SQL

Structured
Query
Language

SQL

SQL

Un langage informatique conçu pour la récupération et la gestion des données dans des systèmes de gestion de bases de données, création de schémas de bases de données et la modification, et objet de base de gestion de contrôle d'accès.

SQL

Fournit des commandes pour diverses tâches, notamment:

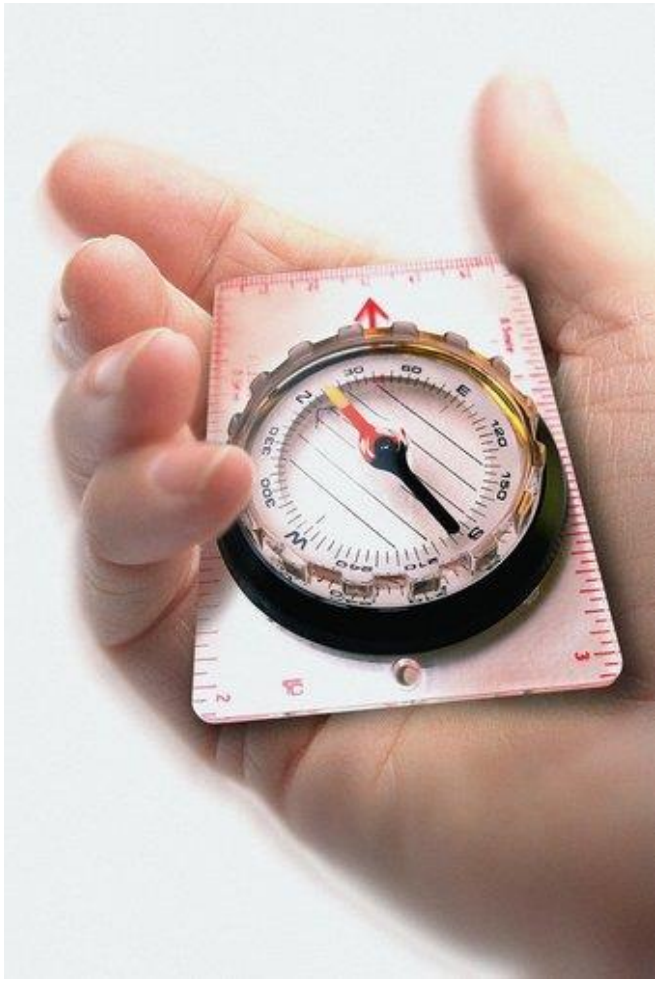
- Interrogation des données
- Insertion, mise à jour, et suppression de lignes dans table
- Création, remplacement, modification et suppression d'objets
- Contrôler l'accès à la base de données et ses objets
- Garantir la cohérence de bases de données et l'intégrité

SQL

Type de commandes	Commandes
Data manipulation language (DML)	SELECT INSERT UPDATE DELETE MERGE
Data definition language (DDL)	CREATE ALTER DROP RENAME TRUNCATE COMMENT
Data control language (DCL)	GRANT REVOKE
Transaction control (TCL)	COMMIT ROLLBACK SAVEPOINT

objectifs du cours

En complétant ce cours, vous serez capable de:



- Exécuter une instruction SELECT de base
- Limite et de trier les lignes qui sont récupérées par une requête

SELECT Commande

Aperçu

- Quelle est l'instruction SELECT?
- Expressions arithmétiques.
- Autres possibilités sur SELECT.

Quelle est l'instruction SELECT?

