Dans ce tp on va simuler tous les exercices étudiés et TD1 concernant la BD gestion_pizza dont le model relationnel :

- `client (`**`NumClient`**`, NomClient, Adresse, Compte, PointegersRaPIzz );`
- `pizza (`**`NomPizza`**`, Prix);`
- `livreur (`**`CodeLivreur`**`, NomLivreur, Telephone);`
- `vehicule (`**`NumImmat`**`, Marque, Type_vehicule );`
- `ingredient (`**`NumIngre`**`, NomIngredient);`
- `tarification (`**`Taille`**`, `**`Coefficient`**`);`
- `commande (`**`NumCommande`**`, DateCom, Taille, Retard, CodeLivreur, NumImmat, NumClient, NomPizza);`
- `compose( `**`NumIngre`**`, `**`NomPizza`**`);`

Question 1. Implémenter cette BD**.** Le script de création bd_pizza.sql est disponible à l'url : `http://nakech.free.fr/dba/`

Question 2. (voir correction en TD1)

A/ Créer les comptes des utilisateurs (clients $\{c_1.. c_{100}\}$, les employés sont $\{e_1.. e_{10}\}$ et le gestionnaire sont $\{g_1.. g_5\}$)

B/ Réaliser les requêtes SQL permettant de réaliser les droits d'accès. Je rappel qu'ici on doit utiliser la notion de role.

C/ Vérifier en consultant les tables systèmes : DBA_users, DBA_roles, et DBA_TAB_PRIVS

Question 3.

Créer les vues des utilisateurs aux besoins plus spécifiques vont utiliser la BD pizza :

- Le service de comptabilité {Michel, Laurent } doit rémunérer les livreurs, il a besoin le nombre de commande livrée.
- Le service du personnel { Thiry, Yves, Armelle } effectue un suivi des livreurs, il a besoin le nombre de retard de chacun et également le nombre de commande livrés.
- Un laboratoire de recherche { Jacques, Micheline } en sociologie effectue des études sur la clientèle, (relation entre les ingrédients et l'adresse des clients).

Indications :

Service comptabilité : Le besoin est uniquement d'avoir accès aux nombres de commandes livrées pour chaque livreur. Il s'agit du résultat d'un calcul classique sur un agrégat. On affiche uniquement les informations du livreur (code et nom) et le résultat du calcul.

Indication :

```
CREATE VIEW commande_livreur AS
SELECT L.CodeLivreur, L.NomLivreur, COUNT(*) AS Nombre_Livraison
FROM commande C , livreur L where C.CodeLivreur=L.CodeEivreur
GROUP BY C.CodeLivreur, L.CodeLivreur
ORDER BY L.NomLivreur;
CREATE ROLE COMPTABILITE
GRANT SELECT ON commande_livreur TO  COMPTABILITE
GRANT COMPTABILITE Michel, Laurent;
```

Service du personnel.

- La vue qui permet de calculer le nombre de retards cumulés par livreur.

Indication :

```
CREATE VIEW retard_livreur AS
SELECT L.CodeLivreur, L.NomLivreur, COUNT(*) AS Nombre_Retard
FROM commande C, livreur L WHERE C.CodeLivreur=L.CodeEivreur
AND C.Retard='O'
GROUP BY C.CodeLivreur , L.NomLivreur
ORDER BY L.NomLivreur;
```

Indication :

```
CREATE ROLE personnel.
GRANT SELECT ON retard_livreur TO personnel;
```

- Le service du personnel a besoin également d'utiliser la précédente (`commande_livreur`)

Indication :

```
GRANT SELECT ON commande_livreur TO personnel;
GRANT  personnel to Thiry, Yves, Armelle ;
```

Laboratoire de sociologie. Le laboratoire de recherche doit pouvoir accéder au contenu de deux champs sans calcul ni mise en forme. En revanche, on ne donne pas la possibilité à ces utilisateurs extérieurs de connaître la structure interne des données et d'avoir accès à d'autres champs. On crée une vue qui est le résultat de la jointure entre les tables 'commande', 'client, 'Pizza, 'compose, 'ingredient'.

