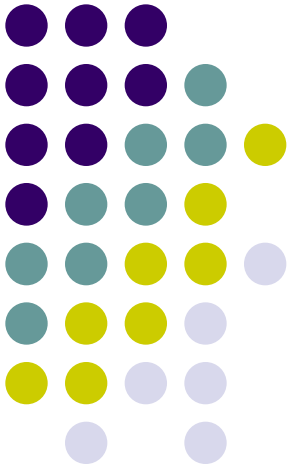
Applied Faculty
Hama University

Database 2

PL / SQL



مقدمة

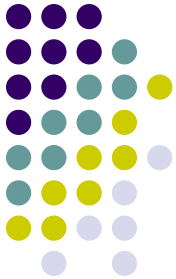
PL/SQL:

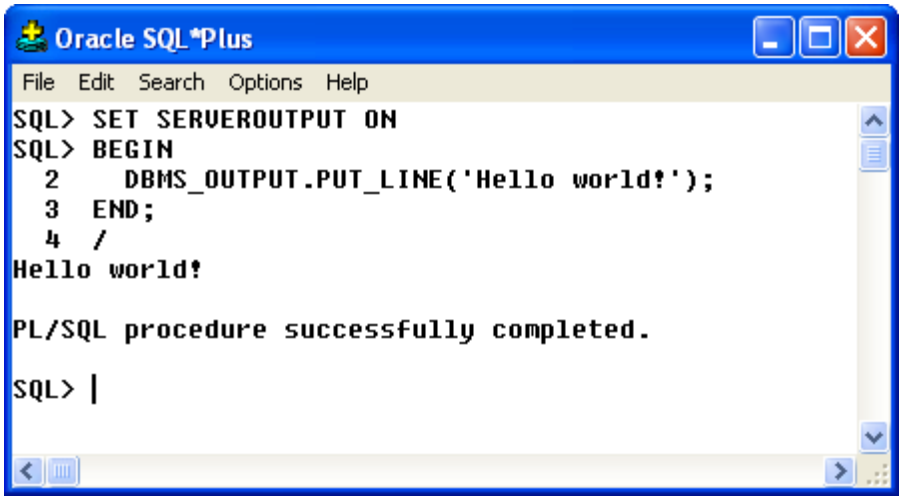
Procedural **L**anguage for **S**tructured **Q**uery **L**anguage

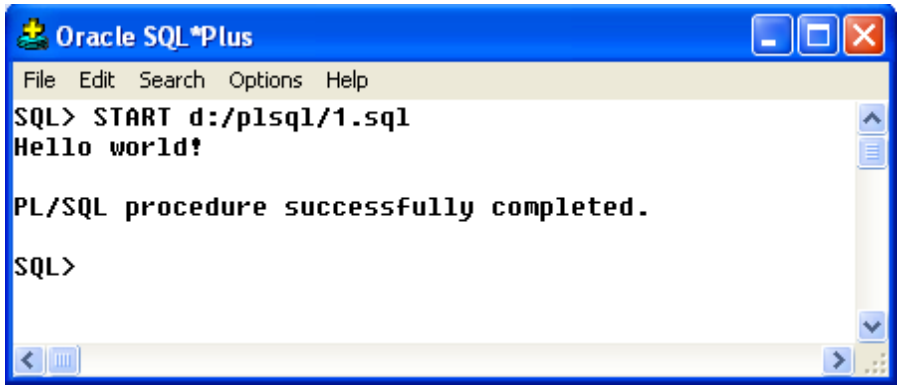
- PL/SQL هي لغة برمجة إجرائية في Oracle
- يمكن تضمين تعابير SQL مباشرة في برامج PL/SQL
- يمكن لتعابير SQL أن تحوي متحولات PL/SQL في الدخل أو الخرج

