

Bases de données Optimisation - Tuning

Michèle Raphalen (UBS)
V1.2 - septembre 2003



Bibliographie (1)

- Documentation Oracle Administrator's Guide
- Oracle 8
R. Chapuis, Editions Dunes & Laser, 1999
- Oracle Essentials (Oracle 8 & Oracle 8i)
R. Greenwald, R. Stackowiak, J. Stern, O'Reilly, 1999
- Oracle performance tuning and optimization
E. Whalen, SAMS Publishing, 1996
- Oracle 8i sous Linux : guide de l'administrateur et du développeur
G. Briard, Eyrolles, 2000

Bibliographie (2)

- Oracle et le Web
D. Deléglise, Eyrolles, 1998
- Le client-serveur
G. et O. Gardarin, Eyrolles, 1996
- Objet-relational sous Oracle 8
C. Soutou, Eyrolles, 1999
- Oracle 7 : Langages, Architecture, Administration
A. Abdellatif, M. Limane, A. Zeroual, Eyrolles, 1995

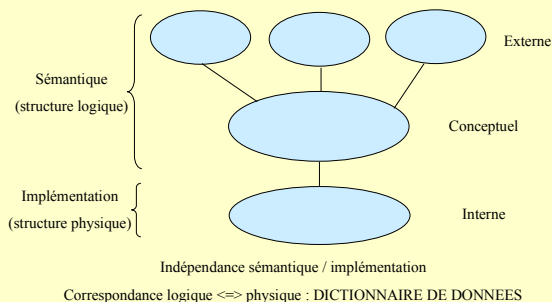


Plan

- Organisation logique / Organisation physique d'une base
- Optimisation / Tuning



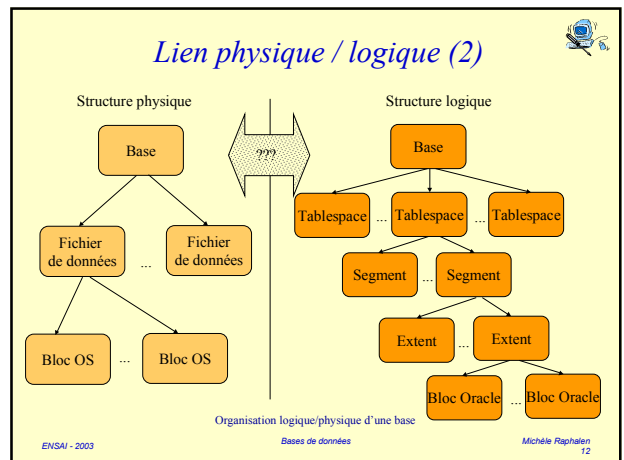
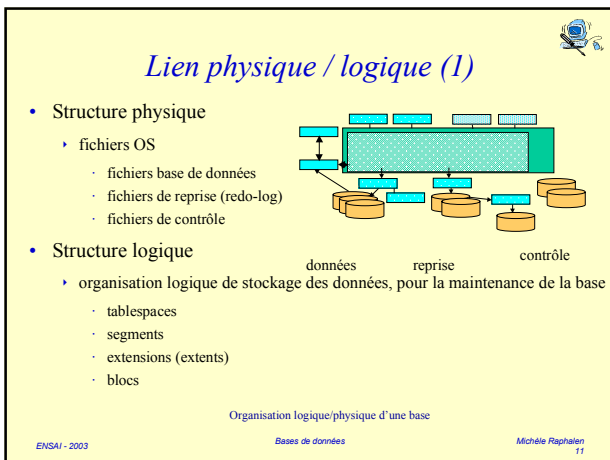
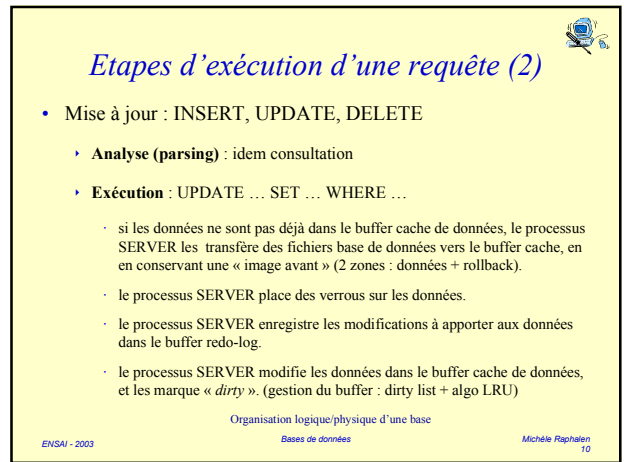
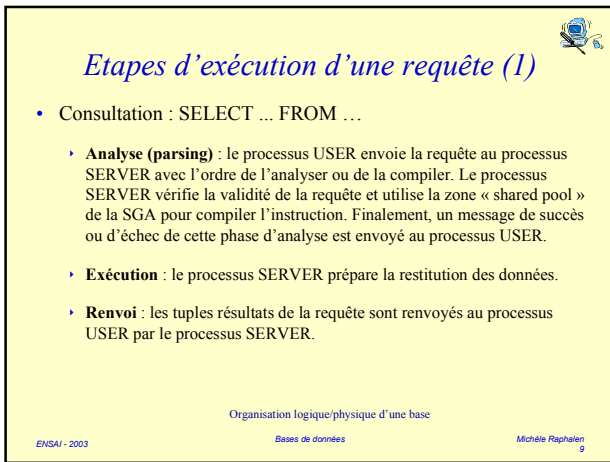
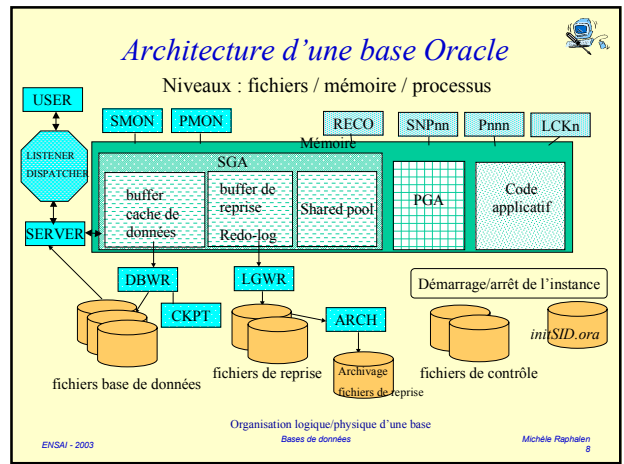
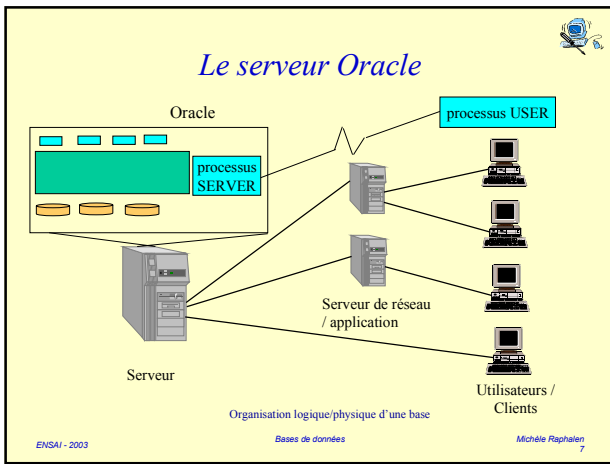
Architecture ANSI/SPARC



