TD1

Optimisation algébrique et physique de requêtes dans Oracle

Vous devez rendre un rapport (format pdf) et le script SQL répondant aux questions en commentant les résultats.

On considère les tables suivantes de *Basetd* : Employe, Service, Projet et Travail :

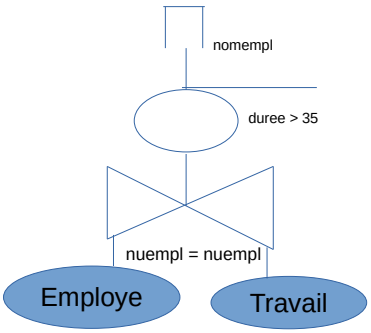
Employe (nuempl, nomempl, hebdo,affect)
Service (nuserv,nomserv,chef)
Projet (nuproj,nomproj,resp)
Travail (nuempl,nuproj,duree)

Exercice 1 On considère la requête SQL suivante formulée sur les tables Employe et Travail :

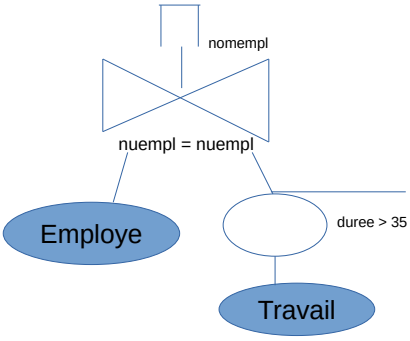
(Q1) : *select nomempl from employe e, travail t where t.duree > 35 and e.nuempl = t.nuempl;*

Voici deux représentations algébriques de Q1.

On suppose que le nombre de tuples de la table Employé (Travail) est 100 (500) et que seuls cinq employés travaillent plus de 35h sur des projets.



Arbre algébrique n° 1



Arbre algébrique n° 2

Evaluer et comparer le coût de chaque représentation selon les deux critères suivants :

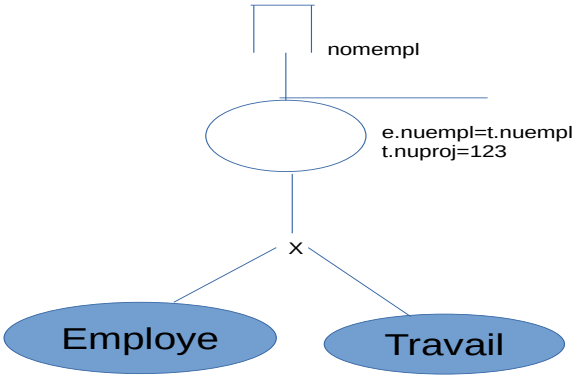
(a) Espace Mémoire et
(b) Nombre d'Opérations

Exercice 2 On considère la requête SQL suivante formulée sur les tables Employe et Travail :

(Q2) select nomempl from employe e, travail t where t.nuproj = 123 and e.nuempl = t.nuempl;

La figure suivante propose un arbre algébrique ***basic*** correspondant à la requête Q2 :

Proposer un arbre algébrique optimisé par transformations successives de cette requête.



Arbre algébrique

Exercice 3 : On possède quelques statistiques et supposisions à propos de deux tables Employe et Travail :

Statistiques :
nombre de tuples de la table Employé = 3000 ;
nombre de tuples de la table Employé par bloc (b) = 30 ;
nombre de valeurs distincte de l'attribut affect dans la table Employé = 500 ;
nombre de tuples de la table Travail = 100 000 ;
nombre de tuples de la table Travail par bloc (b) = 50;

Questions :

On suppose de plus qu'il existe :
- une fonction de hachage définie sur l'attribut ***nuempl*** de la table Employe et
- un index secondaire défini sur l'attribut ***affect*** de la table Employe.

