Rapport Entrepôt de Données et Big Data
TP 2 - Optimisation de requête

FACULTÉ DES SCIENCES
DE MONTPELLIER

Morgan NAVEL, Eric GILLES

10 octobre 2023

Table des matières

Partie 1

Sujet du TP Partie 1

1.1 Coût de plans d'exécution logiques

Soit le modèle relationnel composé des relations suivantes :

- ETUDIANTS(IDE, NOM, AGE) – la relation contenant tous les étudiants
- MODULES(IDM, RESPONSABLE, INTITULE) – la relation contenant tous les modules
- IP(#IDE, #IDM) – la relation contenant la liste des inscriptions pédagogiques (inscription d'un étudiant à un module)
- FORMATION(IDF, NOMF) – la relation contenant toutes les formations
- IA (#IDE, #IDF) – la relation contenant la liste des inscriptions administratives (inscription d'un étudiant à une formation)

```
SELECT NOM
FROM ETUDIANTS E ,MODULES M ,IP I
WHERE E.IDE = I.IDE AND M.IDM=I.IDM
AND INTITULE = "EDBD";
```

Question 1 : Que permet d'obtenir la requête ci-dessus ?

La requête permet d'obtenir les noms des étudiants inscrits au module intitulé "EDBD".

Question 2 : Pour chaque plan d'exécution logique, calculer le coût E/S (en termes de nombre de lignes)

Les hypothèses sur les données comprennent les informations suivantes : 200 étudiants, 70 modules, un total de 4 200 inscriptions pédagogiques, dont 10% sont liées au module EDBD, 50 formations et 250 inscriptions administratives, sachant que certains étudiants peuvent être inscrits à plusieurs formations.

```
Plan 1 :
    Jointure entre Module et IP :
    on lit 70 * 4 200 = 294 000 lignes et on produit 4 200 lignes.
    Jointure entre ETUDIANTS et resultat de la jointure précédente :
    on lit 4 200 * 200 = 840 000 lignes et on produit 4 200 lignes.
    Sélection : on lit 4 200 lignes et on produit 4 200 * 10% = 420 lignes.

    Cout(E/S) = 294 000 + 4 200 + 840 000 + 4 200 + 4 200 + 420 = 1 147 020 lignes

Plan 2 :
    Sélection sur Module : on lit 70 lignes et on produit une ligne.
    Jointure entre Module et IP :
    on lit 1 * 4 200 = 4 200 lignes et on produit 4200 * 10% = 420 lignes.
    Jointure entre ETUDIANTS et resultat de la jointure précédente :
    on lit 420 * 200 = 84 000 lignes et on produit 420 lignes.

    Cout(E/S) = 70 + 1 + 4 200 + 420 + 84 000 + 420 = 89 111 lignes

Plan 3 :
    Jointure entre Module et Etudiants :
    on lit 70 * 200 = 14 000 lignes et on produit 14 000 lignes.
    Jointure entre IP et resultat de la jointure précédente :
    on lit 4 200 * 14000 = 58 800 000 lignes et on produit 4 200 lignes.
    Sélection : on lit 4 200 lignes et on produit 4 200 * 10% = 420 lignes.

    Cout(E/S) = 14 000 + 14 000 + 58 800 000 + 4 200 + 4 200 + 420 = 58 836 820 lignes
```

Question 3 : Quel est le plan d'exécution logique optimal parmi les plans proposés ? Pourquoi ?

Le plan d'exécution logique le plus optimal est le plan n°2 car il a le plus petit coût E/S en nombre de lignes grâce à la sélection au début, ce qui réduit le nombre de lignes lues et traitées par la suite lors des jointures de la requête.

1.2 Définition de plans d'exécution logiques

Requête 1 :

```
SELECT RESPONSABLE
FROM ETUDIANTS E, MODULES M, IP I
WHERE E.IDE = I.IDE AND M.IDM=I.IDM
AND NOM = "DUPOND" AND INTITULE LIKE "HAI%" ;
```

Cette requête permet d'obtenir les responsables des modules avec un intitulé commençant par "HAI%" et dans lesquels un étudiant nommé "DUPOND" est inscrit.

Parmi les trois plans d'exécutions logiques ci-dessous, le plus optimal est le plan 3, car les sélections se font avant les jointures, ce qui réduit le nombre de lignes lues et traitées.

