T.P de BASES DE DONNEES

ADMINISTRATION ORACLE

1 Mise en forme de la base

Ce TP se déroule dans le cadre de la gestion de la médiathèque

La base qui vous est proposée est la propriété de l'utilisateur `MEDIATHEQUE` qui ne vous a donné que des droits de lecture.

- Pour lire ses tables, vous pouvez avoir créé des synonymes afin éviter d'avoir à écrire :
  `select * from mediatheque.adherents ...`
  L'exploration des tables s'est alors faite dans le TP précédent à partir de ces synonymes. Si c'est le cas, détruisez-les (commande : `Drop synonym`).
- Recopiez chez vous les tables suivantes : `adherents`, `auteurs`, `oeuvres`, `ouvrages`, `editeurs`, `exemplaires` et `ecrit`. Pour cela, utilisez la commande CREATE TABLE sous la forme :
  `Create Table ... As Select`.
  L'importation ne permet de récupérer que la structure et les données, pas les contraintes (clé primaire dans le cas présent).
- de préférence à `tab`, ancienne vue qui n'est plus adaptée à l'architecture d'`Oracle10` sous laquelle vous travaillez, construisez et utilisez, pour vérifier votre travail, la vue suivante :

```
create view obj(nom, type) as select substr(object_name,1,30),
                                     object_type from user_objects;
```

  C'est cette vue qui va vous permettre de "voir", à partir du dictionnaire des données, les différents objets que vous possédez (tables, vues, sequences, index,...). En ce qui concerne les contraintes, vues comme des index, vous trouverez davantage d'information en consultant la table `user_constraints` du dictionnaire des données.

  Vous pourrez en effet constater en effet que `Select * from tab;` ne vous donne pas d'informations sur les contraintes dont vous êtes le propriétaire. Grâce à `obj`, qui vous donne une liste plus complète des objets en votre possession, vous pouvez vérifier maintenant que, lorsque vous importez des tables, vous ne pouvez importer en même temps leurs contraintes, qui sont liées au schéma général de la base qui les contient. Il va donc vous falloir recréer ces contraintes sans faire de fautes, ce qui est assez ... contraignant, et moins simple qu'il n'y paraît.
- Introduisez des contraintes (de clé primaire et référence), sans oublier de les nommer, en utilisant la commande : `Alter table nomTable ADD Constraint ...`, sur les tables : `auteurs`, `oeuvres` et `ecrit`.

  Ces contraintes sont définies à l'aide des clauses suivantes :
  (`PRIMARY KEY ...` et `FOREIGN KEY ... REFERENCES ...`).

  En cas d'erreur, vous pouvez, avant de recommencer, détruire une contrainte par la commande :

```
alter table <nomTable> DROP CONSTRAINT <nomContrainte>;
```

  à condition bien sûr d'avoir préalablement pensé à nommer la contrainte. Sinon, son nom lui a été donné par les système. On peut le retrouver en explorant la vue `USER_CONSTRAINTS` du dictionnaire des données (faites un DESC, puis le SELECT approprié).

2 Vues et autorisations

En tant qu'administrateur, vous avez à gérer la base afin qu'elle puisse être convenablement utilisée en fonction des besoins de chacun. Vous allez donc devoir considérer un autre utilisateur, que nous appelerons U, dont vous allez faire jouer le rôle par un de vos voisins de table.

U sera chargé de gérer la médiathèque à partir des vues suivantes :

- la vue GESTION qui donne accès aux informations concernant le prix et l'état des livres empruntés.
- la vue AUDIO pour connaître, sans répétition, quels sont les disques empruntés.
- la vue TOUT qui réunit toutes les informations issues des deux vues qui précèdent.

1. Ecrivez la requête permettant de connaître, à partir de la seconde vue, quels sont les adhérents qui empruntent des CD.
2. Choisissez un voisin pour représenter U. Donnez lui des droits de lecture sur l'une des vues que vous venez de créer. Vous utiliserez la commande SQL :

   ```
   GRANT <liste de droits et/ou privilèges> ON <objet>
                                   TO <utilisateur>|PUBLIC [WITH GRANT OPTION];
   ```

   Quand c'est fait, pensez que vous travaillez en mode Client/Serveur et qu'aucune des opérations que vous réalisez n'est prise en compte au niveau du serveur tant que vous n'avez pas fait de `COMMIT`.

