TP Optimisation de requêtes dans le contexte d'Oracle

Exercice 1:

Num de requête	Recours à l'index	Raison du choix de l'optimiseur	Opérateurs mis en jeu
1	Oui	accès à la table Commune via l'index commune_pk posé sur l'attribut code_INSEE	INDEX FAST FULL SCAN
2	Non	balayage totale de la table Commune (pas d'index sur nom_com)	TABLE ACCESS FULL
3	Non	balayage totale de la table Commune	TABLE ACCESS FULL
4	Oui	à l'identifianf du tuple unique qui correspond à la valeur code_INSEE='34192'	TABLE ACCESS BY INDEX ROWID INDEX UNIQUE SCAN
5	Oui	accès à la table Commune via l'index commune_pk posé sur l'attribut code_INSEE avec un accès direct aux identifianfs des tuples qui satisfont la condition code_INSEE like '34%'	ABLE ACCESS BY INDEX ROWID INDEX RANGE SCAN
6	Non	balayage totale de la table Commune, l'index commune_pk n'est pas utlisée	TABLE ACCESS FULL
7	Non	balayage totale de la table Commune, l'index commune_pk n'est pas utlisée car il y a une converstion du type de code_INSEE avec la fonction (tonumber())	TABLE ACCESS FULL
8	Oui	accès à la table Commune via l'index commune_pk posé sur l'attribut code_INSEE avec un accès direct aux identifianfs des tuples qui font partie de la collection de valeurs renseignées ('09330','09331','09332','09334')	TABLE ACCESS BY INDEX ROWID INDEX UNIQUE SCAN

Requêtes :

```
/*Q1*/
select code_INSEE from commune ;
select /*+ NO_INDEX(commune) */ code_INSEE from commune ;

/*Q2*/
select nom_com from commune ;
select /*+ NO_INDEX(commune) */ nom_com from commune ;

/*Q3*/
select nom_com, code_INSEE from commune;
select /*+ NO_INDEX(commune) */  nom_com, code_INSEE from commune;

/*Q4*/
select nom_com from commune where code_INSEE='34192' ;
select /*+ NO_INDEX(commune) */ nom_com from commune where code_INSEE='34192' ;

/*Q5*/
select nom_com from commune where code_INSEE like '34%';
select /*+ NO_INDEX(commune) */ nom_com from commune where code_INSEE like '34%' ;

/*Q6*/
select nom_com from commune where code_INSEE like '%392' ;
select /*+ NO_INDEX(commune) */ nom_com from commune where code_INSEE like '%392';

/*Q7*/
select nom_com from commune where code_INSEE >= 34;
select /*+ NO_INDEX(commune) */ nom_com from commune where code_INSEE >= 34 ;

/*Q8*/
select nom_com from commune where code_INSEE in ('09330','09331','09332','09334') ;
select /*+ NO_INDEX(commune) */ nom_com from commune where code_INSEE in
('09330','09331','09332','09334') ;
```

Exercice 2:

Partie 1: Sans créer d'index supplémentaires

Q1. Donnez toutes les informations concernant les communes dont le nom commence par 'MO'

```
Select *
From Commune
Where nom_com like 'MO%';
```

Analyse du Plan d'exécution:

```
-- ------------------------------------------------------------------------------
-- | Id  | Operation                 | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-- ------------------------------------------------------------------------------
-- |   0 | SELECT STATEMENT          |         |   7   |  2513 |   581   (1)| 00:00:07 |
-- |*  1 |  TABLE ACCESS FULL        | COMMUNE |   7   |  2513 |   581   (1)| 00:00:07 |
-- ------------------------------------------------------------------------------
-- 1 - filter("NOM_COM" LIKE 'MO%')
```

Balayage séquentiel de la table Commune pour trouver les tuples qui satisfont la condition de sélection ("NOM_COM" LIKE 'MO%') puis projection du résultat.

Temps d'exécution estimé : 00:00:07

Q2. Donnez les noms et les numéros de département des communes dont la population de 2010 est supérieure à 50 000 habitants

```
Select d.dep, nom_dep
From departement d, commune c
Where d.dep=c.dep and pop_2010>50000;
```

Analyse du plan d'exécution:

```
-------------------------------------------------------------------------------
-- | Id  | Operation                 | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-- -------------------------------------------------------------------------------
-- |   0 | SELECT STATEMENT          |               | 35507 |   832K|   585   (1)| 00:00:08 |
-- |*  1 |  HASH JOIN                |               | 35507 |   832K|   585   (1)| 00:00:08 |
-- |   2 |   TABLE ACCESS FULL       | DEPARTEMENT   |   101 |  1313 |     3   (0)| 00:00:01 |
-- |*  3 |   TABLE ACCESS FULL       | COMMUNE       | 35507 |   381K|   582   (1)| 00:00:07 |
-- -------------------------------------------------------------------------------
--   1 - access("D"."DEP"="C"."DEP")
--   3 - filter("POP_2010">50000)
```

Recours à une table de hachage après un parcours séquentiel des tables Departement et Commune
Critère de sélection: pop_2010>=50000

