

**LA SUITE  
MISE EN ŒUVRE  
PL/SQL  $\leftarrow \rightarrow$  BD**

**SCHEMA DE LA BASE  
D'EXEMPLES**

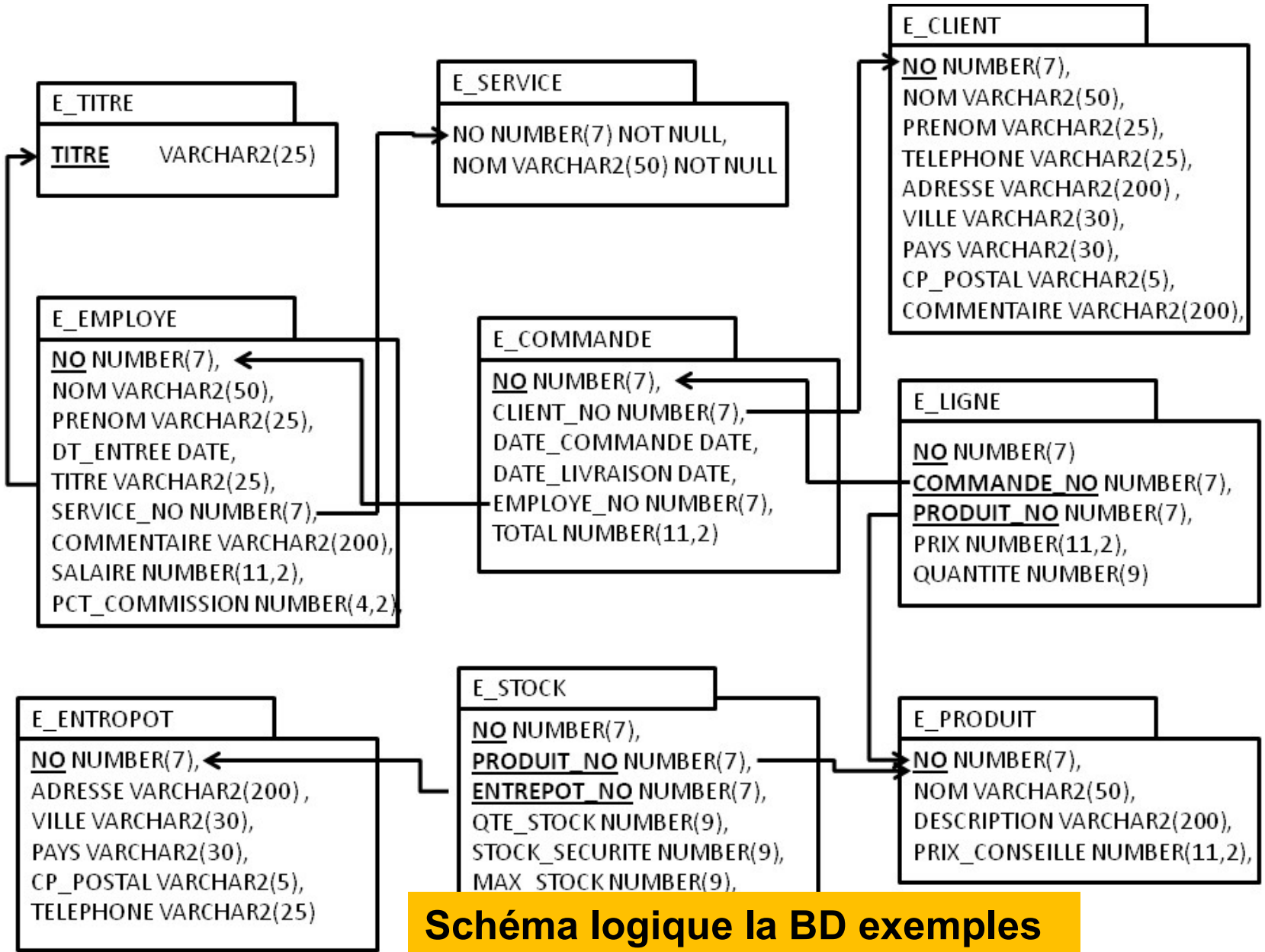


Schéma logique la BD exemples

**INSERT**

# BASES DE DONNEES RELATIONNELLES

Même syntaxe que SQL

**INSERT INTO NOM\_TABLE [(col1, col2, ...,coln)] VALUES (val1, val2, ..., valn);**

Exemple 1:

SQL> DECLARE

2

3 BEGIN

4 **INSERT INTO E\_CLIENT (NO, NOM, PRENOM) VALUES(99,'Filali','Said');**

5 **COMMIT;**

6 END;

7 /

Procédure PL/SQL terminée avec succès.