   Vous pourrez vérifier que les droits ont bien été accordés en examinant la table du dictionnaire des données : `USER_TAB_PRIVS`. Vous pouvez aussi vérifier la définition des vues à partir de la table `USER_VIEWS` du dictionnaire des données. Demandez à U de faire un accès à votre vue et assurez-vous qu'il a réussi.
3. Créez maintenant un rôle pour définir le comportement d'utilisateurs tels que U. Supprimez les privilèges que vous venez d'accorder à ce dernier et attribuez-lui, à la place, le rôle qui lui revient.

   La syntaxe des commandes à utiliser est :

   ```
   REVOKE <droits et/ou privilèges> ON <objet> FROM <utilisateur>;

   CREATE ROLE <nomDuRole>;

   GRANT <liste de droits> TO <nomDuRole>;

   GRANT <nomDuRole> TO <utilisateur>;
   ```

4. Attribuez le rôle à votre voisin. Demandez-lui de tester à nouveau ses droits sur votre vue. Si vous faites un `commit` juste après avoir exécuté le `drop`, vous pouvez lui demander au préalable de constater qu'il a perdu le droit de lecture accordé par le premier `grant`.
5. Retrouvez le texte de définition des vues dans la table `USER_VIEWS` du dictionnaire.
6. Parmi les vues que vous venez d'écrire, y en a t-il pour lesquelles on ne peut pas mettre des données à jour ?

3 Modifications de la stucture des tables

- Créez, *sans lui définir de clé primaire*, la table :

  `Critiques(numcrit, auteur, revue, dateparution,texte)`

  Cette table permet d'enregistrer les critiques parues dans la presse et concernant une œuvre particulière d'un auteur donnée.
  Vérifiez sa structure par la commande `desc`.
- Ajoutez à cette table l'attribut `titre`, désignant le titre de l'ouvrage concerné, correspondant à une chaîne fixe de 10 caractères, par la commande : `ALTER TABLE ... ADD ...`

– La longueur de cette zone n'étant manifestement pas suffisante, modifiez le type de l'attribut pour en faire une chaîne dynamique de 300 caractères, par `ALTER TABLE ... MODIFY...`
– Rentrez des données quelconques (en petit nombre) dans cette table. A cet effet, vous pourrez vous servir par exemple d'un éditeur (tel `notepad+` par exemple) qui ne crypte pas le contenu (surtout pas `word` donc !) ; dans ce cas vous rentrerez dans un fichier `.sql` une série d'instructions :
`INSERT INTO ... VALUES ...` que vous lancerez en effectuant un copier-coller dans la fenêtre d'exécution.

exemple :

```
1.     'Pascal', 'Lire', 'octobre 2009', 'absolument boulversifiant !'

2.      'Descartes', 'Lecture pour tous', 'septembre 2009', 'Le style est assez bon, mais on reche
```

Grâce à ,la pertinence de ces critiques, vous aurez naturellement deviné que le premier article concerne "Les Pensées" et le second les "Méditations philosophiques".
– Il manque dans cette table la liaison avec l'oeuvre critiquée. Rajouter un attribut `numo` destiné à pointer vers la clé des oeuvres.
Assurez la mise à jour des enregistrements pour la valeur de ce champs, en recherchant la bonne valeur de la clé de l'œuvre concernée à l'aide d'un select portant sur le titre. La syntaxe du `select` permet en effet d'écrire :

```
update T set att= (select ...);
```

Dans l'utilisation des informations précédentes, portez une attention particulière à la casse et aux accents (lower, upper, initcap, soundex).
– Arrangez-vous pour faire figurer un doublon dans cette table (deux lignes absolument identiques ; l'absence de contrainte de clé primaire vous permet de le faire sans réaction du processeur de contraintes.
– Il vous sera impossible de rajouter cette contrainte oubliée lors de la création de la table sans supprimer le doublon que vous venez d'insérer. Pour ce faire, vous devrez vous servir de l'attribut caché `rowid` qu'il vous faut visualiser au préalable. Supprimez alors un des termes du doublon. Par la suite, le rajout de la contrainte de clé primaire devient possible.

4 Le processeur de contraintes

Cette partie du TP est destinée à vous faire manipuler le processeur de contraintes, afin d'en mieux comprendre le fonctionnement.