- تعابير SELECT التي تعيد عدة أسطر يجب التعامل معها بشكل مختلف

Hello World

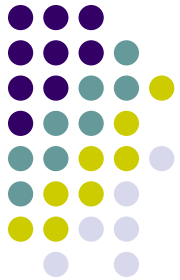




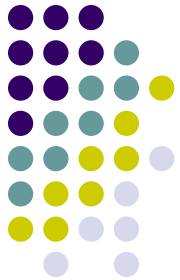


- يمكن كتابة الكتل البرمجية مباشرة في SQL*Plus
- أو كتابتها في ملف خارجي ثم تنفيذها في SQL*Plus باستخدام الكلمة المفتاحية START

أجزاء البرنامج

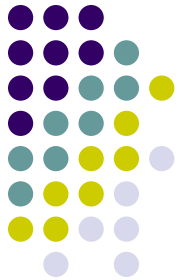


DECLARE *<variable declarations>*	مقطع التصريحات
BEGIN *<executable statements>*	المقطع التنفيذي
EXCEPTION *<exception handling>*	معالجة الاستثناءات
END;	نهاية البرنامج



مقطع التصريحات

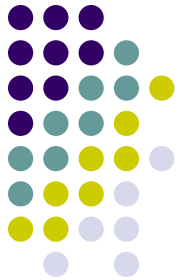
- مقطع اختياري في كتلة البرنامج
- يبدأ بالكلمة المفتاحية DECLARE
- يحتوي على تعاريف الأغراض المحلية التي ستستخدم في البرنامج
  - المتحولات
  - المؤشرات
  - الاستثناءات



المقطع التنفيذي

- المقطع الإجباري الوحيد في كتلة البرنامج
- يبدأ بالكلمة المفتاحية BEGIN
- يحتوي على التعليمات التي سيتم تنفيذها في البرنامج
  - تعليمات DML
  - إجرائيات
  - توابع

مثال

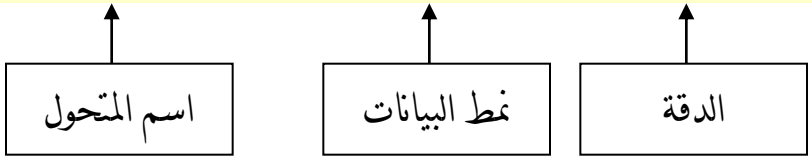


```
DECLARE
     Pi CONSTANT NUMBER(9,7) := 3.1415926;
     radius INTEGER;
     area   NUMBER(14,2);
BEGIN
     radius := 3;
     area := Pi * Power(radius,2);
     INSERT INTO areas VALUES (radius, area);
END;
/
```

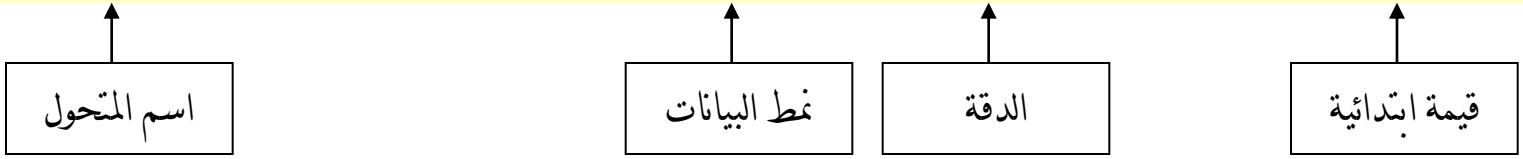
```
SQL> start d:/plsql/3.sql;

PL/SQL procedure successfully completed.

SQL> select * from areas;

    RADIUS       AREA
---------- ----------
         3      28.27
```

تعريف متحول

```
variable_name datatype(precision);
```

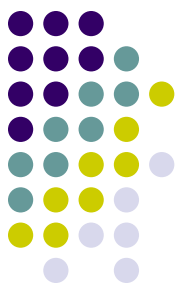



```
variable_name [CONSTANT] datatype(precision) [ := init_value];
```




```
Pi CONSTANT NUMBER(9,7) := 3.1415926;
radius INTEGER := 3;
area   NUMBER(14,2);
```

