

## 1 Création et manipulation d'une base de données

Dans ce TD/TP, nous allons créer la base de données du TD d'algèbre relationnelle. Nous rappelons que cette base est constituée des relations suivantes :

### *Tables*

---

*Produits*(Constructeur : caractere, Modele : entier, Type : chaine)  
*PC*(Modele : entier, Vitesse : entier, RAM : entier, HD : reel, CD : chaine, Prix : reel)  
*Portables*(Modele : entier, Vitesse : entier, RAM : entier, HD : reel, Ecran : reel, Prix : reel)  
*Imprimantes*(Modele : entier, Couleur : boolean, Type : chaine, Prix : reel)

### *Contraintes*

---

*Les clefs donnees ci – dessus*

---

*Pas de valeur nulle*

---

$PC[Modele] \subseteq Produits[modele]$

$Portables[Modele] \subseteq Produits[modele]$

$Imprimantes[Modele] \subseteq Produits[modele]$

---

$dom(Produits.Type) = \{laptop, pc, printer\}$

$dom(PC.CD) = \{6x, 8x\}$

$dom(Imprimantes.Type) = \{dry, ink - jet, laser\}$

### 1.1 Connexion à votre base de données Oracle

Pour manipuler votre base de données, vous devez utiliser l'interpréteur de commande d'Oracle : *sqlplus*. Pour lancer *sqlplus*, utilisez la commande suivante : *sqlplus* (votre login est aussi votre mot de passe). Vous êtes dans l'interpréteur SQL, et maintenant, les seules commandes acceptées sont des commandes SQL. Vous pouvez obtenir la liste des commandes SQL en tapant *help index* et l'aide pour une commande particulière en tapant *help nom\_commande*.

### 1.2 Manipulation des tables via le SGBD

Nous allons voir comment créer une table, la modifier et la supprimer.

#### 1.2.1 Création d'une table

La création d'une table s'effectue grâce à la commande *create table*. Celle-ci s'utilise de la façon suivante :

```
CREATE TABLE nom_table (attribut1 type1, ..., attributn typen);
```

Vous utiliserez les types de données *int*, *float* et *varchar2(longueur max)*.

Créez à l'aide de cette commande créez les tables de la base (sans clef ni contrainte). Tapez *select \* from tab;* pour obtenir un descriptif du contenu courant de la base. Vous pouvez taper *describe nom\_table;* pour obtenir le descriptif d'une table.

**Réponse :** Voir le fichier *creation-ss-key.sql* donné un peu plus loin.

Insérer les trois premiers tuples (voir énoncé de TD) de la table *produits* à l'aide de la commande :

```
INSERT INTO nom_table VALUES (val_attribut1, ..., val_attributn);
```

**Réponse :**

```
INSERT INTO Produits VALUES('A',1001,'pc');
INSERT INTO Produits VALUES('A',1002,'pc');
INSERT INTO Produits VALUES('A',1003,'pc');
```

Nous rappelons que pour voir le contenu d'une table vous pouvez utiliser la requête : `SELECT * FROM nom_table;`

Changer le nom du constructeur du premier tuple à l'aide de la commande :

```
UPDATE nom_table SET attribut = nouvelle_valeur WHERE condition;
```

La commande UPDATE instancie la valeur de *attribut* à *nouvelle\_valeur* pour tous les tuples vérifiant la condition.

**Réponse :**

```
UPDATE produits SET constructeur='Z' WHERE modele=1001;
```

### 1.2.2 Modification d'une table – Ajout de contraintes

À l'aide de la commande ALTER TABLE, modifiez les tables que vous venez de créer afin d'y indiquer les clefs primaires et étrangères, les droits à la valeur nulle, ainsi que les contraintes de domaine d'attribut.

Notez qu'une clef ne doit pas pouvoir prendre la valeur null. Si vous n'avez pas indiqué, lors de la création d'une table, que la valeur null est interdite pour les attributs utilisez la commande suivante :

```
ALTER TABLE nom_table MODIFY attribut NOT NULL;
```

**Réponse :**

```
ALTER TABLE produits MODIFY modele NOT NULL;
ALTER TABLE pc MODIFY modele NOT NULL;
ALTER TABLE portables MODIFY modele NOT NULL;
ALTER TABLE imprimantes MODIFY modele NOT NULL;
...
```

La syntaxe à utiliser pour les clefs primaires est :

```
ALTER TABLE nom_table ADD PRIMARY KEY (attribut);
```

La syntaxe à utiliser pour les clefs étrangères est :

```
ALTER TABLE nom_table1 ADD FOREIGN KEY (attribut1) REFERENCES nom_table2 (attribut2);
```

