Base de données Avancée (BDA), Enseignants : Nakechbandi M.

Thème : Architecture centralisée et Accès concurrentiel, Transactions

I. Présentation :

Soit la base de données CIRQUE(*) dont l'architecture est centralisée.

PERSONNEL

NOM(40)	ROLE(20)
Clovis	**Jongleur**
Reine	**Ecuyer**
Louche	**Clown**
Benard	**Equilibriste**
Bignon	**Musicien**
Bordeau	**Dompteur**
Jerry	**Clown**
Demare	**Jongleur**
Fremez	Musicien
Dolores	Jongleur
Hebert	Jongleur
Sorel	Ecuyer
Loisel	Equilibriste
Jeanne	Jongleur
Sangtrespur	Dompteur
	…

NUMERO

TITRE(30)	NATURE(20)	RESPONSABLE(20)
Les Zoupalas	**Jonglerie**	**Clovis**
Le coche infernal	**Equitation**	**Reine**
Les fauves	**Clownerie**	**Louche**
Les Smilers	**Equilibre**	**Benard**
La passoire magique	Lion	Bordeau
Les Zozos	Clownerie	Jerry
Les Tartarins	Jonglerie	Demare
		…

ACCESSOIRE

NOM(30)	COULEUR(10)	VOLUME	RATELIER	CAMION
Ballon	**Rouge**	**0.3**	**15**	**5**
Barre	**Blanc**	**0.6**	**19**	**5**
Fouet	**Marron**	**0.2**	**11**	**3**
Bicyclette a elephant	**Vert**	**0.4**	**27**	**8**
Trompette	**Rouge**	**0.2**	**2**	**1**
Cercle	Magique Orange	0.2	1	1
Boule	Cristal	0.2	88	8
Cage a lions	Noir	10.0	0	2
Chaise longue de lion	Bleu	0.9	11	5
Peigne de chimpanze	Jaune	0.2	23	3
…				

UTILISATION

TITRE(30)	UTILISATEUR(20)	ACCESSOIRE(30)
Les Zoupalas	**Dolores**	**Ballon**
Les Zoupalas	**Hebert**	**Ballon**
Les Zoupalas	**Dolores**	**Barre**
Le coche infernal	**Jeanne**	**Bicyclette a elephant**
Le coche infernal	**Sorel**	**Fouet**
Les fauves	**Jerry**	**Trompette**
Les Smilers	**Benard**	**Cercle magique**
Les Smilers	**Benard**	**Boule**
Les Smilers	Loisel	Bicyclette a elephant
La passoire magique	Bordeau	Cage a lions
La passoire magique	Bordeau	Chaise longue de lion
Les Zozos	Jerry	Bicyclette a elephant
Les Zozos	Jerry	Peigne de chimpanze
Les Tartarins	Jeanne	Bicyclette a elephant
Le coche infernal	Sangtrespur	Etrier
		…

II. Travail à faire

Pour faire ce TP commencer par créer un utilisateur oracle correspondant à votre prénom :

```
create user CLI1 identified by mot_passe
```

Pour donner à Arnaud les droits de connection et de pouvoir créer des tables :

```
grant connect, ressource to CLI1;
```

(*) S'inspirer de l'ouvrage de E. Pichat, R. Bodin, "Ingénierie des données", Masson, 1990

Idem pour le binome CLI2

A. Saisir de données et copier les données des autres utilisateur

1. Dans la suite vous serai CLI1 votre binome serai CLI2. Vous allez travailler par groupe de deux utilisateurs oracle (CLI1, CLI2) sur même ordinateur en ouvrant deux connections différentes (deux fenêtres sql).
2. Créer la structure des quatre tables de la base en intégrant les contraintes d'intégrité nécessaires (travail à faire dans chaque session (fenêtre) sql)

