TD Transactions

1. Schéma de base de données

Nous allons travailler (en TD comme en TP) sur la base de données employés, dont le schéma relationnel vous est donné ci-dessous :

fonction(**nom_f varchar(15)**, salaire_min float, salaire_max float)
dep (**num_d number**, nom_d, adresse)
emp (**num_e number**, nom varchar(15), prenom varchar(15), fonction varchar(15), salaire float, commission float, date_embauche date, n_sup number, n_dep number)

Avec employe(n_sup) $\subseteq$ employe(num) et employe(fonction) $\subseteq$ fonction(nom_f) et employe(n_dep) $\subseteq$ departement(num_d)

2. Exercice 1 : nombre de transactions et modes d'isolation

2.1 Question 1

Une session utilisateur exécute séquentiellement les ordres suivants :

```
-- debut de session
select * from emp;
insert into dep values (6,'menuiserie','Montpellier');
update dep set adresse = 'Carnon' ;
alter table dep add budget float ;
insert into dep values (7,'ingenierie','Montpellier',30000);
rollback ;
insert into dep values (7,'vente','Paris',400000);
commit ;
```

— Donner le nombre de transactions définies dans cette séquence
— indiquer quel est le mode d'isolation exploité par défaut

2.2 Questions 2

2.2.1 Question 1.2.1

Une nouvelle séquence est donnée :

```
-- debut de session
set transaction read only ;
select * from fonction ;
```

```
alter table fonction add categorie varchar(1) ;
update fonction set categorie = 'A';
select nom_f from fonction where categorie = 'A';
commit ;
```

— Est ce que l'ordre ALTER (LDD) va aboutir à une erreur ? Justifier votre réponse.

2.2.2 Question 1.2.2

La même séquence ou presque est donnée à nouveau et la même question vous est posée. Il faut répondre dans l'esprit des usages car il n'est pas possible avec Oracle de disposer d'une session seulement en consultation.

```
-- debut de session
-- ordre correct mais non exploitable avec Oracle
alter session set transaction read only ;
select * from fonction ;
alter table fonction add categorie varchar(1) ;
update fonction set categorie = 'A';
select nom_f from fonction where categorie = 'A';
commit ;
```

— Dans l'absolu, est ce que l'ordre ALTER (LDD) devrait générer une erreur ? Justifier votre réponse.

2.2.3 Question 1.2.3

Une nouvelle séquence est donnée :

```
-- debut de session
set transaction read only name 'T1' ;
select nom_f from fonction where categorie = 'A';
exec dbms_lock.sleep(10)
select nom_f from fonction where categorie = 'A';
commit ;
```

Pendant la période d'endormissement, une transaction concurrente (session T2) a passé avec succès, les ordres suivants (sur la table fonction du même schéma utilisateur) :

```
-- debut de session T2
update fonction set categorie = 'C';
commit ;
```

— Vous donnerez les effets de la transaction T2 sur la table fonction
— Vous indiquerez ce que voit la transaction nommée T1 ? Justifier votre réponse.
— Vous indiquerez sur quel granule, la transaction nommée T2 a posé un verrou (levé ensuite par la validation) ? S'agit'il d'un verrou exclusif ? Justifier votre réponse.

2.2.4 Question 1.2.4

Quelle est la différence entre les deux ordres suivants ? Qu'apporte le niveau d'isolation indiqué ?

```
ALTER SESSION SET ISOLATION_LEVEL=SERIALIZABLE;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE ;
```

3. Exercice 2 : transactions concurrentes

3.0.1 Question 2.1

Vous expliquerez ce que voit la transaction 2, concernant les effets de la transaction 1

Transaction 1 (read committed)	Transaction 2 (read committed)
update user1.dep set nom_d='atelier design' where nom_d='bureau dessin' ;	——— ———
———	———
———	select * from user1.dep ;

Figure 1 – Exemple de verrou posé sans situation de blocage

3.0.2 Question 2.2

Vous expliquerez ce que voit la transaction 2, concernant les effets de la transaction 1

Transaction 1 (read committed)	Transaction 2 (read committed)
update user1.dep set nom_d='atelier design' where nom_d='bureau dessin' ;	——— ———
commit ;	———
———	select * from user1.dep ;

Figure 2 – Exemple de verrou posé sans situation de blocage

3.0.3 Question 2.3

Vous expliquerez ce que subit la transaction 2, concernant les effets de la transaction 1

Transaction 1 (read committed)	Transaction 2 (read committed)
update user1.dep set nom_d='atelier design' where nom_d='bureau dessin' ;	——— ———
———	———
——— ———	update user1.dep set adresse='Colmar' where nom_d='bureau dessin' ;

Figure 3 – Exemple de verrou posé avec situation de blocage

3.0.4 Question 2.4

Vous expliquerez ce que voit la transaction 2, concernant les effets de la transaction 1

Transaction 1 (read committed)	Transaction 2 (serializable)
update user1.dep set nom_d='atelier design' where nom_d='bureau dessin' ;	——— ———-
commit ;	———
———	select * from user1.dep ;

Figure 4 – Exemple de verrou posé sans situation de blocage

3.0.5 Question 2.5

Vous expliquerez ce que voit la transaction 2, concernant les effets de la transaction 1

Transaction 1 (read committed)	Transaction 2 (read only)
update user1.dep set nom_d='atelier design' where nom_d='bureau dessin' ;	——— ———-
commit ;	———
———	select * from user1.dep ;

Figure 5 – Exemple de verrou posé sans situation de blocage

3.0.6 Question 2.6

Vous expliquerez ce que subit la transaction 2, concernant les effets de la transaction 1

Transaction 1 (read committed)	Transaction 2 (serializable)
update user1.dep set nom_d='atelier design' where nom_d='bureau dessin' ;	——— ———-
———-	———
——— ———	update user1.dep set adresse='Colmar' where nom_d='bureau dessin' ;

Figure 6 – Exemple de verrou posé avec situation de blocage

4. Exercice 3 : reporter la vérification d'une contrainte

Les actions internes d'une une transaction peuvent conduire temporairement à des incohérences (uniquement le temps de la transaction). Ainsi, il est possible d'indiquer, pour une contrainte de clé étrangère, que sa vérification ne sera prise en charge qu'au moment de la fin d'une transaction, et non pas à chaque ordre SQL. Le script du TP va prendre en charge de telles définitions de contrainte.

```
alter table emp add constraint dept_fk
```

```
references dep(num_d) deferrable initially deferred
```

— Expliquer l'intérêt de surseoir la vérification d'une contrainte de clé étrangère.
— Donnez un exemple d'insertion ou de mise à jour pour les tables emp et dep

5. Exercice 4 : graphe de précédence et interblocage

Soient A, B, et C des granules (ici respectivement les tuples de numéro 90, 91 et 92 de la table emp) d'une base de données. Soient T1, T2, et T3, trois transactions qui font des accès aux granules. Une exécution des transactions est donnée (avec L qui désigne une opération de lecture, et E une opération d'écriture (de type LMD)).

```
T2L(B) T1E(B) T3E(A) T2L(C) T1E(A) T3E(B) T2L(A)
```

— Tracer le graphe de dépendances. Pour ce faire (les transactions étant les nœuds du graphe) :
  — Ti lit le granule O avant que Tj n'écrive sur le granule O => TRACER une transition : Ti précède Tj ;
  — Ti écrit sur le granule O avant que Tj n'écrive sur le granule O => TRACER une transition : => Ti précède Tj ;
— Détectez vous des situations de conflit, voire même d'interblocage ?