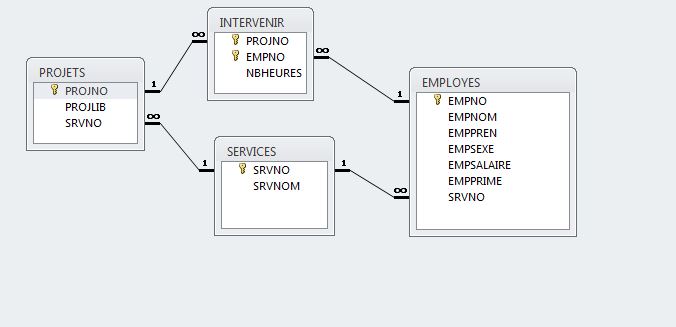
CH2 SQL : LE LANGAGE DE MANIPULATION DES DONNEES

Le langage SQL (Structured Query Language) est un langage informatique normalisé d’interrogation de base de données relationnelles (SGBDR). Il est devenu un standard international en 1986. Il est utilisé par toutes les bases de données (Oracle, dBase, Access...).

C’est un langage qui est proche d’un langage naturel.

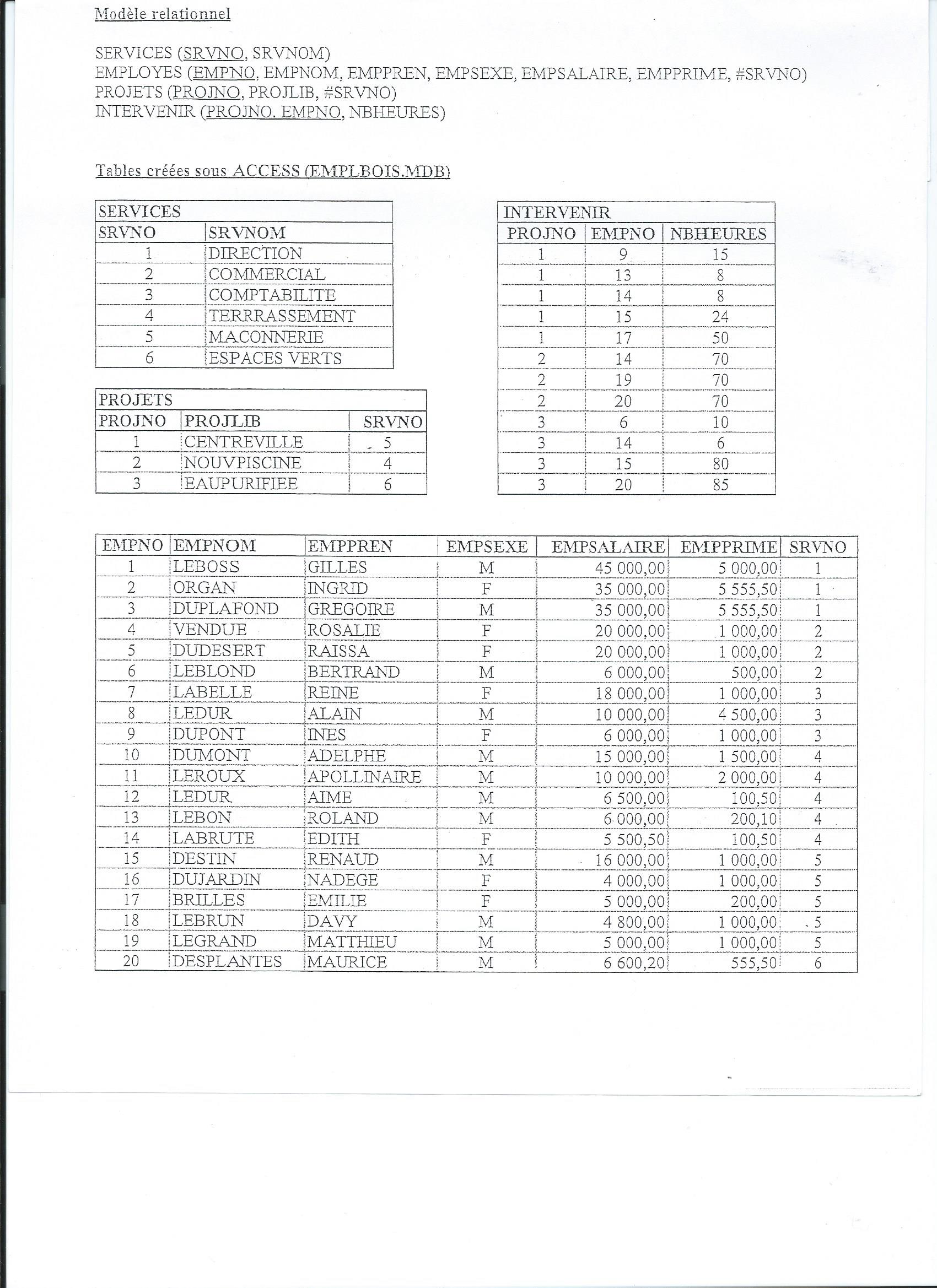
1. **Société exemple**

La société « Emploisbois », est une société prestataire de services, qui propose des projets clés en main. Ci-dessous un extrait de sa base de données.



Description de la base :

* Un employé n’appartient qu’à un seul service.
* Un employé peut intervenir sur plusieurs projets en faisant un nombre d’heures différent pour chaque projet.
* Un projet ne concerne qu’un seul service



1. **Syntaxe des commandes**
   1. Rappel

* Dans une base de données relationnelle les informations sont stockées dans des champs qui sont regroupées dans des tables.
* Les tables regroupent les informations (champs) qui ont un lien commun (Information sur les salariés, sur les services sur les projets …)
* Une requête SQL affiche les contenus de champs qui appartiennent à des tables pour les afficher selon certains critères.
  1. Les instructions avec un ordre à respecter

.





**SELECT**

**FROM**

**WHERE**

**AND (répéter autant de fois que nécessaire)**

**GROUP BY**

**HAVING**

**ORDER BY**

* Il n’y a pas de casse à respecter pour les instructions SQL, la casse doit être respectée pour les noms des tables et propriétés.
* Une instruction sql se finit par un ; (facultatif en mode graphique)
  1. Mémo des principales commandes

Voir annexe en fin de cours.

1. **Etude des commandes**
   1. **Requête d’affichage simple : SELECT … FROM**

