

TD1 Optimisation algébrique et physique de requêtes dans Oracle

Alexandre Clenet - Florian Tran

Année : 3

Groupe : 1-2

Base de donnée

Employe (nuempl, nomempl, hebdo,affect)

Service (nuserv,nomserv,chef)

Projet (nuproj,nomproj,resp)

Travail (nuempl,nuproj,duree)

Première Partie du TD

TD1

BD

a) 500

b) $100 \times 500 + 500 = 50050$

Exercice 2

R.n°4 $\sigma_{p=9}(R)$
 $= \sigma_p(\sigma_q(R))$

R.n°2 $\sigma_{e.numproj=123}(R)$
 $\sigma_{e.numproj=123}(\sigma_{e.numempl=v.numempl}(\text{Travail} \times \text{Employer}))$
 $\sigma_{e.numempl=v.numempl}(\sigma_{e.numproj=123}(\text{Travail}) \times \text{Employer})$
 R.n°2 $\sigma_{e.numempl=v.numproj}(R \times \text{Employe})$
 (Jeune) R \times Employe
 $e.numempl = v.numproj$

A: $\pi_{numempl}(\sigma_{e.numproj=123}(\text{Travail})) \times \text{Employe}$
 $e.numempl = v.numproj$

Exercice 3

1. 100 b 3000 \div 30

2. 2000 b 20000 \div 50

3. a) 1 b
 b) 100 b
 c) 7 b
 446

Exercice 4 : Plan d'exécution de requêtes dans Oracle

a) Collecte de statistiques

```
-- Collecte de statistiques sur les tables
exec dbms_stats.gather_table_stats('S5A08B', 'EMPLOYEE');
```

Collecte de statistiques :

```
select * from user_tab_statistics;
select * from user_tab_col_statistics;
select * from user_tab_histograms;
```

b) Création de l'histogramme

```
-- Création de l'histogramme sur l'attribut "hebdo" de la table "Employe"
exec dbms_stats.gather_table_stats('S5A08B', 'EMPLOYEE', method_opt => 'FOR COLUMNS
HEBDO');
select * from user_tab_histograms;
```

c) Exécution de la requête Q3 et affichage du plan d'exécution

```
-- Exécution de la requête Q3
SELECT e.nuempl, t.duree
FROM Employe e
JOIN Travail t ON e.nuempl = t.nuempl
WHERE t.nuproj = 3;

-- Plan d'exécution
select sql_id, child_number, sql_text from v$sql where sql_text
not like '%v$sql%' and parsing_schema_name like 'S5A08B';
```

sql_id et child_number sont des identificateurs utilisés pour suivre et identifier de manière unique les exécutions de requêtes SQL dans Oracle :

sql_id : Il est unique pour chaque exécution de requête SQL. Cela permet de suivre et d'analyser les performances et les informations liées à cette requête.

child_number : Il représente le numéro d'enfant associé à une requête. Une même requête peut avoir plusieurs plans d'exécution possibles en fonction de différents facteurs, et chaque plan est associé à un child_number distinct.

d) Recherche des paramètres sql_id et child_number


```
-- Recherche des paramètres sql_id et child_number de la requête Q3
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 'S5A08B';
```

e) Affichage du plan d'exécution de la requête Q3

```
-- Affichage du plan d'exécution de la requête Q3
SELECT * FROM table(dbms_xplan.display_cursor('f52w2vurfvr3w', 0, format=>'all'));
```

f) Utilisation de l'ordre explain plan

```
-- Utilisation de explain plan
EXPLAIN PLAN SET statement_id = 'ex_plan1' FOR
SELECT e.nuempl, t.duree
FROM Employe e
JOIN Travail t ON e.nuempl = t.nuempl
WHERE t.nuproj = 3;

-- Affichage du plan d'exécution
SELECT * FROM table(dbms_xplan.display(statement_id => 'ex_plan1', format =>
'all'));
```

g) Comparaison des plans d'exécution

Résultats de la comparaison entre display_cursor et explain plan : ...

Exercice 5 : Utilisation des hints

```
SELECT /*+
      USE_NL(e t) -- Utilise Nested Loop Join
      USE_MERGE(e t) -- Utilise Merge Join
      USE_HASH(e t) -- Utilise Hash Join
*/
e.*, t.*
FROM Employe e
JOIN Travail t ON e.nuempl = t.nuempl;
```

Exercice 6 : INDEX

a) INDEX RANGE SCAN

```
-- Exemple de requete
SELECT /*+ irs */ *
FROM Employe
WHERE nuempl < 1000;
```

b)

La différence entre TABLE ACCESS BY INDEX ROWID et TABLE ACCESS BY INDEX ROWID (BATCHED) réside dans le nombre de ROWIDs que l'opération d'accès utilise. TABLE ACCESS BY INDEX ROWID récupère une seule ROWID à la fois, tandis que TABLE ACCESS BY INDEX ROWID (BATCHED) en récupère plusieurs à la fois.