%TYPE

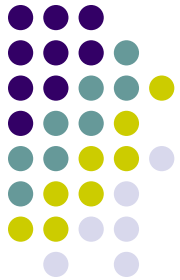


- يمكن تعريف متحول بنفس نمط بيانات عمود جدول أو متحول من نوع مؤشر بدون معرفة نمط بيانات العمود أو المؤشر

```
variable_name table_name.column_name%TYPE;
```

```
lname employee.last_name%TYPE;
```

%ROWTYPE



- يمكن تعريف مصفوفة/نسق من المتحولات مبنية على الأعمدة الموجودة في جدول ما أو مؤشر ما
- يتم إنشاء متحول لكل عمود في الجدول/ المؤشر المحدد بنفس الاسم و النمط

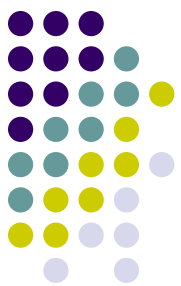
```
array_name table/cursor_name%ROWTYPE;
```

```
Dept_var Department%ROWTYPE;

Dept_var.department_name
...
```

التعليقات

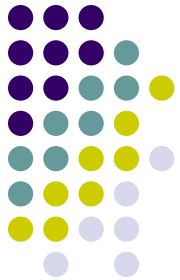
```
/* This is a comment */
-- This is also a comment
```

SELECT / INTO



```
DECLARE
    min_salary NUMBER;
    max_salary NUMBER;
BEGIN
    SELECT MIN(salary), MAX(salary)
    INTO min_salary, max_salary
    FROM employee;
    DBMS_OUTPUT.PUT_LINE('Smallest salary is: '||min_salary);
    DBMS_OUTPUT.PUT_LINE('Biggest  salary is: '||max_salary);
END;
/
```

يمكن استخدامها فقط عندما تعيد SELECT سطراً واحداً فقط



المؤشرات Cursors

- المؤشر هو أداة تستخدم لاسترجاع مجموعة من السجلات من جدول/منظور إلى الذاكرة.
- تسمح المؤشرات للبرنامج بقراءة كل سجل من السجلات و معالجتها

- التصريح عن مؤشر

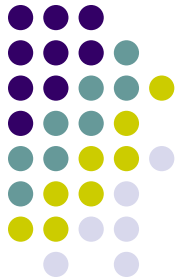
```
CURSOR cursor_name IS select_statment;
```

```
CURSOR emp_cursor IS
        SELECT first_name, last_name FROM employee;
```

أوامر المؤشرات

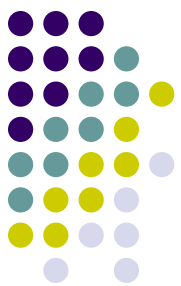
الأمر	مثال	الوصف
OPEN	**OPEN** cursor_name;	فتح المؤشر، تنفيذ تعليمة SELECT و وضع السجلات في الذاكرة
FETCH INTO	**FETCH** cursor_name **INTO** variables;	إسناد القيم من السجل الحالي للمؤشر إلى قائمة المتحولات ، جعل السجل التالي هو السجل الحالي
CLOSE	**CLOSE** cursor_name;	إنهاء المؤشر و تحرير الذاكرة

مثال



```
DECLARE
    Pi CONSTANT NUMBER(9,7) := 3.1415926;
    area   NUMBER(14,2);
    CURSOR rad_cursor IS
        SELECT * FROM radius_vals;
    rad_val rad_cursor%ROWTYPE;
BEGIN
    OPEN rad_cursor;
    FETCH rad_cursor INTO rad_val;
    area := Pi * Power(rad_val.radius,2);
    INSERT INTO areas VALUES (rad_val.radius, area);
    CLOSE rad_cursor;
END;
/
```

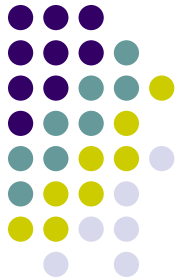
التعابير الشرطية IF/THEN



```
IF <some condition> THEN
    <some command>
ELSIF <some condition> THEN
    <some command>
ELSE
    <some command>
END IF;
```

