TP
Optimisation de requêtes

Si vous ressentez le besoin de lire la référence **ORACLE** : https://docs.oracle.com/en/database.

Conseils

1. Avant de vous connecter, créez un répertoire de travail pour cet enseignement et travaillez à partir de ce répertoire.
2. Utilisez un éditeur de texte pour concevoir vos requêtes puis effectuez des copier/coller dans l'interpréteur **ORACLE** (**sqlplus** ou **sqldeveloper**).
3. Une fois le script **SQL** au point, sauvegardez le fichier avec l'extension **sql** pour une réutilisation future ou un rendu.

Rendu du tp « Optimisation de Requêtes »

Par mail à noel.novelli@lis-lab.fr et « [ILD-GD] TP1 – Optimisation de requêtes » en sujet, un seul fichier **tp1_nom_OR.sql** (si vous êtes en binôme, mettre les deux noms) contenant tout votre script avec tous les commentaires qui me permettront de bien comprendre vos choix et votre démarche. Les mauvaises routes que vous aurez empruntées peuvent vous permettre d'expliquer vos choix.

Je vous conseille fortement de lire entièrement le sujet surtout l'annexe avant de commencer à répondre aux questions. Par exemple, une bonne utilisation de `AUTOTRACE` **vous sera bénéfique.**

Récupération des données

Pour illustrer l'optimisation de requêtes, nous allons travailler sur le schéma suivant qui représente une gestion simplifiée de magasins (clé primaire soulignée, clé étrangère en ***GRAS***) :

```
PRODUIT     (IdP, Libelle, PrixAchat, CodeCat)
CATEGORIE   (CodeCat, Designation, CodeCatGen)
MAGASIN     (IdM, Nom, Type, IdL)
LIEU        (IdL, Nomlieu, IdLrattach)
VENTE       (IdP, IdM, DateV, Quantite, PrixVenteUnitaire)
```

Afin de faciliter le lancement du TP, je vous fournis les fichiers suivants :
`01_dropDB.sql` : suppression de la base de données du TP.
`02_createDB.sql` : création la base de données.
`03_insertData.sql` : insertion des données dans la base de données.

Partie 1 : Les statistiques

L'optimiseur peut optimiser s'il dispose de statistiques sur au moins une table de la requête.
La commande **ANALYZE** permet de calculer des statistiques sur les tables et les index.

Calcul de statistiques sur les tables

On trouve les informations statistiques dans la vue `USER_TABLES`.

1. Observer le schéma de la table `USER_TABLES` puis afficher le nombre de lignes et le nombre de blocs des tables de votre base **en une seule requête**. Que constatez-vous ?
2. Lancer le calcul de statistiques sur toutes vos tables à l'aide de la commande **`ANALYZE TABLE nom_table COMPUTE STATISTICS FOR TABLE;`**
3. Insérer à nouveau au moins un tuple dans au moins une table.
4. Afficher le nombre de lignes et le nombre de blocs des tables de votre base.
   Que constatez-vous ?
5. Lancer à nouveau le calcul de statistiques sur les tables à l'aide de la commande **`ANALYZE TABLE nom_table COMPUTE STATISTICS FOR TABLE;`**
6. Afficher le nombre de lignes et le nombre de blocs des tables de votre base.
   Que constatez-vous ?

Calcul de statistiques sur les index

ANALYZE INDEX nom_index COMPUTE STATISTICS;
ou
ANALYZE TABLE nom_table COMPUTE STATISTICS FOR ALL INDEXES;
On trouve alors les informations statistiques dans les vues DBA_INDEX, ALL_INDEX et USER_INDEXES.

7. Observer le schéma de la table USER_INDEXES et afficher le nombre de niveaux, le nombre de blocs-feuilles, le nombre de valeurs distinctes des index définis sur les clefs primaires ;
8. En suivant le même principe que pour le calcul de statistiques sur les tables, que constatez-vous ? à chaque étape ?

```
SELECT index_name, blevel, leaf_blocks, distinct_keys FROM USER_INDEXES;
```

Partie 2 : Plans d'exécution

Pour les plans d'exécution, utilisez la table Vente et une ou plusieurs autres tables si nécessaire (jointure par exemple).

Comparer et expliquer les différents plans obtenus en fonction de différents contextes.

Contextes d'exécution (attention, le choix des attributs A et B ne doivent jamais changer) :

a) A et B ne sont pas indexés.
b) A est indexé mais pas B.
c) A n'est indexé mais B oui
d) A et B sont indexés.