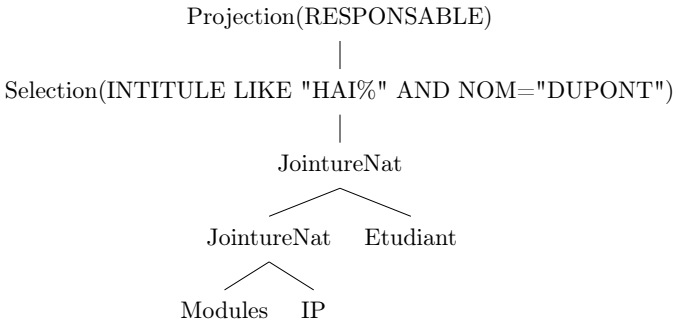

FIGURE 1.1 – Exercice 2 Requête 1 : plan 1

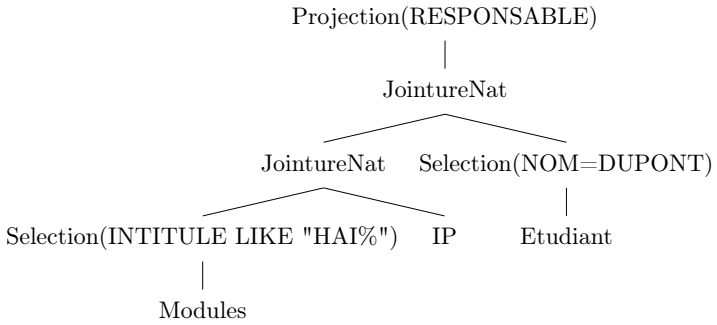

FIGURE 1.2 – Exercice 2 Requête 1 : plan 2

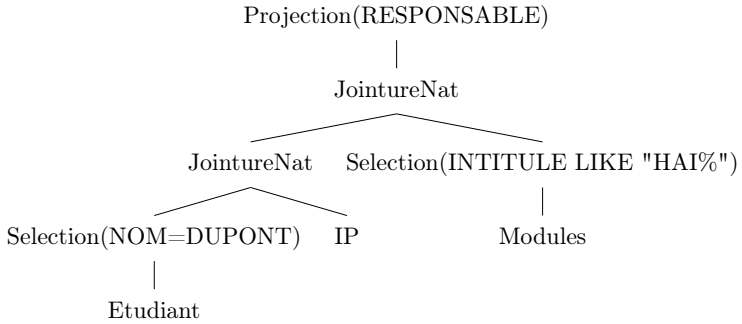

FIGURE 1.3 – Exercice 2 Requête 1 : plan 3

Requête 2 :

```
SELECT NOM
FROM ETUDIANTS E ,FORMATION F ,IA A, IP I, MODULE M
WHERE E.IDE = A.IDE AND F.IDF=A.IDF AND I.IDE=E.IDE AND M.IDM=I.IDM,
AND NOMF = "MASTER GL" AND INTITULE = "EDBD" ;
```

Cette requête permet d'obtenir le nom des étudiants inscrits administrativement et pédagogiquement au master GL et au module EDBD.

Parmi les trois plans d'exécutions logiques ci-dessous, le plus optimal est le plan 3, car les sélections se font avant les jointures avec une parallélisation des opérations de jointures. Mais la jointure entre IA et IP se fait sur la clé primaire d'étudiant(clé étrangère dans IA et IP) ce qui n'est pas intuitif. Le plan 2, non parallélisé, semble presque aussi efficace et plus intuitif.

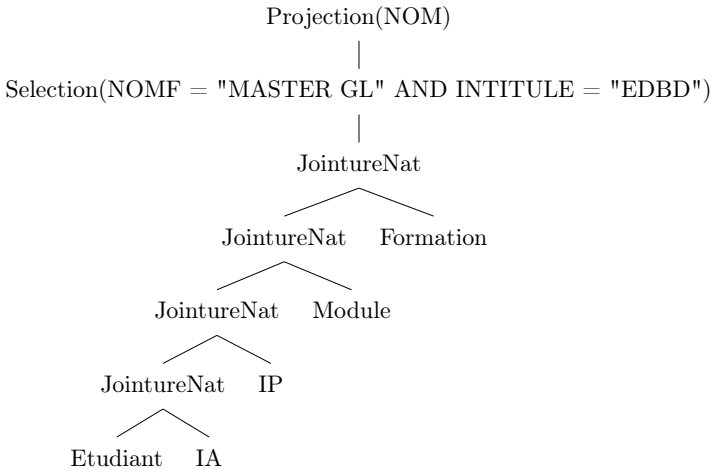

FIGURE 1.4 – Exercice 2 Requête 2 : plan 1

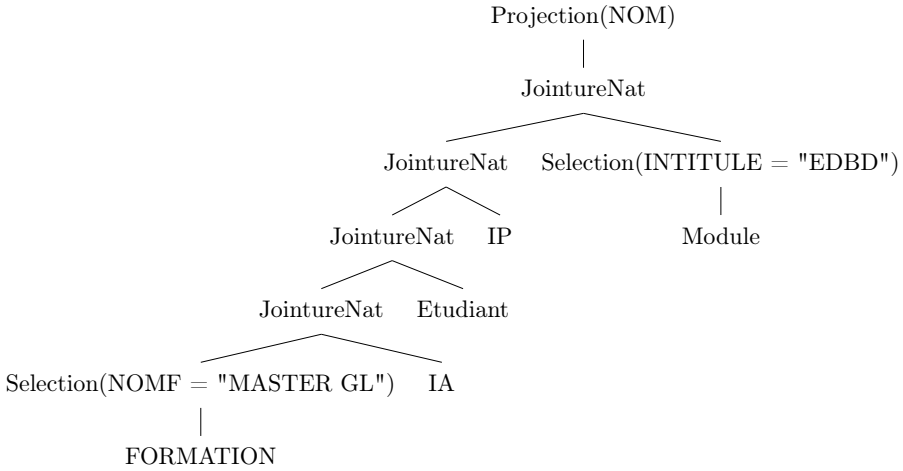

FIGURE 1.5 – Exercice 2 Requête 2 : plan 2

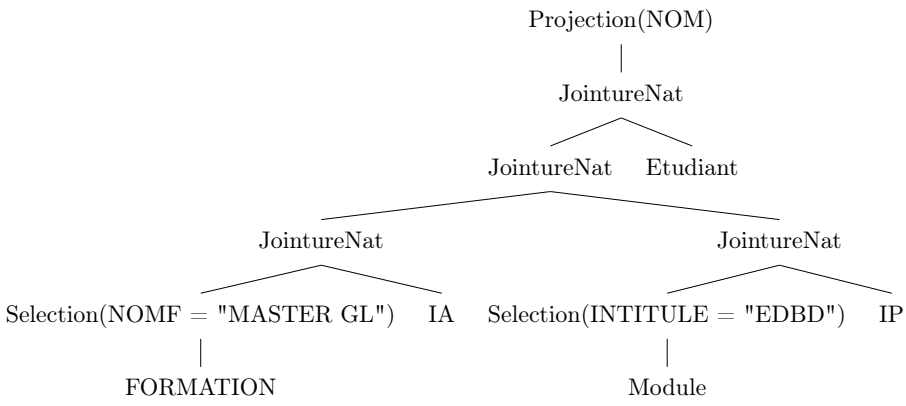

FIGURE 1.6 – Exercice 2 Requête 2 : plan 3

1.3 Réécriture de plans d'exécution logiques

Soit le schéma relationnel suivant :

— JOURNALISTE (IDJ, NOM, PRENOM)
La relation contenant tous les journalistes

— JOURNAL (TITRE, REDACTION, #REDACTEUR_ID)
La relation contenant tous les journaux rédigés par des journalistes

On considère la requête suivante :

```
SELECT NOM
FROM JOURNAL, JOURNALISTE
WHERE TITRE='Le Monde' AND IDJ=REDACTEUR_ID AND PRENOM='Jean';
```

Question 1 : Les deux expressions retournent-elles le même résultat (sont-elles équivalentes) ? Justifiez votre réponse en indiquant les règles de réécriture que l'on peut appliquer

Oui, elles sont équivalentes, car on peut descendre ou monter les conditions sur les sélections et on retrouve les mêmes expressions.

Passage de la 1re à la 2nde expression avec les règles de réécriture :

— Regroupement des Sélections

— Commutativité entre Sélections et Jointure

Question 2 : Une expression vous semble-t-elle meilleure que l'autre si on les considère comme des plans d'exécution ?

Si on considère les expressions comme des plans d'exécution, la 2nde expression semble meilleure, car la jointure se fait sur des tables réduites.

1.4 Tous les plans d'exécution logiques

Soit le modèle relationnel suivant :

— ACTEUR (idA, nom, prenom, nationalite)
la relation contenant tous les acteurs

— FILM (idF, titre, annee, nbspectateurs, #idRealisateurs, #idGenre)
la relation contenant tous les films

— JOUER (#idActeur,#idFilm, salaire)
la relation contenant la liste des acteurs et des films dans lesquels ils jouent

— REALISATEUR (idR, nom, prenom, nationalite)
la relation contenant tous les réalisateurs

— GENRE (idG, description)
la relation contenant tous les genres des films (horeur, comédie, ...)

```
SELECT acteur.nom,acteur.prenom
FROM acteur,jouer,film,genre, realisateur
WHERE (idA=idActeur) AND (idFilm=idF) AND (idGenre=idG) AND (idRealisateur=idR)
AND (acteur.nationalite='France') AND (description='comédie')
AND (realisateur.nom = "Les frères Coen");
```

Question 1 : Pour la requête ci-dessus, donner 2 plans d'exécution logiques : un premier plan fera intervenir les jointures en premier et un deuxième commencera par les sélections.