Dictionnaire

- Créé en même temps que la base
- Propriété de l'utilisateur SYS, accessible en lecture par l'utilisateur SYSTEM
- Mis à jour par les ordres du LDD
- Ensemble de tables/vues contenant toutes les informations sur la base
 - USER_
 - ALL_
 - DBA_
 - V§
- Associé au **tablespace** SYSTEM





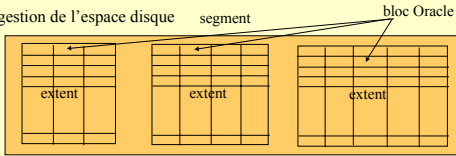
Lien physique / logique (3a)



- tablespace
 - un tablespace appartient à une base et une seule
 - un tablespace sert à regrouper un ensemble d'objets logiques
 - tables, clusters, index, segments de rollback, segments temporaires
 - à un tablespace sont associés un ou plusieurs fichiers de données, un fichier est associé à un seul tablespace

segments + extents + blocs

- gestion de l'espace disque



Organisation logique/physique d'une base

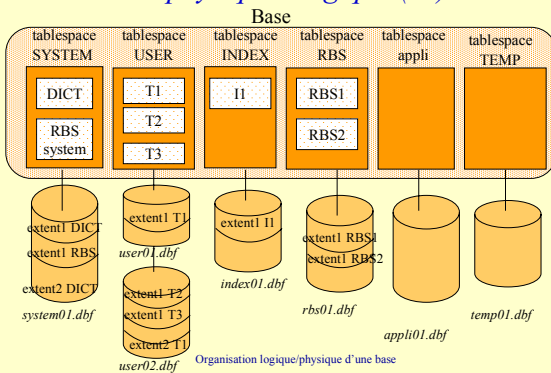
Lien physique / logique (3b)



- segment
 - un segment est un espace alloué pour chaque objet logique
 - segments de données (tables, clusters, ...), segments d'index, segments de rollback, segments temporaires
 - un segment est composé d'extensions (au minimum 1)
- extension
 - une extension est composée d'un ensemble de blocs Oracle contigus
- bloc Oracle
 - un bloc Oracle définit le niveau de granularité le plus fin.
 - un bloc Oracle est la plus petite unité d'entrée/sortie du SGBD. Il est composé de blocs OS
 - taille d'un bloc Oracle : DB_BLOCK_SIZE

Organisation logique/physique d'une base

Lien physique / logique (3c)



Organisation logique/physique d'une base

Tablespace (1)



tablespace SYSTEM

- obligatoire à la création de la base
- associé au 1er fichier de la base
- contient
 - le dictionnaire,
 - le RBS SYSTEM,
 - les packages, procédures/fonctions, triggers stockés de la base
- ne doit pas contenir les données des utilisateurs

Organisation logique/physique d'une base

Tablespace (2)



tablespace NON SYSTEM

- contrôle de l'espace alloué pour les objets de la base
- distribution des données sur différents volumes
 - amélioration des performances, réduction des contentions disques
- gestion de quotas pour les utilisateurs
- sauvegarde, restauration partielle des données
 - tablespace = la plus petite unité de sauvegarde
- possibilité de rendre inaccessibles des parties de la base
 - tablespace ONLINE/OFFLINE

Organisation logique/physique d'une base

Tablespace (3)



Création d'un tablespace NON SYSTEM

```
CREATE TABLESPACE USER
DATAFILE '/.../user01.dbf' SIZE 200M
DEFAULT STORAGE
( INITIAL 50K NEXT 50K
MINEXTENTS 1
MAXEXTENTS 20|UNLIMITED
PCTINCREASE 0
)
[PERMANENT|TEMPORARY]
[ONLINE|OFFLINE]
```

Segment (1)



Segment : ensemble d'extensions (extents) alloués à un objet

- différents types de segments
 - segment de démarrage (bootstrap)
 - segments de données
 - segments d'index
 - segments temporaires
 - segments de rollback
- nombre et taille des extensions définis par
 - la clause STORAGE du tablespace d'appartenance
 - la clause STORAGE de l'objet

Organisation logique/physique d'une base

Bases de données

Michèle Raphaelen
19

ENSAI - 2003

Segment (2)



segment de démarrage (bootstrap)

- utilisé une seule fois, à la création de la base
 - script sql.bsq
- créé dans le tablespace SYSTEM
- sert à l'initialisation du dictionnaire

Organisation logique/physique d'une base

Bases de données

Michèle Raphaelen
20

ENSAI - 2003

Segment (3a)



segments de données

- stockage du dictionnaire
- stockage des données des utilisateurs
 - tables

```
CREATE TABLE uneTable (...)  
[STORAGE (...)]  
[TABLESPACE unTS]
```

⇒ un segment unique pour la table uneTable

Organisation logique/physique d'une base

Bases de données

Michèle Raphaelen
21

ENSAI - 2003

Segment (3b)



segments de données

- clusters (*clusters à index, clusters à hashage*)
employé (**noEmp**, nomEmp, qualifEmp, serviceEmp)
service (**noService**, nomService, locService)

```
CREATE CLUSTER personnel (numService NUMBER)  
[SIZE uneTaille]  
[HASHKEYS unEntier]  
[STORAGE (...)]  
[TABLESPACE unTS]  
  
CREATE INDEX idx_personnel ON CLUSTER personnel  
  
CREATE TABLE employé (...) CLUSTER personnel (serviceEmp)  
  
CREATE TABLE service (...) CLUSTER personnel (noService)
```

⇒ un segment unique pour le cluster personnel

Organisation logique/physique d'une base

Bases de données

Michèle Raphaelen
22

ENSAI - 2003

Segment (3c)



segments de données

- tables partitionnées

```
vol (noVol, dateVol, pilote, ...)  
CREATE TABLE vol (...)  
PARTITION BY RANGE (dateVol)  
(PARTITION P1 VALUES LESS THAN (' 01-JAN-1999 '))  
[STORAGE (...)]  
TABLESPACE unTS1  
(PARTITION P2 VALUES LESS THAN (' 01-JAN-2000 '))  
[STORAGE (...)]  
TABLESPACE unTS2
```