Projection

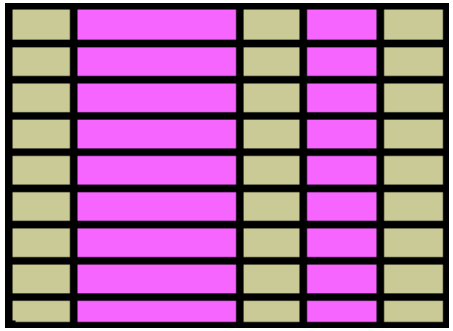


Table 1

Selection

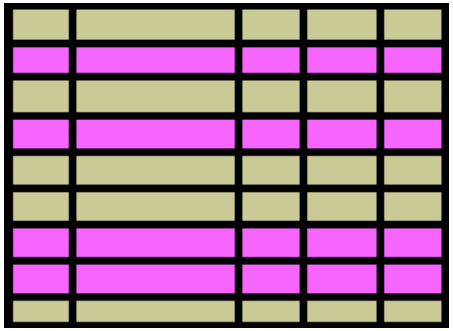


Table 1

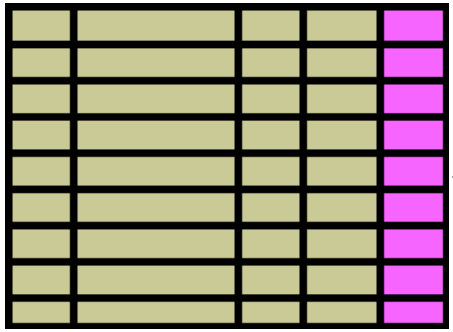


Table 1

Join

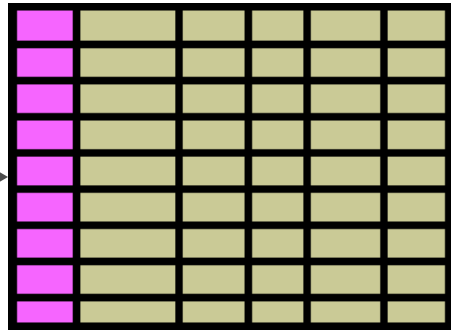


Table 2

Quelle est l'instruction SELECT?

Instruction SELECT de base:

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

- SELECT identifie la colonne à afficher
- FROM Identifie la table contenant les colonnes

Quelle est l'instruction SELECT ?

Sélection de toutes les colonnes :

```
SELECT *
FROM   departments;
```

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.

Quelle est l'instruction SELECT ?

Sélection de colonnes spécifiques :

```
SELECT department_id, location_id
FROM   departments;
```

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

8 rows selected.

Quelle est l'instruction SELECT ?

- Instructions SQL ne sont pas sensibles à la casse.
- Les instructions SQL peuvent être sur une ou plusieurs lignes.
- Mots-clés ne peuvent pas être abrégée ou répartis sur plusieurs lignes.
- Des clauses sont généralement placés sur des lignes séparées.
- Tirets sont utilisés pour améliorer la lisibilité.
- Dans SQL peuvent éventuellement être terminée par un point-virgule (;). Virgule est requis si vous exécutez plusieurs instructions SQL.
- vous êtes tenu de mettre fin à chaque instruction SQL par un point virgule (;).

expressions arithmétiques

Créer des expressions avec le numéro et la date à l'aide des données des opérateurs arithmétiques.

Operator	Description
+	Add
–	Subtract
*	Multiply
/	Divide

expressions arithmétiques

Using Arithmetic Operators:

```
SELECT last_name, salary, salary + 300
FROM   employees;
```

LAST_NAME	SALARY	SALARY+300
King	24000	24300
Kochhar	17000	17300
De Haan	17000	17300
Hunold	9000	9300
Ernst	6000	6300

▪ ▪ ▪

20 rows selected.

expressions arithmétiques

Operator Precedence:

```
SELECT last_name, salary, 12*salary+100
FROM   employees;
```

LAST_NAME	SALARY	12*SALARY+100
King	24000	288100
Kochhar	17000	204100
De Haan	17000	204100

▪ ▪ ▪

20 rows selected.

```
SELECT last_name, salary, 12*(salary+100)
FROM   employees;
```

LAST_NAME	SALARY	12*(SALARY+100)
King	24000	289200
Kochhar	17000	205200
De Haan	17000	205200

▪ ▪ ▪

20 rows selected.

expressions arithmétiques

Définir une valeur null:

- Un nul est une valeur qui n'est pas disponible, sans affectation, inconnu ou inapplicable.
- Un nul n'est pas le même que celui d'un zéro ou un espace blanc.

```
SELECT last_name, job_id, salary, commission_pct
FROM   employees;
```

LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
King	AD_PRES	24000	
Kochhar	AD_VP	17000	
...			
Zlotkey	SA_MAN	10500	.2
Abel	SA_REP	11000	.3
Taylor	SA_REP	8600	.2
...			
Gietz	AC_ACCOUNT	8300	

20 rows selected.

expressions arithmétiques

Les valeurs NULL dans les expressions arithmétiques:

- Les expressions arithmétiques contenant une valeur NULL sont évaluées à NULL.

```
SELECT last_name, 12*salary*commission_pct
FROM   employees;
```

LAST_NAME	12*SALARY*COMMISSION_PCT
King	
Kochhar	
...	
Zlotkey	25200
Abel	39600
Taylor	20640
...	
Gietz	

20 rows selected.

Autres possibilités SELECT

Définir un alias de colonne :

Un alias de colonne:

- Renomme un titre de colonne
- Est utile avec des calculs
- Suit immédiatement le nom de la colonne (Il peut aussi être le mot-clé optionnel AS entre le nom de colonne et alias.)
- Nécessite guillemets s'il contient des espaces ou des caractères spéciaux ou si elle est sensible à la casse

Autres possibilités SELECT

Utiliser les alias :

```
SELECT last_name AS name , commission_pct AS comm
FROM   employees;
```

NAME	COMM
King	
Kochhar	
De Haan	

...

