بسم الله الرحمن الرحيم

دروس PL/SQL

اعداد اخوكم :
محمد ابراهيم العضيب

التعامل مع الاجراءت المخزنة من خلال البارامترات:

قبل البدء بشرح طرق الاتصال بالاجراءت سوف ناخذ شرح مبسط عن الاجراءت :

المعاملات الحسابية المستخدمة

فيمايلي اغلب المعاملات الحسابية المستخدمة بكثرة في لغة pl/sql واذا كنت معتاد على لغة برمجة فلن تكون هذا الشي جديد عليك

+	الجمع
*	الضرب
**	الاس
-	الطرح
/	القسمة

المعاملات العلاقية :

< >	لايساوي
= ^	لايساوي
>	اكبر من
=!	لايساوي
<	اقل من

انواع البيانات شائعة الاستخدام :

١- varchar2

هذا النوع متغير الطول ويشتمل على الاحرف الابجدية والارقام

```
X    varchar2(20)
```

حيث الموجود داخل القوسين هو length الطول

ومن الممكن اعطاءه قيمة ابتدائية كمايلي

```
X   varchar2(20)='hamad'
```

٢- number

يستخدم لتمثيل البيانات الرقمية وتكون صيغة الاعلان كمايلي:

```
Num  number(s)
```

S هي عدد الارقام(الخانات) وتاخذ قيمة بين 1 إلى 38

ويمكن ايضا تعريف اي متغير رقمي من النوع العشري كمايلي:

```
Num  number(s,p)
```

حيث s عدد خانات الرقم الصحيح وايضا العشري

اما p فهي عدد المنازل(الخانات) بعد الفاصلة مثال

```
Num number(12,2)
```

معنى هذا ان الرقم num مكون من ١٠ ارقام صحيحة وؤقمين بعد الفاصلة وبذلك بكون المجموع 12

٣- date :
يستخدم هذا المتغير لتخزين قيم التواريخ مثل

```
Date_brith  date;
```

في الوضع الافتراضي يعرض اوراكل قيمة التاريخ بالشكل
DD-MON-YY

٤- Boolean :
منطقي true او false

نبدأ الان بمكونات pl/sql :
برامج pl/sql تتم كتابتها في كتل من اوامر البرمجة تحتوي على مقاطع منفصلة للاعلان عن المتغيرات واوامر البرامج ومعالجة الاستثناءات(الاخطاء) .
ومن الممكن تخزين الاجراء في قاعدة البيانات كبرنامج فرعي له اسم محدد او كتابتها مباشرة في sql plus * ككتله مجهولة.

وكتلة البرنامج كمايلي:
سنبدا اولا كتابة الاجراء مباشرة في sql * plus وهي كمايلي:

```
DECLARE
```

هنا توجد تعريفات المتغيرات والموشرات

```
BEGIN
```

جسم البرنامج

```
EXCEPTION
```

رموز معالجة الاخطاء

```
END:
```

مع ملاحظة مايلي:
ان قسم declare وقسم exception هما اختيارين اي لايشترط وجودهما
اي اذا كان لايوجد لديك تعريف متغيرات لاتستخدم declare واذا كنت لان تتعامل مع الاخطاء لاتستخدم exception

---

طرق الاسناد
مثل اذا اردت ان تقول ان قيمة i=5 فيتم ذلك كمايلي:

```
i:=5;
```

يجب وضع النقطتين قبل =

سوف ناخذ مثال على ذلك
يوجد ضمن اوامر sql الامر DBMS_OUTPUT.PUT_LINE
يستخدم لكي تعرض النتيجة في sql * plus والصيغة العامة له :
DBMS_OUTPUT.PUT_LINE(massege)

حيث massage هي النص او الشي الذي تريد عرضة
تم شرح هذا الامر لكي نبدأ به ونستخدمه لفهم اوامر ال pl/sql لكن في المستقبل سوف تعرف ان هذا الامر لايهمك كثيرا.مثال :
نريد طباعة "ARABTEAM2000" على الشاشة sql * plus يتم ذلك كمايلي:

```
SET SERVEROUTPUT ON;
BEGIN
DBMS_OUTPUT.PUT_LINE('ARABTEAM2000');
End;
```

جرب هذا ولاحظ النتائج ولعلك تتسال عن سبب وجود السطر الاول
السطر الاول يخبر sql * plus بأن يكتب كل مايعود به المخدم.
ويكفي كتابته مرة واحدة عندما تدخل sql * plus
مثال اخر باستخدام المتغيرات

```
Declare
i   number(5);
BEGIN
i:=5;
DBMS_OUTPUT.PUT_LINE('i = ' || i);
END;
```

جرب هذا الكود ولاحظ النتائج
فائدة || الموجدة ضمن عملة الطباعة هي للوصل بين التعبيرين
اوامر اللغة :
أ- الشرط عبارة if then
تستخدم هذه العبارة مثل اي العبارت الشرطية في لغة سي او سي++ او فيجوال بيسك وغيرها ، ولها استخدمات عديدة وسوف نعرف كيف نستخدمها مقدما مع حقول قواعد البيانات وذلك بعد اخذ المؤشرات
الصيغة العامة لها كمايلي:

```
IF   conditonal      THEN
جواب الشرط
ELSE
جواب الشرط اذا كان خطأ
END IF
```