⇒ un segment pour chacune des partitions

Organisation logique/physique d'une base

Bases de données

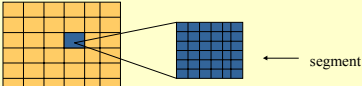
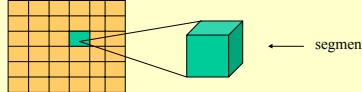
Michèle Raphaelen
23

ENSAI - 2003

Segment (3d)



segments de données

- tables imbriquées (*nested tables*)
- LOB

Organisation logique/physique d'une base

Bases de données

Michèle Raphaelen
24

ENSAI - 2003

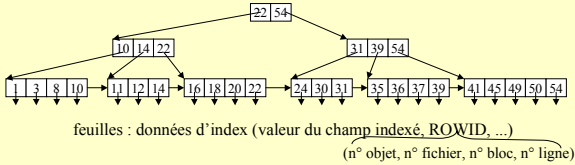
Segment (4a)

segments d'index

stockage des données d'index séparé du stockage des tuples

- index B-arbre

```
CREATE INDEX unIndex ON uneTable (listeAttributs)
[STORAGE (...)]
[TABLESPACE unTS]
```



Organisation logique/physique d'une base

Segment (4b)

segments d'index

- index Bitmap

```
CREATE INDEX BITMAP unIndex ON uneTable (listeAttributs)
[STORAGE (...)]
[TABLESPACE unTS]
```

individu (num, nom, ..., situationMaritale, ...)

(célibataire (0), marié (1), veuf (2), divorcé (3))

situationM...

1	0			
2	3			
3	1			
n	2			

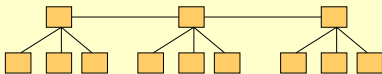
1	1	0	0	0
2	0	0	0	1
3	0	1	0	0
n	0	0	1	0

Organisation logique/physique d'une base

Segment (4c)

segments d'index

- index partitionné
 - cas des tables partitionnées
 - index local
 - index global



Organisation logique/physique d'une base

Segment (5)

segments temporaires

- utilisés pour le traitement des requêtes entraînant des tris
 - SELECT ... ORDER BY
 - SELECT ... GROUP BY
 - CREATE INDEX ...
 - opérateurs ensemblistes

- créés et supprimés automatiquement

NB : prévoir un tablespace pour les segments temporaires et les orienter vers ce tablespace.

Organisation logique/physique d'une base

Segment (6a)

segments de rollback

```
CREATE [PUBLIC] ROLLBACK SEGMENT unRBS
[STORAGE (... [OPTIMAL uneTaille])]
[TABLESPACE unTS]
```

```
ALTER ROLLBACK SEGMENT unRBS ONLINE
```

- utilisés dans les mécanismes transactionnels
 - annulation de transaction (ROLLBACK)
 - lecture consistante
- contiennent les images des données avant modifications
 - rollback entry = {n° fichier, identifiant du bloc contenant les données modifiées, bloc dans son état initial, ...}

Organisation logique/physique d'une base

Segment (6b)

segments de rollback

- segment de rollback SYSTEM
 - créé dans le tablespace SYSTEM à la création de la base
 - utilisé uniquement pour les objets du TS SYSTEM (dictionnaire)
- segments de rollback NON SYSTEM
 - nécessaires pour effectuer les transactions sur les données des utilisateurs
 - prévoir un tablespace dédié à ces segments

Organisation logique/physique d'une base

Segment (6c) segments de rollback



- une transaction est affectée à un segment de rollback et un seul
- un segment de rollback contient plusieurs transactions
 - recommandation : un RBS pour 4 transactions actives en moyenne
- les espaces alloués sont libérés
 - en fin de transaction (COMMIT ou ROLLBACK)
 - en cas de processus défaillant
- utilisation circulaire des segments
 - un extent ne peut être utilisé que si son premier bloc est libre.
 - un segment doit avoir au moins 2 extensions, toutes les extensions doivent être de même taille

Organisation logique/physique d'une base

Segment (6d) segments de rollback



- la charge est répartie sur tous les segments actifs
- si un segment atteint son maximum d'extensions, il y a ROLLBACK des transactions qui lui ont été affectées
- possibilité d'orienter une transaction vers un segment

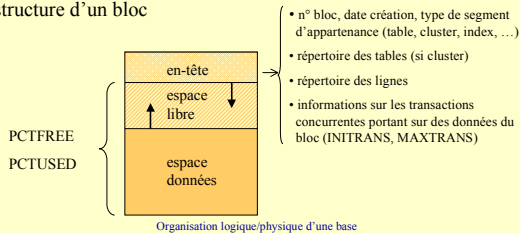

```
CREATE ROLLBACK SEGMENT unRBS ...
SET TRANSACTION USE ROLLBACK SEGMENT unRBS
... 'transaction lourde'
DROP ROLLBACK SEGMENT unRBS
```

Organisation logique/physique d'une base

Bloc (1)



- unité de transfert d'informations entre mémoire centrale (SGA) et disque
 - 1 bloc = la plus petite unité d'E/S
 - taille d'un bloc : DB_BLOCK_SIZE
- structure d'un bloc

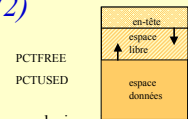


Organisation logique/physique d'une base

Bloc (2)



- espace des données
 - contient les données du bloc
 - une ligne de table peut être répartie sur plusieurs blocs (enimages)
 - taille d'une ligne > espace restant disponible dans le bloc
- espace libre
 - utilisé comme zone de débordement dans les cas de mises à jour
- paramètres de contrôle : PCTFREE, PCTUSED
 - spécifiés au niveau des segments
 - défaut : PCTFREE=10, PCTUSED = 40

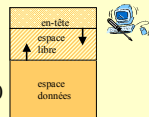


Organisation logique/physique d'une base

Bloc (3)



Exemple
PCTFREE = 20, PCTUSED = 40



- des lignes peuvent être insérées dans le bloc jusqu'à atteindre une limite inférieure d'espace libre égale à 20%
- l'espace libre peut être utilisé pour des mises à jour (UPDATES) entraînant l'allongement de lignes
- aucune nouvelle insertion ne pourra se faire dans le bloc tant que le pourcentage d'espace utilisé n'est pas inférieur à 40%

Organisation logique/physique d'une base

Plan



- Organisation logique/physique d'une base
- Optimisation / Tuning

Organisation logique/physique d'une base

Optimisation



Plusieurs niveaux d'optimisation

- conception optimisée des applications
 - choix des index
 - choix des clusters
 - introduction de redondance
 - partitionnement de tables
 - stockage approprié des objets de la base
 - optimisation des requêtes
- gestion de l'espace mémoire
 - réduction des défauts mémoire
- réduction des contentions
 - niveau E/S disques
 - niveau rollback segments

Optimisation / Tuning

Bases de données

Michèle Raphaelen
37

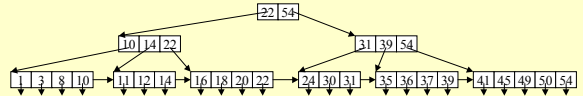
ENSAI - 2003

Optimisation des applications : index (1)



objectif : réduction du nombre d'accès pour retrouver les données.

