7

Optimiseur d'Oracle

OPTIMISEUR DE REQUÊTES

. Déterminer la stratégie optimale d'exécution des opérations élémentaires

. Stratégies :

-basée sur un ensemble de règles(RBO Rule-based Optimizer) (n'est plus utilisée à partir de ORACLE 11G)

-repose sur la syntaxe des ordres SQL

-Et l'indexation

- basée sur les coûts(CBO Cost-based Optimizer) (est le seul mode à partir de ORACLE 11G)

-repose sur des statistiques sur les objets de la base

-I/O, temps CPU, resources mémoire

OPTIMISEUR DE REQUÊTES (CBO – Cost-Based Optimizer)

Rôle

-Déterminer la meilleure stratégie pour exécuter un ordre SQL → performance d'une base de données

Résultat

Plan d'exécution : étapes dans l'exécution de l'ordre SQL

-Ordre dans lequel les tables sont accédées

-façon dont les tables sont accédées ou jointes

-utilisation ou non d'index

OPTIMISEUR DE REQUÊTES
(CBO – Cost-Based Optimizer)

1. Génère plusieurs plans d'exécution pour une requête
2. Estime le coût de chaque plan d'exécution généré sur la base
   - des I/O
   - du temps CPU estimé pour chaque opération
   - des ressources mémoire nécessaires
   - des informations statistiques sur les objets de la base (tables, index et vues)
   - des informations sur les performances du serveur
3. Choisit " un bon plan " et l'exécute

MODE DE L'OPTIMISEUR CBO(Cost-Based Optimizer)

Choix du plus faible coût avec pour objectif :

une meilleure capacité de traitement

(meilleur débit : mode ALL_ROWS (mode par défaut))

utilisation du minimum de ressources pour traiter toutes les lignes concernées par l'ordre SQL → pour les applications en Batch

un meilleur temps de réponse:mode FIRST_ROWS_n(n=1,10,100,1000)

(utilisation du minimum de ressources pour traiter les n premières lignes de l'ens. Résultat → pour les applications interactives)

Modification du mode :

```
alter session set optimizer_mode=FIRST_ROWS_1;
```

ou modification du paramètre optimizer_mode dans le fichier des paramètres d'utilisation init.ora

M.RAMBURRUN

STRATÉGIE DE RECHERCHE ADAPTATIVE

Technique pour limiter l'espace de recherche de plans d'exé.

1. Temps d'exécution d'une requête évalué à une seconde
   →ne pas dépenser 10 secondes pour l'optimisation

2. Temps d'exécution d'une requête évalué à plusieurs minutes ou heures →dépenser plusieurs secondes ou minutes dans la phase d'optimisation dans l'espoir de trouver un meilleur plan

EXEMPLE

Requête avec 5 tables

On a 5! = 120 combinaisons de jointures possibles
Pour chaque jointure, on a une douzaine de possibilités d'exécution basées sur des combinaisons variées :

1. **l'ordre de jointure des tables**
2. **l'utilisation d'index**
3. **méthodes d'accès**
4. **techniques de jointures**

Nombre total de plans d'exécution :
de l'ordre de plusieurs milliers

Statistique sur les tables, colonnes et index

OBTENIR DES STATISTIQUES (procédures du package DBMS_STATS (car analyse deprecated))

Pour donner au CBO le maximum d'infos. : il faut faire l'analyse de tous les objets sur lesquelles portent des interrogations

Statistiques sur tous les objets d'un schéma

dbms_stats.gather_schema_stats(ownername[,method_opt..])

Statistiques sur une table

dbms_stats.gather_table_stats(ownername,tabname[,method_opt..])

Statistiques sur les index

dbms_stats.gather_index_stats(ownername,indname[, …])

RÉSULTATS

Pour une table

. Nombre de lignes d'une table
. Nombre de blocs utilisés pour la table
. Nombre de blocs vides
. Taille moyenne d'une ligne…

Pour une colonne d'une table

. Nombre de valeurs distinctes de la colonne
. Valeur minimale
. Valeur maximale…

Pour un index

. Profondeur du B-tree
. Nombre de clefs distinctes…

RÉSULTATS ACCESSIBLES VIA LES VUES :

DBA_TABLES, DBA_TAB_COLUMNS

DBA_CLUSTERS

DBA_INDEXES, INDEX_STATS

DBA_TAB_PARTITIONS, DBA_IND_PARTITIONS, DBA_PART_COL_STATISTICS

INDEX_HISTOGRAM, DBA_HISTOGRAMS

EXEMPLES

Pour une table donnée

```
select table_name, num_rows, blocks, empty_blocks, avg_space
  from dba_tables
 where owner = user
    and table_name='WORKER';
```

Pour les colonnes d'une table

```
select substr(column_name,1,15)  column_name,
          substr(data_type,1,15) data_type, num_distinct,
          to_char(density,'0.99') density, num_buckets, num_nulls
  from dba_tab_columns
  where owner=user
    and  table_name='WORKER';
```

Pour un index

```
select index_name, uniqueness ,blev,leaf_blocks,distinct_keys,num_rows
 from dba_indexes
  where owner=user  and index_name='ILONGNAME';
```

Plans d'exécution

Un plan d'exécution est un ensemble d'opérations élémentaires à exécuter pour évaluer la réponse d'une requête. La génération du plan d'exécution finale d'une requête est effectuée par l'optimisateur, qui consiste à réécrire la requête sous forme canonique, et à réordonner les opérations élémentaires de telle façon à minimiser le temps d'évaluation [Gardarin, 2003].