Temps d'exécution estimé: 00:00:08

Requête avec l' utilisation forcée des boucles imbriquées

```
select /*+ USE_NL(commune c , departement d) */ d.dep, nom_dep
from departement d, commune c
where d.dep=c.dep and pop_2010>50000;
```

```
-------------------------------------------------------------------------------------------------
-- | Id  | Operation                      | Name              | Rows  | Bytes | Cost (%CPU)| Time    |
-- -------------------------------------------------------------------------------------------------
-- |   0 | SELECT STATEMENT               |                   | 35507 |   832K| 36098 (1)| 00:07:14 |
-- |   1 |  NESTED LOOPS                  |                   |       |       |          |          |
-- |   2 |   NESTED LOOPS                 |                   | 35507 |   832K| 36098 (1)| 00:07:14 |
-- |*  3 |    TABLE ACCESS FULL           | COMMUNE           | 35507 |   381K|   582 (1)| 00:00:07 |
-- |*  4 |    INDEX UNIQUE SCAN           | DEPARTEMENT_PK    |     1 |       |     0 (0)| 00:00:01 |
-- |   5 |   TABLE ACCESS BY INDEX ROWID  | DEPARTEMENT       |     1 |    13 |     1 (0)| 00:00:01 |
-- -------------------------------------------------------------------------------------------------
--    3 - filter("POP_2010">50000)
--    4 - access("D"."DEP"="C"."DEP")
```

Présence de boucles imbriquées après un parcours séquentiel de la table Commune pour trouver les tuples qui satisfont la condition de sélection ("POP_2010">50000) puis accès à la table Departement via l'index departement_pk

Temps d'éxecution estimé: 00:07:14

Q3. Donnez toutes les informations concernant les départements dont le numero de département est compris entre 25 et 45.

```
Select *
From Departement
Where dep>=25 and dep<=45;
```

Analyse du plan d'exécution:

```
-------------------------------------------------------------------------------------
-- | Id  | Operation             | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-- -------------------------------------------------------------------------------------
-- |   0 | SELECT STATEMENT      |                  |     1 |    20 |     3   (0)| 00:00:01 |
-- |*  1 |  TABLE ACCESS FULL    | DEPARTEMENT      |     1 |    20 |     3   (0)| 00:00:01 |
-- -------------------------------------------------------------------------------------
--    1 - filter(TO_NUMBER("DEP")>=25 AND TO_NUMBER("DEP")<=45)
```

Balayage séquentiel de la table Departement pour trouver les tuples qui satisfont les conditions de sélection (TO_NUMBER("DEP")>=25 AND TO_NUMBER("DEP")<=45) puis projection du résultat.

Temps d'exécution estimé : 00:00:01

Q4. Donnez pour les communes, leur nom, et le nom de leur département d'appartenance et de leur region d'appartenance

```
Select nom_com, nom_dep, nom_reg
From Commune c, Departement d, Region r
Where c.dep=d.dep and d.reg=r.reg;
```

Analyse du plan d'exécution:

```
-------------------------------------------------------------------------------------------
-- | Id  | Operation                          | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-- -------------------------------------------------------------------------------------------
-- |   0 | SELECT STATEMENT                   |                   | 36318 |  1525K|   587   (1)| 00:00:08 |
-- |*  1 |  HASH JOIN                         |                   | 36318 |  1525K|   587   (1)| 00:00:08 |
-- |   2 |   MERGE JOIN                       |                   |   101 |  2929 |     6  (17)| 00:00:01 |
-- |   3 |    TABLE ACCESS BY INDEX ROWID     | REGION            |    27 |   378 |     2   (0)| 00:00:01 |
-- |   4 |     INDEX FULL SCAN                | PRIMARY_KEY       |    27 |       |     1   (0)| 00:00:01 |
-- |*  5 |    SORT JOIN                       |                   |   101 |  1515 |     4  (25)| 00:00:01 |
-- |   6 |     TABLE ACCESS FULL              | DEPARTEMENT       |   101 |  1515 |     3   (0)| 00:00:01 |
-- |   7 |   TABLE ACCESS FULL                | COMMUNE           | 36318 |   496K|   581   (1)| 00:00:07 |
-- -------------------------------------------------------------------------------------------
--    1 - access("C"."DEP"="D"."DEP")
--    5 - access("D"."REG"="R"."REG")
```

Fusion des tables Departement et Region après un tri sur chaque table suivant "D"."REG"="R"."REG":

- accès à la table Region via l'index primary_key
- parcours séquentiel de la table Departement

Puis recours à une table de hachage entre la fusion précédente et la table Commune (parcours séquentiel)

Temps d'éxecution estimé: 00:00:08

Q5. Donnez le nombre de communes par numero et nom de département

```
Select nom_dep, d.dep, count(nom_com) as nb
From Commune c, Departement d
Where c.dep=d.dep
Group by nom_dep, d.dep;
```

