Secteur Tertiaire Informatique

Filière « Etude et développement »

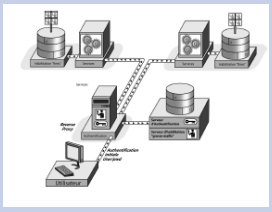
TP – Initiation requêtage SQL

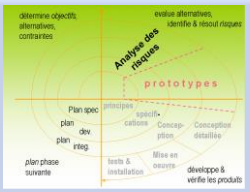
BDD – requêtage SQL

Apprentissage

Mise en situation

Evaluation





Objectif

A l'issue de cette activité, vous serez capable de requêter une base de données en utilisant le langage SQL (« Structured Query Language »), en particulier les opérations de manipulation des données (« DML » pour « Date Manipulation Language »).

Objectifs pédagogiques :

- Comprendre la structure simple d'une base de données
- Effectuer des requête « SELECT »
- Manipuler les paramètres des clauses « SELECT »
- Utiliser « GROUP BY » pour effectuer des regroupements d'enregistrement

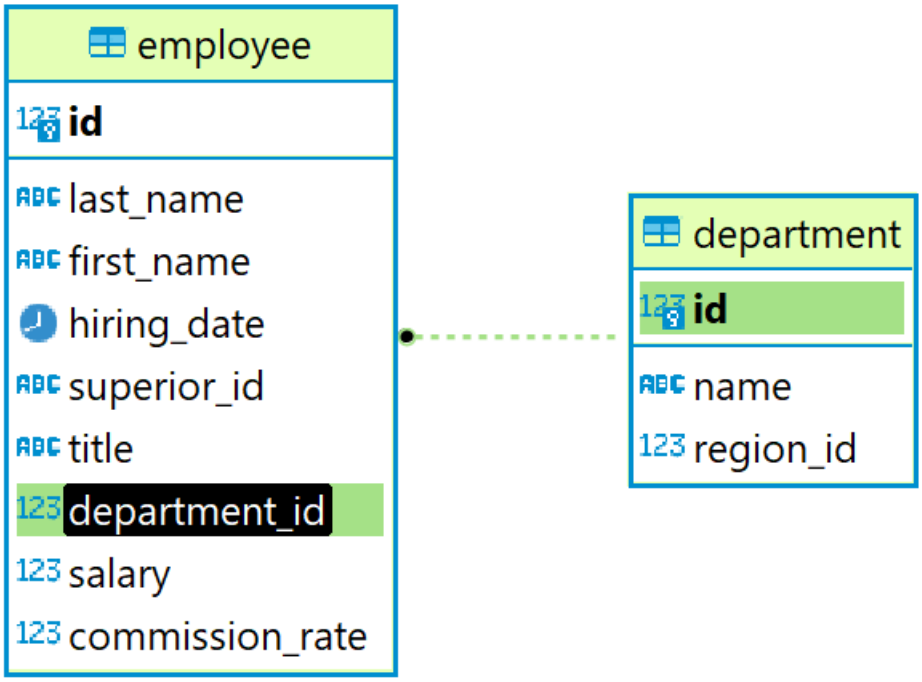
1. STRUCTURE DE LA BASE DE DONNEES

Cette base représente d'une manière simpliste (et remarquablement incomplète) le personnel d'une entreprise. Elle n'est pas destinée à être réaliste, mais seulement à être simple à comprendre pour une première approche de SQL.

Elle est composée de deux tables :

- La table « `employee` » contient les informations sur les employés
- La table « `department` » recense les différents départements de l'entreprise.

Ci-dessous un schéma « entité-relation » présentant la structure de la base de données :



1.1 DICTIONNAIRE DE DONNEES

Ci-dessous les détails concernant les différents champs des tables.

1.1.1 Table « employee »

`id` :

L'identifiant de la personne. Chaque employé a un « id », tous les « id » sont différents et uniques. Il s'agit d'une **clef primaire** (« primary key »).

`last_name` :

Le nom de la personne.

`first_name` :

Le prénom de la personne.

`hiring_date` :

Sa date d'embauche.

`superior_id` :

L'identifiant de son supérieur hiérarchique.

Par exemple, Sophie Fonfec (dont l'identifiant est 17) a 6 pour « superior_id », ce qui signifie que l'identifiant de son supérieur hiérarchique est 6.

`title` :

La fonction de l'employé (directeur, secrétaire, représentant, etc.).

`department_id` :

Le numéro du département (dans l'entreprise) dont dépend l'employé. Ce numéro fait référence à la clef primaire de la table « department ». C'est une **clef étrangère** (foreign key).

`salary` :

Salaire mensuel net de l'employé.

`commission_rate` :

Taux de commission, exprimé en pourcentage.

1.1.2 Table « department »

`id` :

L'identifiant du département. Chaque département a un identifiant, tous les identifiants sont différents. C'est une clef primaire (primary key).

name :

Le nom du département. On peut remarquer que des départements différents peuvent avoir le même nom.

`region_id` :

Référence à une région. Ce numéro fait référence à la clef primaire d'une table qui ne sera pas décrite ici. C'est une **clef étrangère** (foreign key).

2. JOINTURE

Une jointure permet de lier les données des tables entre elles.

La syntaxe de base est :

```
SELECT colonnes d'une ou plusieurs tables séparées par « , »
FROM table1
JOIN table2 ON condition_de_jointure
WHERE conditions logiques séparées par « AND » ou « OR »
```

Dans la base de données fournie la table « employee » a une colonne « department_id » faisant référence à des identifiants de la table « department ».

Ainsi, chaque ligne de « employee » correspond à une ligne de « department » et chaque ligne de « department » correspond à une ou plusieurs lignes de « employee ».