حيث conditional هي الشرط مثال على ذلك :

```
Declare
i   number(5);
BEGIN
i:=5;
IF i=5 then
DBMS_OUTPUT.PUT_LINE('i = ' || i);
ELSE
DBMS_OUTPUT.PUT_LINE('i not eqal 5 ');
```

```
END IF;
END;
```

الشرط باستخدام اكثر من شرط

```
Declare
i   number(5);
BEGIN
i:=5;
IF i>1 then
DBMS_OUTPUT.PUT_LINE(i  || ' > 1');
ELSIF i<1 then
DBMS_OUTPUT.PUT_LINE(i  || ' < 1');
ELSIF i=1 then
DBMS_OUTPUT.PUT_LINE(i  || ' = 1');
END IF;
END;
```

لكن لاحظ هنا فقط كتابة elsif ولاتخطي فيها فهي ليست elseif

ب- الــتكرار :
ويوجد عدة اوامر للتكرار وهي:

loop-exit-end *

وهنا لابد من وضع شرط لأنهاء الحلقة
نأخذ مثال بسيط على ذلك

```
Declare
i   number(5);
BEGIN
i:=1;
LOOP
IF i>10 then
EXIT;
END IF;
```

```
DBMS_OUTPUT.PUT_LINE('i =' || i);
i:=i+1;
End loop;
END;
/
```

شرح الكود السابق
السطر الثاني : تعريف متغير من نوع رقم
السطر الثالث :البداية
السطر الرابع : اعطاء المتغير قيمة ابتدائية وهي i=1
السطر الخامس : شرط الانهاء
السطر السادس : الانهاء اذا كان i>10 ولا يكمل
السطر السابع :انها if
السطر الثامن : طباعة i
السطر التاسع :زيادة قيمة i بواحد
السطر العاشر : نهاية الحلقة
وبعد كتابة هذا الكود يكون الناتج كمايلي:

i =1
i =2
i =3
i =4
i =5
i =6
i =7
i =8
i =9
i =10

LOOP- **----------

WHEN - END EXIT

```
Declare
i   number(5);
BEGIN
i:=1;
LOOP
EXIT WHEN i>10;
DBMS_OUTPUT.PUT_LINE('i =' || i);
i:=i+1;
End loop;
END;
/
```

ويكون الناتج نفس السابق لكن لاحظ استخدم شرط الانهاء

```
EXIT WHEN i>10;
```

*** WHILE - LOOP - END

```
Declare
i  number(5);
BEGIN
i:=1;
WHILE i <= 10 LOOP
DBMS_OUTPUT.PUT_LINE('i =' || i);
i:=i+1;
End loop;
END;
/
```

ويكون الناتج نفس المثال السابق

**** FOR - IN - LOOP - END :

وهذه ايضا طريقة اخرى لاستخدام حلقات التكرار وهي نفس عمل اسلوب حلقات for في اي لغة برمجة

والصيغة العامه لها هي على الاتي

FOR i IN البداية.. النهاية LOOP

الجمل المراد تكرارها

LOOP END

مثـــــــــــــال:

```
Declare
i  number(5);
BEGIN
FOR  i  IN  1..10 LOOP
DBMS_OUTPUT.PUT_LINE('i =' || i);
End loop;
END;
/
```

وسوف يكون الناتج كمايلي(نفس السابق):

١= i

i =2

i =3

i =4

i =5

i =6

i =7

i =8
i =9
١٠= i

ج/ المؤشرات CURSORS
هذا الدرس من اهم المواضيع في pl/sql
تستخدم pl/sql المؤشرات cursors لأدارة عبارات التحديد select في لغة sql وكما لاحظنا الاوامر السابقة مثل if والتكرار لم نستخدمها مع بيانات الجداول المخزنه ولعمل ذلك لابد من استخدام هذه المؤشرات.
وهناك نوعين من المؤشرات هي الضمنية والصريحة وسوف نتطرق لك واحد بالتفصيل والامثلة اللازمة.
أ- المؤشرات الصريحه :
يتم تعريف هذا النوع من المؤشرات كجزء من الاعلان declare ويجب ان تشتمل عبارة sql المعرفه على عبارة التحديد select فقط حيث لايمكن استخدام الكلمات الاساسية
insert,update,delete
وعند استخدام المؤشرات الصريحه دائما ماستكتب اربعة مكونات كمايلي:
١- يتم تعريف المؤشر في الجزء declare
٢- يتم فتح المؤشر بعد عبارة begin

الصيغة العامة لتعريف المؤشر الصريح كمايلي:

DECLARE
CURSOR اسم المؤشر IS
الاستعلام

تقوم باستبدال اسم المؤشر باسم مؤشر حقيقي
وتقوم بوضع جملة الاستعلام select في مكان الاستعلام

ولكي تقوم بفتح هذا المؤشر وتستخدمه نقوم بفتحه باستخدام الامر open كمايلي:
OPEN اسم المؤشر
وبعد فتح المؤشر تقوم باسترجاع او تحميل البيانات سطر(سجل) واحد من المؤشر الذي تم تعريفه باستخدام الامر FETCH كمايلي:
FETCH اسم المؤشر INTO متغير١،متغير٢،.....
ومعنى هذا اي قم باسترجاع البيانات من المؤشر المعطى اسمه وحملها into الى المتغيرات كع ملاحظة ان عدد المتغيرات يساوي عدد الحقول الموجودة في استعلام المؤشر.
وبعد الانتهاء من اجراء العمليات على المؤشر يجب عليك اغلاقه ويتم اغلاقه كمايلي:
close cursor_name