**Réponse :**

```
ALTER TABLE produits ADD PRIMARY KEY (modele);
ALTER TABLE pc ADD PRIMARY KEY (modele);
ALTER TABLE pc ADD FOREIGN KEY (modele) REFERENCES produits (modele);
ALTER TABLE portables ADD PRIMARY KEY (modele);
ALTER TABLE portables ADD FOREIGN KEY (modele) REFERENCES produits (modele);
ALTER TABLE imprimantes ADD PRIMARY KEY (modele);
ALTER TABLE imprimantes ADD FOREIGN KEY (modele) REFERENCES produits (modele);
ALTER TABLE produits ADD CHECK (Type='laptop' or Type='pc' or Type='printer');
ALTER TABLE pc ADD CHECK (CD='56x' or CD='48x');
ALTER TABLE imprimantes ADD CHECK ((Couleur=0 or Couleur=1) and (Type='dry' or Type='ink-jet' or Type='laser'));
```

### 1.2.3 Suppression d'une table

La suppression d'une table s'effectue grâce à la commande `DROP TABLE nom_table`.

Supprimez toutes vos tables en commençant par produit. Que se passe-t-il ? Pourquoi ?

**Réponse :** À cause des contraintes de références, la table *produits* doit être supprimée en dernier.

```
DROP TABLE pc;
DROP TABLE portables;
DROP TABLE imprimantes;
DROP TABLE produits;
```

### 1.3 Remplissage d'une base de données

Récupérez et lisez le fichier `creation.sql` sur le compte de votre chargé de TD. Ce fichier contient les commandes de création des tables de la base de données. Chargez ce fichier depuis *sqlplus* grâce à la commande *start creation.sql*. Affichez le contenu de la base.

Insérez le premier n-uplet de chacune des tables *produits*, *pc*, *portables* et *imprimantes*, dans cet ordre. Que constatez-vous ? Pourquoi ?

**Réponse :** Les premiers tuples de portables et imprimantes ne peuvent être insérés car ils n'existent pas dans produits : violation de la contrainte d'intégrité référentielle.  
Insérer les trois premiers n-uplets de la table *imprimantes* en insérant d'abord les tuples appropriés dans la table *produits*.

**Réponse :**

```
INSERT INTO Produits VALUES('D',3001,'printer');
INSERT INTO imprimantes VALUES(3001,1,'ink-jet',99);
INSERT INTO Produits VALUES('B',3002,'printer');
INSERT INTO imprimantes VALUES(3002,1,'ink-jet',69);
INSERT INTO Produits VALUES('D',3003,'printer');
INSERT INTO imprimantes VALUES(3003,0,'laser',209);
Supprimer les n-uplets du constructeur D de la table produits en utilisant la commande :
```

```
DELETE FROM nom_table WHERE condition
```

Que constatez-vous ? Pourquoi ?

**Réponse :** Il y a violation de la contrainte d'intégrité référentielle. Il aurait d'abord fallu supprimer les tuples correspondants dans la table *imprimantes*.  
Supprimez tous les tuples que vous avez inséré dans les tables (sans supprimer les tables).

**Réponse :**

```
DELETE FROM imprimantes;
DELETE FROM pc;
DELETE FROM portables;
DELETE FROM produits;
```

Récupérez le fichier `remplissage.sql` sur le compte de votre chargé de TD. Ce fichier contient les commandes de remplissage des tables de la base de données. Lisez et chargez ce fichier.

**Réponse :** Fichier `creation-ss-key.sql`

```
drop table pc;
drop table portables;
drop table imprimantes;
drop table produits;

create table produits(
constructeur char,
modele int,
type varchar2(10)
);

create table pc(
modele int,
vitesse float,
ram int,
hd float,
cd char(3),
```

```

prix float
);

create table portables(
modele int,
vitesse float,
ram int,
hd float,
ecran float,
prix float
);

create table imprimantes(
modele int,
couleur int,
type varchar2(10),
prix float
);

```

Fichier **creation.sql**

```

drop table pc;
drop table portables;
drop table imprimantes;
drop table produits;

create table produits(
constructeur char not null,
modele int not null,
type varchar2(10) not null,
primary key (modele),
check (type='pc' or type='laptop' or type='printer')
);

create table pc(
modele int not null,
vitesse float not null,
ram int not null,
hd float not null,
cd char(3) not null,
prix float not null,
primary key (modele),
foreign key (modele) references produits (modele)
);

create table portables(
modele int not null,
vitesse float not null,
ram int not null,
hd float not null,
ecran float not null,
prix float not null,
primary key (modele),
foreign key (modele) references produits (modele)
);

```

```

create table imprimantes(
modele int not null,
couleur int not null,
type varchar2(10) not null,
prix float not null,
primary key (modele),
foreign key (modele) references produits (modele),
check (couleur=0 or couleur=1),
check (type='ink-jet ' or type='laser ' or type='dry ')
);

