الأكاديمية العربية الدولية

الأكاديمية العربية الدولية

المقررات الجامعية

www.aiacademy.info | care@aiacademy.info

PLSQL

جامعة السودان للعلوم والتكنولوجيا

نظم معلومات إدارية

إعداد :: الاء عبدالمنعم حمدنالله

alaa2401@hotmail.com

لغة plsql هي عبارة عن تطوير للغة الاسترجاع sql
وهي إختصار ل
(Programming Language Structured Query Language)
قامت شركة Oracle بعمل هذا التطوير لأعطاء لغة ال sql المزايا اللازمة لمواكبة متطلبات البرمجة الحديثة .
مثلاً لغة ال plsql تستطيع ان تتعامل مع أكثر من امر من اوامر sql فيمكن ان نقوم بعملية استرجاع للبيانات بواسطة select وايضا عملية مسح بواسطة delete
في نفس اللحظة اما لغة الsql لاتسمح بذلك فهي تسمح باستخدام جملة واحدة اي يتم استرجاع البيانات أولا ثم بعد ذلك نقوم بعملية المسح
إذاً يمكن تعريف لغة الplsql بأنها لغة إسترجاع او لغة هيكلية تعني ان البرنامج يمكن تقسيمه الى أجزاء ويتم التعامل مع برنامج الplsql كوحدة برمجية

يمكن تقسيم الوحدات البرمجية الى:

وحدة برمجية مسماة: Named Block

وهي وحدة برمجية لها اسم وهذا النوع من الوحدات البرمجية يمكن تخزينه في قاعدة البيانات والرجوع إليها واستخدامها عن الحاجة كما يمكن إستخدامها بواسطة الوحدات البرمجية الاخرى المخزنة معها في قاعدة البيانات ومن أمثلة هذا النوع Function, Procedure, Trigger, Package

وحدة برمجية غير مسماة: Anonymous Block

هي وحدة برمجية ليس لها اسم وهذا النوع من الوحدات البرمجية لايمكن تخزينه في قاعدة البيانات

تتكون الوحدة البرمجية من ثلاثةأجزاء:

1- جزء الاعلان :Declarative

هذا الجزء يتم فيه تعريف المتغيرات التي سيتم إستخدامها في الوحدة البرمجية وايضا يتم فيه تعريف المؤشر و الإستثناءات وهذا الجزء اختياري اي يمكن كتابة وتنفيذ وحدة لاتحتوي على متغيرات

2- جزء التنفيذ :Executable

هذا الجزء يحتوي على جمل ال sqlالتي تتعامل مع البيانات الموجودة في قاعدة البيانات وايضا تحتوي على جمل ال plsql التي تتعامل مع البيانات الموجودة في الوحدة البرمجية وهذا الجزء اجباري لانه يحتوي على الجمل الواجب تنفيذها

3-جزء الاستثناءات:Exception

هذا الجزء تتم فيه معالجة الأخطاء المحتمل حدوثها في جزء التنفيذ وذلك عن طريق بيان الإجراء الواجب عمله عند حدوث الخطأ

المتغيرات

هي عبارة عن مواقع في الذاكرة يتم استخدامها للتخزين المؤقت للبيانات اثناء تنفيذ الوحدة البرمجية

يوجد لدينا نوعين من المتغيرات:

1- متغيرات متغيرة:

وتعني ان المتغير يمكن تغير قيمته اكثر من مرة داخل الوحدة البرمجية

2-متغيرات ثابتة :

وتعني ان المتغير يحتوي على قيمة ثابتة لايمكن ان تتغير

فوائد المتغيرات :

1- معالجة البيانات المخزنة:

يمكن استخدام المتغيرات لتحتوي على القيم المخزنة في قاعدة البيانات و بالتالي يمكن استخدامها في العمليات الحسابية دون الرجوع إلى قاعدة البيانات

2-إعادة الاستخدام

عند تعريف متغير يتم حجز مكان في الذاكرة و بالتالي يمكن تخزين و استرجاع البيانات في ومن هذ ا المكان أكثر من مرة خلال عملية تنفيذ البرنامج

3- سهولة الصيانة:

عند تعريف متغير بناءًا على عمود موجود في قاعدة البيانات فعند تغير نوع العمود لانحتاج لإعادة تعريف المتغير وهذا يساعد في عملية التعديل والصيانة