OPÉRATIONS D'UN OPTIMISEUR POUR UN ORDRE SQL

1. Evaluation des expressions et conditions contenant des constantes

2. Transformation de l'ordre SQL
(transformation éventuelle des sous-requêtes, vues en jointure, Réécriture de la requête)

3. Choix de l'approche d'optimisation
(mode de l'optimiseur (RBO,CBO via ALL_ROWS ou FIRST_ROWS_n) , hints (select /*+ USE_NL*/ … , statistiques)

4. Choix des chemins d'accès et gestion de jointures

5. Choix de l'ordre des jointures

PLAN D'EXÉCUTION

3. Chemins d'accès et gestion de jointures

. Parcours séquentiel (TABLE ACCES FULL)

. Par accès direct par adresse (TABLE ACCES BY INDEX ROW-ID)

. Parcours de regroupement (JOIN CLUSTER SCAN)

. Parcours par hachage (JOIN HASH SCAN)

. Parcours d'index
(INDEX UNIQUE SCAN, INDEX RANGE SCAN)

OPÉRATIONS PHYSIQUES

. TABLE ACCESS

. INDEX

. SORT

. FILTER

. UNION

. INTERSECT

. MINUS

. AUTRES OPÉRATIONS LIÉES AUX ALGORITHMES DE JOINTURES

ALGORITHMES DE JOINTURE

- **Boucles imbriquées (NESTED LOOP)**
- **Tri/Fusion (SORT et MERGE)**
- **Jointure par hashage HASH JOIN**

NESTED LOOP

Boucles imbriquées (NESTED LOOP)

Lecture séquentielle de la source de données S1

Pour chaque ligne de S1
→ recherche dans la source S2
ou index de s2 puis accès S2

S1 est appelée driving table (ou table externe)
S2 est appelée table interne

NESTED LOOP

Utilisée par le CBO en mode FIRST_ROWS_n pour

- joindre de petits ensembles de données si la condition de jointure permet d'accéder efficacement à la table interne

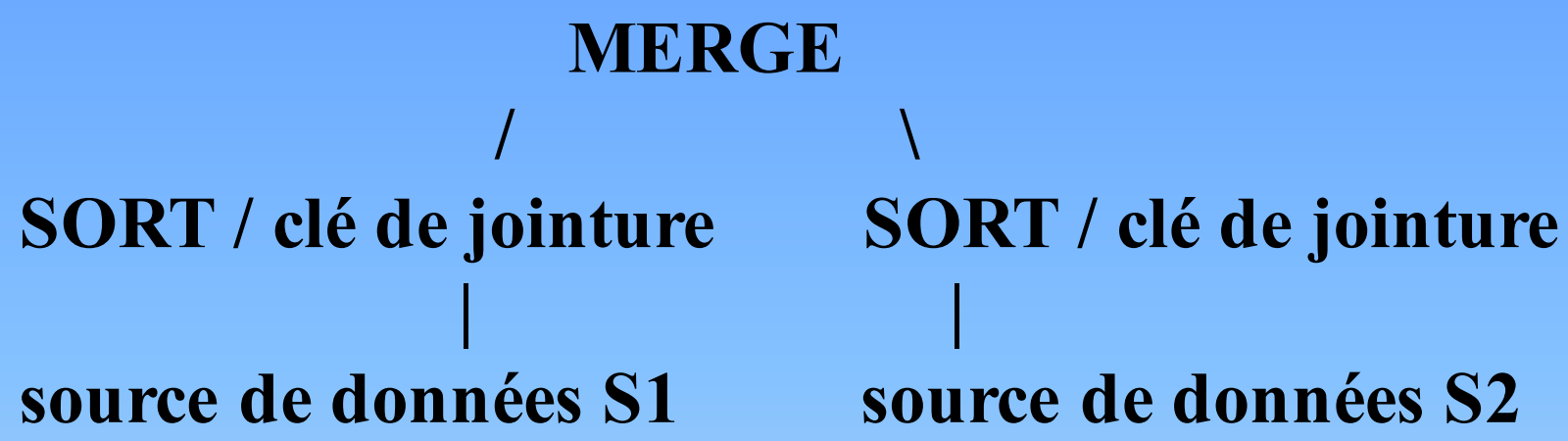
- et si un index au niveau de la table interne sur les attributs de jointure existe

La table interne doit être dépendante de la table maîtresse.

En mode ALL_ROWS, le CBO effectue une HASH JOIN pour une équi-jointure et une SORT MERGE JOIN sinon.

SORT & MERGE

Tri/Fusion (SORT et MERGE)



Utilisée par le CBO(quel que soit le mode) pour joindre de petits ensembles quand :

- la jointure n'est pas une équijointure ou alors des tris ont déjà été réalisés par d'autres opérations
- des index sont utilisables

HASH JOIN

Jointure par hashage HASH JOIN

Version de Nested Loop :

-Une des tables de données est choisie et utilisée pour construire une table de hashage sur la clé de jointure. (Chaque valeur est comparée avec les valeurs HASH de l'autre table

-Plus efficace en théorie que SORT/MERGE et NESTED_LOOP

Utilisée par le CBO si la jointure est une équijointure et si

-de grandes quantités de données sont à joindre

-il n'y a pas d'index

-il y a des index et le mode de l'optimiseur est ALL_ROWS