TP 2 OPTIMISATION

Récupérer les données de travail : DONNEES_COMMANDE_OPTIM.sql. Supprimer les tables existantes et créer les nouvelles. Pensez à mettre à jour la colonne prix total (cf 1ier TD).

Pour identifier les index que vous avez créés :
SQL> *select* ***index_name*** *from user_indexes where index_name not like 'SYS% ' and table_name = 'EMP'*;
Pour supprimer les index : *drop index* <***index_name***> ;
Pensez à collecter les statistiques des tables/index.

Exercice 1 :

1. Exécuter la requête suivante : *CREATE INDEX ix_cl ON CLIENT(id);*
2. Comment expliquez-vous le résultat obtenu ?

Exercice 2 :

Soit la requête suivante :
*SELECT * FROM LIGNE_COMMANDE WHERE COMMANDE_ID=46;*

1. Exécuter la requête et observer le plan d'exécution.
2. Créer l'index suivant : *CREATE INDEX ix_commande_id ON LIGNE_COMMANDE (COMMANDE_ID, PRODUIT_ID) ;*
3. Exécuter la requête. Qu'observez vous au niveau plan d'exécution ?
4. Exécuter la requête suivante : *SELECT PRODUIT_ID FROM LIGNE_COMMANDE WHERE COMMANDE_ID = 46;*
5. Observez le plan d'exécution. Que remarquez-vous ? **N'hésitez pas à m'appeler si vous ne remarquez rien**.
6. Supprimer l'index créé.

Exercice 3 :

On considère la requête suivante :

```
SELECT cl.nom, SUM(lc.prix_total)
FROM CLIENT cl
INNER JOIN COMMANDE c ON c.client_id = cl.id
INNER JOIN LIGNE_COMMANDE lc ON lc.commande_id = c.id
GROUP BY cl.nom
```

1. Exécuter la requête et observer le plan d'exécution. Quelle méthode de jointure est utilisée ? Comment sont accédées les données des différentes tables ?
2. Proposer une solution d'optimisation en créant un ou plusieurs index.
3. Réexécuter la requête avec le(s) index créé(s).
4. Supprimer le(s) index créés.

Exercice 4 :

Soit la requête suivante :
*SELECT **

FROM LIGNE_COMMANDE
WHERE PRODUIT_ID IN (SELECT PRODUIT_ID FROM PRODUIT WHERE PRIX_UNITAIRE >100);

1. Exécuter la et regarder le plan d'exécution
2. Réécrivez la requête en utilisant une jointure. Exécuter la. Que remarquez-vous ?
3. Réécrivez la requête en utilisant NOT EXISTS ou EXISTS. Exécuter la. Que remarquez-vous ?
4. Que pouvez-vous conclure ?

Exercice 5 :

Créer une table BIGBITMAP ayant trois colonnes id, nbitmap et date_insertion.
Exécuter le script ci-dessous 3 fois pour insérer les lignes dans la table.

```
BEGIN
    for id in 1..100000 LOOP
     INSERT INTO BIGBITMAP VALUES
       (  id,
          MOD(id,5) ,
         TO_DATE(sysdate, 'DD/MM/YYYY')
       );
     Commit;
  END LOOP;
  dbms_output.put_line ('100000 lignes inserees');
END;
/
```

Le champs nbitmap aura une faible cardinalité (5 pour cet exemple : nbitmap contient les valeurs 0,1,2,3,4)

1. Créer un index btree sur la colonne nbitmap.
2. Générer le plan d'exécution.
3. Exécuter les requêtes suivantes; Observer le temps et le plan d'exécution .
   select count(*) from bigbitmap where nbitmap = 0 ;
   select sum(id) from bigbitmap where nbitmap = 0 ;

4. Créer un index bitmap sur la colonne nbitmap.
5. Générer le plan d'exécution.
6. Exécuter les deux requêtes ci-dessus. Commenter.
7. Supprimer la table créée.