\_

***Rapport 6 TP ENDO***

# **Objectif : Optimisation par index bitmap et fragmentation horizontale**

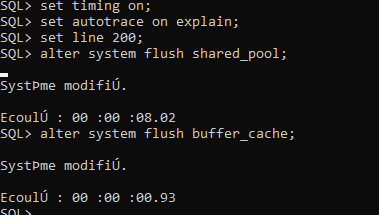
1. ***Ecrire une requête R1 qui donne le nombre de médicaments de type antibiotique (nom du type médicament).***

=> Nous commençons par la connexion avec l'utilisateur qui contient les tables de dimensions et la table de fait, Puis l'activation des variables d'environnement nécessaires:

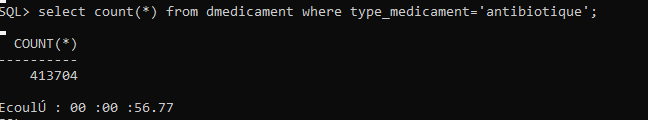
=>Tout d’abord on va modifier le nom de type medicament afin de le retrouver facilement vie le code de type medicament =1



**Configurere le timing et le plan d’execution :**

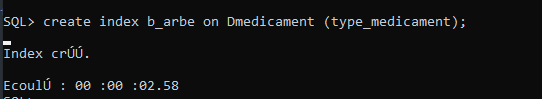


**La requette R1 :**



|  |  |
| --- | --- |
| **Temps d’exe** | **Exploitation des ressources** |
| **56 s77 ms** | **D medicament** |

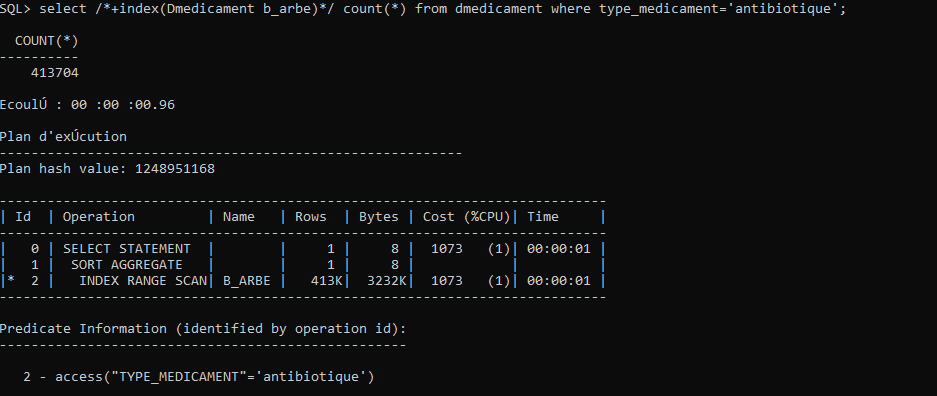
***2. Créer un index b-arbre de la table DMédicament sur l’attribut NomTypeMédicament***



=> L'arbre de recherche (B-Tree) rend l'index rapide. Les nœuds feuilles de l'index sont stockés dans un ordre arbitraire en ce sens que la position sur le disque ne correspond pas à la position logique suivant l'ordre de l'index. C'est comme un annuaire téléphonique avec les pages mélangées.

=>Un index de type B-Tree est fondé sur la notion d'arbre équilibré, qui permet dans le cas général d'assurer un temps de recherche constant quelle que soit la valeur recherchée.

***3. Réexécuter R1 en examinant le temps et le plan de l’exécution.***

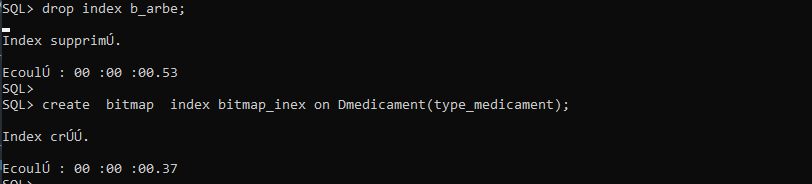


|  |  |
| --- | --- |
| **Temps d’exe** | **Exploitation des ressources** |
| **96 MS** | **INDEX B ARBRE** |