Exemple 2:

DECLARE

ID **E\_CLIENT.NO%TYPE;**

NOM\_EMP **E\_CLIENT.NOM%TYPE;**

PRE\_EMP **E\_CLIENT.PRENOM%TYPE;**

BEGIN

ID:=99; NOM\_EMP:='Filali'; PRE\_EMP:='Said';

**INSERT INTO E\_CLIENT (NO, NOM, PRENOM) VALUES(ID,NOM\_EMP, PRE\_EMP);**

**COMMIT;**

END;

# BASES DE DONNEES RELATIONNELLES

```
CREATE SEQUENCE NOM_SEQ INCREMENT BY PAS_INC START WITH VAL_DEPART  
MAXVALUE VAL_MAX;
```

## Exemple 3

```
SQL>  
SQL> CREATE SEQUENCE SEQ_NO_CL INCREMENT BY 1 START WITH 200 MAXVALUE  
99999;
```

Séquence créée.

```
SQL>  
SQL> DECLARE  
2  
3 BEGIN  
4         INSERT INTO E_CLIENT (NO, NOM, PRENOM)  
VALUES(SEQ_NO_CL.NEXTVAL, 'Filali', 'Said');  
5         COMMIT;  
6 END;  
7 /
```

Procédure PL/SQL terminée avec succès.

**UPDATE**

# BASES DE DONNEES RELATIONNELLES

Même syntaxe que SQL: UPDATE NOM\_TABLE SET .... WHERE ....

```
SQL> DECLARE
2      VILLE_AVM          E_CLIENT.VILLE%TYPE;
3      VILLE_APM          E_CLIENT.VILLE%TYPE;
4 BEGIN
5      SELECT VILLE INTO VILLE_AVM FROM E_CLIENT WHERE NO=1;
6      DBMS_OUTPUT.PUT_LINE ('LA VILLE AVANT MODIFICATION EST :      '|| VILLE_AVM );
7
8      UPDATE E_CLIENT
9      SET VILLE='tantan'
10     WHERE NO=1;
11
12     COMMIT;
13
14     SELECT VILLE INTO VILLE_APM FROM E_CLIENT WHERE NO=1;
15     DBMS_OUTPUT.PUT_LINE ('LA VILLE APRES MODIFICATION EST :      '|| VILLE_APM );
16 END;
17 /
LA VILLE AVANT MODIFICATION EST :      Rabat
LA VILLE APRES MODIFICATION EST :      tantan
```

Procédure PL/SQL terminée avec succès.



**DELETE**

Même syntaxe que SQL: DELETE NOM\_TABLE WHERE ....

```
SQL>
SQL> DECLARE
2 BEGIN
3     DELETE E_CLIENT WHERE NO=9;
4
5     COMMIT;
6 END;
7 /
```

Procédure PL/SQL terminée avec succès.

```
SQL>
```

# **TRAITEMENT DE PLUSIEURS TUPELS**

# BASES DE DONNEES RELATIONNELLES

## Exemple 7:

```
SQL> DECLARE
2     NOM_EMP    VARCHAR2(20);
3 BEGIN
4     SELECT NOM INTO NOM_EMP
5     FROM E_CLIENT
6     WHERE NO=99;
7 END;
8 /
```

```
DECLARE
*
```

ERREUR à la ligne 1 :

ORA-01403: aucune donnée trouvée

ORA-06512: à ligne 4

## Exemple 8:

```
SQL> DECLARE
2     NOM_EMP    VARCHAR2(20);
3 BEGIN
4     SELECT NOM INTO NOM_EMP
5     FROM E_CLIENT
6     WHERE NO=1 OR NO=2;
7 END;
8 /
```

```
DECLARE
*
```

ERREUR à la ligne 1 :

