PARIS PANTHÉON-ASSAS UNIVERSITÉ

Advanced Databas

Lecture 1(b) - PL / SQ

Hanen OCHI

hanen.ochi@efrei.fr

PARIS PANTHÉON-ASSAS UNIVERSITÉ

part IV

-

The Structures of con
conditions and loo

PARIS PANTHÉON-ASSAS UNIVERSITÉ

PARIS PANTHÉON-ASSAS UNIVERSITÉ

control structure in PL /

- **IF conditional:**
  - IF THEN END IF;
  - IF THEN ELSE END IF;
  - IF THEN ELSIF THEN END IF;
- **loops :**
  - LOOP END LOOP;
  - FOR LOOP END LOOP;
  - WHILE LOOP END LOOP;

PARIS PANTHÉON-ASSAS UNIVERSITÉ

IF statement

syntax:

```
IF condition THEN
   statements;
[ELSIF condition THEN
   set;]
[ELSE
   set;]
END IF;
```

Example of a simple IF:

Set the Employee ID 'MARK' 101.

efrei

PARIS PANTHÉON-ASSAS UNIVERSITÉ

PARIS PANTHÉON-ASSAS UNIVERSITÉ

IF statement

syntax:

```
IF condition THEN
   statements;
[ELSIF condition THEN
   set;]
[ELSE
   set;]
END IF;
```

Example 1:

Set the Employee ID 'MARK' 101.

```
IF v_nom = 'MARK
        THEN v_id: 
END IF;
```

efrei

PARIS PANTHÉON-ASSAS UNIVERSITÉ

IF statement

Example 2: If the employee name is 'Clement', then assig

- **the position 'Teacher'**
- **the department No. 102 and**
- **a 25% commission on his current salary.**

PARIS PANTHÉON-ASSAS UNIVERSITÉ

IF statement

Example 2: If the employee name is 'Clement', then assig

- **the position 'Teacher'**
- **the department No. 102 and**
- **a 25% commission on his current salary.**

```
IF v_nom = ‘Clément’ THEN
    v_poste := ‘Enseignant';
    v_deptno := 102;
    v_nouv_comm := sal * 0.25;
END IF;
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

IF statement

Example 3: If the employee name is 'Clement', then

- **assign him the job 'Teacher', the department No. 10**
commission on his current salary.
- **otherwise display 'non-existent employee’.**

PARIS PANTHÉON-ASSAS UNIVERSITÉ

IF statement

Example 3: If the employee name is 'Clement', then

- **assign him the job 'Teacher', the department No. 10**
  commission on his current salary.
- **otherwise display 'non-existent employee.**

```
IF v_nom = ‘Clément’ THEN
    v_poste := ‘Enseignant';
    v_deptno := 102;
    v_nouv_comm := sal * 0.25;
ELSE
    DBMS_OUTPUT.PUT_LINE(‘Employé
END IF;
```

IF statement

Example 4: if the IF is within a function, we can use RETU
a value.

```
IF v_debut > 100 THEN
    RETURN ( 2 * v_debut);
ELSIF v_debut >= 50 THEN
    RETURN ( 5 * v_debut);
ELSE
    RETURN (1 * v_debut);
END IF;
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

basic loop

syntax:

```
LOOP                                  - delimiter
    Statement 1;                      -- States
    …….
                                      - EXIT
    EXIT [ WHEN condition];           statement
END LOOP;                             - delimiter
```

basic loop

Example: Insert in the "article" table 10 items 1 to 10 and with today's date.

efrei

PARIS PANTHÉON-ASSAS UNIVERSITÉ

basic loop

Example: Insert in the "article" table 10 items numbered from 1 to 10 and with today's date.

```
DECLARE
    v_Date              DATE;
    v_compteur          NUMBER(2) := 1;
BEGIN
    v_Date := SYSDATE;
    LOOP
        INSERT INTO article VALUES(v_compteur,v_D
        v_compteur := v_compteur + 1;
        EXIT WHEN v_compteur > 10;
    END LOOP;
END;
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

FOR loop

syntax:

```
FOR index IN [REVERSE] Borne_inf .. Borne_sup LO
    Statement 1;
    Statement 2;
    …… ..
END LOOP;
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

FOR loop

syntax:

```
FOR index IN [REVERSE] Borne_inf .. Borne_sup LO
    Statement 1;
    Statement 2;
    ...... ..
END LOOP;
```

Remarks :

- **we don’t need to declare the index, it is declared**
- **The option REVERSE you can browse the index up**

PARIS PANTHÉON-ASSAS UNIVERSITÉ

PARIS PANTHÉON-ASSAS UNIVERSITÉ

FOR loop

Example: Create Nb articles indexed from 1 to
with the system date using the FOR loop.

PARIS PANTHÉON-ASSAS UNIVERSITÉ

FOR loop

Example: Create Nb articles indexed from 1 t
with the system date using the FOR loop.

```
DECLARE
        v_Date          DATE;
BEGIN
        v_Date := SYSDATE;
        FOR i IN 1 .. &Nb LOOP
                INSERT INTO article VALUES (i, v_Date);
        END LOOP;
END;
/
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

FOR loop

Example: Create Nb articles indexed from 1 to with the system date using the FOR loop.

```
DECLARE
    v_Date        DATE;
BEGIN
    v_Date := SYSDATE;
    FOR i IN 1 .. &Nb LOOP
        INSERT INTO article VALUES (i, v_Date);
    END LOOP;
END;
/
```

With &Nb, the system requires a value to the use

PARIS PANTHÉON-ASSAS UNIVERSITÉ

At the beginning of the loop.

WHILE loop

syntax:

```
WHILE condition LOOP
    Statement 1;
    Statement 2;
    …… ..
END LOOP;
```

Note :

- **The condition is evaluated before each itera**

PARIS PANTHÉON-ASSAS UNIVERSITÉ

WHILE loop

**example: Insert the "Item" table 10 items numbered fr
with today's date.**

PARIS PANTHÉON-ASSAS UNIVERSITÉ

WHILE loop

example: Insert the "Item" table 10 items numbered fr
with today's date.