```

#### Fichier remplissage.sql

```

INSERT INTO Produits values('A',1001,'pc ');
INSERT INTO Produits values('A',1002,'pc ');
INSERT INTO Produits values('A',1003,'pc ');
INSERT INTO Produits values('B',1004,'pc ');
INSERT INTO Produits values('B',1006,'pc ');
INSERT INTO Produits values('B',3002,'printer ');
INSERT INTO Produits values('B',3004,'printer ');
INSERT INTO Produits values('C',1005,'pc ');
INSERT INTO Produits values('C',1007,'pc ');
INSERT INTO Produits values('D',1008,'pc ');
INSERT INTO Produits values('D',1009,'pc ');
INSERT INTO Produits values('D',1010,'pc ');
INSERT INTO Produits values('D',2001,'laptop ');
INSERT INTO Produits values('D',2002,'laptop ');
INSERT INTO Produits values('D',2003,'laptop ');
INSERT INTO Produits values('D',3001,'printer ');
INSERT INTO Produits values('D',3003,'printer ');
INSERT INTO Produits values('E',2004,'laptop ');
INSERT INTO Produits values('E',2008,'laptop ');
INSERT INTO Produits values('F',2005,'laptop ');
INSERT INTO Produits values('G',2006,'laptop ');
INSERT INTO Produits values('G',2007,'laptop ');
INSERT INTO Produits values('H',3005,'printer ');
INSERT INTO Produits values('I',3006,'printer ');

INSERT INTO pc values(1001,3,256,120,'48x',795);
INSERT INTO pc values(1002,2,256,120,'48x',699);
INSERT INTO pc values(1003,3.5,512,180,'48x',999);
INSERT INTO pc values(1004,3.5,1024,180,'56x',1099);
INSERT INTO pc values(1005,3.5,256,160,'56x',1099);
INSERT INTO pc values(1006,4.5,1024,200,'56x',1199);
INSERT INTO pc values(1007,4.5,1024,300,'56x',1249);
INSERT INTO pc values(1008,4,1024,160,'56x',1699);
INSERT INTO pc values(1009,4.5,1024,180,'56x',1299);
INSERT INTO pc values(1010,3.4,256,100,'56x',700);

INSERT INTO portables values(2001,2,256,60,14, 1099);
INSERT INTO portables values(2002,2.1,192,40,15.4,1298);
INSERT INTO portables values(2003,2.1,1024,50,15,1499);
INSERT INTO portables values(2004,3,256,60,15.5,1399);
INSERT INTO portables values(2005,3.5,256,50,15.4,1299);
INSERT INTO portables values(2006,2,128,45,17,1099);

```

```

INSERT INTO portables values(2007,2.5,256,80,17, 1999);
INSERT INTO portables values(2008,2,256,60,17,1199);

INSERT INTO imprimantes values(3001,1,'ink-jet ',99);
INSERT INTO imprimantes values(3002,1,'ink-jet ',69);
INSERT INTO imprimantes values(3003,0,'laser ',209);
INSERT INTO imprimantes values(3004,1,'laser ',391);
INSERT INTO imprimantes values(3005,0,'ink-jet ',59);
INSERT INTO imprimantes values(3006,1,'dry ',129);

```

## 2 Interrogation d'une base de données

Sur la base de données précédemment créée, nous allons poser en SQL les requêtes suivantes.

1. Quels sont les portables (laptop en anglais) référencés dans la relation Produits ?

La réponse doit être :

```

modele
-----
    2001    2008
    2002    2005
    2003    2006
    2004    2007
(8 lignes)

```

2. Quelle est la liste des constructeurs ?

La réponse doit être :

```

constructeur
-----
A
B
C
D
E
F
G
H
I
(9 lignes)

```

3. Quels sont les modèles des ordinateurs référencés ?

La réponse doit être :

```

modele
-----
    1001    1010
    1002    2001
    1003    2002
    1004    2003
    1006    2004
    1005    2008
    1007    2005
    1008    2006
    1009    2007
(18 lignes)

```

**Réponse :** Les contraintes référentielles imposées sur le schéma font que tous les pc, portables et imprimantes sont tous nécessairement dans la table produits, mais un pc, un portable ou une imprimante n'est pas forcément dans la correspondante.

4. Quels sont les PC avec 256 Mo de RAM ?

La réponse doit être :

MODELE

-----

1001

1002

1005

1010

(4 lignes)

5. Quels sont les PC à exactement moins de 1100 Euros ?

La réponse doit être :

MODELE

-----

1001

1002

1003

1004

1005

1010

(6 lignes)

6. Quels sont les portables avec un disque d'au moins (ou égal à) 50 Go et a moins de (ou égal à) 1400 Euros ?

La réponse doit être :

modele

-----

2001

2004

2005

2008

(4 lignes)

7. Quelles sont les imprimantes laser couleur ?

La réponse doit être :

modele

-----

3004

(1 ligne)

8. Quels sont les portables du constructeur D ayant au moins 256 Mo de RAM ?

La réponse doit être :

modele

-----

2001

2003

(2 lignes)