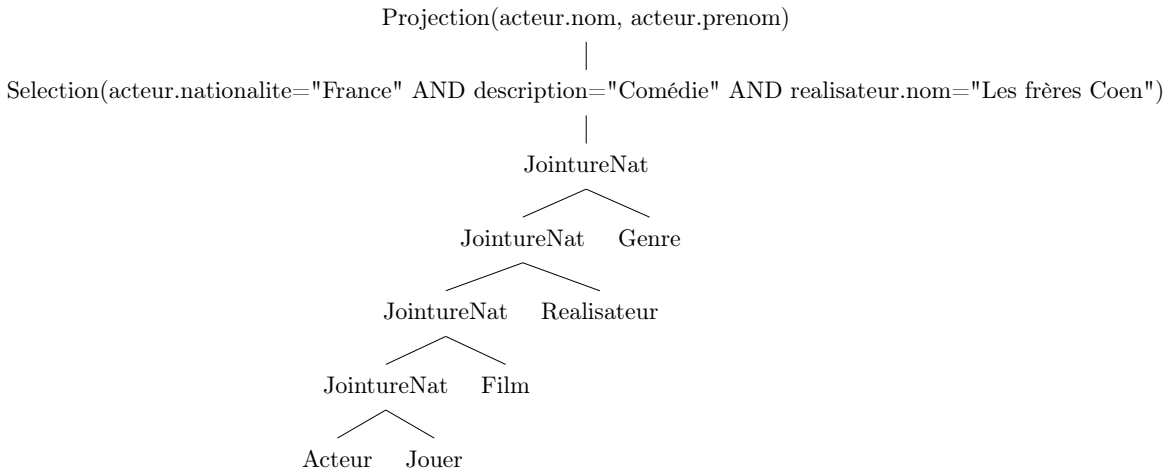

FIGURE 1.7 – Exercice 4 Plan 1

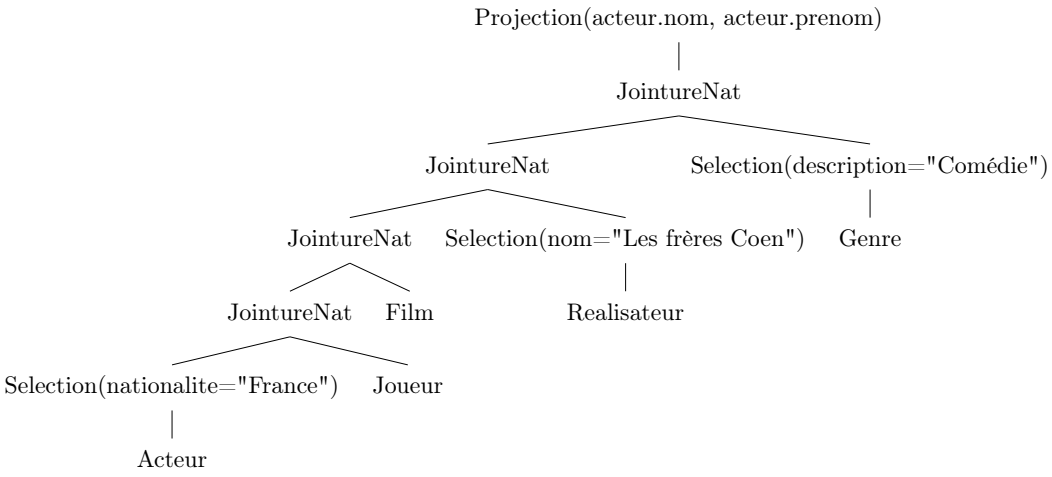

FIGURE 1.8 – Exercice 4 Plan 2

Question 2 : Parmi les plans d'exécution proposés, quel est le plan optimal ? Justifier votre choix.

Le 1er plan est le plus optimal, car la jointure entre Acteur et Jouer (10% des acteurs jouent dans des films) sera plus restrictive que la sélection des acteurs français (30% des acteurs sont français), bien que les sélections soient placées avant les jointures dans le 2nd plan.

Partie 2

Sujet du TP Partie 2

2.1 Les plans d'exécution sous ORACLE

Tous les plans d'exécutions et les statistiques exécutés par l'optimiseur d'oracle pour les différentes questions du TP se trouve en Annexe.

2.1.1 Sélection

Question 1 : Examinez les scripts pour comprendre ce qu'ils font

Le premier script 'script_table' est pour la création de 3 tables ville, région et département avec les relations suivantes : ville en relation avec région et région avec département.
Le second script 'script_remplissage' est pour le remplissage des tables, précédemment créées, avec des données.

Question 2 : Explicitez le plan d'exécution choisi par l'optimiseur et les statistiques obtenues lors de l'exécution d'une requête permettant d'afficher le nom des villes dont le numéro insee est 1293 : testez avec insee=1293 et insee='1293', puis comparez.

```
SELECT nom FROM ville WHERE insee=1293;

SELECT nom FROM ville WHERE insee='1293';
```

Le plan d'exécution choisi pour les deux requêtes pour l'accès aux données comprend l'opération de parcours séquentiel de la table (TABLE ACCESS FULL), ce qui demande un accès au disque très important.

Question 3 : Ajoutez une clé primaire sur la table ville (utiliser l'attribut insee).

```
ALTER TABLE ville
ADD CONSTRAINT PK_ville PRIMARY KEY (insee);
```

Question 4 : Explicitez à nouveau le plan d'exécution choisi par l'optimiseur et les statistiques obtenues lors de l'exécution de la requête précédente (afficher le nom des villes dont le numéro insee est 1293 en testant insee=1293 et insee='1293'). Quelles sont les différences observées par rapport à la Question 2 ?

Le plan d'exécution est le même que la question 2 pour la requête avec insee=1293, car l'index demandé n'existe pas. Pour la seconde requête, on remarque une différence d'accès aux données qui se font via l'opération INDEX SCAN, ce qui permet une recherche plus rapide.

