سلسلة نجم الويب

الاصدار الثاني

Data Base

Program with

PL/SQL

تم اعداد هذا الكتـــــــــاب بواسطة:

WEBSTAR

تمت المساعده في مراجعة هذا الكتاب من قبل:

ESSAM MOBARAK BAKER

2007

E_mail: Web_star10@yahoo.com

الاصدار الثاني لقواعد بيانات قسم PL/SQL والذي يهتم بشرح الاكواد والدروس المتضمنه لهذا القسم

هذا الكتاب لا يعتبر مرجعاً لهذه اللغة ولكنه جزء منها

ارجوا من الله ان يوفقنا في جمع هذه الدروس وفهمها واتقانها

لكل من يستعمل هذا الكتاب

مع خالص تحياتي

WebSt@r

e-mail: web_star10@yahoo.com

Mobile: 7 1 1 2 7 3 9 3 4

هذا الكتاب مجاني ولا يجوز بيعه

خاص بـــنجم الويب

كلية علوم وهندسة الحاسوب

مستوى ثاني

جامعة الحديدة

الإهداء

إلى تلـــك الأنوار التي نزلت إلينـا لتنيـــر عقولنـــا
إلى تلـــك الشمعة المضيئة التي احترقت وتحترق من اجلنا.

إلى تلـــك الروائـــح العطـــرة التي فتحـــت قلـــوبنـا
إلى تلـــك الطيـــور التي غـــردت فرحـــاً بنجاحنـــا
إلى تلـــك الورود التي ترسل روائـــح رحيقها إلى قلـــوبنا
إلى تلـــك الأســـود التي تعبـــت مـــن اجلنـــا
إلى أمـــــــي الـــــــتي وفرت لي سبل الراحة
إلى أبـــــي الذي تحمل أعبائي و شجعني على مرارة الحياة

إلى من مد لي يد العون في مساعدتي على عمل هذا الكتاب
زميلي/ عصـــام مبارك

إهدائي الخاص
إلى أستاذي الدكتور/ طاهر الرشاحي

الفهرس

لغة الإجراءات لبرمجة أوراكل

تعتبر هذه اللغة هي لغة برمجية بغض النظر عن جمل الاستعلامات فهذه لغة استعلامات وهذه لغة برمجية ويمكنك من طريقها عمل الإجرائيات المخزنة والوظائف ويمكنك استخدام أوامر وجمل جمل الاستعلام داخلها إذ أنها وضعت بالأساس لمعالجة بيانات معينة يتم الحصول عليها من قاعدة بيانات أو عند إدخالها كمعطيات لتغذية هذه القاعدة ومن الممكن اعتبار أن الإجرائيات هي عبارة عن كتل برمجية مستقلة من الممكن أن يتم استدعاء أي منها أو أكثر من كتلة في عدة إجراءات لعمل عمليات معينة متخصصة في كل إجرائية ولشرح مبسط عن ذلك إليك هذا المثال :لنفرض أننا قمنا بإنشاء إجرائية تقوم بحساب مجموع رواتب الموظفين وإجرائية أخرى تقوم باقتطاع التأمين الصحي من الرواتب وإجرائية أخرى تقوم بعمل الخصومات وأخرى تقوم بحساب المكافئات فيمكننا استخدام كل من هذه الإجرائيات على حدى للقيام بعملية معينة كما ويمكننا عمل إجرائية تقوم باستدعاء هذه الإجرائيات واحد تلو الأخر لكي تقوم بالنهاية بإعطائنا النتائج المطلوبة

ميزات الإجراءات

كل إجرائية معدة مسبقا مرة واحدة عدم تكرر الأوامر في أكثر من زناد وإجرائية بحيث نتمكن من استدعاء

سهولة إصلاح وتتبع الأخطاء بحيث نقوم بتعديل إجرائية واحدة فيتم إصلاح جميع الإجرائيات التي تقوم عليها بدلا من إصلاح كل إجرائية على حدا

تمكين عدة أشخاص من عمل مشروع مشترك بحيث ليس من الواجب أن تتفق الأفكار البرمجية لكل المبرمجين بحيث يقوم كل شخص بعمل إجرائية لأداء وظيفة معينة ثم يقوم شخص ما بتجميع هذه الإجرائيات لبناء المشروع كاملا

كما وأن هناك من الميزات الأخرى ولكن هذه هي أهم الميزات التي وضعت لأجلها هذه الإجراءات

كما وإن لغة الإجراءات من الممكن أن تستخدم في نفس قاعدة البيانات أي ككتل مضافة لنظام قاعدة البيانات أو يمكن إعدادها في محرر وباني الإجرائيات أو من الممكن أن نستخدمها لبرمجة الأحداث في الديفلوبر في تصميم النماذج حيث أن برمجة الأحداث هي برمجة تقوم بإصدار حدث معين عند قيام المستخدم بعمل معين مثلا عند القيام على ضغط زر أو عند القيام بإدخال بيانات في مربع نصي وغيرها من الأحداث الكثيرة ويوجد الكثير من هذه الأحداث التي تتشابه فيما بينها في الخواص مثل حدث النقر والنقر المضاعف والمرور بالفأرة فوق الأداة أو ضغط زر أو كتابة جملة وغيرها من الأحداث والتي من الممكن أن نختلف بعضها من أداة لأخرى على حسب وظيفتها

بنية الأجراء من اجل فهم الدوال:

تعتمد الإجراءات على تقسيم نفس الأجراء إلى عدة أقسام وهي

قسم التعريف

في هذا القسم يتم تعريف نوع الأجراء هل هي وظيفة أو إجرائية ويتم تعريف أنواع وأسماء المتحولات التي يتم التعامل معها في الإجرائية

جسم الأجراء

يعتبر جسم الأجراء هو لب هذه الإجراءات والذي يتم فيه كل العمليات البرمجية كما ويمكنك إدخال بنية إجرائية أخرى داخله

معالجة الأخطاء

هذا القسم يقوم بعمل مصيدة للأخطاء المتوقعة والغير متوقعة والذي يتم تعريفها وصيادة هذا الخطاء ومعالجته وتوجيهه للقيام بأعمال معينة على أساس الخطاء القادم

نبذه عن مكونات PL/SQL:

PL/SQL برامج تتم كتابتها في كتل من اوامر البرمجة تحتوي على مقاطع منفصلة للاعلان عن المتغيرات واوامر البرامج ومعالجة الاستثناءت (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 (Optionalاختياري)
Variables, cursors, user-defined exceptions
المتغيرات ،مؤشرات،استثناءات معرفة من قبل المستخدم
BEGIN (Mandatory إلزامي)
– SQL statements
└ PL/SQL statements
EXCEPTION (Optional إختياري)
Actions to perform when errors occur
أوامر تفعل عندما تحدث أخطاء
END; (Mandatory إلزامي)

مع ملاحظة مايلي:

ان قسم declare وقسم exception هما اختياريين ولا يشترط وجودهما

أي اذا كان لا يوجد لديك تعريف متغيرات لا تستخدم declare واذا كنت لا تتعامل مع الاخطاء لاتستخدم exception.