L'instruction **SELECT… FROM…** est utilisée pour afficher (Projeter) les enregistrements d’une base de données. **On choisit les colonnes que l’on veut afficher.**

 La commande **SELECT** sélectionne les champs à afficher,

 La commande **FROM** sélectionne les tables dans lesquelles sélectionner les champs.

*Exemple :* ***Afficher les champs : nom et prénom de la table employes***

*SELECT EMPNOM, EMPREN*

*FROM EMPLOYES ;*

**Entrainement**

|  |
| --- |
| Afficher le nom, sexe et salaire de tous les employés |
| SELECT EMPNOM,EMPPREN,EMPSEXE FROM `employes` |
| Que fait la requête suivante : SELECT \* FROM EMPLOYES |
| La requête SQL SELECT \* FROM EMPLOYES sert à récupérer toutes les colonnes (indiquées par \*) et toutes les lignes de la table nommée EMPLOYES dans une base de données. |
| Afficher les noms des projets ainsi que les numéros de service associés au projet |
| SELECT EMPNOM,SRVNO FROM `employes` |
| Afficher les numéros des employés qui sont chargés d’une intervention, trié par ordre croissant. Que remarquez-vous ? |
| SELECT EMPNO FROM `intervenir` ORDER BY EMPNO |
| Faites la même requête que précédemment en utilisant SELECT DISTINCT au lieu de SELECT. Que remarquez-vous ? |
| S[ELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) DISTINCT EMPNO FROM `intervenir` ORDER BY EMPNO  SELECT DISTINCT est utilisé pour éliminer les doublons |
| Afficher le nom prénom, salaire, prime des employés |
| [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) EMPNO,EMPPREN,EMPSALAIRE,EMPPRIME FROM `employes` |
| Faites la requête suivante et expliquer:  SELECT empnom, empsalaire, empprime, (empsalaire + empprime) AS salaireTotal FROM EMPLOYES ; |
| Sa rajoute un tableau salaireTotal qui contient la somme (empsalaire+empprime) des employées |
| ETABLIR VOTRE PROPRE REQUETE |
| [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) EMPNO,EMPPREN,EMPSALAIRE,EMPPRIME, (EMPSALAIRE-EMPPRIME) AS SALAIRETOTAL FROM `employes`; |