الشكل العام لتعريف المتغير:

identifier [CONSTANT] data type [NOT NULL] [:= DEFAULT | expression] ;

اسم المتغير identifier

CONSTANT بمثابة قيد على المتغير لها شرطان:

اولًا المتغير المعرف بواسطة CONSTANT لابد ان يتم اعطاءه قيمة إبتدائية

ثانياً هذه القيمة ثابتة لايمكن ان تتغير اذا لم يتم هنا اعطاء المتغير قيمة ابتدائية سيؤدي ذلك الى خطأ

data type نوع البيانات التي سيتم تخزينها في قاعدة البيانات

NOT NULL ايضا هي قيد على المتغير تعني المتغير لابد ان يتم اعطاءه قيمة ابتدائية ولكن هذه القيمة قابلة للتغير اثناء تنفيذ الوحدة

Expression قيمة المتغير

أمثلة:

```
Declare
v_name VARCHAR2(10) ;
v_no NUMBER(2) NOT NULL:=11;
V_salary CONSTANT NUMBER:=2000;
BEGIN
.....
End;
```

النقاط التي يجب مراعاتها في تسمية المتغيرات:

1-يجب ان لايكون هنالك متغيرين بنفس الاسم في وحدة برمجية واحده

2- يجب ان لايتشابه اسم المتغير مع اسم عمود أو اسم جدول سيستخدم معه في الوحدة البرمجية

3-ان لايزيد طول الاسم عن 30 حرف

4-يمكن استخدام الرموز المتاحة في لغة ال sql في تسمية المتغير مثل _, #,$

5-يجب ان يبدأ المتغير بحرف

6-يجب ان لايكون اسم المتغير كلمة محجوزة

7-يجب وضع قيمة إبتدائية للمتغيرات التي يتم تعريفها بواسطة CONSTANTأو NOT NULL فعند عدم وضع قيمة سيحصل خطأ

8- يتم اسناد القيم للمتغير بواسطة الرمز := أو باستخدام الكلمة المحجوزة default

اسناد القيمة للمتغير:

يمكن اسناد القيمة للمتغير بإستخدام جملة الإسناد التالية:

وفيها يتم كتابة اسم الممتغير ثم الرمز := ثم كتابة القيمة

```
Identifier := expression ;
```

مثل:

```
v_name :='Ali' ;
```

أو باستخدام الكلمة المحجوزة default

مثل :

```
v_name  default 'Ali' ;
```

ايضاً يمكن اسناد القيمة للمتغير خلال جملة الاسترجاع من قاعدة البيانات وهنا لابد من التأكد من ان جملة الاسترجاع تعيد قيمة واحده وإلا سيؤدي ذلك الى خطأ

مثل :

```
SELECT ename INTO v_name FROM empWHERE
empno = 3;
```

ايضا يمكن اسناد القيمة للمتغير خلال الاسترجاع من المؤشر cursor

مثل:

```
fetch c into v_id,v_fname,v_lname,v_sal;
```

انواع البيانات للمتغيرات:Datatypes

1- المتغيرات المفردة:

وهي التي تحتوى على قيمة واحدة

أمثله :

Varchar2(size)
Char(size)
Number
Date
Long
Long Raw
Boolean
Binary_Integer
PLS_Integer

أمثلة:

```
v_count BINARY_INTEGER := 0;
v_total_sal NUMBER(9,2) := 0;
v_heirdate DATE := SYSDATE;
v_valid BOOLEAN NOT NULL := TRUE;
```

المتغير المعرف بواسطة BOOLEAN تكون قيمته :
TRUE أو FLASE أو NULL

2-المتغيرات المركبة :

تحتوى على مجموعة من الأجزاء كل جزء ذا تركيبة مختلفة عن الاخر ويتم التعامل مع كل جزء على حده

مثل :السجلات والجداول فهي تتكون من مجموعة من الحقول لايشترط ان تكون من نفس النوع أو الحجم

مثال : يمكن تعريف السجل التالي Employ ليحتوي على اسم(متغير حرفي (char(30) ورقم(متغير رقمي Number) وتاريخ التعين(متغير تاريخي (Date

Employ _Number	Employ _Name	Employ_
3	Ali	12-may-2011

3-المتغيرات التي تحتوي على كائنات كبيرة الحجم:

تحتوي هذه المتغيرات على كائنات كبيرة الحجم مثل الافلام والصور والكتب

4-متغيرات الربط:

هي المتغيرات التي يتم تعريفها داخل البيئة التي يتم فيها تنفيذ الوحدة البرمجية

هذا النوع من المتغيرات يتم تعريفه خارج الوحدة الوحدة البرمجية

SQL > Varible Emp_Number Number

اي ان هذا التعريف يكون قبل Declare

وايضا تتم طباعته خارج الوحدة البرمجية

Print (Emp_Number)

اي ان الطباعة تكون بعد ال End

ويتم استخدامه داخل الوحدة البرمجية مسبوقا بنقطتين علويتين (:)

:Emp_Number:=24;

يمكن ايضا طباعة متغير الربط داخل الوحدة البرمجية باستخدام DBMS_OUTPUT.PUT_LINE

مثال :

DBMS_OUTPUT.PUT_LINE(
:Emp_Number);

جملة الاخراج:

تتم طباعة المتغيرات باستخدام الاجراء PUT_LINE الموجود في الحزمة DBMS_OUTPUT كلاتي :

DBMS_OUTPUT.PUT_LINE(Emp_Name);

ولتمكين الحزمه من العمل نستخدم الامر

SQL >SET SERVEROUTPUT ON

وهذا الامر يكتب في بداية الوحدة البرمجية

أمثلة:

كتابة وحدة برمجية تقوم بطباعة Well Come to PLSQL

```
Begin
DBMS_OUTPUT.PUT_LINE ('Well Come to PLSQL');
End;
```

في الوحدة البرمجية السابقة لم نحتاج لتعريف متغيرات لذلك لم نستخدم جزء التعريف Declare فكما ذكرنا سابقا بأن جزء التعريف هو جزء اختياري .

قم بكتابة وحدة برمجية تقوم بتعديل مرتب الموظف رقم 33 بحيث يتم إضافة 500 للراتب الاصلي وهو 3000

```
DECLARE
v_sal emp.sal%TYPE := 500;
BEGIN
UPDATE emp
SET sal = sal + v_sal
WHERE no = 33;
END;
```

قم بكتابة وحدة برمجية تحتوي على متغيرين num1 و num2 بحيث تقوم الوحدة البرمجية بجمع العددين ويتم وضع الناتج في متغير ربط ومن ثم قم بطباعة متغير الربط (قيمة المتغير num1=4 وقيمة المتغير num2=5)

```
SET SERVEROUTPUT ON
 VARIABLE  Total Number
Declare
 Num1  Number:=4;
 Num2 Number:=2;
Begin
:Total:=Num1+Num2;
End;
/
Print  Total
```

قم بكتابة وحدة برمجية تقوم بطباعة مجموع الرواتب للموظفين العاملين في قسم ال mis

```
SET SERVEROUTPUT ON
Declare
V_DeptName  char(10):='mis';
V_SumSalary  Number (10,5);
Begin
Select Sum(salary)into V_SumSalary from Employ
Where DeptName =V_DeptName;
DBMS_OUTPUT.PUT_LINE('The Sum Salary Is' ||
V_SumSalary);
  End;
```

مكونات جمل ال plsql:

1- المتغيرات :

أمثلة مقبولة

```
v_Job varchr2(20);
xyx number;
Heir_date date;
```

أمثلة غير مقبولة :

```
22id  number ;
```

هنا تعريف المتغير خطأ وذلك لانه يبدأ برقم

```
A,B,C  number(2);
```

هنا تعريف المتغير خطأ وذلك لانه تم تعريف اكثر من متغير

```
Delete varchar2(10);
```

هنا تعريف المتغير خطأ وذلك لأن اسم المتغير كلمة محجوزة

2- القيم الثابتة :

لدينا ثلاثة أنواع من القيم:

1-رقمية :

قد تكون ارقام صحيحة أو ارقام عشرية ويتم كتابتها مباشرة اي انها لاتكتب بين علامات تنصيص مثل v_no:=20;

2- حرفية:

يجب ان تكتب بين علامتي تنصيص مفردة مثل v_job:='manager';

3-تاريخية :

ايضا تكتب بين علامتي تنصيص مفردة

مثل V_heirdate:'12-may-2009'=

3-العمليا ت :

الاس والنفي (**,Not)

الجمع والطرح (+,-)

الضرب والقسمة (*,%)

عمليات المقارنة (=,>,<,<=, <=, IS NULL, LIKE, BETWEEN)

العمليات المنطقية (AND OR)

