Bases de Données

Rappels : SQL – Langage d'interrogation

Composants

- SQL-LMD (langage de manipulation de données)
  - Interrogation : select
  - Manipulation : insert, update, delete
- SQL-LDD (langage de description de données)
  - Create, alter, drop
- SQL-LCD : langage de contrôle de données)
  - Grant, revoke
- + commit, rollback

Interrogation : select

- Permet d'interrogar une BD (requêtes sur les tables de la BD)
- Syntaxe générale :
- Select … from … where .. <autres clause>
  - où autres clause : group by, order by, ...

- Le résultat d'un SELECT est souvent un ensemle de lignes de (ou des) table(s) sur laquelle (lesquelles) porte le SELECT.

- Il existe plusieurs options pour la commande SELECT : tri, regroupement, projection, selection, sous-interrogation, conditions, ….

On utilise comme exemple une BDR contenant 2 tables EMP et DEPT et une table vide DUAL ayant un seul attribut.

EMP (empno, ename, job, mgr, hiredate, sal, comm, deptno)

DEPT (deptno, dname, loc)

EMP et DEPT ne sont pas vides.

DUAL (dummy) table vide à une colonne

SELECT de base

- **Select * from nom_tab;**
  - Lister toutes les lignes de la table
- **Select col1, col2, …. from nom_tab**
  - projection sur les attributs col1, col2, …

- ex. liste des infos sur les départements :
- **Select * from dept;**
- ex. liste des noms et salaires des employés :
- **Select ename, sal from emp;**

Résultat de la commande SQL : SELECT * FROM emp;

```
=> SELECT  *  FROM  emp;
 empno | ename  |    job          | mgr    |  hiredate     |   sal       |  comm    | deptno
-------+--------+-----------+------+------------+---------+---------+--------
  7369 | SMITH   | CLERK         | 7902  | 1980-12-17 |  800.00  |             |     20
  7499 | ALLEN   | SALESMAN  | 7698  | 1981-02-20 | 1600.00 |  300.00 |     30
  7521 | WARD    | SALESMAN  | 7698 | 1981-02-22 | 1250.00 |  500.00  |     30
  7566 | JONES   | MANAGER   | 7839  | 1981-04-02 | 2975.00 |             |     20
  7654 | MARTIN | SALESMAN  | 7698  | 1981-09-28 | 1250.00 | 1400.00 |     30
  7698 | BLAKE   | MANAGER    | 7839  | 1981-05-01 | 2850.00 |             |     30
  7782 | CLARK   | MANAGER    | 7839  | 1981-06-09 | 2450.00 |             |     10
  7788 | SCOTT   | ANALYST      | 7566 | 1982-12-09  | 3000.00 |             |     20
  7839 | KING       | PRESIDENT |          | 1981-11-17 | 5000.00 |             |     10
  7844 | TURNER | SALESMAN  | 7698  | 1981-09-08 | 1500.00 |    0.00    |     30
  7876 | ADAMS   | CLERK         | 7788   | 1983-01-12 | 1100.00 |             |     20
  7900 | JAMES   | CLERK         | 7698   | 1981-12-03 |  950.00  |             |     30
  7902 | FORD     | ANALYST     | 7566    | 1981-12-03 | 3000.00 |             |     20
  7934 | MILLER   | CLERK         | 7782   | 1982-01-23  | 1300.00 |             |     10
(14 lignes)


=>
```

Eliminer les doublons

Quand on projette sur un ou plusieurs attributs, les valeurs peuvent être dupliquées

=> utiliser DISTINCT pour supprimer les doublons

Liste des métiers (job) :

Select job from emp; - - valeur de job est dupliquée :

```
=> Select job from emp;

    job
-----------
 CLERK
 SALESMAN
 SALESMAN
 MANAGER
 SALESMAN
 MANAGER
 MANAGER
etc …

=> Select DISTINCT job from emp;

    job
-----------
 CLERK
 SALESMAN
 MANAGER
 PRESIDENT
 ANALYST
(5 lignes)

=>
```