- organisation en B-arbre
 - chemins de longueur identique de la racine vers n'importe quelle feuille
 - taux de remplissage d'au moins 50% garanti pour chaque nœud



Un arbre équilibré (B-arbre) d'ordre m est un arbre où

- toutes les feuilles sont au même niveau
- tout nœud non feuille a un nombre NF de fils : $m+1 \leq NF \leq 2*m+1$ exceptée la racine qui a un nombre de fils : $0 \leq NF \leq 2*m+1$

Optimisation / Tuning

Bases de données

Michèle Raphaelen
38

ENSAI - 2003

Optimisation des applications : index (2)



- technique d'accélération d'accès

CREATE INDEX idx_... **ON** <<table>> <<cluster>> (liste d'attributs)

- requêtes non modifiées par la création ou la suppression d'index
- évite le balayage complet de l'objet lors de la recherche de tuples satisfaisant une condition sur les attributs d'index
 - T (A, B, C)
 - SELECT * FROM T WHERE P(A)
 - A non indexé : toute la table est scrutée, bloc par bloc, tuple par tuple
 - A indexé : parcours de l'index qui renvoie les ROWID des tuples tels que P(A)
- clés primaires génératrices d'index (B+)

Optimisation / Tuning

Bases de données

Michèle Raphaelen
39

ENSAI - 2003

Optimisation des applications : index (3)



- pertinence de l'indexation

- cas de l'intégrité référentielle

T1 (A1, B1, ...); T2 (A2, B2, ...)

T1 * T2 : opération fréquente (T1.A1=T2.B2)

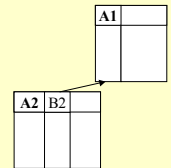
⇒ créer un index sur B2

- attributs servant fréquemment de critère de recherche

T (A, B, ...) et SELECT * FROM T WHERE P(B)

P(B) ::= {B = .. | B <= ... | B >= ... }

⇒ créer un index sur B



Optimisation / Tuning

Bases de données

Michèle Raphaelen
40

ENSAI - 2003

Optimisation des applications : index (4)



- non pertinence de l'indexation

- attributs présentant peu de valeurs distinctes
 - utiliser des index bitmap
- attributs sujets à nombreuses modifications
 - coût de l'équilibrage de l'arbre

- cas de non utilisation d'index T (... , I, ...) I indexé

SELECT * FROM T WHERE I LIKE ' %... '

SELECT * FROM T WHERE fonction (I) ...

SELECT * FROM T WHERE I IS [NOT] NULL

SELECT * FROM T WHERE I != ...

Optimisation / Tuning

Bases de données

Michèle Raphaelen
41

ENSAI - 2003

Optimisation des applications : index (5)



- inconvénients

- occupation d'espace
- possibilité de ralentissement des recherches
 - cas d'attributs présentant peu de valeurs
- pénalisation lors des opérations de mise à jour
 - INSERT, DELETE : équilibrage de tous les B-arbres associés à la table
 - DELETE FROM T WHERE I = ... avec un index sur I
 - recherche des tuples : utilité de l'index
 - suppression : index pénalisant , modification de B-arbre
 - INSERT INTO T ...
 - index pénalisant
 - UPDATE : seul le B-arbre concerné est rééquilibré
- NB : ne pas utiliser d'index en cas de saisie massive de tuples (le désactiver si nécessaire)
 - créer ou réactiver l'index après la saisie ⇒ gain de temps + gain d'espace

Optimisation / Tuning

Bases de données

Michèle Raphaelen
42

ENSAI - 2003

Optimisation des applications : index (6)

client (**refClient**, nomClient, adrClient, ...) commande (**numCom**, refClient, ...)

- select nomClient from client
 - accès séquentiel
- select nomClient from client where refClient = ...
 - accès par l'index
- select adrClient from client where nomClient = ' ... '
 - accès séquentiel, si pas d'index sur nomClient

Optimisation / Tuning

Bases de données

Michèle Raphaelen
43

Optimisation des applications : index (7a)

client (**refClient**, nomClient, adrClient, ...) commande (**numCom**, refClient, ...)

- select * from client, commande
where client.refClient = commande.refClient and adrClient = ' Vannes '

utilisation de l'index client.refClient

pour chaque tuple tCom de commande faire /* accès séquentiel */
accéder au tuple tCli de client tel que client.refClient = tCom.refClient /* accès indexé */
si tCli.adrCli = ' Vannes ' alors <<résultat>> fsi
fpour

Optimisation / Tuning

Bases de données

Michèle Raphaelen
44

Optimisation des applications : index (7b)

client (**refClient**, nomClient, adrClient, ...) commande (**numCom**, refClient, ...)

- select * from client, commande
where client.refClient = commande.refClient and adrClient = ' Vannes '

index sur client.adrClient et commande.refClient

accéder aux tuples de client tels que adrCli = ' Vannes ' ; /* accès indexé */
pour chaque tuple tCli de client vannetais faire
accéder aux tuples tCom de commande tel que tCom.refClient = client.refClient /* accès indexé */
pour chaque tuple tCom faire
<<résultat>>
fpour
fpour

Optimisation / Tuning

Bases de données

Michèle Raphaelen
45

Optimisation des applications : clusters (1)

- technique d'accélération d'accès pour les jointures

client (**refClient**, nomClient, adrClient, ...)

commande (**numCom**, refClient, totCom)

```
CREATE CLUSTER comCli (refClient NUMBER)
[SIZE uneTaille]
[HASHKEYS unEntier]
[STORAGE (...)]
[TABLESPACE unTS]
```

```
CREATE INDEX idx_comcli ON CLUSTER comCli
```

```
CREATE TABLE client (...) CLUSTER comCli (refClient)
```

```
CREATE TABLE commande (...) CLUSTER comCli (refClient)
```

- requêtes non modifiées par la création ou la suppression de cluster

Optimisation / Tuning

Bases de données

Michèle Raphaelen
46

Optimisation des applications : clusters (2)

- proximité physique des tuples

client (**refClient**, nomClient, adrClient, ...)

