SQL

I INTRODUCTION

Le langage SQL (Selected Query Language) sert à 'interroger ' une base de données pour y retrouver des informations selon des critères, en insérer, en modifier….
Il peut être intégrer sans difficultés dans un langage de programmation tel que PHP.

II PREMIERES REQUETES

Les requêtes de base utilisent la commande SELECT. Elle permet de réaliser l'opération la plus courante dans une base de données, à savoir rechercher des informations par une sélection selon différents critères.

• PROJECTION

La projection permet l'affichage des champs sélectionnés.

Syntaxe : Select nom_champs **From** nom_table **;**

nom_champs : champs que l'on veut voir afficher ; l'ordre des champs écrits est l'ordre d'apparition des champs
nom_table **:** table de la base

Le résultat de cette opération est une 'table' résultat de la projection.

Exemple :

Select Nom_Enfant from Enfant ;

Cette requête affiche tous les noms d'enfants de la table.

ENFANT
Cod_ Enfant Nom_enfant Prénom_enfant Datnais Sexe

Empêcher les répétitions de lignes : Pour éviter les répétitions de lignes de même valeur on utilise la clause **Distinct**

Syntaxe : Select Distinct nom_champs **From** nom_table **;**

- **SELECTION DANS UNE TABLE**

La selection permet la 'récupération' d'enregistrements dans une table selon certains critères.

Syntaxe : Select nom_champs **From** nom_table **Where** conditions **;**

Exemple Select Nom_Enfant from Enfant where Sexe='F' ;

Cette requête affiche tous les noms des filles de la table .

❖ **Comparaison à une constante**

Le cas le plus simple consiste à comparer la valeur d'un attribut à une constante. Celle ci doit être compatible avec le type de l'attribut considéré :

- Les chaînes de caractères sont mises entre apostrophes
- Les dates sont mises entre #

❖ **Expression**

On peut comparer la valeur d'un champ à une expression numérique

❖ **Attribut indéterminé**

Dans certains cas certains champs ne sont pas remplis (on ne connaît pas forcement leur valeur) . L'attribut prend alors la valeir NULL

❖ **Opérateur AND**

Pour unir deux conditions par un ET on utilise l'opérateur AND : les deux conditions doivent être satisfaites pour que l'enregistrement soit sélectionné.

Exemple
Select Nom_Enfant from Enfant where Sexe='F' and DatNais=#12/12/1995#;

❖ **Opérateur OR**

Une des deux conditions doit être remplies pour que l'enregistrement soit sélectionné.

Exemple
Select Nom_Enfant from Enfant where Sexe='F' or DatNais=#12/12/1995#;

❖ **Opérateur NOT**

L'opérateur NOT inverse la valeur d'une expression logique.

Exemple Select Nom_Enfant from Enfant where NOT(Sexe='F') ;

- **COMPARAISON A UNE FOURCHETTE DE VALEURS**

L'opérateur BETWEEN permet de sélectionner les tuples dont l'attribut spécifié contient une valeur dans un intervalle déterminé.
L'opérateur NOT BETWEEN permet de sélectionner les tuples dont l'attribut spécifié contient une valeur en dehors de l' intervalle déterminé.

Exemple

Select Nom_Enfant from Enfant where DatNais between #12/12/1995# and #01/12/2000#

- **COMPARAISON A UNE LISTE DE VALEURS**

L'opérateur IN permet de sélectionner les tuples dont l'attribut spécifié contient une valeur apparaissant dans la liste des valeurs suivant l'opérateur

Exemple :

Select Nom_Enfant from Enfant where Prénom_enfant in (« Laurence », « Chloe », « Isabelle », »Daniel » , »marc »);

L'opérateur NOT IN permet de sélectionner les tuples dont l'attribut ne contient pas sa valeur dans la liste

- **COMPARAISON A UN FILTRE**

L'opérateur **LIKE** permet de sélectionner les tuples dont l'attribut spécifié (caractère) contient une valeur correspondant à un filtre donné (possibilité de ne donner qu'une partie du mot).

On montre que le mot est incomplet grâce à :

- % ou * : remplace plusieurs caractères
- _ : remplace un caractére

Exemple :

Select Nom_Enfant from Enfant where Prénom_enfant= »M% » :

- **TRI DES TUPLES**

On peut 'forcer' l'apparition des enregistrements suivant la valeur croissante ou décroissante d'un champ particulier. Pour cela on utilise la clause **ORDER BY** et le critère **ASC (**croissant) ou **DESC (**décroissant).

Syntaxe : Select nom_champrs **From** nom_table **Where** conditions **ORDER BY** nom_champ **critère;**

Par défaut le tri se fait dans l'ordre croissant**.**

III FONCTIONS ET EXPRESSIONS

- **EXPRESSIONS NUMERIQUES**

Il est possible d'incorporer des expressions numériques (+,-,*,/) dans une clause SELECT ou WHERE ou ORDER.