Accès aux attributs dans un SELECT

- Parfois, on a besoin de suffixer les noms d'attributs par le nom de la table : tab.col

- ex.
- **Select ename, job from emp;**

  ou
- **Select emp.ename, emp.job from emp;**

Conditions de recherche : WHERE

- C'est l'opérateur de selection (ou restriction) de l'algèbre relationnelle
- Permet d'extraire les lignes d'une table satisfaisant une ou plusieurs conditions.
- **Select … from …where [not] cond1 [and | or] cond2 …;**
- ex. les employés du département numéro 20 :

  Select * from emp where deptno = 20;
- ex. employés dont le métier est 'OPERATEUR' :

  Select * from emp where job = 'CLERCK';

- Les employés dont le salaire est > 2000 et qui travaillent dans le département numéro 20 :
- **Select * from emp where**

  sal>2000 and deptno=20;
- Employés qui tavaillent dans les départements 10 ou 30 et qui sont 'MANAGER' :

  select * from emp where

  (deptno = 10 OR deptno = 20)

  AND job = 'MANAGER';

Autres exemples

Rmq : les conditions peuvent être reliées par les connecteurs logiques NOT, AND, OR (prio(not)>prio(and)>prio(or))

- Les informations sur les départements numéros 30 et 20 :

Select * from dept where deptno=30 or deptno=20;

Dans les conditions, les opérateurs sont : = , <>, < , > , <= et >=

Ces opérateurs s'appliquent aux types numérique, chaîne de caractères et dates.

La valeur NULL

- Correspond à l'absence de valeur pour un attribut
- Est différent de la valeur 0 (zéro)
- Se teste avec : attibut IS [NOT] NULL
- ex. les employés dont le commission n'est pas renseignée :

  select * from emp where comm is null;

- ex. Les employés qui ont un chef :

  select * from emp where mgr is not null;

```
=> SELECT * FROM emp WHERE comm IS NOT NULL;


empno | ename     |   job           | mgr   |  hiredate      |   sal      |  comm    | deptno
-------+--------+----------+------+-----------+---------+---------+--------
  7499 | ALLEN     | SALESMAN | 7698  | 1981-02-20 | 1600.00 |  300.00  |     30
  7521 | WARD     | SALESMAN  | 7698  | 1981-02-22 | 1250.00 |  500.00  |     30
  7654 | MARTIN  | SALESMAN  | 7698  | 1981-09-28 | 1250.00 | 1400.00 |     30
  7844 | TURNER | SALESMAN  | 7698  | 1981-09-08 | 1500.00 |       0.00 |     30
(4 lignes)

=>
```

Intervalle de valeurs : [not] between

- ex. les noms et salaires des employés dont le salaire est compris entre 1000 et 2000 (bornes incluses) :

  select ename, sal from emp where

  sal between 1000 and 2000;

- Equivalent à :

  select ename, sal from emp where

  sal >= 1000 and sal <= 2000;

- Rmq : on peut comparer avec NOT BETWEEN
- ex. les employés dont le salaire est <= 1250 et > à 2500 :

```
select * from emp where
                    sal not between 1250 and 2500
                or  sal = 1250;
```

liste de valeurs : [not] in

- ex. les noms et salaires des employés qui travaillent dans l'un des départements (10, 20, ou 30 :

  select ename, sal from emp where

  deptno in (10, 20, 30);

- Equivalent à :

  select ename, sal from emp where

  deptno=10 or deptno=20 or deptno=30;

Les employés qui ne sont ni du département 20, ni du département 30 :

Select * from emp where deptno NOT IN (20, 30)

Renommer les colonnes ou les tables

EX: 1. Lister les noms des départements :

```
select dname AS «NOM DEPARTEMENT»
from dept;
```

/ nom est composé => mettre entre guillemets */*

2. Lister les noms et salaires des employés :

```
select ename AS NOM, sal AS SALAIRE
from emp;
```

```
=> SELECT dname   "NOM DEPARTEMENT", deptno AS numero FROM dept;

NOM DEPARTEMENT   | numero
----------------+--------
 ACCOUNTING           |     10
 RESEARCH             |     20
 SALES                |     30
 OPERATIONS           |     40
(4 lignes)

=>
```

Opérateurs arithmétiques

+, -, *, / : combinés pour faire des expressions.

