Entrepôts de Données TP2

Objectif : Maîtrise des vues matérialisées

Syntaxe

CREATE MATERIALIZED VIEW <nomvue>
BUILD { IMMEDIATE|DEFERRED }
REFRESH { COMPLETE|FAST|FORCE|NEVER } { ON DEMAND|ON COMMIT }
ENABLE QUERY REWRITE
AS SELECT ... ;
Options
IMMEDIATE : Création de la vue matérialisée et population de la vue
DEFERRED : Création de la vue matérialisée sans être alimentée en données.
DBMS_MVIEW.REFRESH(<liste_vues>) alimente la vue
ON COMMIT : Rafraîchissement à chaque fin de transaction modifiant les tables sources
ON DEMAND : Rafraîchissement avec DBMS_MVIEW.REFRESH
COMPLETE : Recalcul complet de la vue
FAST : Application d'un rafraîchissement incrémental
FORCE : FAST si possible, COMPLETE sinon NEVER : pas de rafraîchissement

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.

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$_*").
Syntaxe SQL> CREATE MATERIALIZED VIEW LOG ON <NomTable>;

Ceci ayant pour effet de créer une table MLOG$_T sur le host courant, une table qui contiendra à dater de cet instant toutes les modifications des lignes de la table T.

Travail à faire : Attention dans toutes les requêtes évitez les jointures imbriquées.
A partir du schéma du TP1 :

1. Créer une vue matérialisée VM1 contenant les appels réalisée en Juin 2021 en utilisant les options (IMMEDIATE, COMPLETE, ON DEMAND)
2. Créer une vue matérialisée VM2 identique à VM1, en utilisant les options (IMMEDIATE, FAST, ON DEMAND)
3. 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.
4. Activer les options autotrace, et timing de oracle.
5. Ecrire une requête R1 pour obtenir la liste des clients (CodeCl, NomCl) ayant effectué des appels de type internaional.
6. Examiner le temps et le plan d'exécution.
7. Créer une vue matérialisée VM3 (CodeCl, NomCl, CodeTypeAppel, TypeAppel) en utilisant les options (IMMEDIATE, COMPLETE, ON DEMAND, ENABLE QUERY REWRITE) contenant une jointure entre les tables Client, Ligne, Appel et TypeAppel.

8. Ré exécuter la requête R1. Examiner le temps et le plan d'exécution et comparer avec (6).
9. Ecrire une requête R2 pour obtenir le nombre d'appels par mois, année (Mois, Année, NBApp)
10. Examiner le temps et le plan d'exécution.
11. Créer une vue matérialisée VM4 (Mois, Année, NBApp) en utilisant les options (IMMEDIATE, COMPLETE, ON DEMAND, ENABLE QUERY REWRITE)
12. Ré exécuter la requête R2. Examiner le temps et le plan d'exécution et comparer avec (10).
13. Augmenter le nombre d'instances de Appel à ~~3500000,~~ puis à 4000000 et retester la requête R2 avec et sans la vue matérialisée VM4 à chaque fois. (N'oubliez pas de rafraichir la vue après chaque augmentation du nombre d'instances : commande : Execute DBMS_MVIEW.Refersh('VM4') ;).
14. Donner un tableau comparatif des temps d'exécution (avec et sans la vue matérialisée) en fonction du nombre d'instances.
15. Quelles Conclusions tirez-vous de ce TP?

Indications :

- N'utilisez pas de requêtes imbriquées pour les jointures.
- Avant toute exécution des requêtes vider tous les buffers à l'aide des commandes :

```
alter system flush shared_pool;
alter system flush buffer_cache;
```