Analyse du plan d'exécution:

```
-----------------------------------------------------------------------------------
-- | Id  | Operation            | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-- -----------------------------------------------------------------------------------
-- |   0 | SELECT STATEMENT     |               |  6857 |   180K|   586   (1)| 00:00:08 |
-- |   1 |  HASH GROUP BY       |               |  6857 |   180K|   586   (1)| 00:00:08 |
-- |*  2 |   HASH JOIN          |               | 36318 |   957K|   584   (1)| 00:00:08 |
-- |   3 |    TABLE ACCESS FULL | DEPARTEMENT   |   101 |  1313 |     3   (0)| 00:00:01 |
-- |   4 |    TABLE ACCESS FULL | COMMUNE       | 36318 |   496K|   581   (1)| 00:00:07 |
-- -----------------------------------------------------------------------------------
-- 2 - access("C"."DEP"="D"."DEP")
```

Regroupement des résultats obtenus par nom et numéro de département après avoir recours à une table de hachage après un parcours séquentiel des tables Departement et Commune.

Temps d'éxecution estimé: 00:00:08

Partie 2: Création d'index secondaires sur les attributs impliques dans les jointures et sur nom_com

Q1: Donnez toutes les informations concernant les communes dont le nom commence par 'MO'

- Creation d'un index sur nom_com:

```
create index idx_nom_com on commune (nom_com);
```

- Analyse du plan d'exécution:

```
-- ------------------------------------------------------------------------------------------
-- | Id  | Operation                     | Name            | Rows  | Bytes | Cost (%CPU)  | Time     |
-- ------------------------------------------------------------------------------------------
-- |   0 | SELECT STATEMENT              |                 |   7   |  2513 |       9   (0)| 00:00:01 |
-- |   1 |  TABLE ACCESS BY INDEX ROWID  | COMMUNE         |   7   |  2513 |       9   (0)| 00:00:01 |
-- |*  2 |   INDEX RANGE SCAN            | IDX_NOM_COM     |   7   |       |       2   (0)| 00:00:01 |
-- ------------------------------------------------------------------------------------------
--    2 - access("NOM_COM" LIKE 'MO%')
--        filter("NOM_COM" LIKE 'MO%')
```

Accès à la table Commune via l'index idx_nom_com) avec accès direct aux identifiants des tuples (rowid) qui satisfont la condition de sélection (nom_com commençant par MO)

Temps d'execution estimé: 00:00:01

Q2: Donnez les noms et les numéros de département des communes dont la population de 2010 est supérieure à 50 000 habitants

- Creation index sur dep dans la table Commune:

```
create index idx_dep on commune (dep);
```

-Requête:

```
Select  /*+ INDEX(idx_dep) */ d.dep, nom_dep
From departement d, commune c
Where d.dep=c.dep and pop_2010>50000;
```

--> Pas de modifications sur le plan d'exécution

Utilisation de boucles imbriquees :

```
select /*+ USE_NL(commune c , departement d) */ d.dep, nom_dep
from departement d, commune c
where d.dep=c.dep and pop_2010>50000;
```

-Analyse du plan d'Exécution:

```
-------------------------------------------------------------------------------------------
-- | Id  | Operation                     | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-- -------------------------------------------------------------------------------------------
-- |   0 | SELECT STATEMENT              |               | 35507 |   832K|  2314   (1)| 00:00:28 |
-- |   1 |  NESTED LOOPS                 |               |       |       |            |          |
-- |   2 |   NESTED LOOPS                |               | 35507 |   832K|  2314   (1)| 00:00:28 |
-- |   3 |    TABLE ACCESS FULL          | DEPARTEMENT   |   101 |  1313 |  3   (0)   | 00:00:01 |
-- |*  4 |    INDEX RANGE SCAN           | IDX_DEP       |   378 |       |     1   (0)| 00:00:01 |
-- |*  5 |   TABLE ACCESS BY INDEX ROWID| COMMUNE        |   352 |  3872 |24   (0)    | 00:00:01 |
-- -------------------------------------------------------------------------------------------
-- 4 - access("D"."DEP"="C"."DEP")
-- 5 - filter("POP_2010">50000)
```

Presence de boucles imbriquées après un parcours séquentiel de Departement et accès à la table Commune via l'index idx_dep pour trouver les tuples qui satisfont la condition de sélection

Temps d'exécution estimé: 00:00:28

Q3: Donnez toutes les informations concernant les départements dont le numero de d´epartement est compris entre 25 et 45.

```
Select *
From Departement
Where dep>=25 and dep<=45;
```

-- > Pas de changement dans le plan d'exécution

Q4: Donnez pour les communes, leur nom, et le nom de leur departement d'appartenance et de leur region d'appartenance

- Creation index sur reg dans la table Departement:

```
create index idx_reg on departement(reg);
```

- Requête:

```
Select  /*+ INDEX(idx_dep), INDEX(idx_reg) */ nom_com, nom_dep, nom_reg
 From Commune c, Departement d, Region r
  Where c.dep=d.dep and d.reg=r.reg;
```