```
DECLARE
    v_Date              DATE;
    v_compteur          NUMBER(2) := 1;
BEGIN
    v_Date := SYSDATE;
    While v_compteur < 10 LOOP
        INSERT INTO article VALUES(v_compte
        v_compteur := v_compteur + 1;
    END LOOP;
END;
/
```

efrei

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Nested Loops and Labe

- We can Nest loops to multiple levels.
- Use labels to distinguish between blo
  loops.
- Leave the outer loop with an EXIT refe
  the label.
- The label is written as << label_name

Nested Loops and Labels

Example:

```
BEGIN
    << bouc_ext>>
    LOOP
            v_compteur := v_compteur +
    1; EXIT WHEN v_compteur > 10;
            <<bouc_int>>
            LOOP
                ……
                EXIT bouc_ext WHEN total_fait = 1;

                EXIT WHEN int_fait = 1;
                ……
            END LOOP bouc_int;
     END LOOP bouc_ext;
END;
/
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Nested Loops and Labels

Example:

```
BEGIN
      << bouc_ext>>
      LOOP
                 v_compteur := v_compteur +
      1; EXIT WHEN v_compteur > 10;
                 <<bouc_int>>
                 LOOP
                       ......
                       EXIT bouc_ext WHEN total_fait = 1; -- exit the two l

                       EXIT WHEN int_fait = 1; -- exit only the intern loop
                       ......
                 END LOOP bouc_int;
        END LOOP bouc_ext;
END;
/
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

part V

-

Error managemen

Handling exceptions

- Exception handling is a mechanism to handl
encountered when running.

- This allows the execution to continue if the
enough to finish running.

- If an error is encountered and treated in t
exception is processed, program beyond
block and the execution process continues.

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Types of exceptions

- predefined Oracle exceptions
- Non-predefined Oracle Exceptions

}

trigge
i

- User-defined exceptions

}

trigge
e

efrei

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Capture exceptions

syntax:

```
EXCEPTION
 WHEN    exception1 [OR exception.2 ...]  THEN
             Stmt1;
             Stmt2;
             .......
                exception.2 [OR exception.4 ...]
  [WHEN    THEN
             Stmt3;
             Stmt4;
             .......]

  [WHEN    OTHERS THEN
             Stmt5;
             .......]
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Predefined exceptions

- Refer to the name in the part where excepti
  processed.
- Some predefined exceptions:
  - NO_DATA_FOUND
  - TOO_MANY_ROWS
  - INVALID_CURSOR
  - ZERO_DIVIDE
  - DUP_VAL_ON_INDEX

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Predefined exceptions

example:

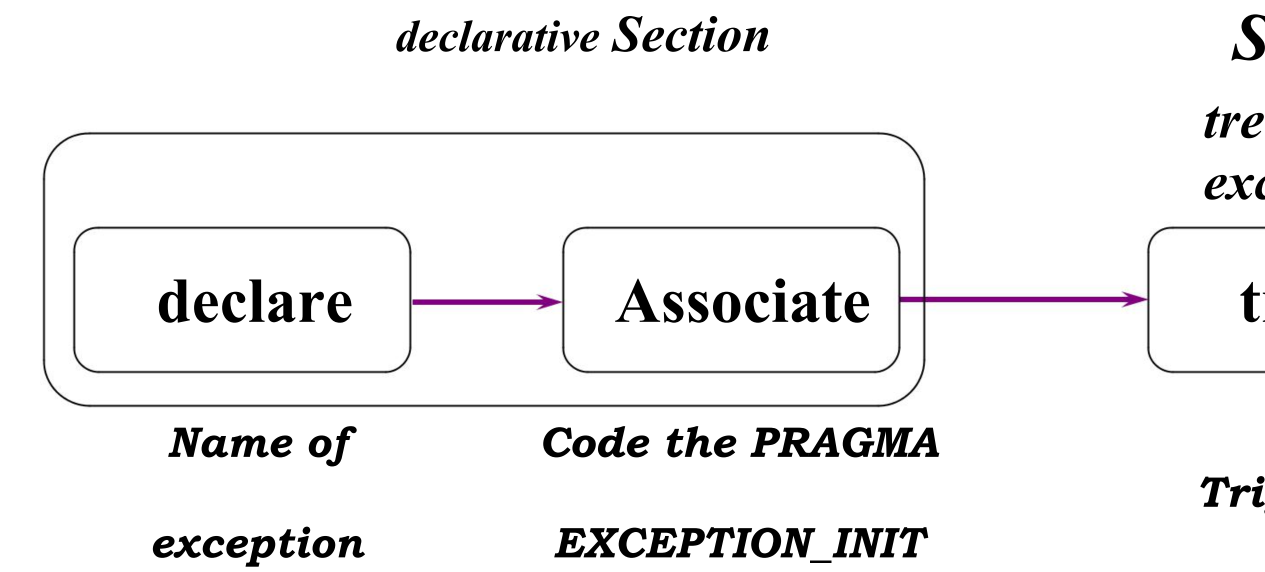
```
BEGIN
      …………
      COMMIT;
EXCEPTION
       WHEN   NO_DATA_FOUND   THEN
              DBMS_OUTPUT.PUT_LINE (TO_CHAR (etudno) || ‘N

      WHEN   TOO_MANY_ROWS  THEN
              énoncé1;
              DBMS_OUTPUT.PUT_LINE (‘ Données invalides’);

      WHEN   OTHERS THEN
              énoncé2;
             DBMS_OUTPUT.PUT_LINE (‘ Autres erreurs ’);
              ROLLBACK;
END ;
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Non-Predefined Excepti



efrei

PARIS PANTHÉON-ASSAS UNIVERSITÉ

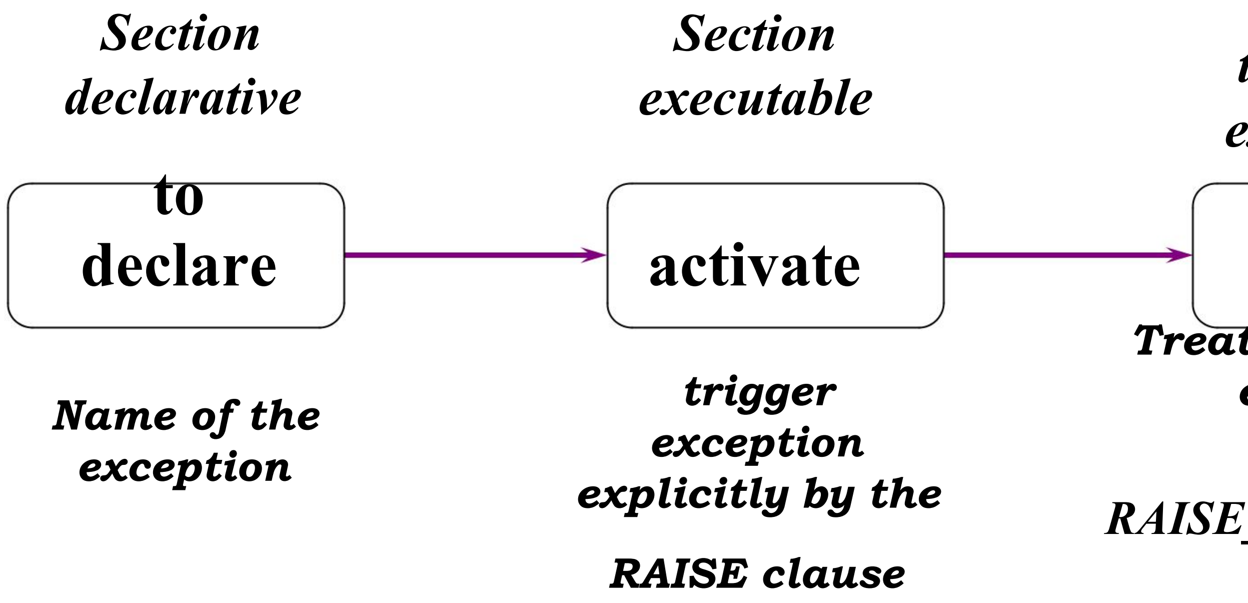
Non-Predefined Excepti

Example: Capture the Error No. 2291 (vio the integrity constraint).

