TP Transactions et vues dynamiques

1. Schéma de base de données

Nous allons travailler 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)
Un script de création des tables est également donné. Avant de créer votre nouveau schéma, veillez à supprimer tous les objects de votre schéma utilisateur. Vous noterez que dans le nouveau script, les vérifications concernant les contraintes d'intégrité référentielle sont repoussées au moment de la validation des transactions.

2. Objectifs

Il s'agit, dans ce TP, d'explorer différents ordres SQL et vues dynamiques orientées vers de la manipulation des mécanismes transactionnels et de la supervision de bases de données Oracle. Vous travaillerez de ce fait en binôme afin d'exécuter des transactions définies au sein de deux sessions utilisateur différentes mais portant sur les mêmes objets.

2.1 Vérifier vos connaissances au sujet du modèle transactionnel

Un fichier, portant une extension .sql, vous est fourni, vous testerez les différents blocs d'ordres contenus et vous répondrez aux questions associées à ces tests (en justifiant vos réponses). Une partie des tests sont à faire à deux (deux sessions utilisateurs sont nécessaires pour tester les effets des transactions).

2.2 Rappel : afficher les informations concernant les sessions utilisateurs en cours

Une requête SQL qui consulte le contenu de la vue dynamique v$session vous est proposée.

```
select
 username, osuser, process, terminal,
 to_char(logon_time, 'YYYY-MM-DD HH24:MI') as logon_time
from v$session where type='USER';
```

La vue v$process (jointure avec v$session sur l'attribut paddr) restitue des informations, concernant les applications clientes, exploitées par les usagers, pour interagir avec la base de données.

Vous pouvez poursuivre l'exemple, en définissant une requête qui restituera le nom des usagers connectés, la date d'ouverture de leur session ainsi que des informations sur leurs applications clientes. Vous noterez qu'une session est identifiée par un identifiant de session, nommé *sid*.

2.3 Vues dynamiques et mécanismes transactionnels

Des vues sont plus, précisément, dédiées à rendre compte des transactions en cours d'exécution, des verrous posés par ces transactions et des blocages éventuels relatant de conflits d'accès à des ressources de la base de données, émanant de transactions concurrentes.
Ces vues vont permettre de donner des réponses à des questions telles que :

- quelles sont les transactions qui ont posé des verrous sur la base de données ?
- quelles sont les ressources verrouillées ?
- quelle transaction est en attente de libération de verrou ?
- quelle transaction est bloquante pour quelle autre transaction ? Si une transaction est bloquante, quels sont les ordres SQL qui ont provoqué d'une part le blocage, et d'autre part la situation d'attente ?
- ...

Pour pouvoir tester l'intérêt de ces vues, vous vous mettrez, par binôme, dans des situations de blocage.

2.3.1 Exemple de la vue v$lock

La vue v$lock est une vue, particulièrement informative, pour tout ce qui concerne les verrous posés sur la base.

```
select * from v$lock;
```

Il est à noter que l'attribut *block* prend la valeur 1, lorsque la session détient un verrou qui est en train de bloquer une autre transaction (et 0 dans le cas contraire). L'attribut *lmode* indique le type de verrou détenu par la transaction et son domaine de valeurs est détaillé plus bas. L'attribut *lmode* peut aussi indiquer (pour la transaction en attente) le type de verrou demandé, mais non encore obtenu.

- 0 : None,
- 1 : Null : Select
- 2 : Row Share : les accès concurrents peuvent exploiter la ressource, mais aucun verrou exclusif ne peut être posé sur la table (dans son intégralité) (posé de manière automatique avec un ordre select ... for update)
- 3 : Row Exclusive : posé de manière automatique avec tout ordre *update, delete et insert*
- 4 : Share : acquis avec l'ordre LOCK TABLE <table> IN SHARE MODE ; les modifications pouvant être apportées par les autres transactions ne sont pas admises.
- 5 : Share Row Exclusive : plus restrictif que Share, ce mode ne permet pas aux transactions de poser un verrou en mode Share sur la table considérée.
- 6 : Exclusive : une seule transaction est autorisée à poser un verrou exclusif sur une table, les autres transactions doivent alors se contenter de simples lectures de la table.