ORA-01422: l'extraction exacte ramène plus que le nombre de lignes demandé

ORA-06512: à ligne 4

CURSEUR

SOLUTION

# **LES CURSEURS**

**Un curseur est une zone mémoire de taille fixe, utilisée par le moteur SQL pour analyser et interpréter un ordre SQL**

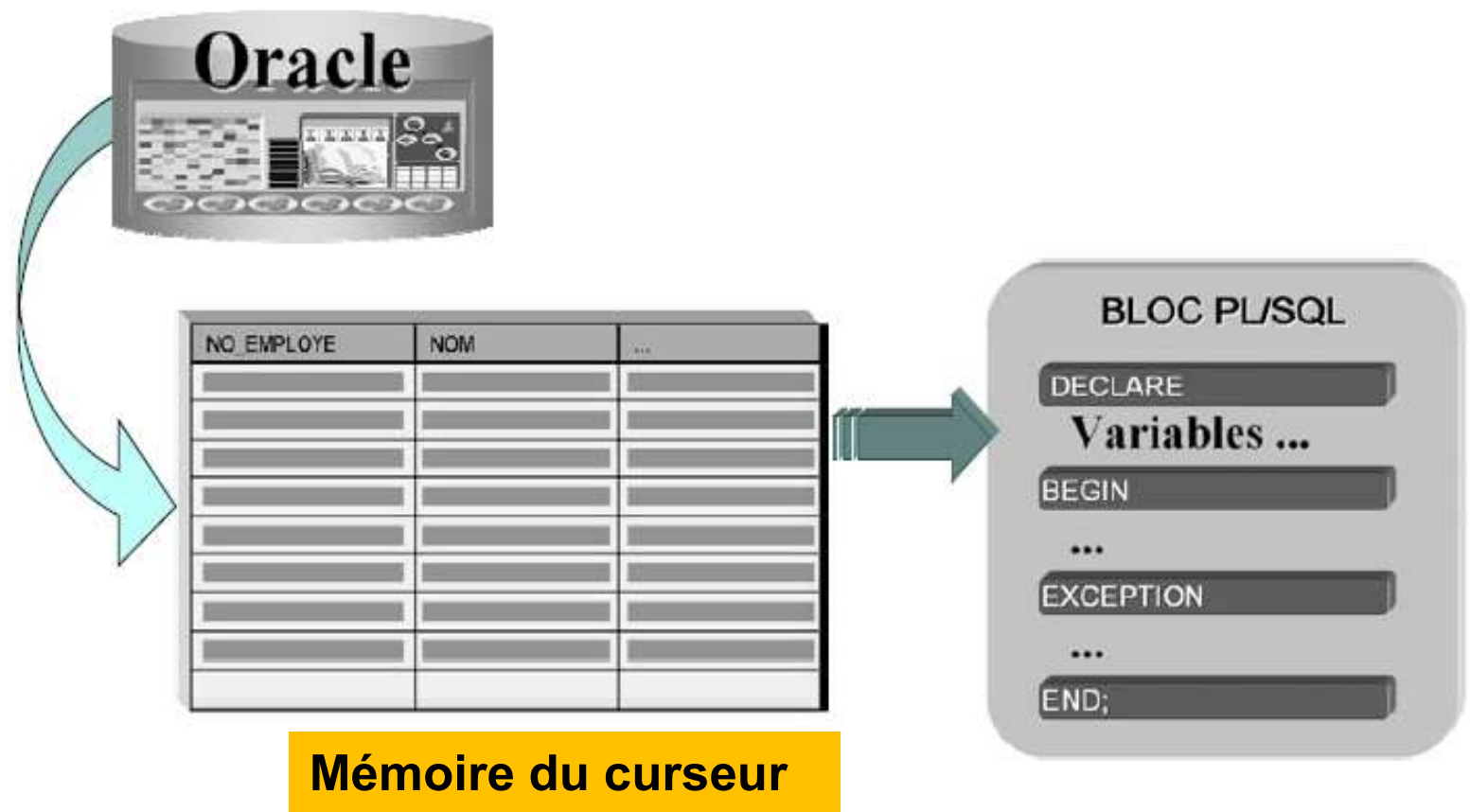
**Curseur implicite: dans un ordre SQL et géré par le compilateur PL/SQL**