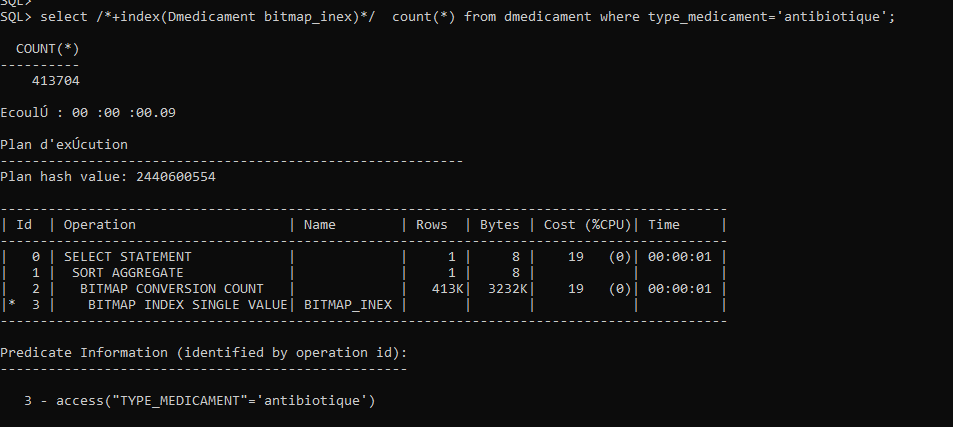
**Remarque:**

Le temps de l’exécution de la requete R1 après la creations de l’index b\_arbe est optimale (minimum) par rapport au la question 1.

1. ***Supprimer l’index b-arbre, et créer un index bitmap de la même table et sur le même attribut****.*



1. ***Réexécuter R1 et comparez entre les trois exécutions.***



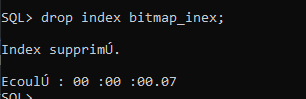
|  |  |
| --- | --- |
| **Temps d’exe** | **Exploitation des ressources** |
| **9 MS** | **INDEX B ARBRE** |

**Comparaison :**

Le bitmap a un temps d’execution minimum par rapoort a l’index et la requette .

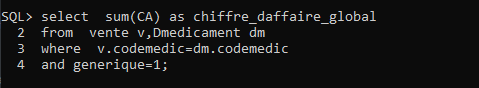
|  |  |  |  |
| --- | --- | --- | --- |
| **l’exécution de R1** | **R1 en qst1** | **R1 avec b\_arbe** | **R1 avec bitmap** |
| **Temps**  **d'exécution** | **00 :00 :56.77** | **00 :00 :00.96** | **00 :00 :00.09** |

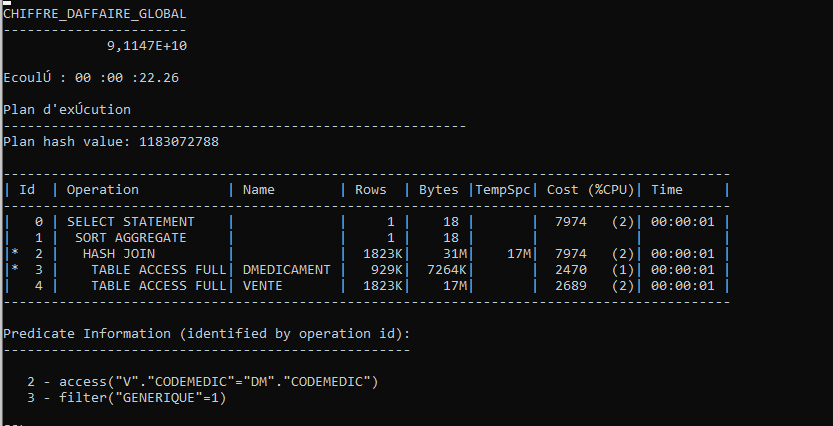
1. **Supprimer les deux index.**



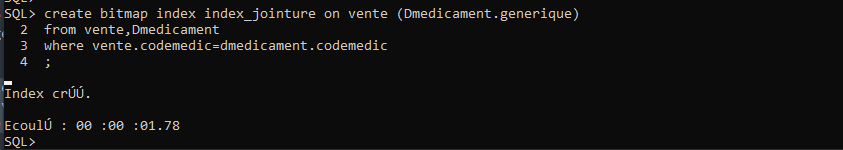
1. ***Ecrire une requête R2 qui donne le chiffre d’affaire global généré par des médicaments génériques.***

Posons que le medicament est generique si l’attribue Generique=1 ET NON GENEREIQUE = 2