Supposons que vous vouliez accéder aux employés par leur numéro de projet en utilisant un index. Voici comment vous pourriez formuler une requête pour utiliser TABLE ACCESS BY INDEX ROWID BATCHED :

```
SELECT /*+ use_batched_rowid */ nuempl
FROM Travail
WHERE nuproj = :votre_numero_de_projet;
```

h) Opérateur MERGE JOIN

```
-- Exemple de requete
SELECT /*+ merge */ e.nuempl, p.nomproj
FROM Employe e
JOIN Projet p ON e.affect = p.nuproj;
```

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

```
-- Distribution statique
Create or replace procedure distribution_static(stat in varchar2 default NULL) as
cursor ma_requete is select id from distribution where statut=stat or statut is
null;
id number(10);
cursor plan is select plan_table_output from table(dbms_xplan.display_cursor);
v_plan_table_output varchar2(1000);
begin
dbms_output.enable(1000000);
open ma_requete;
loop
fetch ma_requete into id;
exit when ma_requete%notfound;
dbms_output.put_line(id);
end loop;
close ma_requete;

dbms_output.put_line(
```

```

chr(10)||'Résultat de la requête suivante :'||
chr(10)||
chr(10)||'select id from distribution where'||
chr(10)||'(statut=stat;');
dbms_output.put_line(chr(10));
dbms_output.put_line('Plan Execution:'||chr(10));
open plan;
loop
fetch plan into v_plan_table_output;
exit when plan%notfound;
dbms_output.put_line(v_plan_table_output);
end loop;
close plan;
end;
/

```

```

-- Distribution dynamique
Create or replace procedure distribution_dynamique(statut in varchar2 default
NULL,code in number default NULL) as

TYPE cursor_type IS REF CURSOR;
c_cursor cursor_type;

ma_requete varchar2(300);
v_id number;
v_statut varchar2(60);
v_plan_table_output varchar2(200);
cursor plan is select plan_table_output from table(dbms_xplan.display_cursor);

begin

dbms_output.enable(1000000);

ma_requete:='select id from distribution where 1 = 1';

if statut is not null then ma_requete:=ma_requete||' and statut=:statut';end if;

if code is not null then ma_requete:=ma_requete||' and ADR_NM_CP=:code';
end if;

if statut is null and code is null then
open c_cursor for ma_requete;
end if;

if statut is not null and code is null then
open c_cursor for ma_requete using statut;
end if;

if statut is null and code is not null then
open c_cursor for ma_requete using code;
end if;

```

```
if statut is not null and code is not null then
open c_cursor for ma_requete using statut,code;
end if;

loop
fetch c_cursor into v_id;
exit when c_cursor%notfound;
dbms_output.put_line(v_id);
end loop;
close c_cursor;
dbms_output.put_line(
chr(10)||'Résultat de ma requête:'||
chr(10)||
chr(10)||ma_requete);

dbms_output.put_line(chr(10)||' Plan : '||chr(10));
open plan;
loop
fetch plan into v_plan_table_output;
exit when plan%notfound;
dbms_output.put_line(v_plan_table_output);
end loop;
close plan;
end;
/
```

