

# Explications de requêtes SQL-DML

## schéma "scott francisé"

### Sommaire:

- 1 INTERROGER SIMPLEMENT UNE BASE
- 2 CLASSER LE RESULTAT D'UNE INTERROGATION
- 3 LES JOINTURES
- 4 LES SOUS-INTERROGATIONS
- 5 EXPRESSIONS ET FONCTIONS SIMPLES
- 6 LES FONCTIONS DE GROUPE
- 7 AJOUT DE LIGNES
- 8 MODIFICATION DE LIGNES
- 9 SUPPRESSION DE LIGNES
- 10 GESTION DES TRANSACTIONS

Dans les tableaux suivants et dans la colonne de droite, vous avez l'explication de la requête.

### 1 Interroger simplement une base

SELECT nom, fonction FROM emp;	Quels sont le nom et la fonction de chaque employé ?
SELECT DISTINCT fonction FROM emp;	Quelles sont toutes les fonctions différentes ?
SELECT nom, salaire, comm FROM emp WHERE comm > salaire;	Quels sont les employés dont la commission est supérieure au salaire?
SELECT nom, salaire FROM emp WHERE salaire BETWEEN 20000 AND 25000;	Quels sont les employés gagnant entre 20000 et 25000?
SELECT num, nom, fonction, salaire FROM emp WHERE fonction IN ('commercial','ingenieur');	Quels sont les employés commerciaux ou ingénieurs?
SELECT nom FROM emp WHERE nom LIKE 'M%';	Quels sont les employés dont le nom commence par M?
SELECT nom FROM emp WHERE n_dept = 30 AND salaire > 25000;	Quels sont les employés du département 30 ayant un salaire supérieur à 25000?
SELECT nom, fonction, salaire, n_dept FROM emp WHERE fonction = 'directeur' OR (fonction = 'commercial' AND n_dept = 10);	Quels sont les employés directeurs, ou commerciaux et travaillant dans le département 10?
SELECT nom FROM emp WHERE comm IS NULL;	Quels sont les employés dont la commission a la valeur NULL?
SELECT nom, salaire "SALAIRE MENSUEL" FROM emp;	Salaire de chaque employé

## 2 Classer le résultat d'une interrogation

<pre>SELECT nom, fonction, salaire FROM emp ORDER BY fonction, salaire DESC;</pre>	Donner tous les employés classés par fonction, et pour chaque fonction classés par salaire décroissant
--	--

## 3 Les jointures

La jointure est une opération permettant de combiner des informations venant de plusieurs tables.

<pre>SELECT emp.nom, lieu FROM emp, dept WHERE emp.n_dept = dept.n_dept;</pre>	Donner pour chaque employé son nom et son lieu de travail.
<pre>SELECT emp.nom, mgr.nom FROM emp, emp mgr WHERE emp.n_sup= mgr.num;</pre>	Donner pour chaque employé le nom de son supérieur hiérarchique.
<pre>SELECT emp.nom, emp.salaire, emp.fonction FROM emp, emp j WHERE emp.salaire &gt; j.salaire AND J.nom = 'SIMON';</pre>	Quels sont les employés gagnant plus que SIMON?
<pre>SELECT emp.nom, lieu FROM emp, dept WHERE emp.n_dept(+) = dept.n_dept;</pre>	Le département 40 ne figurait pas dans le résultat du select précédent. Par contre, il figurera dans le résultat du Select suivant
<pre>SELECT dept.n_dept, emp.nom FROM emp,dept WHERE dept.n_dept = emp.n_dept (+) AND emp.nom IS NULL;</pre>	Retrouver les départements n'ayant aucun employé.

## 4 Les sous-interrogations

### Requête imbriquée

Une caractéristique puissante de SQL est la possibilité qu'un critère de recherche employé dans une clause WHERE (expression à droite d'un opérateur de comparaison) soit lui-même le résultat d'un SELECT; c'est ce qu'on appelle une sous-interrogation.

<pre>SELECT nom FROM emp WHERE fonction = (SELECT fonction                   FROM emp                   WHERE nom = 'Codd');</pre>	Quels sont les employés ayant la même fonction que Codd?
<pre>SELECT nom, salaire FROM emp WHERE salaire &gt; ALL (SELECT salaire                     FROM emp                     WHERE n_dept = 20);</pre>	Quels sont les employés gagnant plus que tous les employés du département 30.
<pre>SELECT nom, fonction, n_sup FROM emp WHERE (fonction, n_sup) = (SELECT fonction, n_sup                           FROM emp                           WHERE nom = 'Codd');</pre>	Quels sont les employés ayant même fonction et même supérieur que Codd?
<pre>SELECT nom FROM emp e WHERE n_dept != (SELECT n_dept</pre>	Quels sont les employés ne travaillant pas dans le même département que leur supérieur

<pre> FROM emp WHERE e.n_sup = num) AND n_sup IS NOT NULL; </pre>	hiérarchique.
<pre> SELECT * FROM emp e WHERE EXISTS (SELECT * FROM emp WHERE embauche &gt;= '01-jan-94' AND n_dept = e.n_dept); </pre>	Quels sont les employés travaillant dans un département qui a procédé à des embauches depuis le début de l'année 94.
<pre> SELECT nom, fonction FROM emp WHERE n_dept = 10 AND fonction IN (SELECT fonction FROM emp WHERE n_dept = (SELECT n_dept FROM emp WHERE nom = 'DUPONT')); </pre>	Liste des employés du département 10 ayant même fonction que quelqu'un du département de DUPONT.

## 5 Expressions et Fonctions simples

Une expression est un ensemble de variables (contenu d'une colonne), de constantes et de fonctions combinées au moyen d'opérateurs. Les fonctions prennent une valeur dépendant de leurs arguments qui peuvent être eux-mêmes des expressions.

<pre> SELECT nom, salaire+comm FROM emp WHERE fonction = 'commercial'; </pre>	Donner pour chaque commercial son revenu (salaire + commission).
<pre> SELECT nom, comm/salaire, comm, salaire FROM emp WHERE fonction = 'commercial' ORDER BY comm/salaire DESC; </pre>	Donner la liste des commerciaux classée par commission sur salaire décroissant.
<pre> SELECT nom, salaire, comm FROM emp WHERE comm &lt;= salaire *.05; </pre>	Donner la liste des employés dont la commission est inférieure à 5% du salaire.
<pre> SELECT nom, ROUND(salaire/22,2) FROM emp; </pre>	Donner pour chaque employé son salaire journalier.
<pre> SELECT NEXT_DAY (embauche,'MONDAY') FROM emp; </pre>	Donner la date du lundi suivant l'embauche de chaque employé.
<pre> SELECT ROUND (embauche,'Y') FROM emp; </pre>	Donner la date d'embauche de chaque employé arrondie à l'année.
<pre> SELECT ROUND (SYSDATE-embauche) FROM emp; </pre>	Donner pour chaque employé le nombre de jours depuis son embauche.
<pre> SELECT TO_CHAR (embauche,'DD/MM/YY HH24:MI:SS') FROM emp; </pre>	On peut également insérer dans le format une chaîne de caractères quelconque, à condition de la placer entre guillemets "".
<pre> SELECT INSTR(fonction,'a',1,2) FROM emp; </pre>	
<pre> SELECT nom FROM emp WHERE SOUNDEX(nom) = SOUNDEX('DUPONT'); </pre>	Donner la liste de tous les employés dont le nom ressemble à DUPONT.
<pre> SELECT LTRIM(nom,'LE') FROM emp; </pre>	Donner la liste de tous les noms des employés en ayant supprimé tous les 'L' et les 'E' en tête des noms.
<pre> SELECT TRANSLATE (nom,'AM','**') FROM emp; </pre>	Donner la liste de tous les noms des employés en ayant remplacé les A et les M par des * dans les noms.

<code>SELECT TO_CHAR (salaire,'99900.00') FROM emp;</code>	Afficher tous les salaires avec un \$ en tête et au moins trois chiffres ( dont deux décimales).
<code>SELECT nom, salaire, comm, salaire+NVL(comm,0) FROM emp;</code>	Donner pour chaque employé ses revenus (salaire + commission).
<code>SELECT nom, DECODE(fonction,'president',1,'directeur',2,3) FROM emp;</code>	Donner la liste des employés avec pour chacun d'eux sa catégorie (président = 1, directeur = 2, autre = 3)
<code>SELECT DECODE (n_dept,10,fonction,nom) FROM emp;</code>	Donner la liste des employés en les identifiant par leur fonction dans le département 10 et par leur nom dans les autres départements.

## 6 Les fonctions de groupe

autre appellation: fonction d'aggrégation, fonction statistique

Il existe un autre type de SELECT qui permet d'effectuer des calculs sur l'ensemble des valeurs d'une colonne.

<code>SELECT SUM(salaire) FROM emp WHERE n_dept = 10;</code>	Donner le total des salaires du département 10.
<code>SELECT nom, fonction, salaire FROM emp WHERE salaire = (SELECT MAX(salaire) FROM emp);</code>	Donner le nom, la fonction et le salaire de l'employé (ou des employés) ayant le salaire le plus élevé.
<code>SELECT SUM(salaire), n_dept FROM emp GROUP BY n_dept;</code>	Total des salaires pour chaque département
<code>SELECT fonction,COUNT(*),AVG(salaire) FROM emp GROUP BY fonction HAVING COUNT(*) &gt; 2;</code>	Donner la liste des salaires moyens par fonction pour les groupes ayant plus de deux employés.
<code>SELECT n_dept, COUNT(*) FROM emp WHERE fonction in ('ingenieur','commercial') GROUP BY n_dept HAVING COUNT(*) &gt;= 2;</code>	Donner le nombre d'ingénieurs ou de commerciaux des départements ayant au moins deux employés de ces catégories.
<code>SELECT n_dept,COUNT(*) FROM emp GROUP BY n_dept HAVING COUNT(*) = (SELECT MAX(COUNT(*)) FROM emp GROUP BY n_dept) ;</code>	Quel est le département ayant le plus d'employés?
<code>SELECT MAX(COUNT(*)) FROM emp GROUP BY n_dept ;</code>	la fonction MAX peut être appliquée aux nombres d'employés de chaque département pour obtenir le nombre d'employés du département ayant le plus d'employés.

# SQL-DML : Modifier le contenu d'une base

## 7 Ajout de lignes

La commande INSERT permet d'insérer une ligne dans une table en spécifiant les valeurs à insérer.

INSERT INTO nom_table(nom_col1, nom_col2, ...) VALUES (val1, val2...)	La syntaxe est la suivante
INSERT INTO bonus SELECT nom, salaire FROM emp WHERE fonction = 'directeur';	Insérer dans la table bonus les noms et salaires des directeurs

## 8 Modification de lignes

La commande UPDATE permet de modifier les valeurs d'une ou plusieurs colonnes, dans une ou plusieurs lignes existantes d'une table.

UPDATE nom_table SET nom_col1 = {expression1   ( SELECT ... ) }, nom_col2 = {expression2   ( SELECT ... ) }, ... nom_colp = {expressionp   ( SELECT ... ) } WHERE predicat	La syntaxe est la suivante :
UPDATE emp SET salaire = salaire * 1.1 WHERE fonction = 'ingenieur' ;	Augmenter de 10% les ingénieurs.

## 9 Suppression de lignes

La commande DELETE permet de supprimer des lignes d'une table.

La syntaxe est la suivante :

```
DELETE FROM nom_table
WHERE predicat ;
```

Toutes les lignes pour lesquelles prédicat est évalué à vrai sont supprimées. En l'absence de clause >WHERE, toutes les lignes de la table sont supprimées.

## 10 Gestion des transactions

Une transaction est un ensemble de modifications de la base qui forme un tout indivisible. Il faut effectuer ces modifications entièrement ou pas du tout, sous peine de laisser la base dans un état incohérent.

Les Systèmes de Gestion de Bases de Données permettent aux utilisateurs de gérer leurs transactions. Ils peuvent à tout moment :

- Valider la transaction en cours par la commande **COMMIT**. Les modifications deviennent définitives et visibles à tous les utilisateurs.
- Annuler la transaction en cours par la commande **ROLLBACK**. Toutes les modifications depuis le début de la transaction sont alors défaites

## 11 Annexe : Schéma de scott