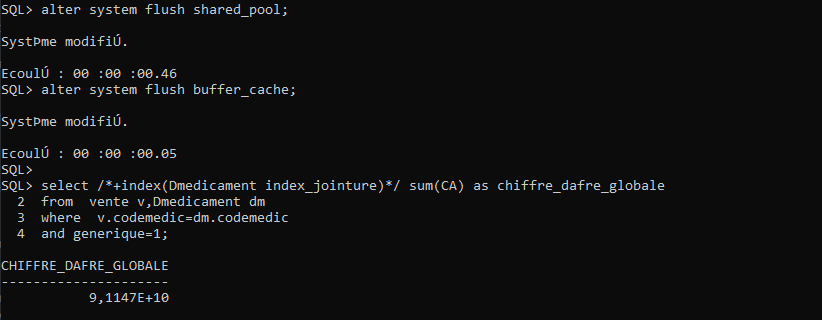


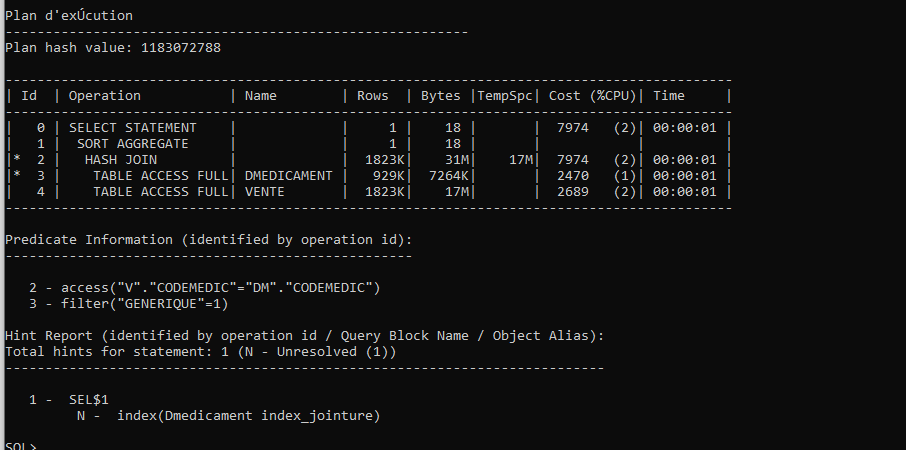


1. ***Créer un index bitmap de jointure entre Vente et DMédicament, basé sur l’attribut ‘Générique’.***



1. ***Réexécuter R2 et comparez entre les deux exécutions.***



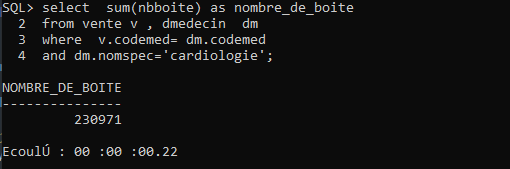


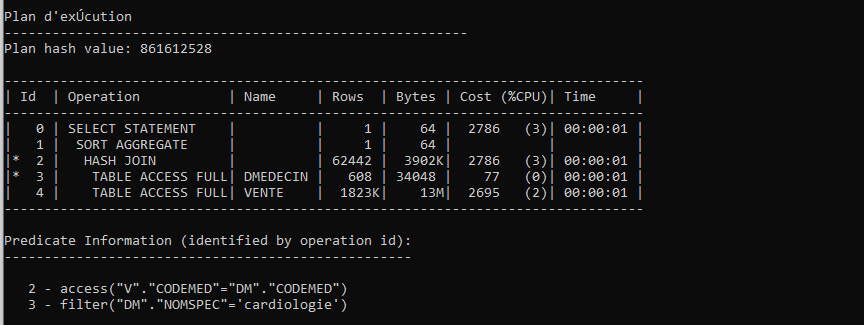
|  |  |
| --- | --- |
| **Temps d’exe** | **Exploitation des ressources** |
| **1 m20ms** | **D medecin vente , index** |

**Remarque :**

L’exécution n'exploite pas l’ index\_jointure, pour optimiser le temps de l’exécution, donc y'a pas de changement entre les deux résultats.

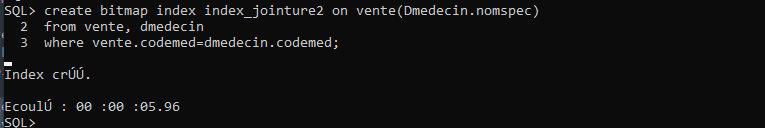
1. ***Ecrire une requête R3 qui donne le nombre de boites prescrites par des médecins cardiologues.***