Indication : Le scripte suivant est disponible sur le URL `http://nakech.free.fr/BDA/creer_cirque.sql`

```
create table personnel (nom varchar(40),
                        role varchar(20),
                        constraint personnel_pri primary key(nom));

create table numero (titre varchar(30),
                     nature varchar(20),
                     responsable varchar(40),
                     constraint numero_pri primary key(titre),
                     constraint numero_etr foreign key(responsable) references
                     personnel(nom));

create table accessoire (nom varchar(30),
                         couleur varchar(10),
                         volume number(4,1),
                         ratelier number(2),
                         camion number(1),
                         constraint accessoire_pri primary key(nom));

create table utilisation (titre varchar(30),
                          utilisateur varchar(40),
                          accessoire varchar(30),
                          constraint utilisation_pri primary
                          key(titre, utilisateur, accessoire),
                          constraint utilisation_et1 foreign
                          key(titre) references numero(titre),
                          constraint utilisation_et2 foreign
                          key(utilisateur) references personnel(nom),
                          constraint utilisation_et3 foreign key(accessoire) references
                          accessoire(nom));
```

3. CLI1 va peupler ses tables par en exécutant le scripte: insert_cirque1.sql à récupérer sur `http://nakech.free.fr/BDA/`
   Exécuter un `commit` pour valider la saisie.
4. Idem, CLI2 va peupler ses tables par en exécutant le scripte insert_cirque2.sql : h`ttp://nakechb.free.fr/BDA/`

5. CLI1 va peupler ses tables par en exécutant le scripte : `http://nakech.free.fr/BDA/`

6. CLI1 (res. CLI2) complète ses tables en recopiant la saisie de son voisin CLI2 (res. CLI1).
   Indications :

```
Pour CLI1 : insert into personnel select * from CLI2.personnel;
            insert into numero select * from CLI2.numero;
            ...
Pour CLI2 : insert into personnel select * from CLI1.personnel;
            ...
```

   Remarque : Ne pas oublier de remplacer CLI1, CLI2 votre nom d'utilisateur oracle et de **donner les autorisations nécessaires à l'autre utilisateur pour pouvoir accéder à vos tables :**

```
Pour CLI2 : grant select on personnel to CLI1 ;
            ...
Pour CLI1 : grant select on personnel to CLI2 ;
       ...
```

B. Les Vues et les tables système

1. Créer la vue ACOUL (NOM, COULEUR) à partir de la table ACCESSOIRE. Vérifier son contenu.
   Indication : `create view ACOUL as select nom, couleur from accessoire;`

2. À travers de la vue ACOUL, modifier la couleur du « Fouet » en « Noir ». Consulter le contenu de la vue ACOUL et de la table ACCESSOIRE.
   Indication : `update ACOUL set COULEUR='Noir' where NOM='Fouet';`

3. Créer la vue RESP (TITRE, RESPONSABLE, ROLE) à partir des tables NUMERO et PERSONNEL. Vérifier son contenu. Quel est l'intérêt de définir cette vue ?

4. Insérer un tuple dans la vue RESP. Que se passe-t-il ?

5. À partir de la vue système USER_TAB_COLUMNS, afficher les attributs de la table PERSONNEL.

6. À partir de la vue système USER_TAB_COLUMNS, afficher le nom des tables et des vues qui ont pour attribut TITRE.

7. À partir de la vue système ALL_TABLES, afficher le nom des tables dont vous êtes propriétaire (OWNER).

8. Créer la vue MES_TABLES à partir du résultat de la requête précédente. Vérifier son contenu.

C. Transactions

1. Se connecter au même compte (le vôtre !) depuis une deuxième fenêtre SQL*Plus (ne pas quitter la première session).

2. Dans l'une des deux fenêtres, créer la table T(TID, LIB) où la clé primaire TID est un nombre à deux chiffres et LIB une chaîne de 20 caractères maximum. Vérifier en consultant la vue système TAB depuis les deux fenêtres que la table est bien créée.

3. Dans chacune des deux fenêtres, insérer un tuple différent dans la table T. Consulter la table depuis chaque fenêtre. Que constate-t-on ?

4. Annuler l'une des insertions précédentes à l'aide de la commande rollback. Consulter la table depuis chaque fenêtre. Que se passe-t-il ? Faire de même dans l'autre fenêtre. Consulter la table depuis chaque fenêtre. Que se passe-t-il ?

5. Recommencer les insertions d'un tuple différent depuis chaque fenêtre, puis valider l'une des insertions à l'aide de la commande commit. Consulter la table depuis chaque fenêtre. Valider dans l'autre fenêtre. Consulter la table depuis chaque fenêtre. Conclusion ?

6. Dans chacune des deux fenêtres, insérer un tuple identique, cette fois, dans la table T. Que se passe-t-il ?

7. Annuler l'insertion dans la première fenêtre. Que se produit-il dans la deuxième ?

8. Essayer de ré-insérer le même tuple depuis la première fenêtre. Que se passe-t-il ?

9. Valider l'insertion dans la deuxième fenêtre. Que se produit-il dans la première ?

10. Insérer un nouveau tuple depuis une des deux fenêtres et quitter SQL*Plus. Consulter la table de-puis

l'autre fenêtre. Conclusion ?

11. Insérer un nouveau tuple et quitter SQL*Plus. Ouvrir une nouvelle session. La dernière transaction a-t-elle été validée ?

12. Insérer deux nouveaux tuples, puis émettre un rollback. Que sont devenus les deux tuples insé-rés ?

13. Conclusion ? Comment est validée une transaction ?

D. Privilèges

Travailler par groupe de deux ordinateurs : vous et votre (vos voisins**), se coordonner pour avancer dans les questions.**

1. Donner à vos voisins le droit de consulter votre table T. Consulter la-leur.

2. Insérer un tuple dans la table T de vos voisins. Que se passe-t-il ?

3. Donner à vos voisins le droit d'insertion dans votre table T. Insérer un tuple dans la leur.

Travailler par groupe de trois ordinateurs : vous et vos deux voisins (se coordonner pour avancer dans les questions).

4. Donner à vos premiers voisins le droit de consulter votre table T, ainsi que le droit de transmettre ce privilège.

5. Transmettre à vos premiers voisins le privilège transmis par les seconds. Consulter la table T de tous vos voisins.

6. Supprimer à vos premiers voisins le droit de consulter votre table T. Vos seconds voisins peuvent-ils toujours consulter votre table ?

7. Supprimer à tous les autres utilisateurs le droit de consulter votre table T. Vos seconds voisins peuvent-ils toujours consulter votre table ?

NB : Pour réaliser la partie à trois, utiliser la technique de la « permutation circulaire ».

E. Vu globale

Donner à tout le monde le droit de consulter la table **T**.

1. Créer une vue **transaction_globale** permettant de visualiser toutes les saisies réalisées par l'ensemble de votre groupe (au moins 3). Vérifier Indication : Utiliser l'opérateur ***union***

Dossier à rendre : Pour chaque question toutes les requêtes utilisées et vos remarques et commentaires à chaque fois où il y a une nouvelle notion abordé.

Indications :

- Utiliser le scripte sqlplus login.sql : (à récupérer sur mon site http://nakechb.free.fr/BDA/ login.sql). qui permet un affichage convenable sous SQL)

```
set pagesize 25
set linesize 79
set pause on
set pause '[Return] pour la suite'
def _editor=emacs
```

- Utiliser la commande `spool` :

`spool filename.txt` : copie la sortie écran sur le fichier filename.txt
`spool off` : suspend l'opération précédente

- Pour faciliter la rédaction du compte rendu, le sujet est disponible sous le URL: `http://nakechb.free.fr/BDA`

Annexe : Complément du cours

Notion de transaction en base de données

Une transaction est un ensemble de modifications de la base qui forme un tout indivisible. Il faut effectuer ces modifications entièrement ou pas du tout, sous peine de laisser la base dans un état incohérent. Les Systèmes de Gestion de Bases de Données permettent aux utilisateurs de gérer leurs transactions. Ils peuvent à tout moment :

- Valider la transaction en cours par la commande `COMMIT`. Les modifications deviennent définitives et visibles à tous les utilisateurs.
- Annuler la transaction en cours par la commande `ROLLBACK`. Toutes les modifications depuis le début de la transaction sont alors défaites.

En cours de transaction, seul l'utilisateur ayant effectué les modifications les voit. Ce mécanisme est utilisé par les systèmes de gestion de bases de données pour assurer l'intégrité de la base en cas de fin anormale d'une tâche utilisateur : il y a automatiquement `ROLLBACK` des transactions non terminées.

