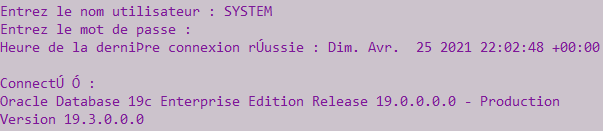


***Rapport 2 TP ENDO***

# **Objectif : Maîtrise des vues matérialisées**

**Tout d’abord on va se Connecter avec le nom user par défaut SYSTEM suivit du mot de passe.**

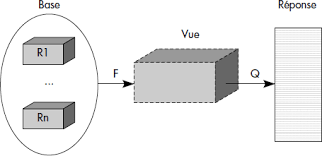


**PS : Dans toutes les requêtes on va évitez les jointures imbriquées.**

**Tout d’abord nous allons définir une vue matérialisée ainsi qu’une vue normale et la différence entre eux :**

1. **Vue normale**

**Une vue,** c'est tout simplement une requête ***SELECTE*** à laquelle on donne un nom. Lorsque l'on sélectionne des données à partir d'une vue, on exécute en réalité la requête SQL de la vue. Par conséquent, les vues ne permettent pas de gagner en performance. (Voir schéma suivant)



**Schéma de Vue**

Note : En fait, dans certains cas, les requêtes sur des vues peuvent même être moins rapides que si l'on fait directement la requête sur la ou les tables

1. **Vue Matérialisée**

Il fallait donc quelque chose qui combine les avantages des vues et les avantages des tables normales : c'est ainsi que le concept des vues matérialisées est né !

Une vue matérialisée est un objet qui permet de stocker le résultat d'une requête SELECT. Là où une vue se contente de stocker la requête, la vue matérialisée va stocker directement les résultats (elle va donc les matérialiser), plutôt que la requête. Lorsque l'on fait une requête sur une vue matérialisée, on va donc chercher directement des données dans celle-ci, sans passer par les tables d'origine et/ou une table temporaire intermédiaire.

1. **Mise à jour des vues matérialisées :**

On l'a dit, il ne suffit pas de créer une vue matérialisée à partir des données, il faut aussi la tenir à jour lorsque les données des tables d'origine changent.

Deux possibilités :

* Une mise à jour sur demande ;
* Une mise à jour automatique chaque fois qu'un changement est fait.

1. **Différence entre eux : Voire le tableau suivant :**

|  | ***Vue*** | ***Vue matérialisée*** |
| --- | --- | --- |
| **Stockage** | Une vue n’est jamais stockée, elle est uniquement affichée. | Une vue matérialisée est stockée sur le disque. |
| **La vitesse** | Traitement lent | Traitement rapide |
| **Définition** | Vue est la table virtuelle formée à partir d’une ou plusieurs tables ou vues de base. | La vue matérialisée est une copie physique de la table de base. |
| **Mis à jour** | La vue est mise à jour chaque fois que la table virtuelle (View) est utilisée. | La vue matérialisée doit être mise à jour manuellement ou à l’aide de déclencheurs. |
| **Utilisation de la mémoire** | La vue ne nécessite pas d’espace mémoire. | La vue matérialisée utilise l’espace mémoire. |

**Syntaxes General:**

**CREATE MATERIALIZED VIEW <**nomvue>  
**BUILD** {IMMEDIATE|DEFERRED}  
**REFRESH** {COMPLETE|FAST|FORCE|NEVER} {ON DEMAND|ON COMMIT}  
**ENABLE QUERY REWRITE  
AS SELECT ...;**

1. ***Créer une vue matérialisée VM1 contenant les ordonnances prescrites en Juillet 2020 en utilisant les*** ***options (IMMEDIATE, COMPLETE, ON DEMAND)***

Ps : Dans le TP précédent nous avons fixé la taille des codes identificateur ainsi que les numéros d’attributs à 10, la taille des noms à 50. Afin d’éviter tous dépassement en cas d’augmentation d’instances (Comme dans la question 13)