**Un curseur explicite est géré par l'utilisateur pour traiter un ordre Select qui ramène plusieurs tuples**

**Remarque**

**EN TERME D'EXECUTION, LES CURSEURS IMPLICITES SONT PLUS RAPIDES QUE LES CURSEURS EXPLICITES**

**LES  
CURSEURS IMPLICITES**

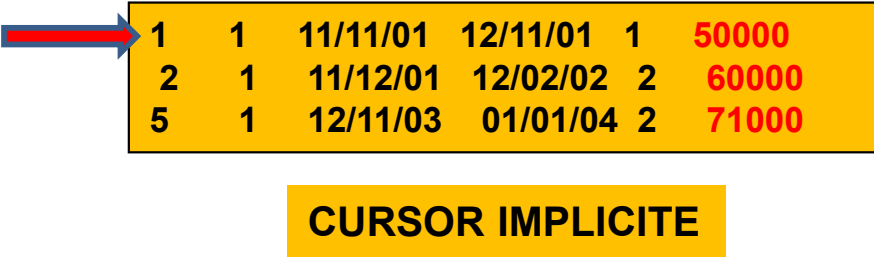




## Curseur implicite: Exemple 1:

```
SQL> SELECT * FROM E_COMMANDE;
NO CLIENT_NO DATE_COM DATE_LIV EMPLOYE_NO    TOTAL
-----
1      1      11/11/01 12/11/01      1      50000
2      1      11/12/01 12/02/02      2      60000
3      2      10/11/01 12/11/01      1      40500
4      3      11/11/02 12/11/02      1      50030
5      1      12/11/03 01/01/04      2      71000
6      4      15/11/03 19/11/03      3      80100
7      6      11/11/03 01/01/04      4     410000
8      6      02/01/04 12/01/04      1      55000
9      8      10/01/04 15/01/04      5      69000
10     13     11/01/04 16/01/04      1     150800

10 ligne(s) sélectionnée(s).
```



1	1	11/11/01	12/11/01	1	50000
2	1	11/12/01	12/02/02	2	60000
5	1	12/11/03	01/01/04	2	71000

**CURSOR IMPLICITE**

```
SQL> DECLARE
2      SOMME E_COMMANDE.TOTAL%TYPE;
3 BEGIN
4      SELECT SUM(TOTAL) INTO SOMME FROM E_COMMANDE WHERE CLIENT_NO=1;
5      DBMS_OUTPUT.PUT_LINE('la somme est : '||SOMME);
6 END;
7 /

la somme est : 181000
Procédure PL/SQL terminée avec succès.
```

## Curseur implicite: Exemple 2:

```
SQL> DECLARE
2         NBR_LIGNE_DELETE          NUMBER(3);
3 BEGIN
4         DELETE E_COMMANDE WHERE CLIENT_NO=1;
5         NBR_LIGNE_DELETE:=SQL%ROWCOUNT;
6         DBMS_OUTPUT.PUT_LINE('Nombre de ligne détruites : '||NBR_LIGNE_DELETE);
7 END;
8 /
```

Nombre de ligne détruites : 3

Procédure PL/SQL terminée avec succès.

## Curseur implicite: Exemple 3:

```
SQL> DECLARE
2          NBR_LIGNE          NUMBER(3):=0;
3          SOMME              E_COMMANDE.TOTAL%TYPE:=0;
4 BEGIN
5     FOR LIGNE IN (SELECT * FROM E_COMMANDE WHERE CLIENT_NO=1) LOOP
6         DBMS_OUTPUT.PUT_LINE('NUM COMMANDE: '|| LIGNE.NO||' NUM CLIENT: '||LIGNE.CLIENT_NO);
7         SOMME:=SOMME+LIGNE.TOTAL;
8         NBR_LIGNE := NBR_LIGNE +1;
9     END LOOP;
10        DBMS_OUTPUT.PUT_LINE('LA somme est : '||SOMME);
11        DBMS_OUTPUT.PUT_LINE('Nombre de lignes est : '||NBR_LIGNE);
12 END;
13 /
```

### Exécution :

```
NUM COMMANDE: 1  NUM CLIENT: 1
NUM COMMANDE: 2  NUM CLIENT: 1
NUM COMMANDE: 5  NUM CLIENT: 1
LA somme est : 181000
Nombre de lignes est : 3
```

