Nom: Akrouchi Rihab
Matricule: 202031050231
groupe: Master 2 IL groupe 2

TP SDED 3

TP 4

Introduction :

Ce rapport présente les étapes suivies pour analyser et améliorer les performances des requêtes SQL dans une base de données. Nous avons commencé par la création et l'initialisation des tables nécessaires, puis l'élargissement de ces tables avec des données fictives générées aléatoirement. Ensuite, nous avons élaboré et exécuté des requêtes complexes avec pl/sql, en mesurant les temps d'exécution. Enfin, nous avons testé l'impact de l'utilisation d'une vue matérialisée sur la performance des requêtes en rafraîchissant cette vue après l'augmentation du volume des données. L'objectif de cette étude est de comparer les temps d'exécution des requêtes avec et sans la vue matérialisée, afin d'évaluer son efficacité dans l'optimisation des performances.

1.ACTIVATION DES OPTIONS :

```
SQL> SET TIMING ON ;
SQL> SET AUTOTRACE  ON EXPLAIN ;
```

2.LA REQUETE 1 :

```
SQL> SELECT t.CodeTrajet,t.DateTrajet
  2  FROM Trajet t ,Panne p , TypePanne tp
  3  where
  4  Typepanne='Moteur'
  5  AND tp.CodeTypePanne=p.CodeTypePanne
  6  AND p.CodeTrajet=t.CodeTrajet;

CODETRAJET DATETRAJE
---------- ---------
      1001 01-APR-23
```

3. EXAMINATION DU TEMPS ET DU PLAN :

```
SQL> SELECT t.CodeTrajet,t.DateTrajet
  2  FROM Trajet t ,Panne p , TypePanne tp
  3  where
  4  Typepanne='Moteur'
  5  AND tp.CodeTypePanne=p.CodeTypePanne
  6  AND p.CodeTrajet=t.CodeTrajet;

CODETRAJET DATETRAJE
---------- ---------
      1001 01-APR-23

Elapsed: 00:00:00.08

Execution Plan
----------------------------------------------------------
Plan hash value: 1844461349

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

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

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

|   0 | SELECT STATEMENT    |           |  8422 |   370K|  9197   (1)| 00:00:01
|

|*  1 |  HASH JOIN          |           |  8422 |   370K|  9197   (1)| 00:00:01
|

|*  2 |   HASH JOIN         |           |  8422 |   254K|   349   (2)| 00:00:01
|

|*  3 |    TABLE ACCESS FULL| TYPEPANNE |     1 |    22 |     3   (0)| 00:00:01
|

|   4 |    TABLE ACCESS FULL| PANNE     |   160K|  1406K|   344   (1)| 00:00:01
|

|   5 |   TABLE ACCESS FULL | TRAJET    |  2502K|    33M|  8829   (1)| 00:00:01
|
```

4. CRÉATION DE LA VUE VMTP

```
SQL> CREATE MATERIALIZED VIEW VMTP
  2  BUILD IMMEDIATE
  3  REFRESH COMPLETE ON DEMAND
  4  ENABLE QUERY REWRITE
  5  AS
  6  SELECT t.CodeTrajet,
  7         t.DateTrajet,
  8         p.CodePanne,
  9         p.CodeTypePanne
 10  FROM Trajet t
 11  JOIN Panne p ON t.CodeTrajet = p.CodeTrajet
 12  JOIN TypePanne tp ON p.CodeTypePanne = tp.CodeTypePanne;

Materialized view created.

Elapsed: 00:00:01.24
```

