TD index

1. Exercice 1 : construction d'un arbre

Nous supposons que nous disposons d'une table Departement qui permet de gérer les informations sur les départements français.
Departement(numD, nomD, codeChefLieu, numReg)
Construire un arbre B+ avec pour valeurs de clé d'accès, les numéros de département (numD), 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 2 : 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) :

— en moyenne, 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

L'exercice 4 est à faire à la fois, sur feuille de papier (cette semaine) et à tester sur machine lors des prochains TPs (seule la première question est donnée cette semaine). L'idée est en effet d'évaluer les ordres de grandeur obtenus de part et d'autre.
Les données considérées correspondent à la table suivante :
ABC(<u>A</u> number, B varchar(20), C varchar(20))

4.1 Question 1

Vous referez les mêmes calculs que lors de l'exercice 2 mais cette fois-ci pour une table ABC 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 (s'appliquant à l'attribut A).

```
SET TRANSACTION READ ONLY NAME 'TransactionUn';
SET TRANSACTION READ WRITE;
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET TRANSACTION USE ROLLBACK SEGMENT
    some_rollback_segment;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
ALTER SESSION SET ISOLATION_LEVEL = SERIALIZABLE;
(session level)
```

Listing 2: Transaction

Insertion : Algorithme d'ajout

1. Trouver dans l'arbre la feuille où l'élément pourrait être ajouté.

2. Si le noeud contient moins de valeurs que le nombre maximum autorisé par noeud, alors ajouter l'élément en respectant le tri

3. Sinon, la feuille est alors éclatée :

(a) L'élément médian est choisi (nouveau père du sous arbre) parmi tous les éléments présents y compris le nouveau: élément médian = (k+1)/2 ème.

(b) Les valeurs < au médian -> fils gauche, et les valeurs > -> fils droit.

(c) L'élément médian (père du sous-arbre) est ajouté au noeud parent .

Un nouvel éclatement peut alors en résulter (continuer ainsi jusqu'à la racine)

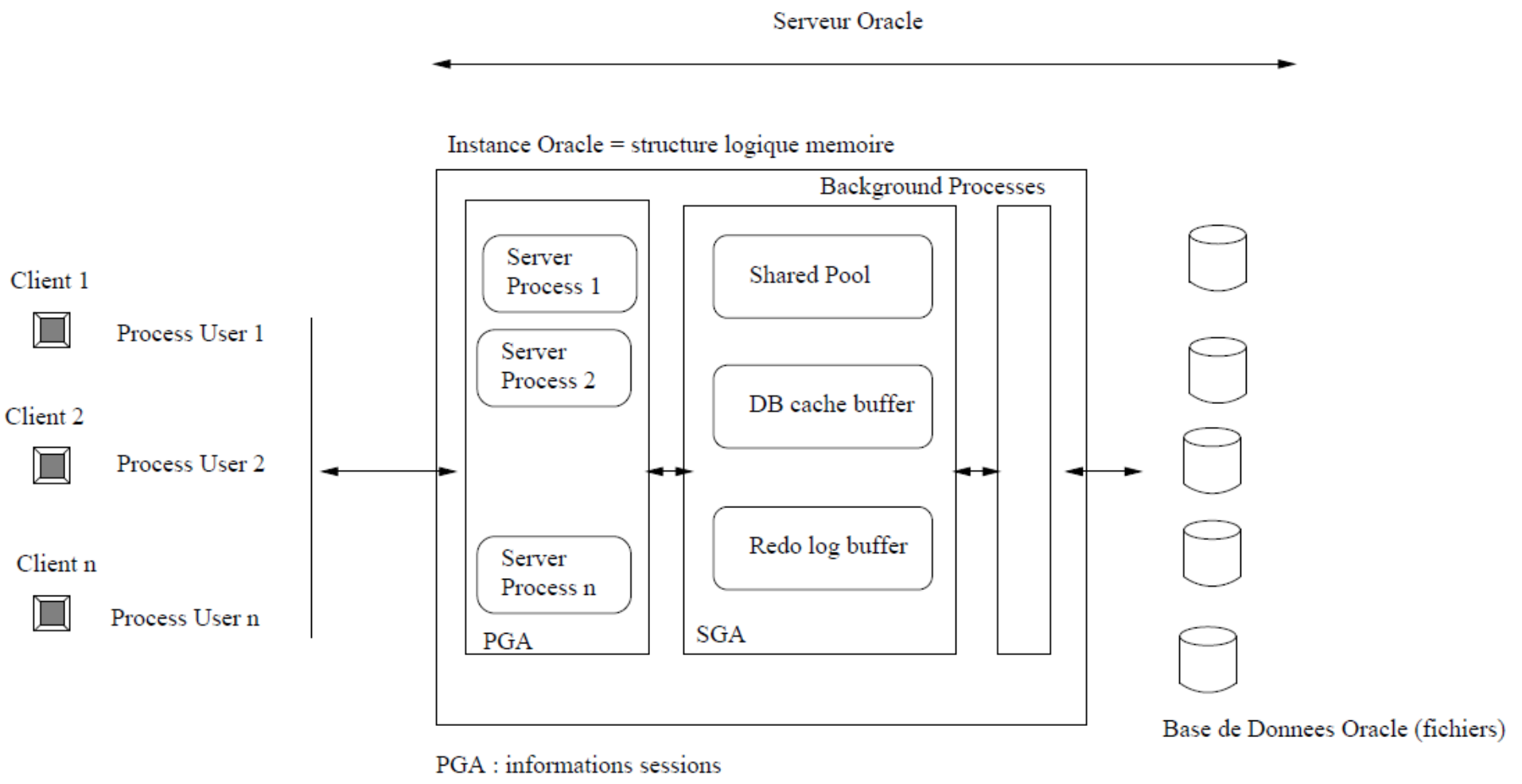

Figure: Rappel Architecture Oracle