Exemple

Select prixséjour*capacitéaccueil as total from sejour

- **FONCTIONS DE GROUPE**

Définition : Un groupe est un sous ensemble de tuples d'une table , pour lesquels la valeur d'un attribut reste constante

SEJOUR
CODE
Nom centre
Adresse centre
Adville centre
Région centre
Age minimal
Age maximal
Datedébutséjour
Duréeséjour
Prixséjour
Capacitéaccueil
Nombre inscrits

Fonctions :

- **COUNT :** compte le nombre d'occurrences de l'attribut
- **SUM :** calcule la somme des valeurs de l'attribut
- **AVG :** calcule la moyenne des valeurs de l'attribut
- **MAX :** recherche la plus grande valeur de l'attribut
- **MIN :** recherche la plus petite valeur de l'attribut

Exemple
Select count(code) from sejour

La notion de groupe introduit deux nouvelles clauses : group by, having

❖ **La clause group by**

La clause group by réarrange la table résultant du **select** en un nombre minimum de groupes tels que, à l'intérieur de chaque groupe, l'attribut spécifié possède la même valeur de chaque tuple.

Exemple : Calculer le prix de vente moyen de séjour par centre

```
Select  avg(Prixséjour) from séjour
                Group by Nom_centre
                Order by Nom_centre
```

Il est possible d'ajouter une clause **Where** qui introduit un critère de sélection sur les tuples

Exemple : Calculer le prix de vente moyen de séjour par centre pour lesquels la capacité d'accueil

```
Select  avg(Prixséjour) from séjour
                where Capacitéaccueil>*50
                Group by Nom centre
                Order by Nom centre
```

❖ **La clause having**

La clause having est l'équivalent du where pour les groupes. Cette clause ne peut être spécifié que si une clause group by l'a été.
En général, le critère spécifié dans la clause having porte sur la valeur d'une fonction calculée sur un groupe. Les groupes ne répondant pas aux critères spécifiés dans la clause having ne figurent pas dans le résultat.

Exemple : rechercher la couleur des articles dont le prix moyen de vente des articles de la couleur est supérieur à 100

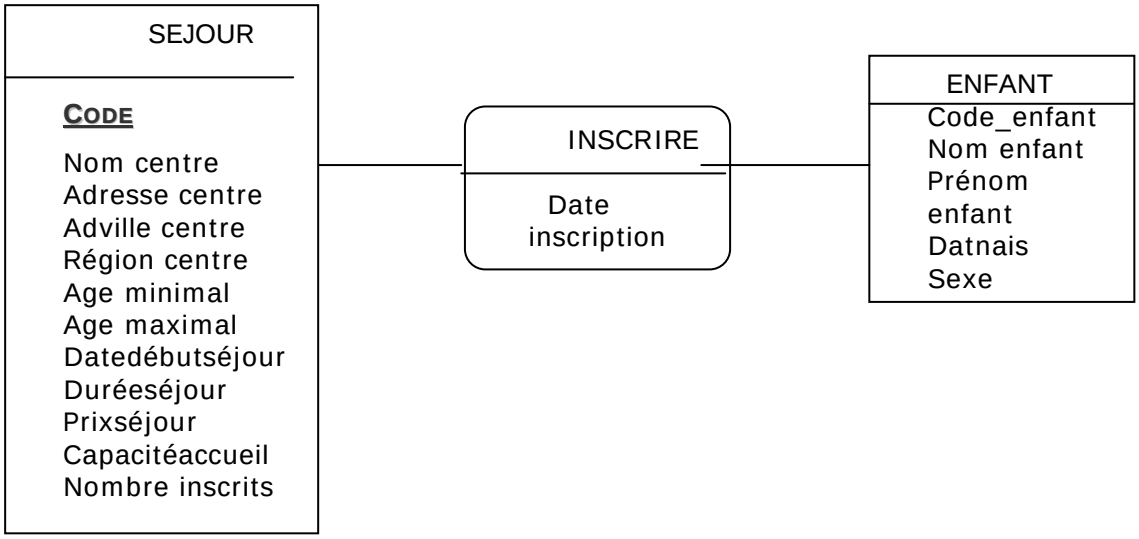
```
Select  avg(Prixséjour) from séjour
                where Capacitéaccueil>*50
                Group by Nom centre
                Having avg(Prixséjour)>500
                Order by Nom centre
```

IV JOINTURE SUR DEUX TABLES

La jointure se fait dans la clause WHERE grâce à l'égalité des champs similaires de deux tables. Lors de ce type de jointure on n'obtient que les tuples vérifiant la jointure.

Syntaxe :
Select nom_champ **from** nom_tables Where
nom_table1.nomchamp1=nom_champ2.nom_table2 **;**

Exemple :



Select nom_enfant **from** sejour,inscrire,enfant Where enfant. Code_enfant = INSCRIRE. Code_enfant and INSCRIRE.code=sejour.code and nom_centre= »ancelles »**;**

IV INSERTION D'ENREGISTREMENT DANS UNE TABLE

On peut insérer des enregistrements dans une table

Syntaxe : Insert Into nomtable[nomchamp] Values (valeur ou variable ou requête)

V MODIFICATION D'ENREGISTREMENT DANS UNE TABLE

Syntaxe :
Update nomtable
Set nomchamp= expression ou valeur
Where condition