بعض القواعد المهم فهمها قبل الدخول للبرمجة:

#قواعد التسمية:

```
DECLARE
  employee_id  NUMBER(6);
BEGIN
  SELECT     employee_id
  INTO       employee_id
  FROM       employees
  WHERE      last_name = 'Kochhar';
END;
/
```

الاسم المؤشر عليه `employee_id`

*متغيران يمكن ان يأخذوا نفس الاسم ،المدخلان في اكواد مختلفة.

*اسم المتغير(المعرف) لا يجب ان يكون ايضا مثل اسم حقول الجدول المستخدم في الكود.

خاصية %TYPE :

*يعلن المتغير طبقا لـ:

-أي تعريف عمود لقاعدة البيانات.

-متغير معلن موجود مسبقا في جدول اخر.

*تبدأ %TYPE مع :

-جدول وعمود لقاعدة بيانات.

-اسم المتغير المعلن سابقاً.

مثال:

```
...
  v_name           employees.last_name%TYPE;
  v_balance        NUMBER(7,2);
  v_min_balance    v_balance%TYPE := 10;
...
```

استخدام ربط المتغيرات:

لإرجاع المتغير المرتبط في PL/SQL يجب ان يتقدم اسم المتغير الرمز (:)

مثال:

```
VARIABLE      g_salary NUMBER
BEGIN
   SELECT     salary
   INTO       :g_salary
   FROM       employees
   WHERE      employee_id = 178;
END;
/
PRINT g_salary
```

#دالة DBMS_OUTPUT.PUT_LINE :

لإجراء عرض البيانات المراد عرضه في هذه الداله يجب اولا وقبل كل شيء تفعيل عرض شاشة SQL* ولو مره واحد فقط عند الدخول بهذا الكود:

Sql>SET SERVEROUTPUT ON

لمنع مرور مترجم SQL في جزء معين من البرنامج:.

-اذا كنت لا تريد الكومبايلر من المرور على سطر واحد معين اعمل قبل السطر الرمز(- -)شرطتين.
-اذا كنت لا تريد مترجم SQL(الكومبايلر) من المرور على اكثر من سطر يمعنى (ملاحظاتك على البرنامج)
وضع كلام ليس له عمل في البرنامج :ضع هذا الكلام بين الرمزين (/* STATEMENT */)
مثال:


```
DECLARE
...
  v_sal NUMBER (9,2);
BEGIN
  /* Compute the annual salary based on the
     monthly salary input from the user */
  v_sal := :g_monthly_sal * 12;
END;      -- This is the end of the block
```

للاستعلام عن بيانات من قاعدة البيانات مع جمل SELECT :

الشكل البنائي:

```
SELECT   select_list
INTO     {variable_name[, variable_name]...
         | record_name}
FROM     table
[WHERE   condition];
```

مثال عملي

```
DECLARE
  v_deptno            NUMBER(4);
  v_location_id       NUMBER(4);
BEGIN
  SELECT      department_id, location_id
  INTO        v_deptno, v_location_id
  FROM        departments
  WHERE       department_name = 'Sales';
  ...
END;
/
```

استعلام عن تأريخ توظيف وراتب لموظف معين :

```
DECLARE
  v_hire_date    employees.hire_date%TYPE;
  v_salary       employees.salary%TYPE;
BEGIN
  SELECT    hire_date, salary
  INTO      v_hire_date, v_salary
  FROM      employees
  WHERE     employee_id = 100;
  ...
END;
/
```

استعلام لإرجاع مجموع الرواتب لجميع العاملين في قسم محدد:

```
SET SERVEROUTPUT ON
DECLARE
  v_sum_sal    NUMBER(10,2);
  v_deptno     NUMBER NOT NULL := 60;
BEGIN
  SELECT       SUM(salary)   -- group function
  INTO         v_sum_sal
  FROM         employees
  WHERE        department_id = v_deptno;
  DBMS_OUTPUT.PUT_LINE ('The sum salary is ' ||
                        TO_CHAR(v_sum_sal));
END;
/
```

(اضافة بيانات)اضافة معلومات موظف جديدة لجدول الموظفين:

```
BEGIN
  INSERT INTO employees
  (employee_id, first_name, last_name, email,
   hire_date, job_id, salary)
  VALUES
   (employees_seq.NEXTVAL, 'Ruth', 'Cores', 'RCORES',
    sysdate, 'AD_ASST', 4000);
END;
/
```

(تحديث بيانات)زيادة راتب كل الموظفين الذين يعملون Stock Clerks :

```
DECLARE
  v_sal_increase   employees.salary%TYPE := 800;
BEGIN
  UPDATE      employees
  SET         salary = salary + v_sal_increase
  WHERE       job_id = 'ST_CLERK';
END;
/
```

(حذف بيانات) احذف الصفوف التي تعود الى القسم ١٠ من جدول الموظفين:

```
DECLARE
  v_deptno   employees.department_id%TYPE := 10;
BEGIN
  DELETE FROM   employees
  WHERE         department_id = v_deptno;
END;
/
```

(دمج الصفوف) ادخل او حدث الصفوف في جدول COPY_EMP لمطابقه بجدول الموظفين:

```
DECLARE
      v_empno employees.employee_id%TYPE := 100;
BEGIN
MERGE INTO copy_emp c
     USING employees e
     ON (e.employee_id = v_empno)
   WHEN MATCHED THEN
     UPDATE SET
       c.first_name     = e.first_name,
       c.last_name      = e.last_name,
       c.email          = e.email,
       . . .
   WHEN NOT MATCHED THEN
     INSERT VALUES(e.employee_id, e.first_name, e.last_name,
          . . .,e.department_id);
END;
```

طرق الاسناد

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

i:=5;

يجب وضع النقطتين قبل=

هناك بعض النقاط الخاصة مثلا :

يوجد ضمن اوامر sql الامر DBMS_OUTPUT.PUT_LINE

يستخدم لكي تعرض في النتيجة sql * plus والصيغة العامه 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 الpl/sql لكن في المستقبل سوف تعرف ان هذا الامر لايهمك كثيرا.مثال:

نريد مثلا طباعة "Welcome in PL/SQL" على شاشة SQL*PLUS يتم ذلك كمايلي:

```
SET SERVEROUTPUT ON;
BEGIN
DBMS_OUTPUT.PUT_LINE(' Welcome in PL/SQL');
End;
```

جرب هذا ولاحظ النتائج ولعلك تتسال عن سبب وجود السطر الاول

السطر الاول يخبر sql*plusبأن 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 || الموجودة ضمن عمل الطباعة هي للوصل بين التعبيرين

اوامر اللغة:

١- الشرط عبارة if then

تستخدم هذه العبارة مثل اي العبارت الشرطية في لغة السي 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
```

اما ***conditonal*** فهي عبارة عن الشرط ونأتي بمثال لذلك:

مثال :اذا كان الاسم الاخير فارجس والراتب اكثر من ٦٥٠٠

```
. . .
IF v_ename  = 'Vargas' AND salary > 6500 THEN
   v_deptno := 60;
END IF;
. . .
```

مثال اخر:

اذا كان الاسم الاخير فارجس اعمل الاتي:

*ضع مكان الوظيفة اسم SA_REP.
*ضع مكان القسم الرقم ٨٠.

