

Chapitre 4: Langage SQL
Langage de Manipulation de
Données
LMD: INSERT, UPDATE &
DELETE

Plan

- · Introduction au Langage SQL
- · Création, modification et suppression des tables
- · Spécification des contraintes d'intégrité
- · Exécution des commandes LDD sous MS Access
- · Création des Tables à l'aide de l'interface de MS Access
- Mettre à jour les données avec SQL : LMD
- · Mettre à jour les données avec MS ACCEES
- · Ecrire des instructions SQL SELECT élémentaires
- · Limiter et trier des données
- · Afficher les données issues de plusieurs tables : Les jointures
- · Agréger des données à l'aide de fonctions de groupe
- Sous-interrogations
- · Sous-interrogations
- · Création des requêtes MS ACCEES
- · Création des formulaires et sous formulaires MS ACCEES
- · Création des états MS ACCEES

Introduction

Le langage de manipulation de données (LMD) est le langage permettant de modifier les informations contenues dans la base.

Il existe trois commandes SQL permettant d'effectuer les trois types de modification des données :

INSERT ajout de lignes UPDATE mise à jour de lignes DELETE suppression de lignes

Définition

 Une transaction est un ensemble d'instructions LMD formant une seule unité (début et fin) de travail logique : qu'on peut annuler en entier, c' est à dire en un seule fois, ou la valider en entier.

Exemple: Une transaction peut transférer une somme d'argent entre deux comptes d'un client d'une banque.

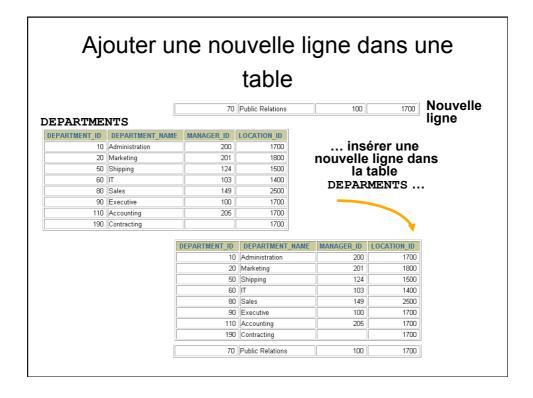
Elle comporte deux ordres : un débit sur un compte et un crédit sur un autre compte.

Si un problème empêche le crédit, le débit doit être annulé.

Insertion

Sous SQL, il existe deux manières de base pour INSÉRER des données dans une table :

- l'une consiste à insérer des données une ligne à la fois,
- l'autre <u>plusieurs lignes</u> à la fois. (on utilise l'instruction SELECT pour spécifier les données à ajouter à la table.)



Syntaxe de l'instruction INSERT

 L'instruction INSERT permet d'ajouter de nouvelles lignes dans une table.

```
INSERT INTO "nom de table" ("colonne 1", "colonne 2", ...) VALUES ("valeur 1", "valeur 2", ...);
```

 Cette syntaxe n'insère qu'une seule ligne à la fois.

Insérer de nouvelles lignes

- Insérez une nouvelle ligne en précisant une valeur pour chaque colonne.
- Indiquez les valeurs dans l'ordre par défaut des colonnes dans la table.
- Indiquez éventuellement les colonnes dans la clause INSERT.

 Placez les valeurs de type caractère entre apostrophes et date entre # #.

EXEMPLE

Table Store_Information

Nom de Colonne	Type de Données
Store_Name	char(50)
Sales	float
Txn_Date	datetime

pour insérer une ligne supplémentaire dans la table représentant les données de ventes pour <u>Los Angeles</u> le <u>10 janvier 1999</u>, dont les ventes de ce magasin, à ce jour, s'élevaient à 900 €,

INSERT INTO Store_Information (Store_Name, Sales, Txn_Date) VALUES ('Los Angeles', 900, #10-Jan-1999#);

Insérer des dates & NULL

- o indiquez le mot-clé NULL dans la clause VALUES.
- Ajoutez un nouvel employé.

```
INSERT INTO employees

VALUES (114,

'Den', 'Raphealy',

'DRAPHEAL', '515.127.4561',

#3-2-1999#,

'AC_ACCOUNT', 11000, NULL, 100, 30);
```

· Vérifiez l'ajout.



Modifier les données d'une table

EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSION_F
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000	90	
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000	90	
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000	90	
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000	60	
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	60	
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	60	
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	

Mettez à jour les lignes de la table EMPLOYEES.



Syntaxe de l'instruction UPDATE

 Utilisez l'instruction UPDATE pour modifier des lignes existantes.

```
UPDATE "nom de table"
SET "colonne 1" = [nouvelle valeur]
WHERE "condition";
```

Si nécessaire, vous pouvez modifier plusieurs lignes à la fois.
 La syntaxe dans ce cas-ci ressemblerait à ce qui suit :

```
UPDATE "nom de table"
SET colonne 1 = [valeur 1], colonne 2 = [valeur 2]
WHERE "condition";
```

Modifier des lignes d'une table

• La clause WHERE permet de modifier une ou plusieurs lignes spécifiques.

```
UPDATE employees
SET   department_id = 70
WHERE employee_id = 113;
1 row updated.
```

En cas d'absence de la clause WHERE, toutes les lignes sont modifiées.

```
UPDATE copy_emp
SET department_id = 110;
22 rows updated.
```

EXEMPLE

Table Store_Information

Store_Name	Sales	Txn_Date
Los Angeles	1500	05-Jan-1999
San Diego	250	07-Jan-1999
Los Angeles	300	08-Jan-1999
Boston	700	08-Jan-1999

et nous nous rendons compte que les ventes pour Los Angeles du 08-Jan-1999 sont en réalité de 500 € au lieu de 300 €, et que cette entrée particulière doit être corrigée. Pour ce faire, nous utiliserons la requête SQL suivante :

UPDATE Store_Information
SET Sales = 500
WHERE Store_Name = 'Los Angeles'
AND Txn_Date = '08-Jan-1999';

EXEMPLE

UPDATE Store_Information
SET Sales = 500
WHERE Store_Name = 'Los Angeles'
AND Txn_Date = '08-Jan-1999';

Table Store_Information

 Store_Name
 Sales
 Txn_Date

 Los Angeles
 1500
 05-Jan-1999

 San Diego
 250
 07-Jan-1999

 Los Angeles
 300
 08-Jan-1999

 Boston
 700
 08-Jan-1999

La table résultante ressemblerait à

Table *Store_Information*

 Store_Name
 Sales
 Txn_Date

 Los Angeles
 1500
 05-Jan-1999

 San Diego
 250
 07-Jan-1999

 Los Angeles
 500
 08-Jan-1999

 Boston
 700
 08-Jan-1999

Supprimer une ligne d'une table

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing		
100	Finance		
50	Shipping	124	1500
60	Π	103	1400

Supprimez une ligne de la table DEPARTMENTS.

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing		
50	Shipping	124	1500
60	IT	103	1400

Instruction DELETE

 Vous pouvez supprimer des lignes d'une table au moyen de l'instruction DELETE.

DELETE FROM table_name
[WHERE condition];

Supprimer des lignes d'une table

• La clause **WHERE** permet de supprimer des lignes spécifiques.

```
DELETE FROM departments
WHERE department_name = 'Finance';
1 ligne supprimée.
```

• En cas d'absence de la clause WHERE, toutes les lignes sont supprimées.

```
DELETE FROM copy_emp;
22 lignes supprimées.
```

Erreur de contrainte d'intégrité lors de la suppression de lignes

```
DELETE FROM departments
WHERE department_id = 60;
```

```
DELETE FROM departments

*

ERROR at line 1:

ORA-02292: integrity constraint (HR.EMP_DEPT_FK)

violated - child record found
```

Vous ne pouvez pas supprimer une ligne qui contient une clé primaire utilisée comme clé étrangère dans une autre table.

EXEMPLE

Dans l'exemple de la table : Store_Information Nous décidons de ne conserver aucune information de Los Angeles dans cette table. Pour ce faire, nous saisissons la requête SQL suivante :

DELETE FROM Store_Information WHERE Store_Name = 'Los Angeles';

La table résultante ressemblerait à

Table Store_Information

 Store_Name
 Sales
 Txn_Date

 Los Angeles
 1500
 05-Jan-1999

 San Diego
 250
 07-Jan-1999

 Los Angeles
 500
 08-Jan-1999

 Boston
 700
 08-Jan-1999

Le contenu de la table devrait paraître à

Table Store_Information

Store_Name	Sales	Txn_Date
San Diego	250	07-Jan-1999
Boston	700	08-Jan-1999

Chapitre 5

Langage SQL Interrogation : SELECT

Plan

- Introduction au Langage SQL
- O Création, modification et suppression des tables
- O Spécification des contraintes d'intégrité
- Exécution des commandes LDD sous MS Access
- O Création des Tables à l'aide de l'interface de MS Access
- Mettre à jour les données avec SQL : LMD
- Mettre à jour les données avec MS ACCEES
- Ecrire des instructions SQL SELECT élémentaires
- Limiter et trier des données
- Afficher les données issues de plusieurs tables : Les jointures
- Agréger des données à l'aide de fonctions de groupe
- Sous-interrogations
- Création des requêtes MS ACCEES

Tables utilisées dans le cours

Table employees

Name	Null?	Туре
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2 (25)
EMAIL	NOT NULL	VARCHAR2 (25)
PHONE_NUMBER		VARCHAR2 (20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

Tables utilisées dans le cours

• Table DEPARTMENTS

Name	Null?	Туре
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700
8 rows selected			

8 rows selected

Ecrire des instructions SQL SELECT élémentaires

Ecrire des instructions SQL SELECT élémentaires

Instruction SELECT élémentaire

```
SELECT *|{[DISTINCT] column|expression [alias],...}
FROM table;
```

- SELECT indique quelles colonnes renvoyer
- FROM indique dans quelle table rechercher

Les requêtes simples

Une requête simple, ou autrement appelé une projection, sert à sélectionner un ensemble d'attribut dans une table. Sa syntaxe est :

```
SELECT liste des attributs (colonnes ) FROM table ;
```

pour afficher toutes les colonnes de données d'une table en place un (*) à la suite du mot-clé SELECT. : SELECT * FROM table;

Sélectionner toutes les colonnes SELECT * FROM departments; DEPARTMENT ID DEPARTMENT NAME 200 1700 10 Administration 20 Marketing 201 1800 50 Shipping

124

103

149

100

1500

1400

2500

1700

1700

1700

8 rows selected.

60 IT

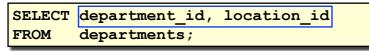
80 Sales

90 Executive

110 Accounting

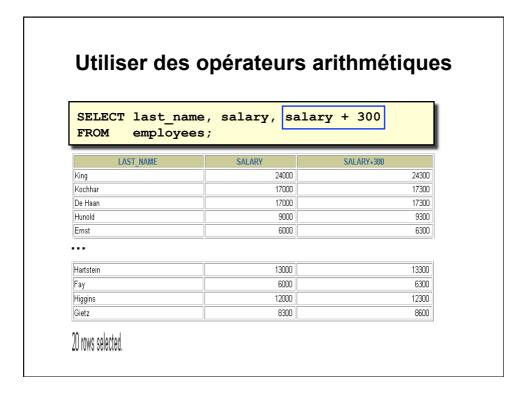
190 Contracting

Sélectionner des colonnes spécifiques



DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1500
60	1400
80	2500
90	1700
110	1700
190	1700

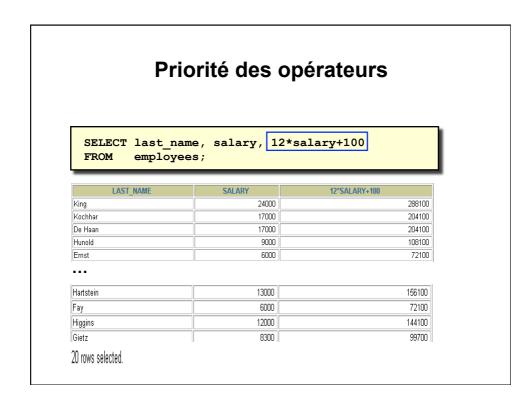
8 rows selected.

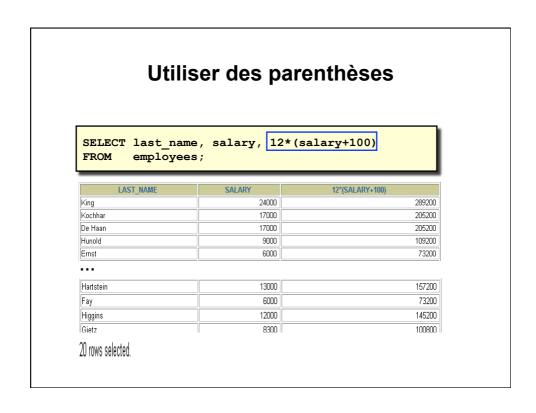


Priorité des opérateurs



- La multiplication et la division ont priorité sur l'addition et la soustraction.
- Les opérateurs de niveau de priorité identique sont évalués de gauche à droite.
- Les parenthèses permettent de forcer la priorité d'évaluation et de clarifier les instructions.

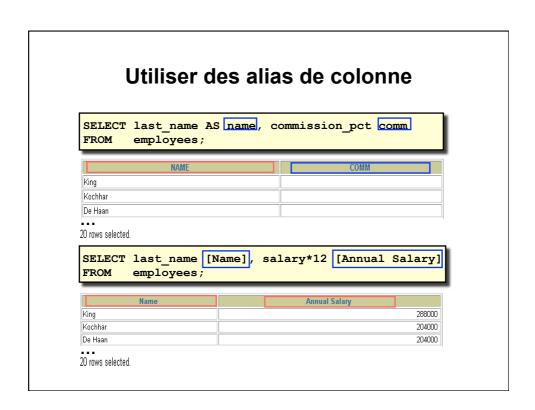




Définir un alias de colonne

L'alias de colonne :

- renomme un en-tête de colonne,
- est utile dans les calculs,
- suit le nom de la colonne (le mot-clé AS facultatif peut être placé entre le nom de la colonne et l'alias),
- doit obligatoirement être placé entre crochets [] s'il contient des espaces ou des caractères spéciaux, ou bien si les majuscules/minuscules doivent être respectées.



DISTINCT

Pour ne pas avoir de redondance (doubles / doublons) dans la sélection on utilise l'expression 'DISTINCT'

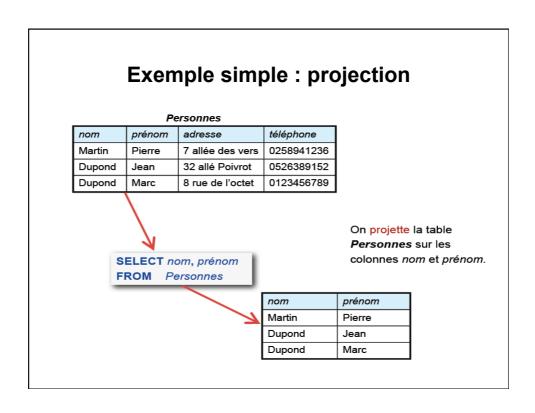
DEPARTEMENT nodep nomdep ville Développement Marseille 35 Développement Nice 45 Toulon Analyse 50 Maintenance Paris 60 Recherche Toulouse

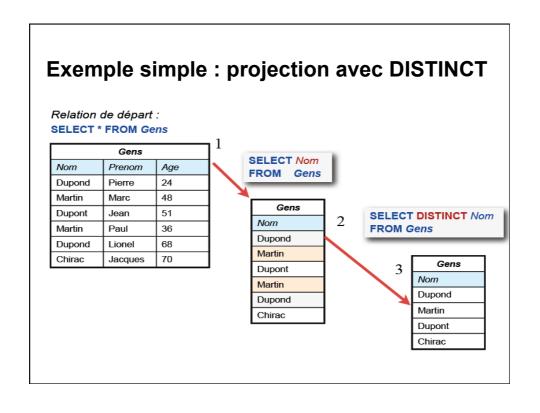
SELECT DISTINCT nomdep FROM DEPARTEMENT;

Le résultat de la requête est :

nomdep
Développement
Analyse
Maintenance
Recherche

DISTINCT Soit la table VOITURE suivante : Marque Modele Serie Numero Renault RL 4698 SJ 45 Renault RL 4568 HD 16 Kangoo Renault Kangoo RL 6576 VE 38 Peugeot 106 **KID** 7845 ZS 83 309 7647 ABY 82 Peugeot chorus 8562 EV 23 Ford Escort Match **SELECT DISTINCT Modele, Serie** FROM VOITURE: Modele Serie 18 RL Kangoo RL 106 KID 309 chorus **Escort** Match





Limiter et trier des données

- Clause WHERE
- Clause ORDER BY

Clause WHERE de restriction

• Limitez le nombre de lignes sélectionnées à l'aide de la clause **WHERE**.

```
SELECT *|{[DISTINCT] column|expression [alias],...}
FROM table
[WHERE condition(s)];
```

 La clause WHERE se place toujours après la clause FROM.

WHERE se compose de trois éléments :

- Nom de colonne
- Condition de comparaison ou expressions
- Constante ou liste de valeurs.

Utiliser la clause WHERE

SELECT employee_id, last_name, job_id, department_id FROM employees WHERE department_id = 90 ;

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID		
100	King	AD_PRES		90	
101	Kochhar	AD_VP		90	
102	De Haan	AD_VP		90	





nom prénom téléphone Martin 0258941236 Pierre 7 allée des vers 0526389152 Dupond 32 allé Poivrot Jean 0123456789 Dupond Marc 8 rue de l'octet

> FROM Personnes WHERE nom = "Dupond"

On ne sélectionne que les tuples dont l'attribut nom est égale à 'Dupond'.

prénom adresse téléphone 32 allé Poivrot 0526389152 Dupond Jean Dupond Marc 8 rue de l'octet 0123456789

Conditions de comparaison

Opérateur	Signification
II	Egal à
>	Supérieur à
>=	Supérieur ou égal à
<	Inférieur à
<=	Inférieur ou égal à
<>	Différent de

Conditions de comparaison

- Ces opérateurs de comparaison s'utilisent dans des conditions qui comparent une expression avec une autre valeur ou expression. Dans la clause WHERE, ils s'utilisent de la façon suivante :
- Syntaxe

```
... WHERE expr operator value
```

- Exemple:

```
... WHERE hire_date=#01-01-95#
... WHERE salary>=6000
... WHERE last_name='Smith'
```

- Vous ne pouvez pas utiliser d'alias dans la clause WHERE.

Utiliser des conditions de comparaison

SELECT last_name, salary
FROM employees
WHERE salary <= 3000;

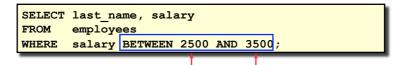
LAST_NAME	SALARY
Matos	2600
Vargas	2500

Autres conditions de comparaison

Opérateur	Signification
[NOT]BETWEEN	Compris entre et (bornes comprises)
[NOT] IN (set)	Correspond à une valeur de la liste
[NOT] LIKE	Ressemblance partielle de chaînes de caractères
IS[NOT] NULL	Correspond à une valeur NULL

Utiliser la condition BETWEEN

Utilisez la condition BETWEEN pour afficher des lignes en fonction d'une plage de valeurs.

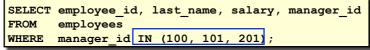


Limite inférieure Limite supérieure

LAST_NAME	SALARY
Rajs	3500
Davies	3100
Matos	2600
Vargas	2500

Utiliser la condition IN

Utilisez la condition d'appartenance IN pour vérifier la présence de valeurs dans une liste.



EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
202	Fay	6000	201
200	Whalen	4400	101
205	Higgins	12000	101
101	Kochhar	17000	100
102	De Haan	17000	100
124	Mourgos	5800	100
149	Zlotkey	10500	100
201	Hartstein	13000	100

8 rows selected.

Utiliser la condition LIKE

- Utilisez la condition LIKE pour rechercher des chaînes de caractères valides à l'aide de caractères génériques.
- Les conditions de recherche peuvent contenir des caractères ou des nombres littéraux :

```
SELECT first_name
FROM employees
WHERE first_name LIKE \S*';
```

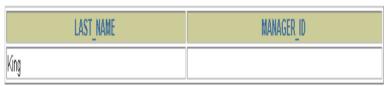
- * représente zéro ou plusieurs caractères.
- ? représente un caractère.

```
SELECT first_name
FROM employees
WHERE first_name LIKE \?S*';
```

Utiliser les conditions NULL

Recherchez des valeurs NULL avec l'opérateur Is NULL.



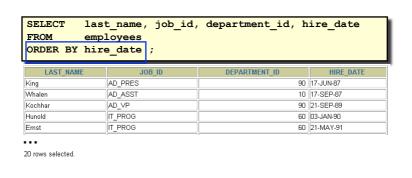


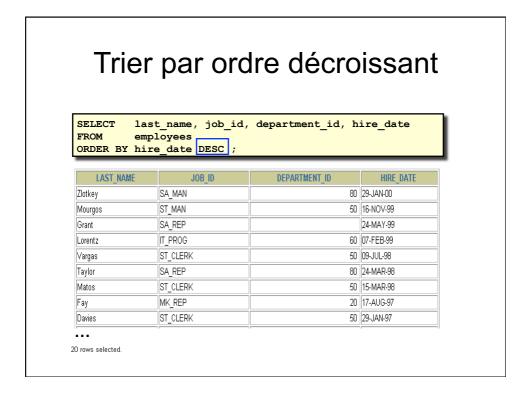
Conditions logiques

Opérateur	Signification
AND	Renvoie TRUE si les deux conditions sont vraies
OR	Renvoie TRUE si <i>l'une</i> des conditions est vraie
NOT	Renvoie la valeur TRUE si la condition qui suit l'opérateur est fausse

Clause ORDER BY

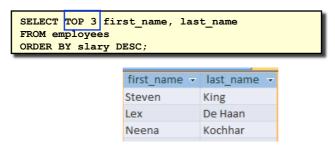
- Triez des lignes à l'aide de la clause ORDER BY.
 - ASC : ordre croissant (par défaut)
 - **DESC**: ordre décroissant
- La clause ORDER BY se place à la fin de l'instruction SELECT.





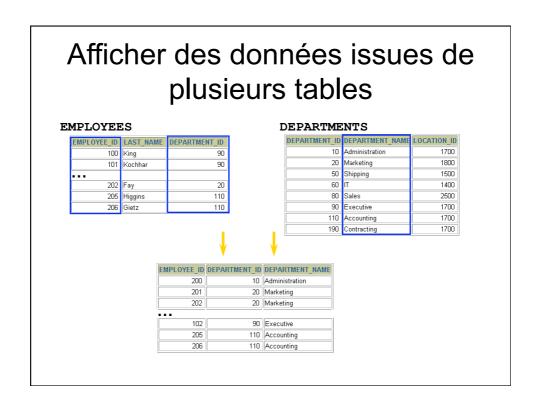
Traitement les n premiers : TOP n

• Quels sont les 3 salariés les mieux payés :



 L'emploi de la clause ORDER BY est obligatoire.

Les jointures Afficher des données issues de plusieurs tables



Joindre des tables : Définition d'une équijointure

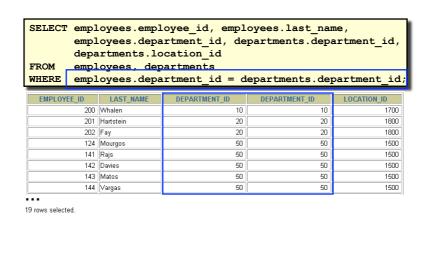
Une jointure sert à interroger des données issues de plusieurs tables.

```
SELECT table1.column, table2.column
FROM table1, table2
WHERE table1.column1 = table2.column2;
```

•

Définition d'une équijointure **EMPLOYEES DEPARTMENTS** EMPLOYEE_ID DEPARTMENT NAMI 200 Administration 20 Marketing 201 20 Marketing 202 50 Shipping 124 Shipping 50 141 50 Shipping 142 50 Shipping 143 50 50 Shipping 144 60 IT 103 60 60 IT 149 80 Sales 80 Sales 176 80 80 Sales Clé étrangère Clé primaire

Extraire des enregistrements à l'aide d'équijointures



Utiliser des alias de table

 Simplifiez la rédaction des interrogations à l'aide des alias de table.

```
SELECT e employee_id, e last_name, e department_id,
d.department_id, d.location_id

FROM employees e , departments d

WHERE e.department_id = d.department_id;
```

Joindre plus de deux tables

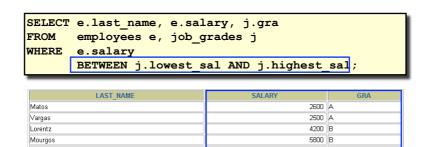
EMPLOYE	ES	DEPARTMEN	TS	LOCATI	ONS
LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID	LOCATION_ID	CITY
King	90	10	1700	1400	Southlake
Kochhar	90	20	1800	1500	South San Francisco
De Haan	90	50	1500	1700	Seattle
Hunold	60	60	1400	1800	Toronto
Ernst	60	80	2500	2500	Oxford
Lorentz	60	90	1700		
Mourgos	50	110	1700		
Rajs	50	190	1700		
Davies	50	3 rows selected.			
Matos	50				
Vargas	50				
Zlotkey	80				
Abel	80				
Taylor	80				

Pour joindre **n** tables entre elles, il faut au minimum **n-1** conditions de jointure. Par exemple, deux jointures au moins sont nécessaires pour joindre trois tables.

Non-équijointures

- Une non-équijointure est une condition de jointure contenant un opérateur qui n'est pas un opérateur d'égalité.
- La relation entre les tables EMPLOYEES et JOB_GRADES est un exemple de non-équijointure. Cette relation indique que les valeurs de la colonne SALARY de la table EMPLOYEES doivent être comprises entre les valeurs des colonnes LOWEST_SALARY et HIGHEST_SALARY de la table JOB_GRADES.
- Elle est obtenue à l'aide d'un opérateur autre que le signe égal (=).

Extraire des enregistrements à l'aide de non-équijointures



3500 B

3100 B

20 rows selected.

Rajs

Davies

Whalen Hunold Ernst

Agréger des données à l'aide de fonctions de groupe

Définition des fonctions de groupe

Les fonctions de groupe agissent sur des groupes de lignes et donnent un résultat par groupe.

DEPARTMENT_ID	SALARY		
90	24000		
90	17000		
90	17000		
60	9000		
60	6000		
60	4200	Salaire	
50	5800	Maximum	
50	3500	dans	
50	3100		MAX(SALARY)
50	2600	la table	2400
50	2500	EMPLOYEES.	
80	10500		
80	11000		
80	8600		
	7000		
10	4400		

Types de fonction de groupe

- AVG (calcule la moyenne)
- COUNT (Compte des lignes)
- MAX (calcule la maximum)
- MIN (calcule le minimum)
- SUM (calcule la somme)

Syntaxe des fonctions de groupe

```
SELECT [column,] group function(column), ...

FROM table
[WHERE condition]
[GROUP BY column]
[ORDER BY column];
```

Utiliser les fonctions MIN et MAX

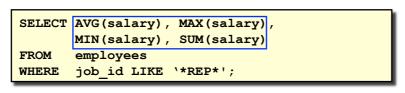
Les fonctions MIN et MAX s'utilisent avec tous les types de données.

```
SELECT MIN(hire_date), MAX(hire_date)
FROM employees;
```

MIN(HIRE_	MAX(HIRE_	
117- HINL87	29-JAN-00	

Utiliser les fonctions AVG et SUM

Les fonctions AVG et SUM s'utilisent avec des données numériques.



AVG(SALARY)	MAX(SALARY)	MIN(SALARY)	SUM(SALARY)	
8150	11000	6000	32600	

Utiliser la fonction COUNT

La fonction COUNT (*) renvoie le nombre de lignes d'une table

```
SELECT COUNT(*)
FROM employees
WHERE department_id = 50;
```



Utiliser la fonction COUNT

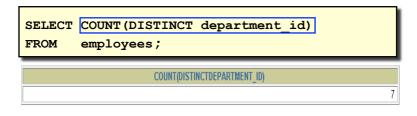
- La fonction COUNT (expr) renvoie le nombre de lignes contenant des valeurs non NULL dans la colonne expr.
- Affichez le nombre de valeurs contenues dans la colonne commission_pct de la table EMPLOYEES, à l'exception des valeurs NULL.

```
SELECT COUNT(commission_pct)
FROM employees
WHERE department_id = 80;

COUNT(COMMISSION_PCT)
```

Utiliser le mot-clé DISTINCT (Oracle)

- La fonction COUNT (DISTINCT expr) renvoie le nombre de valeurs non NULL distinctes de la colonne expr.
- Affichez le nombre de services distincts contenus dans la table EMPLOYEES.



Fonctions de groupe et valeurs NULL

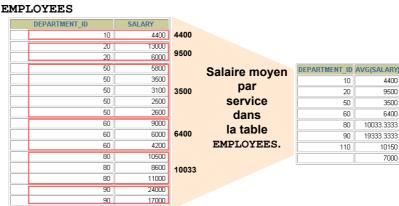
Les fonctions de groupe ignorent les valeurs NULL des colonnes.

SELECT AVG(commission_pct)
FROM employees;

AVG(COMMISSION_PCT) .2125

- la moyenne est calculée uniquement sur les lignes pour lesquelles la colonne COMMISSION PCT est correctement renseignée.
- Le calcul de la moyenne s'effectue par division du total des commissions réellement versées à tous les employés par le nombre d'employés touchant une commission.

Créer des groupes de données



20 rows selected

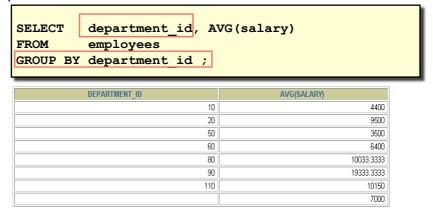
Créer des groupes de données : syntaxe de la clause GROUP BY

```
SELECT column, group_function(column)
FROM table
[WHERE condition]
[GROUP BY group_by_expression]
[ORDER BY column];
```

La clause GROUP BY permet d'organiser les lignes d'une table en groupes restreints.

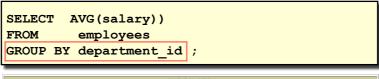
Utiliser la clause GROUP BY

La clause GROUP BY doit inclure obligatoirement toutes les colonnes, de la liste SELECT, utilisées avant les fonctions de groupe.

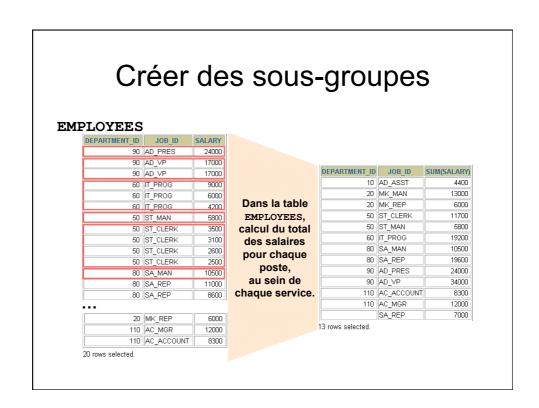


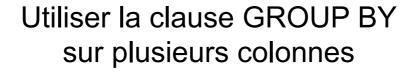
Utiliser la clause GROUP BY

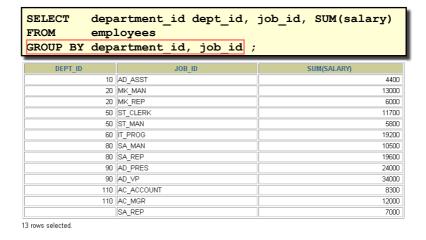
La colonne utilisée en clause GROUP BY ne doit pas nécessairement figurer dans la liste SELECT.

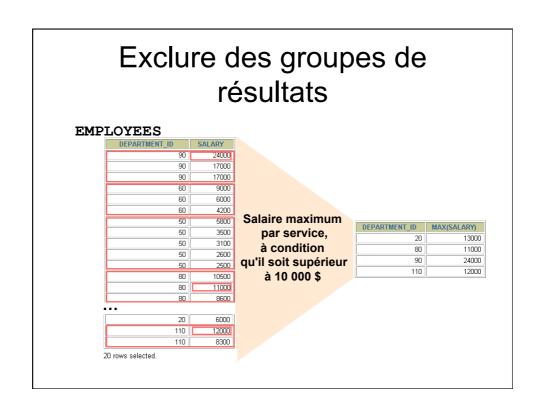


AVG(SALARY)	
	4400
	9500
	3500
	6400
	10033.3333
	19333.3333
	10150
	7000









Exclure des groupes de résultats : clause HAVING

Utilisez la clause HAVING pour restreindre les groupes.

- 1. Les lignes sont regroupées.
- 2. La fonction de groupe est appliquée.
- 3. Les groupes qui correspondent à la Clause HAVING s'affichent.

```
SELECT column, group_function

FROM table
[WHERE condition]
[GROUP BY group_by_expression]
[HAVING group condition]
[ORDER BY column];
```

Utiliser la clause HAVING

SELECT department_id, MAX(salary)
FROM employees
GROUP BY department_id
HAVING MAX(salary)>10000 ;

DEPARTMENT_ID	MAX(SALARY)
20	13000
80	11000
90	24000
110	12000

Utiliser la clause HAVING

```
SELECT job_id, SUM(salary) PAYROLL
FROM employees
WHERE job_id NOT LIKE '*REP*'
GROUP BY job_id
HAVING SUM(salary) > 13000
ORDER BY SUM(salary);
```

JOB_ID	PAYROLL
IT_PROG	19200
AD_PRES	24000
AD_VP	34000

ref.web

file://localhost/Users/ouzarf/Desktop/BD14-15/ langagesql.free.fr/3d1_base-de-donnees.htm

file://localhost/Users/ouzarf/Desktop/BD14-15/Cours %20langage%20SQL%20-%20langage%20SQL %20initiation.html#sql-commande-select

http://www.actualitix.com/cours-langage-sql-langage-sql-initiation.html#sql-commande-select

http://www.1keydata.com/fr/sql/sql-order-by.php