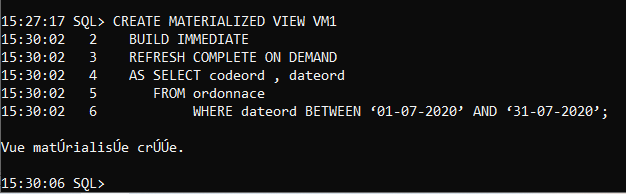
1. **Définir les options (options (IMMEDIATE, COMPLETE, ON DEMAND) :**

* IMMEDIATE : Création de la vue matérialisée et population de la vue
* COMPLETE : Recalcul complet de la vue
* ON DEMAND : Rafraîchissement avec DBMS\_MVIEW.REFRESH

1. **On rajoute une commande afin activer l’heur pour voir le temps d’exécution :**



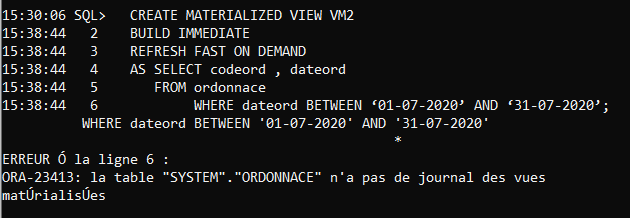
Maintenant on créer la vue matérialisée avec la commande ci jointe :



**Temps d’exécution : 4 s**

1. ***Créer une vue matérialisée VM2 identique à VM1, en utilisant les options (IMMEDIATE, FAST, ON DEMAND) :***

***Ps : Si on crée la vue de la même façon que VM1 on aura une erreur comme la suite, il nous a demander de créer un journal de vue matérialisées.***



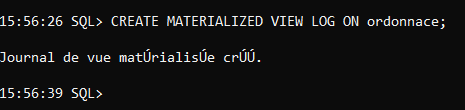
**Pour le cas FAST :** Un fichier de journalisation des données modifiées doit être créé à cet effet, on parle de MATERIALIZED VIEW LOG (tables “MLOG$\_\*”).

**Syntaxes SQL> CREATE MATERIALIZED VIEW LOG ON <NomTable>;**

Ceci ayant pour effet de créer une table MLOG$\_T sur l’host courant, une table qui

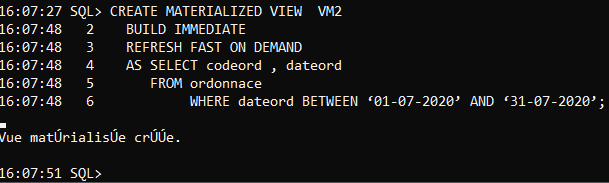
Contiendra à dater de cet instant toutes les modifications des lignes de la table T.

**Pour ce fait :**



**Temps d’exécution : 13 s**

**Maintenant on créer la VM2 :**

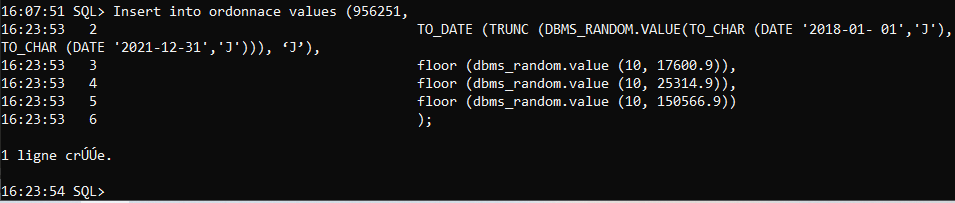


**Temps d’exécution : 3 s**

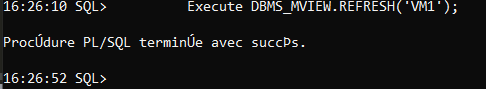
1. ***Testez les répercussions des mises à jour de la base de données, sur les deux vues (ajout, suppression, modification), en examinant et comparant les temps d’exécution du rafraichissement des deux vues.***