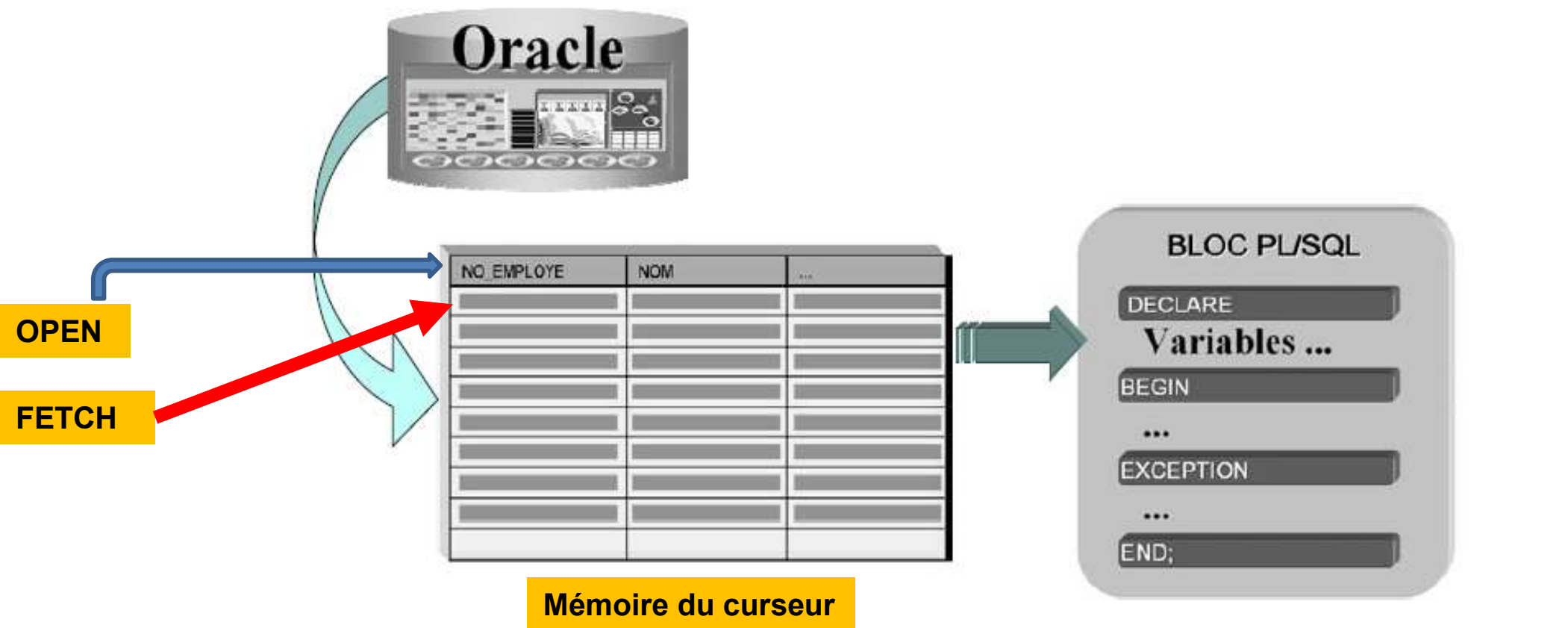
Procédure PL/SQL terminée avec succès.

**LES  
CURSEURS EXPLICITES**

**CURSOR** nom\_curseur [(parametres) ] **IS** requête\_select;

<b>OPEN</b> nom_curseur	OUVRE LE CURSEUR. Aucune EXCEPTION si la requête ne ramène aucune ligne.
<b>FETCH</b> nom_curseur <b>INTO</b> listes_variables nom_Record	Positionnement sur la ligne suivante et chargement des valeurs dans les variables INTO ou record
<b>CLOSE</b> nom_curseur	Fermeture du curseur
Nom_curseur% <b>ISOPEN</b>	Retourne TRUE si le curseur est ouvert sinon FALSE
Nom_curseur% <b>NOTFOUND</b>	Retourne FALSE si le dernier FETCH n'a pas renvoyé de ligne
Nom_curseur% <b>FOUND</b>	Retourne TRUE si le dernier FETCH a renvoyé une ligne
Nom_curseur% <b>ROWCOUNT</b>	Retourne le nombre totale de lignes traités jusqu'à maintenant

FONCTIONS ASSOCIEES AU CURSEUR



## Curseur explicite: Exemple 4:

```
SQL> DECLARE
2  CURSOR CLIENT_RABAT IS
3  SELECT NO,NOM FROM E_CLIENT
4  WHERE VILLE='Rabat';
5
6  NOMCL E_CLIENT.NOM%TYPE;
7  NOCL E_CLIENT.NO%TYPE;
8
9  BEGIN
10 OPEN CLIENT_RABAT;
11
12 DBMS_OUTPUT.PUT_LINE('Nbr lignes traitées : '|| CLIENT_RABAT%ROWCOUNT);
13 FETCH CLIENT_RABAT INTO NOCL, NOMCL;
14 WHILE (CLIENT_RABAT%FOUND) LOOP
15     DBMS_OUTPUT.PUT_LINE('Client no : '|| NOCL ||' de nom : ' || NOMCL);
16     DBMS_OUTPUT.PUT_LINE('Nbr lignes traitées : '|| CLIENT_RABAT%ROWCOUNT);
17     FETCH CLIENT_RABAT INTO NOCL, NOMCL;
18 END LOOP;
19
20 CLOSE CLIENT_RABAT;
21 END;
22 /
```