```
DECLARE
            cont_integrit_viol     EXCEPTION;
            PRAGMA EXCEPTION_INIT(Cont_integrit_vio
BEGIN
      ...........
EXCEPTION
            WHEN cont_integrit_viol     THEN
            DBMS_OUTPUT.PUT_LINE ( 'violation of inte
constraint');
END;
/
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

User-defined exception



PARIS PANTHÉON-ASSAS UNIVERSITÉ

User-defined exception

The command RAISE_APPLICATION_ERROR displays a messa
error code for an exception defined by the user.

syntax:

RAISE_APPLICATION_ERROR(Error_code, messa

- The error code must be between -20000 and -20999
- The error message will be displayed as for a classic mistake.
- The PL / SQL code stops immediately and displays th

PARIS PANTHÉON-ASSAS UNIVERSITÉ

User-defined exceptio

example:

```
DECLARE
      x NUMBER;
      x_trop_petit     EXCEPTION;
BEGIN
      ............
      IF  x <5 THEN RAISE x_trop_petit;
      END IF;
      .............
EXCEPTION
      WHEN x_trop_petit THEN
      RAISE_APPLICATION_ERROR (-20 002, 'the value of
      !!');
       ..........
END;
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

The capture function

- **SQLCODE**

  - Returns the numeric value of the error co

- **SQLERRM**

  - Returns the message associated with the e

PARIS PANTHÉON-ASSAS UNIVERSITÉ

The capture functions

example:

```
DECLARE
        v_code_erreur      NUMBER;
        v_message_erreur   VARCHAR2(255);
BEGIN
………
EXCEPTION
        ………
        WHEN OTHERS THEN
                v_code_erreur := SQLCODE;
                v_message_erreur := SQLERRM;
                INSERT  INTO     erreurs
                VALUES    (v_code_erreur, v_message_erreur);
END;
/
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

part VI

-

Cursors

Cursors

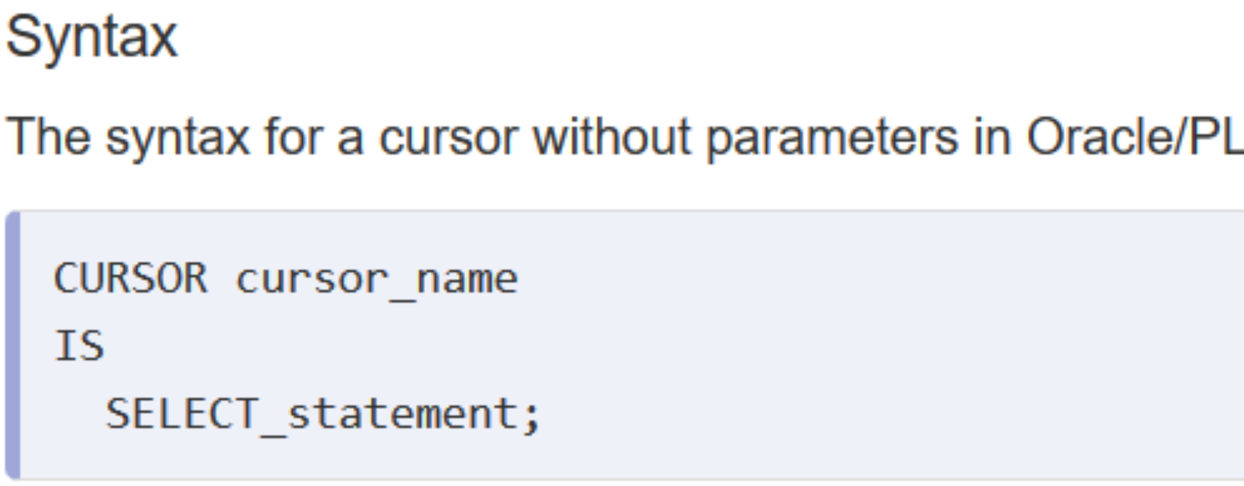
A cursor is a pointer to a private SQL memory area alloc treatment of an SQL statement. The cursor treats re rows) one by one.

efrei

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Cursors

A cursor is a SELECT statement that is defined v declaration section of your PLSQL code. We'll ta three different syntaxes to declare a cursor.



Syntax

The syntax for a cursor without parameters in Oracle/PL

```
CURSOR cursor_name
IS
  SELECT_statement;
```

Declaration of cursor

syntax:

```
CURSOR nom_du_curseur IS a
States SELECT;
```

- **Do not include the INTO clause in the cursor dec**
- **If the rows processing must be done in a specific o ORDER BY clause is used in the query.**

Declaration of cursor

example:

```
DECLARE
        CURSOR C1 IS
        SELECT  RefArt,    NomArt, QteArt
        FROM article
        WHERE QteArt <500;
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Opening the cursors

syntax:

```
OPEN nom_du_curseur;
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Opening the cursors

syntax:

```
OPEN nom_du_curseur;
```

- **Open the cursor to run the query and identify the active set.**
- **Use attributes of the cursor to test the result of FETCH.**

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Search data

syntax:

```
FETCH nom_du_curseur
INTO variable1, [variable2, ...];
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Search data

syntax:

```
FETCH nom_du_curseur
INTO variable1, [variable2, ...];
```

- **Find information of current and put them in variables.**
- **The lines are treated with the same order as the ta FETCH, the next row is treated.**

PARIS PANTHÉON-ASSAS UNIVERSITÉ

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Search data

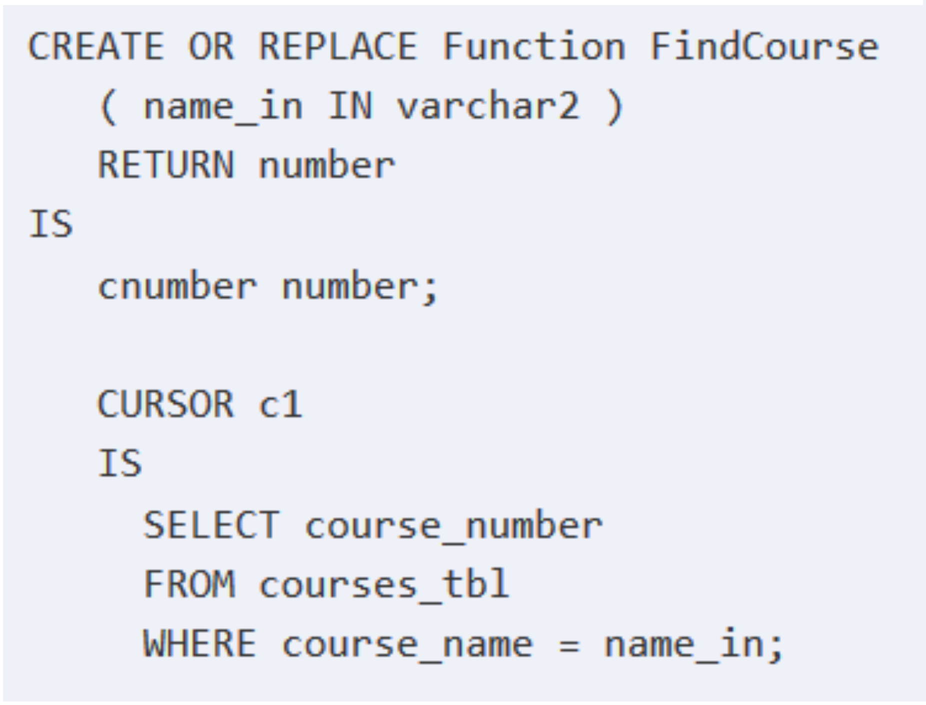
Example 1:

```
FETCH c1 INTO      v_RefArt, v_NomArt, v_QteArt;
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

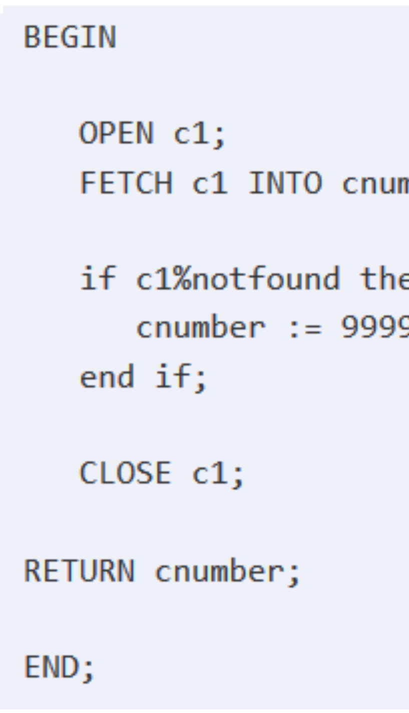
Search data

Example 2:



```
CREATE OR REPLACE Function FindCourse
   ( name_in IN varchar2 )
   RETURN number
IS
   cnumber number;

   CURSOR c1
   IS
     SELECT course_number
     FROM courses_tbl
     WHERE course_name = name_in;
```



```
BEGIN

   OPEN c1;
   FETCH c1 INTO cnum

   if c1%notfound the
      cnumber := 999
   end if;

   CLOSE c1;

RETURN cnumber;

END;
```

efrei

PARIS PANTHÉON-ASSAS UNIVERSITÉ

efrei
PARIS PANTHÉON-ASSAS UNIVERSITÉ

cursor closure

syntax:

```
CLOSE    nom_du_curseur;
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

cursor closure

syntax:

```
CLOSE   nom_du_curseur;
```

- **Close the cursor after the end of lines treatment.**
- **Reopen the cursor, if necessary.**
- **You can not search for information in a cursor if i**

PARIS PANTHÉON-ASSAS UNIVERSITÉ

The attributes of explicit c

Get the cursor status information:

Attribute	Type	Descriptio
ISOPEN%	BOOLEAN	Is TRUE if the cursor is open.
% NOTFOUND	BOOLEAN	Is TRUE if the FETCH latest returns no rows.
% FOUND	BOOLEAN	Is TRUE if the latest FE returns a row.
% ROWCOUNT	NUMBER	Returns the number of r so far.

PARIS PANTHÉON-ASSAS UNIVERSITÉ

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Attribute% ISOPEN

- The research lines is possible only if the cursor is
- Use the attribute ISOPEN% before FETCH to te the cursor is opened or not.

example:

```
IF   NOT C1%ISOPEN   THEN
          OPEN C1
END IF;
LOOP
          FETCH C1 …….
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

The attributes% FOUND,% NOT and% ROWCOUNT

- **Use the attribute % ROWCOUNT to provi exact number of rows processed.**
- **Using attributes % FOUND and % NOTF formulate the test stop of the loop.**

PARIS PANTHÉON-ASSAS UNIVERSITÉ

The attributes% FOUND,% NOT and% ROWCOUNT

example:

```
LOOP
    FETCH   curs1     INTO     v_etudid, v_nom;
    IF   curs1%ROWCOUNT > 20   THEN
    ...........
    END IF;
    EXIT   WHEN   curs1%NOTFOUND;
END LOOP;
```

the records

A record is a variable that contains an entire row of a table.

the records

A record is a variable that contains an entire row of a table.

example:

```
DECLARE
    CURSOR    Etud_Curs   IS
    SELECT   etudno, nom, age, ard
    FROM     etud    WHERE    age < 26;
    Etud_Record   Etud_Curs%ROWTYPE;              -- definition
BEGIN
    OPEN    Etud_Curs;
    ……….
FETCH                                          Etud_Curs INTO I
                                                        -- filli
    IF Etud_Record.age <18 THEN                     -- using of t
```

………

END;

FOR Loops for curso

syntax:

```
FOR   nom_record   IN cursor_name
LOOP
    -- information processing
END LOOP;
```

- A shortcut to process explicit cursors.
- OPEN, FETCH and CLOSE are implicitly.