```
. . .
IF v_ename       = 'Vargas' THEN
   v_job         := 'SA_REP';
   v_deptno      := 80;
END IF;
. . .
```

```
Sql>Declare
2       i number(5);
3       BEGIN
4       i:=5;
5       IF i=5 then
6       DBMS_OUTPUT.PUT_LINE('i = ' || i);
7       ELSE
8       DBMS_OUTPUT.PUT_LINE('i not eqal 5 ');
9       END IF;
10      END;
11      /
```

الشرط باستخدام اكثر من شرط

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

ملاحظة هامة: لكن لاحظ هنا فقط كتابة **ELSIF** ولاتخطئ فيها فهي ليست**elseif**

مثال يبين الشرط:

```
Sql>Declare
2       i number(5);
3       BEGIN
4       i:=5;
5       IF i>1 then
6       DBMS_OUTPUT.PUT_LINE(i || ' > 1');
7       ELSIF i<1 then
8       DBMS_OUTPUT.PUT_LINE(i || ' < 1');
9       ELSIF i=1 then
10      DBMS_OUTPUT.PUT_LINE(i || ' = 1');
11      END IF;
12      END;
13      /
```

النوع الثاني باستخدام الشرط CASE:

الشكل البنائي:

```
CASE selector
   WHEN expression1 THEN result1
   WHEN expression2 THEN result2
   ...
   WHEN expressionN THEN resultN
  [ELSE resultN+1;]
END;
```

مثال:

```
SET SERVEROUTPUT ON
DECLARE
   v_grade CHAR(1) := UPPER('&p_grade');
   v_appraisal VARCHAR2(20);
BEGIN
    v_appraisal :=
      CASE v_grade
         WHEN 'A' THEN 'Excellent'
         WHEN 'B' THEN 'Very Good'
         WHEN 'C' THEN 'Good'
         ELSE 'No such grade'
      END;
DBMS_OUTPUT.PUT_LINE ('Grade: '|| v_grade || '
                      Appraisal ' || v_appraisal);
END;
/
```

٢- التكرار:

ويوجد عدة اشكال (اوامر) للتكرار وهي:
الشكل الاول * loop-exit-end
وهنا لابد من وضع شرط لأنهاء الحلقة

الشكل البنائي:

```
LOOP                                -- delimiter
  statement1;                       -- statements
  . . .
  EXIT [WHEN condition];            -- EXIT statement
END LOOP;                           -- delimiter
```

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 وهي1 =i
السطر الخامس : شرط الانهاء
السطر السادس : الانهاء اذا كان10 <i 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 تتبع البرنامج كمايلي:
i =1
i =2
I=3
i =4
i =5
i =6
i =7
i =8
i =9
i =10

الشكل الثاني:
FOR Loops
*استعمل حلقة FOR لاختصار اختبار لعدد التكرار.
*لا يعلن العداد ،هو معلن ضمنياً

الشكل البنائي

```
FOR counter IN [REVERSE]
    lower_bound..upper_bound LOOP
  statement1;
  statement2;
  . . .
END LOOP;
```

مثال: ادخل هويات ثلاث مواقع جديدة لرمز البلد CA ومدينة مونتريال.

```
DECLARE
  v_country_id    locations.country_id%TYPE := 'CA';
  v_location_id   locations.location_id%TYPE;
  v_city          locations.city%TYPE := 'Montreal';
BEGIN
  SELECT MAX(location_id) INTO v_location_id
    FROM locations
    WHERE country_id = v_country_id;
  FOR i IN 1..3 LOOP
    INSERT INTO locations(location_id, city, country_id)
    VALUES((v_location_id + i), v_city, v_country_id );
  END LOOP;
END;
/
```

حلقة WHILE:

استعمال الحلقة التكراريه WHILE لتكرار البيانات بينما الشرط صحيح. والشرط دائما في بداية الحقله.

الشكل البنائي:

```
WHILE condition LOOP
  statement1;
  statement2;
  . . .
END LOOP;
```

Condition is evaluated at the beginning of each iteration.

مثال:

```
Sql>DECLARE
2     v_country_id locations.country_id%TYPE := 'CA';
3     v_location_id locations.location_id%TYPE;
4     v_city locations.city%TYPE := 'Montreal';
5     v_counter NUMBER := 1;
6     BEGIN
7     SELECT MAX(location_id) INTO v_location_id FROM locations
8     WHERE country_id = v_country_id;
9     WHILE v_counter <= 3 LOOP
10    INSERT INTO locations(location_id, city, country_id)
11    VALUES((v_location_id + v_counter), v_city, v_country_id);
12    v_counter := v_counter + 1;
13    END LOOP;
14    END;
15    /
```

انشاء سجل في PL/SQL :

الشكل البنائي:

```
TYPE type_name IS RECORD
     (field_declaration[, field_declaration]…);
identifier    type_name;
```

اعلن المتغيرات لخزن الاسم ،وظيفة،وراتب موظف جديد.

```
...
  TYPE emp_record_type IS RECORD
    (last_name    VARCHAR2(25),
     job_id       VARCHAR2(10),
     salary       NUMBER(8,2));
  emp_record      emp_record_type;
...
```

خاصية %ROWTYPE:

مثلا عند إعلان متغير لخزن معلومات حول القسم من جدول الأقسام بالكود:

```
Sql>dept_record departments%ROWTYPE;
```

وعند إعلان متغير لخزن معلومات حول موظف من جدول الموظفين بالكود:

```
Sql>emp_record employees%ROWTYPE;
```

ومن أهم أوامر PL/SQL التي سنشرحها

٣- المؤشرات CURSORS:

تستخدم pl/sql المؤشرات cursors لإدارة عبارات التحديد select لغة في 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 فقط حيث لا يمكن استخدام الكلمات select المعرفه على عبارة التحديد sql عبارة الاساسيةinsert,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 الامرFETCH كمايلي :

```
FETCH اسم المؤشر INTO .....متغير١ ،متغير٢
```

ومعنى هذا اي قم باسترجاع البيانات من المؤشر المعطى اسمه وحملها into الى المتغيرات مع 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
```

ماهي مواقع الثوابت الموجودة في اللغة:

Constant

لتعريف ثابت لا يتغير عند الانتهاء من الأجراء بل تبقى القيمة كما هي

```
Declare

:Global.kk := 1;
```

Public

لتعريف متغيير عام يقوم البرنامج وجميع الإجرائيات باستخدامه

```
Declare

:Global.kk ;
```