EX : Afficher les employés avec leur salaire total (salaire + commission quand elle existe).

Select ename, sal, comm, sal+comm as «salaire total» where comm is not null;

Tri des résultats

Les résultats d ’une sélection peuvent être triés, sur une ou plusieurs colonnes, par ordre croissant (par défaut) ou décroissant.

Syntaxe :

Select nom_col [,nom_col ...] from nom_tab

where …..

order by nom_col [asc|desc] [,nom_col [asc|desc] ...] ;

Exemples

Employés dont le salaire est < 1500, triés par ordre alphabétique croissant :

Select * from emp where sal < 1500 ORDER BY ename;

/ Ordre croissant par défaut */*

Employés par ordre alphabétique du métier, et par ordre alphabétique inverse du nom :

Select * from emp where sal < 1500
ORDER BY job, ename DESC;

Correspondances de chaînes : LIKE , NOT LIKE

Les caractères jokers :

% : toute chaîne de caractère (même vide)

_ : un et un seul caractère

Exemples

Liste des employés dont le nom commence par 'a' et se termine par 'e' :

```
Select * from emp
      where ename like 'a%e' or ename like 'A%E';
```

On peur utiliser des fonctions (transforment le mot en majuscules ou minuscules)

```
Select * from emp
   where ename like upper('a%e')
         or ename like lower( 'A%E');
```

Exemples

Liste des employés dont le nom commence possède exactement 8 caractères et se termine par 'e' :

```
Select * from emp
   where ename like upper('_______e') ;
```

Exemples

Noms des employés se terminant par 's' :

SELECT * FROM emp WHERE ename LIKE '%s';

Nom des répartements commençant par 't' ou 'T' et se terminant par 's' :

SELECT * FROM dept

WHERE dname LIKE 't%s' OR dname LIKE 'T%s';

Nom des répartements commençant par 't' ou 'T' et se terminant par 's' et ayant 5 caractères exactement:

SELECT * FROM dept

WHERE dname LIKE 't___s' OR dname LIKE 'T___s';

Exemples

Noms des employés contenant la chaîne 'abc' au début du nom :

SELECT * FROM emp WHERE ename LIKE 'abc%';

Nom des répartements commençant par 'B' et ayant 7 caractères :

SELECT * FROM dept

WHERE dname LIKE upper('b_____');

Noms des employés ne contenant pas la lettre 'e' :

SELECT * FROM emp WHERE ename NOT LIKE '%e%';

```
=> SELECT * FROM emp WHERE ename like upper('s%H');

 empno | ename |  job       | mgr   |  hiredate      |  sal      | comm | deptno
-------+-------+-------+------+------------+--------+------+--------
  7369   | SMITH  | CLERK | 7902 | 1980-12-17 | 800.00 |           |     20
(1 ligne)



=> SELECT * FROM dept WHERE loc like upper('c%');

 deptno | dname |   loc
--------+-------+---------
     30    | SALES | CHICAGO
(1 ligne)

=>
```

La Jointure

- Association de lignes de plusieurs tables en fonction d ’un critère (de jointure).

Select … from nom_tab [nom_alias]

where <critère de jointure>;

EX. Afficher les noms des employés (table emp) avec les noms de leur département (table dept) :

SELECT ename, dname FROM emp, dept

WHERE emp.deptno = dept.deptno ;

ou bien (standard SQL)

SELECT ename, dname FROM emp JOIN dept ON *emp.deptno = dept.deptno* ;

Rmq : colonne commune *deptno*

- Rmq :
-

1. le nom de colonne étant identique => on le préfixe par le nom de la table.

