TD index

1. Exercice 1 : construction d'un arbre

Construire un arbre B+ avec pour valeurs de clé d'accès, les numéros de département, en supposant qu'il y a au plus 2 enregistrements par bloc (ordre 1), et en prenant les enregistrements dans l'ordre donné ci-dessous.
17, 14, 20, 3, 8, 25, 30, 19

2. Exercice 3 : Tailles table et index

Une table contient 50 000 tuples de 100 octets environ chacun. La taille du bloc (ou page) de données est de 8192 octets et 1192 octets sont réservés pour l'en-tête et de nouvelles insertions/modifications (PCTFREE). L'espace de bloc disponible pour héberger les enregistrements de tuples est donc de 7000 octets.

2.1 Question 1

Vous calculerez le facteur de blocage (nombre de tuples / bloc). Vous donnerez le nombre de blocs (plus taille en octets) nécessaire au stockage du contenu de la table.

2.2 Question 2

Pour l'instant aucun index n'est construit. Vous donnerez le coût en nombre de blocs à parcourir, lors d'une requête de consultation sélective (renvoi d'un tuple résultat) :

— quand ce tuple existe
— quand ce tuple n'existe pas

Nous considérons que les blocs des données la table sont seulement en mémoire secondaire, et que le temps d'accès à chaque bloc est de l'ordre de la centième de seconde (0.01 sec). Vous donnerez aussi le temps nécessaire à l'exécution de la requête (lorsque le tuple recherché est présent et lorsqu'il est absent).

2.3 Question 3

Un index unique et dense de type arbre B+ est défini à partir de l'attribut clé de la table. La taille des enregistrements d'index est de 20 octets. Vous calculerez le facteur de blocage pour les enregistrements d'index. Vous donnerez le nombre de blocs (plus taille en octets) nécessaire au stockage de l'index. Vous donnerez ensuite la hauteur et l'ordre (nombre de tuples par bloc divisé par 2) de l'arbre B+.

2.4 Question 4

Supposons que l'on consulte la table sur la base d'une sélection sur l'attribut clé (comparaison avec opérateur d'égalité). Quel est le coût en nombre de blocs pour obtenir le tuple recherché :

— quand ce tuple existe
— quand ce tuple n'existe pas

2.5 Question 5

Supposons que l'on consulte la table sur la base d'une sélection sur un attribut non clé (comparaison avec opérateur d'égalité). Quel est le coût en nombre de blocs pour obtenir le tuple recherché :

— quand ce tuple existe
— quand ce tuple n'existe pas

2.6 Question 6

La table compte pour l'instant, 50 000 tuples. Combien faudrait il de tuples supplémentaires pour que l'index B-Arbre associé nécessite un niveau de branche intermédiaire supplémentaire ?

3. Exercice 3 : index et contrainte

Quelles sont la ou les contraintes qui donnent lieu à la définition implicite d'un index associé ?

1. contrainte de clé primaire (primary key)
2. contrainte de clé étrangère (foreign key)
3. contrainte d'unicité (unique)
4. contrainte de non nullité (not null)
5. contrainte de domaine (check)

Pour l'index ou les index construits implicitement, s'agit t'il d'un arbre B+, d'un index de hachage ou bien d'un index bitmap ? Comment désactiver ou encore supprimer ces index ?

4. Exercice 4

Cet exercice est à faire à la fois, sur feuille de papier et à tester sur machine. L'idée est en effet d'évaluer les ordres de grandeur obtenus de part et d'autre.

4.1 Question 1

Vous referez les mêmes calculs mais cette fois-ci pour une table de 1 000 000 de tuples de 50 octets chacun, une capacité du bloc de 8192 octets privés de 10% et un index dense de type arbre B+ avec des tuples de 15 octets.

4.2 Question 2

Un programme principal alimente une table nommée ABC avec 1 000 000 de tuples de 50 octets. Le paquetage DBMS_RANDOM est mobilisé pour générer des valeurs textuelles aléatoirement. La fonction string de ce paquetage va permettre de générer des chaînes de caractères d'une taille spécifiée en minuscules (L pour Lowercase) ou en majuscules (U pour Uppercase).

```
drop table abc;
create table ABC (A number, B varchar(20), C varchar(20)) ;

 declare i number;
  begin
  for i in 1..1000000
  loop
  insert into abc values (i,dbms_random.string('L', 20),dbms_random.string('U', 20)) ;
  end loop ;
  commit ;
 end ;
/
```

Un arbre B+ est ensuite créé lors de la définition de la contrainte de clé primaire sur l'attribut A (voir ordres ci-dessous). Vous comparerez et commenterez les résultats obtenus par les requêtes portant sur user_tables et index_stats avec vos calculs précédents.

```
alter table abc add constraint abc_pk primary key (a);

analyze table abc compute statistics ;

select avg_row_len, num_rows, blocks, avg_space, empty_blocks
from user_tables where table_name = 'ABC';

analyze index abc_pk validate structure ;


set linesize 180
select height, blocks, lf_rows, lf_rows_len, lf_blks, br_rows, br_blks, br_rows_len
from index_stats;

select segment_name, blocks, bytes/1024/1024 from dba_segments where owner = 'E20....';
```

4.3 Question 3

Pensez vous que l'index est utilisé pour les ordres de consultation suivants ? (justifiez) :

```
select A from ABC where A = 10001 ;

select A from ABC ;

select A, B from ABC ;

select C from ABC where A >=20 AND A <=40

select A from ABC where C like '%ABC%';
```

4.4 Question 4

Pensez vous que l'index est utilisé et comment lors d'un ordre de suppression ? Considérez les ordres DELETE suivants :

```
delete from ABC where A = 100;

delete from ABC where A >=20 AND A <=40 ;

delete from ABC where B like 'ab%';
```

Pensez vous qu'un ordre de suppression quel qu'il soit, induit un coût supplémentaire sur l'index, et lequel ?