HMIN122M Rendu
TP2

Bachar Rima Joseph Saba Tasnim Muhammad

26 septembre 2018

Table des matières

1 Sélection

1.1 Question 1

@script_table.sql : création des tables de la base de données.

@script_remplissage.sql : remplissage des tables crées.

set autotrace on : permettre de faire des statistiques sur les requêtes exécutées.

set linesize 200 : étendre la taille des tables affichées pour une meilleure visibilité.

1.2 Question 2

```
1  explain plan for select nom from ville where
       insee='34172';
2  select plan_table_output from
       table(dbms_xplan.display());
```

Listing 1 – plan d'exécution choisi par l'optimiseur pour la requête permettant d'afficher le nom des villes dont le numéro insee est 34172

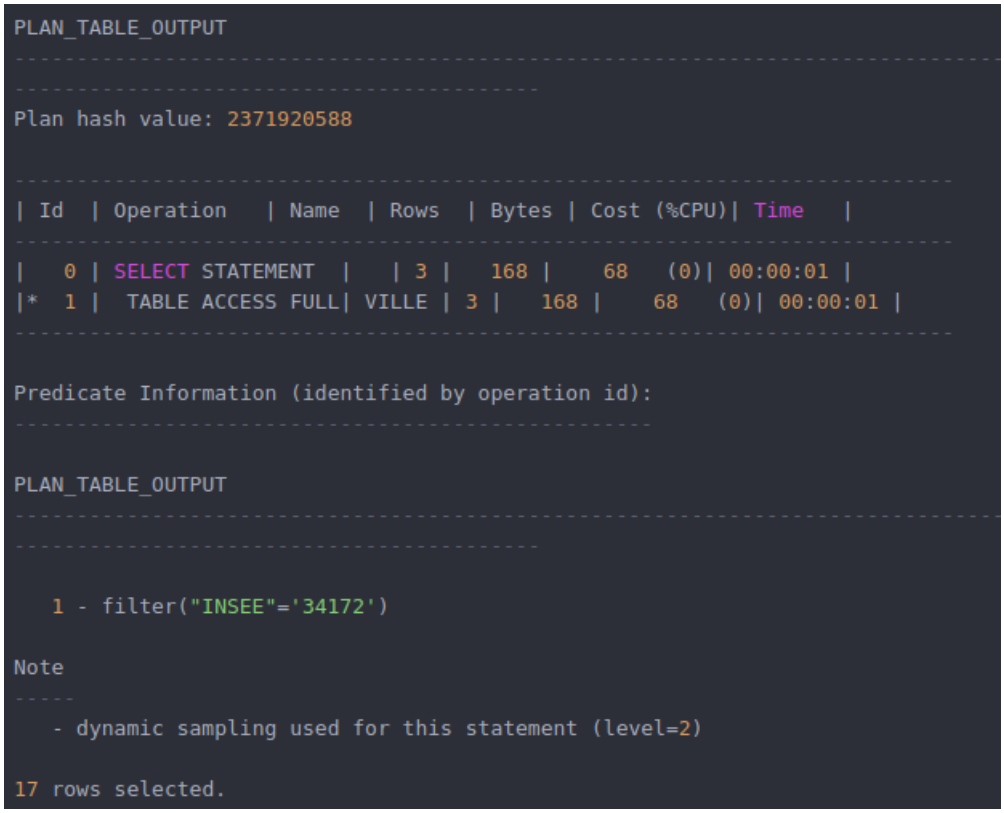


FIGURE 1 – Plan d'exécution

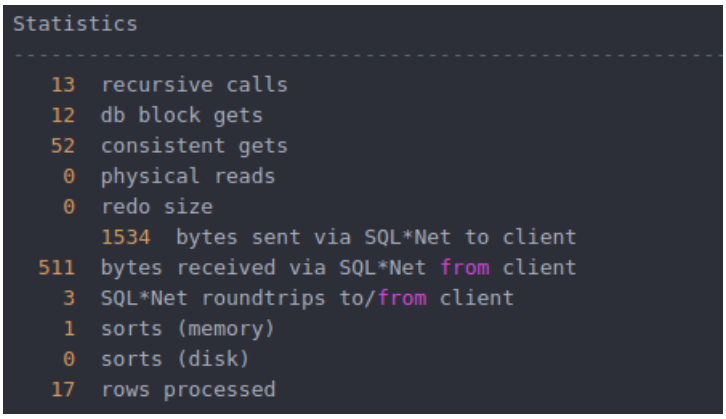


FIGURE 2 – Statistiques

1.3 Question 3

```
1  alter table ville add constraint PK_VILLE PRIMARY
     KEY(insee);
```