20 rows selected.

```
SELECT last_name AS "Name" , salary*12 AS "Annual
Salary"
FROM   employees;
```

Name	Annual Salary
King	288000
Kochhar	204000
De Haan	204000

...

20 rows selected.

Autres possibilités SELECT

Opérateur de concaténation :

- Liens colonnes ou chaînes de caractères à d'autres colonnes
- Est représenté par deux barres verticales (||)
- Crée une colonne résultante qui est une expression de caractères

```
SELECT last_name ||  job_id AS "Employees"
FROM   employees;
```

Employees
KingAD_PRES
KochharAD_VP
De HaanAD_VP
...

20 rows selected.

Autres possibilités SELECT

Literal Character Strings:

- A literal is a character, a number, or a date that is included in the **SELECT** statement.
- Date and character literal values must be enclosed by single quotation marks.
- Each character string is output once for each row returned.

Autres possibilités SELECT

Using Literal Character Strings:

```
SELECT last_name || ' is a '  || job_id
       AS "Employee Details"
FROM   employees;
```

Employee Details
King is a AD_PRES
Kochhar is a AD_VP
De Haan is a AD_VP
Hunold is a IT_PROG
Ernst is a IT_PROG
Lorentz is a IT_PROG
Mourgos is a ST_MAN
Rajs is a ST_CLERK

...

20 rows selected.

Autres possibilités SELECT

Dupliquer les lignes:

- L'affichage par défaut des requêtes est que toutes les lignes, y compris les lignes dupliquées.

```
SELECT department_id
FROM   employees;
```

DEPARTMENT_ID
90
90
90

...

20 rows selected.

```
SELECT DISTINCT department_id
FROM   employees;
```

DEPARTMENT_ID
10
20
50

...

8 rows selected.

Comment limiter les données

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90
103	Hunold	IT_PROG	60
104	Ernst	IT_PROG	60
107	Lorentz	IT_PROG	60
124	Mourgos	ST_MAN	50

...

20 rows selected.

"Récupérer tous les employés dans le département 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

Comment limiter les données

Limiter les lignes qui sont sélectionnés:

- Restreindre les lignes qui sont retournées en utilisant la clause WHERE:

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

- The **WHERE** clause follows the **FROM** clause.

Comment limiter les données

Utilisation de la **WHERE Clause:**

```
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

Comment limiter les données

Chaînes de caractères et les dates:

- Character strings and date values are enclosed by single quotation marks.
- Character values are case-sensitive, and date values are format-sensitive.
- The default date format is DD-MON-RR.

```
SELECT last_name, job_id, department_id
FROM   employees
WHERE  last_name = 'Whalen' ;
```

Conditions Comparaison

Créer des expressions avec le numéro et la date à l'aide des données des opérateurs arithmétiques.

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
BETWEEN ... AND ...	Between two values (inclusive)
IN (set)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

Conditions Comparaison

Utilisation de conditions de comparaison:

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

LAST_NAME	SALARY
Matos	2600
Vargas	2500

Conditions Comparaison

Utilisation de la **BETWEEN** Condition:

- Utilisation de la **BETWEEN** condition d'afficher les lignes basée sur une plage de valeurs:

```
SELECT last_name, salary
FROM   employees
WHERE  salary BETWEEN 2500 AND 3500 ;
```

Lower limit

Upper limit

LAST_NAME	SALARY
Rajs	3500
Davies	3100
Matos	2600
Vargas	2500

Conditions Comparaison

Utilisation de la `IN` Condition:

- Utilisez la condition d'appartenance IN pour tester les valeurs dans une liste:

```
SELECT employee_id, last_name, salary, manager_id
FROM   employees
WHERE  manager_id IN (100, 101, 201) ;
```

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.

Conditions Comparaison

Utilisation de la **`LIKE` Condition:**

- Utiliser la condition LIKE pour effectuer des recherches génériques des valeurs valides chaîne de recherche.
- Conditions de recherche peut contenir soit des caractères littéraux ou les numéros:
  - `%` denotes zero or many characters.
  - `_` denotes one character.

```
SELECT first_name
FROM   employees
WHERE  first_name LIKE 'S%' ;
```

Conditions Comparaison

Utilisation de la `LIKE` Condition:

- Vous pouvez combiner pattern-matching caractères:

```
SELECT first_name
FROM   employees
WHERE  first_name LIKE '_o%' ;
```

LAST_NAME
Kochhar
Lorentz
Mourgos

Conditions Comparaison

Utilisation de la `IS NULL` Condition:

- Test for nulls with the `IS NULL` operator.