2. Auto-jointure : jointure de lignes différentes de la même table

Résultat de la commande de jointure précédente :

=> Select ename, dname from emp, dept where emp.deptno = dept.deptno ;

```
ename     |   dname
--------+------------
 SMITH    | RESEARCH
 ALLEN    | SALES
 WARD     | SALES
 JONES    | RESEARCH
 MARTIN   | SALES
 BLAKE    | SALES
 CLARK    | ACCOUNTING
 SCOTT    | RESEARCH
 KING     | ACCOUNTING
 TURNER | SALES
 ADAMS    | RESEARCH
 JAMES    | SALES
 FORD     | RESEARCH
 MILLER   | ACCOUNTING
(14 lignes)
```


=>

L'Autojointure

- Lister les employés avec leur manager : le critère d ’auto-jointure est réalisé en associant EMPNO et MGR (qui est aussi un numéro d ’employé, c-à-d du chef) :

select emp.empno, emp.ename, empbis.empno, empbis.ename

from emp, emp empbis

where emp.mgr = empbis.empno;

- Pour effectuer l ’auto-jointure, il convient de citer 2 fois la même table en utilisant un alias, car on traite 2 tables identiques : les mêmes noms de colonnes apparaissent dans les deux tables.

Résultat de la commande d'auto-jointure :

```
=> SELECT emp.empno, emp.ename  AS "EMPLOYE",  empbis.empno ,
                empbis.ename   AS "SON CHEF"
      FROM emp,  emp     empbis
      WHERE emp.mgr = empbis.empno;

 Empno  | EMPLOYE | empno | SON CHEF
-------+---------+-------+----------
  7369    | SMITH    |  7902 | FORD
  7499    | ALLEN    |  7698 | BLAKE
  7521    | WARD    |  7698  | BLAKE
  7566    | JONES    |  7839 | KING
  7654    | MARTIN  |  7698 | BLAKE
  7698    | BLAKE    |  7839  | KING
  7782    | CLARK    |  7839  | KING
  7788    | SCOTT    |  7566  | JONES
  7844    | TURNER  |  7698 | BLAKE
  7876    | ADAMS    |  7788 | SCOTT
  7900    | JAMES     |  7698 | BLAKE
  7902    | FORD       |  7566 | JONES
  7934    | MILLER    |  7782 | CLARK
(13 lignes)

=>
```

Sous-interrogation

- **Select … from …**

  where nom_col <opérateur>

  (Select … From … Where…);

- EX: Afficher les employés qui sont dans le même département que 'ALLEN' : d'abord recherche du département de ALLEN, puis connaissant son numéro (30), on cherche les employés qui y travaillent.

Select * From emp Where

deptno = (Select deptno from emp

where ename like'ALLEN');

Sous-interrogation rapportant plusieurs lignes : IN, ANY, ALL

Le résultat d ’une sous-interrogation peut comporter plusieurs lignes. Dans ce cas les opérateurs =, <, > … ne conviennent plus. L ’égalité sera traitée par l ’opérateur IN (un ‘=‘ par rapport à une suite de valeurs),

et Les inégalités seront traitées par l ’opérateur ANY et ALL.

Opérateur IN

Exemple : Les employés travaillant dans le même département que l ’un des employés dépendant du président.

```
Select * from emp
   Where deptno in
        (select deptno
         from emp
         Where mgr = (Select empno from emp
                      Where job=‘ PRESIDENT ’
                      )
        );
```

Opérateur ANY

Le résultat de la comparaison est VRAI, s ’il l ’est pour au moins un élément de l ’ensemble (de la sous-interrogation)

EX: Afficher les employés ayant un salaire supérieur à celui de l ’un des employés travaillant dans le département 10

```
Select * from emp
        where sal > ANY (select sal
                         from emp
                         where deptno=10
                         );
```

