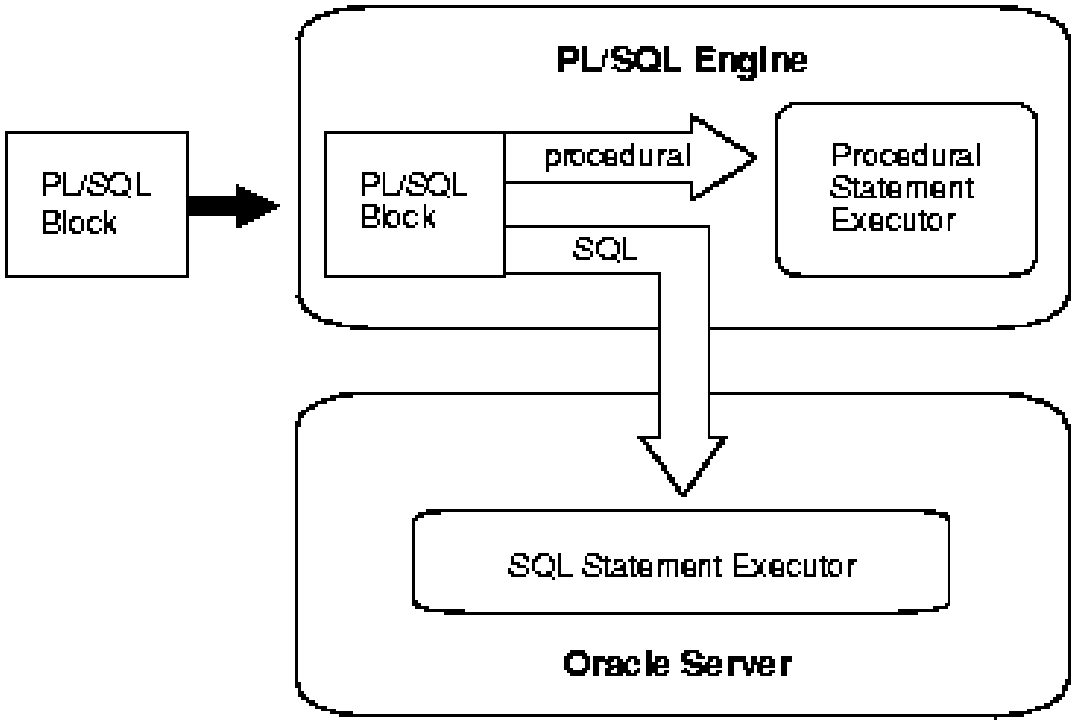
PL/SQL FOR ORACLE

R. OULAD HAJ THAMI

LES FONDAMENTAUX

SOMMAIRE GENERAL

MOTIVATION


PL/SQL Engine
PL/SQL Block
PL/SQL Block
procedural
Procedural Statement Executor
SQL
SQL Statement Executor
Oracle Server

Noyau PL/SQL

Avantages de PL / SQL

- Prise en charge de SQL
- Prise en charge de la programmation orientée objet
- Meilleure performance
- Une productivité accrue
- La portabilité
- L'intégration très forte avec Oracle
- Haute sécurité

PARTIE 1:
LES FONDAMENTAUX

PARTIE 1: SOMMAIRE

STRUCTURE D'UN BLOC PL/SQL

```
DECLARE              --section optionnelle
        déclaration variables, constantes, types, curseurs,...
BEGIN                --section obligatoire
        contient le code PL/SQL
EXCEPTION            --section optionnelle
        traitement des erreurs
END; --obligatoire
```

```
DECLARE --section optionnelle
déclaration variables, constantes, types, curseurs,...
BEGIN --section obligatoire
contient le code PL/SQL

        DECLARE --section optionnelle
                déclaration variables, constantes, types, curseurs,...
        BEGIN --section obligatoire
                contient le code PL/SQL
        EXCEPTION --section optionnelle
                traitement des erreurs
        END; --obligatoire

EXCEPTION --section optionnelle
        traitement des erreurs
END; --obligatoire
```