* 1. Requête d’affichage avec restriction : WHERE

La commande **WHERE** permet de filtrer les enregistrements. La syntaxe de la commande est la suivante : WHERE « champs » **opérateur critère de filtre. La commande permet de sélectionner des lignes dans la table.**

|  |  |  |
| --- | --- | --- |
| **Opérateurs disponibles** | |  |
|  | |  |  |  | | --- | --- | --- | | **Opérateurs** | **Effets** | **Exemples** | | = | égal | WHERE salaire = 2000 | | < | inférieur à | WHERE salaire < 2000 | | > | supérieur à | WHERE salaire > 2000 | | <= | inférieur ou égal à | WHERE salaire <= 2000 | | >= | supérieur ou égal à | WHERE salaire >= 2000 | | <> | différent de | WHERE salaire <> 2000 | | NOT NULL | non vide | WHERE salaire IS NOT NULL | | NULL | vide | WHERE salaire IS NULL | | LIKE (voir ci-dessous) | contient | WHERE Prenom LIKE « L% » | | IN | Choix parmi des valeurs | WHERE prenom IN(« Ursule », « Jules », “toto”) | | NOT IN | Tous SAUF | WHERE prenom NOT IN(« Ursule », « Jules », “toto”) | | BETWEEN | entre | WHERE salaire BETWEEN 200 AND 2000 | | |

La commande LIKE peut être associée à 2 symboles :

* au symbole **%** qui remplace une chaîne de caractère. Dans ce cas elle affiche uniquement les enregistrements qui contiennent le texte situé avant, après ou entre les signe % (chaine de caractères)

*Exemple 1 :* ***Afficher les villes qui commencent par la lettre : L*** *=> WHERE Ville\_sal LIKE “L%” Exemple 2 :* ***Afficher les villes dont le nom se termine par : ian*** *=> WHERE Ville\_sal LIKE '%ian Exemple 3 :* ***Afficher les villes dont le nom contient les lettres : Du*** *=> WHERE Ville\_sal LIKE '%Du%'*

* *au symbole \_ (barre du 8 sur le clavier) qui remplace 1 caractère.*

*Exemple : afficher les villes qui ont un E en 2ième position => WHERE ville LIKE «\_E% »*

* selon les SGBD, les symboles peuvent être différents : \* et ?

*Exemple :* ***Afficher les champs : nom et prénom des employées (les femmes) de la table employes***

*SELECT EMPNOM, EMPREN*

*FROM EMPLOYES*

*WHERE empsexe = “F”;*

A NOTER :

* pour les propriétés de type chaines de caractères, les “ “ sont obligatoires.
* pour les propriétés de type numérique : pas de «  »
* Quand il y a plusieurs conditions, on les combine selon le cas avec des AND ou des OR
  + - L’opérateur **AND** permet d’associer des critères qui s’ajoutent. (Les deux critères doivent être remplis).
    - L’opérateur **OR** permet d’associer des critères qui s’éliminent. (Un des deux critères doit être rempli).