4- التعليقات :

هي عبارة عن جمل توضح عمل البرنامج ودلالات المتغيرات بحيث تسهل على من يريد استخدام او تعديل البرنامج فهم تركيب وعمل البرنامج

يوجد لدينا نوعين من التعليقات :

1-تعليقات السطر الواحد ويتم وضع-- قبل النص وذلك يعني ان هذا النص توضيحي وليس تنفيذي اي ان هذه العلامة تمنع المعالج من المرور على النص

2-تعليقات السطور المتعددة وفيها يتم وضع (/*) قبل النص ووضع (*/) بعد النص وهذا يعني ان النص الذي يكون بين هاتين العلامتين هو نص توضيحي وليس تنفيذي

مثل:

```
.....
V_name  varchar2(20) ;-- this variable used to hold the
employee name
Begin
/* this code is used to read
The employee salary and calculate the annual salary
And print the annual salary
/*
.....
End ;
```

استخدام الدوال :

يوجد لدينا نوعين من الدوال :

1- دوال الصف الواحد وهي تتعامل مع صف واحد و الناتج منها قيمة واحدة وهي تنقسم الى :

i. دوال حرفية.
ii. دوال تاريخية .
iii. دوال رقمية .
iv. دوال التحويل.

2- دوال تجميعية : وهي تتعامل مع اكثر من صف و الناتج منها قيمة واحدة

في لغة ال sql نستطيع التعامل مع دوال الصف الواحد والدوال التجميعية

مثال:

```
Select Sum(salary)into V_SumSalary from Employ where
DeptName =V_DeptName;
```

في المثال السابق تم استخدام الدالة Sum وهي من الدوال التجميعية وعن طريقها استطعنا التعرف على مجموع رواتب الموظفين العاملين في قسم معين

مثال:

```
Select  lower(name)into V_ename from Employ where No=1;
```

هنا تم استخدام احدى الدوال الحرفية lower وهي تعيد اسم الموظف رقم واحد بالحروف الصغيرة

اما لغة ال plsql فنستطيع ان نستخدم معها فقط دوال الصف الواحد

مثال :

```
V_name:=initcap(V-name);
Num_months := MONTHS_BETWEEN(SYSDATE,v_date);
```

الوحدات المتداخلة :

يمكن كتابة وتنفيذ وحدة برمجية داخل وحدة برمجية اخرى وتعامل الوحدة الداخلية كجملة تنفيذية . الوحدة الداخلية يمكن ان تكتب في الجزء التنفيذي أو جزء الإستثناءات.

مثال :

```
Declare
x Number;
BEGIN
...
DECLARE
y NUMBER;
BEGIN
...
END;
...
END;
```

في الوحد ة البرمجية السابقة نجد ان مجال المتغير X يكون في الوحدة الداخلية والوحدة الخارجية اما مجال المتغير Y فهو يكون في الوحدة الداخلية فقط .

إذا تم تعريف متغيرين بنفس الاسم احداهما في الوحدة الداخلية والاخرفي الوحدة الخارجية فأن الوحدة البرمجية ستتعامل مع المتغير الاقرب وهو متغير الوحدة الداخلية .

```
SET SERVEROUTPUT ON
Declare
x Number :=3;
Begin
Declare
x Number :=5;
Begin
DBMS_OUTPUT.PUT_LINE (x);
END;
END;
```

إذاً قيمة X في المثال السابق هي 5

ولكن اذا اردنا ان نفرق بين المتغير X التابع للوحدة الخارجية و المتغير X التابع للوحدة الداخلية لابد ان ان نقوم بأعطاء كل وحدة عنوان كالاتي:

```
Declare <outer block>
x Number;
BEGIN
...
DECLARE <inner block>
y NUMBER;
BEGIN
...
END;
...
END;
```

ويتم الوصول للمتغير X التابع للوحدة الخارجية كالاتي :

outer block .x

اما المتغير X التابع للوحدة الداخلية كالاتي:

Inner block .X

جمل التحكم

❖جملة الشرط البسيطة :IF Statement

تحتوي علي الشرط وعلى الجمل الواجب تنفيذها عند تحقق الشرط

```
IF condition THEN
statements;
END IF;
```

اذا لم يتحقق الشرط لايعطي اي نتيجة .