مثال على طريقة تعريف مؤشر :
افرض انه لدينا هذا الجدول

age	name	no
23	mohammed	111
22	talal	222
24	majed	333

اولا قم بانشاء هذا الجدول كمايلي:

```
create table stud(
no number(4),
name varchar2(40),
age number(2));
```

ثانيا قم بادخال البيانات السابقة في هذا الجدول كمايلي:

```
insert into stud values(111,'mohammed',23);
insert into stud values(222,'talal',22);
insert into stud values(333,'majed',24);
```

ثم قم بتنفيذ مايلي

```
set serveroutput on;
DECLARE
name_stu varchar2(40);
CURSOR  name_student IS
select name from stud
where no=111;
BEGIN
OPEN name_student;
FETCH name_student INTO name_stu;
DBMS_OUTPUT.PUT_LINE(name_stu);
CLOSE name_student;
END;
/
```

بعد التنفيذ سوف يظهر لك الناتج كمايلي:

mohammed

وهذا الشي صحيح

لاحظ اننا اتبعنا نفس الخطوات التي ذكرناه لكي نتعامل مع مؤشر صريح

لاحظنا في المثال السابق ان الاستعلام في cursor سوف يعود بسجل واحد لكن ماذا يحدث لو اعاد المؤشر اكثر من سجل واردنا المرور على كافة السجلات ؟

لحل السؤال السابق لابد من استخدام حلقة بها شرط وهذا هو هل سجلات المؤشر انتهت ام لا ونعرف ذلك من خلال خاصية found للمؤشر كمايلي:

found%mycur

حيث :

mycur : هي اسم المؤشر.

% : توضح انا مايلي اسم المؤشر هي احد خصائصه.

found : خاصية التي من خلالها نعرف هل تم الانتهاء من جميع السجلات ام لا

مثال :

اسم الطالب	كود المقرر	الدرجة	النتيجة
NO_STU	SUBJECT	MARK	RESULT
111	216CS	88	
222	225CS	75	
333	225CS	40	

نريد انشاء اجراء يقوم بالمرور على الجدول وينظر الى درجة الطالب اذا كان ناجح في المقرر ام لا فاذا كان mark اكبر او يساوي ٥٠ ضع قيمة true في حقل result والا ضع قيمة false في حقل result

نقوم اولا بانشاء هذا الجدول :

```
create table stu_study(
NO_STU number(4),
SUBJECT varchar2(8),
MARK number(3),
RESULT varchar2(20));
```

المدخلات السابقة وبعد انشاء الجدول نقوم بادخال

```
insert into stu_study (NO_STU,SUBJECT,MARK) values
(111,'216CS',88);

insert into stu_study (NO_STU,SUBJECT,MARK) values
(222,'225CS',75);
```

```
insert into stu_study (NO_STU,SUBJECT,MARK) values
(333,'225CS',40);
```

بعد ذلك نقوم بانشاء الاجراء:

```
declare
mar number(3);
no number(3);
cursor res_stu is
select no_stu,mark
from stu_study;
begin
open res_stu;
loop
fetch res_stu into no,mar;
exit when res_stu%notfound;
if mar>=50 then
update stu_study set result='TRUE' where no_stu=no;
else
update stu_study set result='FALSE' where no_stu=no;
end if;
end loop;
close res_stu;
end;
/
```

وبهذا تكون النتائج في الجدول كمايلي :

NO_STU	SUBJECT	MARK	RESULT

111	216CS	88	TRUE
222	225CS	75	TRUE
333	225CS	40	FALSE

هل رأيتم سهوله ذلك والفائد الكبيرة من المؤشرات.

هناك طريقة اخرى لتعريف المتغيرات لاحظ في الجدول السابق ان الحقل no_Stu تم تعريفه على انه من نوع number وتم تعريف المتغير no في الاجراء على انه number ايضا لكي يتم وضع رقم الطالب فيه لكن لاحظ لو تم تغير نوع الحقل في الجدول من number الى varchar2 فانه يجب عليك تغير نوع المتغير no في الاجراء ايضا لكن هناك طريقه تجعلك لاتعدل الاجراء كل مرة وهي استخدام الامر التالي لتعريف المتغير no في الاجراء

type%no_stu.stu_study NO

حيث :

NO هي اسم المتغير

stu_study : اسم الجدول

no_stu : الحقل المطلوب في الجدول

type% : خاصية نوع الحقل

ومعنى ماسبق قم بتعريف متغير اسمه no له نفس نوعية الحقل الذي اسمه NO_STU الموجود في الجدول stu_study .

وبهذا لان تقوم بتغير نوع العنصر في الاجراء في كل مرة تغير النوع وهكذا مع جميع المتغيرات التي لها صله بالجداول

وبذلك يصبح الاجراء بعد التعديل كمايلي:

```
declare
mar stu_study.mark%type;
no stu_study.no_stu%type;
cursor res_stu is
select no_stu,mark
from stu_study;
begin
open res_stu;
loop
```

```
fetch res_stu into no,mar;
exit when res_stu%notfound;
if mar>=50 then
update stu_study set result='TRUE' where no_stu=no;
else
update stu_study set result='FALSE' where no_stu=no;
end if;
end loop;
close res_stu;
end;
/
```

ب -المؤشرات الضمنية :

وهي اسهل من المؤشرات الصريحه

وتوجد نقطتين هامتين عند التعامل مع المؤشرات الضمنيه :

* يظهر المؤشر الضمني في جسم الاجراء body وليس في declare الخاص بالاجراء كمافي المؤشرات الصريحه

* لابد ان يسترجع مؤشر select الضمني سطر واحد.