REMARQUE

LA PORTEE DES VARIABLES EST LA MEME QUE DANS LES LANGAGES DE PROGRAMMATION

LES VARIABLES

nom variable [CONSTANT] type [[NOT NULL] [:= expression | DEFAULT expression];

nom variable	représente le nom de la variable composé de lettres, chiffres, $, _ ou # Le nom de la variable ne peut pas excéder 30 caractères
CONSTANT	indique que la valeur ne pourra pas être modifiée dans le code du bloc PL/SQL
NOT NULL	indique que la variable ne peut pas être NULL, et dans ce cas **expression** doit être indiqué.
type	représente de type de la variable correspondant à l'un des types suivants :

Remarque

Si une variable est déclarée avec l'option CONSTANTE, elle doit être initialisée
Si une variable est déclarée avec l'option NOT NULL, elle doit être initialisée

TYPE	SEMANTIQUE
NUMBER[(e,d)]	Nombre réel avec e chiffres significatifs stockés et d décimales
PLS_INTEGER	Nombre entier compris entre -2 147 483 647 et +2 147 483 647
CHAR [(n)]	Chaîne de caractères de longueur fixe avec n compris entre 1 et 32767 (par défaut 1)
VARCHAR2[(n)]	Chaîne de caractères de longueur variable avec n compris entre 1 et 32767
BOOLEAN	
DATE	
LONG	Chaîne de caractères de longueur variable avec au maximum 32760 octets
ROWID	Permet de stocker l'adresse absolue d'une ligne dans une table sous la forme d'une chaîne de caractères

Exemples de types de bases PL/SQL

SUBTYPE nom_sous_type IS type ;

Exemple:

```
SUBTYPE  nom_employe IS VARCHAR2(20)  NOT NULL:='inconnu';

nom        nom_employe;
```

nom_variable nom_table.nom_colonne%TYPE ;
nom_variable nom_variable_ref%TYPE ;

Exemple:

```
Nom              E_EMPLOYE.NOM%TYPE;
Dat_COM          DATE;
Dat_LIV           Dat_COM%TYPE;
```

nom_variable nom_table%ROWTYPE ;

Exemple:

```
EMPLOYE          E_EMPLOYE%ROWTYPE;
```

LES ENREGISTREMENTS

```
TYPE  nom_type_rec  IS RECORD (
            nom_champ1        type_élément1 [[ NOT NULL] := expression ],
             nom_champ2       type_élément2 [[ NOT NULL] := expression ],
            …
             nom_champN       type_élémentN[[ NOT NULL] := expression ]
 ) ;
Nom_variable        nom_type_rec ;
```

Exemple:

```
TYPE  T_REC_EMP  IS RECORD (
            Num       E_EMPLOYE.NO%TYPE,
            Nom       E_EMPLOYE.NOM%TYPE,
            Pre       E_EMPLOYE.PRENOM%TYPE
);

EMP       T_REC_EMP ;
```

ACCES:

```
EMP.num          EMP.NOM          et          EMP.Pre
```

ASSIGNATION DES VARIABLES (AFFECTATION)

VARIABLE := EXPRESSION

- Lors de la déclaration
- Dans le bloc PL/SQL

```
MON_NUM:= 10 ;

MA_CHAINE := 'Chaîne de caractères' ;

MA_TAXE :=PRIX*TAUX;

MON_BOOLEAN := FALSE; MON_BOOLEAN := (NOM='toto');

BONUS := SALAIRE * 0.10;

MA_LIMITE_BUDGET CONSTANT REAL := 5000.00;

MA_DATE:='12/12/2012'

MON_DEP := DEPARTEMENT.NUMDEP;
```

AFFICHAGE DES VALEURS DES VARIABLES

ACTIVATION DU SERVEUR D'AFFICHAGE:

```
SQL> SET SERVEROUTPUT ON
```

EXEMPLES:

```
MA_CHAINE := 'Chaîne de caractères' ;

DBMS_OUTPUT.PUT_LINE ('Affichage de la valeur de la chaine: '|| MA_CHAINE);

DBMS_OUTPUT.PUT_LINE ('Le prix TTC'|| PRIX*TAUX);

DBMS_OUTPUT.PUT_LINE ('NOM:'||nom_emp ||'  prenom : '||pre_emp||'DT Naissance'||
DT)
```

