TP Oracle et les index

1. Préalable

Le TP se consacre à une meilleure compréhension de l'utilisation d'index dans le périmètre du SGBD Oracle. Nous nous limitons pour l'instant aux index de type B+Tree. Vous créerez une table COMMUNE à partir de la table COMMUNE du schéma utilisateur IMOUGENOT.

2. Consultation des vues du méta-schéma relatives aux index

Les vues index_stats et user_indexes aident à la compréhension des structures d'index manipulées par un serveur de base de données. L'ordre SQL donné ci-dessous exploite user_indexes et permet par exemple de consulter pour l'ensemble des index définis sur le schéma utilisateur, le nom de l'index, le nom de la table impactée par l'index ainsi que la hauteur de l'arbre (sans le niveau des feuilles) [1].

```
SELECT index_name, blevel, table_name FROM user_indexes;
```

La vue index_stats donne des informations complémentaires (parfois chevauchantes) à la vue user_indexes. Il est ainsi possible de disposer d'informations sur la place mémoire occupée par l'index, le nombre de blocs occupés par les nœuds branches (BR_BLKS) et les nœuds feuilles de l'arbre (LF_BLKS). Il est cependant nécessaire de collecter les statistiques sur les index avant de consulter cette vue. La consultation ci-dessous retourne respectivement le nom de l'index, l'espace occupé en octets, le nombre de répétitions pour la valeur de clé la plus répétée, le nombre de tuples (clé, rowid, pointeur tuple gauche, pointeur tuple droit) au niveau feuille, le nombre de tuples (clé, pointeur) au niveau nœud des branches et la hauteur de l'arbre (avec le niveau feuille et donc égal à blevel+1).

```
-- mettre a jour les statistiques pour un index donné
ANALYZE INDEX <index_name> VALIDATE STRUCTURE;

SELECT name, btree_space, most_repeated_key, lf_rows, br_rows, height FROM index_stats;
```

2.1 Exercices

Vous exploiterez les vues du méta-schéma et notamment les vues index_stats et user_indexes pour répondre aux questions suivantes

1. Que renvoie la requête suivante ?

   ```
   select rowid, rownum, code_insee from commune;
   ```

2. quels sont les index déjà présents sur votre schéma utilisateur ? Expliquez leur présence

1. La valeur 0 indique que l'index n'est constitué que d'un niveau racine

3. Indiquez si ces index sont uniques ? denses ?
4. Quelle est la hauteur de la taille de l'index de la table COMMUNE ?
5. Quels sont les nombres de blocs de branches et de feuilles, qui ont été réservés pour l'index de la table COMMUNE ?
6. Quelle est la taille de chaque tuple (clé, pointeurs, rowid) présent au niveau des blocs des feuilles (on considère que les blocs sont pleins) ?
7. Par comparaison, quelle est la taille moyenne de chaque tuple de la table COMMUNE et combien de tuples peuvent être stockés dans un bloc (tenir compte de l'espace libre et donc de la valeur de PCT_FREE de la vue USER_TABLES) ?
8. Calculer le nombre de blocs théorique nécessaire au stockage de l'ensemble des tuples de la table et comparez ce nombre avec la valeur de l'attribut BLOCKS de USER_TABLES. Le nombre calculé vous semble-t'il réaliste ? Penser à mettre à jour au préalable, les statistiques de la table d'une des deux manières suivantes :

   ```
   --
   ANALYZE TABLE COMMUNE COMPUTE STATISTICS;
   -- paquetage statistiques et schéma utilisateur ISA
   EXEC DBMS_STATS.gather_table_stats('ISA', 'COMMUNE');
   EXEC DBMS_STATS.gather_index_stats('ISA', 'COMMUNE');
   ```

9. Dans quel espace de tables est organisé logiquement l'index COMMUNE_PK ?
10. Quel est l'espace de stockage (en Mo) nécessaire pour la table COMMUNE ? Quel espace de stockage supplémentaire faut'il rajouter pour stocker l'index COMMUNE_PK ?

3. Manipulation du paquetage DBMS_ROWID

L'adresse de chaque enregistrement sur le disque (identifiant de ligne ou rowid) renferme différentes informations à l'exemple du numéro de l'enregistrement, du numéro de l'objet associé (table ou index), du bloc de données qui contient cet enregistrement, ou de l'adresse relative du fichier qui contient les blocs de données. Le paquetage DBMS_ROWID permet d'exploiter l'ensemble de cette information. Des exemples vous sont donnés :

```
DECLARE
  object_no   integer;
  row_no integer;
  row_id ROWID;

BEGIN
  SELECT ROWID INTO row_id FROM commune
    WHERE codeInsee = '34172';
  object_no := DBMS_ROWID.ROWID_OBJECT(row_id);
  row_no := DBMS_ROWID.ROWID_ROW_NUMBER(row_id);
  DBMS_OUTPUT.PUT_LINE('The obj. # is '||object_no||' '||row_no);
END;
/

SELECT DBMS_ROWID.ROWID_BLOCK_NUMBER(rowid), DBMS_ROWID.ROWID_OBJECT(rowid), nom_com
   FROM Commune where codeInsee = '34172';
```

```
-- nom de la structure de table ou d'index
select owner, object_name from dba_objects where data_object_id = ...
```

