

. : Exercice – la base exemple : .

Développer une application Client-Serveur – Développer des composants d'accès
Formaliser des requêtes SQL



Jointures

La bonne syntaxe pour écrire des jointures est la suivante :

```
SELECT *  
FROM table1  
JOIN table2 ON condition_de_jointure_table1_table2  
WHERE condition_de_restriction
```

Il existe une ancienne syntaxe sous cette forme :

```
SELECT *  
FROM table1, table2  
WHERE condition_jointure_table1_table2 AND condition_de_restriction
```

Dans la table EMP la colonne `nodept` contient des valeurs qui se trouvent dans la colonne `nodept` de la table DEPT et inversement.

Logiquement chaque ligne de EMP *correspond* à une ligne de DEPT et, chaque ligne DEPT *correspond* à une ou plusieurs lignes de EMP.

Cette correspondance se fait par l'opérateur égalité = entre les valeurs des deux colonnes des deux tables : c'est une équi-jointure.

Rechercher le prénom des employés et le numéro de la région de leur département.

Derrière SELECT il peut y avoir nécessité de préfixer la colonne par le nom de la table. En effet si la colonne apparaît dans plusieurs tables il y a ambiguïté.

Pour faciliter l'écriture des requêtes, les tables citées derrière FROM peuvent être renommées temporairement, et l'alias peut être utilisé pour préfixer les colonnes.

Rechercher le numéro du département, le nom du département, le nom des employés classés par numéro de département (renommer les tables utilisées).

Rechercher le nom des employés du département Distribution.

. : Exercice – la base exemple : .

Développer une application Client-Serveur – Développer des composants d'accès
Formaliser des requêtes SQL



Auto-jointures

La possibilité de renommer temporairement une table dans une requête permet de faire la jointure d'une table sur elle-même, c'est à dire l'auto-jointure.

Rechercher le nom et le salaire des employés qui gagnent plus que leur patron, et le nom et le salaire de leur patron.

. : Exercice – la base exemple : .

Développer une application Client-Serveur – Développer des composants d'accès
Formaliser des requêtes SQL



Sous-requêtes

Le résultat d'une requête peut servir dans une clause de restriction d'une autre requête, on parlera alors de sous-requête imbriquée.

```
SELECT *  
FROM emp  
WHERE nodep IN  
    (SELECT nodept FROM dept WHERE nom='...');
```

La recherche suivante peut se faire de deux manières.

Rechercher le nom et le titre des employés qui ont le même titre que Amartakaldire.

Dans ce qui précède, la sous-requête retourne une seule valeur, l'opérateur égalité peut être utilisé. Si ce n'est pas le cas il faut utiliser les clauses ANY ou ALL.

Rechercher le nom, le salaire et le numéro de département des employés qui gagnent plus qu'un seul employé du département 31, classés par numéro de département et salaire.

Rechercher le nom, le salaire et le numéro de département des employés qui gagnent plus que tous les employés du département 31, classés par numéro de département et salaire.

En fait : « IN » est équivalent à « = ANY », tandis que « NOT IN » est équivalent à « != ALL ».

Rechercher le nom et le titre des employés du service 31 qui ont un titre que l'on trouve dans le département 32.

Rechercher le nom et le titre des employés du service 31 qui ont un titre l'on ne trouve pas dans le département 32.

Rechercher le nom, le titre et le salaire des employés qui ont le même titre et le même salaire que Fairant.

. : Exercice – la base exemple : .

Développer une application Client-Serveur – Développer des composants d'accès
Formaliser des requêtes SQL



Requêtes externes

LEFT JOIN, RIGHT JOIN

Dans la table `DEPT` il y a des lignes avec un numéro de département qui ne *correspondent* à aucune ligne de `EMP`.

Cette ligne est à l'extérieur de la jointure entre les deux tables.

Si on souhaite, malgré tout, obtenir dans le résultat de la jointure ces lignes *extra* on utilise un `LEFT JOIN`.

Pour bien comprendre le sens et la syntaxe de `LEFT JOIN` il faut utiliser la syntaxe originale des jointures en SQL.