Opérateur ALL

Le résultat de la comparaison est VRAI, s ’il l ’est pour tous les éléments de l ’ensemble (sous-interrogation)

EX: Afficher les employés ayant un salaire supérieur à celui des employés travaillant dans le département 20.

```
Select * from emp
        where sal > ALL (select sal
                         from emp
                         where deptno=20
                        );
```

Extraction de lignes si le résultat de la sous-interrogation comporte au moins une ligne : [NOT] EXIST

Select … From …. Where
[NOT] EXISTS
(Select … From … Where …);

Ex1: lister les employés s'il y en un parmi eux qui a une commission > 1000;

Select * from emp where
EXISTS
(select * from emp where comm > 1000);

Extraction de lignes si le résultat de la sous-interrogation comporte au moins une ligne : [NOT] EXIST

- **Ex2** : lister les employés s'il n 'y a aucun parmi eux qui a une commission > 1000

Select * from emp where
NOT EXISTS
(select * from emp where comm > 1000);

Fonctions de groupes

→ applique des fonctions sur un ensemble de données :

- Min : retourne le minimum
- Max : retourne le maximum
- Avg : retourne la moyenne arithmétique
- Count : retourne le nombre (de lignes)
- Sum : retourne le total (la somme)

On utilise la clause GROUP BY, pour indiquer au moteur SQL que la fonction porte sur une sélection de données (groupe de lignes).

Exemple

Afficher la sommes des salaires par département :

```
Select deptno, sum(sal) as “somme salaires”
from emp
group by deptno;
```

Exemple

Ex:

1. Afficher le total des salaires :

   Select sum(sal) as total_sal from emp;

2. Afficher le total et la moyenne des commissions :

   Select sum(comm) as «somme_comm,
   avg(comm) as moy_comm
   from emp;

•**Rmq** : la valeur NULL n ’entre pas dans les différents calculs.

Ex : la moyenne de la colonne COMM est calculée en cumulant le nombre de commissions divisée par le nombre de valeurs non NULL.

Ex. Afficher le nombre de commissions à NULL :

Select count(comm) from emp;

- **Rmq** : Il est impossible d ’utiliser à la fois une projection et une fonction de groupe : *Select ename, sum(sal) from emp;* **SERA REJETEE.**

- **Rmq** : COUNT (*) : nombre de lignes non NULL.

Conditions de selection sur les groupes : HAVING

Créer des critères de sélection qui portent sur un ensemble de données :

→ associer la clause GROUP BY à la clause HAVING (ou NOT HAVING).

EX. Les départements qui ont des salaires > 5000 :

```
Select deptno, sum(sal) as "somme salaires"
    from emp
    group by deptno having sum(sal) > 5000;
```

Opérateurs sur plusieurs tables
union, différence, intersection

Soit 2 tables de même structure et de contenus différents : depot1 et deptot2:

Structure :

depot1(numprod, design, conditionnemt, stock, rayon, pu, tva)

depot2(numprod, design, conditionnemt, stock, rayon, pu, tva)

L'union : UNION

En faisant l'union de 2 SELECT, on obtient les lignes de l'une ou de l'autre table (Les doublons sont éliminés)

Ex : Lister les produits en stocks dans lun ou l'autre des 2 dépôts :

Select numpro, design, pu, from depot1
UNION
Select numpro, design, pu, from depot2;

Rmq : les noms de colonnes du résultat sont les noms figurant dans le 1er SELECT

La différence :
MINUS (Oracle), EXCEPT (Posgres)

En faisant la différence entre 2 SELECT, on obtient les lignes se trouvant dans la 1ère table mais pas dans la 2ème.