refClient	nomClient	adrClient	...
10	Legrand	Vannes	
11	Lepetit	Lyon	
12	Lemaigre	Pau	

comCli

refClient	nomClient	adrClient	...
numCom	totCom		
10	Legrand	Vannes	
	1	125	
	3	50	
	4	600	
11	Lepetit	Lyon	
	2	238	
12	Lemaigre	Pau	

commande (**numCom**, refClient, totCom)

numCom	refClient	totCom
1	10	125
2	11	238
3	10	50
4	10	600

Optimisation / Tuning

Bases de données

Michèle Raphaelen
47

Optimisation des applications : clusters (3)

- inconvénients

- un sens de parcours privilégié (clé de cluster)

SELECT ... WHERE refClient = 10 ; **efficace**

SELECT ... WHERE numCom = 4 ; **peu efficace**

- risque d'occupation de place inutile

- répartition non homogène des données
- et zone fixe importante

refClient	nomClient	adrClient	...
numCom	totCom		
10	Legrand	Vannes	
	1	125	
	3	50	
	4	600	
11	Lepetit	Lyon	
	2	238	
12	Lemaigre	Pau	

NB : ne pas utiliser de cluster en cas de modification fréquente de la clé de cluster

Optimisation / Tuning

Bases de données

Michèle Raphaelen
48

Optimisation des applications : redondance calculée (1)



- introduction volontaire de redondance pour favoriser certaines requêtes d'interrogation

client (**refClient**, nomClient, adrClient, ...)
 article (**refArticle**, nomArticle, prixArticle, ...)
 commande (**refCom**, refClient, ...)
 lignecom (**refCom**, **refArticle**, qteCom, ...)

Tables en 3NF

- dénormalisation
- vues matérialisées
- stockage de valeurs calculables

NB : techniques utilisées le plus souvent dans les modèles décisionnels - OLAP
entrepôts de données (peu de mises à jour)

Optimisation / Tuning

Bases de données

Michèle Raphaelen
49

ENSAI - 2003

Optimisation des applications : redondance calculée (2a)



- dénormalisation

(R) : nom et adresse des clients ayant commandé des huîtres

```
SELECT nomClient, adrClient
FROM client, article, commande, lignecom
WHERE client.refClient = commande.refClient
AND commande.refCom = lignecom.refCom
AND lignecom.refArticle = article.refArticle
AND article.nomArticle = ' huîtres ' ;
```

client (**refClient**, nomClient, adrClient, ...)
 article (**refArticle**, nomArticle, prixArticle, ...)
 commande (**refCom**, refClient, ...)
 lignecom (**refCom**, **refArticle**, qteCom, ...)

Plan d'exécution : recherche de la référence des huîtres dans article,
 recherche des numéros de commande dans lignecom,
 recherche des références des clients dans commande,
 recherche des noms et adresses des clients dans client.

Solution : lignecom (refCom, refArticle, qteCom, nomArticle, refClient, nomClient, adrClient)

Optimisation / Tuning

Bases de données

Michèle Raphaelen
50

ENSAI - 2003

Optimisation des applications : redondance calculée (2b)



- inconvénients de la dénormalisation
 - stockage coûteux en espace disque
 - risques d'incohérence en cas de mise à jour
 - introduction de redondance
 - influence sur les programmes d'application
 - structure des objets modifiées, contrairement au cas des index ou clusters

NB : Solution à envisager en dernier ressort

Optimisation / Tuning

Bases de données

Michèle Raphaelen
51

ENSAI - 2003

Optimisation des applications : redondance calculée (3)



- vues matérialisées
 - stockage de vues résultats de requêtes complexes
 - utilisé dans les entrepôts
 - préférable à la dénormalisation
 - stockage de valeurs calculables
 - stockage dans les tables de valeurs résultats de requêtes
- (R) : établir la facture des clients
- client (**refClient**, nomClient, adrClient, factuClient, ...)
- factuClient mis à jour automatiquement, au moyen d'un trigger

attribut calculé

Optimisation / Tuning

Bases de données

Michèle Raphaelen
52

ENSAI - 2003

Optimisation des applications : partitionnement des tables (1)



- système réparti
 - localisation des données sur les sites qui les utilisent le plus
- système centralisé
 - découpage en fonction des traitements
 - isolation d'attributs peu utilisés / beaucoup utilisés
 - isolation de tuples peu utilisés / beaucoup utilisés
- partitionnement horizontal / vertical / mixte

Optimisation / Tuning

Bases de données

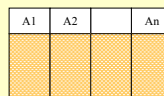
Michèle Raphaelen
53

ENSAI - 2003

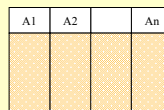
Optimisation des applications : partitionnement des tables (2)



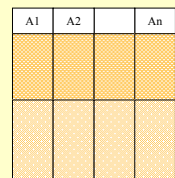
- partitionnement horizontal



T1



T2



T = T1 U T2

Optimisation / Tuning

Bases de données

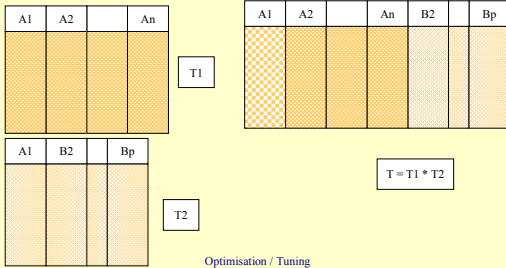
Michèle Raphaelen
54

ENSAI - 2003

Optimisation des applications : partitionnement des tables (3)



- partitionnement vertical



ENSAI - 2003

Bases de données

Michèle Raphaelen 56

Optimisation des applications : stockage approprié des objets



- paramétrage d'un bloc Oracle ($2^n K$, $2^{n+1} K$, $2^{n+2} K$)
 - bases de données transactionnelles : `DB_BLOCK_SIZE = 2^n K`
 - bases de données décisionnelles : `DB_BLOCK_SIZE = 2^{n+2} K`
 - occupation d'un bloc
 - `PCTFREE`, `PCTUSED`
- stockage des données dans les tables
 - performances dégradées
 - au-delà de 4 extensions par segment
 - si trop de lignes chaînées dans les blocs

Optimisation / Tuning

Bases de données

ENSAI - 2003

Michèle Raphaelen 56

Optimisation des applications : à savoir ...



