TP1 Architecture Oracle (1h30)

1. Les tables et les blocs (ou pages) Oracle

Le système alloue un certain nombre de blocs de données à chaque table en fonction des besoins de stockage. Les meileurs compromis, pour accéder le plus rapidement possible aux enregistrements au sein des blocs, sont également recherchés. Nous explorons les allocations de blocs opérées en fonction des activités menées sur les tables. Nous regardons dans un second temps l'organisation logique (tablespace, segment, extent) qui va faciliter l'accès aux blocs de données au regard des besoins.

1.1 Exploiter les vues USER_TABLES et DBA_TABLES

Les informations concernant l'organisation logique et physique des tables de votre schéma utilisateur sont disponibles depuis les vues système user_tables et dba_tables. Les attributs suivants donnent une idée du nombre de blocs alloués à chaque table :

1. NUM_ROWS : nombre de lignes
2. BLOCKS : nombre de blocs utilisés
3. EMPTY_BLOCKS : nombre de blocs alloués, mais non utilisés
4. AVG_SPACE : espace libre moyen dans les blocs utilisés
5. AVG_ROW_LEN : longueur moyenne des lignes (tuples) (en octets)

1.1.1 Questions préalables

1. **Espace de table cible**
   Vérifiez que que toutes vos tables sont situées dans le même espace de tables (attribut TABLESPACE_NAME dans USER_TABLES et DBA_TABLES) et qu'il en va de même pour les autres schémas utilisateurs. Vous donnerez le nom de ce tablespace.

2. **Taille du bloc Oracle**
   Vous rechercherez également la taille en octets du bloc oracle (multiple de 1024 octets). Regardez pour ce faire les commandes sqlplus et sql :

```
show parameter
show parameter db_block_size

select name, value from v$parameter where name like 'db_block%';
```

Listing 1 – taille du bloc

3. **Nombre de blocs Oracle pour une table d'un schéma**
   Combien de blocs de données sont utilisés pour contenir les données d'une de vos tables au choix ? Ces informations sont évaluées ou ré-évaluées par la commande

```
analyze table nom_table compute statistics
```

Vous pouvez évaluer toutes les tables de votre schéma utilisateur avec l'aide du paquetage DBMS_STATS.

```
-- semble ne pas fonctionner, preferer analyze table
exec dbms_stats.gather_schema_stats(USER)
-- usage autre paquetage effectif
exec dbms_utility.analyze_schema(USER,'COMPUTE')
```

Renvoyez le nombre de blocs alloués par table (BLOCKS + EMPTY_BLOCKS). Donnez le coût en octets.

1.2 Mécanismes d'allocation

Nous allons définir une table de test, puis l'alimenter à différents temps à l'aide d'une procédure PL/SQL qui vous est donnée (remplissage.sql).
Créer la table test(num char(3), commentaire char(97)), et appliquer sur num une contrainte de domaine : nombres entiers compris entre 0 et 999

1.2.1 Question 1

Est-ce que la longueur des tuples de cette table est variable ? Relever le nombre de blocs utilisés, alloués mais inutilisés, . . .après :

1. la création de la table
2. l'insertion de 50 lignes
3. l'insertion de 100 lignes supplémentaires
4. l'insertion de 100 lignes supplémentaires
5. l'insertion de 100 lignes supplémentaires

Comparer à chaque étape l'espace réellement occupé par la table et l'espace théoriquement nécessaire à sa mémorisation. Comment expliquer les différences ?

1.2.2 Question 2

Supprimer un tiers des tuples de la table (par exemple, ceux dont le num est divisible par 3).

```
select num from test where mod(num,3) = 0;
```

Listing 2 – exemple pour suppression

Réévaluer les statistiques sur cette table à l'issue de l'opération. Validez la suppression et tester à nouveau. Supprimer l'intégralité des tuples, avec avec l'ordre

```
delete from test;
```

Listing 3 – delete

valider (commit) et regénérez les statistiques. Que constatez-vous ? Passer par l'ordre suivant

```
truncate table test;
```

Listing 4 – truncate

et ré-évaluer les statistiques. Que constatez vous ?

1.3 Vues système associées à l'organisation logique

Vous réalimenterez la table TEST en insérant 600 tuples. Puis vous testerez les requêtes suivantes qui exploitent l'organisation logique des tables :

```
select tablespace_name, segment_name, blocks, bytes/1024 as Koctets, extents from
    user_segments order by segment_name ;

select segment_type, extent_id, bytes/1024 enKo, blocks from user_extents where
    segment_name = 'TEST';
```

Listing 5 – organisation logique

Vous répondrez aux questions :

— Quel est le nombre de segment(s) associé(s) à la table TEST ?
— Est ce que le segment associé à la table TEST organise logiquement l'information d'une autre table de la base ?
— Quel est le nom du segment associé à la table la table TEST ?
— Quel est le type du segment associé à la table TEST ?
— Quel est le nom de l'espace de tables dans lequel se range ce segment ?
— De combien d'extensions (extents) se compose le segment ? Est que ce nombre d'extensions peut évoluer avec l'ajout de nouveaux tuples dans la table ?