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

Pour chaque requete :

```
-- EXEC LA REQUETE ICI

EXPLAIN PLAN FOR SELECT * FROM service s;
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

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

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

select * from service;

Toutes les lignes extraites : 8 en 0,210 secondes

PLAN_TABLE_OUTPUT							
1	Plan hash value: 2190496544						
2							
3	-----						
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
5	-----						
6	0	SELECT STATEMENT		5	135	3 (0)	00:00:01
7	1	TABLE ACCESS FULL	SERVICE	5	135	3 (0)	00:00:01
8	-----						

select * from employe where nuempl = 17;

Toutes les lignes extraites : 13 en 0,264 secondes

PLAN_TABLE_OUTPUT							
1	Plan hash value: 1519211061						
2							
3	-----						
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
5	-----						
6	0	SELECT STATEMENT		1	30	3 (0)	00:00:01
7	* 1	TABLE ACCESS FULL	EMPLOYE	1	30	3 (0)	00:00:01
8	-----						
9							
10	Predicate Information (identified by operation id):						
11	-----						
12							
13	1 - filter("NUEMPL"=17)						

select nomempl,duree from travail t, employe e where e.nuempl = t.nuempl;

PLAN_TABLE_OUTPUT							
1	Plan hash value: 269265833						
2							
3	-----						
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
5	-----						
6	0	SELECT STATEMENT		38	1140	6 (0)	00:00:01
7	* 1	HASH JOIN		38	1140	6 (0)	00:00:01
8	2	TABLE ACCESS FULL	EMPLOYE	21	504	3 (0)	00:00:01
9	3	TABLE ACCESS FULL	TRAVAIL	38	228	3 (0)	00:00:01
10	-----						
11							
12	Predicate Information (identified by operation id):						
13	-----						
14							
15	1 - access("E"."NUEMPL"="T"."NUEMPL")						

select * from employe where nuempl in (select nuempl from travail);

PLAN_TABLE_OUTPUT									
1	Plan hash value: 432089935								
2									
3	-----								
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time		
5	-----								
6	0	SELECT STATEMENT		19	627	6 (0)	00:00:01		
7	* 1	HASH JOIN SEMI		19	627	6 (0)	00:00:01		
8	2	TABLE ACCESS FULL	EMPLOYE	21	630	3 (0)	00:00:01		
9	3	TABLE ACCESS FULL	TRAVAIL	38	114	3 (0)	00:00:01		
10	-----								
11									
12	Predicate Information (identified by operation id):								
13	-----								
14									
15	1	- access("NUEMPL"="NUEMPL")							

select * from employe where nuempl not in (select nuempl from travail);

PLAN_TABLE_OUTPUT									
1	Plan hash value: 2205114684								
2									
3	-----								
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time		
5	-----								
6	0	SELECT STATEMENT		2	66	6 (0)	00:00:01		
7	* 1	HASH JOIN ANTI SNA		2	66	6 (0)	00:00:01		
8	2	TABLE ACCESS FULL	EMPLOYE	21	630	3 (0)	00:00:01		
9	3	TABLE ACCESS FULL	TRAVAIL	38	114	3 (0)	00:00:01		
10	-----								
11									
12	Predicate Information (identified by operation id):								
13	-----								
14									
15	1 - access("NUEMPL"="NUEMPL")								

select nomempl from employe e where exists (select * from travail t where t.duree > 25 and t.nuempl=e.nuempl);

PLAN_TABLE_OUTPUT									
1	Plan hash value: 432089935								
2									
3	-----								
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time		
5	-----								
6	0	SELECT STATEMENT		7	210	6 (0)	00:00:01		
7	* 1	HASH JOIN SEMI		7	210	6 (0)	00:00:01		
8	2	TABLE ACCESS FULL	EMPLOYE	21	504	3 (0)	00:00:01		
9	* 3	TABLE ACCESS FULL	TRAVAIL	8	48	3 (0)	00:00:01		
10	-----								
11									
12	Predicate Information (identified by operation id):								
13	-----								
14									
15	1	access("T"."NUEMPL"="E"."NUEMPL")							
16	3	filter("T"."DUREE">25)							

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

PLAN_TABLE_OUTPUT									
1	Plan hash value: 3075749639								
2									
3	-----								
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time		
5	-----								
6	0	SELECT STATEMENT		14	420	6 (0)	00:00:01		
7	* 1	HASH JOIN ANTI		14	420	6 (0)	00:00:01		
8	2	TABLE ACCESS FULL	EMPLOYE	21	504	3 (0)	00:00:01		
9	* 3	TABLE ACCESS FULL	TRAVAIL	8	48	3 (0)	00:00:01		
10	-----								
11									
12	Predicate Information (identified by operation id):								
13	-----								
14									
15	1	access("T"."NUEMPL"="E"."NUEMPL")							
16	3	filter("T"."DUREE">25)							

Avec une clé primaire définie sur la table Employe

select * from employe where nuempl =17;

Toutes les lignes extraites : 14 en 0,235 secondes

PLAN_TABLE_OUTPUT							
1	Plan hash value: 3865044209						
2							
3	-----						
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
5	-----						
6	0	SELECT STATEMENT		1	51	0 (0)	00:00:01
7	1	TABLE ACCESS BY INDEX ROWID	EMPLOYE	1	51	0 (0)	00:00:01
8	* 2	INDEX UNIQUE SCAN	PK_EMPL	1		0 (0)	00:00:01
9	-----						
10							
11	Predicate Information (identified by operation id):						
12	-----						
13							
14	2	- access("NUEMPL"=17)					

Avec une clé primaire définie sur la table Travail

select nomempl,duree from employe e,travail t where e.nuempl = t.nuempl;

Toutes les lignes extraites : 21 en 0,184 secondes

PLAN_TABLE_OUTPUT							
1	Plan hash value: 1494032871						
2							
3	-----						
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
5	-----						
6	0	SELECT STATEMENT		2	102	3 (0)	00:00:01
7	1	NESTED LOOPS		2	102	3 (0)	00:00:01
8	2	NESTED LOOPS		2	102	3 (0)	00:00:01
9	3	TABLE ACCESS FULL	TRAVAIL	2	52	3 (0)	00:00:01
10	* 4	INDEX UNIQUE SCAN	PK_EMPL	1		0 (0)	00:00:01
11	5	TABLE ACCESS BY INDEX ROWID	EMPLOYE	1	25	0 (0)	00:00:01
12	-----						
13							
14	Predicate Information (identified by operation id):						
15	-----						
16							
17	4	- access("E"."NUEMPL"="T"."NUEMPL")					
18							
19	Note						
20	-----						
21	- dynamic statistics used: dynamic sampling (level=2)						

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
```