- **Do not declare the record, it is declared implicit**

PARIS PANTHÉON-ASSAS UNIVERSITÉ

FOR Loops for curso

example:

```
DECLARE
    CURSOR Cur_Etud IS
        SELECT  *
        FROM Etud ;
BEGIN
    FOR Rec_Etud  IN  Cur_Etud  LOOP
        DBMS_OUTPUT.PUT_LINE(Rec_Etud.nom ||' '|| Re
    END LOOP;
END;
/
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

PARIS PANTHÉON-ASSAS UNIVERSITÉ

The attributes of implicit cu

They are used to test the results of SQL statements:

SQL% ROWCOUNT	Number of rows affected by the th recent SQL statement (returns an integer).
SQL% FOUND	Boolean attribute that is TRUE if the most recent SQL statement a one or more lines.
SQL% NOTFOUND	Boolean attribute that is TRUE if the most recent SQL statement c affect any line.
SQL% ISOPEN	always FALSE because PL / SQL closes implicit cursors

	immediately after their execution.

The attributes of SQL cu

example: Remove from the ITEM table lines corresponding to
Display the number of deleted rows.

```
DECLARE
        v_lot       NUMBER: = 605;
BEGIN
        DELETE FROM item WHERE lot = v_lot;
        DBMS_OUTPUT.PUT_LINE (SQL% ROWCOUNT | | 'Lines
END;
/
```

efrei

PARIS PANTHÉON-ASSAS UNIVERSITÉ

PARIS PANTHÉON-ASSAS UNIVERSITÉ

part VII

-

Triggers (Triggers

Triggers

- A trigger is a PL / SQL program whic automatically before or after a LMD op (insert, Update, Delete).

- Unlike procedures, a trigger is automatically following an LMD order.

Event-Condition-Actic

A trigger is activated by an event:

- Insertion, deletion or modification on a table

If the trigger is activated, a condition is evaluated:

- Predicate must return true

If the condition is true, the action is performed:

- Inserting, deleting, or modifying database

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Trigger Components

When the trigger is activated?

- **BEFORE : The code in the body of the trigger ru**
  LMD trigger events.

- **AFTER : The code in the body triggers runs afte**
  trigger events.

PARIS PANTHÉON-ASSAS UNIVERSITÉ

The components of the tr

The triggering events:

What LMD operations that cause the execution of the tri

- **INSERT**
- **UPDATE**
- **DELETE**
- **The combination of these operations**

PARIS PANTHÉON-ASSAS UNIVERSITÉ

The components of the tr

The body of the trigger is defined by a PL / SQ

```
DECLARED
BEGIN
EXCEPTION
END;
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

The components of the tr

syntax:

```
CREATE [OR REPLACE] TRIGGER <Trigger_na
[BEFORE | AFTER] [INSERT [OR] DELETE [OR]
ON <Table_name>
[FOR EACH ROW] [WHEN <Condition>]
DECLARE
BEGIN
EXCEPTION
END ;
/
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

The components of the

example:

```
CREATE OR REPLACE TRIGGER StartInvoice
AFTER INSERT ON Invoice
      FOR EACH ROW
DECLARE
      VNbInsert Number;
BEGIN
      SELECT Nb_Insert INTO VNbInsert
            FROM Statistical
            WHERE Table_name = 'Invoice' ;
      UPDATE Statistical
            SET Nb_Insert = VNbInsert + 1
            WHERE Table_name = 'Invoice' ;
EXCEPTION
      WHEN NO_DATA_FOUND THEN
            INSERT INTO Statistical VALUES (1, 'Invo
END ;
```

efrei

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Handling triggers

Enable or disable a Trigger:

ALTER TRIGGER <Trigger_name> [ENABLE | DISA

Delete Trigger:

DROP TRIGGER <Trigger_name>;

Identify your triggers on your DB:

SELECT FROM trigger_name User_Triggers ;

PARIS PANTHÉON-ASSAS UNIVERSITÉ

:Old and :New Attribut

These two attributes are used to manage the old and ne values .

Insert (.........) ...	→ **: New.nom_att**
Delete .... **Where (......) ...**	→ **: Old.nom_att**
Update ... **Set (......) ...** **Where (......) ...**	→ **: New.nom_att** → **: Old.nom_att**

Beware ":" before Old and New in execution section!

PARIS PANTHÉON-ASSAS UNIVERSITÉ

:Old and :New Attribut

Example 1: create a trigger that updates the class ta inserting a new student.

Student (Id_Etu, Name, ..., Id_Classe)

Class (Id_Classe, Nbr_Etu)

PARIS PANTHÉON-ASSAS UNIVERSITÉ

:Old and :New Attribut

Example 1: create a trigger that updates the class ta inserting a new student.

Student (Id_Etu, Name, ..., Id_Classe)

Class (Id_Classe, Nbr_Etu)

```
CREATE OR REPLACE TRIGGER MajNbEtud
    AFTER INSERT ON Student
    FOR EACH ROW
BEGIN
    UPDATE Class
        SET Nbr_Etud = Nbr_Etud + 1
        WHERE Id_Classe = : New.Id_Classe;
END ;
/
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

:Old and :New Attribut

Example 2: create a trigger that updates the class ta inserting a new student if he has more than 10 years ol

Student (Id_Etu, Name, ..., Id_Classe)

Class (Id_Classe, Nbr_Etu)

PARIS PANTHÉON-ASSAS UNIVERSITÉ

PARIS PANTHÉON-ASSAS UNIVERSITÉ

:Old and :New Attribut

Example 2: create a trigger that updates the class ta inserting a new student if he has more than 10 years ol

Student (Id_Etu, Name, ..., Id_Classe)

Class (Id_Classe, Nbr_Etu)

```
CREATE OR REPLACE TRIGGER MajNbEtud
        AFTER INSERT ON Student
        FOR EACH ROW
        WHEN New.Age> 20
BEGIN
        UPDATE Class
                SET Nbr_Etud = Nbr_Etud + 1
                WHERE Id_Classe = : New.Id_Classe;
END ;
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

/

:Old and :New Attribut

Example 2: **create a trigger that updates the class ta inserting a new student if he has more than 10 years ol**

Student (Id_Etu, Name, ..., Id_Classe)

Class (Id_Classe, Nbr_Etu)