Vous construirez une procédure PL/SQL qui permet d'afficher tous les enregistrements (codeInsee, nom_com) contenus dans le même bloc de données qu'un enregistrement donné de la table Commune (par exemple l'enregistrement dont le code INSEE est 34172). Le nombre de tuples listés par bloc est t'il en accord avec les informations collectées dans la vue user_tables

```
select blocks, avg_row_len from user_tables where table_name ='COMMUNE';
```

4. Construction, utilisation d'index et évaluation du temps

4.1 Construction et manipulation d'index

Vous créerez un index non unique (DEP_IDX) sur l'attribut DEP de COMMUNE et un index unique (COM_IDX) sur la fonction très simple LOWER(NOM_COM). Consulter user_indexes, index_stats et user_extents pour avoir quelques informations sur les index créés. Vous exploiterez les syntaxes suivantes pour soit désactiver, soit supprimer les index (DEP_IDX, COM_IDX, COMMUNE_PK) :

```
alter index <nom_index> disable;
alter index <nom_index> enable;
drop index <nom_index>;
```

Que remarquez vous ?

4.2 Premières évaluations de performance

Nous reviendrons dans un prochain TP sur la manière de tracer une requête et d'en évaluer la qualité d'exécution avec des mécanismes tels que EXPLAIN PLAN. Pour l'instant, nous allons nous contenter de porter une réflexion sur l'utilisation ou non d'une structure d'index par le système selon la nature de la requête. Une liste de requêtes vous est donnée et vous essaierez de déterminer si l'index COMMUNE_PK est exploité ou non (justifiez vos réponses). Exploitez le temps écoulé pour l'exécution pour tenter d'étayer vos réponses. Assurez vous d'avoir supprimé les index sur NOM_COM et DEP au préalable (à défaut supprimez les).

1. select code_insee from commune ?
2. select nom_Com from commune ;
3. select nom_Com, code_insee from commune ;
4. select nom_Com from commune where code_insee='34192' ;
5. select nom_Com from commune where code_insee like '34%' ;
6. select nom_Com from commune where code_insee >= 34 ;
7. select nom_Com from commune where code_insee in ('09330','09331','09332','09334') ;

Une directive (ou hint) dans une requête SQL permet entre autres de faire le choix sur l'exploitation d'un index ou non dans une requête. Des exemples de manipulation de cette directive vous sont donnés. Vous les exploiterez pour là encore tenter de vérifier vos réponses.

```
select  /*+ NO_INDEX(Commune) */ *
    from    Commune;

select  /*+ NO_INDEX(Commune) */ code_insee, nom_com from commune
where code_insee = '09342' ;
```

4.3 Evaluation du temps

Il existe différents manières d'évaluer le temps d'exécution d'une requête dans Oracle. Nous avons dans le cours précédent que l'information pouvait être obtenue à partir des attributs cpu_time et elapsed_time de la vue v$sql. Une alternative simple est d'exploiter la variable d'environnment TIMING (set timing on pour l'activation et set timing off pour la désactivation). Un exemple d'utilisation vous est donné que vous testerez sur une requête élémentaire sur COMMUNE.

```
set timing on
select dep from commune;
... les résultats
Elapsed: 00:00:00.62
```

Vous pouvez manipuler aussi l'endormissement du processus, pour avoir une meilleure maîtrise du temps écoulé.

```
begin
    dbms_lock.sleep(10);
end;
/
```

Le paquetage dbms_utility donne accès aux méthodes get_time et set_time pour manipuler le temps. Testez également le programe principal suivant. Pourquoi à votre avis le temps écoulé est bien plus long avec ce code ?

```
DECLARE
   l_start NUMBER DEFAULT dbms_utility.get_time;
BEGIN
   FOR i IN
        (SELECT dep
         FROM   commune)
 LOOP
   dbms_output.put_line ('name    : '|| i.dep);
   END LOOP;
   dbms_output.put_line(round( (dbms_utility.get_time-l_start)/100, 2 ) || ' seconds...');
END;
/
```

4.4 Autres tests

L'index COMMUNE_PK a été construit automatiquement par le système lors de la définition de la contrainte de clé primaire COMMUNE_PK. Si la contrainte de clé primaire est désactivée, est ce

que l'index peut continuer à être utilisé ? Pourquoi à votre avis ? Que se passe t'il à la réactivation de la contrainte ? De même, quel est le devenir de l'index si la contrainte de clé primaire est supprimée ?

5. Estimation de la taille d'un index

La construction d'un index entraîne un surcoût de stockage qui peut s'avérer non négligeable. Vous testerez les fonctionnalités d'estimation de la taille en blocs (alloués et utilisés) d'un index dont la création est envisagée (testez sur DEP_IDX et COM_IDX). Vous vérifierez ensuite la pertinence de ces estimations en recréant DEP_IDX et COM_IDX et en interrogeant la vue USER_SEGMENTS. L'utilisation de la procédure CREATE_INDEX_COST se fait par le biais de variables d'environnement (bind variables) et donc l'utilisation du SQL Dynamique. Nous verrons dans le prochain TP que ces variables présentent un intérêt tout particulier dans tout ce qui relève de l'exécution de requêtes.

```
-- declaration des variables de session
variable used_bytes number
variable alloc_bytes number
exec dbms_space.create_index_cost( 'create index testCommune_idx on commune(nom_com)',
:used_bytes, :alloc_bytes );
-- affichage contenu variables
print :used_bytes
print :alloc_bytes

-- valeur reelle de la taille

create index testCommune_idx on commune(nom_com);
select trim(to_char(bytes, '999,999,999')) actual_bytes
    from user_segments where segment_name = 'TESTCOMMUNE_IDX';
```