والصيغة العامه للمؤشر الضمني كمايلي:

```
SELECT COLUM1,COLUM2,.....  INTO
VARIABLE1,VARIABLE2,.....          FROM  table_name
```

ومعنى هذا قم باختيار الحقل١ و الحقل٢ وضعها في المتغيرات منغير١ و متغير٢ من الجدول table_name

سوف نأخذ مثال على ذلك وسوف نستخدم الجدول الذي انشئناه سابق في الدرس الثاني عندما تعاملنا مع المؤشرات الصريحة وكان اسم الجدول stud

no	name	age
111	mohammed	23
222	talal	22
333	majed	24

واردنا مثلا كتابة اجراء يقوم بحساب متوسط اعمار الطلاب (قد يقول البعض انه لايحتاج ذلك الى اجراء فمجرد استخدام جملة select نستطيع عمل ذلك انا اقول نعم هذا صحيح لكن احب استخدام

الاجراء في هذا المثال لكي نرى طريقة عمل المؤشر الضمني ولكن سوف نرى بعد قليل مثال شامل يتم فيه استخدام الموشرات الصريحه والضمنية في نفس الوقت) والان نقوم بكتابة الاجراء كمايلي :

```
set serveroutput on;
declare
aveage number(4,2);
begin
select avg(age)
into aveage
from stud;
DBMS_OUTPUT.PUT_LINE(aveage);
end;
/
```

*** مثال شامل لاستخدام الموشرات الصريحة والضمنيه في نفس الوقت :

لنفرض انه لدينا الجدولين التاليين : الجدول الاول اسمه courses (المقررات):

رقم المقرر	اسم المقرر	عدد الساعات
code	course_name	hours
216CS	NETWORK	3
225CS	ASSEMBLY	3
325CS	DATABASE	4

وقم بانشاء الجدول كمايلي:

```
create table courses(
code varchar2(8),
course_name varchar2(40),
hours number(3),
primary key(code));
```

وقم بادخال البيانات الموجود بالجدول كمايلي:

```
insert into courses values('216CS','NETWORK',3);

insert into courses values('225CS','ASSEMBLY',3);

insert into courses values('325CS','DATABASE',4);
```

ثم نقوم بتكوين الجدول الثاني وهو studys :

اسم الطالب	كود المقرر	الدرجة	عدد النقاط
NO_STU	COURSE_CODE	MARK	POINT
111	216CS	88	
222	225CS	75	
333	225CS	40	
111	225CS	90	
222	216CS	78	
333	216CS	85	

ويتم الانشاء كمايلي:

```
create table studys(
NO_STU varchar2(6),
COURSE_CODE varchar2(8),
MARK number(3),
point number(5,2),
primary key(NO_STU,COURSE_CODE));
```

ويتم ادخال بالبيانات الموجودة بالجدول كمايلي:

```
insert into studys(NO_STU,COURSE_CODE,MARK)
values ('111','216CS',88);
insert into studys(NO_STU,COURSE_CODE,MARK)
values ('222','225CS',75);
insert into studys(NO_STU,COURSE_CODE,MARK)
values ('333','225CS',40);
insert into studys(NO_STU,COURSE_CODE,MARK)
values ('111','225CS',90);
insert into studys(NO_STU,COURSE_CODE,MARK)
values ('222','216CS',75);
insert into studys(NO_STU,COURSE_CODE,MARK)
values ('333','216CS',85);
```

بعد الانتهاء من انشاء وادخال البيانات المطلوب انشاء اجراء يقوم بحساب عدد النقاط لكل طالب وفي كل مادة وهو الحقل عدد النقاط الذي لم ندخل فيه اي شيء ويجب نعلم ان :

MARK	average
95-100	5
90-94	4.75

85-89	4.5
80-84	4
75-79	3.5
70-74	3
65-69	2.5
60-64	2
1-59	1

ويتم حساب النقاط كمايلي :

عدد النقاط في اي مقرر = معدل المادة (وليس الدرجة كمافي الجدول السابق) * عدد ساعات المقرر

مثال لحساب معدل الطالب الذي رقمه ١١١ في المقرر 216CS

نلاحظ من جدول studys ان الطالب قد تحصل على درجة ٨٨ ونلاحظ ان الدرجة من الجدول السابق هي بين ٨٥- ٨٩ وبالتالي فإن معدل الطالب في هذا المقرر هو ٤,٥ (وهي الطريقة المتبعة في اغلب الجامعات) ، ومن جدول courses نحصل على عدد الساعات للمقرر وبالتالي فان :

عدد النقاط = ٤,٥ * ٣ = ١٣,٥ وهكذا في جميع الطلاب وهذا هو المطلوب من الاجراء عمله.