9. Quels sont les ordinateurs du constructeur D ayant au moins 256 Mo de RAM ?

La réponse doit être :

modele

-----

1008

1009

1010

2001

2003

(5 lignes)

10. Quels sont les ordinateurs coûtant exactement moins de 1200 Euros ?  
La réponse doit être :

MODELE

-----

1001  
1002  
1003  
1004  
1005  
1006  
1010  
2001  
2006  
2008

(10 lignes)

11. Quels sont les constructeurs fournissant des imprimantes et des pc ?  
La réponse doit être :

constructeur

-----

B  
D

(2 lignes)

12. Quels sont les constructeurs couvrant tous les types de produits ?  
La réponse doit être :

constructeur

-----

D

(1 ligne)

13. Quels sont les constructeurs ne fournissant que des PC ?  
La réponse doit être :

constructeur

-----

A  
C

(2 lignes)

14. Quels sont les constructeurs ne fournissant qu'un type de materiel ?  
La réponse doit être :

constructeur

-----

A  
C  
E  
F  
G  
H  
I

(7 lignes)

15. Quels sont les constructeurs disposant de pc avec au moins 1024 Mo de RAM, un cd 56x et d'un portable avec un écran supérieur a 15 pouces et pouvant fournir une imprimante à moins de 300 Euros ?  
La réponse doit être :

constructeur

-----

D

(1 ligne)



16. Sachant que pour faire de la PAO il faut au moins 1024 Mo de RAM, une vitesse de 3 Ghz ou 3.5 , 160 Go minimum de hd ainsi qu'une imprimante couleur, quelle est le constructeur fournissant cette configuration et à quel prix ?

La réponse doit être :

```

constructeur | prix
-----+-----
B             | 1168
B             | 1490
(2 lignes)

```

17. Quelles sont les tailles de disques durs qui sont utilisées par au moins 2 PC ?

La réponse doit être :

```

hd
----
120
160
180
(3 lignes)

```

18. Quels sont les constructeurs qui vendent au moins deux différents ordinateurs avec une fréquence d'au moins 3 Ghz ?

La réponse doit être :

```

constructeur
-----
A
B
C
D
(4 lignes)

```

19. Quels sont les constructeurs vendant l'ordinateur le plus rapide ?

La réponse doit être :

```

constructeur
-----
B
C
D
(3 lignes)

```

20. Quels sont les constructeurs vendant exactement trois types de PC ?

La réponse doit être :

```

constructeur
-----
A
D
(2 lignes)

```

Réponse :

```

/* ----- requete 1 ----- */

```

```

select modele
from produits
where type = 'laptop';

```

```

/* ----- requete 2 ----- */

```

```

select distinct constructeur

```

```
from produits;
```

```
/* ----- requete 3 ----- */
```

```
select modele
from produits
where type = 'pc' or type = 'laptop';
```

```
/* ----- requete 4 ----- */
```

```
select modele
from pc
where ram = 256;
```

```
/* ----- requete 5 ----- */
```

```
select modele
from pc
where prix < 1100;
```

```
/* ----- requete 6 ----- */
```

```
select modele
from portables
where hd>=50 and prix<=1400;
```

```
/* ----- requete 7 ----- */
```

```
select modele
from imprimantes
where couleur=1 and type='laser';
```

```
/* ----- requete 8 ----- */
```

```
select pr.modele
from produits pr,portables po
where pr.modele=po.modele and constructeur='D' and ram>=256;
```

```
/* ----- requete 9 ----- */
```

```
(select pr.modele
from produits pr,pc
where pr.modele=pc.modele and constructeur='D' and ram>=256)
union
(select pr.modele
from produits pr,portables po
where pr.modele=po.modele and constructeur='D' and ram>=256);
```

```
/* ----- requete 10 ----- */
```

```
(select modele
from pc
where prix<1200)
union
(select modele
from portables
where prix<1200);
```

```
/* ----- requete 11 ----- */
```

```

(select constructeur
 from produits
 where type='pc')
intersect
(select constructeur
 from produits
 where type='printer ');

select distinct p1.constructeur
from produits p1, produits p2
where p1.constructeur = p2.constructeur
      and p1.type='pc' and p2.type='printer ';

/* ----- requete 12 ----- */

(select constructeur
 from produits
 where type='pc')
intersect
(select constructeur
 from produits
 where type='printer ')
intersect
(select constructeur
 from produits
 where type='laptop ');

select constructeur
from produits
group by constructeur
having count(distinct type)=(select count(distinct type) from produits);

select distinct p1.constructeur
from produits p1, produits p2, produits p3
where p1.constructeur = p2.constructeur
      and p2.constructeur = p3.constructeur
      and p1.type='pc' and p2.type='printer ' and p3.type='laptop ';

/* ----- requete 13 ----- */

(select constructeur
 from produits
 where type='pc')
minus
(select constructeur
 from produits
 where type='printer ')
minus
(select constructeur
 from produits
 where type='laptop ');

/* ----- requete 14 ----- */
select constructeur
from produits
group by constructeur
having count(distinct type)=1;

(
(select constructeur
 from produits