***=> Testez les répercussions des mises à jour de la table ordonnance sur les 2 vues :***

***1) L’insertion :***

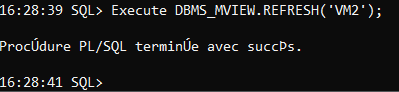


* **Rafraichissement des vues :**



**VM1**

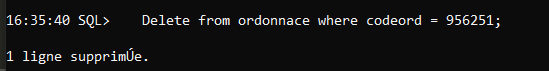
**Temps d’exécution : 42 s**



**VM2**

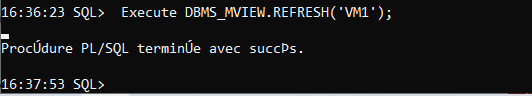
**Temps d’exécution : 2 S**

***2) Suppression :***

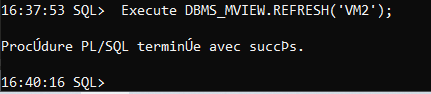


* **Rafraichissement des vues :**

**VM1**



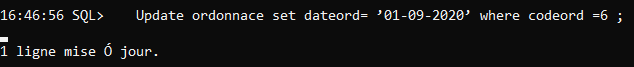
**Temps d’exécution : 90 s**



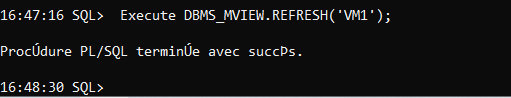
**VM2**

**Temps d’exécution : 93 s**

**3) La modification :**



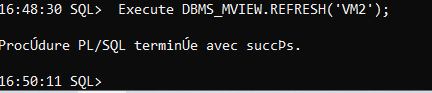
* **Rafraichissement des vues :**



**Temps d’exécution : 74 s**

**VM1**

**VM2**



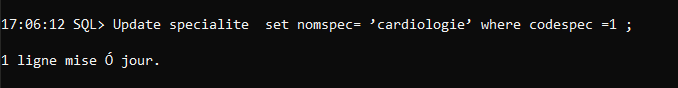
**Temps d’exécution : 68 s**

1. ***Activer les options autotrace, et timing d’oracle. (set autotrace on, set timing on)***

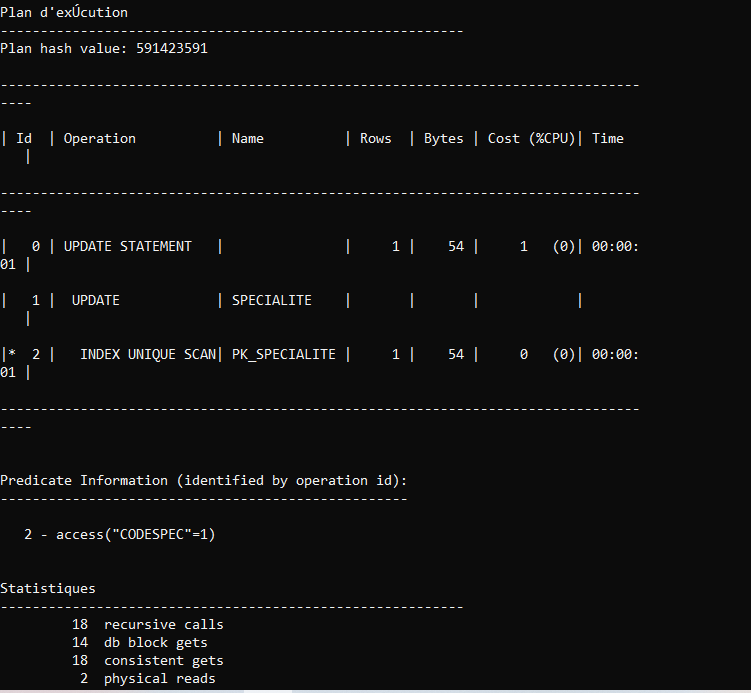


***Ps : Nous avons exécuter le SET TIMING ON dans la 1 erre étape du TP***

1. ***Ecrire une requête R1 pour obtenir la liste des Patients (CodePat, NomPat) traités par des médecins cardiologues.***

* ***Tout d’abord nous allons*** modifier une ligne quelconque de la table spécialité pour avoir la valeur cardiologie

