HMIN328 : Contrôle continu 22/11

1. Schéma de base de données

Le schéma relationnel demeure le même :

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. Transactions concurrentes

Vous travaillerez également toujours en binôme (ou seul avec deux sessions) afin d'exécuter des transactions définies au sein de deux sessions utilisateur différentes mais portant sur les mêmes objets. Plusieurs situations vous sont données. Vous expliquerez les résultats obtenus et les éventuels problèmes rencontrés pour chacune de ces situations (donnez votre compréhension) .

Transaction 1	Transaction 2
———	alter session set
———	isolation_level = serializable ;
———	select salaire from user1.emp
———	where fonction = 'president' ;
update user1.emp set salaire = salaire + 100	———
where fonction = 'president' ;	———
———	select salaire from user1.emp
———	where fonction = 'president' ;
select salaire from user1.emp	———
where fonction = 'president' ;	———

Figure 1 – Question 1

Transaction 1	Transaction 2
———	alter session set
———	isolation_level = read committed ;
———	select salaire from user1.emp
———	where fonction = 'president' ;
update user1.emp set salaire = salaire + 100	———
where fonction = 'president' ;	———
———	select salaire from user1.emp
———	where fonction = 'president' ;
select salaire from user1.emp	———
where fonction = 'president' ;	———

Figure 2 – Question 2

Transaction 1	Transaction 2
———	alter session set
———	isolation_level = serializable ;
———	select salaire from user1.emp
———	where fonction = 'president' ;
update user1.emp set salaire = salaire + 100	———
where fonction = 'president' ;	———
commit ;	———
———	select salaire from user1.emp
———	where fonction = 'president' ;
select salaire from user1.emp	———
where fonction = 'president' ;	———

Figure 3 – Question 3

Transaction 1	Transaction 2
———	alter session set
———	isolation_level = read committed ;
———	select salaire from user1.emp
———	where fonction = 'president' ;
update user1.emp set salaire = salaire + 100	———
where fonction = 'president' ;	———
commit ;	———
———	select salaire from user1.emp
———	where fonction = 'president' ;
select salaire from user1.emp	———
where fonction = 'president' ;	———

Figure 4 – Question 4

Transaction 1	Transaction 2
———	alter session set
———	isolation_level = serializable ;
———	select salaire from user1.emp
———	where fonction = 'president' ;
update user1.emp set salaire = salaire + 100	———
where fonction = 'president' ;	———
commit ;	———
———	update user1.emp set salaire = salaire + 300
———	where fonction = 'president' ;

Figure 5 – Question 5

Transaction 1	Transaction 2
———	alter session set
———	isolation_level = read committed ;
———	select salaire from user1.emp
———	where fonction = 'president' ;
update user1.emp set salaire = salaire + 100	———
where fonction = 'president' ;	———
———	update user1.emp set salaire = salaire + 300
———	where fonction = 'president' ;
exec dbms_lock.sleep(30) ;	———
rollback ;	———

Figure 6 – Question 6

Transaction 1	Transaction 2
———	alter session set
———	isolation_level = serializable ;
———	select salaire from user1.emp
———	where fonction = 'president' ;
update user1.emp set salaire = salaire + 100	———
where fonction = 'president' ;	———
commit ;	———
———	update user1.emp set salaire = salaire + 300
———	where fonction = 'ingenieur' ;

Figure 7 – Question 7

Transaction 1	Transaction 2
———	alter session set
———	isolation_level = read committed ;
———	select salaire from user1.emp
———	where fonction = 'president' ;
select * from user1.emp	———
for update ;	———
———	update user1.emp set salaire = salaire + 300
———	where fonction = 'president' ;
rollback ;	———

Figure 8 – Question 8

3. Vues sur les transactions en cours

Vous mettrez une de vos sessions en situation de blocage et vous testerez les requêtes qui vous sont données ci-dessous. Vous en donnerez les significations et principaux intérêts.

3.1 Requête 1

```
select s1.sid, s1.username, s1.osuser, s2.sid, s2.username, s1.osuser
from v$session s1, v$session s2 where s2.sid = s1.blocking_session;
```

3.2 Requête 2

```
select s.username ,s.sid,w.seconds_in_wait as n_seconds
from  v$session s, v$session_wait w where s.sid = w.sid and type ='USER';
```

3.3 Requête 3

```
select sid, username, osuser,  object_name, object_type
from v$session, dba_objects where row_wait_obj# = object_id
```

```
and blocking_session is not null;
```

3.4 Requête 4

```
select a.SID, b.SID, b.request
from v$lock a, v$lock b
  where a.SID != b.SID and a.ID1 = b.ID1
  and a.ID2 = b.ID2 and b.request > 0
  and a.block = 1;
```

3.5 Requête 5

Construisez une requête (exploitant des vues parmi lesquelles v$session, dba_objects, v$lock, dba_blockers, v$locked_object ou encore v$session_wait) qui vous semble apporter une information pertinente concernant les transactions en cours.

4. Performances

Vous donnerez en SQL et de deux manières différentes, la requête : "donnez les départements (num_d) dans lesquels toutes les fonctions sont exercées (dans lesquels travaillent des employés qui exercent toutes les fonctions référencées dans fonction)". Vous définirez les plans d'exécution et indiquerez celui qui vous semble le moins coûteux.