Ex : Lister les produits en stocks seulement dans le 1er dépôt :

Select numpro, design, pu, from depot1
EXCEPT
Select numpro, design, pu, from depot2;

Rmq : les noms de colonnes du résultat sont les noms figurant dans le 1er SELECT

Intersection : INTERSECT

En faisant l'intersection de 2 SELECT, on obtient les lignes communes aux 2 tables

Ex : Lister les produits en stocks dans les 2 dépôts :

Select numpro, design, pu, from depot1
INTERSECT
Select numpro, design, pu, from depot2;

Rmq : les noms de colonnes du résultat sont les noms figurant dans le 1er SELECT

Les types de données : Les DECIMAUX PRECIS

Contrairement au type « float », le type NUMERIC (ou DECIMAL) permet de stocker la valeur exacte d'un réel.

Syntaxe :

NUMERIC (long, dec) ou DECIMAL (long, dec)
long : nombre maximum de chiffres à stocker
dec : nombre de chiffres après la virgule

Exemple NUMERIC (9, 3)

permet de stocker un nombre à 9 chiffres dont 3 après la virgule.

Les types de données : Les CHAINES DE CARACTERES

- chaîne fixe : CHAR ou CHARACTER
  CHAR (n) ou CHARACTER (n)
  => Stockage de n caractères

- chaîne variable : VARCHAR ou CHARACTER VARYING
  VARCHAR (n) ou CHARACTER VARYING (n)
  => Stockage de n caractères au maximum

- chaîne illimitée : TEXT
  TEXT, stockage d'un nombre illimité de caractères
  Entre apostrophes (ou quottes)

Exemples

• En déclarant un type CHAR(8), la chaîne «lundi» (5 caractères) et la chaîne «vendredi» (8 caractères) occuperont chacune 8 caractères.

• Les données stockées dans une colonne de type VARCHAR occuperont exactement l'espace nécessaire à leur stockage.

• VARCHAR2 (sensiblement équivalent au VARCHAR)
Est spécifique au SGBD Oracle.

Exemples

BOOLEAN

Le type booléen (BOOLEAN ou BOOL) : ce type peut contenir

pour vrai : true, t, yes, y ou 1

Pour faux : false, f, no, n ou 0

Expressions et Fonctions

Une expression : combinaison de variables (contenu d ’une colonne), de constantes et d ’autres expressions à l ’aide des opérateurs : +, -, *, /

Une fonction : routine ayant des arguments et ramenant un résultat (un argument peut être une expression)

Il y a 3 types d ’expressions : arithmétiques, chaînes de caractères et date.

A chaque type ==> il y a des opérateurs spécifiques.

Fonction NVL (Oracle), Coalesce (Postgres)

NVL : NULL VALUE (Oracle) : permet de remplacer une valeur NULL par une valeur significative.

Syntaxe : NVL(exp1, exp2) :

→ renvoie la valeur de exp1, si exp1 non NULL et de exp2 sinon.

NVL (comm, 0) = comm, si comm non NULL
= 0 sinon

Ex: **Select ename, sal, comm, sal+NVL(comm, 0) from emp;**

Coalesce : PostrgreSQL

COALESCE(arg1, arg2, …, argn) où argi est une colonne d ’une table.

La valeur de la fonction est la première valeur NOT NULL.

EX. Select ename, sal, comm, sal+COALESCE(comm, 0)
from emp;

Concaténation

met bout à bout 2 chaînes de caractères.

EX. Afficher les noms des employés concaténé à leur métier.

Select ename||‘ - ’||job AS «NOM - METIER» from emp;

ARRONDI : ROUND et TRUNC

ROUND (n, m) : permet d'arrondir le nombre n avec m décimales.

EX. Noms et salaires horaires des employés du département 20., en renommant les colonnes.

Select ename as «nom,

ROUND(sal / 150, 2) as « sal aire horaire »

from emp where deptno = 20; */* 150 heure /mois */*

TRUNC (n , m) : permet de tronquer le nombre n à m décimaes.

Opérateur de choix : CASE

CASE WHEN cond1 THEN arg1

WHEN cond1 THEN arg2 ….