5.LA REEXECUTION DE LA REQUETE R1 :

```
  4  Typepanne='Moteur'
  5  AND tp.CodeTypePanne=p.CodeTypePanne
  6  AND p.CodeTrajet=t.CodeTrajet;

CODETRAJET DATETRAJE
---------- ---------
      1001 01-APR-23

Elapsed: 00:00:00.01

Execution Plan
----------------------------------------------------------
Plan hash value: 1844461349

-------------------------------------------------------------------------------
-
| Id  | Operation           | Name      | Rows  | Bytes | Cost (%CPU)| Time
|
-------------------------------------------------------------------------------
-
|   0 | SELECT STATEMENT    |           |  8422 |   370K|  9197   (1)| 00:00:01
|
|*  1 |  HASH JOIN          |           |  8422 |   370K|  9197   (1)| 00:00:01
|
|*  2 |   HASH JOIN         |           |  8422 |   254K|   349   (2)| 00:00:01
|
|*  3 |    TABLE ACCESS FULL| TYPEPANNE |     1 |    22 |     3   (0)| 00:00:01
|
|   4 |    TABLE ACCESS FULL| PANNE     |   160K|  1406K|   344   (1)| 00:00:01
|
|   5 |   TABLE ACCESS FULL | TRAJET    |  2502K|    33M|  8829   (1)| 00:00:01
|
-------------------------------------------------------------------------------
-


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

   1 - access("P"."CODETRAJET"="T"."CODETRAJET")
   2 - access("TP"."CODETYPEPANNE"="P"."CODETYPEPANNE")
   3 - filter("TYPEPANNE"='Moteur')

Note
-----
   - this is an adaptive plan
```

REMARQUE : Le temps a diminué avec la vue matérialisée en comparant avec la première exécution de la requête sans la vue VMTP

AVANT VMTP	APRÈS VMTP
`Elapsed: 00:00:00.08` **La requête R1 nécessitait environ 0.08s**	`Elapsed: 00:00:00.01` **Le temps d'exécution a été réduit à environ 0.01 s, ce qui représente une réduction significative du temps d'exécution.**

6. La requete R2 :

```
SELECT EXTRACT(MONTH FROM DateTrajet) AS Mois,
EXTRACT(YEAR FROM DateTrajet) AS Annee,
 COUNT(*) AS NBT
FROM Trajet
GROUP BY EXTRACT(MONTH FROM DateTrajet), EXTRACT(YEAR FROM DateTrajet);
```

7. Le temps et le plan d'exécution :

```
SQL> SELECT EXTRACT(MONTH FROM DateTrajet) AS Mois, EXTRACT(YEAR FROM DateTrajet) AS Annee, COUNT(*) AS NBT
  2  FROM Trajet
  3  GROUP BY EXTRACT(MONTH FROM DateTrajet), EXTRACT(YEAR FROM DateTrajet);

      MOIS      ANNEE        NBT
---------- ---------- ----------
        12       2023     213496
         1       2023     212588
        10       2023     211910
         5       2023     212788
         3       2023     212279
         7       2023     212126
        11       2023     205813
         6       2024          7
         2       2023     191270
         6       2023     205843
         4       2023     206019

      MOIS      ANNEE        NBT
---------- ---------- ----------
         8       2023     212517
         9       2023     205896

13 rows selected.

Elapsed: 00:00:00.74

Execution Plan
----------------------------------------------------------
Plan hash value: 604744428

--------------------------------------------------------------------------------
| Id  | Operation          | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |        |   369 |  2952 |  9012   (3)| 00:00:01 |
|   1 |  HASH GROUP BY     |        |   369 |  2952 |  9012   (3)| 00:00:01 |
|   2 |   TABLE ACCESS FULL| TRAJET |  2502K|    19M|  8829   (1)| 00:00:01 |
--------------------------------------------------------------------------------

SQL>
```

8. CRÉATION DE LA VUE VMNBT

```
SQL> CREATE MATERIALIZED VIEW VMNBT
  2  BUILD IMMEDIATE
  3  REFRESH COMPLETE ON DEMAND
  4  ENABLE QUERY REWRITE
  5  AS
  6  SELECT EXTRACT(MONTH FROM DateTrajet) AS Mois, EXTRACT(YEAR FROM DateTrajet) AS Annee, COUNT(*) AS NBT
  7  FROM Trajet
  8  GROUP BY EXTRACT(MONTH FROM DateTrajet), EXTRACT(YEAR FROM DateTrajet);

Materialized view created.

Elapsed: 00:00:00.92
```

9.La reexecution de la requête R2 et l’examination du temps :

```
      MOIS      ANNEE        NBT
---------- ---------- ----------
         8       2023     212517
         9       2023     205896

13 rows selected.

Elapsed: 00:00:00.01

Execution Plan
----------------------------------------------------------
Plan hash value: 1052901248

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

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

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

|   0 | SELECT STATEMENT             |       |    13 |   156 |     3   (0)| 00:0
0:01 |

|   1 |  MAT_VIEW REWRITE ACCESS FULL| VMNBT |    13 |   156 |     3   (0)| 00:0
0:01 |

--------------------------------------------------------------------------------
------
```