```

```

    where type='pc')
minus
(select constructeur
 from produits
 where type='printer')
minus
(select constructeur
 from produits
 where type='laptop')
)
union
(
(select constructeur
 from produits
 where type='printer')
minus
(select constructeur
 from produits
 where type='pc')
minus
(select constructeur
 from produits
 where type='laptop')
)
union
(
(select constructeur
 from produits
 where type='laptop')
minus
(select constructeur
 from produits
 where type='printer')
minus
(select constructeur
 from produits
 where type='pc')
);

/* ----- requete 15 ----- */

(select constructeur
 from produits p,pc
 where p.modele=pc.modele and ram>=1024 and cd='56x')
intersect
(select constructeur
 from produits p, portables po
 where p.modele=po.modele and ecran>15)
intersect
(select constructeur
 from produits p,imprimantes i
 where p.modele=i.modele and prix<300);

/* ----- requete 16 ----- */
/*----- sol 1 : sans calcul du prix -----*/
(
(select produits.constructeur
 from (select * from pc where ram>=1024 and (vitesse=3 or vitesse=3.5)
 and Hd>=160 ) ordi, produits
 where ordi.modele=produits.modele)

```

```

union
(select constructeur
from (select * from portables where ram>=1024 and (vitesse=3 or vitesse=3.5)
      and Hd>=160 ) laptops, produits
where laptops.modele=produits.modele)
)
intersect
(select produits.constructeur
from (select * from imprimantes where couleur=1) impr, produits
where impr.modele=produits.modele);

/*----- sol 2 -----*/
(select ordi.constructeur, ordi.prix+impr.prix prix
from (select * from produits p,pc where p.modele=pc.modele and ram>=1024
      and (vitesse=3 or vitesse=3.5) and Hd>=160 ) ordi,
      (select * from produits p,imprimantes i where p.modele=i.modele
      and couleur=1) impr
where ordi.constructeur=impr.constructeur)
union
(select ordi.constructeur, ordi.prix+impr.prix prix
from (select * from produits p,portables po where p.modele=po.modele
      and ram>=1024 and (vitesse=3 or vitesse=3.5) and Hd>=160 ) ordi,
      (select * from produits p,imprimantes i where p.modele=i.modele
      and couleur=1) impr
where ordi.constructeur=impr.constructeur);

/* ----- requete 17 ----- */

select HD
from pc
group by HD
having count(distinct modele)>=2;

select distinct p1.HD
from pc p1, pc p2
where p1.HD=p2.HD and p1.modele <> p2.modele;

/* ----- requete 18 ----- */

select distinct ordi1.constructeur
from (select constructeur,p.modele,vitesse from pc,produits p
      where pc.modele=p.modele and vitesse>=3
union
select constructeur,p.modele,vitesse from portables po,produits p
      where po.modele=p.modele and vitesse>=3) ordi1,
      (select constructeur,p.modele,vitesse from pc,produits p
      where pc.modele=p.modele and vitesse>=3
union
select constructeur,p.modele,vitesse from portables po,produits p
      where po.modele=p.modele and vitesse>=3) ordi2
where ordi1.constructeur=ordi2.constructeur
and ordi1.modele<>ordi2.modele;

/* ----- requete 19 ----- */

select distinct constructeur
from ((select modele,vitesse from pc)
union (select modele,vitesse from portables)) ordi, produits pr
where ordi.modele=pr.modele
and vitesse>=all((select vitesse from pc)

```

```

union (select vitesse from portables));

select constructeur
from produits p,
((select distinct modele from produits where type='pc' or type='laptop')
 minus
 (select distinct ordi1.modele
  from ((select modele,vitesse from pc )
        union (select modele,vitesse from portables )) ordi1,
        ((select modele,vitesse from pc )
        union (select modele,vitesse from portables )) ordi2
  where ordi1.modele<ordi2.modele
        and ordi1.vitesse<ordi2.vitesse)) rapide
where p.modele=rapide.modele;

/* ----- requete 20 ----- */

select constructeur
from produits
where type='pc'
group by constructeur
having count(distinct modele)=3;

(select distinct p1.constructeur
from (select modele,constructeur from produits where type='pc') p1,
     (select modele,constructeur from produits where type='pc') p2,
     (select modele,constructeur from produits where type='pc') p3
where p1.constructeur=p2.constructeur
     and p2.constructeur=p3.constructeur
     and p1.modele<p2.modele
     and p1.modele<p3.modele
     and p2.modele<p3.modele)
minus
(select distinct p1.constructeur
from (select modele,constructeur from produits where type='pc') p1,
     (select modele,constructeur from produits where type='pc') p2,
     (select modele,constructeur from produits where type='pc') p3,
     (select modele,constructeur from produits where type='pc') p4
where p1.constructeur=p2.constructeur
     and p2.constructeur=p3.constructeur
     and p3.constructeur=p4.constructeur
     and p1.modele<p2.modele
     and p1.modele<p3.modele
     and p1.modele<p4.modele
     and p2.modele<p3.modele
     and p2.modele<p4.modele
     and p3.modele<p4.modele);