**Entrainement**

|  |
| --- |
| Afficher la liste des employées qui ont une prime < 1000 € |
| SELECT EMPNOM,EMPPREN,EMPPRIME FROM `employes` WHERE EMPPRIME<1000 |
| Afficher le nom et prénom des employés qui ont une prime comprise entre 500 et 1000 € |
| SELECT EMPNOM,EMPPREN,EMPPRIME FROM `employes` WHERE 500<EMPPRIME<1000 |
| Afficher le nom, prénom, le salaire et la prime des employés qui ont une prime supérieure à 3000€ ou un salaire inférieur à 7000€ |
| SELECT EMPNOM,EMPPREN,EMPPRIME,EMPSALAIRE FROM `employes` WHERE EMPPRIME<3000 OR EMPSALAIRE<7000 |
| Afficher les noms des employés qui appartiennent au service n° 4 ou 5 ou 6 |
| SELECT EMPNOM,EMPPREN FROM `employes` WHERE SRVNO=4 OR SRVNO=5 or SRVNO=6 |
| Afficher les employés dont le nom commence par D |
| SELECT EMPNOM,EMPPREN FROM `employes` WHERE EMPNOM LIKE "D%" |
| Afficher les employés dont le nom finit par T |
| SELECT EMPNOM,EMPPREN FROM `employes` WHERE EMPNOM LIKE "%T" |
| Afficher les employés dont le nom a un B en 3ième position |
| [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) EMPNOM,EMPPREN FROM `employes` WHERE EMPNOM [LIKE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/string-comparison-functions.html%23operator_like) "\_\_B%"; |
| Afficher les employés dont le nom ne comporte que 5 lettres |
| [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) EMPNOM,EMPPREN FROM `employes` WHERE EMPNOM [LIKE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/string-comparison-functions.html%23operator_like) "\_\_\_\_\_"; |
| Afficher les employés dont la rémunération totale est >= à 50000 € |
|  |
| Afficher la liste des employés qui ne travaillent pas dans le service n° 3 |
|  |
| CREER VOTRE PROPRE REQUETE |
|  |

* 1. Trier les données  **ORDER BY**

La commande ORDER BY paramètre le nom du champ sur lequel trier les données ainsi que le critère de tri : ASC (croissant) ou DESC (décroissant)

*Exemple :* ***Afficher les champs numero, nom et prenom de la table employes en les triant sur le nom trié par ordre croissant***

*SELECT empno, empnom, emppren*

*FROM employes*

*ORDER BY Nom\_sal ASC*

* Il est possible d’indiquer plusieurs clés de tri en les saisissant les un à la suite des autres

*Exemple :* ***Afficher les champs : numero, nom et prénom de la table employés triés sur le numéro par ordre croissant et sur le nom par ordre décroissant***

*SELECT empno, empnom, emppren*

*FROM employes*

*ORDER BY empno ASC, empnom DESC*

* 1. Les fonctions arithmétiques

Elles permettent d’afficher des statistiques sur des champs numériques ou monétaires ou de compter le nombre d’occurrences d’un champ. La fonction est placée devant le nom du champ qui est mis entre parenthèse.

Le résultat attendu n’affiche qu’un chiffre.

Pour donner un nom au champ crée, on ajoute l’instruction AS

E*xemple 1 :* ***Afficher le nombre de salariés***

*SELECT COUNT(\*)AS nombre\_salariés FROM employes;*

*Exemple 2 :* ***Afficher le salaire moyen des employés***

*SELECT AVG(empsalaire) AS sal\_moyen FROM employes ;*

A NOTER :

* COUNT() compte le nombre de lignes répondant au critère.
* Sauf COUNT, les autres fonctions attendent un champ de type numérique.
* Il existe beaucoup de fonctions. Les plus utiles sont ci-dessous.

|  |  |
| --- | --- |
| **Fonctions** | **Effets** |
| MAX(argument) | Affiche la valeur maximum d’un champ |
| MIN(argument) | Affiche la valeur minimum d’un champ |
| AVG(argument) | Affiche la moyenne des données d’un champ |
| SUM(argument) | Affiche la somme des données d’un champ |
| COUNT(\*) | Affiche le nombre de valeur d’un champ |

L’argument représente le champ sur lequel porte le calcul. C’est donc forcément un champ de type numérique