ELSE arg-n

END

EX. Dans emp, afficher le salaire seul si deptn = 10, le salaire plein si deptno = 20, le numéro de département sinon.

```
SELECT CASE WHEN deptno=10 THEN sal
            WHEN deptno = 20 THEN sal+coalesce(comm,0)
            ELSE deptno
            END
FROM emp;
```

Fonctions sur Chaînes de caractères

SUBSTR (chaine, pos, n) : extrait de la chaîne ‘chaine’, à partir de la position pos, n caractères (ou le reste de la chaîne, si n absent)

UPPER (chaine) ou LOWER (chaîne) : transforme en majuscules (ou minuscules) la chaîne

LENGTH (chaine) : donne le nombre de caractères de la chaîne

Les fonctions de DATES

TO_CHAR (col, format) col est une colonne d'une table, ou une valeur de type DATE.

⇒ Met col (correspondant à une date) selon le format : YYYY année, YY : 2 chiffres de l'année, MM : mois, HH24 : heure sur 24 heures, …

⇒ *EX. Transformer le date d'embauche au format 'DD/MM/YYYY'.*

⇒ *Select TO_CHAR(hiredate, 'DD-MM-YYYY') as date from emp;*

⇒ MONTH , DAY : mois (jour) en lettres

⇒ Par défaut : la date : YYYY-MM-DD

Autres fonctions DATE

CURRENT_DATE : date du jour

(Sous Oracle : SYSDATE)

Commandes de modifications des données

Principales commandes :

- INSERT pour ajouter les données

- UPDATE pour modifier les données

- DELETE pour supprimer les données

INSERT

INSERT INTO nom_tab [(nom_col,...)]
    VALUES ({expr | DEFAULT},...)

Si les colonnes sont présentes, leur nombre doit correspondre au nombre d'expressions (ou valeurs) .

Ex1: Insérer un nouveau département : 50, ‘EDUCATION’, ‘MIAMI’ :

Insert INTO dept VALUES (50, ‘EDUCATION’, ‘DENVER’);

Rmq : On n ’a pas à citer les colonnes quand on insère des valeurs pour toutes les colonnes !

Ex2:Insérer un employé en ne connaissant que le numéro, le nom, le job et le numéro de département :

Insert into EMP (empno, ename, job, deptno)
VALUES (7950, ‘ JOHN ’, ‘ TRAINER ’, 50);
/ on doit citer les colonnes */*

Mise à jour : UPDATE

```
UPDATE nom_tab
    SET nom_col1=exp1 [,nom_col2=exp2 ...]
    [WHERE conditions] [LIMIT nb_lignes]
```

- SET permet d'attribuer une nouvelle valeur à un attribut.

- On peut mettre à jour plusieurs attributs en même temps.

- WHERE permet de préciser quelles données on veut mettre à jour.

- Si clause WHERE absente, alors toutes les données de la table sont mises à jour.

- LIMIT permet de limiter le nombre de lignes à modifier.

Ex: Remplacer dans le département 30, la localité ‘CHICAGO’ par ‘LOS ANGELES’ :

UPDATE dept
SET loc = ‘LOS ANGELES’
Where deptno=30;

Ex. Augmenter la commission de 10% pour tous les employés :

Update emp SET comm = comm * 1.1;

Ex. Mettre le numéro de département à 10 et leur salaire augmenté de 100 pour les employés du département 20 :

Update emp SET deptno=20, sal=sal+100
where deptno=10;

Suppression : DELETE

DELETE FROM nom_tab
    [WHERE conditions] [LIMIT nb_lignes]

- WHERE permet de préciser quelles données on veut supprimer.
- Si clause WHERE absente, alors toutes les données de la table sont supprimées (sauf si clause LIMIT).
- LIMIT permet de limiter le nombre de lignes à supprimer.

EX1. Supprimer toutes les lignes de la table BONUS :

delete from bonus;

EX2. Supprimer les employés ayant une commission non NULL :

delete from emp where comm is not null;