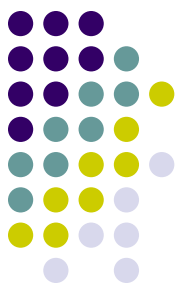
```
IF <some condition> THEN
    IF <some condition> THEN
        <some command>
    END IF;
ELSE
    <some command>
END IF;
```

التعابير الشرطية IF/THEN – مثال



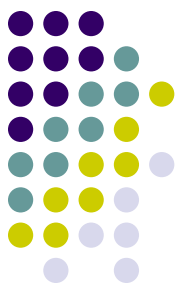
```
DECLARE
     Pi CONSTANT NUMBER(9,7) := 3.1415926;
     area   NUMBER(14,2);
     CURSOR rad_cursor is
          SELECT * FROM radius_vals;
     rad_val rad_cursor%ROWTYPE;
BEGIN
     OPEN rad_cursor;
     FETCH rad_cursor INTO rad_val;
     area := Pi * Power(rad_val.radius,2);
     IF area > 30 THEN
       INSERT INTO areas VALUES (rad_val.radius, area);
     END IF;
     CLOSE rad_cursor;
END;
/
```

التعابير الشرطية CASE/WHEN



```
CASE variable
  WHEN value1 THEN expression1;
  WHEN value2 THEN expression2;
  ...
  ELSE expression;
END CASE;
```

التعابير الشرطية CASE/WHEN – مثال

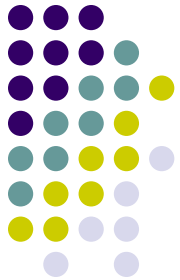


```
DECLARE
   I INTEGER := 3;
BEGIN
   CASE I
     WHEN 1 THEN DBMS_OUTPUT.PUT_LINE('ONE');
     WHEN 2 THEN DBMS_OUTPUT.PUT_LINE('TOW');
     WHEN 3 THEN DBMS_OUTPUT.PUT_LINE('THREE');
     WHEN 4 THEN DBMS_OUTPUT.PUT_LINE('FOUR');
     WHEN 5 THEN DBMS_OUTPUT.PUT_LINE('FIVE');
     ELSE        DBMS_OUTPUT.PUT_LINE('?');
   END CASE;
END;
/
```

الحلقات

- LOOP
- FOR
- WHILE

الحلقات البسيطة

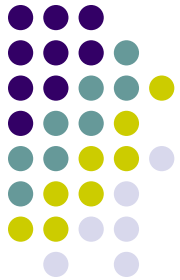


```
LOOP
   <executable statements>
END LOOP;
```

```
LOOP
   ...
   IF <stop_condition> THEN EXIT;
   ...
END LOOP;
```

```
LOOP
   ...
  EXIT WHEN <stop_condition>;
   ...
END LOOP;
```

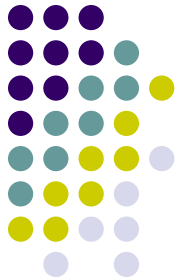
الحلقات البسيطة – مثال



```
DECLARE
     Pi CONSTANT NUMBER(9,7) := 3.1415926;
     radius INTEGER;
     area   NUMBER(14,2);
BEGIN
     radius := 3;
     LOOP
       area := Pi * Power (radius, 2);
       INSERT INTO areas VALUES (radius, area);
       radius := radius + 1;
       EXIT WHEN area > 100;
     END LOOP;
END;
/
```

```
SQL> select * from areas;

    RADIUS       AREA
---------- ----------
         3      28.27
         4      50.27
         5      78.54
         6      113.1
```