Rappel de quelques instructions SQL utile pour ce TP

operateur `UNION` **:**

```
SELECT ...
UNION
  (SELECT ...
```

operateur `MINUS` **:**

```
SELECT ...
      MINUS
            (SELECT ...)
```

Annexe 1 : Les Tables systeme d'Oracle

Quelques Vues de "méta-base" (dictionnaire de données ou tables système) **décrivant les objets utiles pour ce TP :**

- desc[ribe] tablename : donne le schéma de la relation tablename
- all_catalog : table système donnant toutes les tables accessibles
- user_catalog : table système donnant les seules tables du USER
- all_objects : table système donnant tous les objets accessibles
- user_objects : table système donnant les seuls objets du USER
- user_sys_privs : table système donnant les privilèges système du USER
- user_tab_privs : table système donnant les privilèges sur les objets accessiles

- USER_TAB_GRANTS_MADE : Droits sur ses objets, tables ou vues, donnés par l'utilisateur.
- USER_USERS :Informations sur l'utilisateur courant.
- USER_VIEWS :Texte des vues créées par l'utilisateur.

- USER_TABLES : Description des tables créées par l'utilisateur.
- USER_TAB_COLUMNS : Description des colonnes de chaque table ou vue créée par l'utilisateur courant. Chaque ligne de la vue col décrit une colonne.
- USER_TAB_COMMENTS :Commentaires sur les tables et les vues créés par l'utilisateur.
- USER_TAB_GRANTS_MADE : Droits sur ses objets, tables ou vues, donnés par l'utilisateur.
- USER_USERS :Informations sur l'utilisateur courant.
- USER_VIEWS :Texte des vues créées par l'utilisateur.

- Exemples : les requêtes suivantes permettent de visualiser vos tables actuelles.
  `desc` USER_TABLES **`;`** `(permet de décrire la structure de la table` USER_TABLES)
  `SELECT table_name FROM` USER_TABLES**`;`**

- Synonymes **:** Les noms des vues étant assez longs, les synonymes suivants ont étés définis :

  COLS : Synonyme pour USER_TAB_COLUMNS
  MYPRIV : Synonyme pour USER_USERS
  TABS : Synonyme pour USER_TABLES
  OBJ : Synonyme pour USER_OBJECTS
  SYN : Synonyme pour USER_SYNONYMS
  cat : synonyme pour user_catalog

Exemples :

la requete : `select table_name from tabs;` Affiche une liste des tables de l'utilisateur.
la requete : `select * from SYN …`

Annexe 2 : Exemple du fichier `listener.ora` (ORACLE_HOME/network/admin)

Ce ficher gère les demande d'accès à distance.

```
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC0))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = LIH-NAKECH)(PORT = 1521))
      )  )  )

SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = /oracle/ora92)
      (PROGRAM = extproc)
    )
    (SID_DESC =
      (GLOBAL_DBNAME = ora1)
      (ORACLE_HOME = /oracle/ora92)
      (SID_NAME = ora1)))
```

Nom de la machine serveur

Nom de la base serveur

Annexe 3 : Exemple du fichier `tnsnames.ora`

```
ORACLE_HOME/network/admin
   ORCL =
     (DESCRIPTION =
       (ADDRESS = (PROTOCOL = TCP)(HOST = damspc)(PORT = 1521))
         (CONNECT_DATA =
         (SERVER = DEDICATED)
         (SERVICE_NAME = orcl)       )     )
   EXTPROC_CONNECTION_DATA =
     (DESCRIPTION =
       (ADDRESS_LIST =
         (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC))       )
       (CONNECT_DATA =
         (SID = PLSExtProc)
         (PRESENTATION = RO)        )     )
   BD_MICHEL =
     (DESCRIPTION =
       (ADDRESS_LIST =
         (ADDRESS = (PROTOCOL = TCP)(HOST = 172.16.20.108)(PORT = 1521))       )
       (CONNECT_DATA =
         (SERVICE_NAME = orcl)       )     )
```

Nom local de la BD à distance

```
BD_AURLIEN =
  (DESCRIPTTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL   TCP)(HOST = alO4-07)(PORT   1521))
      )
      (CONNECT_DATA =
        (SERVICE_NAME = orcl)
      )
   )
```

Ip la la machine distante

Nom de la base distante