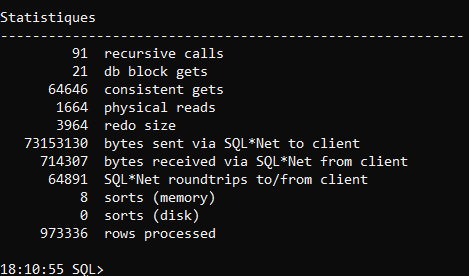
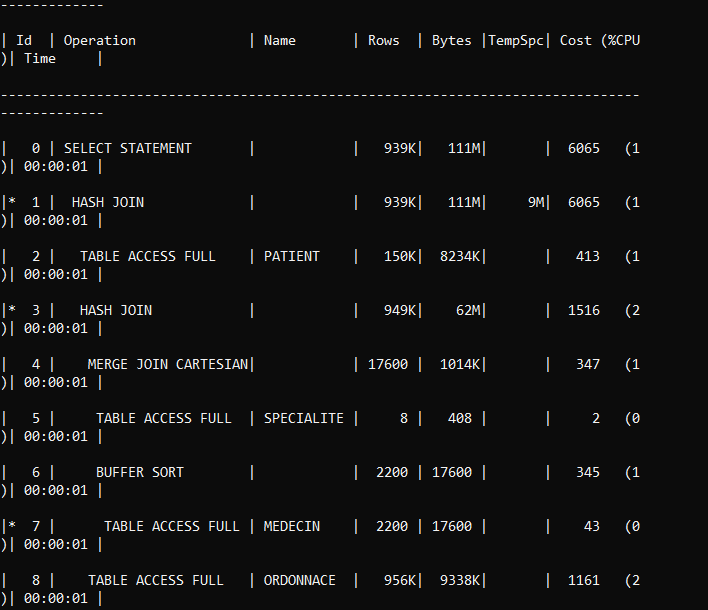
**On aura ausii un plan d’execution :**



**Maintenant écrire R1**



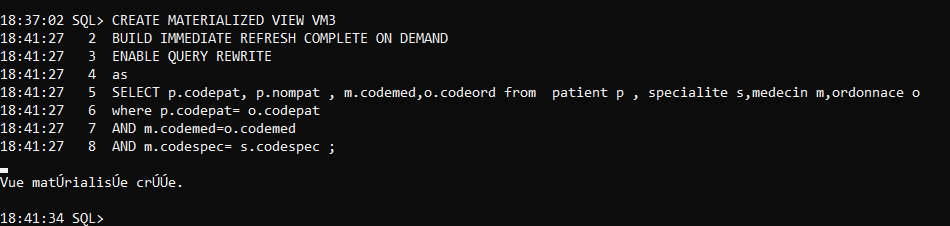
**Temps d’exécution : 11 min**

**Après avoir attendre voici les statistiques :** **973336 lignes sélectionnées.**

1. ***Créer une vue matérialisée VM3 en utilisant les options (IMMEDIATE, COMPLETE, ON DEMAND, ENABLE QUERY REWRITE) contenant une jointure entre les tables Patient, ordonnance, médecin et spécialité***

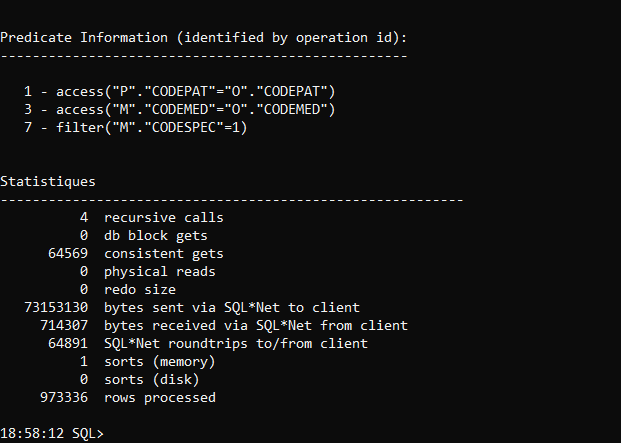
***L’OPTION Enable query rewrite :*** dans la commande de création d’une vue

Matérialisée permet l’exploitation de celle-ci dans la réécriture des requêtes sur les tables sources dans le but d’optimiser les temps d’accès.