مثال :

```
IF  job = 'manager' THEN
v_sal := v_sal +3500;
END IF;
```

الشرط في المثال السابق الوظيفة تكون manager لكي يتحقق الشرط فاذا تحقق الشرط يتم إضافة 3500 للمرتب واذا لم يتحقق الشرط مثلا اذا كانت الوظيفى Clark لا يتم إضافة 3500 للمرتب

❖جملة الشرط IF THEN ELSE :

تتكون من الشرط ومن الجمل الواجب تنفيذها عند تحقق الشرط والجمل الواجب تنفيذها عند عدم تحقق الشرط

```
IF CONDITION THEN
Statement1;
ELSE
Statement2;
END IF;
```

```
IF  job = 'manager' THEN
v_sal := v_sal +3500;
else
v_sal := v_sal +3000;
END IF;
```

الشرط في المثال السابق الوظيفة تكون manager لكي يتحقق الشرط فاذا تحقق الشرط يتم إضافة 3500 للمرتب واذا لم يتحقق الشرط مثلا اذا كانت الوظيفى Clark يتم إضافة 3000 للمرتب

❖جملة الشرط IF THEN ELSe IF:

تتكون من الشرط والجمل الواجب تنفيذها عند تحقق الشرط وجملة IF جديدة وهي بدورها تحتوي على شرط والجمل الواجب تنفيذها عند تحقق الشرط وهنا يشترط ان يكون لدينا End if(اي نهاية) لكل If

```
IF CONDITION1 THEN
Statement1;
ELSE
IF CONDITION2 THEN
Statement2;
END IF;
END IF;
```

مثال :

```
IF v_deptname = 'mis' THEN
UPDATE emp
SET sal = sal * 200
WHERE deptno = v_deptno;
ELSE
IF v_job = 'manager' THEN
UPDATE emp
SET sal = sal * 100
WHERE job = v_job;
END IF;
END IF;
```

في المثال السابق يتم السؤال عن اسم القسم اذا كان mis يتم تعديل المرتب بضربه في 200 اما اذا لم يتحقق الشرط يتم اختبار الشرط الثاني واذا لم يتحقق ايضا لا ينفذ شيء

❖جملة الشرط IF THEN ELSIF:

تتكون من الشرط والجمل الواجب تنفيذها عند تحقق الشرط وجملة IF جديدة وهي بدورها تحتوي على شرط والجمل الواجب تنفيذها عند تحقق الشرط وهنا لا يشترط ان يكون لدينا End if(اي نهاية) لكل If بل توجد End if واحدة فقط تكتب بعد اخر جملة ويمكن لاخر جملة ان تحتوي على جزء عدم تحقق الشرط

IF CONDITION1 THEN
Statement1;
ELSIF CONDITION2 THEN
Statement2;
ELSIF CONDITION3 THEN
Statement3;
.
.
.
ELSE
StatementN;
END IF;

مثال :

```
IF v_grade >100 OR v_grade < 0 THEN
DBMS_OUTPUT.PUT_LINE('Invalid Grade ');
ELSEF v_grade >= 90 THEN
DBMS_OUTPUT.PUT_LINE('A');
ELSIF v_grade >= 80 THEN
DBMS_OUTPUT.PUT_LINE('B');
ELSIF v_grade >= 70 THEN
DBMS_OUTPUT.PUT_LINE('C');
ELSIF v_grade >= 60 THEN
DBMS_OUTPUT.PUT_LINE('D');
ELSE
DBMS_OUTPUT.PUT_LINE('F');
END IF;
```

في المثال السابق يتم السؤال عن الدرجة اذا كانت اكبرمن 100 واقل من صفر يتم طباعة رسالة تنص بأن القيمة غير مقبولة واذا الدرجة اكبر أو تساوي 90 يتم طباعة الرمز A وهكذا .

واذا لم تتحقق كل الشروط السابقة يتم طباعة الرمز F .

كما ذكرنا سابقا ان لغة ال sql لديها نواقص جعلت شركة oracle تفكر في تطويرها حتى تواكب متطلبات البرمجة الحديثة . ففي لغة ال sql لانستطيع تكرر الجملة اكثر من مرة دون كتابتها ولكن يمكن ذلك في لغة ال plsql وذلك باستخدام حلقات الدوران .