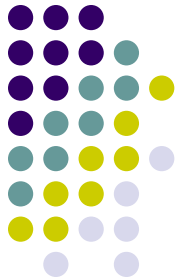
الحلقات البسيطة مع المؤشرات



- Cursor properties
  - %FOUND
  - %NOTFOUND
  - %ISOPEN
  - %ROWCOUNT

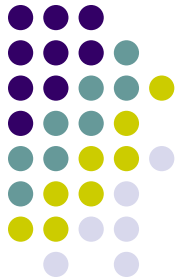
- **EXIT WHEN** <cursor>**%NOTFOUND;**

الحلقات البسيطة مع المؤشرات



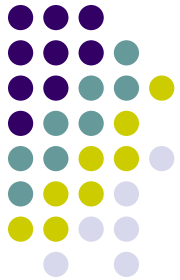
```
OPEN <cursor>;
LOOP
   FETCH <cursor> INTO <row_variable>;
   EXIT WHEN <cursor>%NOTFOUND;
   ...
END LOOP;
CLOSE <cursor>;
```

الحلقات البسيطة مع المؤشرات – مثال



```
DECLARE
     Pi CONSTANT NUMBER(9,7) := 3.1415926;
     area   NUMBER(14,2);
     CURSOR rad_cursor IS
          SELECT * FROM radius_vals;
     rad_val rad_cursor%ROWTYPE;
BEGIN
     OPEN rad_cursor;
     LOOP
       FETCH rad_cursor INTO rad_val;
       EXIT WHEN rad_cursor%NOTFOUND;
       area := Pi * Power(rad_val.radius,2);
       INSERT INTO areas VALUES (rad_val.radius, area);
     END LOOP;
     CLOSE rad_cursor;
END;
/
```

حلقات FOR

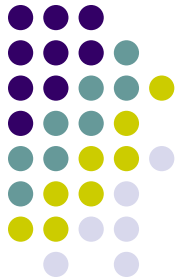


```
FOR <counter> IN [REVERSE] 1..n LOOP
   <executable statements>
END LOOP;
```

```
FOR <row_variable> IN <cursor> LOOP
   <executable statements>
END LOOP;
```

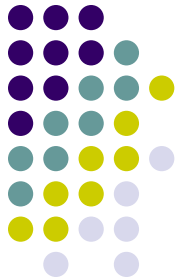
لا داعي لوجود تعليمات **OPEN** و **FETCH/INTO** و **CLOSE**

حلقات FOR



```
DECLARE
     Pi CONSTANT NUMBER(9,7) := 3.1415926;
     radius INTEGER;
     area   NUMBER(14,2);
BEGIN
     FOR radius IN 1..7 LOOP
       area := Pi * Power (radius, 2);
       INSERT INTO areas VALUES (radius, area);
     END LOOP;
END;
/
```

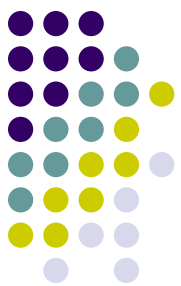
حلقات FOR مع المؤشرات



```
DECLARE
     Pi CONSTANT NUMBER(9,7) := 3.1415926;
     area   NUMBER(14,2);
     CURSOR rad_cursor IS
          SELECT * FROM radius_vals;
     rad_val rad_cursor%ROWTYPE;
BEGIN
     FOR rad_val IN rad_cursor LOOP
       area := Pi * Power (rad_val.radius, 2);
       INSERT INTO areas VALUES (rad_val.radius, area);
     END LOOP;
END;
/
```

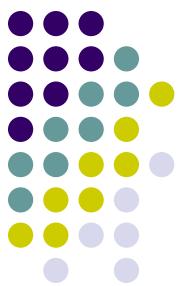
لا توجد تعليمات **OPEN** و **FETCH/INTO** و **CLOSE**

حلقات WHILE



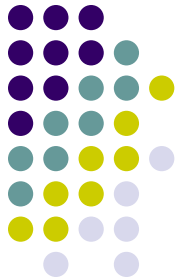
```
WHILE <condition> LOOP
    <executable statements>
END LOOP;
```

حلقات WHILE



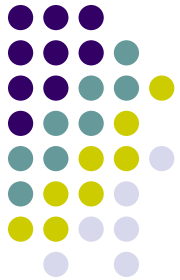
```
DECLARE
     Pi CONSTANT NUMBER(9,7) := 3.1415926;
     radius INTEGER;
     area   NUMBER(14,2);
BEGIN
     radius := 3;
     WHILE radius <= 7 LOOP
       area := Pi * Power (radius, 2);
       INSERT INTO areas VALUES (radius, area);
       radius := radius + 1;
     END LOOP;
END;
/
```