1. **Ré exécuter la requête R1. Examiner le temps et le plan d’exécution et comparer avec (6)**

**Après avoir attendre voici les statistiques :** **973336 lignes sélectionnées.**



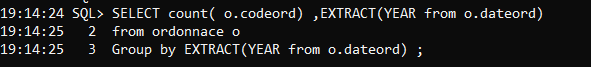
**Temps d’exécution : 9min 42 S**

**Ps : Temps d’exe(quest8) < Temps d’exe(quest6)**

1. ***Ecrire une requête R2 pour obtenir le nombre d’ordonnances par Année (Année, NBOrd) :***

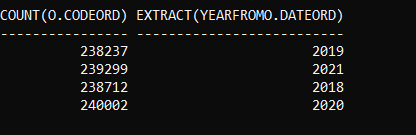
Il faut savoir qu’il existe une fonction dans oracle prédéfini qui extrait le jour / mois / année d’une date. C’est la fonction EXTRACT :

Voici la requête en utilisant aussi la fonction arithmétique de la somme Count qui nous renvoi le nombre d’ordonnances :

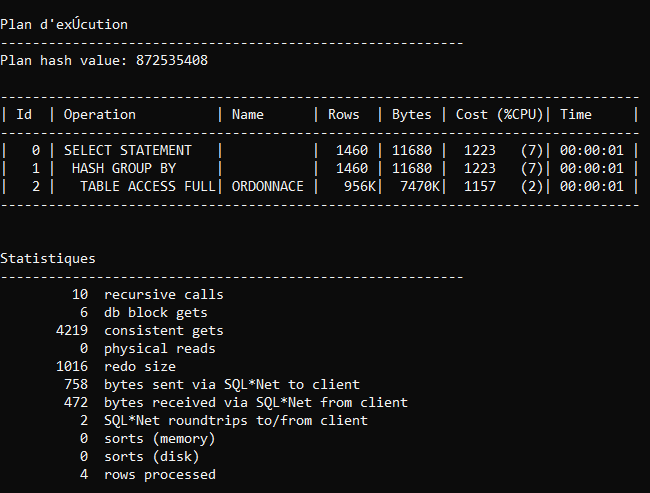


**Temps d’exécution : 2 s**

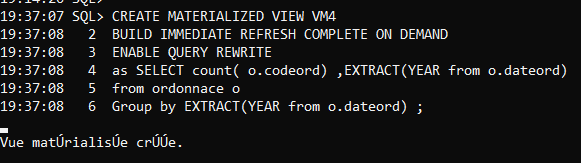
**Après avoir attendre voici les résultats ainsi que les statistiques :**



***Ainsi que*** ***le plan d’exécution :***

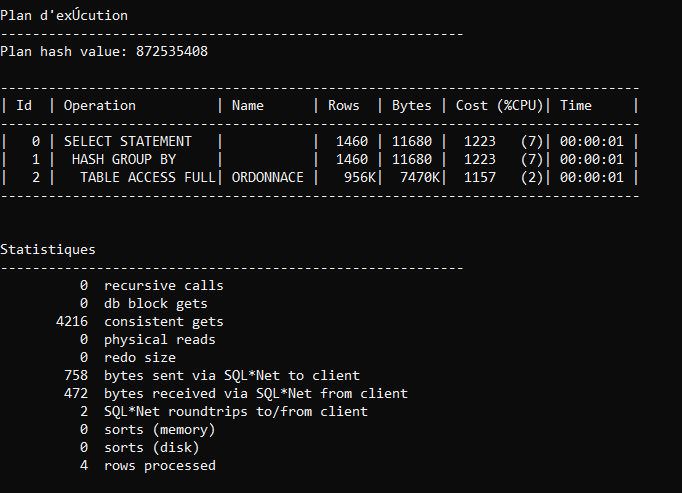


1. ***Créer une vue matérialisée VM4 (Année, NbOrd) en utilisant les options (IMMEDIATE, COMPLETE, ON DEMAND, ENABLE QUERY REWRITE)***