AFFECTATION DES VARIABLES A PARTIR D'UNE BD

SCHEMA DE LA BASE D'EXEMPLES

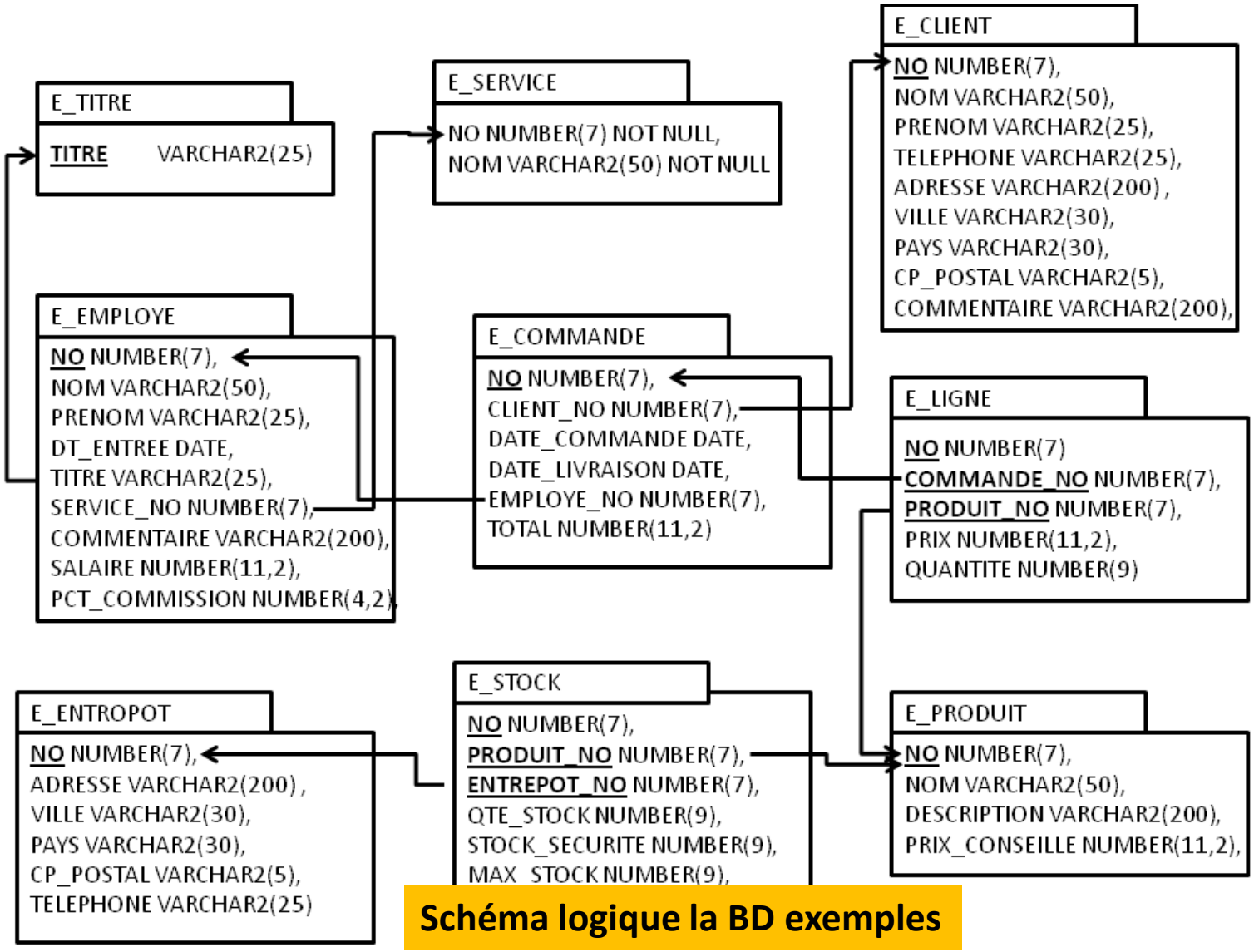


Schéma logique la BD exemples

SELECT <COLONNE_OU_TUPLE> TO <VAR> FROM NOM_TABLE WHERE CONDITION;

