
        (select immatriculation from voiture)
```
l'ajout de la table **voiture** provoque le calcul du produit cartésien
(table_temp x voiture) donc s'il y a n lignes dans la table voiture, multiplication par n du nombre d'immatriculations à entrer, MEME SI, COMPTE TENU DE LA CONDITION, LE RESULTAT SERA LE MEME

1.2.
```
Insert into personne(nom,adresse)
select table_temp.nom,  table_temp.adresse from table_temp
where not  exists      (select personne.nom, personne.adresse from personne
                        where (table_temp.nom= personne.nom)
                        and
                        (table_temp.adresse= personne.adresse ) )
```

c) Bien faire comprendre la différence avec

```
Insert into personne(nom,adresse)
select table_temp.nom,  table_temp.adresse from table_temp
where not  exists (select personne.nom, personne.adresse from personne)
```

qui signifie "entrer les noms et adresse seulement si la table personne est VIDE"

d) montrer la différence avec

```
Insert into personne(nom,adresse)
select table_temp.nom,  table_temp.adresse from table_temp
where
        nom not in (select nom from personne)
        and
        adresse not (select adresse from personne)
```

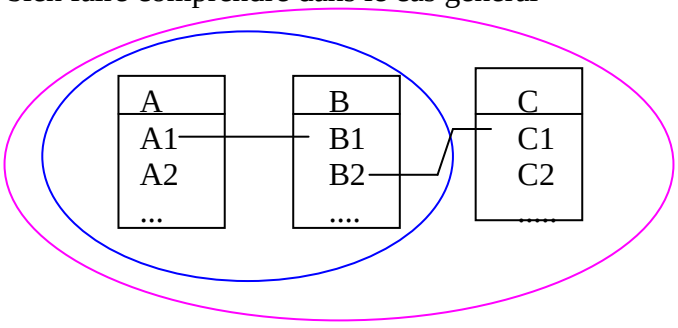
e) montrer que les deux requêtes précédentes peuvent être groupées, mais acun intérêt, au contraire.

3) on écrit une requête R3 qui prend les n° dans personne et voiture (à condition qu'ils correspondent aux info de table_temp), et les ranges dans achète

3.1.

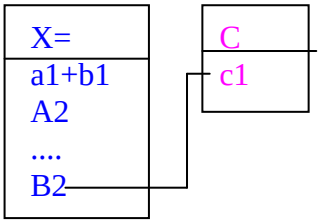
```
insert into achète (n° client, n°voiture)
select   personne.n°,   voiture.n°   from
(personne inner join personne_tmp on
        (personne.nom=personne_tmp.nom)
        and
        (personne.adresse=personne_tmp.adresse)
) inner join voiture on voiture.immatriculation=personne_tmp.immatriculation
```

3.2. bien faire comprendre dans le cas général



SELECT * FROM (A INNER JOIN B ON A1=B1) INNER JOIN C ON C=B2

Faire ressortir qu'une jointure donne un ensemble (rangé dans une table intermédiaire) X qu'on peut très bien réutiliser pour une autre jointure:



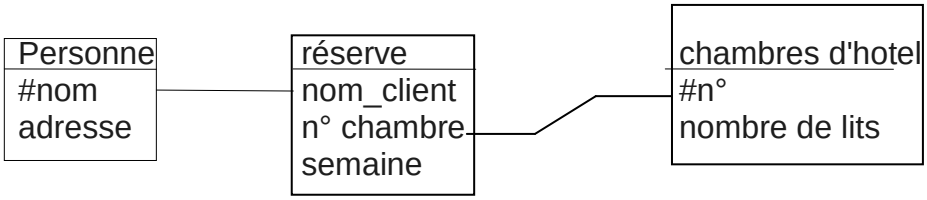
rappeler une fois encore la différence entre

select * from A,B where A.x=B.Y
qui fait le produit cartésien avant d'éliminer les couples indésirables ==> cher

et

select * from A inner join B on A.x=B.x
rappeler que si a comporte n éléments et B (trié) comporte p éléments ==> cout n.LOG2(p)

PROBLEME 2



Problème 1: faire une requête sélection qui affiche les chambres disponibles une semaine donnée (infaisable en mode graphique)

1.1. faire une requête R1 qui affiche les chambres libres .
E = SELECT [n°_chambre] , semaine FROM réserve WHERE semaine= [quelle semaine?]

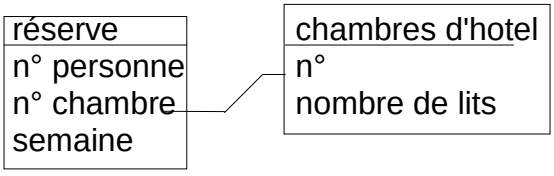
1.2. Faire la requête R2 qui sélectionnera les chambres d'hotel "NOT IN E".

a) faire une requête qui sélectionne toutes les chambres de l'hotel:

SELECT [chambres_d'hotel].n°, [nombre de lits] FROM [chambres_d'hotel]

b) enlever les chambres réservées cette semaine:
SELECT [chambres_d'hotel]n°, [nombre de lits] FROM [chambres_d'hotel],
WHERE chambres.n° NOT IN
(SELECT [n°_chambre] FROM réserve WHERE semaine= [quelle semaine?])

1.2. Constater qu'il y a deux versions:
a) **en laissant la jointure** réserve.n°chambre chambre.n°



qui génère une clause INNER JOIN:

```
SELECT [chambres_d'hotel].n° j,[nombre de lits]   FROM
[chambres_d'hotel] INNER JOIN  réserve ON [n°chambre]=n°
WHERE chambres.n°  NOT IN
(SELECT [n°_chambre] FROM réserve  WHERE semaine= [quelle semaine?])
```

Dans cette version, un numéro de chambre n'apparaît que si la chambre a été réservée une autre semaine que celle demandée, mais pas les chambres jamais réservées.
Discuter la solution de remplacer INNER JOIN par LEFT JOIN.

b) **en supprimant la jointure** cela affiche effectivement toutes les chambres libres. Expliquer la raison!

```
SELECT [chambres_d'hotel].n°    FROM [chambres_d'hotel]
WHERE [chambres d'hotel].n°  NOT IN
(SELECT [n°_chambre] FROM réserve  WHERE semaine= [quelle semaine?])
```

Discuter la solution:

```
SELECT DISTINCTROW [chambres d'hotel].n°
FROM [chambres d'hotel], réserve
WHERE ((([chambres d'hotel].n°) NOT IN
(SELECT réserve.[n° chambre] FROM réserve
WHERE ([semaine]=[quelle semaine?])));
```

En fait c'est idiot de faire le produit cartésien pour ne s'intéresser qu'à ce qui est distinct!

Problème 2: annulation des réservations: il ne faut pas enlever les réservations de la table car on veut garder un historique. faire une saisie contrôlée: créer une table des annulations de réservations: il y a une contrainte: les annulations doivent correspondre à des réservations (même chambre, même semaine, même personne)

```
INSERT INTO annulations ( personne, [date], [n° chambre] )
SELECT DISTINCTROW annulation_tmp.personne, annulation_tmp.date,
annulation_tmp.[n° chambre]
FROM annulation_tmp, annulations, réserve
WHERE (((Exists (SELECT DISTINCTROW [nom client], chambre, semaine
        FROM réserve
        WHERE
        (
                    ( [nom client]=              annulation_tmp.personne)
           AND      (  chambre=                  annulation_tmp.[n° chambre])
           AND      (  semaine=              annulation_tmp.date  )
        )
    ))<>False)
AND ((Not Exists (SELECT DISTINCTROW [nom client], chambre, [numéro de
semaine]
    FROM annulations
    WHERE
```

```
(
                ( [nom client]=                 annulations.personne)
    AND      (  chambre=                        annulations.[n° chambre])
    AND      (  semaine=                 annulations.date  )
)
))<>False));
```