TP IBD N°05 (semaine 40)
Langage SQL – Interrogation des données

Syntaxe générale d'une requête SQL :

SELECT [DISTINCT] …
FROM …
[WHERE …]
[ORDER BY …] ;

Opérateurs:

[NOT] LIKE,
BETWEEN .. AND ..,
IN (…, …, …) ,
IS [NOT] NULL
=, !=, < >, <=, >=, < , >

Opérateurs logiques : OR, AND, NOT

Concaténation : ||

*Créer un fichier **tp05.sql** pour y stocker les requêtes demandées dans ce TP.*

On donne en gras, pour chaque requête, l'intitulé des colonnes à afficher. Si nécessaire, utiliser les commandes SQLPLUS pour une meilleure présentation des résultats de vos requêtes.

Il est demandé d'indenter correctement les requêtes (s'inspirer des exemples du cours), de créer un fichier de trace (tp05.lst) et mettre des commentaires indiquant ce que fait chaque requête ainsi que son numéro.

Projections – Sélections – Tris

R1 (15 min) : Liste des clients (nom et numéro de téléphone) qui sont des collectivités publiques (typeClt = '03') de la ville de REIMS.

Tri par ordre alphabétique sur le nom du client (ORDER BY).

Afficher : **Nom Client** **Téléphone**

Attention, contrairement à ACCESS, **ORACLE** est sensible à la casse pour les chaînes de caractère utilisées dans les critères de sélection.
Pour lever toute incertitude, on pourra utiliser la fonction **UPPER** sur le champ recherché.
Exemple : **UPPER(villeClt) = 'REIMS'**
ATTENTION : Sous Oracle, les chaînes de caractères utilisées dans les conditions de la clause WHERE doivent être entre simples quotes : 'TOTO'

Représenter d'abord la requête sous forme d'arbre relationnel puis en langage SQL.

R2 (15 min) : Liste des menus contenant le mot « HAWAI » ainsi que les menus de type traditionnel (typeMenu 4). Tri par ordre alphabétique du libellé de menu.

Afficher : **Menu**
(libMenu)

Sous **ORACLE**, les caractères génériques, utilisés pour remplacer les caractères inconnus d'une chaîne de caractère dans une condition de sélection, sont :

% (n caractères inconnus)
_ (1 caractère inconnu)

Ils s'utilisent avec l'opérateur **`LIKE`** ou **`NOT LIKE`**

Exemple : `UPPER(villeClt) NOT LIKE '%REIMS%'`
(pour vérifier que le champ villeClt ne contient pas le mot « REIMS »)

Représenter d'abord la requête sous forme d'arbre relationnel puis en langage SQL.

R3 (20 min) : Liste des éléments de menus dont le nombre de calories (nbCalories) est compris entre 50 et 100 et dont soit le prix unitaire prévu (puPrévu) est inférieur à 1€ soit le coût de revient (coutRevient) est inférieur à 1.50€
Utiliser opérateur **`BETWEEN`** et opérateurs logiques **`AND`** et **`OR`** en faisant attention aux priorités.

Afficher : **Element Menu**
(libEltMenu)

R4 (30 min) : Liste des clients qui sont soit des comités d'entreprise (typeClt 01) soit des collectivités publiques (typeClt 03) mais qui sont, dans tous les cas, sans parrain (`NULL`) et localisés dans la Marne (51).
Tri et rupture sur la ville. Afficher à chaque rupture le nombre de clients et sauter une ligne.

Afficher : **Client** (nomClt) **Ville** (villeClt) **Client depuis** (dateClt)

Où Client depuis : datClt sous la forme « `month YYYY` » (Ex. : Juin 2005)

La fonction **`TO_CHAR`** utilisée dans la clause `SELECT` d'une requête permet de formater un champ de type `DATE` dans le format souhaité.

Syntaxe : `TO_CHAR (ma_date, format)`

où format composé de `DD, DAY, MM, MON, MONTH, YY, YYYY` etc…

Exemples : `TO_CHAR (SYSDATE, 'Day dd MON')` → Jeudi 18 SEPT.
`TO_CHAR (SYSDATE, 'YYYY')` → 2014

R5 (40 min) : Liste des clients qui sont :

- Soit des municipalités (typeClt = '02') dont le nom commence par COM, clients depuis au moins 2000 (dateClt) et ayant un parrain (numParrain IS NOT NULL),
- Soit des collectivités publiques (typeClt = '03') localisées à VAUCIENNES, AY ou SEZANNE (villeClt) et dont le nom (nomClt) contient le mot 'Communaute'.

Liste triée par type puis sur l'ancienneté (du plus ancien au plus récent).
On utilisera un IN plutôt que plusieurs OR pour vérifier la localisation des clients de type 3.

Afficher : **Client** **Type Client** **Client depuis** **Localisation**

Où Client : Nom du client
Type Client : {Municipalité | Collectivité} *(utiliser la fonction DECODE)*
Client depuis : dateClt sous la forme « DD Month YYYY » (Ex. : 15 Août 2005)
Localisation : cpClt et villeClt *(concaténés et séparés par un espace)*

La fonction **`DECODE`** utilisée dans la clause `SELECT` d'une requête permet de décoder un champ prenant différentes valeurs.

Syntaxe : `DECODE (col, val_1, si_val_1, val_2, … , si_autres)`

Exemples : `DECODE (sexe, 1, 'Homme', 'Femme')`
`DECODE (cvlt, 1, 'M', 2, 'Mme', 'Melle')`

La fonction **`EXTRACT`** permet d'extraire une partie donnée d'une date.

Syntaxe : `EXTRACT(YEAR|MONTH FROM col_date)`

Exemple : `EXTRACT(MONTH FROM SYSDATE)` → 09