- clé (identifiant)
 - doit contenir un minimum d'attributs
 - séquence vs ensemble d'attributs
- choix judicieux des types
 - numérique si opérateurs arithmétiques
 - date si la donnée contient une date / heure
 - caractère sinon
 - CP : images, sons, videos, ...
- définition des attributs servant de critère
 - indexer les attributs sur lesquels portent les accès les plus courants
 - éviter les valeurs NULL sur les attributs indexés
- répartition des informations sur les tables
 - utilisation de tables de références
 - partitionnement de l'information
- « packager » le plus possible les traitements

Optimisation / Tuning

ENSAI - 2003

Bases de données

Michèle Raphaelen 57

Optimisation des applications : optimiseur : transformations automatiques (1)



- `<<attribut>> compareur <<expression de constantes>>` transformé en `<<attribut>> compareur <<constante>>`
- `<<attribut>> LIKE <<valeur exacte (sans % ou _)>>` transformé en `<<attribut>> = <<valeur exacte (sans % ou _)>>`
- `<<attribut>> IN (a1, a2, ..., an)` transformé en `<<attribut>> = a1 OR <<attribut>> = a2 OR ... <<attribut>> = an`
- `<<attribut>> compareur ANY (a1, a2, ..., an)` transformé en `<<attribut>> compareur a1 OR <<attribut>> compareur a2 OR ...`
- `<<attribut>> compareur ALL (a1, a2, ..., an)` transformé en `<<attribut>> compareur a1 AND <<attribut>> compareur a2 AND ...`
- `<<attribut>> BETWEEN a1 AND a2` transformé en `<<attribut>> >= a1 AND <<attribut>> <= a2`

Optimisation / Tuning

ENSAI - 2003

Bases de données

Michèle Raphaelen 58

Optimisation des applications : optimiseur : transformations automatiques (2)



- `<<att1>> compareur ANY (SELECT <<att2>> FROM ... WHERE ...)` transformé en `EXISTS (SELECT <<att2>> FROM ... WHERE ... AND <<att1>> compareur <<att2>>)`
- `<<att1>> compareur ALL (SELECT <<att2>> FROM ... WHERE ...)` transformé en `NOT EXISTS (SELECT <<att2>> FROM ... WHERE ... AND <<att1>> négationCompareur <<att2>>)`
- `NOT <<att1>> compareur <<valeur>> | <<requête>>` transformé en `<<att1>> négationCompareur <<valeur>> | <<requête>>`
- une requête avec prédicats contenant des OR est transformée en une UNION (ALL) de requêtes statistiquement plus favorable
- les requêtes imbriquées sont transformées en une requête de jointure

ATTENTION aux conversions implicites de type (inhibition de l'index)

Optimisation / Tuning

ENSAI - 2003

Bases de données

Michèle Raphaelen 59

Optimisation des applications : Etapes d'exécution d'une requête SQL (1)



- DECLARE : déclaration de curseur
 - zone de travail, identifiée par un numéro, contenant des informations sur l'exécution de la requête
- PARSE : analyse de l'ordre SQL
 - analyse syntaxique, remplacement des synonymes et vues
 - chargement de la traduction dans la SHARED SQL AREA
 - vérification des autorisations d'accès aux objets mis en œuvre
 - détermination du chemin d'accès aux données
 - calcul du plan d'exécution de la requête : *optimiseur de requête*
 - affectation des ressources (avec verrouillage éventuel)
- BIND : édition de liens
 - remplacement des paramètres (clauses WHERE / HAVING) par les valeurs
- EXECUTE : exécution de la requête par le noyau
 - si SELECT ... : génération des lignes résultats au niveau du noyau

Optimisation / Tuning

ENSAI - 2003

Bases de données

Michèle Raphaelen 60

Optimisation des applications : Etapes d'exécution d'une requête SQL (2)



- DESCRIBE : génération des résultats
 - détermination de la structure des données résultats
- DEFINE : définition des sorties
 - détermination des formats de sortie
- FETCH : distribution du résultat
 - transfert ligne à ligne du résultat dans le curseur

Requête
SELECT

- CLOSE : fermeture du curseur
 - libération des ressources

Optimisation / Tuning

Bases de données

Michèle Raphaelen
61

ENSAI - 2003

Optimisation des applications : Optimiseur (1)



- Détermination du plan d'exécution de la requête
 - stratégie optimale d'exécution des opérations composant la requête
 - étapes d'accès et recherche des données dans la base
 - mise en forme des données pour des étapes ultérieures
 - but
 - obtention du meilleur débit
 - ensemble des tuples résultats obtenu en un temps minimum
 - ou
 - obtention du meilleur temps de réponse
 - premier tuple résultat obtenu en un délai minimum

Optimisation / Tuning

Bases de données

Michèle Raphaelen
62

ENSAI - 2003

Optimisation des applications : Optimiseur (2)



- Stratégies
 - Optimisation basée sur des heuristiques (rule-based, mode statique)
 - respect d'un ensemble de règles enregistrées au niveau du SGBD
 - calcul du coût basé sur l'ordonnement des chemins d'accès conformément aux règles
 - dirigé par la syntaxe
 - Optimisation basée sur des statistiques (cost-based, mode statistique)
 - examen de toutes les possibilités de résolution et choix du « moindre coût »
 - calcul du coût basé sur le nombre de transferts de données entre mémoire secondaire et mémoire centrale (reads)
 - dirigé par les statistiques
 - Optimisation basée sur la syntaxe
 - exécution guidée par l'écriture de la requête
 - pas d'utilisation de fonctions de coût ou de statistiques
 - Compilation des plans d'exécution
 - plans compilés stockés pour être réutilisés

Optimisation / Tuning

Bases de données

Michèle Raphaelen
63

ENSAI - 2003

Optimisation des applications : optimiseur Oracle (1)



- Mode statique
 - Choix d'un plan d'exécution
 - réduction du coût d'accès aux données
 - basé sur
 - syntaxe SQL
 - écriture des prédicats dans les clauses WHERE
 - objets utilisés dans la requête
 - utilisation éventuelle d'index
 - techniques d'accès aux données
 - parcours complet de la table (FULL TABLE SCAN)
 - accès par ROWID (accès direct à 1 tuple)
 - parcours par index (INDEX SCAN)
 - indexé / cluster
 - hash scan / cluster

ATTENTION : veiller à l'utilisation judicieuse des attributs indexés !!!