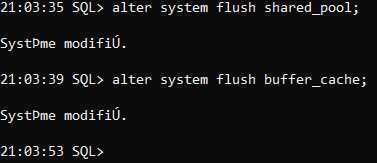


1. ***Ré exécuter la requête R2. Examiner le temps et le plan d’exécution et comparer avec (10). :***

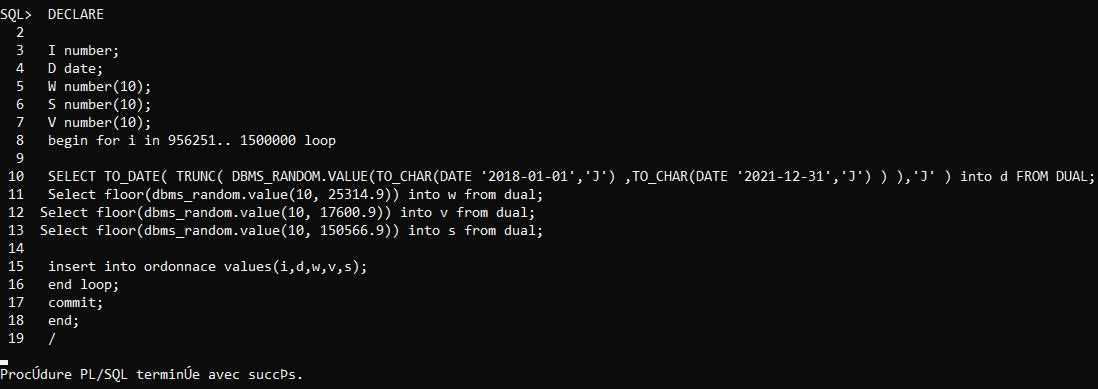
***Apres avoir exécuter R2 une deuxième fois le temps d’exécution est 1 Seconde et voici le plan d’exécution :***



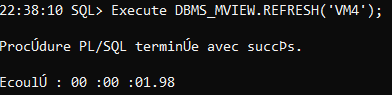
***H) On n’oublie pas de vider les buffers après chaque exécution des requêtes :***



1. ***Augmenter le nombre d’instances d’ordonnance à 1500000, puis à 2000000 et retester la requête R2 avec et sans la vue matérialisée VM4 à chaque fois.***

***a) Instances d’ordonnance à 1500000 : On a Just à modifier la requête de remplissage de table Ordonnance avec la nouvelle instance :***

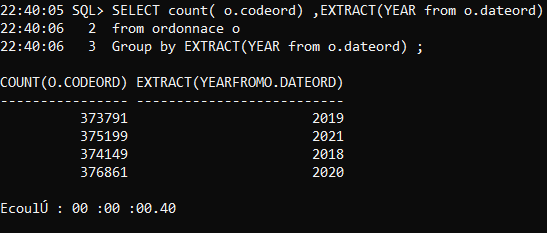
***On N’oublie pas de rafraichir les Vues :***