```
SELECT last_name, manager_id
FROM   employees
WHERE  manager_id IS NULL ;
```

LAST_NAME	MANAGER_ID
King	

conditions logiques

Operator	Meaning
AND	Returns TRUE if *both* component conditions are true
OR	Returns TRUE if *either* component condition is true
NOT	Returns TRUE if the following condition is false

conditions logiques

Utilisation de la **AND** Operator:

- **AND** requires both conditions to be true:

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
AND    job_id LIKE '%MAN%';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
149	Zlotkey	SA_MAN	10500
201	Hartstein	MK_MAN	13000

conditions logiques

Utilisation de la OR Operator:

- **OR** requires either condition to be true:

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
OR     job_id LIKE '%MAN%';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
100	King	AD_PRES	24000
101	Kochhar	AD_VP	17000
102	De Haan	AD_VP	17000
124	Mourgos	ST_MAN	5800
149	Zlotkey	SA_MAN	10500
174	Abel	SA_REP	11000
201	Hartstein	MK_MAN	13000
205	Higgins	AC_MGR	12000

8 rows selected.

conditions logiques

Utilisation de la NOT Operator:

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id
       NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP');
```

LAST_NAME	JOB_ID
King	AD_PRES
Kochhar	AD_VP
De Haan	AD_VP
Mourgos	ST_MAN
Zlotkey	SA_MAN
Whalen	AD_ASST
Hartstein	MK_MAN
Fay	MK_REP
Higgins	AC_MGR
Gietz	AC_ACCOUNT

10 rows selected.

conditions logiques

Règles de priorité

Order	Operators
1	Arithmetic operators
2	Concatenation operator
3	Comparison conditions
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	Not equal to
7	NOT logical condition
8	AND logical condition
9	OR logical condition

- Vous pouvez utiliser des parenthèses pour modifier les règles de préséance.

conditions logiques

Rules of Precedence

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id = 'SA_REP'
OR     job_id = 'AD_PRES'
AND    salary > 15000;
```

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000
Abel	SA_REP	11000
Taylor	SA_REP	8600
Grant	SA_REP	7000

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  (job_id = 'SA_REP'
OR     job_id = 'AD_PRES')
AND    salary > 15000;
```

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000

Comment trier des données

Utilisation de la **ORDER BY Clause:**

- Tris les lignes avec **ORDER BY** :
  - **ASC**: ascending order, default
  - **DESC**: descending order
- The **ORDER BY** clause comes last in the **SELECT** statement:

```
SELECT   last_name, job_id, department_id, hire_date
FROM     employees
ORDER BY hire_date ;
```

LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
King	AD_PRES	90	17-JUN-87
Whalen	AD_ASST	10	17-SEP-87
Kochhar	AD_VP	90	21-SEP-89
Hunold	IT_PROG	60	03-JAN-90
Ernst	IT_PROG	60	21-MAY-91

...

20 rows selected.

Comment trier des données

Trier:

- Le tri dans l'ordre décroissant:

```
SELECT   last_name, job_id, department_id, hire_date
FROM     employees
ORDER BY hire_date DESC ;
```

Tri par alias de colonne:

```
SELECT   employee_id, last_name, salary*12 as
annsal
FROM     employees
ORDER BY salary*12 ;
```

Le tri sur plusieurs colonnes:

```
SELECT   last_name, department_id, salary
FROM     employees
ORDER BY department_id, salary DESC;
```

Labs: Basic Orders

Preview

- SELECT Statement
- Restricting and Sorting Data

SELECT Statement - Exercise

1. l'instruction SELECT suivante s'exécute correctement?

```
SELECT last_name, job_id, salary AS Sal
FROM employees;
```

2. l'instruction SELECT suivante s'exécute correctement?

```
SELECT *
FROM job_grades;
```

3. Il ya quatre erreurs de codage dans cette déclaration:

```
SELECT employee_id, last_name sal x 12 ANNUAL SALARY
FROM employees;
```

`SELECT` Statement - Correction

1. TRUE
2. TRUE
3. La table Employés ne contient pas une colonne appelée sal; l'opérateur de multiplication est "*", et non pas "x", le " **`ANNUAL SALARY`** " alias ne peut pas inclure d'espaces. L'alias doit être placé entre guillemets doubles; une virgule est manquante après la colonne LAST_NAME.