Normal

لتعريف قيم تستخدم في الأجراء وتنتهي عن الانتهاء من الأجراء

```
Declare

MMM_Salary Salary%Rowtype;

MMM_ID Salary.ID%Rowtype;
```

في المتغيرين السابقين قمنا بتعريف متغير يحتوي على جدول ومتغير أخر يحتوي على حقل معين من جدول

```
Declare

MM_Salary Number(10);

MM_Date Date;

MM_Name Varchar2(20);
```

هذه مثال على كيفية تعريف متغيرات عادية

بقي علينا معرفة كيفية التعامل مع هذه المتغيرات من حيث الإسناد والحصول على النتائج

إعطاء قيمة	MM := 12 ;
وضع قيمة داخل أخرى	:MM := B ;
للمساواة	MM = 12 ;
والثابتة التعامل مع المتغيرات العامة	:Global.KK + 1 ;

لنقم الآن بالقيام بعمل مثال عملي أخر يضم ما تعلمناه سابقا ولتطبقووه انتم ايضا ليس بنسخه ولصقه بل بكتابة كل سطر بأنفسكم وهذا لمصلحتكم :

```
Declare

Cursor C1

Select empno,ename,sal From emp;

Cursor C2

Select amunt From add Where empno = empno;

Cursor C3

Select amunt From dept Where empno = empno;

Begin

Open C1;

Loop

Fetch C1 into Xempno,Xename,Xsal;

Exit when C1%notfound;

Tot_Add=0

Open C2;

Loop

Fetch C2 into Xamount;

Exit when C2%notfound;

Tot_Add := Tot_add + Xamount;

End loop;

Close C2;

Open C3;

Tot_ded = 0;

Loop

Fetch C3 into Xamount;

Exit when C3%notfound;
```

```
Tot_ded :=Tot_ded + xamount;

End loop;

Close C3;

Xnet_sal = Xsal + (Tot_add – Tot_ded)

Insert into New_emp

(empno,ename,sal,Tot_add,Tot_ded,Total)

Values

(Xempno,Xenqme,Xsal,Tot_add,Tot_ded,Xnet_sal);

End loop;

Close C1;

Exception

When ni_data_found then

Text_IO.Put line( 'Data is Empty');
```

في المثال السابق قمنا بوضع عدة حلقات داخل بعضه البعض كما قمنا بحساب مكافئات الموظف والخصومات لاستخراج إجمالي الراتب الخاص بالموظف وقمنا بوضعه في جدول جديد وبهذا تكون قد وضعت أول خطواتك في علم لغة الإجراءات في حال قد فهمت هذا المثال جيدا كما قمنا أيضا بوضع مصيدة للخطاء تقوم بإخراج سطر عبارة عن رسالة تقوم أنت بتعريفها لتخرج للمستخدم

مثال على طريقة تعريف مؤشر:

افرض انه لدينا هذا الجدول

age	name	no
22	Ali	1
24	Essam	2
25	Fayez	3

اولا قم بانشاء هذا الجدول كمايلي:

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

ثانيا قم بادخال البيانات السابقة في هذا الجدول كمايلي:

```
Sql>insert into stud values(1,' Ali ',22);
Sql>insert into stud values(2,' Essam ',24);
Sql>insert into stud values(3,' Fayez',25);
```

ثم قم بتنفيذ ما يلي:

```
Sql>DECLARE
2       name_stu varchar2(40);
3       CURSOR name_student IS
4       select name from stud
5       where no=1;
6       BEGIN
7       OPEN name_student;
8       FETCH name_student INTO name_stu;
9       DBMS_OUTPUT.PUT_LINE(name_stu);
10      CLOSE name_student;
11      END;
12      /
```

بعد التنفيذ سوف يظهر لك الناتج كمايلي:

Ali

وهذا الشي صحيح لاحظ اننا اتبعنا نفس الخطوات التي ذكرناه لكي نتعامل مع مؤشر صريح
نلاحظ في المثال السابق ان الاستعلام فيcursor سوف يعود بسجل واحد لكن ماذا يحدث لو اعاد المؤشر اكثر من سجل واردنا المرورعلى 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 ان مايلي اسم المؤشر هي احد خصائصه.
found : خاصية التي من خلالها نعرف هل تم الانتهاء من جميع السجلات ام لا
مثال:

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 50 ضع قيمة true في حقل result وإلا ضع قيمة Falseفي حقل result
البرنامج كالتالي:
نقوم اولا بانشاء هذا الجدول:

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

المدخلات السابقة وبعد انشاء الجدول نقوم بادخال

```
Sql>insert into stu_study (NO_STU,SUBJECT,MARK) values (111,'216CS',88);
Sql>insert into stu_study (NO_STU,SUBJECT,MARK) values (222,'225CS',75);
Sql>insert into stu_study (NO_STU,SUBJECT,MARK) values (333,'225CS',40);
```

بعد ذلك نقوم بانشاء الاجراء:

```
Sql>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

هناك طريقة اخرى لتعريف المتغيرات لاحظ في الجدول السابق ان الحقل no_Stu تم تعريفه على انه من نوع number وثم 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 في الاجراء no

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:

```
Sql>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
end if;
end loop;
close res_stu;
end;
/
```

مثال:

```
Sql>Declare
2       Cursor C1 is
3       Select empno,empname,sal from emp;
4       ST emp%Rowtype
```

قمنا هنا بالبدء في قسم تعريف الأجراء وقمنا بتعريف مؤشر ووظيفة المؤشر هي كمخزن لمجموعة من البيانات التي يتم الحصول عليها بناء على جملة استعلام محددة

ثم قمنا بتعريف متغير أخر وجعلنا نوعه مربوط بنفس نوع الحقل بحيث يحتوي على نفس أنواع الحقول الموجودة في جدول الموظفين والفائدة من هذه الطريقة هي عند القيام بتغيير أي نوع من أنواع الحقول الموجودة في الجدول سوف يقوم الأجراء بالتعرف على النوع الجديد دون الحاجة لتعديل هذا الأجراء

```
5       Begin
6       Open C1;
7       Loop
8       Fetch C1 Into
9       St.empno , st.empname, st.sal;
10      Exit when C1%notfound;
```

```
11      End Loop;

12      Close C1;

13      End;

14      /
```

قمنا هنا في جسم الأجراء بفتح المؤشر وهذا لكي يتم تنفيذ جملة الاستعلام ووضع القيم في داخله ثم قمنا ببدء حلقة لكي تقوم بالمرور على جميع القيم الموجودة داخل هذا المؤشر وقمنا بوضع هذه القيم داخل المتحول الذي قمنا بتعريفه ثم قمنا بعمل مصيدة للخطاء وهي تقوم بالخروج من الحلقة في حال انتهت الحلقة من المرور على جميع البيانات الموجودة داخل المؤشر ثم قمنا بإغلاق الحلقة وإغلاق المؤشر وانتهينا من الأجراء

وبهذا نكون انهينا المؤشرات الصريحه

اما النوع الاخر وهو 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 الحقل ١ و الحقل ٢ وضعها في المتغيرات منغير ١ و متغير ٢ 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

age	name	no
22	Ali	1
24	Essam	2
25	Fayez	3

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:

```
Sql>set serveroutput on;
2       declare
3       aveage number(4,2);
4       begin
5       select avg(age)
6       into aveage
7       from stud;
8       DBMS_OUTPUT.PUT_LINE(aveage);
9       end;
10      /
```

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

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

code	course_name	hours
216CS	NETWORK	3
225CS	ASSEMBLY	3
325CS	DATABASE	4

وقم بإنشاء الجدول كما يلي:

```
Sql>create table courses(
2       code varchar2(8),
3       course_name varchar2(40),
4       hours number(3),
5       primary key(code));
```

وقم بادخال البيانات الموجود بالجدول كمايلي:

```
Sql>insert into courses values('216CS','NETWORK',3);
Sql>insert into courses values('225CS','ASSEMBLY',3);
Sql>insert into courses values('325CS','DATABASE',4);
```

ثم نقوم بتكوين الجدول الثاني وهو studys :

NO_STU	COURSE_CODE	MARK T	POINT
1	216CS	88	
2	225CS	75	
3	225CS	40	
1	225CS	90	
2	216CS	78	
3	216CS	85	

ويتم الانشاء كمايلي:

```
Sql>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:

```
Sq>insert into studys(NO_STU,COURSE_CODE,MARK)
2       values ('111','216CS',88);
3       insert into studys(NO_STU,COURSE_CODE,MARK)
4       values ('222','225CS',75);
5       insert into studys(NO_STU,COURSE_CODE,MARK)
6       values ('333','225CS',40);
7       insert into studys(NO_STU,COURSE_CODE,MARK)
8       values ('111','225CS',90);
9       insert into studys(NO_STU,COURSE_CODE,MARK)
10      values ('222','216CS',75);
11      insert into studys(NO_STU,COURSE_CODE,MARK)
12      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 كما في الجدول السابق *)عدد ساعات المقرر
مثلا لحساب معدل الطالب الذي رقمه ١ 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=١٣,٥وهكذا في جميع الطلاب وهذا هو المطلوب من الاجراء عمله .
وبالتالي فان الاجراء سوف يكون كمايلي:

```
sqL>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 الضمن الي 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 216cs ودرجته هي ٨٨ سوف يضع هذه البيانات في المتغيرات، ثم يستخدم مؤشر ضمني لكي يحضر
عدد ساعات المادة التي درسها الطالب ١١١ وهي CS216 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 CS216 وهي ٣ ساعات ثم بدأ يختبر الدرجة وذلك طبقا للجدول الدرجات والمعدلات حيث كانت درجة الطالب الذي رقمه ١١١ في المقرر CS216 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
1	216CS	88	13.5
2	225CS	75	10.5
3	225CS	40	3
1	225CS	90	14.25
2	216CS	78	10.5
3	216CS	85	13.5

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

في الجداول 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:

```
Sql>DECLARE
2       TYPE num_array IS TABLE OF number(4) INDEX BY BINARY_INTEGER;
3       num num_array;
4       BEGIN
5       ........
6       ........
7       END;
8       /
```

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

مثال:

```
Sql>DECLARE
2       TYPE num_array IS TABLE OF number(4) INDEX BY BINARY_INTEGER;
3       i number(4);
4       num num_array;
5       BEGIN
6       FOR i IN 1..10 LOOP
7       num(i) := i * i ;
8       END LOOP;
9       FOR i IN 1..10 LOOP
10      DBMS_OUTPUT.PUT_LINE(i || '*' || i || '= ' || num(i) );
11      END LOOP;
12      END;
13      /
```

التوضيح:

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
```

وبعد ان اخذنا تعريفات بالدوال Function ودوال الاستدعاء والمؤشرات نطبقها الان جميعا في الاتي:

مثال : في الجدول الذي قمنا والذي كان بأسم studys وكان كمايلي:

NO_STU	COURSE_CODE	MARK	POINT
1	216CS	88	13.5
2	225CS	75	10.5
3	225CS	40	3
1	225CS	90	14.25
2	216CS	78	10.5
3	216CS	85	13.5

لو أردنا 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;
31    /
```

التوضيح:

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 المتغيرhour
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
السطر رقم ٣١ : انهاء البرنامج

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 الساعات للمقرر CS216)+(عدد الساعات للمقرر CS225)
= ٣ + ٣ = ٦
وبالتالي فان معدل الطالب = ٢٧,٧٥÷٦=٤,٦٣
لكن الان دعنا نستدعي الدالة ونشاهد النتائج

```
Sql>SELECT distinct(NO_STU),stu_avea(no_stu)
2       from studys
3       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:

```
Sql>SELECT NO_STU,stu_avea(no_stu)
2       from studys
3       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

الفرق بين الإجرائية والوظيفة

الوصف	الفرق	الأسم
إجرائية	لا تقوم بإعادة قيمة	Procedure
وظيفة	تقوم بإعادة قيمة	Function

Function

```
FUNCTION name
RETURN datatype
IS
BEGIN
   --statements
   RETURN value;
[EXCEPTION]

END;
```

Procedure

```
PROCEDURE name
IS

BEGIN
   --statements

[EXCEPTION]

END;
```

Anonymous

```
[DECLARE]


BEGIN
   --statements

[EXCEPTION]

END;
```

مثال

```
Sql>Funcation CountSal

    2       Return NewSal
    3        is Mm number;

4       Begin

5       Mm=Select sal * 1.2 from emp;

6       NewSal = Mm

End;
```

هذا المثال غير عملي ولكنه يقوم بتعريف البنية الرئيسية للوظيفة

كيف يتم استخدام هذه الإجرائيات

يتم استخدام هذه الإجرائيات كا إجرائيات مخزنة داخل كائن داخل قاعدة البيانات لاستدعائها عن طريق جملة استعلام أو من الممكن أن نضعها كزناد يحدث عند حدوث أي عمل ما على جدول أو أي كائن مرتبط بهذا الزناد كما وأنك تحتاجها للقيام ببرمجة الديفلوبر في الواجهة الرسومية وهي كالأحداث الموجودة في لغة الفيجوال بيسك ولغة السكريبت فيقوم بتنفيذ إجراء معين يطلق عليه اسم الزناد عند حدوث أي حدث على قاعدة البيانات بشكل عام أو على أي كائن موجود في قاعدة البيانات وفي داخل الأدوات الموجودة في الديفلوبر كأداة الزر مثلا