**Temps d’exécution : 2S**

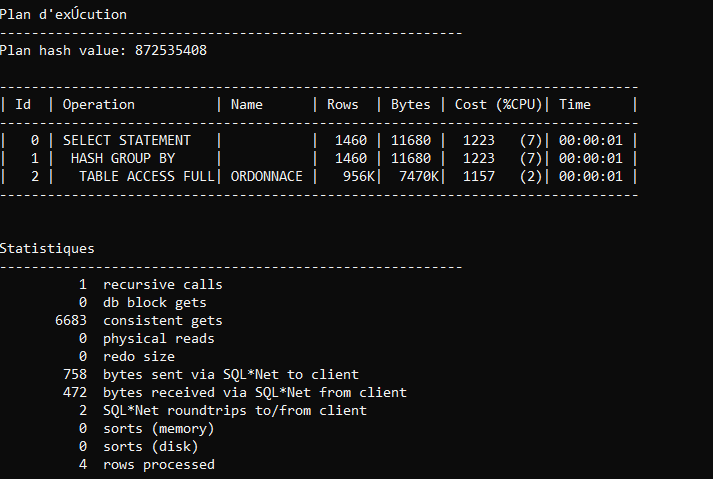
* ***Exécuter R2 sans vue :***

***Voilà le résultat :***

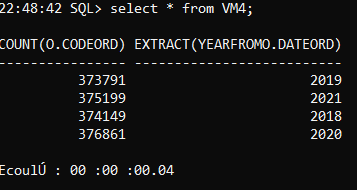


**Temps d’exécution : 40 ms**

***Plan d’exécution R2 AVEC 150000 :***

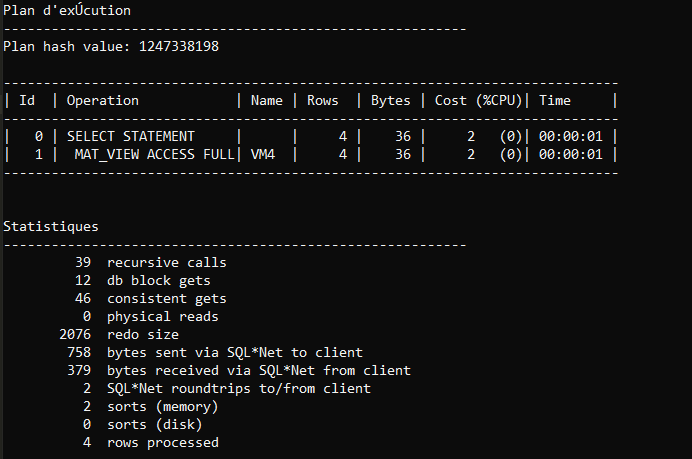


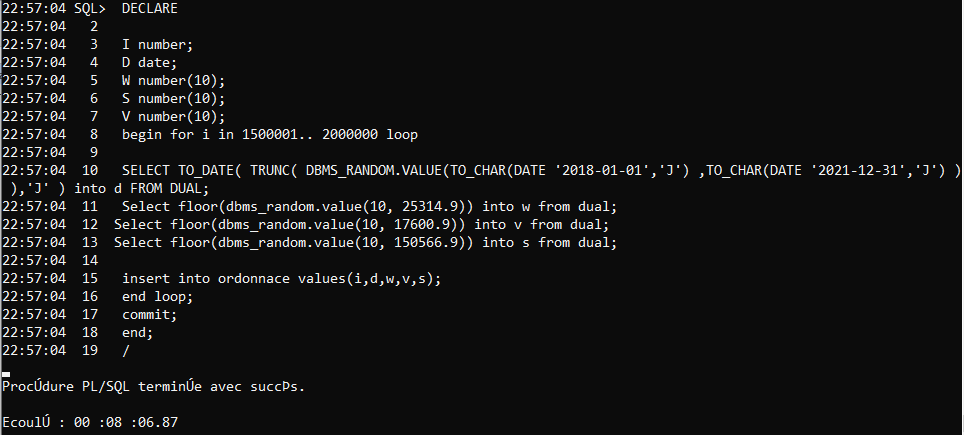
* ***Exécuter R2 avec VR4 :***



**Temps d’exécution : 4 ms**

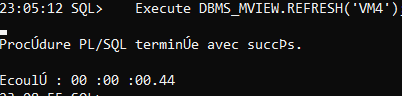
***Plan d’exécution 200000 :***



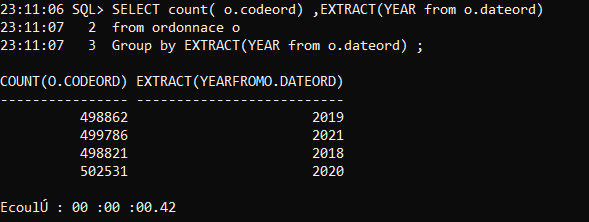
***b)*** ***Instances d’ordonnance à 2000000 : On a Just à modifier la requête de remplissage de table Ordonnance avec la nouvelle instance :***