10.a/ Augmentation de nombre d'instances de transactions à 50%

```
SQL> DECLARE
  2      CodeTrajet NUMBER;
  3      DateTrajet DATE;
  4      DureeTrajet INTERVAL DAY TO SECOND;
  5      KilometrageParcouru NUMBER;
  6      CarburantConsomme NUMBER;
  7      CodeVehicule NUMBER;
  8      CodeDepart NUMBER;
  9      CodeArrivee NUMBER;
 10      CodeChauffeur NUMBER;
 11      BEGIN
 12         FOR i IN 2502553..3753830 LOOP
 13           CodeTrajet := i;
 14           DateTrajet := TO_DATE('2023-01-01','YYYY-MM-DD') + TRUNC(dbms_random.value(0, 365));
 15           DureeTrajet := NUMTODSINTERVAL(TRUNC(dbms_random.value(1, 100)), 'HOUR');
 16           KilometrageParcouru := dbms_random.value(10, 1000);
 17           CarburantConsomme := dbms_random.value(10, 500);
 18           CodeVehicule := TRUNC(dbms_random.value(1, 600));
 19           CodeDepart := TRUNC(dbms_random.value(1, 127344));
 20           CodeArrivee := TRUNC(dbms_random.value(1, 127344));
 21           CodeChauffeur := TRUNC(dbms_random.value(1, 2003));
 22           INSERT INTO Trajet (CodeTrajet, DateTrajet, DureeTrajet, KilometrageParcouru, CarburantConsomme, CodeVehicule, CodeDepart, CodeArrivee, CodeChauffeur)
 23           VALUES (CodeTrajet, DateTrajet, DureeTrajet, KilometrageParcouru, CarburantConsomme, CodeVehicule, CodeDepart, CodeArrivee, CodeChauffeur);
 24         END LOOP;
 25         COMMIT;
 26       END;
 27       /

PL/SQL procedure successfully completed.
```

b/ Augmentation de nombre d'instances de transactions à 100%

```
SQL> DECLARE
  2      CodeTrajet NUMBER;
  3      DateTrajet DATE;
  4      DureeTrajet INTERVAL DAY TO SECOND;
  5      KilometrageParcouru NUMBER;
  6      CarburantConsomme NUMBER;
  7      CodeVehicule NUMBER;
  8      CodeDepart NUMBER;
  9      CodeArrivee NUMBER;
 10      CodeChauffeur NUMBER;
 11      BEGIN
 12         FOR i IN 3753831..5005106 LOOP
 13           CodeTrajet := i;
 14           DateTrajet := TO_DATE('2023-01-01','YYYY-MM-DD') + TRUNC(dbms_random.value(0, 365));
 15           DureeTrajet := NUMTODSINTERVAL(TRUNC(dbms_random.value(1, 100)), 'HOUR');
 16           KilometrageParcouru := dbms_random.value(10, 1000);
 17           CarburantConsomme := dbms_random.value(10, 500);
 18           CodeVehicule := TRUNC(dbms_random.value(1, 600));
 19           CodeDepart := TRUNC(dbms_random.value(1, 127344));
 20           CodeArrivee := TRUNC(dbms_random.value(1, 127344));
 21           CodeChauffeur := TRUNC(dbms_random.value(1, 2003));
 22           INSERT INTO Trajet (CodeTrajet, DateTrajet, DureeTrajet, KilometrageParcouru, CarburantConsomme, CodeVehicule, CodeDepart, CodeArrivee, CodeChauffeur)
 23           VALUES (CodeTrajet, DateTrajet, DureeTrajet, KilometrageParcouru, CarburantConsomme, CodeVehicule, CodeDepart, CodeArrivee, CodeChauffeur);
 24         END LOOP;
 25         COMMIT;
 26       END;
 27       /

PL/SQL procedure successfully completed.

Elapsed: 00:05:09.32
```