Requête :

```
1.  SELECT A FROM Vente;
2.  SELECT DISTINCT A FROM Vente;
3.  SELECT A FROM Vente WHERE A = <une constante présente>;
4.  SELECT A FROM Vente WHERE A = <une constante non présente>;
5.  SELECT A FROM Vente WHERE B = <une constante présente>;
6.  SELECT A FROM Vente WHERE B = <une constante non présente>;
7.  SELECT A FROM Vente WHERE A > <une constante présente>;
8.  SELECT A FROM Vente WHERE B > <une constante présente>;
9.  SELECT A FROM Vente ORDER BY A;
10. SELECT A FROM Vente ORDER BY B;
11. SELECT COUNT( A ) FROM Vente;
12. SELECT COUNT( A ) FROM Vente GROUP BY A;
13. SELECT A, COUNT( A ) FROM Vente GROUP BY A;
14. SELECT A, COUNT( DISTINCT A ) FROM Vente GROUP BY A;
15. SELECT A, COUNT( B ) FROM Vente GROUP BY A;
16. SELECT A, COUNT( DISTINCT B ) FROM Vente GROUP BY A;
17. SELECT B, COUNT( A ) FROM Vente GROUP BY B;
18. SELECT COUNT( A ), COUNT( B ) FROM Vente;
19. SELECT COUNT( A ), COUNT( B ) FROM Vente GROUP BY A, B;
```

Comparer et expliquer les différents plans en présence d'opérateurs ensemblistes.

20. Rédiger quatre requêtes d'interrogation illustrant chacune l'utilisation d'un des quatre opérateurs ensemblistes UNION, INTERSECT, MINUS et UNION ALL.
21. Justifier votre choix de requête et comparer les plans d'exécution de chaque requête notamment les plans obtenus pour UNION et UNION ALL.

Comparer et expliquer les différents plans en présence de vues.

22. Rédiger deux requêtes de votre choix, que vous justifierez, pour lesquelles vous afficherez les plans d'exécution.

23. Créer les deux vues correspondantes à vos deux requêtes.
24. Afficher les plans d'exécution des requêtes `SELECT * FROM Vue1(Vue2).`
25. Rédiger deux requêtes qui utilisent les vues créées en ajoutant une sélection et/ou une projection.
26. Comparer tous les plans de requêtes obtenus.

Comparer et expliquer les différents plans en présence de jointure.

Jointures entre deux attributs non indexés

27. Rédiger une requête réalisant une jointure prédicative (dans la clause `WHERE`).

Jointures entre la clé primaire et une clé étrangère non indexée

28. Rédiger une requête réalisant une jointure prédicative (dans la clause `WHERE`).
29. Rédiger une requête réalisant une jointure prédicative (dans la clause `FROM`).
30. Rédiger une requête réalisant une jointure prédicative à gauche (dans la clause `FROM`).
31. Rédiger une requête réalisant une jointure imbriquée (sous requête).

Jointures entre la clé primaire et une clé étrangère indexée

32. Rédiger une requête réalisant une jointure prédicative (dans la clause `WHERE`).
33. Rédiger une requête réalisant une jointure prédicative (dans la clause `FROM`).
34. Rédiger une requête réalisant une jointure prédicative à gauche (dans la clause `FROM`).
35. Rédiger une requête réalisant une jointure imbriquée (sous requête).

Opération spécifique Group By (cf. cours datacube)

36. Rédiger la requête Group by Cube(A, B, …, X) en utilisant l'UNION des Group By.
37. Rédiger la requête Group by Cube(A, B, …, X) en utilisant Group By Cube (A, B, …, X).

ANNEXES

Optimisation

L'outil EXPLAIN donne le plan d'exécution d'une requête. La description comprend :

- Le chemin d'accès utilisé.
- Les opérations physiques (tri, fusion, intersection…).
- L'ordre des opérations. Il est représentable par un arbre.

Cet outil permet de comprendre les actions d'Oracle ou de MySQL lors d'une requête SQL afin d'améliorer la rapidité d'exécution. L'une des méthodes pour améliorer la performance des requêtes SQL est l'utilisation des index.

Les méthodes d'optimisation

Pour déterminer le meilleur plan d'exécution, l'optimiseur dispose de deux méthodes, l'une statique et l'autre statistique.