-- Analyse du plan d'Execution:

```
-------------------------------------------------------------------------------------------
-- | Id  | Operation                          | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-- -------------------------------------------------------------------------------------------
-- |   0 | SELECT STATEMENT                   |                     | 36318 |  1525K|   587   (1)| 00:00:08 |
-- |*  1 |  HASH JOIN                         |                     | 36318 |  1525K|   587   (1)| 00:00:08 |
-- |   2 |   MERGE JOIN                       |                     |   101 |  2929 |     6  (17)| 00:00:01 |
-- |   3 |    TABLE ACCESS BY INDEX ROWID     | DEPARTEMENT         |   101 |  1515 |     2   (0)| 00:00:01 |
-- |   4 |     INDEX FULL SCAN                | IDX_REG             |   101 |       |     1   (0)| 00:00:01 |
-- |*  5 |    SORT JOIN                       |                     |    27 |   378 |     4  (25)| 00:00:01 |
-- |   6 |     TABLE ACCESS FULL              | REGION              |    27 |   378 |     3   (0)| 00:00:01 |
-- |   7 |   TABLE ACCESS FULL                | COMMUNE             | 36318 |   496K|   581   (1)| 00:00:07 |
-- -------------------------------------------------------------------------------------------
--    1 - access("C"."DEP"="D"."DEP")
--    5 - access("D"."REG"="R"."REG")
--        filter("D"."REG"="R"."REG")
```

Fusion des tables Departement et Region après un tri sur chaque table suivant "D"."REG"="R"."REG":

-- accès à la table Departement via l'index idx_reg

-- parcours séquentiel de la table Region

Puis recours à une table de hachage entre la fusion précédente et la table Commune (parcours séquentiel)

Temps d'exécution estimé: 00:00:08

Utilisation de boucles imbriquées

```
select /*+ USE_NL(commune c , departement d, region r) */  nom_com, nom_dep, nom_reg
From Commune c, Departement d, Region r
Where c.dep=d.dep and d.reg=r.reg;
```

-- Plan d'Execution

```
-------------------------------------------------------------------------------------------
-- | Id  | Operation                          | Name           | Rows  | Bytes     | Cost (%CPU)| Time     |
-- -------------------------------------------------------------------------------------------
-- |   0 | SELECT STATEMENT                   |                | 36318 |      1525K|  2340   (1)| 00:00:29 |
-- |   1 |  NESTED LOOPS                      |                |       |           |            |          |
-- |   2 |   NESTED LOOPS                     |                | 36318 |      1525K|  2340   (1)| 00:00:29 |
-- |   3 |    NESTED LOOPS                    |                |   101 |      2929 |    30   (0)| 00:00:01 |
-- |   4 |     TABLE ACCESS FULL              | REGION         |    27 |       378 |     3   (0)| 00:00:01 |
-- |   5 |     TABLE ACCESS BY INDEX ROWID    | DEPARTEMENT    |     4 |        60 |     1   (0)| 00:00:01 |
-- |*  6 |      INDEX RANGE SCAN              | IDX_REG        |     4 |           |     0   (0)| 00:00:01 |
-- |*  7 |    INDEX RANGE SCAN                | IDX_DEP        |   378 |           |     1   (0)| 00:00:01 |
-- |   8 |   TABLE ACCESS BY INDEX ROWID      | COMMUNE        |   360 |      5040 |    24   (0)| 00:00:01 |
-- -------------------------------------------------------------------------------------------
--     6 - access("D"."REG"="R"."REG")
--     7 - access("C"."DEP"="D"."DEP")
```

Presence de boucles imbriquées après un parcours séquentiel de Région et accès à la table Département via l'index idx_reg avec un accès aux identifiants des tuples pour trouver ceux correspondants (« D"."REG"="R"."REG") puis un accès à la table Commune via l'index idx_dep avec un accès aux identifiants des tuples pour trouver ceux correspondants ("C"."DEP"="D"."DEP")

Temps d'exécution estimé: 00:00:29

Q5: Donnez le nombre de communes par numero et nom de département

```
Select /*+ INDEX(idx_dep) */ nom_dep, d.dep, count(nom_com) as nb
From Commune c, Departement d
Where c.dep=d.dep
Group by nom_dep, d.dep;
```

-- Plan d'exécution

```
-------------------------------------------------------------------------------------
-- | Id  | Operation            | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-- -------------------------------------------------------------------------------------
-- |   0 | SELECT STATEMENT     |            |  6857 |   180K|   586   (1)| 00:00:08 |
-- |   1 |  HASH GROUP BY       |            |  6857 |   180K|   586   (1)| 00:00:08 |
-- |*  2 |   HASH JOIN          |            | 36318 |   957K|   584   (1)| 00:00:08 |
-- |   3 |    TABLE ACCESS FULL | DEPARTEMENT|  101  |  1313 |   3   (0)  | 00:00:01 |
-- |   4 |    TABLE ACCESS FULL | COMMUNE    |36318  |   496K|   581   (1)| 00:00:07 |
-- -------------------------------------------------------------------------------------
-- 2 - access("C"."DEP"="D"."DEP")
```