وبالتالي فان الاجراء سوف يكون كمايلي:

```
DECLARE
no_Student studys.NO_STU%type;
hou courses.hours%type;
mark studys.mark%type;
cou_code courses.code%type;
poi studys.point%type;
cursor st_point is
select NO_STU,COURSE_CODE,MARK from studys;
BEGIN
open st_point;
loop
exit when st_point%notfound;
fetch st_point into no_Student,cou_code,mark;
select hours
into hou
from courses
where code=cou_code ;
if (mark>=95)and(mark<=100) then
```

```
poi:=5 * hou;
elsif mark>=90 then
poi:=4.75 * hou;
elsif mark>=85 then
poi:=4.5 * hou;
elsif mark>=80 then
poi:=4 * hou;
elsif mark>=75 then
poi:=3.5 * hou;
elsif mark>=70 then
poi:=3 * hou;
elsif mark>=65 then
poi:=2.5 * hou;
elsif mark>=60 then
poi:=2 * hou;
else
poi:=1 * hou;
end if;
update studys set POINT=poi
where NO_STU=no_Student and COURSE_CODE=cou_code ;
end loop;
close st_point;
end;

/
```

لاحظ هنا اننا استخدمنا المؤشرات الصريحة والمؤشرات الضمنيه والصريحة استخدمناه لكي تقوم بفتح سجلات الجدول studys والمؤشر الضمنلي استخدمنا لكي يعود بعدد الساعات في كل مرة يدور بالحلقة.

شرح الاجراء :

في التعريفات اتوقع انه لاتوجد هناك مشكلة لديكم ، اما جسم البرنامج ابتدأ من begin فهو كمايلي :

اولا يفتح المؤشر الصريح والذي يحتوي على جميع سجلات الجدول studys ثم يكون حلقة دورانية لكي يمر على جميع سجلات الطلاب الموجودة في المؤشر الصريح وطبعا شرط الانهاء لهذه الحلقة هو الوصول الى اخر سجل . ثم يقوم بعملية تحديث سجلات الطالب الاولى في المتغيرات كمايلي:

```
fetch st_point into no_Student,cou_code,mark;
```

وطبعا المتغيرات هي رقم الطالب ورقم المقرر والدرجة في المقرر ولنفرض الان نحن الان عند السجل الاول ووهو الطالب الذي رقمه ١١١ ورقم المقرر CS٢١٦ ودرجته هي ٨٨ سوف يضع هذه البيانات في المتغيرات، ثم يستخدم مؤشر ضمني لكي يحضر عدد ساعات المادة التي درسها الطالب ١١١ وهي CS٢١٦ و المؤشر هو

```
select hours
into hou
from courses
where code=cou_code ;
```

ومعنى هذا احضر عدد ساعات المقرر الذي رقمه هو cou_code وهذا المتغير هو معروف من المؤشر الصريح الاول وسبب استخدامنا هذا المؤشر هو ان عدد ساعات المقرر موجودة في جدول اخر ولابد من استخدام هذا المؤشر لكي يحضر عدد الساعات. وبما اننا فرضنا اننا عند السجل الاول فسوف يحضر عدد ساعات المقرر CS٢١٦ وهي ٣ ساعات ثم بدأ يختبر الدرجة وذلك طبقا للجدول الدرجات والمعدلات حيث كانت درجة الطالب الذي رقمه ١١١ في المقرر cCS٢١٦ هي ٨٨ وبالتالي يكون عدد النقاط كمايلي = ٤,٧٥ * ٣ = ١٣,٥ ، وبعد الانتهاء من حساب المعدل يقوم بتعديل الجدول وتحديث قيمة point بقيمتها الجديدة ، وهكذا يمر على كل طالب بنفس الطريقة السابقة الى ان يصل الى نهاية السجلات.وبالتالي تكون النتائج كمايلي في الجدول studys :

NO_STU	COURSE_CODE	MARK	POINT
111	216CS	88	13.5
222	225CS	75	10.5
333	225CS	40	3
111	225CS	90	14.25
222	216CS	78	10.5
333	216CS	85	13.5

بعد الانتهاء من هذا المثال نكون انهينا المؤشرات بنوعيها بشكل تام وبامثله واقعيه وتوصل المعلومة بشكل سليم.

الـــــــــــــجـــــــداول في pl/sql (المصفوفات):

تستخدم هذه الجداول (المصفوفات) مثل المصفوفات في هي لغة من لغات البرمجة مثل لو كانت لديك سلسلة من الارقام وتريد تخزينها فانك تستخدم هذه الجداول للتخزين ويتم تعريف متغير من هذا النوع كمايلي اولا يتم تعريف هذا النوع :

```
TYPE  اسم_النوع IS TABLE OF نوع_المتغير INDEX BY BINARY_INTEGER
```

مثال على ذلك :

```
DECLARE
TYPE num_array IS TABLE OF number(4) INDEX BY
BINARY_INTEGER;
num num_array;
BEGIN
.....
.....
END;
```

لاحظ اولا تم تعريف نوع واسماه num_array ، ثم قام بتعريف متغير num واعطاه نوع num_array وهو النوع الجديد الذي قمنا بانشاءه.مثال عملي /

```
set serveroutput on;
DECLARE
TYPE num_array IS TABLE OF number(4) INDEX BY
BINARY_INTEGER;
i   number(4);
num    num_array;
BEGIN
FOR  i  IN  1..10 LOOP
num(i) := i * i ;
END LOOP;
FOR  i  IN  1..10 LOOP
DBMS_OUTPUT.PUT_LINE(i || '*' || i || '= ' || num(i) );
END LOOP;
END;
/
```

وويكون عمل هذا الاجراء كمايلي : الحلقة الاولى تقوم بضرب العدد i في نفسه وتخزنه في المتغير num برتبه i وهكذا والحلقة الثانية للطباعة ويكون الناتج كمايلي :

```
1*1= 1
2*2= 4
3*3= 9
4*4= 16
5*5= 25
6*6= 36
7*7= 49
8*8= 64
9*9= 81
10*10= 100
```

ولكي نقوم بانشاء اجراء مخزن نقوم بمايلي :

CREATE [OR REPLACE] PROCEDURE procedure_name(الادخال متغيرات
(الممررة ومتغيرات الاخراج

حيث تمثل procedure_name اسم الاجراء المستخدم.

