**Statistique**

1) select * from user_tab_statistics where table_name = 'UNICODE';

Renvoie suite gather_table_state, à le nombre de blocs, de lignes, la longueur moyenne de ligne, la date de la dernière analyse...

select * from user_tab_col_statistics where table_name = 'UNICODE';

Renvoie les valeur minimale, maximales, la densité, la taille...

2) La colonne histogramme de la table user_tab_col_statistics indique 'FREQUENCY' pour Category.

**Plan d'éxécution**

1) SELECT u1.codepoint, u1.charname FROM unicode u1 JOIN unicode u2 ON u2.codepoint=u1.uppercase WHERE u2.category='Lu'

-> l'optimisseur effectue un full-scan sur les 2 tables en filtrant la catégorie sur la table u2, pour pouvoir procédé a une jointure par hachage car il ne possède pas d'index sur les attributs de jiontures des 2 tables

**Les opérateurs**

1) select * from unicode : l'optimiseur effectue un full-scan car on veut récupérer tous les tuples de la table unicode

2) select * from unicode where category='Lu';: l'optimiseur effectue un full-scan et un filter pour obtenir les tuples de catégorie 'Lu'

3) select * from unicode where codepoint = '0107'; l'optimisuer effectue un index unique scan sur le codepoint car celui-ci possède un index unique, pour filtrer le codepoint '0107'

4) select * from unicode where codepoint between '0105' and '0112';

l'optimiseur effectue un index range scan pour trouver les valeurs comprises entre 005E' et '00BA'; Etat un index sur clé primaire, celui-ci est trié par ordre croissant, donc le range scan est plus efficace.

5) select * from unicode where numeric_='1'; l'optimiser effectue un talbe acces by index rowid car numeric_ possède un index, donc il filtre grâce à l'index.

6) même que dans le plan d'éxécution

7) SELECT u1.codepoint, u1.charname FROM unicode u1 JOIN unicode u2 ON u2.codepoint>u1.uppercase WHERE u2.category='Lu';

L'optimiseur effectue un merge join, car il doit trié les tables pour savoir quelle éléments joindre (dù à l'opérateur de jointure '>')

8)NESTED LOOP: Il faut faire une jointure avec une petit table pour que l'optimiseur favorise cette jointure

9) select codepoint, count(*) from unicode group by codepoint; l'optimiseur procéde à un hash group by car c'est l'opérateur le plus efficace pour effectuer des group by.

10) select * from unicode order by codepoint; L'optimiseur utilise l'opérateur sort pour effectuer l'order by

11) select * from unicode where category='Lu' union  select * from unicode where codepoint = '0107';  L'optimiseur effectue l'opérateur union pour effectuer les 2 requêtes.