Pas de changement

Temps d'exécution estimé: 00:00:29

Suppression des index

```
drop index idx_dep;
drop index idx_reg;
drop index idx_nom_com;
```

Exercice 3:

Q1: Donnez le nom, la latitude et la longitude des communes qui se situent dans les departements de l'Herault et du Gard sous differentes formes

set timing on;

- **Cas 1: sous la forme d'une jointure**

```
Select nom_com, latitude, longitude
From Commune c, Departement d
Where c.dep=d.dep and (nom_dep='HERAULT' or nom_dep='GARD');
```

—> temps d'exécution: 00:00:00.07

```
-------------------------------------------------------------------------------
| Id  | Operation           | Name       |  Rows  | Bytes | Cost (%CPU)      | Time    |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |            |    757 | 28766 |   584   (1)      | 00:00:08 |
|*  1 |  HASH JOIN          |            |    757 | 28766 |   584   (1)      | 00:00:08 |
|*  2 |   TABLE ACCESS FULL| DEPARTEMENT|      2 |    26 |     3   (0)      | 00:00:01 |
|   3 |   TABLE ACCESS FULL| COMMUNE    | 36318 |   886K|   581   (1)      | 00:00:07 |
-------------------------------------------------------------------------------
    1 - access("C"."DEP"="D"."DEP")
    2 - filter("NOM_DEP"='GARD' OR "NOM_DEP"='HERAULT')
```

- **Cas 2: sous la forme de requêtes imbriquées => test de vacuité**

```
Select nom_com, latitude, longitude
From Commune c
Where exists (select nom_dep from Departement d where c.dep=d.dep and
(nom_dep='HERAULT' or nom_dep='GARD'));
```

--> temps d'exécution: 00:00:00.05

```
-------------------------------------------------------------------------------
| Id  | Operation              | Name              | Rows  | Bytes | Cost (%CPU)   | Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |                   |   757 | 28766 |   584   (1)   | 00:00:08 |
|*  1 |  HASH JOIN RIGHT SEMI  |                   |   757 | 28766 |   584   (1)   | 00:00:08 |
|*  2 |   TABLE ACCESS FULL    | DEPARTEMENT       |     2 |    26 |     3   (0)   | 00:00:01 |
|   3 |   TABLE ACCESS FULL    | COMMUNE           | 36318 |   886K|   581   (1)   | 00:00:07 |
*/
```

- **Cas 3: sous la forme de requêtes imbriquées => test d'appartenance**

```
Select nom_com, latitude, longitude
From Commune c
Where c.dep in (select dep from Departement where nom_dep='HERAULT' or
nom_dep='GARD');
```

--> temps d'éxécution: 00:00:00.05

```
-------------------------------------------------------------------------------
| Id  | Operation                | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                   |   757 | 28766 |   584   (1)| 00:00:08 |
|*  1 |  HASH JOIN               |                   |   757 | 28766 |   584   (1)| 00:00:08 |
|*  2 |   TABLE ACCESS FULL      |  DEPARTEMENT      |     2 |    26 |     3   (0)| 00:00:01 |
|   3 |   TABLE ACCESS FULL      | COMMUNE           | 36318 |   886K|   581   (1)| 00:00:07 |
-------------------------------------------------------------------------------
```

Quelle est l'écriture qui vous semble la moins couteuse (utilisez set timing et explain) ?
Celle sous forme de requête imbriqués

Quels sont les opérateurs physiques exploites respectivement pour exprimer la jointure?

Les opérateurs physiques exploités pour exprimer la jointure sont:

- **nested loop join** : la table à droite de la jointure est confrontée à chaque tuple de la table à gauche.
- **hash join** : la table à droite de la jointure fait l'objet d'une table de hachage en mémoire vive. La table à gauche est alors examinée et la jointure s'effectue alors via la table de hachage.
- **merge sort join** : chaque table est triée sur les attributs impliqués dans la condition de jointure. Une fois le tri réalisé, l'opération de jointure par fusion peut alors être effectuée.

Exploiter les directives (hint) pour forcer le choix d'un opérateur (par exemple use nl):

```
Select nom_com, latitude, longitude
From Commune c, Departement d
Where c.dep=d.dep and (nom_dep='HERAULT' or nom_dep='GARD');
```

—> Sans directive l'optimiseur choisi HASHJOIN comme opérateur

- **Utilisation de nested loop join:**

```
Select  /*+ USE_NL(commune c , departement d */  nom_com, latitude, longitude
From Commune c, Departement d
Where c.dep=d.dep and (nom_dep='HERAULT' or nom_dep='GARD');
```

```
-------------------------------------------------------------------------------
-- | Id  | Operation            | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-- -------------------------------------------------------------------------------
-- |   0 | SELECT STATEMENT     |                   |   757 | 28766 |  1163   (1)| 00:00:14 |
-- |   1 |  NESTED LOOPS        |                   |   757 | 28766 |  1163   (1)| 00:00:14 |
-- |*  2 |   TABLE ACCESS FULL  | DEPARTEMENT       |     2 |    26 |     3   (0)| 00:00:01 |
-- |*  3 |   TABLE ACCESS FULL  | COMMUNE           |   378 |  9450 |   580   (1)| 00:00:07 |
-- -------------------------------------------------------------------------------
--    2 - filter("NOM_DEP"='GARD' OR "NOM_DEP"='HERAULT')
--    3 - filter("C"."DEP"="D"."DEP")
```