Listing 2 – ajout d'une clé primaire sur la table ville en utilisant l'attribut insee

1.4 Question 4

Avant l'ajout d'une clé primaire à la table `Ville`, l'algorithme choisi par l'optimiseur pour optimiser la requête est *Table Access Full*. Ce dernier est généralement gourmand en termes de consommation de ressources, vu qu'il doit balayer la table entièrement, ligne par ligne, et attribut par attribut.

Après l'ajout d'une clé primaire à la table `Ville`, notamment sur l'attribut **insee**, l'algorithme choisi par l'optimiseur est ainsi *Index Unique Scan*. Ce dernier permet de parcourir la table selon l'indexe placé sur la clé primaire et est ainsi moins gourmand en termes de consommation de ressources.

2 Jointure

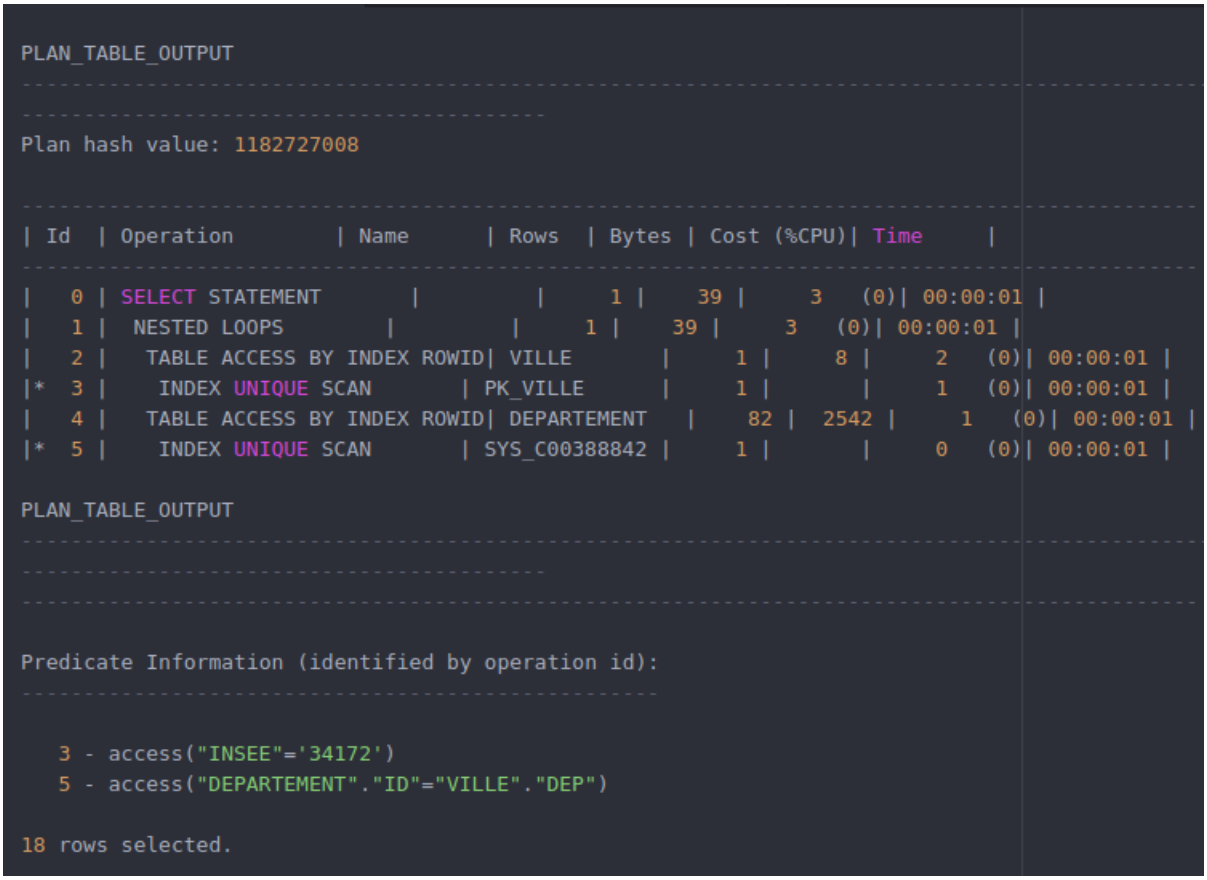
2.1 Question 5

```
1  explain plan for select departement.nom from
     departement, ville where insee='34172' and
     departement.id = ville.dep;
2  select plan_table_output from
     table(dbms_xplan.display());
```

Listing 3 – plan d'exécution choisi par l'optimiseur pour la requête permettant d'afficher le nom du département pour la ville dont le numéro insee est 34172

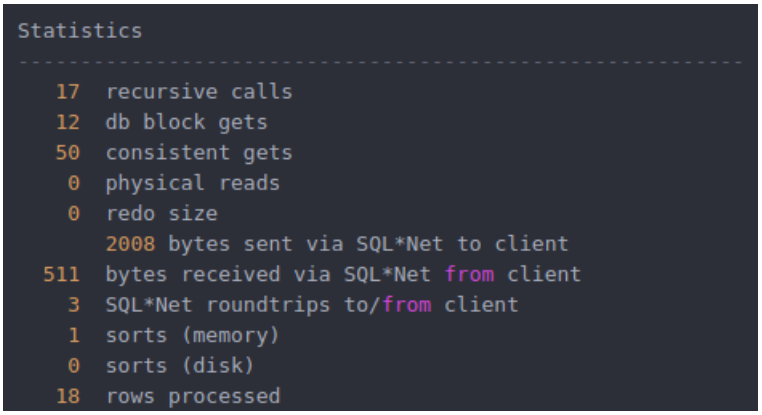
2.2 Question 6

Dans la requête précédente, nous avons utlisé les algorithmes *Index Unique Scan* puis *Nested Loops* lors de l'opération de jointure sur les tables `Ville` et `Departement`, vu que la jointure se fait sur les attributs clés primaires, respectivement **insee** et **id** de ces deux tables, après avoir sélectionné les tuples vérifiant une condition de sélection (en particulier, la ville ayant le code **insee** 34172).



```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------------------------------------------
Plan hash value: 1182727008

--------------------------------------------------------------------------------
| Id  | Operation          | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |              |     1 |    39 |     3   (0)| 00:00:01 |
|   1 |  NESTED LOOPS      |              |     1 |    39 |     3   (0)| 00:00:01 |
|   2 |   TABLE ACCESS BY INDEX ROWID| VILLE        |     1 |     8 |     2   (0)| 00:00:01 |
|*  3 |    INDEX UNIQUE SCAN       | PK_VILLE     |     1 |       |     1   (0)| 00:00:01 |
|   4 |   TABLE ACCESS BY INDEX ROWID| DEPARTEMENT  |    82 |  2542 |     1   (0)| 00:00:01 |
|*  5 |    INDEX UNIQUE SCAN       | SYS_C00388842 |     1 |       |     0   (0)| 00:00:01 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------------------------------------------
--------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - access("INSEE"='34172')
   5 - access("DEPARTEMENT"."ID"="VILLE"."DEP")

18 rows selected.
```

FIGURE 3 – Plan d'exécution



```
Statistics
----------------------------------------------------------
   17  recursive calls
   12  db block gets
   50  consistent gets
    0  physical reads
    0  redo size
       2008 bytes sent via SQL*Net to client
  511  bytes received via SQL*Net from client
    3  SQL*Net roundtrips to/from client
    1  sorts (memory)
    0  sorts (disk)
   18  rows processed
```

FIGURE 4 – Statistiques

D'autre part, la requête courante ne spécifie aucune condition de sélection précédant la jointure sur les deux tables et utilise ainsi, respectivement, les

algorithmes *Table Access Full* pour balayer les tables et *Hash Join* pour faire la jointure.

```
1  explain plan for select departement.nom from
      departement, ville where departement.id = ville.dep;
2  select plan_table_output from
      table(dbms_xplan.display());
```

Listing 4 – plan d'exécution choisi par l'optimiseur pour la requête permettant d'afficher le nom du département pour toutes les villes

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------------------------------------------
Plan hash value: 211249738

--------------------------------------------------------------------------------
| Id  | Operation          | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |            | 40241 |  1375K|    72   (2)| 00:00:01 |
|*  1 |  HASH JOIN         |            | 40241 |  1375K|    72   (2)| 00:00:01 |
|   2 |   TABLE ACCESS FULL| DEPARTEMENT |   104 |  3224 |     3   (0)| 00:00:01 |
|   3 |   TABLE ACCESS FULL| VILLE      | 40241 |   157K|    68   (0)| 00:00:01 |
--------------------------------------------------------------------------------


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("DEPARTEMENT"."ID"="VILLE"."DEP")

Note
-----
   - dynamic sampling used for this statement (level=2)

19 rows selected.
```

FIGURE 5 – Plan d'exécution

2.3 Question 7

```
1  explain plan for select ville.nom, departement.nom
      from ville, departement where departement.id='91'
      and ville.dep=departement.id;
```

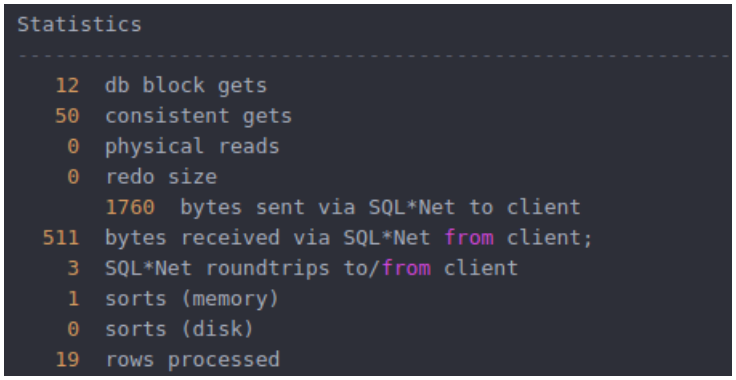


FIGURE 6 – Statistiques

```
2   select plan_table_output from
       table(dbms_xplan.display());
```

Listing 5 – plan d'exécution choisi par l'optimiseur pour la requête permettant d'afficher le nom des villes et du département dont le numéro est 91 (id)

3 Modification du comportement de l'optimiseur

3.1 Question 8

4 Utilisation d'index

4.1 Question 9

4.2 Question 10

4.3 Question 12

4.4 Question 13

5 Les statistiques des tables

5.1 Question 14

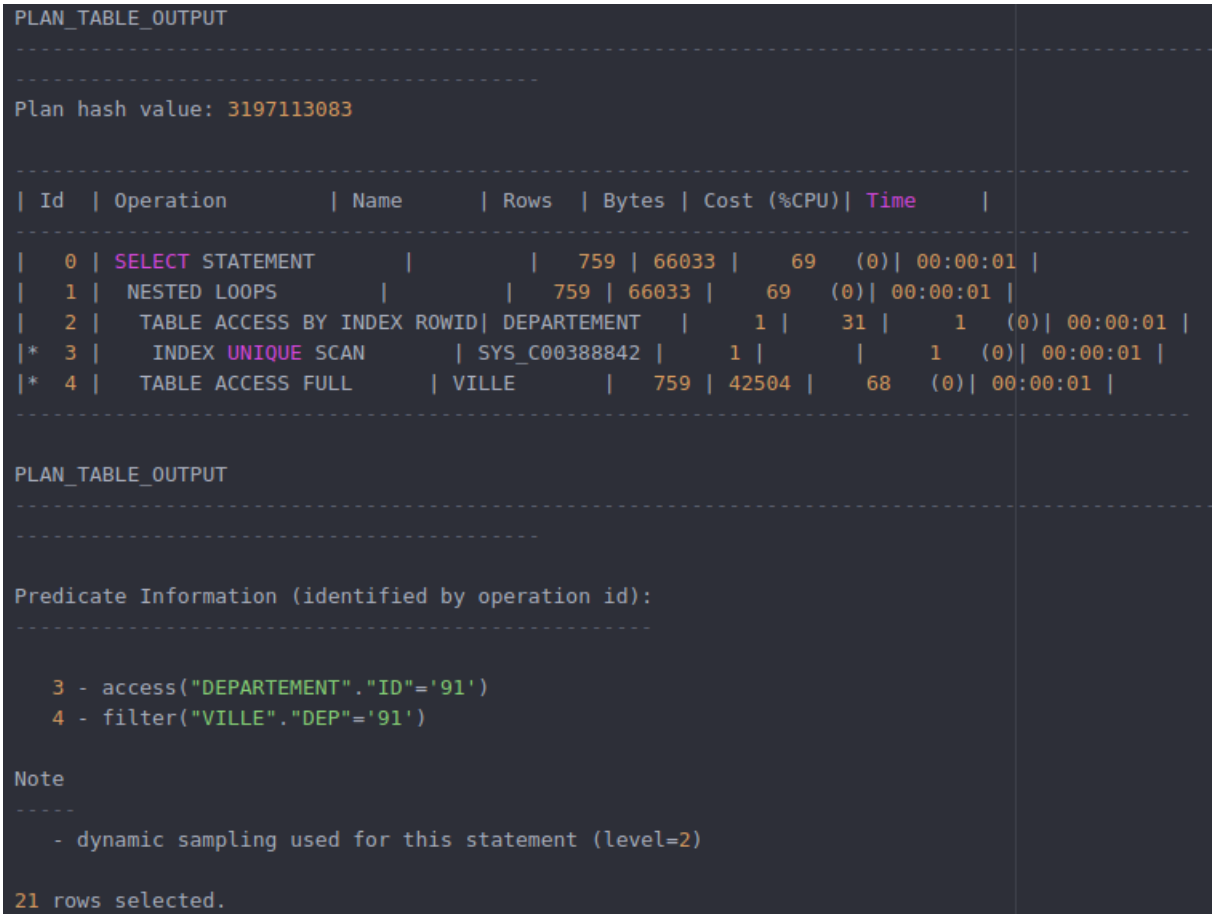


FIGURE 7 – Plan d'exécution

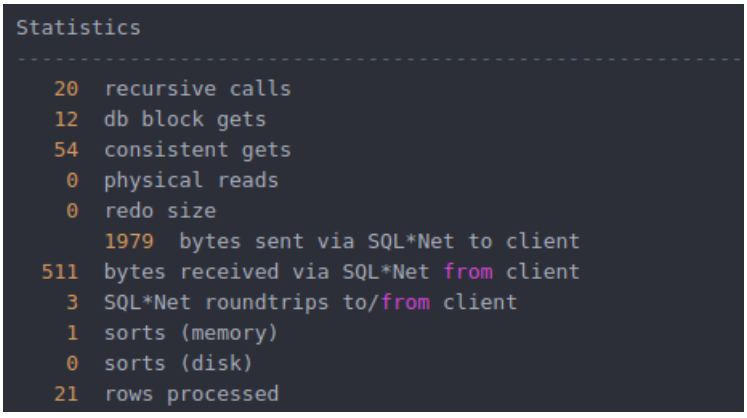


FIGURE 8 – Statistiques