c/ Tester la requête R2 sans VMNBT :

```
SQL> SELECT EXTRACT(MONTH FROM DateTrajet) AS Mois, EXTRACT(YEAR FROM DateTrajet) AS Annee
  2  FROM Trajet
  3  GROUP BY EXTRACT(MONTH FROM DateTrajet), EXTRACT(YEAR FROM DateTrajet);

      MOIS      ANNEE        NBT
---------- ---------- ----------
        12       2023     413941
         1       2023     413814
        10       2023     412264
         5       2023     414596
         3       2023     412446
         7       2023     412570
        11       2023     400418
         6       2024          7
         2       2023     372831
         6       2023     401000
         4       2023     400848

      MOIS      ANNEE        NBT
---------- ---------- ----------
         8       2023     413288
         9       2023     399635

13 rows selected.

Elapsed: 00:00:05.97

Execution Plan
----------------------------------------------------------
Plan hash value: 6047444428

--------------------------------------------------------------------------------
| Id  | Operation          | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |        |   369 |  2952 |  9012   (3)| 00:00:01 |
|   1 |  HASH GROUP BY     |        |   369 |  2952 |  9012   (3)| 00:00:01 |
|   2 |   TABLE ACCESS FULL| TRAJET |  2502K|    19M|  8829   (1)| 00:00:01 |
--------------------------------------------------------------------------------
```

APRES REFRESH :

```
SQL> EXECUTE DBMS_MVIEW.REFRESH('VMNBT');

PL/SQL procedure successfully completed.

Elapsed: 00:00:48.26
```

d/ Tester la requête R2 avec VMNBT :

```
SQL> SELECT EXTRACT(MONTH FROM DateTrajet) AS Mois, EXTRACT(YEAR FROM DateTrajet) AS Annee, COUNT(*) AS NBT
  2  FROM Trajet
  3  GROUP BY EXTRACT(MONTH FROM DateTrajet), EXTRACT(YEAR FROM DateTrajet);

      MOIS      ANNEE        NBT
---------- ---------- ----------
        12       2023     413941
         1       2023     413814
        10       2023     412264
         5       2023     414596
         3       2023     412446
         7       2023     412570
        11       2023     400418
         6       2024          7
         2       2023     372831
         6       2023     401000
         4       2023     400848

      MOIS      ANNEE        NBT
---------- ---------- ----------
         8       2023     413288
         9       2023     399635

13 rows selected.

Elapsed: 00:00:00.01

Execution Plan
----------------------------------------------------------
Plan hash value: 1052901248

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

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

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

|   0 | SELECT STATEMENT            |       |    13 |   156 |     3   (0)| 00:0
0:01 |
```

11.Le tableau comparatif de temps d'execution avant et apres :

AVANT VMNBT	APRES VMNBT
`Elapsed: 00:00:05.97` **La requête R2 nécessitait environ 0.5s secondes pour s'exécuter directement sur les tables.**	`Elapsed: 00:00:00.01` **Le temps d'exécution a été considérablement réduit à environ 0,01 seconde, illustrant ainsi l'importance de la vue matérialisée pour optimiser les performances de requête en diminuant les temps de calcul nécessaires pour obtenir les résultats.**

12. Conclusion :

-Amélioration de la Performance : Les vues matérialisées réduisent significativement le temps d'exécution des requêtes complexes en stockant des résultats pré-calculés.
-Optimisation des Ressources : Grâce à PL/SQL, il est possible d'automatiser l'insertion massive de données, rendant les processus plus efficaces et réduisant la charge sur le serveur.
-Simplicité d'Accès aux Données : Les vues matérialisées permettent de simplifier les requêtes, facilitant l'accès aux données sans recalculer à chaque fois.
-Maintenance : Bien que les vues matérialisées nécessitent des rafraîchissements réguliers, elles offrent une solution efficace pour les rapports et analyses avec des données semi-statique

Conclusion générale :

En conclusion, cette expérience a permis de mettre en évidence l'impact de l'augmentation du volume des données sur les performances des requêtes SQL. L'initialisation des tables et l'exécution des requêtes ont montré une augmentation significative des temps d'exécution avec un volume de données accru. Cependant, l'utilisation d'une vue matérialisée a permis de réduire ces temps, prouvant son efficacité dans l'optimisation des requêtes sur de grandes bases de données. Cette étude confirme que la gestion des vues matérialisées est une solution pertinente pour améliorer la réactivité des systèmes de gestion de bases de données lorsqu'on traite de gros volumes de données.