### Exécution :

Nbr lignes traitées : 0

Client no : 1 de nom : Idrissi

Nbr lignes traitées : 1

Client no : 2 de nom : Soufiani

Nbr lignes traitées : 2

Client no : 6 de nom : Doukkali

Nbr lignes traitées : 3

Client no : 8 de nom : Zahraoui

Nbr lignes traitées : 4

Client no : 11 de nom : Meknassi

Nbr lignes traitées : 5

Procédure PL/SQL terminée avec succès.

## Curseur explicite: Exemple 5:

```
SQL> DECLARE
 2  CURSOR CLIENT_RABAT IS
 3  SELECT NO,NOM FROM E_CLIENT
 4  WHERE VILLE='Rabat';
 5
 6  CL_RB CLIENT_RABAT%ROWTYPE;
 7
 8  BEGIN
 9  OPEN CLIENT_RABAT;
10
11  FETCH CLIENT_RABAT INTO CL_RB;
12  WHILE (CLIENT_RABAT%FOUND) LOOP
13      DBMS_OUTPUT.PUT_LINE('Client no : ' || CL_RB.NO || ' de nom : ' || CL_RB.NOM);
14      FETCH      CLIENT_RABAT INTO CL_RB;
15  END LOOP;
16
17  CLOSE CLIENT_RABAT;
18 END;
19 /
```

### Exécution :

Client no : 1 de nom : Idrissi  
Client no : 2 de nom : Soufiani  
Client no : 6 de nom : Doukkali  
Client no : 8 de nom : Zahraoui  
Client no : 11 de nom : Mekkassi

Procédure PL/SQL terminée avec succès.



## Curseur explicite: Exemple 6: BOUCLE FOR

```
SQL>
SQL> DECLARE
2  CURSOR CLIENT_RABAT IS
3  SELECT NO,NOM FROM E_CLIENT
4  WHERE VILLE='Rabat';
5
6 BEGIN
7  FOR CL_RB IN CLIENT_RABAT LOOP
8      DBMS_OUTPUT.PUT_LINE('Client no : ' || CL_RB.NO || ' de nom : ' || CL_RB.NOM);
9  END LOOP;
10 END;
11 /
```

### Exécution :

Client no : 1 de nom : Idrissi  
Client no : 2 de nom : Soufiani  
Client no : 6 de nom : Doukkali  
Client no : 8 de nom : Zahraoui  
Client no : 11 de nom : Meknassi

Procédure PL/SQL terminée avec succès.

•PAS DE OPEN: OUVERTURE IMPLICITE  
•PAS DE CLOSE: FERMETURE IMPLICITE  
•PAS DE FETCH: PARCOURS IMPLICITE  
•PAS DE DECLARATION DE TYPE DE LA  
VARIABLE DE RETOUR:  
IMPLICITE CURSOR%ROWTYPE

## Curseur explicite: Exemple 7: PARAMETRE DE CURSOR

```
SQL> DECLARE
 2  CURSOR CLIENT_PAR_VILLE ( NOM_VILLE IN E_CLIENT.VILLE%TYPE) IS
 3  SELECT NO,NOM FROM E_CLIENT
 4  WHERE VILLE=NOM_VILLE;
 5
 6  NV E_CLIENT.VILLE%TYPE;
 7
 8  BEGIN
 9  NV:='Rabat';
10  DBMS_OUTPUT.PUT_LINE('Les clients de '|| NV ||' sont : ');
11  FOR CL_RV IN CLIENT_PAR_VILLE(NV) LOOP
12      DBMS_OUTPUT.PUT_LINE('Client no : '|| CL_RV.NO ||' de nom : ' || CL_RV.NOM);
13  END LOOP;
14  NV:='Casa';
15  DBMS_OUTPUT.PUT_LINE('Les clients de '|| NV ||' sont : ');
16  FOR CL_RV IN CLIENT_PAR_VILLE(NV) LOOP
17      DBMS_OUTPUT.PUT_LINE('Client no : '|| CL_RV.NO ||' de nom : ' || CL_RV.NOM);
18  END LOOP;
19  END;
20 /
```