كيف نقوم ببناء الإجرائية

يمكنك بناء الإجرائيات بعدة طرق منها عن طريق مستكشف

ويمكنك ذلك عن طريق أداة مرفقة مع الديفلوبر تسمى

Procedure Builder – Line mode

أو عن طريق أداة رسومية أخرى تسمى

Procedure Builder

وهي ذات واجهة رسومية تقوم بتمكينك من تجربة الإجرائية ومعرفة الأخطاء وإصلاحها

كما ويمكنك ذلك عن طريق محرر الاستعلامات

البينة لانشاء FUNCTION:

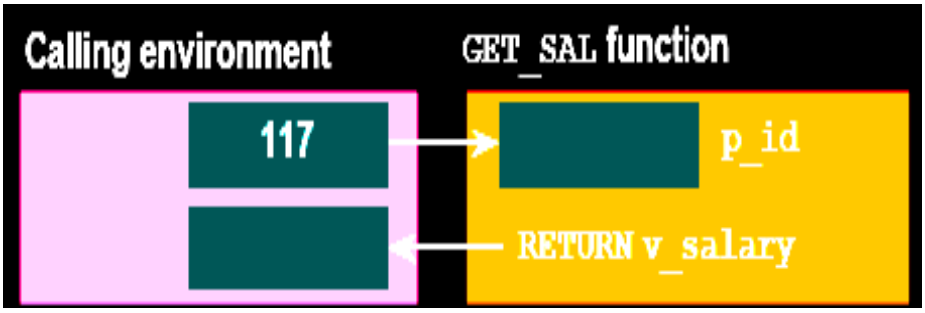
```
CREATE [OR REPLACE] FUNCTION function_name
[(parameter1 [mode1] datatype1,
parameter2 [mode2] datatype2,
. . .)]
RETURN datatype
IS|AS
PL/SQL Block;
```

مثال :انشاء استدعاء دالة باستخدام iSQL*Plus:

get_salary.sql

```
Sql>CREATE OR REPLACE FUNCTION get_sal
(p_id IN employees.employee_id%TYPE)
RETURN NUMBER
IS
v_salary employees.salary%TYPE :=0;
BEGIN
SELECT salary
INTO v_salary
FROM employees
WHERE employee_id = p_id;
RETURN v_salary;
END get_sal;
/
```

لتنفيذ الدوالFUNCTIONS :



١- تحميل وتنفيذ الملف get_salary.sql لانشاء الدالة

٢- امر لاستدعاء الملف:

```
Sql>VARIABLE g_salary NUMBER
```

٣-امر تنفيذ الملف:

```
Sql>EXECUTE :g_salary := get_sal(117)
```

٤- امر الطباعة:

```
Sql>PRINT g_salary
```

البنية لحذف الدالة:

```
Sql>DROP FUNCTION function_name
```

الاجراءات:

نظرا لضيق الوقت اختصرنا بمثال للاجراءات :

```
CREATE PROCEDURE query_employee
(p_id IN employees.employee_id%TYPE,
 p_name OUT employees.last_name%TYPE,
 p_salary OUT employees.salary%TYPE,
 p_comm OUT
   employees.commission_pct%TYPE)
AUTHID CURRENT_USER
IS
BEGIN
  SELECT last_name, salary,
         commission_pct
   INTO  p_name, p_salary, p_comm
   FROM  employees
   WHERE employee_id=p_id;
END query_employee;
/
```

اعذرونا على التقصير اختصرنا المثال والشرح بسبب ضيق الوقت
ارجوا من الله التوفيق لنا ولكم جميعاً..

الحـــــزمة package ـــــ:

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

فوائد استخدام الحزمة:

١ -تجميع وحدات pl/sql المرتبطة.
٢ -أداء أفضل.
٣ -تكون السرية أفضل.
٤ -أهم شيء هو في عملية الصيانة حيث تسهل عملية الصيانة باستخدام الحزم.

بنية الحزم:

تتكون الحزمة من جزءين الاول وهو الوصف specification ويحتوي على التعاريف مثل متغيرات او مؤشرات او أسماء الاجراءت ومتحولتها.
اما الجز الثاني فهو جسم الحزمة ويحتوي على تفاصيل الاجراءت والعمليات وغيرها

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
.....
جسم الحزمة
.....
end;
```

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

مثال:

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

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

الان نقوم بانشاء جسم الحزمة والتي تحتوي على التفاصيل.كمايلي:

```
Sql>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 كما نلاحظ على مكونات الوظيفة والإجراء الذي تم تعريفهما في وصف الحزمة حيث إن الوظيفة لحساب المعدل والإجراء 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 مثال:

```
Sql>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.

وبعد ان تعرفنا سويا على فائدة الحزم ما عليك الا التعود على كتابة الاجراءات والدوال باستخدام الحزم.

حذف حزمـــــة:

....

البنــــية:

```
Sql>DROP PACKAGE package_name;
```

حذف جسم الحزمة:

البنــــية:

```
Sql>DROP PACKAGE BODY package_name;
```

^^

الزنـــ TRIGGER ـــادات (النوابض):

تتشابه الزنادات مع البرامج الفرعية إلا في الطرق التالية:
*يتم تنفيذ الزنادات ضمنيا،عندما يعدل الجدول بالرغم من عمل المستخدم او التطبيقات على الجدول.
*يتم تعريف الزنادات للجدول الخاص بقاعدة البيانات
*لا تقبل الزنادات المعاملات
تعد الزنادات هامة جدا في تطوير نظم البيانات الموجهة الخاصة بالإنتاج.

فوائد النابض:

يمكنك استخدام النوابض للاتي:
- الامن.
- مراجعة الحسابات.
- سلامة قواعد البيانات.
- سلامة تفضيلية.
- جواب الجدول.
- اشتق استعمال حاسبات بيانات آلياً.
- تسجيل الحدث.

مثال للتحكم على الامن ضمن السيرفر (الخادم):

```
Sql>GRANT SELECT, INSERT, UPDATE, DELETE
ON employees
TO clerk; -- database role
GRANT clerk TO scott;
```

مثال للسيطرة على الامن مع نابض قاعدة البيانات:

```
Sql>CREATE OR REPLACE TRIGGER secure_emp
     BEFORE INSERT OR UPDATE OR DELETE ON employees
     DECLARE
     v_dummy VARCHAR2(1);
     BEGIN
     IF (TO_CHAR (SYSDATE, 'DY') IN ('SAT','SUN'))
     THEN RAISE_APPLICATION_ERROR (-20506,'You may only
     change data during normal business hours.');
     END IF;
     SELECT COUNT(*) INTO v_dummy FROM holiday
     WHERE holiday_date = TRUNC (SYSDATE);
     IF v_dummy > 0 THEN RAISE_APPLICATION_ERROR(-20507,
     'You may not change data on a holiday.');
     END IF;
     END;
     /
```

Auditing by Using a Trigger

```
CREATE OR REPLACE TRIGGER audit_emp_values
 AFTER DELETE OR INSERT OR UPDATE ON employees
 FOR EACH ROW
BEGIN
 IF (audit_emp_package.g_reason IS NULL) THEN
    RAISE_APPLICATION_ERROR (-20059, 'Specify a reason
     for the data operation through the procedure SET_REASON
     of the AUDIT_EMP_PACKAGE before proceeding.');
 ELSE
    INSERT INTO audit_emp_table (user_name, timestamp, id,
      old_last_name, new_last_name, old_title, new_title,
      old_salary, new_salary, comments)
    VALUES (USER, SYSDATE, :OLD.employee_id, :OLD.last_name,
      :NEW.last_name, :OLD.job_id, :NEW.job_id, :OLD.salary,
      :NEW.salary, audit_emp_package.g_reason);
 END IF;
END;
```

```
CREATE OR REPLACE TRIGGER cleanup_audit_emp
 AFTER INSERT OR UPDATE OR DELETE ON employees
BEGIN
  audit_emp_package.g_reason := NULL;
END;
```

مكونات الزناد:

اطلاق حدث المستخدم: أي بيان DML يحدث في الزناد الى التنفيذ؟ يمكننا استعمال في الزناد الاتي:
*الاضافة Insert
*التحديث Update
*الحذف Delete .

نوع الزناد:

هل يجب على جسم الزناد التنفيذ لكل صف ،بالتأثير على الجملة ام عليه لوحده؟
*الجملة:جسم الزناد ينفذ مره لإطلاق الحدث،هذا افتراضي.زناد الجملة يطلق مره،حتى اذا كان الصفوف لا تؤثر مطلقاً.
الصف: جسم الزناد ينفذ مره لكل صف أثر عليه بواسطة اطلاق الحدث. زناد الصف لا ينفذ ،اذا كان اطلاق الحدث لن يؤثر على الصفوف.
جسم الزناد: أي تنفيذ يجب على الزناد عمله؟؟ ان جسم الزناد هو PL/SQL block او استدعاء الى الاجراء.

الشكل البنائي:

CREATE [or replace] Trigger <TRRIGER_NAME>
<before|after> [instead of بدلاً من] trigger event on <table name>
[for Each row [whene triggering restriction]] عند اطلاق تقييد
<trigger body>

ملاحظة : اسماء الزناد يجب ان تكون فريدة فيما يتعلق بالنوابض الاخرى في نفس المخطط.

كماهو مع الاجراءات المخزنة امكانية استخدام replace لكي تقوم بالتعديل على الزناد اذا كان موجود ولاتقوم بانشاءه من جديد.
ينفذ التوقيت الخاص بالزناد سواء نفذ الزناد قبل او بعد اغلاق الزناد بواسطة الخيارين after وbefore لكن خيار after اكثر كفاءة لان قطع البيانات المؤثرة يجب ان تقراء منطقيا مرة للزناد ومرة لعبارة trigger

البنية لانشاء نابض او زناد على جمل تعريف بيانات اللغة DDL:

```
CREATE [OR REPLACE] TRIGGER trigger_name
timing
[ddl_event1 [OR ddl_event2 OR ...]]
ON {DATABASE|SCHEMA}
trigger_body
```

البنية لانشاء النوابض على احداث النظام:

```
CREATE [OR REPLACE] TRIGGER trigger_name
timing
[database_event1 [OR database_event2 OR ...]]
ON {DATABASE|SCHEMA}
trigger_body
```

مثال:

لنفرض انه لدينا الثلاث جداول التالية:
الاول : هو جدولplayer بيانات جميع اللاعبين في النادي سواء درجة شباب او درجة ممتاز:

no_player	name	date_birth	phone	address	levels
1	talal	11/11/1973	123456	riyadh1	1
2	mohammed	1/1/1982	654321	riyadh2	2
3	sami	1/1/1988	123789	riyadh3	2
4	yosif	12/3/1970	123123	riyadh4	1

حيث level تمثل الدرجة التي يلعب بها اللاعب حيث ١ تمثل الدرجة الاولى الممتاز - و ٢ تمثل الشباب.
ولانشاء الجدول كمايلي:

```
SQL>create table player(
2       no_player varchar2(6) primary key,
3       name varchar2(50),
4       date_birth date,
5       phone varchar2(9),
6       address varchar2(20),
7       levels number(2));
```

الثاني : هو جدول اللاعبين في درجة الممتاز وهو خاص بالرواتب واسم الجدول larg_player

no_player	level_no	salary
1	1	
4	1	

ولانشاء الجدول كمايلي:

```
SQL>create table larg_player(
2       no_player varchar2(6) primary key,
3       level_no number(2),
4       salary number(7,2));
```

الثالث : هو جدول اللاعبين في درجة الشباب وهو خاص بالرواتب واسم الجدول youth

no_player	level_no	salary
2	2	
3	2	

ولانشاء الجدول كمايلي:

```
SQL>create table youth(
2       no_player varchar2(6) primary key,
3       level_no number(2),
4       salary number(7,2));
```

الان نريد عمل زناد بحيث حينما يقوم المستخدم بادخال اسم لاعب جديد وتحديد مستواه(شباب او ممتاز) يقوم الزناد باختبار المستوى فاذا كان شباب اضاف رقم اللاعب في جدول الشباب وكذلك لوكان مستواه درجة اولى الممتاز فانه يضيف رقم اللاعب في جدولlarg_player وبذلك يكون الزناد كمايلي:

```
SQL>create or replace trigger player_age
2       before insert on player
3       for each row
4       begin
5       if inserting then
6       if :new.levels=1 then
7       insert into larg_player(no_player,level_no) values (:new.no_player,:new.levels);
8       elsif :new.levels=2 then
9       insert into larg_player(no_player,level_no) values (:new.no_player,:new.levels);
10      end if;
11      end if;
12      end;
13      /
```

بعد ذلك قم بادخال مايلي:

```
Sql>insert into player values('1','talal','11/11/1973','123456','riyadh1',1);
```

لاحظ ان المدخلات تمت على جدول player بعد ذلك اذهب وقما بالاستعلام في جدول larg_player سوف تجد انه اضاف قم اللاعب هناك.

لقد يقول احدكم لماذا لاندمج حقل salary الموجود في الجدولين الاخرين مع جدول player ولاداعي لهذا التفصيل فأقول هذا صحيح لكن انا تعمدت ذلك حتى واضح لكم طريقة التعامل مع الزنادات وسوف تشاهدون في بعض الامثلة ان الزنادات تستخدم للتحقق من القيم المدخلة مثلا لديك جدول لتسجيل بيانات اشخاص لكن يشترط ان يكون عمر الشخص الذي تريد تسجيلة اكبر من ١٦ مثلا فانه لابد من عمل زناد يتحقق من مدخلات هذا الجدول لاختبار عمر الشخص المدخل.وغيرها من الامثلة التي يجب عليك تطبيقها..

بنية حذف النوابض او الزناد:

```
DROP TRIGGER trigger_name;
```

انتهى،،،

متغيـــرات مؤشـــرة Cursor Variables:

متغيّرات مؤشّرةِ مثل سي أو مؤشرات باسكال، التي تَحْملُ موقعَ الذاكرةَ (عنوان) فقرة بدلاً مِنْ الفقرة نفسها.

- في PL/SQL ، مؤشر يعلن بالشكل Ref X، حيث أنَّ Ref مختصرة من REFERENCE و X يُؤيّدانِ نوع الأغراض.
- أي متغيّر مؤشّرةِ نوع بياناتها REF CURSOR .
- أي المؤشّر يكون ثابتاً(Static)، لكنِ متغيّر المؤشّر يكون ديناميكي(dynamic).
- متغيّرات المؤشّر تَعطيك مرونةَ أكثر

لماذا تَستعملُ متغيّراتُ المؤشّرةِ؟

أنت يُمْكِنُ أنْ تَستعملَ متغيّراتَ المؤشّرةِ لعُبُور مواقع نتيجـةِ الاستعلام بين PL/SQLخَزنَ برامجَ subprograms ومستخدمون مُخْتَلِفين.

- PL/SQL يُمْكِنُ أنْ يَشتركَ في مؤشـر Pointer إلى منطقةِ عملِ الاستعلام الذي فيه موقع النتيجةَ مَخْزُونةٌ.
- أنت يُمْكِنُ أنْ تَعْبرَ قيمةَ متغيّر مؤشّرةِ بحرية من مدى إلى آخر.
- أنت يُمْكِنُ أنْ تُخفّضَ مرورَ شبكةِ بإمتِلاك بلوك PL/SQL يَفْتحُ (أو يَغْلقُ) عِدّة متغيّرات مؤشّرةِ مضيّفِ في رحلة ذهاب وإياب وحيدة.

لتعريف نوع REF CURSOR :

```
TYPE ref_type_name IS REF CURSOR [RETURN return_type];
```

اعلن متغير مؤشر لذلك النوع :

```
ref_cv ref_type_name;
```

مثال على ماسبق:

```
Sql>DECLARE
2    TYPE DeptCurTyp IS REF CURSOR RETURN
3    departments%ROWTYPE;
4    dept_cv DeptCurTyp;
```

استخدام جمل OPEN-FOR, FETCH, and CLOSE:

- جملة OPEN-FOR تربط متغيّر المؤشّر مَع إستعلام متعدّد صفّ، يُنفّذُ الاستعلام، يعرف موقع النتيجةَ، ويَضِعُ المؤشّرة للإشارة إلى الصفِّ الأولِ لموقع النتيجةَ.
- جملة FETCH يرجع من قيمة الصفّ مِنْ موقع النتيجةَ الى استعلام متعدّد صفّ، يحدد قِيَمَ فقرات قائمةِ الاختيـار select إلى المتغيّراتِ المطابقةِ أو يختاره في الفقرة (INTO) ، زيادة باقي العداد مِن قِبل (%ROWCOUNT)، ويتقدّمُ المؤشّرَ إلى الصفِّ التالي.
- جملة CLOSE توقف متغيّر المؤشّر.

مثال على الجالـ Fetch ـــب:

```
DECLARE
TYPE EmpCurTyp IS REF CURSOR;
emp_cv EmpCurTyp;
emp_rec employees%ROWTYPE;
sql_stmt VARCHAR2(200);
my_job VARCHAR2(10) := 'ST_CLERK';
BEGIN
sql_stmt := 'SELECT * FROM employees
WHERE job_id = :j';
OPEN emp_cv FOR sql_stmt USING my_job;
LOOP
FETCH emp_cv INTO emp_rec;
EXIT WHEN emp_cv%NOTFOUND;
-- process record
END LOOP;
CLOSE emp_cv;
END;
/
```

مثال لاستعلام لأكبر راتب لكل وظيفة في كل قسم

```
Sql> declare
2    Type deptRecTyp IS RECORD;
3    ( deptno number(2),
4    dname char(10),
5    loc char(10));
6    Type dept_cursor IS REF CURSOR;
7    return dept%ROWTYPE ;
8    dept_cur dept_cursor;
9    dept_rec deptRecTyp;
10   begin
11   open dept_cur For SELECT * from dept;
12   loop
13   FETCH dept_cur INTO dept_rec;
14   exit when dept_cur%NOTFOUND;
15   DBMS_output.put_line(dept_rec.deptno ||" "||
16   dept_rec.dname||" "||dept_rec.loc);
17   CLOSE dept_cur;
18   end;
19   /
```

لا توجد أمثله او شرح عميق عن متغيرات المؤشر كثير ونادراً ما توجد في الكتب ومواقع الانترنت.

اعذروني على التقصير،،

لعمل رسائل الاخطاء المتوقعه:

الشكل البياني:

```
EXCEPTION
WHEN exception1 [OR exception2 . . .] THEN
statement1;
statement2;
. . .
[WHEN exception3 [OR exception4 . . .]
THEN
statement1;
statement2;
. . .]
[WHEN OTHERS THEN
statement1;
statement2;
. . .]
```

ابسط رسائل اخطاء:

بعض رسائل الخطاء المهمة

When Others then في حال كان هناك خطاء غير معرف

When> DUP_VAL_ON_INDEX في حال كان هناك قيم متكررة

When> No_data_found في حال عدم وجود بيانات

When> Zero Divide في حال القسمة على صفر

When> Too_Many_Rows في حال لم يكن هناك أي سجلات

When> INVALID_CURSOR في حالة ان المؤشر عاجزة

رجاءً ان تعذروني اذا كان في هناك أي تقصير مني ،،،
مع تحياتي لكم..

المصادر :

*مقدمة الى الاوراكل PL/SQL 9i
-للمؤلفان: Priya Nathan & Nagavalli Pataballa
* سلسلة تعلم :PL/SQL بقلم الاخ: ابو ابراهيم

ملاحظة:

اعلموا يا إخواني ويا أخواتي انه يمكنكم طرح استفساراتكم حول لغة ال أس كيو أل او لغة ال بي أل /أس كيو أل على بريدي الالكتروني وسوف يتم الرد عليها ان شاء الله

مع تحياتي

السيرة الذاتية:

الاسم:علي احمد علي قاسم

الجمهورية اليمنية

مواليد محافظة الحديدة مديرية الخوخة

العمر: ٢٠ سنه

طالب جامعي في كلية علوم وهندسة الحاسوب

المؤهلات: برمجة وصيانة الحاسوب مصمم مواقع

الخاتمة:

أتمنى من الله أن اكون قد وفقت في عملي هذا واتمنى من الله أيضا أن ينال هذا البحث إعجابكم ورضاكم وفي الأخير واعتذر منكم على التقصير في بعض الاشياء.

صلوا على نبي الامه وسيد الخلق

محمداً صلى الله عليه وسلم

ورجائي لكم بالدعاء لي ولجميع من صلى على نبينا الصادق الامين

والحمد لله رب العالمين....

خاص بـــنجم الويب

كلية علوم وهندسة الحاسوب

مستوى ثاني

جامعة الحديدة