Optimisation / Tuning

Bases de données

Michèle Raphaelen
64

ENSAI - 2003

Optimisation des applications : optimiseur Oracle (2)



- Mode statistique
 - Choix d'un plan d'exécution
 - réduction du temps de traitement de la requête
 - basé sur la répartition des données
 - volume
 - nombre de valeurs distinctes dans chaque colonne
 - distribution de la clé
 - distribution des index
 - occupation des blocs, ...
 - statistiques stockées dans le dictionnaire
 - DBA_TABLES
 - DBA_TAB_COLUMNS
 - DBA_INDEXES
 - DBA_CLUSTERS
- ATTENTION : pour utiliser ce mode d'optimisation, les statistiques doivent être mises à jour régulièrement par
- la commande `ANALYZE TABLE|CLUSTER <<clause>> STATISTICS`
 - ou - la procédure `dbms_utility.analyze_schema (<<paramètres>>)`

Optimisation / Tuning

Bases de données

Michèle Raphaelen
65

ENSAI - 2003

Optimisation des applications : optimiseur Oracle (3)



- Positionnement du mode de l'optimiseur
 - Niveau instance : paramètre `OPTIMIZER_MODE` dans `init<<sid>>.ora`
 - CHOOSE : mode statique ou statistique
 - RULE : mode statique uniquement
 - FIRST_ROWS : mode statistique avec meilleur temps de réponse
 - ALL_ROWS : mode statistique avec meilleur débit
 - Niveau session : `ALTER SESSION SET OPTIMIZER_GOAL = <<valeur>>`
 - modification du mode de l'optimiseur en cours de session
 - mêmes valeurs que `OPTIMIZER_MODE`

Optimisation / Tuning

Bases de données

Michèle Raphaelen
66

ENSAI - 2003

Optimisation des applications : optimiseur Oracle (4)



- Positionnement du mode de l'optimiseur
 - Niveau requête : utilisation de directives 'HINTS'
 - force les options de l'optimiseur
 - CHOOSE : mode statique ou statistique
 - RULE : mode statique uniquement
 - FIRST_ROWS : mode statistique avec meilleur temps de réponse
 - ALL_ROWS : mode statistique avec meilleur débit
 - FULL (table) : parcours complet de la table
 - INDEX (index) : force l'utilisation de l'index
 - INDEX : calcule le coût pour chaque index disponible et utilise le « meilleur »
 - ROWID (table) : force l'accès par ROWID pour la table
 - ORDERD : effectue une jointure de table dans l'ordre spécifié par la requête
 - ...

Optimisation / Tuning

Bases de données

Michèle Raphaelen
67

ENSAI - 2003

Optimisation des applications : optimiseur Oracle (5)



- Positionnement du mode de l'optimiseur
 - Niveau requête : utilisation de directives 'HINTS'
 - client (**refClient**, nomClient, adrClient, ...) et adrClient indexé
 - article (**refArticle**, nomArticle, prixArticle, ...)

```
SELECT /* + FULL (tClient) */ nomClient, adrClient
FROM client tClient
WHERE adrClient = 'Vannes'
```

```
SELECT /* + FIRST_ROWS */ nomArticle
FROM article
WHERE prixArticle > 10 ;
```

ATTENTION : ne pas abuser des 'HINTS' !!!

Optimisation / Tuning

Bases de données

Michèle Raphaelen
68

ENSAI - 2003

Optimisation des applications : analyse et mesures (1)



- Génération de statistiques
 - commande ANALYZE
 - Mesures sur les objets de la base
 - taille de l'objet (nb lignes, nb blocs)
 - High water mark
 - taux de remplissage des blocs
 - taille moyenne d'une ligne
 - nombre de lignes chaînées
 - ...
- ATTENTION : calcul non dynamique des statistiques
résultats dans les vues : DBA_TABLES, DBA_TAB_COLUMNS

Optimisation / Tuning

Bases de données

Michèle Raphaelen
69

ENSAI - 2003

Optimisation des applications : analyse et mesures (2)



- Trace d'exécution des ordres SQL
 - Niveau instance
 - paramètres SQL_TRACE et TIMED_STATISTICS à TRUE dans init<<sid>>.ora
 - Niveau session : ALTER SESSION SET SQL_TRACE = TRUE
 - Exploitation des résultats : commande TKPROF <<fichier.trc>> <<ficRésultat>>
 - mesures pour chacune des phases PARSE, EXECUTE, FETCH
 - nombre d'exécutions de l'item (*count*)
 - nombre de blocs physiques lus (*disk*)
 - nombre de lignes ramenées (*rows*)
 - temps CPU et total (*elapsed*), en secondes
 - nombre de tampons utilisés pour les lectures consistantes (*query*)
 - Nombre de tampons courants utilisés (*current*)
- ATTENTION : la trace peut entrainer une dégradation des performances du serveur !!!

Optimisation / Tuning

Bases de données

Michèle Raphaelen
70

ENSAI - 2003

Optimisation des applications : analyse et mesures (3)



- Etude des plans d'exécution des requêtes
 - commande EXPLAIN PLAN
 - nécessite la création préalable de la table d'audit PLAN_TABLE
 - script \$ORACLE_HOME/rdbms/admin/utlxplan.sql
 - EXPLAIN PLAN
 - SET STATEMENT_ID = <<identificateur>>
 - FOR <<requête SQL>>
 - fournit un arbre d'exécution de la requête

Optimisation / Tuning

Bases de données

Michèle Raphaelen
71

ENSAI - 2003

Optimisation de la mémoire



- SGA
 - shared pool area
 - zone LC, zone DC
 - buffer de données
 - tables, clusters, index, ...
 - buffer de reprise
- minimiser les défauts de mémoire

Optimisation / Tuning

Bases de données

Michèle Raphaelen
72

ENSAI - 2003

Optimisation de la mémoire (1) shared pool area : LC



- contient les ordres SQL et blocs PL/SQL des utilisateurs
- gérée par un algorithme LRU
- objectifs
 - minimiser les phases d'analyses (parsing) ie avoir « le plus possible » d'ordres, procédures, ... en mémoire
- mesures (V\$LIBRARYCACHE)
 - pour chacun des items dans la zone LC (ordres SQL, procédures, ...)
 - PINS : nombre d'exécutions mettant en œuvre l'item
 - RELOADS : nombre de défauts mémoire pour chaque item
 - dus à la non-présence d'ordres, procédures, ... en mémoire
 - dus à des invalidations : ordres portant sur des objets à structure modifiée, procédures recompilées, ...
 - Pin-ratio : $\Sigma(\text{RELOADS}) / \Sigma(\text{PINS})$ doit être $< 1\%$

Optimisation / Tuning

Bases de données

Michèle Raphaelen
73

ENSAI - 2003

Optimisation de la mémoire (2) shared pool area : LC



- solutions
 - augmenter la valeur de SHARED_POOL_SIZE
 - fichier initSID.ora
 - utiliser des procédures génériques (packages, procédures/fonctions)
 - même texte pour la même requête
vendeur (refVendeur, nomVendeur, ageVendeur, locVendeur)
- ```
SELECT * FROM VENDEUR
WHERE ageVendeur > 20 AND locVendeur = ' Vannes ';
```
- ```
SELECT refVendeur, nomVendeur, ageVendeur, locVendeur FROM VENDEUR
WHERE locVendeur = ' Vannes ' AND ageVendeur > 20 ;
```
- référencer les mêmes objets par les mêmes noms

Optimisation / Tuning

Bases de données

Michèle Raphaelen
74

ENSAI - 2003

Optimisation de la mémoire (3) shared pool area : DC



- contient des objets du dictionnaire
- objectifs
 - minimiser les défauts de mémoire cache dictionnaire
- mesures (V\$ROWCACHE)
 - pour chacun des items dans la zone DC
 - GETS : nombre de requêtes mettant en œuvre l'item
 - GETMISSES : nombre de requêtes avec défauts mémoire pour chaque item
 - Get-ratio : $\Sigma(\text{GETMISSES}) / \Sigma(\text{GETS})$ doit être $< 15\%$
- solutions
 - augmenter la valeur de SHARED_POOL_SIZE

NB : Défauts mémoire DC très fréquents, relevés principalement au démarrage de l'instance.