اما OR REPLACE فهي توضع حينما تعلم ان الاجراء موجود من السابق.

اما عن المغيرات التي بين القوسين فهي اما متغيرات مدخله مثل اذا كان لديك اجراء حساب معدل طالب وتريد تمرير رقم الطالب الذي تريد حساب معدله فهذه هي تعتبر كمدخلات ولتعريف متغير بهذا الشكل يكون كمايلي :

student_id in number(9)

لاحظ اسم المتغير هو student_id ثم بعده وضعنا الكلمة in ومعنى ان هذا المتغير يعتبر كمدخل

اما لتعريف متغير يعود بقيمة من الاجراء مثلا لو اردنا تعرف متغبر يرجع بمعدل الطالب يتم التعريف كمايلي :

ave out number(5,2)

بعد تنفيذ الاجراء يكون هذا المتغير يحتوي على معدل الطالب الذي تم تمرير رقمه مثلا.

مع العلم انه يمكن تعريف متغير للمدخلات والمخرجات حيث تمرر به القيمه اولا وبعد تنفيذ الاجراء يتم وضع القيمه في نفس المتغير وتتم كمايلي :

ave in out number(5,2)

ومعنى هذا اي مدخل ومخرج في نفس الوقت .

مـــــــــــــثال :

في الجدول الذي قمنا بدراسته في الدرس الرابع وكان بأسم studys وكان كمايلي :

NO_STU	COURSE_CODE	MARK	POINT
111	216CS	88	13.5
222	225CS	75	10.5
333	225CS	40	3
111	225CS	90	14.25
222	216CS	78	10.5
333	216CS	85	13.5

لو اردنا تصميم اجراء مخزن لكي يقوم بطباعة درجة الطالب بعد تمرير رقم الطالب ورقم المقرر.

الاجراء المخزن سوف يكون كمايلي :

```
create or replace procedure stu_mark(
stu_id in studys.NO_STU%type,
cou in studys.COURSE_CODE%type)
as
mar studys.mark%type;
begin
select mark
into mar
from studys
where NO_STU=stu_id
and COURSE_CODE=cou;
DBMS_OUTPUT.PUT_LINE(mar);
end;
/
```

بعد الانتهاء من تنفيذ الاجراء يكون الاجراء مخزن في قاعدة البيانات ولكي نقوم باستدعاءه نقوم بمايلي :

```
begin
stu_mark(111,'216CS');
end;
/
```

لاحظ كيف تم استدعاء الاجراء السابق من خلال اسم الاجراء وبذلك سوف يكون الناتج على الشاشة كمايلي 88 وهي صحيحة بعد تمرير رقم الطالب 111 ومقرر 216CS

لكن لاحظ اننا لم نستخدم متغيرات اخراج لكن مأرايك ان نصمم اجراء اخر يقوم بنفس الوظيفة التي يقوم بها الاجراء السابق لكن عملية الطباعة تكون بعد الاستدعاء لكي نجعل الاجراء يقوم بارجاع درجة الطالب بمتغير لذلك فان الاجراء كمايلي :

```
create or replace procedure stu_mark22(
stu_id in studys.NO_STU%type,
cou in studys.COURSE_CODE%type,
mara out studys.mark%type)
as
begin
select mark
into mara
from studys
where NO_STU=stu_id
```

```
and COURSE_CODE=cou;
end;
/
```

بعد ذلك نقوم باستدعاء الاجراء ومن ثم طباعة الدرجة لان لو لاحظت الاجراء لايقوم بالطباعة ولاحظ ايضا ان الدرجة تم وضعها في المتغير mara ولذلك سوف يعود بهذه القيمة وسوف يكون الاستدعاء كمايلي :

```
declare
m studys.mark%type;
begin
stu_mark1(111,'225CS',m);
DBMS_OUTPUT.PUT_Line(m);
end;
/
```

وسوف يكون الناتج هو ٩٠ وهذا صحيح بناء على الجدول.

تكلمنا في مامضى عن الاجراءات المخزنة واليوم لدينا درس مشابه له وهو الوظائف المخزنة لكن الفرق ان الوظائف لابد ان تعيد قيمة بعد التنفيذ

والصسغة العامة لتكوين وظيفة كمايلي:

CREATE [OR REPLACE] FUNCTION function_name(متغيرات الادخال الممررة والاخراج ومتغيرات)
RETURN datatype

حيث تمثل function_name اسم الوظيفة المستخدمه.

اما REPLACE OR فهي توضع حينما تعلم ان الاجراء موجود من السابق.

اما عن المغيرات التي بين القوسين فهي اما متغيرات مدخله مثل اذا كان لديك اجراء حساب معدل طالب وتريد تمرير رقم الطالب الذي تريد حساب معدله فهذه هي تعتبر كمدخلات ، وهي بنفس الطريقة التي تعاملنا بها مع الاجراءت المخزنة لاتغير على المتغيرات وطرق تعريفها.اما RETURN datatype فهي تدل على نوع القيمة المعادة من الوظيفة .