لدينا عدة اشكال لصيغة الدوران:

حلقة الدوران البسيطة Basic Loop

حلقة الدوران for

حلقة الدوران WHILE

حلقة الدوران البسيطة Basic Loop:

تتكون من جملة بداية الدوران وجملة نهاية الدوران والجمل الواقعة بينهما يتم تكرارها عدد لانهائي من المرات (اي ان المعالج لايستطيع الخروج من الحلقة وانهاء البرنامج) ولحل هذه المشكلة يتم عمل شرط خروج يمكن ان يتواجد بعد جملة بداية الدوران مباشرةً كما يمكن ان يتواجد قبل جملة نهاية الدوران

الشكل العام لحلقة الدوران البسيطة :

```
Begin
LOOP
statement1;
. . .
EXIT [WHEN
condition];
End loop;
END;
```

مثال :

وحدة برمجية تقوم بطباعة الارقام من 1-10

```
DECLARE
v_counter NUMBER :=1;
BEGIN
LOOP
DBMS_OUTPUT.PUT_LINE('v_counter = '||v_counter);
EXIT WHEN v_counter >10;
v_counter:=v_counter+1;
END LOOP;
END ;
```

طريقة اخرى لحل الثال السابق:

```
DECLARE
v_counter NUMBER :=1;
BEGIN
LOOP
If   v_counter>10 then
Exit;
End if;
DBMS_OUTPUT.PUT_LINE('v_counter = '||v_counter);
v_counter:=v_counter+1;
END LOOP;
END ;
```

حلقة الدوران for :

تستخدم حلقة الدوران For في الحالات التي يكون فيها عدد المرات التكرار المطلوب تنفيذها معروف قبل بدأ التنفيذ.

```
Begin
For counter  in [REVERSE]
Lower_bound .. upper_pound
LOOP
statement1;
statement2;
. . .
End Loop;
END;
```

Counter عبارة عن متغير يعرف ضمنياً اي لا يتم تعريفه في جزء التعريف لديه قيمة إبتدائية Lower_bound وتزداد قيمته بعد كل دوران الى ان تصل قيمته الحد الاعلي upper_pound

Lower_bound : الحد الادنـي لقيمة بداية الدوران ويجب ان يكون عدد صحيح

upper_pound : الحد الاعلى لقيمة نهاية الدوران ويجب ان يكون عدد صحيح

REVERSE : تستخدم هذه الدالة اذا اردنا ان يبدأ الدوران بطريقة عكسية اي تستخدم لانقاص قيمة المتغير

مثال :

وحدة برمجية تقوم بطباعة الارقام من 1-10

```
BEGIN
FOR i IN 1..10 LOOP
DBMS_OUTPUT.PUT_LINE('i= '||i);
END Loop;
END ;
```

هنا سيتم طباعة الاعداد من 1-10 حيث يبدأ بالعدد 1 وينتهي بالعدد 10

حل المثال السابق بطريقة اخري اذا اردنا ان يبدأ العد بطريقة عكسية .

```
BEGIN
FOR i IN [REVERSE]  1..10 LOOP
DBMS_OUTPUT.PUT_LINE('i= '||i);
END Loop;
END ;
```

هنا سيتم طباعة الاعداد من 1-10 حيث يبدأ بالعدد10 وينتهي بالعدد 1

حلقة الدوران WHILE:

تستخدم حلقة الدوران WHILE في الحالات التي يكون فيها عدد مرات التكرار المطلوب تنفيذها غير معروف وتستمر عملية الدوران مادام الشرط متحقق ودائماً الشرط يكون في بداية الحلقة .

```
Begin
WHILE (condition)
statement1;
. . .
End Loop;
END ;
```

مثال :

وحدة برمجية تقوم بطباعة الارقام من 1-10

```
DECLARE
v_counter NUMBER :=1;
BEGIN
WHILE (v_counter <= 10) LOOP
DBMS_OUTPUT.PUT_LINE('v_counter = '||v_counter);
v_counter:=v_counter+1;
END LOOP;
END ;
```

السجلات

كما ذكرنا سابقاً ان السجل يمكن ان يحتوي على عدة حقول هذه الحقول قد تكون مختلفة الانواع اي ان احد الحقول قد يكون من النوع الحرفي بينما بقية الحقول من النوع الرقمي .

