# TP Bases de données : indexation

## *Création d'une relation, indexation, mesure du temps et explication de plans d'exécution (source : Laurent Amsaleg)*

1. Copiez les fichiers du tp dans votre compte.
2. Connectez-vous à la machine oracle10 : *ssh <votre login>@oracle10*

.

L'environnement d'Oracle en mode texte étant rédhibitoire, il est conseillé d'avoir deux fenêtres :

- un éditeur de texte pour rédiger ses commandes
- une fenêtre shell pour lancer les commandes sous Oracle

Le fichier scramble.data contient les données qui vont plus tard être insérées dans relation XXXX. Le schéma de cette relation est :

*nss INTEGER,*
*nom VARCHAR(100) NOT NULL,*
*dist_10 INTEGER,*
*dist_100 INTEGER,*
*dist_1000 INTEGER,*
*dist_10000 INTEGER,*
*random FLOAT*

La base n'est pas encore créée dans Oracle. Utilisez des commandes Unix pour répondre aux deux questions suivantes :

3. Combien de tuples contient cette relation? (commande wc -l)

Le but du TP est d'essayer de créer des index sur la relation XXXX et de voir l'effet qu'a leur utilisation sur les performances du système. Le but est aussi d'essayer de voir ce qui est exécuté par la machine. A partir de maintenant, on va mettre des données dans Oracle.

## *Création d'une table en mode non indexé (HEAP)*

4. Regardez le scripts "heap_create_table.sql". Il créé une table non indexée dans Oracle. Exécuter ce script (sqlplus est l'interface de commande avec Oracle) :
   *sqlplus / @heap_create_table.sql*

5. Chargez les données dans la table XXXX :
   *sqlldr / control=load.sql data=scramble.data*

6. Que font les scripts point_query.sql et range_query.sql ?
7. Quelle est la différence fondamentale entre point_query.sql et "p_q_mauvais.bash" ? Pourquoi ne faut il pas faire des tests de performance avec celui-ci?

8. lancez sqlplus pour qu'il exécute point_query.sql, en effectuant un chronométrage (notez le résultat, faite cela 2 fois de suite) :

   *time sqlplus / @point_query.sql*

9. Que représentent les 3 lignes qui ont sont montrées dans le résultat ?
10. Faire de même pour range_query.sql. Les résultats diffèrent-ils ?

**Visualisation du plan d'exécution choisi**

11. Lancez l'interprêteur interactif sqlplus

    *sqlplus /*

12. Activez le traçage des exécutions

    *set autotrace on*

13. Lancez la requête

    *select * from XXXX where nss=21954;*

Qu'obtenez vous ? Essayez d'interprêter chaque ligne du résultat.

**Vous êtes arrivé au checkpoint, bravo ! Prévenir votre chargé de TP !**

14. En utilisant *set autotrace traceonly*, vous pouvez-faire l'économie de l'affichage du résultat de la requête.
15. Expliquez l'exécution de la requête :

    *select * from XXXX where nss>16873 and nss<16973*

16. Faire *drop table XXXX ;*

## *Création d'une table avec index*

17. Regardez maintenant le script *hash_create_table.sql* et refaites la séquence de tests vous donnant les temps de réponse puis les explications de plans.
18. Cherchez dans la doc en ligne ce que créé hash_create_table.sql.
19. Que penser des temps de réponse comparés a ceux du stockage HEAP ?
20. Supprimez la table et l'Index en faisant

    *drop table XXXX;*
    *drop cluster h_primaire_XXXX;*

21. Refaire les mêmes tests avec *bTree_create_table.sql*.Proposez une conclusion.
22. Supprimer l'index avec drop index XXXX_ix_nss.

**Vous êtes arrivé au checkpoint 2 ! Prévenir votre chargé de TP !**

23. Sur le modèle de la question 9, mesurer le temps d'une requête ponctuelle pour une base de plus en plus grande (10 000 n-uplets, 50 000, 100 000, 500 000, 1 000 000), sans index.
24. Tracez la courbe.
25. Refaire de même mais avec un index b-tree. Comparez.

**Avant de partir, effacez scramble.data de votre répertoire.**