```

## 3 Réorganisation, schémas et tables

### 3.1 Réorganisation

Réorganiser le travail à faire selon les fichiers suivants :

- **creation.sql** contenant la définition du schéma de la base de données relationnelle en prenant en compte ses contraintes. Afin de pouvoir travailler en groupe lors de ce TP et échanger les données avec les membres de votre groupe, respecter les noms des relations et attributs de l'énoncé,
- **insertion.sql** contenant les  $n$ -uplets,
- **affichage.sql** permettant d'afficher le contenu des tables créées,
- **requete.sql** pour les requêtes qui auront du Ãatre testÃ©es une Ã une,

- `nettoyage.sql` dont l'exécution permet de détruire toutes les tables que vous avez créés.

## 3.2 Connection

Connectez vous à Oracle depuis deux fenêtres de terminal distinctes (on les appellera F1 et F2).

- 1) Exécuter dans F1 et dans F2, votre fichier `nettoyage.sql` pour détruire toutes vos tables, et vérifier sur F1 et sur F2 que vos tables sont bien détruites.
- 2) Exécuter seulement dans F1, vos fichiers `creation.sql` et `insertion.sql` pour recréer le schéma et l'instance de la base du TP précédent. Vérifier que le schéma et l'instance créés correspondent aux schéma et instance souhaités. Vérifier que la base est aussi lisible à partir de la fenêtre F2. Qu'en déduisez-vous ?
- 3) Dans la fenêtre F1 insérer en ligne de commande dans une de vos tables un nouveau  $n$ -uplet  $t$  sans faire `commit`, et regarder si ce  $n$ -uplet se trouve bien dans la table. Vérifier dans la fenêtre F2 si le  $n$ -uplet  $t$  se trouve dans la table. Qu'en déduisez-vous ? Faut-il faire `commit` dans la fenêtre F1 ?

## 3.3 Schéma

Un schéma SQL est un ensemble de tables. Le nom du schéma est le nom de login du créateur de la table. Le créateur a tous les privilèges sur ses tables, et il peut céder un ou plusieurs de ces privilèges à un autre utilisateur. La personne qui a un privilège sur une de vos tables doit préfixer votre table par votre nom lors de son utilisation, ou bien créer un synonyme de votre table auparavant (voir ANNEXE).

- 1) Former un groupe de 2 à 4 utilisateurs (avec vos voisins) et donner différentes sortes de privilèges (1 par 1) sur vos tables à un autre membre de votre groupe. Tester ensemble des commandes SQL sur les tables qui ne sont pas les vôtres.
- 2) Annuler tous les privilèges que vous avez cédé aux membres de votre groupe. Céder à un de ces membres seulement le privilège `insert` pour une table. Refaire l'exercice 1, en supposant que maintenant vous êtes F1 et l'autre membre est F2. Qu'en déduisez-vous ?

**Remarque :** Cet exercice n'a de sens que si vous avez bien respecté les exigences du TP précédent et que le schéma de vos tables sont tout à fait identique.

- 3) Annuler tous les droits que vous avez cédé aux membres de votre groupe. Céder à tous les membres de votre groupes seulement le privilège `select` pour toutes les tables de votre base. Maintenant vous avez une base de données commune dont les données sont réparties sur différents site, où un membre est vu comme un site.

Reformuler les requêtes 1, 2 et 3 du TP précédent sur cette "base répartie".

**Remarque :** Cet exercice n'a de sens que si vous avez bien respecté les exigences du TP précédent et que le schéma de vos bases sont tout à fait identique.

## 3.4 Tables/Vues systèmes

Oracle gère les informations concernant ses différents utilisateurs, ainsi que ses propres informations, sous forme d'une base de données. Les caractéristiques de certaines de ces tables sont données en Annexe.

- Regarder la description de ces tables et consulter leurs contenus.
- Répondez aux questions suivantes en utilisant ces tables.
  1. Donnez le nom des tables que vous avez créées.
  2. Donnez le nom et le type des colonnes que vous avez créées ainsi que le nom de la table auxquelles elles appartiennent.
  3. Donnez le nom des tables auxquelles vous avez accès et leur propriétaire.
  4. Donnez le nom et le propriétaire de toutes les tables auxquelles vous avez accès mais que vous n'avez pas créées.
  5. En dehors des tables systèmes, quelles sont les tables sur lesquelles vous avez des privilèges.
  6. Parcourez la table `DICTIONARY` (qui a pour synonyme `DICT`), et pour les tables dont vous ne voyez pas l'utilité, lire le commentaire associé avec `help` (si cette commande est accessible).