Pour écrire un `LEFT JOIN`, la syntaxe est la même, en remplaçant bien entendu `JOIN` par `LEFT JOIN`.

Qu'est-ce que cela va changer ? On se souvient que pour une jointure *normale* on ne prend que les enregistrements de chaque table qu'on peut relier par la condition de jointure. Avec un `LEFT JOIN` on prendra en plus les enregistrements de la table écrite à gauche (car `LEFT`) de l'expression `LEFT JOIN` et qui ne sont reliés à aucun enregistrement de celle de droite.

Il existe également `RIGHT JOIN` qui fonctionne de manière tout à fait symétrique.

Rechercher le numéro de département, le nom du département, le nom des employés, en affichant aussi les départements dans lesquels il n'y a personne, classés par numéro de département.

. : Exercice – la base exemple : .

Développer une application Client-Serveur – Développer des composants d'accès
Formaliser des requêtes SQL



Utilisation de fonctions de groupe

Par exemple : AVG (moyenne), MIN (minimum), MAX (maximum), SUM (somme), COUNT (dénombrement)...

Ces fonctions *travaillent* au niveau de groupe de lignes et non plus au niveau des lignes.

Exemple . Moyenne

Par exemple pour rechercher la moyenne des salaires des secrétaires :

```
SELECT AVG(salaire)
FROM emp
WHERE titre = 'Secrétaire'
;
```

Avec SELECT on ne peut pas *travailler* à la fois au niveau des lignes et des groupes.

Si vous recherchez le nom et la moyenne des salaires des employés (cette phrase a-t-elle d'ailleurs un sens ?), vous allez essayer :

```
SELECT nom, AVG(salaire)
FROM emp
;
```

Ce qui ne produira qu'un message d'erreur.

Par contre avec deux SELECT imbriqués on peut rechercher le nom et le salaire des employés dont le salaire est le plus grand.

```
SELECT nom, salaire
FROM emp
WHERE salaire = (SELECT MAX (salaire)
                FROM emp
                )
;
```

. : Exercice – la base exemple : .

Développer une application Client-Serveur – Développer des composants d'accès
Formaliser des requêtes SQL



Les groupes

Pour exprimer le groupe sur lequel doit porter la fonction de groupe on utilise la clause `GROUP BY`.

Ces fonctions et clauses peuvent s'utiliser avec une jointure.

Toute colonne qui intervient dans l'affichage sans être utilisée dans une fonction de groupe doit être aussi incluse dans la clause `GROUP BY`.

Pour rechercher la moyenne des salaires de chaque département on écrira :

```
SELECT nodept, AVG(salaire)
FROM emp
GROUP BY nodept
;
```

- 1. Calculer le nombre d'employés de chaque titre.**
- 2. Calculer la moyenne des salaires et leur somme, par région.**

. : Exercice – la base exemple : .

Développer une application Client-Serveur – Développer des composants d'accès
Formaliser des requêtes SQL



La clause HAVING

La clause `WHERE` permet d'écrire une restriction au niveau ligne, la clause `HAVING` permet d'écrire une restriction au niveau groupe.

Pour rechercher les titres et le nombre d'employés pour les titres représentés plus de 2 fois, on écrira :

```
SELECT titre, COUNT(*)  
  
FROM emp  
  
GROUP BY titre  
  
HAVING COUNT(*) > 2  
  
;
```

- 3. Afficher les numéros des départements ayant au moins 3 employés.**
- 4. Afficher les lettres qui sont l'initiale d'au moins trois employés.**
- 5. Rechercher le salaire maximum et le salaire minimum parmi tous les salariés et l'écart entre les deux.**
- 6. Rechercher le nombre de titres différents.**
- 7. Pour chaque titre, compter le nombre d'employés possédant ce titre.**
- 8. Pour chaque nom de département, afficher le nom du département et le nombre d'employés.**
- 9. Rechercher les titres et la moyenne des salaires par titre dont la moyenne est supérieure à la moyenne des salaires des Représentants.**
- 10. Rechercher le nombre de salaires renseignés et le nombre de taux de commission renseignés.**