**Entrainement**

|  |
| --- |
| Afficher le salaire le plus élevé des employés |
|  |
| Afficher l’écart entre le salaire le plus élevé et le salaire le plus bas |
|  |
| Afficher le nombre d’employés du service n° 4 |
|  |
| Afficher le montant total des salaires que doit verser l’entreprise |
|  |
| Afficher le nombre d’employés qui sont des hommes et qui travaillent au service n° 1 |
|  |
| ETABLIR VOTRE PROPRE REQUETE |
|  |

* 1. Le regroupement des données : GROUP BY

La commande GROUPE BY regroupe à l’affichage les données sur le champ spécifié.

Dans le SELECT, il ne faut mettre que le champ portant sur le calcul et celui du regroupement.

Exemple : afficher le nombre de salariés par sexe

SELECT empsexe, COUNT(\*) AS nombre\_salariés

FROM employes

GROUP BY empsexe

**Entrainement**

|  |
| --- |
| Afficher le salaire le plus élevé des employés en fonction du service |
|  |
| Afficher le nombre d’heures total par numéro de projet |
|  |
| Afficher le nombre d’employés par service |
|  |
| Afficher le nombre d’employés par service (pour les services n° 4, 5 et 6). 2 solutions dont 1 avec la clause having |
|  |

La clause HAVING

Une clause HAVING est similaire à une clause WHERE si ce n'est qu'elle s'applique uniquement aux groupes dans leur ensemble (c'est-à-dire aux lignes du jeu de résultats qui représentent des groupes), tandis que la clause WHERE s'applique aux lignes individuelles.

Exemple : afficher le total des salaires par service, pour les services dont le total est >= 50000 €

SELECT empno, SUM(empsalaire) AS total

FROM employes

GROUP BY empsexe

HAVING SUM(empsalaire) >= 50000;

* 1. Les jointures entre les tables

Les jointures en SQL permettent d’associer plusieurs tables dans une même requête.

Les règles :

* 2 tables peuvent être liées si et seulement si elles possèdent une propriété commune.
* La propriété commune est à chercher entre les clés primaires et les clés étrangères.
* On ne peut lier que 2 tables à la fois. (pour lier 3 tables, on fera 2 jointures).

Méthodologie :

1. On recherche dans quelles tables se trouvent les propriétés dont on a besoin.
2. On vérifie que les tables dont on a besoin ont bien une propriété commune. Dans la négative, on ajoutera une table supplémentaire.
3. Obligatoirement dans l’instruction sql, on ajoute les jointures.

2 syntaxes :

1ère méthode : dans le from, on met le nom des tables et dans le where les liaisons

SELECT nom des champs à afficher

FROM tableA, tableB

WHERE tableA.propriétécommune = tableB.propriétécommune

AND ……

2ième méthode : on met tout dans le from avec l’instruction INNER JOIN

SELECT nom des champs à afficher

FROM tableA INNER JOIN tableB ON tableA.propriétécommune = tableB.propriétécommune

WHERE ….

Exemple : afficher le nom et prénom des employés ainsi que le nom du service auquel ils appartiennent.

Taper la requête ci-dessous et regarder le résultat. Est-il cohérent ? (justifier):

select empnom, emppren, srvnom from employes, services

bonne syntaxe :

select empnom, emppren, srvnom

from employes, services

WHERE employes.srvno = services.srvno;

Ou

select empnom, emppren, srvnom

from employes INNER JOIN services ON employes.srvno = services.srvno;

**Entrainement**

|  |
| --- |
| Afficher le nombre d’heures effectuées dans un projet par l’employé DUPONT INES |
|  |
| Afficher le nombre d’heures effectuées dans un projet en précisant le nom du projet par l’employé Desplantes Maurice |
|  |
| Afficher le nombre d’employés par service en indiquant le nom du service |
|  |
| Afficher le nom des employés ainsi que le nom du service et celui des projets pour les employés masculins qui ont effectués plus de 20 heures par intervention. |
|  |
| Créer votre propre requête |
|  |

ANNEXE :