**Annexe : DATE** Le type de données **DATE** correspond aux informations calendaires et horaires.

Bien que ces dernières puissent être représentée par une chaîne de caractères ou par des entiers, le type de données **DATE** a des propriétés particulières associées. Pour chaque valeur de ce type, oracle enregistre les champs suivants : siècle, année, jour, heure, minute et seconde.

Une valeur peut-être donnée comme un littéral (au format 'YYYY-MM-DD') ou convertie d'une chaîne de caractères ou d'un entier avec la fonction **TO\_DATE** :

```
DATE '2005-03-22'  
TO_DATE('05-MAR-22:14:30', 'YY-MON-DD:HH24:MI')
```

Le format par défaut des littéraux sous Oracle est défini par le paramètre **NLS\_DATE\_FORMAT**. Dans l'exemple ci-dessus, le format indique les deux derniers chiffres de l'année, une abréviation pour le mois, le jour du mois sur deux chiffres, l'heure sur 24 heures et les minutes.

- 1) Oracle convertit les valeurs des chaînes de caractères au format par défaut en valeur de type **DATE** quand elles sont utilisées dans les expressions de type **DATE**.

Lorsque l'on spécifie une date sans composante horaire la valeur par défaut est minuit. Lorsque l'on spécifie une date sans composante de jour, la valeur par défaut est le premier jour du mois en cours.

Une colonne de type **DATE** contient toujours les composantes calendaires et horaires. Si une requête utilise une date sans sa composante horaire alors cette composante est traitée comme ayant la valeur par défaut. Il faut toujours s'en préoccuper afin que les requêtes exprimées sur des dates renvoient bien ce qui est demandé.



## Annexe : ALTER TABLE

```
ALTER TABLE nom_de_table une_alteration ;
une_alteration ::=
    ADD nom_de_colonne domaine [DEFAULT expression] [ contrainte_de_colonne ] |
    ADD contrainte_de_table |
    DROP COLUMN nom_de_colonne [ RESTRICT | CASCADE ] |
    MODIFY nom_de_colonne [domaine] [DEFAULT expression] [contrainte_de_colonne] |
    RENAME COLUMN nom_de_colonne TO nouvelle_colonne |
    RENAME TO nouveau_nom |
    DROP CONSTRAINT nom_de_contrainte [ RESTRICT | CASCADE ] |
```

**ALTER TABLE** change la définition d'une table existante. Il y a plusieurs variantes :

- **ADD nom\_de\_colonne domaine** et **ADD (liste\_de\_colonnes)** ajoutent une nouvelle colonne à la table en utilisant la même syntaxe que **CREATE TABLE**. Une nouvelle colonne ne peut avoir la contrainte **NOT NULL** que si la table est vide.
- **ADD contrainte\_de\_table** ajoute une nouvelle contrainte à une table en utilisant la même syntaxe que **CREATE TABLE**.
- **DROP COLUMN nom\_de\_colonne** supprime une colonne d'une table. Les indexes et les contraintes de table référençant cette colonne sont automatiquement supprimés. Il faut utiliser l'option **CASCADE** si certains objets hors de la table dépendent de cette colonne, comme par exemple des références de clés étrangères ou des vues.
- **DEFAULT expression** ne modifient pas les lignes déjà présentes dans la table. Les valeurs par défaut ne s'appliquent qu'aux prochaines commandes **INSERT**. Des valeurs par défaut peuvent aussi être créées pour les vues.
- **RENAME TO** change le nom d'une table (ou d'un index, d'une séquence, d'une vue) ou le nom d'une colonne de la table. Elle est sans effet sur les données stockées.
- **DROP CONSTRAINT** supprime des contraintes d'une table. Actuellement, les contraintes n'ont pas nécessairement un nom unique, si bien qu'il peut y avoir plusieurs contraintes qui ont le nom donné en paramètre. Toutes ces contraintes sont supprimées.

**ALTER TABLE** effectue une copie temporaire de la table originale. Les modifications sont faites sur cette copie, puis l'original est effacée, et enfin la copie est renommée pour remplacer l'originale. Cette méthode permet de rediriger toutes les commandes automatiquement vers la nouvelle table sans pertes. Durant l'exécution de **ALTER TABLE**, la table originale est lisible par d'autres clients. Les modifications et insertions sont reportées jusqu'à ce que la nouvelle table soit prête.