```
CREATE OR REPLACE TRIGGER MajNbEtud
      AFTER INSERT ON Student
      FOR EACH ROW
      WHEN New.Age> 20                          - In the "WHE
BEGIN
      UPDATE Class
            SET Nbr_Etud = Nbr_Etud + 1
              WHERE Id_Classe = : New.Id_Classe;   - In BEGIN
END ;
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

the inserting, updating and de predicates

- **inserting :**

  True: The trigger is enabled after insertion

  False: Otherwise

- **Updating :**

  True: the trigger is enabled due to an update

  False: Otherwise

- **Deleting :**

  True: the trigger is enabled after a deletion

  False: Otherwise

PARIS PANTHÉON-ASSAS UNIVERSITÉ

the inserting, updating and de
predicates

example:

```
 CREATE OR REPLACE TRIGGER MajNbEtud
      AFTER INSERT OR DELETE ON Student
    FOR EACH ROW
BEGIN
    IF inserting THEN
            UPDATE Class SET Nbr_Etud = Nbr_Etu
            WHERE Id_Cla =:New.Id_Cla;
    END IF ;
    IF Deleting THEN
            UPDATE Class SET Nbr_Etud = Nbr_Etu
            WHERE Id_Cla =:Old.Id_Cla;
    END IF ;
END ;
/
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

part VIII

-

Functions and Procec

Sub-Programs

- **A subprogramm is a PL / SQL set of stat that has a name.**
- **There are two types of sub-programs:**
  - ***The procedures***
  - ***The functions***

efrei

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Sub-Programs

- A procedure is a subprogramm that execu
SQL code and does not return a result.

- A function is a subprogramm which perfo
/ SQL code and returns the results. A fun
not make transactions.

PARIS PANTHÉON-ASSAS UNIVERSITÉ

The procedures

syntax:

```
DECLARE
        ...
        PROCEDURE <Proc_name> [(P1, ..., Pnot)] IS
              [Local declarations]
         BEGIN
              ...
         EXCEPTION
              ...
         END ;
 BEGIN
      / * procedure call
      ....
 EXCEPTION
      ....
 END ;
 /
```

efrei

PARIS PANTHÉON-ASSAS UNIVERSITÉ

The procedures

syntax:

P1 ... Pn have the following syntax:

<Nom_Arg> [IN | OUT | IN OUT] <Type> Where

IN : input Parameter

OUT : Output parameter

IN OUT : Input / Output Parameter

By default the setting is IN

PARIS PANTHÉON-ASSAS UNIVERSITÉ

The procedures

A simple example:

```
create or replace procedure hello
        IS
BEGIN
            dbms_output.put_line('Hello!');
END;
/
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

The procedures

Example 2:

PARIS PANTHÉON-ASSAS UNIVERSITÉ

We're going to develop a procedure named `adjust_salary()` in HR sample database pro
Oracle. We'll update the salary information of employees in the `employees` table by using
statement.

The following is the source code of the `adjust_salary()` procedure :

```
CREATE OR REPLACE PROCEDURE adjust_salary(
    in_employee_id IN EMPLOYEES.EMPLOYEE_ID%TYPE,
    in_percent IN NUMBER
) IS
BEGIN
   -- update employee's salary
   UPDATE employees
   SET salary = salary + salary * in_percent / 100
   WHERE employee_id = in_employee_id;
END;
```

How it works.

- The procedure has two parameters: `IN_EMPLOYEE_ID` and `IN_PERCENT`.
- The procedure adjusts the salary of a particular employee specified the `IN_EMPLOYEE`
  given percentage `IN_PERCENT.`
- In the procedure body, we use the `UPDATE` statement to update the salary informati

The procedures

Calling PL/SQL Procedure

A procedure can call other procedures. A procedure without parameters can be calle
`EXEC` statement or `EXECUTE` statement followed by the name of the procedure as

```
EXEC procedure_name();
EXEC procedure_name;
```

A procedure with parameters can be called by using `EXEC` or `EXECUTE` statement fo
procedure's name and its parameters in the order corresponding to the parameters
procedure as shown below:

```
EXEC procedure_name(param1,param2…paramN);
```

Now, we can call `adjust_salary()` procedure as the following statements:

```
-- before adjustment
SELECT salary FROM employees WHERE employee_id = 200;
-- call procedure
exec adjust_salary(200,5);
-- after adjustment
SELECT salary FROM employees WHERE employee_id = 200;
```

The functions

Syntax

```
DECLARE
        [Global declarations]
        FUNCTION <Nom_fonc> [(P1 ... Pn)] RETURN Typ
            IS [Local declarations]
        BEGIN
            ...
              RETURN value;
        EXCEPTION
            ...
        END ;
 BEGIN
     / * Call a function
     ....
 EXCEPTION
     ....
 END ;
 /
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

The functions

example:

We are going to create a function named `try_parse` that parses a string and retur
input string is a number or `NULL` if it cannot be converted to a number.

```
CREATE OR REPLACE FUNCTION try_parse(
    iv_number IN VARCHAR2)
  RETURN NUMBER IS
BEGIN
   RETURN to_number(iv_number);
   EXCEPTION
     WHEN others THEN
        RETURN NULL;
END;
```

The `iv_number` is an `IN` parameter whose data type is `VARCHAR2` so that you ca
the `try_parse()` function.

Inside the function, we used the built-in PL/SQL function named `to_number()` to c
number. If any exception occurs, the function returns NULL in the exception sectio
returns a number.

The functions

Calling PL/SQL Function

The PL/SQL function returns a value so you can use it on the right-hand side of an as
SELECT statement.

Let's create an anonymous block to use the `try_parse()` function.

```sql
SET SERVEROUTPUT ON SIZE 1000000;
DECLARE
  n_x number;
  n_y number;
   n_z number;
BEGIN
   n_x := try_parse('574');
   n_y := try_parse('12.21');
   n_z := try_parse('abcd');

   DBMS_OUTPUT.PUT_LINE(n_x);
   DBMS_OUTPUT.PUT_LINE(n_y);
   DBMS_OUTPUT.PUT_LINE(n_z);
END;
/
```

The stored procedures and fu

- Are PL / SQL blocks that have names.
- Are used to store the PL / SQL block con into the database (CREATE).
- Can be reused without being recompiled (EXECUTE).
- Can be called from any block PL / SQL.
- May be grouped together in a package.

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Stored Procedures

syntax:

```
CREATE [ OR REPLACE ] PROCEDURE <Proc_name> [(P1,
    Pn)] IS [Local variables Statements]
    BEGIN
    ...
    EXCEPTION
    ...
    END ;
/
```

→ Created Procedure → *The procedure is correct*

→ or

→ **Procedure Created with compilation errors** → **Correct th**

SHOW ERRORS;

Stored Procedures

example:

```
CREATE [ OR REPLACE ] PROCEDURE
AjoutProd (PrefPro Prod.RefPro%TYPE,…, PPriUni Prod.PriUn
PErr OUT Number) IS