Exemple 1:

```
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
  2              NOM_EMP     VARCHAR2(20);
  3  BEGIN
  4    SELECT NOM INTO NOM_EMP
  5    FROM E_CLIENT
  6    WHERE NO=1;
  7    DBMS_OUTPUT.PUT_LINE ('Le nom du client NO 1 est '|| NOM_EMP);
  8  END;
  9  /
Le nom du client NO 1 est Idrissi

Procédure PL/SQL terminée avec succès.
```

Exemple 2:

```
SQL> DECLARE
  2              NOM_EMP     E_CLIENT.NOM%TYPE;
  3  BEGIN
  4    SELECT NOM INTO NOM_EMP
  5    FROM E_CLIENT
  6    WHERE NO=1;
  7    DBMS_OUTPUT.PUT_LINE ('Le nom du client NO 1 est '|| NOM_EMP);
  8  END;
  9  /
Le nom du client NO 1 est Idrissi

Procédure PL/SQL terminée avec succès.
```

Exemple 3:

```
SQL> DECLARE
  2              NOM_EMP     VARCHAR2(20);
  3              PRE_EMP     VARCHAR2(20);
  4  BEGIN
  5    SELECT NOM, PRENOM INTO NOM_EMP, PRE_EMP
  6    FROM E_CLIENT
  7    WHERE NO=1;
  8    DBMS_OUTPUT.PUT_LINE ('Le nom du client NO 1 est '|| NOM_EMP);
  9    DBMS_OUTPUT.PUT_LINE ('son prénom est '|| PRE_EMP);
 10  END;
 11  /
Le nom du client NO 1 est Idrissi
son prénom est Mohammed

Procédure PL/SQL terminée avec succès.
```

Exemple 4:

```
SQL> DECLARE
  2              TYPE T_EMP IS RECORD (
  3                NOM_EMP    VARCHAR2(20),
  4                PRE_EMP    VARCHAR2(20)
  5              );
  6              EMP  T_EMP;
  7  BEGIN
  8    SELECT NOM, PRENOM INTO EMP.NOM_EMP, EMP.PRE_EMP
  9    FROM E_CLIENT
 10     WHERE NO=1;
 11    DBMS_OUTPUT.PUT_LINE ('Le nom du client NO 1 est '|| EMP.NOM_EMP);
 12    DBMS_OUTPUT.PUT_LINE ('son prénom est '|| EMP.PRE_EMP);
 13  END;
 14  /
Le nom du client NO 1 est Idrissi
son prénom est Mohammed

Procédure PL/SQL terminée avec succès.
```

Exemple 5:

```
SQL> DECLARE
  2              TYPE T_EMP IS RECORD (
  3                NOM_EMP    VARCHAR2(20),
  4                PRE_EMP    VARCHAR2(20)
  5              );
  6              EMP  T_EMP;
  7  BEGIN
  8    SELECT NOM, PRENOM INTO EMP
  9    FROM E_CLIENT
 10     WHERE NO=1;
 11    DBMS_OUTPUT.PUT_LINE ('Le nom du client NO 1 est '|| EMP.NOM_EMP);
 12    DBMS_OUTPUT.PUT_LINE ('son prénom est '|| EMP.PRE_EMP);
 13  END;
 14  /
Le nom du client NO 1 est Idrissi
son prénom est Mohammed

Procédure PL/SQL terminée avec succès.
```