### Exécution :

Les clients de Rabat sont :

Client no : 1 de nom : Idrissi

Client no : 2 de nom : Soufiani

Client no : 6 de nom : Doukkali

Client no : 8 de nom : Zahraoui

Client no : 11 de nom : Meknassi

Les clients de Casa sont :

Client no : 3 de nom : Miliani

Client no : 4 de nom : Zamouri

Client no : 7 de nom : Idrissi

Procédure PL/SQL terminée avec succès.

## Curseur explicite: Exemple 7\_1: PARAMETRE DE CURSOR

```
SQL>
SQL> DECLARE
2  CURSOR COMMANDE_DATE_LIV ( DTD IN DATE, DTF IN DATE) IS
3  SELECT NO,DATE_COMMANDE, DATE_LIVRAISON FROM E_COMMANDE
4  WHERE DATE_LIVRAISON BETWEEN DTD AND DTF;
5
6  BEGIN
7  DBMS_OUTPUT.PUT_LINE('Les commandes à livrer entre 01/01/04 et le 15/01/04 sont :');
8  FOR COM IN COMMANDE_DATE_LIV('01/01/04','15/01/04') LOOP
9      DBMS_OUTPUT.PUT_LINE('Commande no : '||COM.NO|| ' à livrer entre
      '||COM.DATE_COMMANDE ||' et ' || COM.DATE_LIVRAISON);
10  END LOOP;
11  END;
12 /
```

### Exécution :

Les commandes à livrer entre 01/01/04 et le 15/01/04 sont :

Commande no : 5 à livrer entre 12/11/03 et 01/01/04

Commande no : 7 à livrer entre 11/11/03 et 01/01/04

Commande no : 8 à livrer entre 02/01/04 et 12/01/04

Commande no : 9 à livrer entre 10/01/04 et 15/01/04

Procédure PL/SQL terminée avec succès.

**LES  
CURSEURS EXPLICITES  
Et  
LA MISE A JOUR DE LA BASE  
(UPDATE, DELETE)**

# BASES DE DONNEES RELATIONNELLES

**CURSOR** nom\_curseur [(parametres) ] [RETURN ROWTYPE] IS requête\_select  
FOR UPDATE;

## OBJECTIF:

Verrouiller les tuples du curseur à MODIFIER ou à SUPPRIMER dans la table

## CONDITION:

- PAS DE DISTINCT DANS LA REQUETE DU CURSEUR
- PAS DE GROUP BY DANS LA REQUETE DU CURSEUR
- PAS DE UNION, INTERSECT ou MINUS DANS LA REQUETE DU CURSEUR
- PAS DE FONCTION D'AGREGATION DANS LA REQUETE DU CURSEUR

## PRECAUTION:

UN COMMIT A LA FIN, SINON LES MODIFICATION ET LES SUPPRESSION  
SERONT ANNULEES

# BASES DE DONNEES RELATIONNELLES

## Curseur explicite: Exemple 8: CURSOR...FOR UPDATE

```
SQL> DECLARE
 2  CURSOR CLIENT_RABAT IS
 3  SELECT NO,NOM, VILLE FROM E_CLIENT
 4  WHERE VILLE='Rabat'
 5  FOR UPDATE;
 6
 7  NC_TANTAN    NUMBER(3);
 8 BEGIN
 9  DBMS_OUTPUT.PUT_LINE('Les clients avant modification: ');
10  FOR CLR IN CLIENT_RABAT LOOP
11      DBMS_OUTPUT.PUT_LINE('Client no : ' || CLR.NO || ' de nom : ' || CLR.NOM || ' et de ville : ' || CLR.VILLE);
12  END LOOP;
13
14  SELECT COUNT(*) INTO NC_TANTAN FROM E_CLIENT WHERE VILLE='TANTAN';
15  DBMS_OUTPUT.PUT_LINE('Les clients habitant TANTAN sont en nombre : ' || NC_TANTAN);
16  --modification de la ville Rabat à TANTAN
17
18  FOR CLR IN CLIENT_RABAT LOOP
19      UPDATE E_CLIENT SET VILLE='TANTAN' WHERE CURRENT OF CLIENT_RABAT;
20  END LOOP;
21
22  COMMIT;
23
24  DBMS_OUTPUT.PUT_LINE('Les clients APRES modification : ');
25  FOR CLR IN (SELECT NO, NOM, VILLE FROM E_CLIENT WHERE VILLE='TANTAN') LOOP
26      DBMS_OUTPUT.PUT_LINE('Client no : ' || CLR.NO || ' de nom : ' || CLR.NOM || ' et de ville : ' || CLR.VILLE);
27  END LOOP;
28 END;
29 /
```