     BEGIN
        INSERT  INTO Prod VALUES(PrefPro,…,PPriUni) ;
        COMMIT ;
        PErr :=0 ;

     EXCEPTION
         WHEN OTHER THEN
         PErr:=1;
        END ;
   /
```

Call stored procedure

syntax:

The stored procedure is called by applications:

- Using his name in a PL / SQL block (another procedure
- By EXECUTE in SQL * Plus.

· In a PL / SQL block:

```
BEGIN
  <Procedure-name> [<P1> ... <Pn>];
END ;
```

· Under SQL * PLUS:

```
EXECUTE <Procedure-name> [<P1> ...
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Call stored procedure

example:

```
ACCEPT VRefPro                  - request a value to the u
ACCEPT VPriUni
......
DECLARE
  VErr NUMBER;
BEGIN
  AjoutProd(VRefPro &, ..., & VPriUni, VErr);
  IF  VErr = 0 THEN
       DBMS_ DBMS_ OUTPUT.PUT_LINE (‘Operation
  Performed ');
       ELSE DBMS_ DBMS_ OUTPUT.PUT_LINE ( 'error'
  END IF ;
END ;
/
```

Stored Functions

syntax:

```
CREATE [ OR REPLACE ] FUNCTION <Nom_Fonc> [(P1 ...
    Pn)] RETURN Type IS
    [Local variables Statements]
    BEGIN
       SQL and PL / Sql
       RETURN (Value)
    EXCEPTION
       Handling Exceptions
    END ;
/
```

→ Created function → The function is correct

or

→ Created with function compilation errors → Correct the

PARIS PANTHÉON-ASSAS UNIVERSITÉ

SHOW ERRORS;

Stored Functio

example:

```
CREATE [ OR REPLACE ]
FUNCTION NbEmp (PNumDep Emp.Dept_Id% Type, PErr OU
RETURN Number IS
        VNB Number (4);

      BEGIN
         SELECT Count (*) INTO VNB FROM Emp WHERE Dept_id =
         PErr: 0 =
         RETURN VNB;

     EXCEPTION
         WHEN NO_DATA_FOUND THEN
          PErr: = 1;
          RETURN null;

      END ;
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Call stored functio

syntax:

The stored function is called by applications is:

- **In an expression in a PL / SQL block.**
- **With the cmd EXECUTE (In SQL * PLUS**

· **In a PL / SQL block:**

```
BEGIN
        <Var>: <Function name> [<P1> ... <Pn>]
END ;
```

· **Under SQL * PLUS:**

```
EXECUTE <Var>: <Function name> [<P1> ... <
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Call stored functio

example:

```
ACCEPT VDep
DECLARED
   VErr Number;
   VNB Number (4);
BEGIN
   VNB: = NbEmp(& VDep, VErr);
   IF VErr = 0 THEN
     Dbms_output.put_line ( 'The number of employees is' |
   ELSE
     Dbms_output.put_line ( 'error');
   END IF ;
END ;
```

efrei

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Call stored functio

example:

```
SQL> VARIABLE VNB
SQL> EXECUTE :VNB: =NbEmp(& VDep, VErr);
PL / SQL procedure successfully completed.
SQL> PRINT VNB

      VNB
----------
       300
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Call stored functio

example:

```
SQL> VARIABLE VNB
SQL> EXECUTE :VNB: =NbEmp(& VDep, Caps);
PL / SQL procedure successfully completed.
SQL> PRINT VNB

      VNB
----------
       300
```

The VARIABLE and PRINT cmd are used to declare va (Bind Variables) and display their values in SQL * Plus

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Delete stored procedures functions

syntax:

```
DROP PROCEDURE      procname;
DROP FUNCTION       functionname;
```

stored procedures and fun

Useful commands:

- SELECT object_name, object_type FROM o
- DESC procname
- DESC functionname

PARIS PANTHÉON-ASSAS UNIVERSITÉ

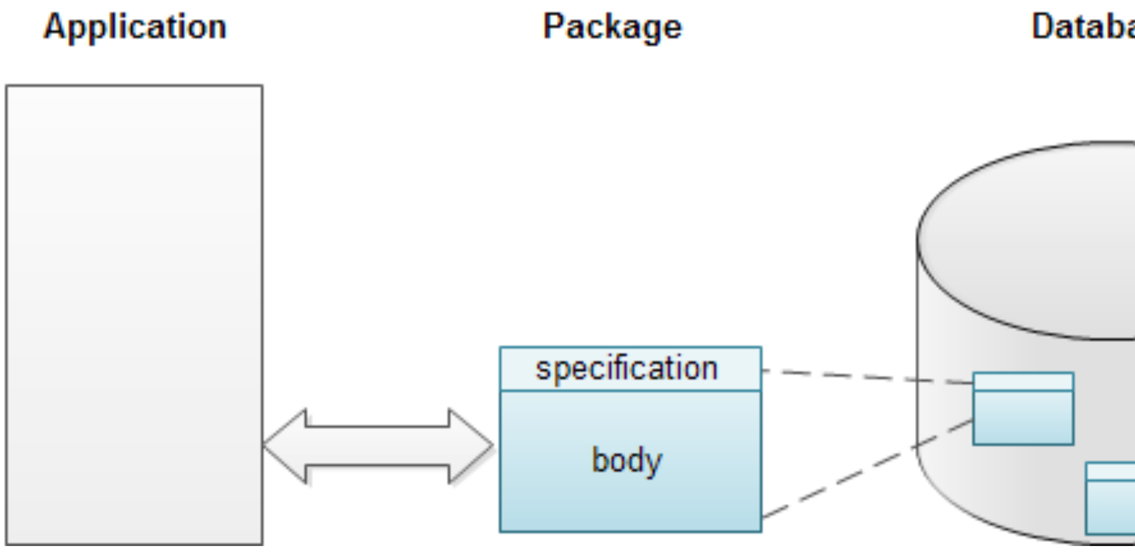
packages

- A PL / SQL object that stores other types of objects: procedures, functions, cursors, varia
- Consists of two parts:
  - Specification (declaration)
  - Body (implementation)
- Can not be called or set or nested
- Allows Oracle to read multiple objects at or memory

packages

A PL/SQL package has two parts: package specification and package body.

- A package specification is the public interface of your applications. The pub function, procedures, types, etc., are accessible from other applications.
- A package body contains the code that implements the package specificati


Application
Package
Databa
specification
body

Create Package: Specifica