Pour joindre deux tables, il faut faire une égalité entre les valeurs des colonnes contenant ces informations.

Ainsi, pour récupérer toutes les informations concernant les employés (y compris les informations concernant le département) il est possible d'utiliser la requête suivante :

```
SELECT *
FROM employee e
JOIN department d
ON e.department_id = d.id
```

Les caractères 'e' et 'd' sont des alias utilisés pour nommer les tables utilisés dans la requête. Ceci permet de gagner en concision de requête.

Ils sont à utiliser pour faire référence aux colonnes des tables qu'ils concernent.

1. **Affichage de tous les employés et leur département (requête ci-dessus).**

Dans la suite de ce TP vous devrez écrire un ensemble de requêtes utilisant les jointures.

3. REQUETES A ECRIRE

3.1 SELECT ET JOINTURE

Dans cet exemple de requête est utilisé '*' pour retourner toutes les colonnes dans le résultat. Il est possible de plutôt utiliser une liste de colonnes.

2. **Rechercher le numéro du département, le nom du département, le nom des employés classés par numéro de département (utilisez des alias pour les tables).**
3. **Rechercher le nom des employés du département « Distribution ».**

3.2 AUTO-JOINTURES

Une table peut être jointe avec elle-même pour comparer les lignes qui la compose. Il vous faudra, pour se faire, mettre en place une « auto-jointure » en utilisant bien les alias des tables.

Par exemple, pour joindre la table « employee » avec elle-même vous pouvez utiliser :

```
SELECT *
FROM employee e1
JOIN employee e2
ON e1.id = e2.superior_id
```

Sachant ceci, écrivez la requête permettant de répondre à la demande suivante :

4. **Rechercher le nom et le salaire des employés qui gagnent plus que leur supérieur hiérarchique, et le nom et le salaire du supérieur.**

3.3 SOUS-REQUETE

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 employee
WHERE department_id
IN (SELECT id FROM department WHERE name LIKE '...');
```

5. **Rechercher les employés du département « finance » en utilisant une sous-requête.**
6. **Rechercher le nom et le titre des employés qui ont le même titre que « Amartakaldire »**

Pour cette dernière requête (6), la sous-requête utilisée retourne qu'une seule valeur. Il est donc possible d'utiliser l'opérateur d'égalité.

Dans le cas où la sous requête retournerait plusieurs valeurs vous pourrez utiliser les clauses « IN » ou « ALL » ou « ANY ».

Sachant ceci, vous pourrez continuer l'écriture des requêtes.

7. **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.**
8. **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.**

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

Si vous souhaitez obtenir l'inverse de « **IN** » vous pourrez utiliser « **NOT IN** ».

10. **Rechercher le nom et le titre des employés du service 31 qui ont un titre que l'on ne trouve pas dans le département 32.**
11. **Rechercher le nom, le titre et le salaire des employés qui ont le même titre et le même salaire que « Fairant ».**

3.4 REQUETES EXTERNES

Cette partie permet de travailler les « LEFT JOIN » et « RIGHT JOIN ».

Contexte :

Dans la table « department » il y a des lignes avec un numéro de département qui ne correspondent à aucune ligne de la table « employee ».

On dira que ces lignes sont 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 écrire un LEFT JOIN, la syntaxe est la même, en remplaçant « JOIN » par « LEFT JOIN ».

Qu'est-ce que ça 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.

12. **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.**

3.5 UTILISATION DE FONCTIONS D'AGREGATION (OU DE GROUPE)

Vous allez vous servir de fonctions d'agrégation permettant d'effectuer des calculs : AVG (moyenne), MIN (minimum), MAX (maximum), SUM (somme), COUNT (dénombrement).

Ces fonctions agiront au niveau de **groupes de lignes** et non plus au niveau des lignes.

Exemple : calcul d'une moyenne de salaires

13. Calculer la moyenne des salaires des secrétaires (requête fournie ci-dessous)

```
SELECT AVG(salary)
FROM employee
WHERE title LIKE '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 name, AVG(salary)
FROM employee;
```

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

```
SELECT name, salary
FROM employee
WHERE salary = (SELECT MAX(salary) FROM employee);
```

3.6 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 department_id, AVG(salary)
FROM employee
GROUP BY department_id;
```

14. Calculer le nombre d'employé de chaque titre.

15. **Calculer la moyenne des salaires et leur somme, par région.**

3.7 LA CLAUSE « HAVING »

La clause « WHERE » permet d'écrire une restriction au niveau des lignes, la clause « HAVING » permet d'écrire une restriction au niveau des groupes.

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

```
SELECT title, COUNT(*)
FROM employee
GROUP BY title
HAVING count(*) > 2;
```

16. **Afficher les numéros des départements ayant au moins 3 employés.**
17. **Afficher les lettres qui sont l'initiale d'au moins trois employés.**
18. **Rechercher le salaire maximum et le salaire minimum parmi tous les salariés et l'écart entre les deux.**
19. **Rechercher le nombre de titres différents.**
20. **Pour chaque titre, compter le nombre d'employés possédant ce titre.**
21. **Pour chaque nom de département, afficher le nom du département et le nombre d'employés.**
22. **Rechercher les titres et la moyenne des salaires par titre dont la moyenne est supérieure à la moyenne des salaires des « Représentant ».**
23. **Rechercher le nombre de salaires renseignés et le nombre de taux de commission renseignés.**

CREDITS

ŒUVRE COLLECTIVE DE l'AFPA
Sous le pilotage de la DIIP et du centre d'ingénierie sectoriel Tertiaire-Services

Equipe de conception (IF, formateur, mediatiseur)
Michel Coulard – Formateur Evry
Chantal Perrachon – IF Neuilly sur Marne

Date de mise à jour : 02/04/2024