Exemple 6:

```
SQL> DECLARE
  2                EMP  E_CLIENT%ROWTYPE;
  3  BEGIN
  4    SELECT * INTO EMP
  5    FROM E_CLIENT
  6    WHERE NO=1;
  7    DBMS_OUTPUT.PUT_LINE ('client NO 1 est :');
  8    DBMS_OUTPUT.PUT_LINE ('NOM            '|| EMP.NOM);
  9    DBMS_OUTPUT.PUT_LINE ('PRENOM         '|| EMP.PRENOM);
 10    DBMS_OUTPUT.PUT_LINE ('TELEHPONE      '|| EMP.TELEPHONE);
 11    DBMS_OUTPUT.PUT_LINE ('ADRESSE        '|| EMP.ADRESSE);
 12    DBMS_OUTPUT.PUT_LINE ('VILLE          '|| EMP.VILLE);
 13    DBMS_OUTPUT.PUT_LINE ('PAYS           '|| EMP.PAYS);
 14    DBMS_OUTPUT.PUT_LINE ('CP_POSTAL      '|| EMP.CP_POSTAL);
 15    DBMS_OUTPUT.PUT_LINE ('COMMENTAIRE    '|| EMP.COMMENTAIRE);
 16  END;
 17  /
client NO 1 est :
NOM             Idrissi
PRENOM          Mohammed
TELEHPONE       O60000000
ADRESSE         Rue 1, N° 23
VILLE           Rabat
PAYS            Maroc
CP_POSTAL       5000
COMMENTAIRE     Pas de commentaire

Procédure PL/SQL terminée avec succès.
```

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
```

CURSEUR

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
```

STRUCTURES DE CONTRÔLE

Structures de contrôles PLSQL

IF - THEN - ELSE - END IF

```
IF condition THEN
        instruction1        ;
        instruction 2       ;
        ........
         instruction 2      ;
END IF;
```

PLSQL IF-THEN-END IF: Exemple 1

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2    x  integer := 10; y integer := 15;
  3    BEGIN
  4              IF x<Y THEN
  5                        DBMS_OUTPUT.PUT_LINE (x || ' < ' ||y);
  6              END IF  ;
  7  END;
  8  /
10 < 15

Procédure PL/SQL terminée avec succès.
```

```
IF condition1 THEN
        instruction1;
        instruction 2;
ELSE
        instruction3;
END IF;
```

PLSQL IF-THEN-ELSE-END IF: Exemple 2

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2    x  integer := 20; y integer := 15;
  3    BEGIN
  4              IF x<Y THEN
  5                        DBMS_OUTPUT.PUT_LINE (x || ' < ' ||y);
  6              ELSE
  7                        DBMS_OUTPUT.PUT_LINE (x || ' >= ' ||y);
  8              END IF  ;
  9  END;
 10  /
20 >= 15

Procédure PL/SQL terminée avec succès.
```

```
IF condition1 THEN
        instruction1;
        instruction 2;
ELSIF condition2 THEN
        instruction 3;
        instruction 4;
ELSIF condition3 THEN
        instruction 5;
        instruction 6;
ELSE    instruction 7;
END IF;
```

PLSQL IF-THEN-ELSIF-THEN…..END IF: Exemple 3

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2    x  integer := 20; y integer := 20;
  3    BEGIN
  4              IF x<Y THEN
  5                        DBMS_OUTPUT.PUT_LINE (x || ' < ' ||y);
  6              ELSIF x=Y THEN
  7                        DBMS_OUTPUT.PUT_LINE (x || ' = ' ||y);
  8              ELSIF x<Y THEN
  9                        DBMS_OUTPUT.PUT_LINE (x || ' < ' ||y);
 10              ELSE
 11                       DBMS_OUTPUT.PUT_LINE ('Bizzare!!');
 12              END IF  ;
 13  END;
 14  /
20 = 20

Procédure PL/SQL terminée avec succès.
```

Structures répétitives PLSQL

WHILE - LOOP - END LOOP

```
WHILE   conditions
LOOP
            instruction1;
            instruction2;
END LOOP;
```

PLSQL LOOP – WHILE LOOP-END LOOP: Exemple 1

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2             cpt             INTEGER  := 0;
  3  BEGIN
  4             WHILE cpt<10 LOOP
  5                DBMS_OUTPUT.PUT_LINE ('Valeur suivante de X :  ' || cpt);
  6                cpt:=cpt+1;
  7             END LOOP;
  8  END;
  9  /
Valeur suivante de X :  0
Valeur suivante de X :  1
Valeur suivante de X :  2
Valeur suivante de X :  3
Valeur suivante de X :  4
Valeur suivante de X :  5
Valeur suivante de X :  6
Valeur suivante de X :  7
Valeur suivante de X :  8
Valeur suivante de X :  9

Procédure PL/SQL terminée avec succès.
```