- **Utilisation de merge join:**

```
Select  /*+ USE_MERGE(commune c , departement d */  nom_com, latitude, longitude
From Commune c, Departement d
Where c.dep=d.dep and (nom_dep='HERAULT' or nom_dep='GARD');
```

```
-------------------------------------------------------------------------------------------------
-- | Id  | Operation                          | Name                | Rows  | Bytes |TempSpc| Cost (%CPU)  | Time    |
-- -------------------------------------------------------------------------------------------------
-- |   0 | SELECT STATEMENT                   |                     |   757 | 28766 |       |   848     (1)| 00:00:11 |
-- |   1 |  MERGE JOIN                        |                     |   757 | 28766 |       |   848     (1)| 00:00:11 |
-- |*  2 |   TABLE ACCESS BY INDEX ROWID      | DEPARTEMENT         |     2 |    26 |       |     2     (0)| 00:00:01 |
-- |   3 |    INDEX FULL SCAN                 | DEPARTEMENT_PK      |   101 |       |       |     1     (0)| 00:00:01 |
-- |*  4 |   SORT JOIN                        |                     | 36318 |   886K|  2872K|   846     (1)| 00:00:11 |
-- |   5 |    TABLE ACCESS FULL               | COMMUNE             | 36318 |   886K|       |   581     (1)| 00:00:07 |
-- -------------------------------------------------------------------------------------------------
--    2 - filter("NOM_DEP"='GARD' OR "NOM_DEP"='HERAULT')
--    4 - access("C"."DEP"="D"."DEP")
--        filter("C"."DEP"="D"."DEP")
```

-- Construisez sur papier un arbre algébrique puis les plans physiques correspondant a chaque plan d'execution choisi

Q2: Donnez le nom et la population 2010 des communes qui ont plus d'habitants que le nombre moyen d'habitants par commune.

```
Select nom_com, pop_2010
From Commune
Where pop_2010 > (Select avg(pop_2010) from Commune);
```

-- Exploitez le plan d'execution pour cette requête.

```
-------------------------------------------------------------------------------
-- | Id  | Operation           | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-- -------------------------------------------------------------------------------
-- |   0 | SELECT STATEMENT    |          |  1816 | 38136 |  1163   (1)| 00:00:14 |
-- |*  1 |  TABLE ACCESS FULL  | COMMUNE  |  1816 | 38136 |   582   (1)| 00:00:07 |
-- |   2 |   SORT AGGREGATE    |          |     1 |     9 |            |          |
-- |   3 |    TABLE ACCESS FULL| COMMUNE  | 36318 |   319K|   582   (1)| 00:00:07 |
-- -------------------------------------------------------------------------------
--    1 - filter("POP_2010"> (SELECT AVG("POP_2010") FROM "COMMUNE"
--        "COMMUNE"))
```

Parcours séquentiel de la table Commune pour trouver les tuples satisfaisant la condition (avoir plus d'habitants en2010 que la moyenne), après un premier parcours séquentiel de la table Commune pour calculer la moyenne du nombre d'habitants en 2010

-- Quelles sont les operations exploitées par l'optimiseur ?

-- Table Access Full (2 fois)
-- Sort Aggregate
-- Select Statement

Q3: Donnez le nom des communes, le nom de leur département et de leur région respectifs lorsque ces communes sont situées dans les régions Midi- Pyrénées, Languedoc-Roussillon et Provence-Alpes-Côte d'Azur.

- **R1: Jointures Naturelles**

```
Select nom_com, nom_dep, nom_reg
From commune c, departement d, region r
Where c.dep=d.dep and d.reg=r.reg and nom_reg in ('MIDI-PYRENEES','LANGUEDOC-
ROUSSILLON','PROVENCE-ALPES-COTE D%AZUR');
```

-- Plan d'Execution

```
-------------------------------------------------------------------------------------------
-- | Id  | Operation                             | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-- -------------------------------------------------------------------------------------------
-- |   0 | SELECT STATEMENT                      |                     |  4035 |   169K|   587   (1)| 00:00:08 |
-- |*  1 |  HASH JOIN                            |                     |  4035 |   169K|   587   (1)| 00:00:08 |
-- |   2 |   MERGE JOIN                          |                     |    11 |   319 |     6  (17)| 00:00:01 |
-- |*  3 |    TABLE ACCESS BY INDEX ROWID        | REGION              |     3 |    42 |     2   (0)| 00:00:01 |
-- |   4 |     INDEX FULL SCAN                   | PRIMARY_KEY         |    27 |       |     1   (0)| 00:00:01 |
-- |*  5 |    SORT JOIN                          |                     |   101 |  1515 |     4  (25)| 00:00:01 |
-- |   6 |     TABLE ACCESS FULL                 | DEPARTEMENT         |   101 |  1515 |     3   (0)| 00:00:01 |
-- |   7 |   TABLE ACCESS FULL                   | COMMUNE             | 36318 |   496K|   581   (1)| 00:00:07 |
-- -------------------------------------------------------------------------------------------
--    1 - access("C"."DEP"="D"."DEP")
--    3 - filter("NOM_REG"='LANGUEDOC-ROUSSILLON' OR "NOM_REG"='MIDI-PYRENEES' OR
--        "NOM_REG"='PROVENCE-ALPES-COTE D%AZUR')
--    5 - access("D"."REG"="R"."REG")
--        filter("D"."REG"="R"."REG")
```