2.1.2 Jointure

Question 5 : Explicitez maintenant le plan d'exécution choisi par l'optimiseur et les statistiques obtenues lors de l'exécution d'une requête permettant d'afficher le nom du département pour la ville dont le numéro insee est insee='1293'.

```
SELECT departement.nom FROM departement
JOIN ville ON departement.id=ville.dep
WHERE insee='1293';
```

Le plan d'exécution choisi par l'optimiseur implique une jointure imbriquée (NESTED LOOP) entre les tables VILLE et DEPARTEMENT et utilise deux accès par index (un index lié à la colonne insee de VILLE et un index lié à la condition de jointure entre les deux tables).

Question 6 : Faites de même avec la requête permettant d'afficher le nom des départements de toutes les villes.
Quelles sont les différences observées par rapport à la Question 5 ?

```
SELECT departement.nom FROM departement
JOIN ville ON departement.id=ville.dep;
```

Le plan choisi pour cette requête implique une jointure de hachage (HASH JOIN) entre les tables VILLE et DEPARTEMENT et le parcours séquentiel des deux tables (TABLE ACCESS FULL).

Les différences par rapport au plan de la question précédente sont une jointure de hachage (HASH JOIN) et un accès complet (TABLE ACCESS FULL) aux données des tables, ce qui entraine un coût plus élevé par rapport à une jointure imbriquée (NESTED LOOP) et un accès aux données via des index (INDEX SCAN).

2.1.3 Modification du comportement de l'optimiseur

Question 7 : Essayez maintenant la requête de la Q 6, mais en forçant l'utilisation de boucles imbriquées (nested loops par la directive "/*+ use_nl(table1 table2) */") et explicitez le plan d'exécution choisi par l'optimiseur et les statistiques obtenues

```
SELECT /*+ use\_nl(departement ville) */
departement.nom FROM departement
JOIN ville ON departement.id=ville.dep;
```

En forçant l'utilisation de jointure imbriquée (NESTED LOOP), on augmente par 10 le nombre de "consistent gets" (accès aux blocs de données) et par 100 le coût CPU, c'est pourquoi l'utilisation de jointure de hachage (HASH JOIN) a été choisi par l'optimiseur pour afficher la requête.

2.1.4 Utilisation d'index

Question 8 : Créer un index secondaire sur l'attribut dep de la table ville : create index idx_dep_ville on ville (dep). Ré-exécutez les requêtes des Qs 5 et 6 et explicitez le plan d'exécution choisi par l'optimiseur et les statistiques obtenues.

```
CREATE INDEX idx_dep_ville ON ville(dep);
```

Pour la requête de la question 5, le plan est choisi par l'optimiseur est le même et la création de l'index ne change rien.

Pour la requête de la question 6, On remarque ici une grosse amélioration du coût de la requête. On passe de 73% à 26% avec l'ajout de l'index (INDEX SCAN). L'index permet de donner le bloc dans lequel se trouve la donnée, ce qui permet un accès plus rapide et une utilisation moins massive du CPU.

Question 9 : Exécutez la requête suivante et explicitez le plan d'exécution choisi par l'optimiseur et les statistiques obtenues : afficher le nom des villes, de leurs départements et de leurs régions.

```
SELECT departement.nom, ville.nom, region.nom FROM ville
JOIN departement ON ville.dep=departement.id
JOIN region ON region.id=departement.reg;
```

Le plan choisi par l'optimiseur contient deux jointure par hachage (HASH JOIN) et des accès aux données en parcours séquentiel des trois tables (TABLE ACCESS FULL).

Question 10 : Créer un index secondaire sur l'attribut reg de la table département. Ré-exécutez la requête précédente et explicitez le plan d'exécution choisi par l'optimiseur et les statistiques obtenues.

```
CREATE INDEX idx_dep_reg ON departement(reg);
```

Le plan choisi par l'optimiseur contient une jointure par hachage (HASH JOIN), une jointure par tri-fusion (MERGE SORT JOIN), des accès aux données en parcours séquentiel des tables (TABLE ACCESS FULL) région et ville et un accès aux données par index pour la table departement.

Question 11 : Exécutez maintenant la requête suivante : afficher le nom des villes, de leurs départements et de la région pour la région dont le numéro (id) est 91. Explicitez le plan d'exécution choisi par l'optimiseur et les statistiques obtenues.

```
SELECT departement.nom, ville.nom, region.nom FROM ville
JOIN departement ON ville.dep=departement.id
JOIN region ON region.id=departement.reg
WHERE region.id='91';
```

Ici, on remarque l'utilisation de trois jointures imbriquées (NESTED LOOP) et d'accès aux données tables par index (INDEX SCAN).

Question 12 : Exécutez maintenant la requête suivante : afficher le nom des villes dont le numéro de département (dep) commence par '7'. Explicitez le plan d'exécution choisi par l'optimiseur et les statistiques obtenues. Qu'en est-il de l'utilisation de l'index secondaire ?

```
SELECT nom FROM ville WHERE dep LIKE '7%';
```

On voit que l'optimiseur se sert des index que nous avons précédemment créés, cela permet de réduire le coût, mais il effectue ensuite depuis le bloc de l'index un scan qui est assez couteux.

2.1.5 Les statistiques des tables

Question 13 : Regardez les données disponibles dans la table USER_TAB_COL_STATISTICS pour les tables précédentes. Est-ce que les statistiques correspondent bien aux données présentes dans vos tables ?