**Temps d’exécution : 8 min 6 s 87 ms**

* ***RafraichireVM4 :***

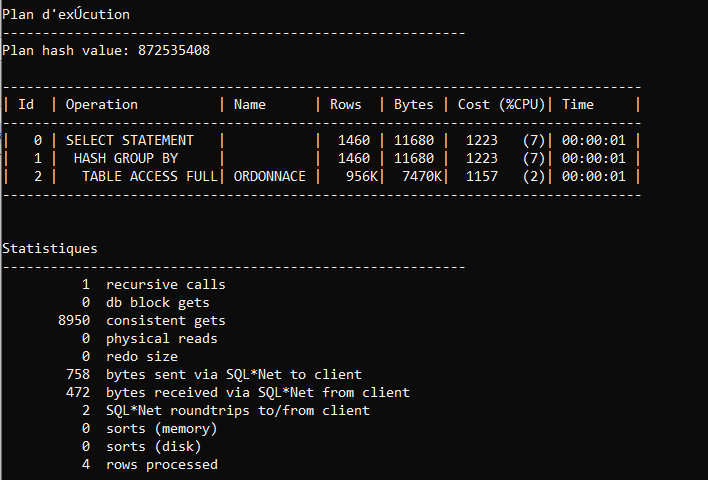


* ***Exécuter R2 sans Vue avec 200000 :***

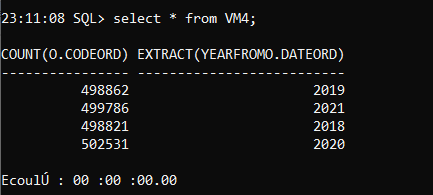


**Temps d’exécution : 42 ms**

***Plan d’execution :***

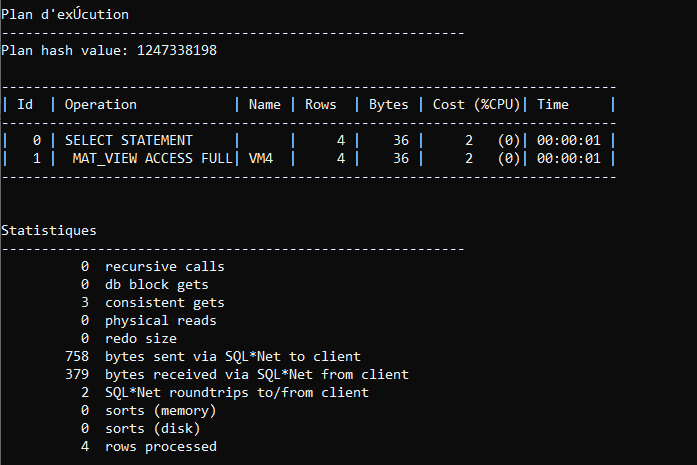


* ***Executer VR4 avec 200000 :***



**Temps d’exécution : 1 ms**

***Plan d’ecution VM4 200000 :***



1. ***Donner un tableau comparatif des temps d’exécution (avec et sans la vue matérialisée) en fonction du nombre d’instance.***

|  |  |  |  |
| --- | --- | --- | --- |
| ***/////////*** | ***956250*** | ***1500000*** | ***2000000*** |
| ***R2 sans Vue*** | ***//*** | ***40 MS*** | ***42 MS*** |
| ***VR4*** | ***//*** | ***4 MS*** | ***1 ms (Presque O négligeable)*** |
| ***R1 sans Vue*** | ***11 min*** | ***//*** | ***//*** |
| ***VR3*** | ***7 min 31 s 73 ms*** | ***//*** | ***//*** |

***Conclusion de ce TP :***

* Les **vues** sont essentiellement des structures logiques de type table remplies à la volée par une requête donnée. Les résultats d’une requête de **vue** ne sont stockés nulle part sur le disque et la vue est recréée chaque fois que la requête est exécutée. Les **vues matérialisées** sont des structures réelles stockées dans la base de données et écrites sur le disque. Ils sont mis à jour en fonction des paramètres définis lors de leur création.
* Le temps d’exécution de requête normal est moins rapide qu’une exécution d’une vue matérialisée (temps d’exe d’une vue matérialisée peut être négligé dans quelques cas)