```
CREATE [OR REPLACE] PACKAGE <Nom_Package>
   IS [Déclaration des variables et Types
      [Déclaration des curseurs]
      [Déclaration des procédures et fonc
      [Déclaration des exceptions]
END[<Nom_Package>];
/
```

Example of Specificati

```
Create Or Replace Package PackProd
    Is Cursor CProd Is Select RefPro, D
            From Produit;
    Procedure AjoutProd(PrefPro Prod.Re
                                ...., PErr Ou
    Procedure ModifProd(PrefPro Prod.Re
                                ..., PErr Ou
    Procedure SuppProd(PrefPro Prod.Ref
                                ...,PErr Out
    Procedure AffProd;
EndPackProd;
/
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Create Package: the bo

```
Create [Or Replace] Package Body <Nom_Pa
          [Implémentation procédures | fonc
End [<Nom_Package>];
/
```

efrei

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Example body

```
Create Or Replace Package Body PackProd
Is
Procedure AjoutProd (PrefPro Prod.RefPro%Type,
                          ..., PErr Out Number)
Is
Begin
        Insert Into Prod Values(PrefPro,..., PI
        Commit;
        PErr:=0;
Exception
       When Dup_Val_On_Index Then   PErr:=1;
       When Others Then   PErr:= 1;
End;

Procedure ModifProd (PrefPro Prod.RefPro%Type,
                          ..., PErr Out Number)
      Is B Boolean;
Begin
      ...
EndPackProd;
/
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Using the package

```
<NomPackage>.<NomProcedure>[(Paramètr
Var:= <NomPackage>.<NomFonction>[(Para
```

efrei

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Using the package

example:

```
Accept VRef Prompt '......';
Accept VPri Prompt '......';
Declare
        VErr Number;
Begin
      PackProd.ModifProd(&VRef, ..., &VPr
      If VErr= 0 Then
         DBMS_Output.Put_Line('Traitement
      Else
         DBMS_Output.Put_Line('Erreur');
      End If;
End;
/
```

Part IX

-

Transaction Contr

COMMIT and ROLLBA

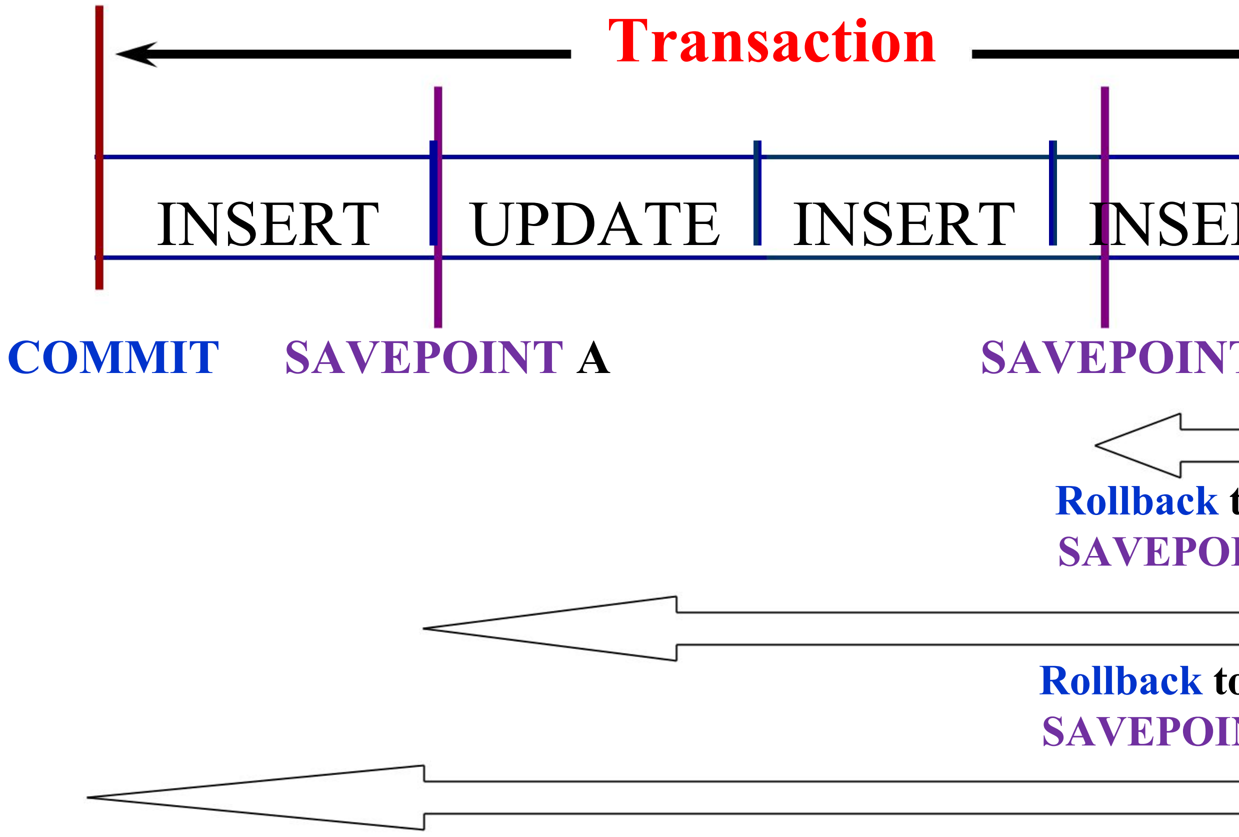
- A transaction begins with the first SQL com
  COMMIT or a Rollback.

- use COMMIT or the Rollback to complete a
  transaction.

- COMMIT apply all that has been done in th
  Rollback cancels all operations.

PARIS PANTHÉON-ASSAS UNIVERSITÉ

The ROLLBACK



efrei

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Transaction Control

Analyse this PL / SQL block:

```
BEGIN
    INSERT INTO temp VALUES (1, 1 'ROW 1');
    SAVEPOINT at;
    INSERT INTO temp VALUES (2, 2 'ROW 2');
    SAVEPOINT b;
    INSERT INTO temp VALUES (3, 3 'ROW 3');
    SAVEPOINT c;
    Rollback TO SAVEPOINT b;
    COMMIT;
END;
```

PARIS PANTHÉON-ASSAS UNIVERSITÉ

PARIS PANTHÉON-ASSAS UNIVERSITÉ

ALTER & DROP table DDL Execute Immediate of Dy SQL

Video

More information

https://www.youtube.com/playlist?list=PLL_L
xzq9GKwORoH6nvaRnOQ

https://www.w3schools.com/sql/default.asp

In French:
http://didier.deleglise.free.fr/plsql/plsql_paza