Question 14 :

```
exec dbms_stats.gather_table_stats('login', 'ville') ;
exec dbms_stats.gather_table_stats('login', 'departement') ;
exec dbms_stats.gather_table_stats('login', 'region') ;
exec dbms_stats.gather_schema_stats('login');
```

Après exécution des commandes,

2.2 Annexe

```
Plan d'execution
----------------------------------------------------------
Plan hash value: 2371920588

-------------------------------------------------------------------------------
| Id  | Operation         | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |       |     4 |   224 |    69   (2)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| VILLE |     4 |   224 |    69   (2)| 00:00:01 |
-------------------------------------------------------------------------------

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

   1 - filter(TO_NUMBER("INSEE")=1293)

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)


Statistiques
----------------------------------------------------------
         58  recursive calls
         10  db block gets
         92  consistent gets
          0  physical reads
        960  redo size
        554  bytes sent via SQL*Net to client
         52  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          0  rows processed
```

FIGURE 2.1 – Plan d'exécution 1 Question 2

```
Plan d'execution
----------------------------------------------------------
Plan hash value: 2371920588

--------------------------------------------------------------------------------
| Id  | Operation         | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |       |     4 |   224 |    68   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| VILLE |     4 |   224 |    68   (0)| 00:00:01 |
--------------------------------------------------------------------------------

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

   1 - filter("INSEE"='1293')

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)


Statistiques
----------------------------------------------------------
         13  recursive calls
          0  db block gets
        277  consistent gets
         98  physical reads
       1584  redo size
        560  bytes sent via SQL*Net to client
         52  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

FIGURE 2.2 – Plan d'exécution 2 Question 2

```
Plan d'execution
----------------------------------------------------------
Plan hash value: 2371920588

--------------------------------------------------------------------------------
| Id  | Operation         | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |       |     1 |    56 |    69   (2)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| VILLE |     1 |    56 |    69   (2)| 00:00:01 |
--------------------------------------------------------------------------------

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

   1 - filter(TO_NUMBER("INSEE")=1293)

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)


Statistiques
----------------------------------------------------------
         49  recursive calls
          4  db block gets
         97  consistent gets
          0  physical reads
          0  redo size
        554  bytes sent via SQL*Net to client
         52  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          0  rows processed
```

FIGURE 2.3 – Plan d'exécution 1 Question 4

```
Plan d'execution
----------------------------------------------------------
Plan hash value: 781934248

-------------------------------------------------------------------------------
-------------

| Id  | Operation                    | Name          | Rows  | Bytes | Cost (%CPU) | Time     |

-------------------------------------------------------------------------------------------

|   0 | SELECT STATEMENT             |               |     1 |    56 |     2   (0) | 00:0:01 |

|   1 |  TABLE ACCESS BY INDEX ROWID | VILLE         |     1 |    56 |     2   (0) | 00:00:01 |

|*  2 |   INDEX UNIQUE SCAN          | SYS_C00443099 |     1 |       |     1   (0) | 00:00:01 |

-------------------------------------------------------------------------------------------


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

   2 - access("INSEE"='1293')


Statistiques
----------------------------------------------------------
         43  recursive calls
          0  db block gets
         51  consistent gets
          1  physical reads
          0  redo size
        427  bytes sent via SQL*Net to client
         41  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
          6  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

FIGURE 2.4 – Plan d'exécution 2 Question 4

```
Plan d'execution
----------------------------------------------------------
Plan hash value: 1247517613

-------------------------------------------------------------------------------------------------

| 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   |     1 |    31 |     1   (0) | 00:00:01 |

|*  5 |    INDEX UNIQUE SCAN         | SYS_C00442871 |     1 |       |     0   (0) | 00:00:01 |

-------------------------------------------------------------------------------------------------


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

   3 - access("V"."INSEE"='1293')
   5 - access("V"."DEP"="D"."ID")


Statistiques
----------------------------------------------------------
         29  recursive calls
          6  db block gets
         45  consistent gets
          3  physical reads
       1004  redo size
        556  bytes sent via SQL*Net to client
         52  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          4  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

FIGURE 2.5 – Plan d'exécution Question 5

```
Plan d'execution
----------------------------------------------------------
Plan hash value: 211249738

-------------------------------------------------------------------------------------

| Id  | Operation          | Name        | Rows  | Bytes | Cost (%CPU) | Time |

-------------------------------------------------------------------------------------

|   0 | SELECT STATEMENT   |             | 27855 |   952K|    72   (2) | 00:00:01 |

|*  1 |  HASH JOIN         |             | 27855 |   952K|    72   (2) | 00:00:01 |

|   2 |   TABLE ACCESS FULL| DEPARTEMENT |   104 |  3224 |     3   (0) | 00:00:01 |

|   3 |   TABLE ACCESS FULL| VILLE       | 27855 |   108K|    68   (0) | 00:00:01 |

-------------------------------------------------------------------------------------


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

   1 - access("V"."DEP"="D"."ID")

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)