### Exécution :

#### Les clients avant modification :

Client no : 1 de nom : Idrissi et de ville : Rabat  
Client no : 2 de nom : Soufiani et de ville : Rabat  
Client no : 6 de nom : Doukkali et de ville : Rabat  
Client no : 8 de nom : Zahraoui et de ville : Rabat  
Client no : 11 de nom : Meknassi et de ville : Rabat

Les clients habitant TANTAN sont en nombre : 0

#### Les clients APRES modification :

Client no : 1 de nom : Idrissi et de ville : TANTAN  
Client no : 2 de nom : Soufiani et de ville : TANTAN  
Client no : 6 de nom : Doukkali et de ville : TANTAN  
Client no : 8 de nom : Zahraoui et de ville : TANTAN  
Client no : 11 de nom : Meknassi et de ville : TANTAN

Procédure PL/SQL terminée avec succès.

## Travaux pratiques

### Question 1:

**Créer une table E\_Augmentation comprenant les champs suivants :**

**No Number(7), Augmentation Number(11,2), Date\_Augmentation Date, Emp\_No Number(7)**

### Question 2: (Curseur, Instructions OPEN, FETCH, CLOSE, %FOUND)

**Ecrire un programme permettant :**

- la mise à jour du salaire de tous les employés de la table E\_Employe selon les conditions suivantes:**

- si son année d'entrée dans la société est 1995, augmenter le salaire de 50%.**

- si son année d'entrée dans la société est 1996, augmenter le salaire de 25%.**

- si son année d'entrée dans la société est 1997, augmenter le salaire de 10%.**

- l'insertion des modifications dans la table E\_Augmentation des informations suivantes:**

- le montant d'augmentation, la date d'augmentation, le numéro de l'employé, ainsi que le champ No qui sert d'identifiant de ligne pour la table.**

## **Question 3: (Utilisation d'un curseur avec la boucle FOR...LOOP)**

**Créer une table E\_Resultat :**

**E\_Resultat (No Number(2), LB\_Resultat Varchar2(60), VL\_Resultat Number(11,2))**

**Ecrire un programme permettant de faire le total des commandes gérées par chaque employé.**

**Pour les employés qui ont géré des commandes, faire les insertions dans la table E\_Resultat selon le schéma suivant :**

**-No --> <<No-programme>>**

**-LB\_Resultat --> <<Nom\_Employé>> totalise**

**-VL\_Resultat --> <<Total des commandes gérées>>**



## Question 4: (Utilisation d'un curseur paramétré)

**Ecrire un programme qui insère dans la table E\_Resultat le nom de l'employé et son salaire pour les employés dont le salaire vérifie les conditions suivantes :**

**•Si le salaire >3500, insérer dans la table E\_Resultat les données suivantes :**

**-No --> <<No-programme>>**

**-LB\_Resultat --> <<Variable\_Nom\_Employé>> a un salaire > 3500**

**-VL\_Resultat --> <<Variable\_Salaire\_Employé>>**

**•Si le salaire >4500, insérer dans la table E\_Resultat les données suivantes :**

**-No --> <<No-programme>>**

**-LB\_Resultat --> <<Variable\_Nom\_Employé>> a un salaire > 4500**

**-VL\_Resultat --> <<Variable\_Salaire\_Employé>>**

## Question 5 (Curseur et clause CURRENT OF)

**Ecrire un programme permettant :**

- **d'abaisser de 30% le prix conseillé des produits qui ne figurent sur aucune des commandes.**
- **Insérer dans la table E\_Resultat les produits concernés par la réduction :**
  - No --> <<No-programme>>
  - LB\_Resultat --> <<Variable\_Numéro\_Produit>> - <<Nom\_Produit>> baisse de 30%
  - VL\_Resultat --> <<Variable\_Prix\_Conseillé\_avant\_de\_30%>>