لدينا طريقتين لانشاء السجل :

الطريقة الاولى :

```
Type Type _Name Is Record (Field_Name1  Data_Type  ,
F ield_Name2  Data _Type ,...);
Identifier Type _Name ;
```

الطريقة الثانية :

Record_Name Table_Name%RowTYpe

في الطريقة الاولى الاسترجاع يكون لبعض الاعمده أو كلها أما في الطريقة الثانية يتم استرجاع كل الاعمده الموجوده في الجدول

مثال:

انشاء سجل يتم فيه إسترجاع كل الاعمده الموجوده في جدول الاقسام

```
declare
v_recdepartments % rowtype;
begin
select *
into v_rec
from departmen
where department_id=10;
dbms_output.put_line('The recorsis : ');
dbms_output.put_line('ID : '|| v_rec.department_id);
dbms_output.put_line('Name : '|| v_rec.department_name);
dbms_output.put_line('Manager: '||v_rec.manager_id);
dbms_output.put_line('Location : '|| v_rec.location_id);
end ;
```

المؤشرات

هنالك نوعين من المؤشرات :

1- صريحة

2- ضمنية

المؤشرات الصريحة:

يتم تعريف هذا النوع من المؤشرات في جزء التعريف declare كالاتي :

```
Declare
Cursor  cursor_name is select statement;
```

يتم بعد ذلك فتح المؤشر بعد عبارة begin ولكي يتم فتح المؤشر يتم استخدام الامر open كالاتي:

Open cursor_name;

في كل مره يتم فيها فتح المؤشر يجب ان يتم استرجاع سجل(صف) واحد فقط

بعد ان تم فتح المؤشر يتم إسترجاع البيانات من المؤشر باستخدام الامر fetch كالاتي :

Fetch cursor_name into varible1,varible2,....;

يجب ان يساوي عدد المتغيرات عدد الحقول الموجوده في استعلام المؤشر

بعد الانتهاء من إجراء العمليات على المؤشر يتم اغلاقه بإستخدام الامر close كالاتي:

Close cursor_name;

مثال :

قم بإنشاء cursorيقوم بإسترجاع وطباعة اسم الطالب اذا كان رقم الطالب يساوي 1

```
set server out put on
Declare
Student_name char (20);
Cursor  student_cursor  is select name from student
Where no=1;
Begin
Open student_cursor;
Fetch student_cursor into student_name;
Dbms _output.putline(student_name);
Close student_cursor;
End;
```

خصائص المؤشرات :

SQL%ROWCOUNT:

تعيد عدد الصفوف التي تأثرت بآخر جملة sql

SQL%FOUND :

تعيد true إذا تأثر صف او اكثر بآخر جملة sql

SQL%NOTFOUND :

تعيد true إذا لم تؤثر آخر جملة sql باي صف

SQL%ISOPEN :

تعيد true إذا كان المؤشر مفتوح

مثال :

قم بإنشاء cursorيقوم بإسترجاع وطباعة اسم ورقم ومرتب الموظف اذا كان رقم القسم يساوي 10

```
declare
v_id  employees.employee_id%type;
v_fname  employees.first_name%type;
v_sal employees.salary%type;
Cursor  c  is select employee_id,first_name,salary
From  employees  where  department_id=10;
begin
open c;
loop
fetch c into v_id,v_fname, v_sal;
exit when c % notfound;
dbms_output.put_line(v_id||' '||v_fname||' '||v_sal);
end loop;
close c;
end ;
```

يمكن استخدام المؤشرات مع السجلات فبدلا من إرجاع قيمة المؤشر في المتغيرات يتم ارجاعها في سجل

مثال:

قم بإنشاء cursorيقوم بإسترجاع وطباعة اسم ورقم ومرتب الموظف اذا كان رقم القسم يساوي 10

```
declare
Cursor  c is  select  employee_id,first_name,salary
From employees where  department_id=50;
v_rec  c %rowtype;
begin
open c;
fetch c into v_rec;
while c %found loop
dbms_output.put_line(v_rec.employee_id||' '||v_rec.first_name||'
'||v_rec.salary);
Fetch c into  v_rec;
end loop;
close c;
end ;
```

يمكن حل المثال السابق بإستخدام حلقة for

```
declare
Cursor  c is select employee_id , first_name ,salary
from employees where department_id=50;
begin
For  v_recin  in c loop
dbms_output.put_line(v_rec.employee_id||'
'||v_rec.first_name||' '||v_rec.salary);
end loop;
end;
```

عند استخدام حلقة for مع المؤشرات لا نحتاج الى :

fetch ,open, close

مثال :