Structures répétitives PLSQL

FOR - IN - LOOP

```
FOR  compteur IN [REVERSE]  borne_inf..borne_sup LOOP

        instruction1   ;
        instruction2   ;
        instruction3   ;

[EXIT WHEN condition];
END LOOP;
```

PLSQL FOR –IN-LOOP: Exemple 1

```
SQL> SET SERVEROUTPUT ON
SQL> BEGIN
  2    FOR  i IN 1..5 LOOP
  3              DBMS_OUTPUT.PUT_LINE (i);
  4    END LOOP;
  5  END;
  6  /
1
2
3
4
5

Procédure PL/SQL terminée avec succès.
```

PLSQL FOR –IN-LOOP: Exemple 2

```
SQL> SET SERVEROUTPUT ON
SQL> BEGIN
  2    FOR i IN REVERSE 1..5 LOOP
  3              DBMS_OUTPUT.PUT_LINE (i);
  4    END LOOP;
  5  END;
  6  /
5
4
3
2
1

Procédure PL/SQL terminée avec succès.

SQL>
```

PLSQL FOR –IN-LOOP: Exemple 3

```
SQL>
SQL> SET SERVEROUTPUT ON
SQL> BEGIN
  2    FOR i IN 1..5 LOOP
  3              DBMS_OUTPUT.PUT_LINE (i);
  4    EXIT WHEN i>3;
  5    END LOOP;
  6  END;
  7  /
1
2
3
4

Procédure PL/SQL terminée avec succès.

SQL>
```

Structures de contrôles PLSQL

CASE - WHEN - THEN - ELSE - END CASE

```
CASE selecteur
        WHEN expression1 THEN instruction1;
        WHEN expression2 THEN instruction2;
        …
        WHEN expression3 THEN instruction3;
        ELSE instruction4;
END CASE;
```

PLSQL CASE -WHEN -ELSE -END CASE: Exemple 1

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2    x  integer := 2;
  3  BEGIN
  4                CASE X
  5                 WHEN 1 THEN DBMS_OUTPUT.PUT_LINE ('Le premier');
  6                 WHEN 2 THEN DBMS_OUTPUT.PUT_LINE ('Le deuxième');
  7                 WHEN 3 THEN DBMS_OUTPUT.PUT_LINE ('Le troisième');
  8                 ELSE        DBMS_OUTPUT.PUT_LINE ('Le dernier');
  9                END CASE;
 10  END;
 11  /
Le deuxième

Procédure PL/SQL terminée avec succès.
```

```
CASE selecteur
        WHEN expression1 THEN instruction1;
        WHEN expression2 THEN instruction2;
        …
        WHEN expression3 THEN instruction3;
        ELSE instruction4;
END CASE;
```

PLSQL CASE -WHEN -ELSE -END CASE: Exemple 2

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2   x  integer := 2;
  3  BEGIN
  4              CASE
  5               WHEN  x=1 THEN DBMS_OUTPUT.PUT_LINE ('Le premier');
  6               WHEN  x=2 THEN DBMS_OUTPUT.PUT_LINE ('Le deuxième');
  7               WHEN  x=3 THEN DBMS_OUTPUT.PUT_LINE ('Le troisième');
  8               ELSE       DBMS_OUTPUT.PUT_LINE ('Le dernier');
  9              END CASE;
 10  END;
 11  /
Le deuxième

Procédure PL/SQL terminée avec succès.
```

TRAVAUX PRATIQUES

PL/SQL : séance 1

Objectifs:

manipuler les variables,
les bloc PL/SQL
et les structures de contrôle

On considère le schéma de la base de données précédente.

Sachant que les numéro des clients sont numéroté dans l'ordre (1,2,…):

Ecrire un bloc PL/SQL

1. **Qui affiche le nom et la ville du client numéro 3;**
2. **qui affiche le numéro, le nom et la ville de chaque client;**
3. **qui affiche uniquement les client qui habitent rabat**
4. **Qui vérifie si le client numéro 25 existe, si oui, il affiche ses informations, sinon, il affiche un message d'erreur**