Fusion des tables Departement et Region après un tri sur chaque table suivant "D"."REG"="R"."REG":

-- accès à la table Region via l'index primary_key avec un accès aux identifiants des tuples correspondants ("NOM_REG"='LANGUEDOC-ROUSSILLON' OR "NOM_REG"='MIDI-PYRENEES' OR "NOM_REG"='PROVENCE-ALPES-COTE D%AZUR')

-- parcours séquentiel de la table Departement

Puis recours à une table de hachage entre la fusion précédente et la table Commune (parcours séquentiel)

Temps d'exécution: 00:00:08

- **R2: Semi Jointure => test de vacuité**

```
Select nom_com
From Commune c
Where exists (select nom_dep from Departement d where c.dep=d.dep and
exists (Select nom_reg from Region r where d.reg=r.reg and nom_reg in ('MIDI-
PYRENEES','LANGUEDOC-ROUSSILLON','PROVENCE-ALPES-COTE D%AZUR')
));
```

-- Plan d'Execution

```
-------------------------------------------------------------------------------------------
-- | Id  | Operation                             | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-- -------------------------------------------------------------------------------------------
-- |   0 | SELECT STATEMENT                      |                     |  4161 | 74898 |   587     (1)| 00:00:08 |
-- |*  1 |  HASH JOIN RIGHT SEMI                 |                     |  4161 | 74898 |   587     (1)| 00:00:08 |
-- |   2 |   VIEW                                | VW_SQ_1             |    11 |    44 |       6  (17)| 00:00:01 |
-- |   3 |    MERGE JOIN                         |                     |    11 |   209 |       6  (17)| 00:00:01 |
-- |   4 |     SORT UNIQUE                       |                     |     3 |    42 |     2     (0)| 00:00:01 |
-- |*  5 |      TABLE ACCESS BY INDEX ROWID      | REGION              |     3 |    42 |     2     (0)| 00:00:01 |
-- |   6 |       INDEX FULL SCAN                 | PRIMARY_KEY         |    27 |       |     1     (0)| 00:00:01 |
-- |*  7 |     SORT JOIN                         |                     |   101 |   505 |       4  (25)| 00:00:01 |
-- |   8 |      TABLE ACCESS FULL                | DEPARTEMENT         |   101 |   505 |     3     (0)| 00:00:01 |
-- |   9 |   TABLE ACCESS FULL                   | COMMUNE             | 36318 |   496K|   581     (1)| 00:00:01 |
-- -------------------------------------------------------------------------------------------
--  1 - access("C"."DEP"="ITEM_1")
--    5 - filter("NOM_REG"='LANGUEDOC-ROUSSILLON' OR "NOM_REG"='MIDI-PYRENEES' OR
--        "NOM_REG"='PROVENCE-ALPES-COTE D%AZUR')
--    7 - access("D"."REG"="R"."REG")
--        filter("D"."REG"="R"."REG")
```

- **R3: Semi Jointure => test d'appartenance**

```
Select nom_com
From Commune c
Where c.dep in (select dep from Departement where reg in
(select reg from Region where nom_reg in ('MIDI-PYRENEES','LANGUEDOC-
ROUSSILLON','PROVENCE-ALPES-COTE D%AZUR') ) );
```

-- Plan d'Execution

```
-- | Id  | Operation                           | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-- ---------------------------------------------------------------------------------------------------
-- |   0 | SELECT STATEMENT                    |                |  4161 | 74898 |   587   (1)| 00:00:08 |
-- |*  1 |  HASH JOIN RIGHT SEMI               |                |  4161 | 74898 |   587   (1)| 00:00:08 |
-- |   2 |   VIEW                              | VW_NSO_1       |    11 |   44  |     6  (17)| 00:00:01 |
-- |   3 |    MERGE JOIN                       |                |    11 |  209  |     6  (17)| 00:00:01 |
-- |*  4 |     TABLE ACCESS BY INDEX ROWID     | REGION         |     3 |   42  |     2   (0)| 00:00:01 |
-- |   5 |      INDEX FULL SCAN                | PRIMARY_KEY    |    27 |       |     1   (0)| 00:00:01 |
-- |*  6 |     SORT JOIN                       |                |   101 |  505  |     4  (25)| 00:00:01 |
-- |   7 |      TABLE ACCESS FULL              | DEPARTEMENT    |   101 |  505  |     3   (0)| 00:00:01 |
-- |   8 |   TABLE ACCESS FULL                 | COMMUNE        | 36318 |  496K |   581   (1)| 00:00:07 |
-- ---------------------------------------------------------------------------------------------------
--  1 - access("C"."DEP"="DEP")
--    4 - filter("NOM_REG"='LANGUEDOC-ROUSSILLON' OR "NOM_REG"='MIDI-PYRENEES' OR
--        "NOM_REG"='PROVENCE-ALPES-COTE D%AZUR')
--    6 - access("REG"="REG")
--        filter("REG"="REG")
```