1. Estimer le nombre de blocs nécessaires pour contenir les tuples de la relation Employe
2. Estimer le nombre de blocs nécessaires pour contenir les tuples de la relation Travail
3. Estimer le coût maximal (exprimé en nombre de blocs lus) de chacune des opérations de sélection suivantes :

a) $\sigma_{nuempl=123}$
b) $\sigma_{hebdo=15}$
c) $\sigma_{affect=1}$

4. Pour l'opération de jointure Employé ∞_{nuempl} Travail,

a) Evaluer le coût de cette opération par application des l'algorithme *Nested-loop join.*
b) On suppose maintenant qu'il existe une clé primaire défini sur la table Travail et chaque employé est impliqué dans un seul projet.. Evaluer le coût de cette opération par application de l'algorithme *Single-loop join.*

Plan d'exécution de requêtes dans Oracle

Exercice 4 : *On cherche dans cet exercice à extraire le plan d'exécution généré par Oracle suite à l'exécution d'une requête SQL et on cherche à améliorer les performances d'exécution de cette requête. Vous êtes invités à consulter votre cours ainsi que tous les documents sur internet (essentiellement la documentation Oracle) pour répondre aux questions.*

Tout d'abord, vous effectuez dans votre espace de travail une copie de la base de données ***basetd*** *composée des tables Employe, Service, Projet et Travail.*

Nous allons apprendre à extraire les statistiques collectées par Oracle. Ces statistiques sont généralement utilisées par l'optimiseur pour choisir le plan d'exécution le plus performant.

Il est possible pour un utilisateur de demander des statitstiques d'une table à l'aide de la commande SQL suivante :

exec dbms_stats.gather_table_stats('{nom-espace-travail}', '{nom-de-la-table}')

a) *Exécuter cette commande sur les tables de basetd, observer et commenter les résultats en consultant la table user_tab_statistics et la table user_tab_col_statistics du dictionnaire de données.*

b) *Créer un histogramme sur l'attribut hebdo de la table Employe :*

exec dbms_stats.gather_table_stats('{nom-du-schema}', 'EMPLOYE', method_opt=>'FOR COLUMNS HEBDO')

l'option 'method_opt=>for all columns' concerne tous les attributs de la table Employe.

À l'aide des tables de statistiques, reporter la preuve que l'histogramme a bien été créé. Ce type d'histogramme est très pertinent car il donne une image précise de la distribution d'une colonne à un instant t. Consulter les tables user_tab_histograms et user_tab_col_statistics et commenter les resultats.

```
select ENDPOINT_NUMBER , ENDPOINT_VALUE
from user_tab_histograms where table_name='EMPLOYE' and column_name='HEBDO' ;

select column_name, num_distinct, num_buckets, histogram
from user_tab_col_statistics where table_name = 'EMPLOYE' AND column_name = 'HEBDO';
```

Nous nous intéressons aux ***plans d'exécution et au travail de l'optimiseur****. Les informations sur l'exécution de la requête sont temporairement consignées dans une zone mémoire (cursor cache) et la trace d'exécution peut être présentée à l'aide d'un ordre SQL de formatage :*

SELECT * FROM table (dbms_xplan.display_cursor);

Deux commandes SQL sont utiles pour simplifier l'affichage à l'écran des résultats :

```
alter system flush shared_pool;
alter system flush buffer_cache;
```

Nous commençons par étudier la requête suivante :

(Q3) : SELECT e.nuempl, t.duree FROM Employe e JOIN Travail t ON e.nuempl = t.nuempl WHERE t.nuproj = 3;

c) *Exécuter la requête Q3 suivi de la commande display_cursor. Que représente sql_id et child_number. Inutile de s'attarder sur toutes les informations.*

d) *Exécuter et expliquer l'ordre SQL suivant en remplacant nom-du-schéma par le nom de votre schéma ou le login Oracle:*

(Q4) : select sql_id, child_number, sql_text from v$sql where sql_text like '%t.duree%' and sql_text not like '%v$sql %' and parsing_schema_name like 'NOM-DU-SCHEMA';

La vue v$sql contient les éléments du cursor cache et va nous permettre de retrouver les paramètres sql_id et child_number de la requête Q3 .

e) *On souhaite maintenant afficher le plan d'exécution de la requête Q3. Pour cela, adapter la requête suivante en remplaçant* ***id*** *et* ***num*** *par des valeurs retournées par la requêtes SQL Q4 :*

SELECT * FROM table(dbms_xplan.display_cursor('sql_id', child_number, 'all'));

La recherche de l'identifiant de requête dans la vue v$sql et l'affichage du plan d'exécution à l'aide de display_cursor peuvent être combinés de la sorte :

(Q5) : select t.plan_table_output from v$sql s,
table(dbms_xplan.display_cursor(s.sql_id,s.child_number,'all')) t
where s.sql_text like '%find me%' and s.sql_text not like '%v$sql%'
and s.parsing_schema_name like '{NOM-DU-SCHEMA}';

f) *Adapter et exécuter la requête SQL Q5.*

g) *L'ordre explain plan offre une alternative à l'examen de la trace d'exécution :*

-- calcul de l'optimiseur
(Q6) : explain plan set statement_id = 'ex_plan1' for requete;

-- affichage du plan

(Q7) : select * from table(dbms_xplan.display(statement_id=>'ex_plan1', format=>'all'));

À la différence de l'ordre display_cursor, explain plan propose un plan d'exécution mais il n'évalue pas la requête ! Cet outil est donc très utile pour analyser les problèmes de performance de requêtes extrêmement coûteuses .

i) Vérifier sur la requête Q3 que l'analyse produite par explain plan est conforme (ou non) à celle de display_cursor. Reporter les différences le cas échéant.

Exercice 5 : *utilisation des hints*

Les hints sont des ***conseils*** *que vous proposez à l'optimiseur dans la génération du plan d'exécution d'une requête. Il est possible que oracle ne suit pas vos conseils et génère son propre plan d'exécution s'il estime que le sien est plus optimisé. +Par exemple, pour imposer l'usage d'un index :*

SELECT /*+ index(<nom-table> <nom-idx>) */ * FROM t WHERE ...;

a) Ecrire une requête qui renvoie toutes les informations concernant les employés travaillant sur des projets. On conseil à Oracle l'utilisation les algorithmes suivants :

- ***Nested loop***
- ***Merge join***
- ***Hash join***

Exercice 6 : *INDEX*

Pour améliorer les performances dans le traitement des requêtes nous allons créer des index. La création d'index se fait en exécutant la commande suivante :

CREATE INDEX nom_index ON table (attribut).

Une clé primaire définie sur une table constitue un index créé par ORACLE.

Par exemple, la commande CREATE INDEX INDX_NUMFO ON Distribution (numfo) créé un index nommé INDX_NUMFO sur l'attribut numfo dans la table Distribution.

Dans cet exercice, vous formulez vos requêtes sur les tables Distribution, Opérateur, Commune et Département existantes dans ***basetd***. Le but est de proposer des requêtes incitant l'optimiseur Oracle à utiliser des index pour améliorer la performance des requêtes.

a) *Proposer une requête SQL permettant à l'optimiseur ORACLE d'utiliser l'opérateur INDEX RANGE SCAN.*

En général, pour inciter l'optimiseur à choisir INDEX RANGE SCAN, il faut choisir une intervalle de recherche de valeurs dans la clause Where suffisamment petit, car s'il inclut trop d'enregistrements, l'optimiseur préférera faire un TABLE ACCESS FULL SCAN.

Vous pouvez forcer l'optimiseur avec le hint /+ irs */.*

b) Quelle est la différence entre *les opérateurs suivants : TABLE ACCESS BY INDEX ROWID et TABLE ACCESS BY INDEX ROWID (BATCHED). Proposer une requête SQL permettant à l'optimiseur ORACLE d'utiliser l'opérateur TABLE ACCESS BY INDEX ROWID BATCHED.*

Pour inciter l'optimiseur, vous pouvez spécifier plusieurs intervalles, de sorte que le nombre de tuples à récupérer l'incite à utiliser un TABLE ACCESS.

h) *Proposer une requête SQL permettant à l'optimiseur ORACLE d'utiliser l'opérateur MERGE JOIN.*

Une jointure avec beaucoup d'enregistrements incite Oracle à utiliser MERGE JOIN.