حلقات WHILE مع المؤشرات



```
DECLARE
     Pi CONSTANT NUMBER(9,7) := 3.1415926;
     area   NUMBER(14,2);
     CURSOR rad_cursor IS
          SELECT * FROM radius_vals;
     rad_val rad_cursor%ROWTYPE;
BEGIN
     OPEN rad_cursor;
     FETCH rad_cursor INTO rad_val;
     WHILE rad_cursor%FOUND LOOP
       area := Pi * Power(rad_val.radius,2);
       INSERT INTO areas VALUES (rad_val.radius, area);
       FETCH rad_cursor INTO rad_val;
     END LOOP;
     CLOSE rad_cursor;
END;
/
```

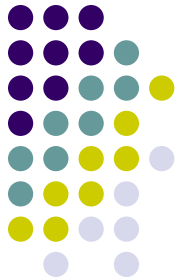
معالجة الاستثناءات



```
DECLARE
     Pi CONSTANT NUMBER(9,7) := 3.1415926;
     radius INTEGER;
     area   NUMBER(14,2);
     some_variable NUMBER(14,2);
BEGIN
     radius := 3;
     LOOP
       some_variable := 1/(radius-4);
       area := Pi * Power (radius, 2);
       INSERT INTO areas VALUES (radius, area);
       radius := radius + 1;
       EXIT WHEN area > 100;
     END LOOP;
END;
/
```

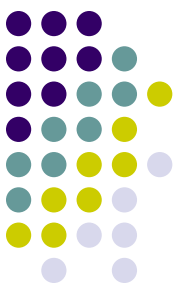
```
DECLARE
*
ERROR at line 1:
ORA-01476: divisor is equal to zero
ORA-06512: at line 9
```

معالجة الاستثناءات



```
DECLARE
     Pi CONSTANT NUMBER(9,7) := 3.1415926;
     radius INTEGER;
     area   NUMBER(14,2);
     some_variable NUMBER(14,2);
BEGIN
     radius := 3;
     LOOP
       some_variable := 1/(radius-4);
       area := Pi * Power (radius, 2);
       INSERT INTO areas VALUES (radius, area);
       radius := radius + 1;
       EXIT WHEN area > 100;
     END LOOP;
EXCEPTION
     WHEN ZERO_DIVIDE THEN
       INSERT INTO areas VALUES (0, 0);
END;
/
```

معالجة الاستثناءات

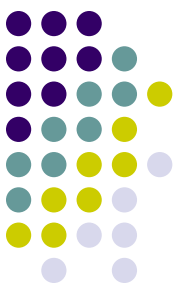


```
EXCEPTION
  -- Generic error handling to handle any type of error
  WHEN OTHERS THEN
    -- print out an error message
    RAISE_APPLICATION_ERROR (-20100,
        'Error#' || sqlcode || ' Desc: ' sqlerrm);
```

Oracle error number

Oracle error message

الاستثناءات الشائعة



Exception Name	Explanation	Oracle Error
NO_DATA_FOUND	When a select statement returns no rows, this error may be raised. It usually occurs when you use an implicit cursor and perform a SELECT INTO.	ORA-01403
TOO_MANY_ROWS	When a case that should only return a single row returns multiple rows, this exception is raised.	ORA-01422
DUP_VAL_ON_INDEX	This exception is raised when you try to insert a record into a table that has a primary key on it and the record that you are inserting is a duplicate of one that already exists in the table.	ORA-00001
VALUE_ERROR	This error occurs when you attempt to put a value into a variable, but the value is either incompatible or you input a value that is too big.	ORA-06502
ZERO_DIVIDE	This error is encountered when you attempt to divide by zero.	ORA-01476
OTHERS	This exception is used to catch any errors not handled by specific error handles.	Non-specific