L'attribut *type* permet aussi de caractériser le verrou mobilisé, et a pour domaine de valeurs (non exhaustif) :

1. TM : DML enqueue (verrou associé à un ordre DML sur la table)
2. TX : Transaction (verrou au niveau du tuple)
3. TD : DDL enqueue (verrou associé à un ordre DDL sur la table)
4. TS : Temporary Segment

5. AE : Edition enqueue
6. ...

Vous exploiterez v$lock de manière à faire ressortir les transactions qui se révèlent bloquantes pour d'autres. Vous testerez à titre d'exemple, diverses situations explicitées dans les tableaux ci-dessous.

Transaction 1	Transaction 2
update user1.dep set nom_d='atelier design'	———
where nom_d='bureau dessin'	———
———	———
———	———

Figure 1 – Exemple de verrou posé sans situation de blocage

Transaction 1	Transaction 2
update user1.dep set nom_d='atelier design'	———
where nom_d='bureau dessin'	———
———	update user1.dep set nom_d='bureau design'
———	where nom_d='bureau dessin'

Figure 2 – Exemple de conflit bloquant

Transaction 1	Transaction 2
select * from user1.dep for update	———
———	update user1.dep set nom_d='bureau design'
———	where nom_d='bureau dessin'

Figure 3 – Autre exemple de conflit bloquant

Une requête exploitant une utilisation différentielle de la vue v$lock vous est fournie (en fin de document), vous la testerez et vous l'exploiterez de concert avec v$session pour faire apparaître le nom des usagers à l'origine des sessions concernées.

2.3.2 Autres vues d'intérêt

Vous pouvez, également, regarder de plus près, le schéma des vues dba_blockers, dba_objects, v$locked_object ou encore v$session_wait (au travers de la commande sqlplus desc) et à les exploiter au travers de simples requêtes. Différentes requêtes portant sur ces vues sont listées ci-dessous. Vous en proposerez des exploitations et vous en testerez les fonctionnalités, en provoquant délibéremment des situations de blocage. Vous considérerez aussi la vue v$transaction.
Différents ordres de consultation portant sur ces vues, vous sont donnés. Vous pouvez les adapter en fonction de vos aspirations.

2.3.3 Qui bloque quoi ?

Le nom de l'utilisateur Oracle (doublé de son identifiant en terme de système d'exploitation) et le nom et le type de l'objet sur lequel cet utilisateur a posé des verrous, sont retournés.

```
col object_name for a20
col object_type for a10
col oracle_username for a10
col os_user_name for a20

select  oracle_username, os_user_name, locked_mode,
object_name,  object_type from  v$locked_object a,dba_objects b
where  a.object_id = b.object_id
```

2.3.4 Qui se trouve en situtation d'attente ?

Le nom de l'usager qui se trouve en attente d'une ressource, son identifiant de session et son temps d'attente sont retournés.

```
col username for a12

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 ;
```

2.3.5 Qui a posé des verrous bloquants ?

Des exemples qui permettent de renseigner les sessions (et transactions) qui ont posé des verrous qui peuvent s'avérer éventuellement bloquants pour d'autres sessions.

```
select sid,username from V$SESSION where sid in
(select sid from V$LOCK where block=1);
ou
select username, v$lock.sid, lmode, request, block, v$lock.type
from v$lock, v$session where v$lock.sid = v$session.sid
```

2.3.6 Qui bloque qui ?

Les colonnes id1 et id2 de la table v$lock permettent de mettre en correspondance les couples session bloquante - session bloquée (qui partagent ces mêmes identifiants, corrélés au segment d'annulation exploité de manière sous-jacente). La session bloquée est dans l'impossibilité de voir sa requête exécutée, et de sorte sa requête (colonne request) a une valeur supérieure à 0 (numéro correspondant au type de verrou demandé).

```
select a.SID as blockingSession, b.SID as blockedSession, 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;
```

2.3.7 Procédure PL/SQL

Vous exploiterez de façon appropriée les différentes vues explorées (v$lock, v$session_wait, dba_blockers, ...) dans une procédure PL/SQL qui permet pour une session en train de passer une transaction qui pose différents verrous en écriture (mises à jour de tuples), de lister les autres sessions concurrentes possiblement bloquées.

2.3.8 Modes d'isolation

Quels sont les modes d'isolation à même de prévenir les lectures non répétables

1. quand la transaction ne fait que des opérations en lecture sur les données ?
2. quand la transaction fait aussi des accès en écriture sur les données ?