1. Commencez à créer les tables suivantes :

```
create table essai(cle number, valeur varchar2(50), constraint pk_es primary key(cle));

create table liaison(key number, lien number, constraint pk_l primary key(key),
                         constraint fk_le foreign key(lien) references essai(cle) );
```

2. Essayez maintenant d'insérer dans `essai` le tuple suivant : (null, 'abc')
Efforcez-vous de trouver une explication rationnelle à l'action du SGBD.
3. Insérez ensuite deux fois dans cette même table essai le tuple : (1,'abc') et interprétez la réponse d'Oracle.
4. Insérez maintenant le tuple de valeur (1,2) dans la table `liaison`
Que montre cette expérience ?
Corrigez en insérant le tuple (1,1) dans `liaison`
5. Poursuivez en tentant une mise à jour de `essai` : donnez à la clé la valeur 2 au lieu de 1 et expliquez la réponse d'Oracle.
6. Tentez maintenant de supprimer la ligne d'`essai`.

7. Supprimez la table `liaison` et recréez-la avec la définition suivante :

```
create table liaison(key number, lien number, constraint pk_l primary key(key),
     constraint fk_le foreign key(lien) references essai(cle)   on delete cascade);
```

   Insérez-y le tuple (1,1), puis supprimez la ligne d'`essai`, et enfin affichez la table `liaison`. Que constatez-vous ?

8. Supprimez à nouveau la table `liaison` et recréez-la avec la définition :

```
create table liaison(key number, lien number, constraint pk_l primary key(key),
    constraint fk_le foreign key(lien) references essai(cle)   on delete set null);
```

   Insérer dans `essai` le tuple (1,'abc') et dans `liaison` le tuple (1,1). Détruisez ensuite la ligne créée dans `essai` et affichez le contenu de la table `liaison`. Expliquez le résultat.

5 Dictionnaire des données

Au cours de ce TP, vous avez dû assez souvent explorer les tables ou les vues du dictionnaire des données. Si vous êtes passionné(e) ou simplement curieux (se), vous pouvez maintenant vous plonger dans ce fabuleux outil d'administration, à la fois pour vous faire une idée de sa richesse et de sa complexité et pour vérifier sous quelle forme les opérations de gestion que vous avez mené ont été prises en compte.

5.1 Tables du dictionnaire

La base du dictionnaire est détenue par le super-administrateur `SYS`, qui voit les tables suivantes (toutes ne sont pas indiquées) du véritable dictionnaire :

USER$	Liste des utilisateurs et des rôles
OBJ$	Objets des utilisateurs
TS$	Tablespaces
FILE$	Fichiers
SEG$	Segments
TAB$	Tables de la base
CLU$	Clusters
IND$	Indexes
ICOL$	Colonnes accessibles par un index
CON$	Liste des contraintes
CDEF$	Définitions des contraintes
CCOL$	Colonnes sur lesquelles portent des contraintes

Vous ne pouvez voir ces tables qu'en préfixant leur nom par celui de leur propriétaire, qui vous a donné les droits de lecture : select * from sys.nomTable$;

Le dictionnaire courant accessible à l'administrateur `SYSTEM` ainsi qu'aux utilisateurs ordinaires est composé de vues générales, permettant d'avoir la liste des objets de la base. Ce sont des vues construite sur le dictionnaire interne, même quand elles sont qualifiées de tables. Certaines sont issues des anciennes versions d'`Oracle` pour des raisons de compatiblité.

- `TAB`, depuis la V5, indique la liste générales des objets de l'utilisateur.
- `DICTIONARY` donne une liste exhaustive des tables du dictionnaire.
- Les tables renseignant les objets d'un utilisateur sont préfixées par `USER_`. Des tables de même schéma existent avec les préfixes `ALL_` et `DBA_`.

Recherchez dans les tables du dictionnaire la trace des tables ou synonymes que vous possédez.

5.2 Gestion des données

Essayez les commandes :

```
SELECT object_name, object_type FROM user_objects;
SELECT * FROM all_tables;
SELECT table_name, column_name, data_default
```

```
       FROM user_tab_columns WHERE tablename = 'unNomDeTable';
SELECT table_owner, table_name, synonym_name
       FROM sys.dba_synonyms where owner ='votreLogin';
```

Retrouvez la liste des objets de l'utilisateur MEDIATHEQUE à partir de la table `ALL_TABLES`. Au préalable, demandez sa description (commande `desc`) et ne retenez que les objets dont MEDIATHEQUE est le propriétaire.