Statistiques
----------------------------------------------------------
         36  recursive calls
          5  db block gets
       2748  consistent gets
         92  physical reads
       1036  redo size
     654200  bytes sent via SQL*Net to client
      26892  bytes received via SQL*Net from client
       2442  SQL*Net roundtrips to/from client
          4  sorts (memory)
          0  sorts (disk)
      36601  rows processed
```

FIGURE 2.6 – Plan d'exécution Question 6

```
Plan d'execution
----------------------------------------------------------
Plan hash value: 1651012225

-------------------------------------------------------------------------------------

| Id  | Operation          | Name        | Rows  | Bytes | Cost (%CPU)| Time |

-------------------------------------------------------------------------------
--

|   0 | SELECT STATEMENT   |             | 25647 |   876K|  6922   (1)| 00:00:01 |

|   1 |  NESTED LOOPS      |             | 25647 |   876K|  6922   (1)| 00:00:01 |

|   2 |   TABLE ACCESS FULL| DEPARTEMENT |   104 |  3224 |     3   (0)| 00:00:01 |

|*  3 |   TABLE ACCESS FULL| VILLE       |   247 |   988 |    67   (2)| 00:00:01 |

-------------------------------------------------------------------------------------


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

   3 - filter("DEPARTEMENT"."ID"="VILLE"."DEP")

Hint Report (identified by operation id / Query Block Name / Object Alias):
Total hints for statement: 1 (U - Unused (1))
---------------------------------------------------------------------------

   2 -  SEL$58A6D7F6 / DEPARTEMENT@SEL$1
         U -  use_nl(departement ville)

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)


Statistiques
----------------------------------------------------------
          7  recursive calls
          0  db block gets
      22404  consistent gets
          0  physical reads
          0  redo size
     650629  bytes sent via SQL*Net to client
      26892  bytes received via SQL*Net from client
       2442  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
      36601  rows processed
```

FIGURE 2.7 – Plan d'exécution Question 7

```
Plan d'execution
----------------------------------------------------------
Plan hash value: 1247517613

-------------------------------------------------------------------------------------------------
| 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   |     1 |    31 |     1   (0)| 00:00:01 |
|*  5 |    INDEX UNIQUE SCAN         | SYS_C00442871 |     1 |       |     0   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------

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

   3 - access("V"."INSEE"='1293')
   5 - access("V"."DEP"="D"."ID")


Statistiques
----------------------------------------------------------
         69  recursive calls
          0  db block gets
        163  consistent gets
          4  physical reads
          0  redo size
        556  bytes sent via SQL*Net to client
         52  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
         12  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

FIGURE 2.8 – Plan d'exécution 1 Question 8

```
Plan d'execution
----------------------------------------------------------
Plan hash value: 3151218067

-------------------------------------------------------------------------------------------
| Id  | Operation             | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |               | 27855 |   952K|    26   (0)| 00:00:01 |
|*  1 |  HASH JOIN            |               | 27855 |   952K|    26   (0)| 00:00:01 |
|   2 |   TABLE ACCESS FULL   | DEPARTEMENT   |   104 |  3224 |     3   (0)| 00:00:01 |
|   3 |   INDEX FAST FULL SCAN| IDX_DEP_VILLE | 27855 |   108K|    23   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

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

   1 - access("V"."DEP"="D"."ID")

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
   - this is an adaptive plan


Statistiques
----------------------------------------------------------
         20  recursive calls
          0  db block gets
       2613  consistent gets
         80  physical reads
          0  redo size
     650624  bytes sent via SQL*Net to client
      26892  bytes received via SQL*Net from client
       2442  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
      36601  rows processed
```

FIGURE 2.9 – Plan d'exécution 2 Question 8

```
    Plan d'execution
----------------------------------------------------------
Plan hash value: 424771235

-------------------------------------------------------------------------------------
| Id  | Operation           | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |             | 27855 |  3808K|    75   (2)| 00:00:01 |
|*  1 |  HASH JOIN          |             | 27855 |  3808K|    75   (2)| 00:00:01 |
|*  2 |   HASH JOIN         |             |   104 |  8736 |     6   (0)| 00:00:01 |
|   3 |    TABLE ACCESS FULL| REGION      |    27 |  1080 |     3   (0)| 00:00:01 |
|   4 |    TABLE ACCESS FULL| DEPARTEMENT |   104 |  4576 |     3   (0)| 00:00:01 |
|   5 |   TABLE ACCESS FULL | VILLE       | 27855 |  1523K|    68   (0)| 00:00:01 |
-------------------------------------------------------------------------------------

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

   1 - access("D"."ID"="V"."DEP")
   2 - access("D"."REG"="R"."ID")

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
   - this is an adaptive plan


Statistiques
----------------------------------------------------------
        130  recursive calls
         17  db block gets
       2746  consistent gets
        100  physical reads
       2856  redo size
    1115643  bytes sent via SQL*Net to client
      26892  bytes received via SQL*Net from client
       2442  SQL*Net roundtrips to/from client
          8  sorts (memory)
          0  sorts (disk)
      36601  rows processed
```