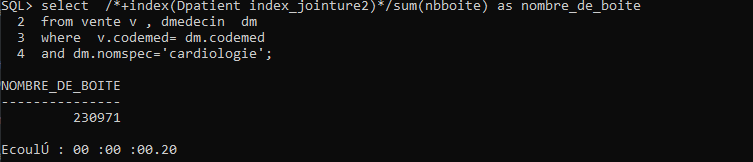


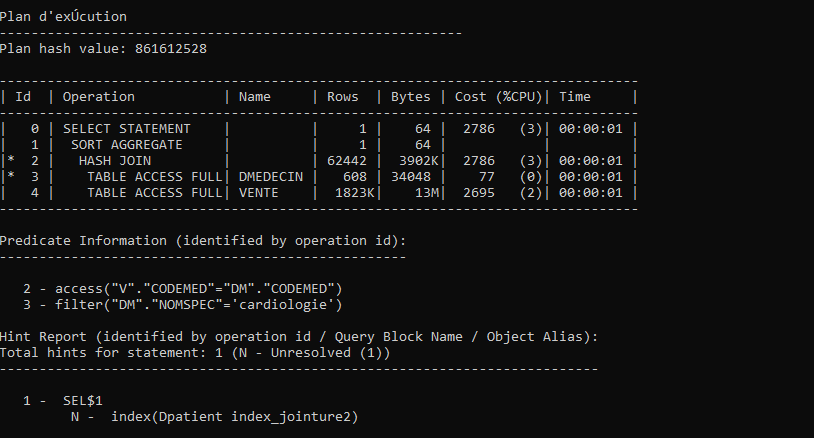
|  |  |
| --- | --- |
| **Temps d’exe** | **Exploitation des ressources** |
| **19 MS** | **D medecin** |

1. ***Créer un index bitmap de jointure qui améliore le temps de cette requête***



***12. Réexécuter R3 et comparer les deux exécutions.***



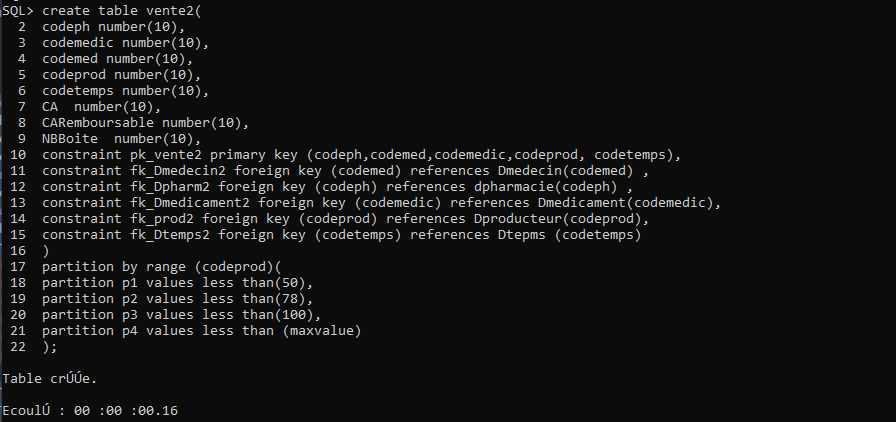


|  |  |  |
| --- | --- | --- |
| **l’exécution de R3** | **R1 en qst1** | **R3 avec bitmap** |
| **Temps**  **d'exécution** | **00 :00 :00:22** | **00 :00 :00.20** |

**Remarque :**

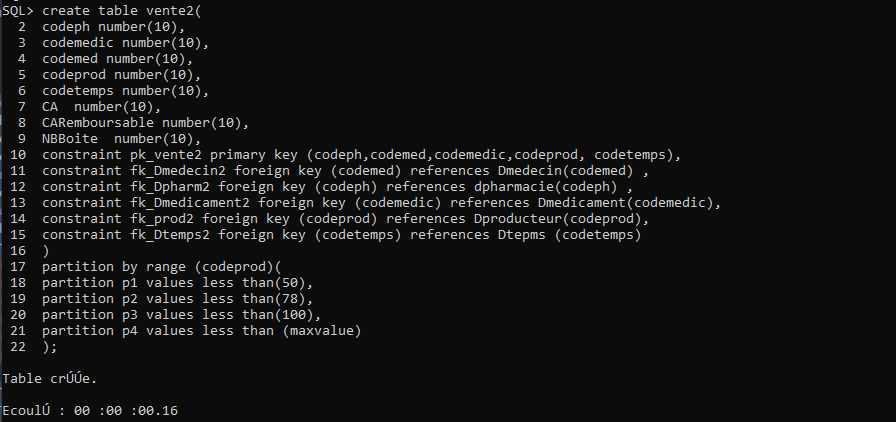
L’exécution exploite l’ index\_jointure, pour optimiser le temps de l’exécution, donc il ya de changement entre les deux résultats.