مثـــــال : في الجدول الذي قمنا بدراسته في الدرس الرابع وكان بأسم studys وكان كمايلي :

NO_STU	COURSE_CODE	MARK	POINT
111	216CS	88	13.5
222	225CS	75	10.5
333	225CS	40	3
111	225CS	90	14.25
222	216CS	78	10.5
333	216CS	85	13.5

لو اردنا تصميم وظيفة ترجع بمعدل الطالب الفصل اي يتم تمرير رقم الطالب الى الوظيفة ثم يتم حساب المعدل الفصلي للطالب

ويتم حساب المعدل الفصل للطالب كمايلي =مجموع النقاط ÷ مجموع عدد الساعات لمقررات

ولانشاء الوظيفة كمايلي :

```
1  create or replace function stu_avea(stnum in studys.NO_STU%type)
2  return real
3  as
4  hour courses.hours%type;
5  avrage number(4,2);
6  sum_hours courses.hours%type:=0;
7  point studys.POINT%type;
8  total_Point studys.POINT%type:=0;
```

```
 9 codem courses.CODE%type;
10 cursor sumpoint
11 is
12 select COURSE_CODE,POINT
13 from studys
14 where NO_STU=stnum;
15 begin
16 open sumpoint;
17 loop
18 fetch sumpoint into codem,point;
19 exit when sumpoint%notfound;
20 select hours
21 into hour
22 from courses
23 where code=codem;
24 total_Point:=total_Point+point;
25 sum_hours:=sum_hours+hour;
26 end loop;
27 close sumpoint;
28 avrage:=total_Point/sum_hours;
29 return avrage;
30 end;
```

الشرح :

السطر رقم ١: لتعريف الوظيفة
السطر رقم ٢: نوع القيمة التي سوف ترجع بها الوظيفة
السطر رقم ٤: تعريف متغير عدد الساعات وهو نفس حقل عدد ساعات المقرر الموجودة في جدول courses
السطر رقم ٥: تعريف متغير الذي سوف نضع به المعدل
السطر رقم ٦: تعريف متغير لكي يوضع به مجموعات الساعات للطلاب في كل المواد
السطر رقم ٧: تعريف متغير لكي يوضع به عدد نقاط الطلاب في اي مقرر
السطر رقم ٨: تعريف متغير لكي يوضع به مجموع عدد نقاط الطالب في كل المقرر
السطر رقم ٩: تعريف متغير لكود المادة
السطر رقم ١٠: تعريف مؤشر صريح للحصول على كود المادة لكي نستفيد منه في الحصول على عدد الساعات وعدد النقاط في ذلك المقرر لكي نضيفها الى مجموع النقاط
السطر رقم ١٦: فتح هذا المؤشر لكي نتعامل معه
السطر رقم ١٧: الدخول على حلقة لكي نمر على جميع الجدول
السطر رقم ١٨: تحديث قيم المؤشر للسجل الحالي في المتغيرات codem,point
السطر رقم ١٩: شرط انهاء الحلقة وهو اذا لم يجد اي سجل في المؤشر
السطر رقم ٢٠: مؤشر ضمني لكي يقوم بالحصول على عدد ساعات الطلاب في المقرر الموجود حاليا

في المؤشر الصريح ويضع عدد الساعات في المتغير hour
السطر رقم ٢٤: اضافة عدد النقاط للمقرر الحالي الى مجموع النقاط السابق
السطر رقم ٢٥: اضافة عدد الساعات للمقرر الحالي الى مجموع الساعات السابق
السطر رقم ٢٦: الخروج من الحلقة
السطر رقم ٢٧: انهاء المؤشر الضمني
السطر رقم ٢٨: حساب المعدل وهو مجموع النقاط تقسيم مجموع عدد الساعات
السطر رقم ٢٩: الرجوع بقيمة المعدل
السطر رقم ٣٠: الانهاء

الان بعد الانتهاء من شرح طريقة تصميم الوظيفة جاء دور طريقة الاستدعاء :

لكن قبل الاستدعاء لنحسب يدويا معدل الطالب الذي رقمه ١١١ مثل لكي نقارنه بالنتائج بعد الاستعلام :

مجموع نقاط الطالب = ١٣,٥+١٤,٢٥= ٢٧,٧٥

مجموع عدد الساعات = (عدد ساعات المقرر CS٢١٦) + (عدد ساعات المقرر CS٢٢٥)
= ٣ + ٣ = ٦

وبالتالي فان معدل الطالب = ٢٧,٧٥ ÷ ٦ = ٤,٦٣

لكن الان دعنا نستدعي الدالة ونشاهد النتائج

```
SELECT distinct(NO_STU),stu_avea(no_stu)
from studys
where no_stu=111;
```

لاحظ كيف تم استدعاء الدالة من خلال الاستعلام ولاحظ استخدام الدالة distinct وهي لعدم تكرار السجل واليك النتائج :

```
NO_STU          STU_AVEA(NO_STU)
------     ;    ----------------
111                    4.63
```

لاحظ لو كان الاستعلام بدون وجود الدالة distinct فسوف يتكرر رقم الطالب عدد ظهوره في الجدول لذلك لوكان كمايلي :

```
SELECT NO_STU,stu_avea(no_stu)
from studys
where no_stu=111;
```