- **R4: Double Jointure**

```
Select nom_com, nom_dep, nom_reg
From Commune c inner join (Departement d inner join Region r on d.reg=r.reg) on
c.dep=d.dep
Where nom_reg in ('MIDI-PYRENEES','LANGUEDOC-ROUSSILLON','PROVENCE-ALPES-COTE
D%AZUR');
```

-- Plan Execution

```
-----------------------------------------------------------------------------------------------
-- | Id  | Operation                           | Name           | Rows      | Bytes | Cost (%CPU)| Time     |
-- ---------------------------------------------------------------------------------------------
-- |   0 | SELECT STATEMENT                    |                |   4035    |   169K|   587   (1)| 00:00:08 |
-- |*  1 |  HASH JOIN                          |                |  4035     |   169K|   587   (1)| 00:00:08 |
-- |   2 |   MERGE JOIN                        |                |       11  |   319 |     6  (17)| 00:00:01 |
-- |*  3 |    TABLE ACCESS BY INDEX ROWID      | REGION         |        3  |    42 |     2   (0)| 00:00:01 |
-- |   4 |     INDEX FULL SCAN                 | PRIMARY_KEY    |       27  |       |     1   (0)| 00:00:01 |
-- |*  5 |    SORT JOIN                        |                |      101  |  1515 |     4  (25)| 00:00:01 |
-- |   6 |     TABLE ACCESS FULL               | DEPARTEMENT    |      101  |  1515 |     3   (0)| 00:00:01 |
-- |   7 |   TABLE ACCESS FULL                 | COMMUNE        | 36318     |   496K|   581   (1)| 00:00:07 |
-- ---------------------------------------------------------------------------------------------
-- 1 - access("C"."DEP"="D"."DEP")
--    3 - filter("R"."NOM_REG"='LANGUEDOC-ROUSSILLON' OR "R"."NOM_REG"='MIDI-PYRENEES'
--        OR "R"."NOM_REG"='PROVENCE-ALPES-COTE D%AZUR')
--    5 - access("D"."REG"="R"."REG")
--        filter("D"."REG"="R"."REG")
```

- **R5: Avec une Vue**

```
Create or Replace view V_Commune
as
select nom_com, nom_dep, nom_reg
From Commune c inner join (Departement d inner join Region r on d.reg=r.reg) on
c.dep=d.dep
Where nom_reg in ('MIDI-PYRENEES','LANGUEDOC-ROUSSILLON','PROVENCE-ALPES-COTE
D%AZUR');

Select nom_com, nom_dep, nom_reg from V_Commune;
```

-- Plan Execution

```
--| Id      | Operation                          | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-- ---------------------------------------------------------------------------------------------------
-- |   0     | SELECT STATEMENT                   |               |  4035 |   169K|   587   (1)| 00:00:08 |
-- |*  1     |  HASH JOIN                         |               |  4035 |   169K|   587   (1)| 00:00:08 |
-- |   2     |   MERGE JOIN                       |               |    11 |   319 |     6  (17)| 00:00:01 |
-- |*  3     |    TABLE ACCESS BY INDEX ROWID     | REGION        |     3 |    42 |     2   (0)| 00:00:01 |
-- |   4     |     INDEX FULL SCAN                | PRIMARY_KEY   |    27 |       |     1   (0)| 00:00:01 |
-- |*  5     |    SORT JOIN                       |               |   101 |  1515 |     4  (25)| 00:00:01 |
-- |   6     |     TABLE ACCESS FULL              | DEPARTEMENT   |   101 |  1515 |     3   (0)| 00:00:01 |
-- |   7     |   TABLE ACCESS FULL                | COMMUNE       | 36318 |   496K|   581   (1)| 00:00:07 |
-- ---------------------------------------------------------------------------------------------------
--    1 - access("C"."DEP"="D"."DEP")
--    3 - filter("R"."NOM_REG"='LANGUEDOC-ROUSSILLON' OR "R"."NOM_REG"='MIDI-PYRENEES'
--        OR "R"."NOM_REG"='PROVENCE-ALPES-COTE D%AZUR')
--    5 - access("D"."REG"="R"."REG")
--        filter("D"."REG"="R"."REG")
```

Drop View V_Commune;

Remarque :

-- Les plans d'exécution des différentes requêtes sont similaires (Merge Join entre Region et Departement puis HASH Join avec Commune)

Exercice 4: