SQL : Deuxième partie

I : Les prédicats NULL, IN, LIKE, BETWEEN

A : le prédicat NULL :

Un attribut peut avoir la valeur "NULL" soit en raison d'information incomplète (la valeur n'était pas connue au moment de la saisie des données, *ex : numéro de téléphone*) soit parce que la donnée n'est pas pertinente. Dans une table qui modélise des individus, l'attribut "Nom Marital" n'est pas pertinent pour les individus de sexe masculin et a donc la valeur "NULL". La valeur "NULL" est différente de la valeur par défaut de l'attribut : zéro pour un attribut de type numérique et espace pour un attribut de type caractère.

La syntaxe est : **IS NULL** et sa négation **IS NOT NULL.**

- Numéro, nom des articles dont le prix d'achat n'est pas connu

SELECT art_num,art_nom
FROM ARTICLES
WHERE art-pa **IS NULL**;

B : le prédicat IN :

Il comporte une liste de valeurs et vérifie si une valeur particulière apparaît sur cette liste.

La syntaxe est : **IN**(val1, val2, ...) et sa négation **NOT IN**(val1, val2, ...).

Lorsque la liste des valeurs est connue et fixe, le prédicat **IN** peut être remplacé par une suite d'opérateurs logiques **OR**. Lorsque la liste des valeurs n'est pas connue à priori, comme dans certaines sous-interrogations, ce prédicat est nécessaire.

- Numéro et nom des articles qui sont soit VERT soit ROUGE

Version IN	Version OR
SELECT Num_art, Nom_art **FROM** ARTICLES **WHERE** art_coul **IN**('ROUGE', 'VERT');	**SELECT** Num_art, Nom_art **FROM** ARTICLES **WHERE** art_coul='ROUGE' **OR** art_coul='VERT';

C : le prédicat LIKE :

Ce prédicat permet de faire des recherches à l'intérieur d'une chaîne de caractères, lorsque l'on dispose d'informations incomplètes. Il utilise 2 caractères génériques :

% ou * : utilisé pour représenter une chaîne de caractère de longueur quelconque.
_ ou ?: utilisé pour représenter un caractère unique.

La syntaxe est : **LIKE** '*Chaîne de recherche*' et sa négation **NOT LIKE** '*Chaîne de recherche*'

Nom et couleur des articles dont la 2° lettre du nom est 'a'.

SELECT Nom_art, art_coul
FROM ARTCILES
WHERE Nom_art **LIKE** '_a%';

D : le prédicat BETWEEN :

Il permet de comparer la valeur d'un champ par rapport à une borne inférieure et une borne supérieure (bornes incluses).

Nom des articles qui ont un prix d'achat compris entre 10 t 25€

SELECT Nomart
FROM ARTCILES
WHERE art_pa **BETWEEN** (10 **AND** 25);

II : RAPPEL SUR LA CLAUSE ORDER BY :

Il est possible, grâce à la clause ORDER BY, d'ordonner les résultats d'une requête en fonction de la valeur de certains attributs (colonnes).

Afficher la liste des articles par couleur et par ordre alphabétique décroissant des noms.

SELECT art_coul, Nom_art
FROM ARTICLES
ORDER BY art_coul, Nom_art **DESC**;

III : JOINTURES :

A : Equi-jointure :

Nous avons précédemment vu l'équi-jointure. Cette opération permet de relier, avec une relation d'égalité, des tables qui ont au moins un attribut commun. On doit avoir n-1 conditions de jointure, n étant le nombre de tables qui interviennent dans la requête.
Si aucune condition de jointure est précisée, la requête correspondante réalisera le *produit cartésien* des tables impliquées.

Nom du fournisseur et libellé de chaque article

SELECT art_nom **AS** Nom article, frs_nom **AS** Nom fournisseur
FROM ARTICLE A, FOURNISSEUR F
WHERE A.art_frs=F.frs_num

B : Auto-jointure :

C'est la jointure d'une table avec elle-même. Cette opération est utile lorsque l'on souhaite relier des attributs qui se trouvent à l'intérieur d'une même table

- Afficher le numéro des étudiants qui ont une note dans la matière numéro 1 inférieure à la note qu'ils ont obtenue dans la matière numéro 2.

SELECT N1.Numetu
FROM NOTES N1, NOTES N2
WHERE N1.Numetu=N2.Numetu
AND N1.Numat=1
AND N2.Numat=2
AND N1.Note<N2.Note;

C : Théta-jointure :

Cette opération, qui doit être utilisée avec précaution, permet de relier deux tables lorsque la colonne de jointure d'une table n'est pas reliée à la colonne de jointure de l'autre table par le signe égal. L'opérateur de comparaison peut être : supérieur(>), inférieur(<), supérieur ou égal(>=), inférieur ou égal(<=), différent(<>).

Afficher pour chaque article, combien se trouvent avant lui sur la liste des articles rangés par ordre alphabétique.

SELECT A1.Art_nom, COUNT(*)
FROM ETUDIANT A1, ETUDIANT A2
WHERE A1.art_nom>A2.art_nom
GROUP BY A1.Art_nom
ORDER BY A1.Art_nom;

Cette requête donne un résultat correct pour tous les articles, sauf pour le premier de la liste alphabétique qui n'apparaît pas. Cette erreur peut être évitée en utilisant, entre autre, une jointure externe.

D : Jointure externe :

Une jointure "classique" entre 2 tables, ou *jointure interne*, ne renvoie des lignes que si la colonne de jointure d'une table est égale à la colonne de jointure de l'autre table. Il peut être utile, dans certaines circonstances, d'afficher toutes les lignes d'une table particulière qu'il y ait ou non correspondance avec l'autre table de jointure. Les colonnes pour lesquelles il n'y a pas de correspondance sont remplies avec la valeur NULL. Cette opération s'appelle une jointure externe.

Il existe 3 types de jointures externes et la façon dont est réalisée une jointure externe dépend de la position des tables par rapport à l'instruction de jointure.

La requête suivante affiche tous les étudiants, qu'ils aient ou non une note. Si un étudiant n'a pas de note, le champ Numetu de la table correspondante (NOTES) aura la valeur NULL.

```
SELECT DISTINCT(E.Nometu), N.Numetu
FROM ETUDIANT E
LEFT OUTER JOIN NOTES N
ON E.Numetu=N.Numetu;
```

La syntaxe, en SQL-92, est la suivante :

a) Jointure externe gauche

SELECT ...

FROM *table1* [*alias*]

LEFT OUTER JOIN *table2* [*alias*]

ON <Conditions de jointure>

La requête suivante affiche tous les étudiants qui ont au moins une note.

```
SELECT DISTINCT(E.Nometu), N.Numetu
FROM ETUDIANT E
RIGHT OUTER JOIN NOTES N
ON E.Numetu=N.Numetu;
```

La syntaxe, en SQL-92, est la suivante :

a) Jointure externe gauche

SELECT ...
FROM *table1* [*alias*]
RIGHT OUTER JOIN *table2* [*alias*]
ON <Conditions de jointure>

IV : Regroupements :

SELECT	Précise les colonnes qui vont apparaître dans la réponse
FROM	Précise la (ou les) table intervenant dans l'interrogation
WHERE	Précise les conditions à appliquer sur les lignes. On peut trouver : - Des comparateurs : =, >, <, >=, <=,<> - Des opérateurs logiques : AND, OR, NOT - Les prédicats : IN, LIKE, NULL, ALL, SOME, ANY, EXISTS...
GROUP BY	Précise la (ou les) colonne de regroupement
HAVING	Précise la (ou les) conditions associées à un regroupement
ORDER BY	Précise l'ordre dans lequel vont apparaître les lignes de la réponse : - ASC : En ordre ascendant (par défaut) - DESC: En ordre descendant

A : Clause GROUP BY

Cette clause permet de créer des sous-ensembles (regroupements) de lignes pour lesquels la valeur d'une (ou plusieurs) colonne est identique.

- Regroupement par couleur d'article

SELECT nom_art
FROM ARTICLES
GROUP BY art_coul;

Cette requête va afficher les articles par couleurs.

B : Clause HAVING

Cette clause, contrairement à la clause WHERE qui précise les conditions à appliquer sur les lignes d'une table, permet de préciser des conditions au niveau des sous-ensembles créés par GROUP BY.

- Regroupement par couleur d'article de type 'VERT'

SELECT art_nom
FROM ARTCILES
GROUP BY art-coul
HAVING art_coul='VERT';

V: Fonctions statistiques

Fonctions	Rôle
COUNT([DISTINCT] *expr* [*alias*]	Renvoie le nombre de valeurs dans la colonne. Si DISTINCT est précisé, les doublons sont éliminés.
COUNT(*)	Compte toutes les lignes de la table.
AVG(*expr* [*alias*])	Renvoie la moyenne des valeurs de la colonne spécifiée.
MIN(*expr* [*alias*])	Renvoie la plus petite valeur de la colonne spécifiée.
MAX(*expr* [*alias*])	Renvoie la plus grande valeur de la colonne spécifiée.
SUM(*expr* [*alias*])	Renvoie la somme des valeurs de la colonne spécifiée.

- Afficher le prix maximum et minimum de la base articles.

SELECT MIN(art_pa) **AS** "Moins chère", **MAX**(art_pa) **AS** "Plus chère"
FROM ARTCILES;

- Afficher le nombre total d'articles

SELECT COUNT(*)
FROM ARTICLES;