فان النتائج ستصبح هكذا

NO_STU	STU_AVEA(NO_STU)
;	
111	4.63
111	4.63

وهذا سبب ظهور الدالة distinct

المخزنة. لكن مارأيك لو وجد لدينا قاعدة بيانات كبيرة جدا ولنضرب مثال انها تحتوي تعلمنا سابق كيفية انشاء الاجراءت والوظائف تجاري ضخم يحتوي على بيانات على ٥٠ اجراء او وظيفة وظيفة او اجراء لها عمل خاص ولنفرض ان هذه القاعدة هي لمحل وبيانات المخزون وغيرها من بيانات ، ولذلك فان بعض هذه الاجرائيات العملاء وبيانات الموظفين وبيانات الاصناف التجارية اجرئيات خاصة مختص بالعملاء مثلا وجود اجراء لحساب اجمالي عميل وغيرها من الاجرائيات ، ومثل وجود والوظائف المخزنة واضافة العلاوات وغيرها ايضا ، لكن وضعها في هذا بالموظفين مثلا اجرائية خاصة بحساب راتب الموظف بعد حذف الحسومات مجموعة لك بعض الارباك لذلك مارأيك بان تجمع كل الوظائف والاجرائيات الخاصة بكل قسم في الشكل في قاعدة البيانات قد يسبب نجمع كل اجرائيات والوظائف الخاصة بالعملاء في حزمة خاصة مثلا **package** لوحدها وهذه المجموعة تدعي الحزمه

فوائد استخدام الحزمة :

١ - تجميع وحدات **pl/sql** المرتبطة.
٢- اداء افضل.
٣- تكون السرية افضل.
٤- اهم شيء هو في عملية الصيانه حيث تسهل عملية الصيانة باستخدام الحزم.

مكونات الحزم :

ويحتوي على التعاريف مثل متغيرات او مؤشرات او اسماء **specification** تتكون الحزمة من جزءين الاول وهو الوصف .ومتحولتها الاجراءت
والعمليات وغيرها اما الجز الثاني فهو جسم الحزمة ويحتوي على تفاصيل الاجراءت
: والصيغة العامة لانشاء الجزء الاول كمايلي

```
CREATE OR REPLACE PACKAGE pack_name AS
.....
.....
.....
end;
```

والصيغة العامة لانشاء الجزء الثاني كمايلي :

```
CREATE OR REPLACE PACKAGE BODY pack_name AS
.....
الحزمة جسم
.....
end;
```

.الجزء الاول هو نفس اسم الحزمة في الجزء الثاني لكن يجب ان يكون اسم الحزمة في

مـــــــــــثال :

طالب واجراء لطباعة المعدل لنقم بانشاء حزمة تحتوي على وظيفة لحساب معدل
والتي تقوم بحساب معدل الطالب **stu_avea** السادس والتي اسمها ولذلك سوف نستخدم نفس الوظيفة التي انشناها في الدرس
: كمايلي **specification** الحزمة . الجزء الاول من الحزمة والان نبداء بانشاء

```
CREATE OR REPLACE PACKAGE student AS
function stu_avea(stnum in studys.NO_STU%type)return real;
procedure print_ave(avrage in real);
end;
```

الان نقوم بانشاء جسم الحزمة والتي تحتوي على التفاصيل.كمايلي

```
CREATE OR REPLACE PACKAGE BODY student AS
function stu_avea(stnum in studys.NO_STU%type)
return real
as
hour courses.hours%type;
avrage number(4,2);
sum_hours courses.hours%type:=0;
point studys.POINT%type;
total_Point studys.POINT%type:=0;
codem courses.CODE%type;
cursor sumpoint
is
select COURSE_CODE,POINT
from studys
where NO_STU=stnum;
begin
open sumpoint;
loop
fetch sumpoint into codem,point;
exit when sumpoint%notfound;
select hours
into hour
from courses
where code=codem;
total_Point:=total_Point+point;
sum_hours:=sum_hours+hour;
end loop;
close sumpoint;
avrage:=total_Point/sum_hours;
return avrage;
end;

procedure print_ave(avrage in real)
as
begin
DBMS_OUTPUT.PUT_LINE(avrage);
end;
end;
```

ويحتوي جسم الحزمة كمانلاحظ على مكونات الوظيفة والاجراء الذي تم تعريفهما في وصف الحزمة حيث ان الوظيفة لحساب المعدل والاجراء لطباعة المعدل.

***** طريقة استدعاء اجراء او وظيفة موجود داخل حزمة :**

تتم عملية الاستدعاء كمايلي : **pack_name.func_proc_name**

اي اسم الحزمة اولا ثم نقطة ثم اسم الاجراء او الوظيفة مـــــثال :

```
set serveroutput on
declare
aa real;
begin
aa:=student.stu_avea(111);
student.print_ave(aa);
```

```
end;
/
```

وبعد التفيذ يكون الناتج هو معدل الطالب الذي رقمه ١١١ لاحظ اول شي استدعينا داله حساب المعدل ووضعناها في المتغير aa ثم استدعينا اجراء الطباعة ليتم طبعة على الشاشة.

من الان فصاعد ان تستخدم الحزم في كتابة الاجرائيات والوظائف والان و بعد ان تعرفت على فائدة الحزم مارأيك