قم بإنشاء cursorيقوم بتعديل وطباعة المرتب بإضافة 200 للمرتب اذا كان المرتب اكبر من 3000وطباعة الرسالة التالية إذا كان المرتب اقل من 3000 (No Changes)

```
declare
v_sal  employees.salary%type;
v_id   employees.employee_id%type;
Cursor  c   is   select    employee_id  ,  salary   from   employees;
begin
open   c;
loop
fetch c into    v_id,  v_sal;
exit when c%notfound;
If   v_sal   > 3000 then
Update employees  set   salary   =salary+200   where   employee_id= v _id;
dbms_output.put_line(v_sal );
else
dbms_output.put_line('No Changes');
end if;
end loop;
close c;
end ;
```

الاستثناءات

الاستثناء :

عباره عن خطأ يحدث اثناء تنفيذ الوحده البرمجية ويؤدي الى وقف التنفيذ لذلك لابد ان تتم معالجته .

لمعالجه الاستثناء لابد من الامساك به اولا وتتم عملية الامساك بالإستثناء من خلال الكلمة when متبوعة بالإستثناء المراد معالجته وذلك في الجزؤ الخاص بمعالجة الإستثناءات .وبعد ذلك يتم تحديد الاجراء الواجب عمله او تنفيذه عند حدوث هذا الخطأ

انواع الاستثناءات :

1- الاخطاء المعرفة مسبقا :

هذا النوع من الاستثناءات كثيرا ماتتكرر في البرامج ويجب عدم اظهارها لانها تظهر ضمنيا من قبل خادم oracle

ومن أمثلة هذا النوع من الاخطاء:

CURSOR_ALREADY_OPEN تعني ان المؤشر مفتوح

DUP_VAL_ON_INDEX محاولة وضع قيم مماثلة

INVALID_CURSOR عملية مؤشر غير صحيحة

LOGIN_DENIED خطأ في اسم المستخدم او كلمة المرور

NO_DATA_FOUND جملة إسترجاع لاتعيد اي نتيجة

TOO_MANY_ROWS جملة إسترجاع تعيداكثر من صف

مثال:

```
declare
v_lnameemployees.last_name%type;
begin
select  last_name  into v_lname
from employees
Where  employee_id=101;
dbms_output.put_line(v_lname);
exception
when   no_data_foundthen
dbms_output.put_line('No data returned from sqlstatement');
end;
```

2- الاخطاء غير المعرفة مسبقا :

هي اي خطأ من اخطاء oracle غير تلك المعرفة مسبقا ويتم تعريفها في جزء التعريف ويجب عدم اظهارها لانها تظهر ضمنيا من قبل خادم oracle

يتم تعريف الإستثناء كالاتي :

```
declare
  EXCEPTION   Exception_name ;
```

يتم ربط الإستثناء مع الخطأ بأستخدام

PRAGMA_EXCEPTION_INIT

بعد ذلك يتم معالجة الإستثناء

مثال :

```
DECLARE
EXCEPTION  e_emps_remaining ;
PRAGMA  EXCEPTION_INIT ( e_emps_remaining , -2292);
v_deptno  dept.deptno%TYPE;
BEGIN
DELETE  FROM  dept  WHERE  deptno = v_deptno;
COMMIT;
EXCEPTION
WHEN e_emps_remaining THEN
DBMS_OUTPUT.PUT_LINE ('Cannot removedept ' ||
TO_CHAR(v_deptno) || '. Employees exist. ');
END;
```

3- استثناءات المستخدم :

هو اي حدث يعتبره المستخدم على انه خطأ ويجب وقف تنفيذ الوحده عند حدوث هذا الخطأ وهذا النوع من الإستثناءات تعرف وتظهر صراحة من قبل المستخدم .

يتم تعريف الإستثناء كالاتي :

```
Declare
EXCEPTION  Exception_name ;
```

ثم يتم الإستثناء في الجزء الخاص بالتنفيذ باستخدام الامر Raise متبوعا بإسم الاستثناء

```
RAISE Exception_name ;
```

بعد ذلك يتم معالجة الإستثناء

مثال :

```
declare
v_sal   employees.salary%type;
Exception   my_exp;
begin
Update   employees  set  salary=salary*1.7   where  employee_id=200;
Select    salary    into   v_sal   from employees    where employee_id=200;
If  v_sal>5500  then
Raise   my_exp;
else
dbms_output.put_line('The salary is updated');
end if;
exception
when my_expthen
dbms_output.put_line('The salary is more than 7500');
end;
```