INSERT Statement

Data Manipulation Language

- Une instruction DML est exécutée lorsque vous:
  - Ajouter de nouvelles lignes à une table
  - Modifier les lignes existantes dans une table
  - Supprimer les lignes existantes d'une table
- Une transaction se compose d'une collection d'instructions DML qui forment une unité logique de travail.

Ajouter une nouvelle ligne à une Table

70	Public Relations	100	1700

New row

DEPARTMENTS

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

Insert new row into the **DEPARTMENTS** table

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
70	Public Relations	100	1700

INSERT Syntaxe des commandes

- Ajouter de nouvelles lignes à une table en utilisant l'instruction INSERT:

```
INSERT INTO table [(column [, column...])]
VALUES     (value [, value...]);
```

- Avec cette syntaxe, une seule ligne est insérée à la fois.

Insertion de nouvelles lignes

- Insérer une nouvelle ligne contenant des valeurs pour chaque colonne.
- Les valeurs de liste dans l'ordre par défaut des colonnes dans la table.
- En option, la liste des colonnes dans la clause INSERT.

```
INSERT INTO departments(department_id,
            department_name, manager_id,location_id)
VALUES     (70, 'Public Relations', 100, 1700);
```

```
1 row created.
```

- Joindre caractère et des valeurs de date entre guillemets simples.

Insertion de lignes avec des valeurs nulles

- Implicit method: Omettre la colonne de la liste des colonnes.

```
INSERT INTO departments (department_id,
            department_name  )
VALUES      (30, 'Purchasing');
```

```
1 row created.
```

- Explicit method: Spécifiez le mot clé NULL dans la clause VALUES.

```
INSERT INTO departments
VALUES      (100, 'Finance', NULL, NULL);
```

```
1 row created
```

Insertion de valeurs spéciales

- La fonction SYSDATE enregistre la date et l'heure actuelles.

```
INSERT INTO employees (employee_id,
                       first_name, last_name,
                       email, phone_number,
                       hire_date, job_id, salary,
                       commission_pct, manager_id,
                       department_id)
VALUES         (113,
                'Louis', 'Popp',
                'LPOPP', '515.124.4567',
                sysdate, 'AC_ACCOUNT', 6900,
                NULL, 205, 100);
```

```
1 row created.
```

Copie de lignes forment une autre table

- Écrivez votre instruction INSERT avec une sous-requête:

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
  SELECT employee_id, last_name, salary, commission_pct
  FROM employees
  WHERE job_id LIKE '%REP%';
```

```
4 rows created.
```

- Ne pas utiliser la clause VALUES.
- Faites correspondre le nombre de colonnes dans la clause INSERT à ceux de la sous-requête.

UPDATE Statement and DEFAULT

Modification des données dans une TABLE

- **EMPLOYEES**

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSION_PCT
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	

- Update rows in the **EMPLOYEES** table:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSION_PCT
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	70	
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	70	
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	70	
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	

Syntaxe des commandes UPDATE

- Modifier les lignes existantes avec l'instruction UPDATE:

```
 UPDATE table
 SET    column = value [, column = value, ...]
[WHERE  condition];
```

- Mise à jour plus d'une rangée à la fois (si nécessaire).

Mise à jour des lignes dans une Table

- Ligne spécifique ou les lignes sont modifiés si vous spécifiez la clause WHERE:

```
UPDATE employees
SET    department_id = 70
WHERE  employee_id = 113;
```

```
1 row updated.
```

- Toutes les lignes de la table sont modifiés si vous omettez la clause WHERE:

```
UPDATE copy_emp
SET    department_id = 110;
```

```
22 row updated.
```

DELETE Statement

Suppression de lignes 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	IT	103	1400

Delete a row from the **DEPARTMENTS** table:

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 existantes à partir d'un tableau en utilisant l'instruction DELETE:

```
 DELETE [FROM] table
[WHERE        condition];
```

Suppression de lignes d'une table

- Lignes spécifiques sont supprimés si vous spécifiez la clause WHERE:

```
DELETE FROM departments
WHERE       department_name = 'Finance';
```

```
1 row deleted.
```

- Toutes les lignes de la table sont supprimés si vous omettez la clause WHERE:

```
DELETE FROM copy_emp;
```

```
22 rows deleted.
```

Effacer des lignes basé sur une autre Table

- Utiliser les sous-requêtes dans les états DELETE pour supprimer des lignes dans une table basée sur des valeurs d'une autre table:

```
DELETE FROM employees
WHERE       department_id =
                 (SELECT department_id
                  FROM   departments
                  WHERE  department_name
                         LIKE '%Public%');
```

```
1 row deleted.
```