FIGURE 2.10 – Plan d'exécution Question 9

```
    Plan d'execution
----------------------------------------------------------
Plan hash value: 1980903393

-------------------------------------------------------------------------------------------------

| Id  | Operation                     | Name        | Rows  | Bytes | Cost (%CPU)| Time     |

-------------------------------------------------------------------------------------------------

|   0 | SELECT STATEMENT              |             | 25647 |  3506K|    75   (3)| 00:00:01 |

|*  1 |  HASH JOIN                    |             | 25647 |  3506K|    75   (3)| 00:00:01 |

|   2 |   MERGE JOIN                  |             |   104 |  8736 |     6  (17)| 00:00:01 |

|   3 |    TABLE ACCESS BY INDEX ROWID| DEPARTEMENT |   104 |  4576 |     2   (0)| 00:00:01 |

|   4 |     INDEX FULL SCAN           | IDX_DEP_REG |   104 |       |     1   (0)| 00:00:01 |

|*  5 |    SORT JOIN                  |             |    27 |  1080 |     4  (25)| 00:00:01 |

|   6 |     TABLE ACCESS FULL         | REGION      |    27 |  1080 |     3   (0)| 00:00:01 |

|   7 |   TABLE ACCESS FULL           | VILLE       | 25647 |  1402K|    68   (0)| 00:00:01 |

-------------------------------------------------------------------------------------------------


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

   1 - access("VILLE"."DEP"="DEPARTEMENT"."ID")
   5 - access("REGION"."ID"="DEPARTEMENT"."REG")
       filter("REGION"."ID"="DEPARTEMENT"."REG")

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)


Statistiques
----------------------------------------------------------
         67  recursive calls
         15  db block gets
       2704  consistent gets
         11  physical reads
       3076  redo size
    1114951  bytes sent via SQL*Net to client
      26892  bytes received via SQL*Net from client
       2442  SQL*Net roundtrips to/from client
          8  sorts (memory)
          0  sorts (disk)
      36601  rows processed
```

FIGURE 2.11 – Plan d'exécution Question 10

```
Plan d'execution
----------------------------------------------------------
Plan hash value: 4285023071

-------------------------------------------------------------------------------------------------------
| Id  | Operation                                | Name          | Rows  | Bytes | Cost (%CPU)|Time |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |               |  2842 |   388K| 22  (0)|00:00:01 |
|   1 |  NESTED LOOPS                            |               |  2842 |   388K| 22  (0)|00:00:01 |
|   2 |   NESTED LOOPS                           |               |  2842 |   388K| 22  (0)|00:00:01 |
|   3 |    NESTED LOOPS                          |               |     5 |   420 | 3   (0)|00:00:01 |
|   4 |     TABLE ACCESS BY INDEX ROWID          | REGION        |     1 |    40 | 2   (0)|00:00:01 |
|*  5 |      INDEX UNIQUE SCAN                   | SYS_C00442870 |     1 |       | 1   (0)|00:00:01 |
|   6 |     TABLE ACCESS BY INDEX ROWID BATCHED  | DEPARTEMENT   |     5 |   220 | 1   (0)|00:00:01 |
|*  7 |      INDEX RANGE SCAN                    | IDX_DEP_REG   |     5 |       | 0   (0)|00:00:01 |
|*  8 |    INDEX RANGE SCAN                      | IDX_DEP_VILLE |   568 |       | 1   (0)|00:00:01 |
|   9 |   TABLE ACCESS BY INDEX ROWID            | VILLE         |   568 | 31808 | 7   (0)|00:00:01 |
-------------------------------------------------------------------------------------------------------

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

   5 - access("R"."ID"=91)
   7 - access("D"."REG"=91)
   8 - access("D"."ID"="V"."DEP")

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
   - this is an adaptive plan


Statistiques
----------------------------------------------------------
        110  recursive calls
          0  db block gets
        470  consistent gets
         18  physical reads
          0  redo size
      47645  bytes sent via SQL*Net to client
       1174  bytes received via SQL*Net from client
        104  SQL*Net roundtrips to/from client
         18  sorts (memory)
          0  sorts (disk)
       1543  rows processed
```

FIGURE 2.12 – Plan d'exécution Question 11

```
Plan d'execution
----------------------------------------------------------
Plan hash value: 1694568309

-------------------------------------------------------------------------------------------------
| Id  | Operation                           | Name          | Rows  | Bytes | Cost (%CPU)|Time|
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |               |  3149 |   172K|  40 (0)|00:00:01|
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| VILLE         |  3149 |   172K|  40 (0)|00:00:01|
|*  2 |   INDEX RANGE SCAN                  | IDX_DEP_VILLE |  3149 |       |  11 (0)|00:00:01|
-------------------------------------------------------------------------------------------------

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

   2 - access("V"."DEP" LIKE '7%')
       filter("V"."DEP" LIKE '7%')

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)


Statistiques
----------------------------------------------------------
         49  recursive calls
          0  db block gets
        768  consistent gets
         22  physical reads
          0  redo size
     171967  bytes sent via SQL*Net to client
       3408  bytes received via SQL*Net from client
        287  SQL*Net roundtrips to/from client
          6  sorts (memory)
          0  sorts (disk)
       4278  rows processed
```

FIGURE 2.13 – Plan d'exécution Question 12