Indication :

```
CREATE VIEW adresse_ingredient AS
SELECT CL.Adresse, I.NomIngre
FROM commande C , client CL , pizza P , compose CO , ingredient I
where C.NumClient = CL.NumClient AND C.NomPizza = P.NomPizza
AND P.NomPizza = CO.NomPizza AND CO.NumIngre = I.NumIngre


CREATE ROLE sociologie ;
GRANT SELECT ON adresse_ingredient to sociologie ;
GRANT sociologie to Jacques, Micheline ;
```

1. Ne pas oublier la Création des utilisateurs : Michel, Laurent, Thiry, Yves, Armelle, Jacques, et Micheline.
2. Vérifier en consultant les tables systèmes : USER_TAB_PRIVS et DBA_TAB_PRIVS
3. Faire quelques requêtes sql testant les droits d'accès précédents.

Question 4.

On désir effectuer une simulation de mise à jour de la BD. l'idée est de procéder à une augmentation générale de 10 % du prix des pizzas tout en évitant de faire fuir les clients. On essaie de modifier les différents coefficients affectés aux différentes tailles de pizzas (naine, humaine, grosse) et sur les prix de base. L'objectif est de mesurer les effets de ces modifications sur le compte des clients. On modifie dans un premier temps les prix ; ensuite, les coefficients en se donnant la possibilité de revenir en arrière pour tester différentes combinaisons de coefficients et de prix. Une fois les essais terminés, on abandonne toutes les modifications : ce n'était qu'une simulation.

Réaliser la simulation précédente. On utilise évidemment le mécanisme des transactions en définissant des points de retour `savepoint` (voir ch2 page 4) pour pouvoir annuler les effets des modifications des coefficients et des prix. On abandonne toutes les modifications, y compris celle des prix à la fin de transaction.

Indication

```
# Démarrage de la transaction
# Définition d'un point de retour que l'on nomme 'PRIX'

SAVEPOINT 'PRIX' ;

# Augmentation des prix des pizzas de 10 9.1

UPDATE pizza SET Prix=Prix*l.l;
```

On a défini ici un point de retour 'PRIX' qui permet de revenir à l'état de la base avant la modification de prix.

```
# Définition d'un point de retour que l'on nomme 'COEFF_NAINE'
SAVEPOINT 'COEFF_NAINE' ;

# Modification des coefficients de la taille 'naine' dans la table tarification

UPDATE tarification SET Coefficient=0.5 WHERE Taille='naine'

# Définition d'un point de retour que l'on nomme 'COEFF_OGRE'
```

```
SAVEPOINT 'COEFF_OGRE' ;

# Modification des coefficients de la taille 'ogresse' dans la table tarification

UPDATE tarification SET Coefficient=1.5 WHERE Taille='ogresse' ;
```

On a défini ici deux points de retour 'COEFF_NAINE', et 'COEFF_OGRE' qui permettent de revenir à l'état de la base avant les modifications des coefficients. Pour revenir à l'état de la base avant modification de tous les coefficients, on retourne au point 'COEFF_NAINE'

```
# Annulation des modifications faites depuis le point 'COEFF_NAINE'
ROLLBACK TO 'COEFF-NAINE';
```

La transaction ne se termine pas lorsque l'on fait un ROLLBACK vers un point de retour. On peut donc essayer d'autres coefficients, sans oublier de redéfinir de nouveaux points de retour.

```
# Définition d'un point de retour que l'on nomme 'COEFF-NAINE2'

SAVEPOINT 'COEFF-NAINE2' ;

# Modification des coefficients de la taille 'naine' dans la table tarification

UPDATE tarification SET Coefficient=0.5 WHERE Taille='naine' ;
```

Ainsi de suite. Puis on abandonne l'ensemble de modifications par :

```
ROLLBAK
```

qui termine la transaction et remet la base dans l'état de départ.