1 Méthode statique

Disponible dans les versions antérieures à la version 7 d'Oracle, l'optimiseur choisit le plan d'exécution en fonction des chemins d'accès disponibles pour chaque table utilisée dans la commande SQL en utilisant l'opération ayant le poids le moins élevé.

2 Méthode statistique

Disponible à partir de la version 7 d'Oracle, elle se base sur des statistiques relatives aux tables pour déterminer le meilleur chemin d'accès.

Ces statistiques concernent le volume des tables (nombre de lignes), le nombre de blocs alloués à la table, le nombre de valeurs distinctes de chaque colonne, la distribution de la clé, le nombre de valeurs différentes d'un index.

Le but de cette méthode est le meilleur débit, ou le meilleur temps pour traiter les lignes concernées par la commande.

Utilisation de la commande EXPLAIN PLAN

ORACLE et MySQL fournissent un outil, EXPLAIN, qui donne une description du plan d'exécution choisi par le système pour une requête quelconque.

EXPLAIN PLAN est une commande qui permet d'expliquer le plan d'exécution ou le chemin d'accès que l'optimiseur va utiliser pour exécuter la commande SQL. Les résultats de la commande EXPLAIN PLAN sont mis dans la table PLAN_TABLE.

Pour ORACLE

La description globale de l'optimisation est là (attention, ne vous perdez pas dans la documentation) :
http://docs.oracle.com/cd/B28359_01/server.111/b28274/ex_plan.htm.
La description des colonnes de la table PLAN_TABLE est là :
http://docs.oracle.com/cd/B28359_01/server.111/b28274/ex_plan.htm#i18300
L'utilisation de EXPLAIN PLAN est là :
http://docs.oracle.com/cd/B28359_01/server.111/b28274/ex_plan.htm#i17492
La description nécessaire à la bonne lecture d'un plan d'exécution est là :
http://docs.oracle.com/cd/B28359_01/server.111/b28274/ex_plan.htm#i23461

Explications complémentaires pour ORACLE : On choisit un identifiant pour identifier le plan d'exécution et on spécifie la requête à analyser :

```
SQL> EXPLAIN PLAN
SET STATEMENT_ID=' Identifiant_requête'
FOR requête ;
```

Exemple:

```
SQL> EXPLAIN PLAN
SET STATEMENT_ID='exemple'
FOR SELECT NOM from ACTEURS order by NOM;
```

On interroge ensuite la table PLAN_TABLE pour afficher le plan d'exécution :

```
SQL> SELECT * FROM PLAN_TABLE; -- par exemple;
```

On peut ne pas afficher tous les champs et améliorer la présentation :

```
SQL> SELECT STATEMENT_ID, OPERATION, OPTIONS, ID, PARENT_ID, POSITION
FROM PLAN_TABLE
WHERE STATEMENT_ID='exemple';
```

On peut améliorer considérablement la lecture du plan d'exécution :

```
SQL> select lpad(' ', 2*(level-1)) || operation ||' '||options||' ' ||
object_name ||' '||decode(id, 0, 'Cost = ' ||position) from plan_table
start with id = 0 and STATEMENT_ID='exemple'
connect by prior id = parent_id
and statement_id='exemple';
```

Commande AUTOTRACE

Cette commande permet d'obtenir le plan d'exécution lorsqu'une requête est exécutée.
Dans SQL*Plus ou sqldeveloper il est possible d'obtenir directement le plan d'exécution et des statistiques avec la commande AUTOTRACE.

La commande s'utilise tout simplement en tapant : SET AUTOTRACE ON.

```
SQL> SET AUTOTRACE ON
```

Syntaxe : SET AUTOT[RACE] {OFF | ON | TRACE[ONLY]} [EXP[LAIN]] [STAT[ISTICS]]

OFF : Désactive le suivi d'exécution automatique des instructions SQL.
ON : Active le suivi d'exécution automatique des instructions SQL.
TRACEONLY : active le suivi d'exécution automatique des instructions SQL et supprime la sortie des instructions.
EXPLAIN : Affiche le plan d'exécutions mais pas les statistiques.
STATISTICS : Affiche les statistiques mais pas les plans d'exécution.
Si les deux dernières options sont omises, les statistiques et les plans d'exécution sont affichés par défaut.

Exemple :

```
SQL> SET AUTOTRACE ON EXP
```

Après cette commande, le plan d'exécution sera affiché après l'exécution d'une requête SQL.

On peut connaître le temps précis d'une requête (temps exprimé en centième de seconde) par la commande :

```
SQL> SET TIMING ON;
```