Exercice 7 : PL/SQL, Curseur et Plan d'exécution

On considère la table distribution de Basetd.

a) Ecrire une procédure en PL/SQL appelée ***distribution_static (utilisant le type Cursor)*** *prenant en entrée un paramètre : statut de l'opérateur. Cette procédure affiche l'id de opérateur.*
On distingue les cas suivant :

- la valeur de statut est NULL. Dans ce cas on affiche les ids de tous les opérateurs

- la valeur de statut est différent de NULL. Dans ce cas on affiche les ids des opérateurs qui respectent la condition.

Vous devez afficher le plan d'exécution de la requête à chaque fois.

Aide pour écrire cette procédure:

```
Create or replace procedure distribution_static(para1 in type default NULL, para2 in type default NULL ) as
cursor ma_requete is select id from distribution where ...;
cursor pe is select plan_table_output from table(dbms_xplan.display_cursor);
v_pe varchar2(1000);
id number(10);
begin
...
open ma_requete ;
loop
fetch ma_requete into id;
exit when ma_requete%notfound;
dbms_output.put_line(id);
end loop;
close ma_requete;
…
/* On affiche le plan d'execution */
open pe
…
end ;/
```

b) Ecrire une procédure en PL/SQL appelée ***distribution_dynamique (utilisant le type is ref cursor)*** *prenant en entrée deux paramètres : ADR_NM_CP et STATUT. Cette procédure affiche les id des opérateus selon les valeurs des paramètres lors des appels à cette procédure. On distingue plusieurs cas :*

- les valeurs de numfo et statut sont NULL. Dans ce cas vous affichez les ids de tous les opérateurs
- la valeur fr adr_NM_CP est NULL
- la valeur de statut est Null
- les valeurs de numfo et statut sont différents de NULL

A chaque fois vous affichez le résultat et le plan d'exécution de la requête.

Aide pour écrire cette procédure :

```
Create or replace procedure distribution_static(para1 in type default NULL, para2 in type default NULL ) as
TYPE cursor_type IS REF CURSOR;
ma_requete cursor_type;
v_requete varchar2(300);
v_id number;
v_pe varchar2(200);
cursor pe is select plan_table_output from table(dbms_xplan.display_cursor);
begin
v_requete:='select  id from distribution2 where 1 = 1';
if statut is not null then v_requete=v_requete||' and statut=:statut';end if;
if numfo is not null then v_requete:=v_requete||' and numfo=:numfo'; end if;
if statut is null and code is null then
open ma_requete for v_requete;
end if;
if statut is not null and code is null then
open ma_requete for v_requete using statut;
end if;
...
end;
/
```

Vous devez faire plusieurs appels aux deux procédure PL/SQL avec des valeurs de paramètres différentes. Vous commentez le Plan d'exécution à chaque fois

Exercice 8 Générer et analyser les plans d'exécutions des requêtes suivantes

Sans clé primaire définie sur les tables Employe, Travail et Service

select * from service;
select * from employe where nuempl = 17;
select nomempl,duree from travail t, employe e where e.nuempl = t.nuempl;
select * from employe where nuempl in (select nuempl from travail);
select * from employe where nuempl not in (select nuempl from travail);
select nomempl from employe e where exists (select * from travail t where t.duree > 25 and t.nuempl=e.nuempl);
select nomempl from employe e where not exists (select * from travail t where t.duree > 25 and t.nuempl=e.nuempl);

Avec une clé primaire définie sur la table Employe
select * from employe where nuempl =17;

Avec une clé primaire définie sur la table Travail
select nomempl,duree from employe e,travail t where e.nuempl = t.nuempl;

Avec un index secondaire sur l'attribut durée de la table Travail
select nomempl from employe e, travail t where duree >25 and t.nuempl=e.nuempl
select nomserv, nomempl from service, Employe where affect = nuserv and affect = 2