***13. Créer une table Vente2 identique à Vente, en la partitionnant en fonction des code de producteur comme suit : P1{1…50}, P2{51..78}, P3{79..100}, P4>100.***

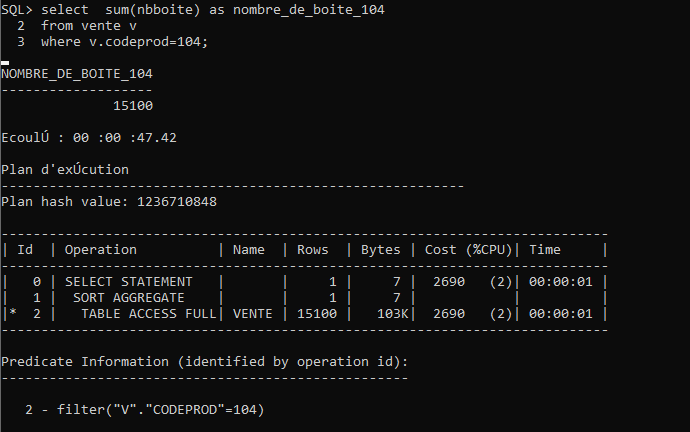


1. ***Remplir Vente2 les avec les mêmes instances que Vente.***

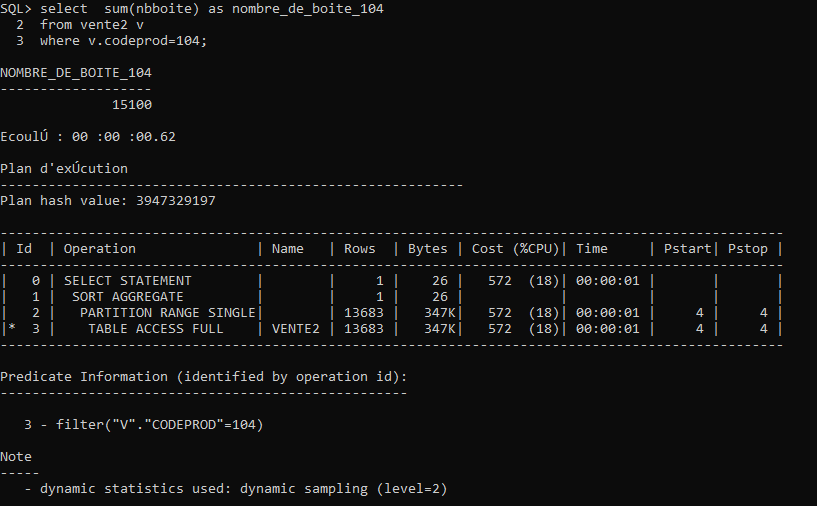
On utlise le meme script de remplissage de Vente dans TP3



15 ***Ecrire une requête R4 qui donne le nombre de boites global du producteur N°104 en utilisant la table Vente.***



1. ***Modifier R4 pour utiliser la table Vente2 et comparez les deux exécutions.***



On remarque que le temps d’execution dimiue au mininum apres utilsation de la table frangmenté ce qui facilte la recherceh et la projection des requettes

|  |  |  |
| --- | --- | --- |
| **Requete R4** | **Fhospitalisation** | **Fhospitalisation2** |
| **Temps d'exécution** | **00 :00 :47:42** | **00 :00 :00.62** |

1. ***Y-a-t-il une solution pour partitionner une table existante, si oui l’appliquer pour partitionner vente (selon le même critère que vente2).***

**=> Voici quelques suggestion que j’ai trouvé :**

* Pour partitionner une table, vous pouvez suivre les étapes ci-dessous:
* créez d'abord une fonction de partition et un schéma de partition .
* Après cela, vous pouvez partitionner une table.
* SI votre table a un index clusterisé, vous devez le supprimer et le recréer sur la partition de droite ou vous pouvez utiliser la DROP\_EXISTINGclause pour recréer l'index clusterisé.
* Si votre table n'a pas d'index clusterisé, vous pouvez simplement en créer un sur la partition de droite à l'aide du schéma de partition.
* Enterprise Edition a également la ONLINE=ONpossibilité d'utiliser l' option de l'instruction CREATE INDEX pour minimiser les temps d'arrêt de votre application. Notez que vous verrez une dégradation des performances lors de la reconstruction de l'index à l'aide de l'option EN LIGNE.
* Il faut utiliser le package DBMS\_REDEFINITION mais afin de travailler celui-ci aura besoin d'une table intermédiaire dont la structure devra avoir été créée à l'image de ce que l'on veut.

***-\*-\*-\*-\*-\* Merci \*-\*-\*-\****