Optimisation / Tuning

Bases de données

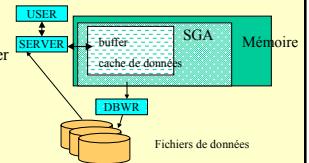
Michèle Raphaelen
75

ENSAI - 2003

Optimisation de la mémoire (4) buffer cache de données



- contient des blocs, copies des données des fichiers de données
 - taille : $\text{DB_BLOCK_BUFFERS} * \text{DB_BLOCK_SIZE}$
 - SERVER : fichier -> buffer
 - DBWR : buffer -> fichiers
 - deux listes de blocs dans le buffer
 - LRU
 - dirty list
- objectifs
 - minimiser les accès aux fichiers de données
 - le nombre de lectures sur disque doit être faible devant le nombre d'accès mémoire



Optimisation / Tuning

Bases de données

Michèle Raphaelen
76

ENSAI - 2003

Optimisation de la mémoire (5) buffer cache de données



- mesures (V\$SYSSTAT)
 - hit ratio : $1 - (\text{physical reads} / (\text{db block gets} + \text{consistent gets}))$ doit être $\geq 90\%$
 - db block gets : nombre d'accès aux blocs non RBS
 - consistent gets : nombre d'accès aux blocs RBS (consistance)
 - physical reads : nombre de lectures de blocs sur disque
 - db block gets + consistent gets : nombre total de demandes formulées pour avoir des données
- solutions
 - augmenter la valeur de DB_BLOCK_BUFFERS
 - utiliser de la mémoire cache pour les tables

Optimisation / Tuning

Bases de données

Michèle Raphaelen
77

ENSAI - 2003

Optimisation de la mémoire (6) buffer de reprise



- contient les modifications effectuées sur les données
 - redo entry : {bloc modifié, localisation de la modification dans le bloc, nouvelle valeur}
- buffer géré cycliquement
- mesures (V\$SYSSTAT)
 - key ratio : redo log space requests / redo entries doit être $\leq 1/5000$
 - redo log space requests :
 - UPDATE : LGWR a fini d'écrire dans le fichier de reprise écriture de la redo entry dans le buffer
 - INSERT : LGWR a fini d'écrire dans le fichier de reprise écriture de la redo entry dans le buffer
 - UPDATE : LGWR n'a pas fini d'écrire dans le fichier de reprise une 'redo log space request' + écriture de la redo entry dans le buffer

Optimisation / Tuning

Bases de données

Michèle Raphaelen
78

ENSAI - 2003

Optimisation de la mémoire (7) buffer de reprise



- solutions
 - augmenter la valeur de LOG_BUFFER
 - elle doit rester multiple de la taille d'un bloc OS

Optimisation / Tuning

Bases de données

Michèle Raphaelen
79

ENSAI - 2003

Contentions (1) E/S disques



- distribuer les E/S
 - séparer la localisation des fichiers de reprise et des fichiers de données
 - réserver le tablespace SYSTEM pour le dictionnaire uniquement
 - séparer les tables et les index (tablespaces différents)
 - partitionner les tables et distribuer les fragments
 - créer des tablespaces pour les rollback segments
 - stocker les TRES GROS objets dans leur propre tablespace
 - créer des tablespaces temporaires
- mesures (V\$FILESTAT)
 - par fichier de données
 - PHYRDS : nombre de lectures réalisées
 - PHYWRTS : nombre d'écritures

Optimisation / Tuning

Bases de données

Michèle Raphaelen
80

ENSAI - 2003

Contentions (2) segments de rollback



- utilité des segments de rollback
 - annulation de transactions
 - recouvrement après incident
 - lecture consistante
- rappels sur leur utilisation
 - contiennent les images des données avant modifications
 - rollback entry = {n° fichier, identifiant du bloc contenant les données modifiées, bloc dans son état initial, ...}
 - une transaction est affectée à un segment de rollback et un seul
 - un segment de rollback contient plusieurs transactions
 - recommandation : un RBS pour 4 transactions actives en moyenne

Optimisation / Tuning

Bases de données

Michèle Raphaelen
81

ENSAI - 2003

Contentions (3) segments de rollback



- utilisation circulaire des segments
 - une extension ne peut être utilisé que si son premier bloc est libre.
 - un segment doit avoir au moins 2 extensions, toutes les extensions doivent être de même taille
 - les espaces alloués sont libérés
 - en fin de transaction (COMMIT ou ROLLBACK)
 - en cas de processus défaillant
 - la charge est répartie sur tous les segments actifs
- en-tête (header) de segment de rollback
 - contient une table des transactions associées au segment

Optimisation / Tuning

Bases de données

Michèle Raphaelen
82

ENSAI - 2003

Contentions (4) segments de rollback



- mesures
 - V\$ROLLNAME
 - noms et nombre de RBS online
 - V\$ROLLSTAT
 - nombre d'attentes sur la table des transactions (header)
 - volume des transactions
 - V\$WAITSTAT
 - données cumulées sur les attentes (header et données) pour l'ensemble des segments de rollback (SYSTEM et non SYSTEM)
 - par classe de blocs (system undo header, system undo block, undo header, undo block), on compte le nombre d'attentes.
 - si le nombre d'attentes pour chaque classe est plus grand que 1% du total des demandes d'accès (select SUM (value) from V\$SYSSTAT WHERE name=' consistent gets '), il y a **contention**.

Optimisation / Tuning

Bases de données

Michèle Raphaelen
83

ENSAI - 2003

Contentions (5) segments de rollback



- solutions
 - création de nouveaux segments de rollback
 - affectation de transactions lourdes, consommatrices de segments, à des segments spécialement créés :
- ```
CREATE [PUBLIC] ROLLBACK SEGMENT unRBS ...
SET TRANSACTION USE ROLLBACK SEGMENT unRBS
... 'transaction lourde'
DROP ROLLBACK SEGMENT unRBS
```

Optimisation / Tuning

Bases de données

Michèle Raphaelen  
84

ENSAI - 2003