## Annexe : SELECT

- `SELECT [ALL|DISTINCT] colonne | (fct(colonne) [AS nom]) [, liste_col_ou_fct(col)]`  
Liste de colonne ou fonctions sur une colonne (`AVG|MAX|MIN|SUM [ALL|DISTINCT]`) ou sur le nombre de ligne d'une table (`COUNT [ALL|DISTINCT]`). S'il y a des regroupements de colonnes `GROUP BY` alors les fonctions se font sur chaque regroupement (`AVG, MAX, MIN`) ou sur le nombre de ligne de chaque regroupement (`COUNT`).

- `FROM nom_de_table [alias] [,liste_de_nom_de_table]`

Indique les tables à partir desquelles récupérer les données. Un alias permet un renommage local. On utilise l'alias à la place du nom de la table afin d'enlever des ambiguïtés, de simplifier l'écriture ou d'être plus explicite.

- `[WHERE condition]`

Exemples d'opérateurs utilisables dans une condition :

- opérateurs logiques (`AND, OR, NOT, IS NULL, IS NOT NULL`)
- opérateurs de chaînes (`[NOT] IN|BETWEEN|LIKE`)
- opérateurs arithmétiques (`+, -, *, /, mod(,)`)
- comparateurs arithmétiques (`<, >, =, <>, <=, >=, [NOT] IN`)

Conditions avec sous-requêtes :

`WHERE [NOT] EXISTS|IN (clause_select)`

`WHERE colonne | (fct(colonne) [AS nom]) op [ALL|ANY] (clause_select)`

avec

**ALL** : la condition est vraie seulement si elle est vraie pour toutes les valeurs de la clause select.

Donc la condition est vraie si la sous-requête est vide. Donc : `<ALL` = moins que le minimum, `>ALL` = plus que le maximum, `<>ALL` équivalent à `NOT IN`.

**ANY** : la condition est vraie seulement si elle est vraie pour au moins l'une des valeurs de la clause select. Donc : `<ANY` = moins que le maximum, `>ANY` = plus que le minimum, `=ANY` équivalent à `IN`.

- `GROUP BY colonne [,list_col]`

Regroupe des résultats selon une(des) colonne(s).

- `HAVING condition`

est à `GROUP BY` ce que `WHERE` est à `FROM`.

- `[ORDER BY colonne [ASC|DESC] [,list_col]]`

Tri des résultats selon une (des) colonne(s). Toutes colonnes du select qui n'est pas sous une fonction doit être spécifiée dans la liste des colonnes de `ORDER BY` si on souhaite effectuer un tri.

Clauses obtenues par opération ensembliste :

`(clause_select) UNION [ALL] (clause_select)`

`(clause_select) INTERSECT [ALL] (clause_select)`

`(clause_select) MINUS [ALL] (clause_select)`

Sans l'option `ALL` seuls les *n*-uplets distinct sont conservés (à tester : support de `ALL` sous Oracle 10i pour les deux dernières opérations). Rappel : il faut respecter les types (i.e. les domaines en algèbre relationnelle) colonne à colonne.

## Annexe : GRANT

SQL> GRANT <privilège> ON <objet> TO <utilisateur_x>	donne un privilège à un utilisateur_x sur l'objet mentionné parmi les privilèges sont : insert, select, update, delete, alter
SQL> GRANT ALL ON <obj> TO <x>	donne tous les privilèges à x
SQL> GRANT <priv> ON <obj> TO PUBLIC	donne droit à tout le monde
SQL> CREATE SYNONYM <tab1> FOR <u>.<tab>	crée un synonyme de nom tab1 pour la table tab créée par l'utilisateur u.
SQL> REVOKE <privilège> ON <objet> FROM <utilisateur_x>	révocation des droits de manière identique au GRANT

## Quelques tables de système :

- Table contenant la description des tables (colonnes) auxquelles vous avez accès :  
ALL\_TABLES (owner, table\_name,...)  
ALL\_TAB\_COLUMNS (owner, table\_name, column\_name, data\_type, nullable,...)
- Table contenant la description des tables, des vues, des contraintes et des colonnes que vous avez créées :  
USER\_TABLES (table\_name,...)  
USER\_TAB\_COLUMNS (table\_name, column\_name, data\_type, nullable,...)  
USER\_CONSTRAINTS(...)  
USER\_CONSTRAINTS\_COLUMNS(...)      USER\_VIEWS\_COLUMNS(...)
- Table contenant tous les utilisateurs de la base de données :  
ALL\_USERS (user\_name, user\_id, created)
- Table contenant toutes les tables (colonnes) sur lesquelles vous avez des privilèges :  
TABLE\_PRIVILEGES (grantee, owner, table\_name, grantor, select\_priv, insert\_priv,  
delete\_priv, update\_priv, alter\_priv, created,...)  
COLUMN\_PRIVILEGES (grantee, owner, table\_name, column\_name, grantor,  
update\_priv, created,...)
- Table contenant toutes les tables (colonnes) systèmes :  
DICTIONARY (table\_name, comments)  
DICT\_COLUMNS (table\_name, column\_name, comments)