- Vérifier en affichant la table `tarification` Apres chaque `update` ou `rollbak ;`

Question 5.

On désire effectuer des mise à jour dans la base de donné pizza. Lorsque l'on détruit une pizza (table `pizza`), on veut que les entrées de la table `compose` correspondant à cette pizza disparaissent automatiquement. Ainsi, la table `compose` ne contiendra aucune entrée orpheline.

Indication :

```
CREATE TRIGER maj_compose BEFOR DELETE ON pizza FOR EACH ROW
BEGIN
DELETE FROM compose
Where NomPizza =  :OLD.NomPizza
END
```

1. Vérifier en consultant la table système DBA_trigger

2. Faire également quelques teste de suppression de lignes

Dossier à rendre : Pour chaque question toutes les requêtes commentées.
Le compte rendu est à envoyer au **nakech@free.fr** avec comme objet **tp4 _nom1_nom2**
Date limite :

Annexe bd_pizza.sql

```
create table client (
                            NumClient integer primary key,
                            NomClient varchar(30) not null,
                            Adresse varchar(50),
                            Compte varchar(10),
                            PointegersRaPIzz integer);

create table pizza (
                            NomPizza varchar(20) primary key,
                            Prix integer not null);

create table livreur (
                            CodeLivreur integer primary key,
                            NomLivreur varchar(30) not null,
                            Telephone integer);

create table vehicule (
                            NumImmat varchar(10) primary key,
                            Marque varchar(20),
                            Type_vehicule varchar(20));

create table ingredient (
                            NumIngre integer primary key,
                            NomIngredient varchar(20) not null);

create table tarification (
                            Taille char(30) primary key,
                            Coefficient float not null);

create table commande (
                            NumCommande integer primary key,
                            DateCom date,
                            Taille integer,
                            Retard integer,
                            CodeLivreur integer references livreur(CodeLivreur),
                            NumImmat varchar(10) references vehicule(NumImmat),
                            NumClient integer references client(NumClient),
                            NomPizza varchar(20) references pizza(NomPizza));
create table  compose(
                            NumIngre integer references ingredient(NumIngre),
                            NomPizza varchar(20) references pizza(NomPizza));


insert into client values(1,'toto','26 rue des sous-bois','123456',12);
insert into client values(2,'titi','27 rue des sous-bois','234567',1);
insert into client values(3,'tata','28 rue des sous-bois','345678',46);

insert into pizza values ('Royal',10);
insert into pizza values ('Calzone',11);
insert into pizza values ('Chorizo',9);

insert into livreur values (1,'Coco',0656327854);
insert into livreur values (2,'Cici',0656327855);
insert into livreur values (3,'Caca',0656327856);

insert into vehicule values ('7642FR76','SMART','SMART');
insert into vehicule values ('4612RF76','107','Peugeot');
insert into vehicule values ('6534GD76','AUSTINE','mini');

insert into ingredient values (1,'tomate');
insert into ingredient values (2,'champignon');
insert into ingredient values (3,'chorizo');
insert into ingredient values (4,'jambon');
insert into ingredient values (5,'mozzarella');

insert into tarification values ('naine',2/3);
insert into tarification values ('humaine',1);
insert into tarification values ('ogresse',4/3);

insert into commande values (1,null,2,0,1,'6534GD76',1,'Calzone');
insert into commande values (2,null,3,0,1,'6534GD76',3,'Chorizo');

insert into compose values (1,'Calzone');
insert into compose values (2,'Calzone');
insert into compose values (3,'Calzone');
insert into compose values (4,'Calzone');
insert into compose values (1,'Royal');
insert into compose values (2,'Royal');
insert into compose values (3,'Royal');
insert into compose values (4,'Royal');
insert into compose values (5,'Royal');
insert into compose values (3,'Chorizo');
insert into compose values (1,'Chorizo');
/
```