5.3 gestion des utilisateurs

Vous pouvez vous faire une idée des données prises en compte en essayant :

```
SELECT * FROM all_users;
SELECT * FROM sys.dba_sys_privs;
SELECT * FROM sys.dba_role_privs;
SELECT table_name, privilege, grantable
       FROM sys.dba_tab_privs WHERE grantee='votreLogin';
SELECT grantee, table_name, column_name, privilege
       FROM sys.dba_col_privs;
SELECT * from session_privs;
SELECT * FROM session_roles;
SELECT * FROM sys.dba_roles;
SELECT * FROM role_sys_privs WHERE role='LICENCE0910';
```

Les différents privilèges existants, (certains sont détenus pas ce dernier rôle) sont listés dans la table `system_privilege_map`

5.4 Gestion physique des donnés

Les informations concernant les données de l'utilisateur sont contenues dans les tables : USER_EXTENTS, USER_SEGMENTS, USER_FREE_SPACE, DBA_USERS, DBA_TS_QUOTAS, USER_TABLESPACES, DBA_DATA_FILES, et V$DATAFILES.

Essayez les commandes SQL et retrouvez l'occupation des tables de l'utilisateur MEDIATHEQUE. :

```
SELECT tablespace_name, initial_extent, next_extent, max_extents, min_extents, pct_increase
       FROM sys.dba_tablespaces;
SELECT segment_name, tablespace_name, bytes, blocks, extents
       FROM sys.dba_segments WHERE segment_type='ROLLBACK';
SELECT segment_name, bytes, blocks
       FROM sys.dba_extents WHERE segment_type='ROLLBACK';
SELECT tablespace_name, file_id, bytes, blocks
       FROM sys.dba_free_space;
```

6 L'optimiseur - introduction au "tuning"

L'optimiseur du SGBD en constitue un élément fondamental. Sans lui, ses performances ne seraient pas assurées et dépendraient fortement du style d'écriture du concepteur des requêtes et des aléas de l'implémentation.

Il existe sous Oracle deux outils pour appréhender les éléments de son fonctionnement et éventuellement pour modifier sa stratégie d'exécution : les commandes `Analyse` et `Explain Plan`.

La première crée des statistiques de coût pour les accès aux tables, la seconde permet l'analyse très fine du coût d'exécution des requêtes.

1. ANALYZE :

   La syntaxe de la commande existe sous trois forme :

```
analyze table <nomTable> compute statistics;
analyze table <nomTable> estimate statistics;
analyze index <nomIndex> estimate statistics;
```

La commande retourne les résultats suivants : nombre de lignes, nombre de blocs de donnée, nombre de blocs inutilisés, espace libre de blocs, longueur moyenne es enregistrements, nombre de lignes chaînées en ce qui concerne les tables. Toutes ces informations sont alors accessibles à travers le dictionnaire des données :

```
select * from user_tables where table_name=<nomTable>
```

Pour ce qui est des index, elle fournit : la profondeur de l'arbre B+ qui implémente l'index ; le nombre de blocs d'index, le nombre de clés, les valeurs mini et maxi des clés, le nombre moyen de feuille d'index par clé et de feuilels de données par clé.

Toutes ces notions qui vous paraissent actuellement abstraites seront précisées dans l'exposé sur l'architecture des SGBD. Vous pouvez cependant dès à présent les acquérir en vous reportant au poly de cours auquel vous pouvez accéder par le portail.

Calculez et visualisez ces statistiques pour la table `adherents` et sa clé primaire

2. EXPLAIN PLAN :

   Cette commande est plus complexe et nécessiterait à elle seule une séance de TP. Cependant, là encore pour vous faire une idée des éléments sur lesquels opère l'optimiseur, vous pouvez lancer la commande :

   ```
   set autotrace  on explain.
   ```

   A partir de maintenant, et pour la durée de la session (à moins que vous ne tapiez `set autotrace off`), toute exécution d'une requête sera accompagnée du plan d'exécution suivi par l'optimiseur.

   Testez cette commande sur des requêtes simples à écrire et ne ramenant pas trop de résultats (afin éviter d'avoir à balayer plusieurs pages avant d'accéder au résultat de l'analyse).