نظرة عامة على الكورس

مقدمه :

- **PL / SQL هو أمتداد للغة SQL بأضافة مميزات لغات البرمجة الاجرائية.**
- **جمل التعامل والأستعلام الخاصة بلغة SQL يتم أدراجها داخل الكود الخاص بلغة PL / SQL**

مزايا لغة PL / SQL :-

أ- التكامل : وهذه اللغة تلعب دور أساسى بين اجزاء وأدوات أوراكل حيث يكتب بها اكواد (FORMS) ويتم بها برمجة أجزاء وأدوات الاوراكل.

ب- تحسين الأداء : حيث يمكن للـ PL / SQL تحسين اداء التطبيقات وذلك من خلال :

- **تجميع جمل SQL معا فى بلوك واحد (كتلة واحدة) وأرسلهما الى خادم (Database) لتنفيذها دفعة واحدة وذلك يؤدى الى أرتفاع مستوى الاداء عامة.**
- **يمكن لـ PL / SQL العمل داخل اى جزء من أجزاء وأدوات أوراكل وذلك يضيف قوة المعالجة الاجرائية الى هذه الادوات مثل oracle reports ، Oracle forms ، ..... مما يؤدى الى تحسن مستوى الأداء .**

جـ- تطوير البرنامج Modularized :

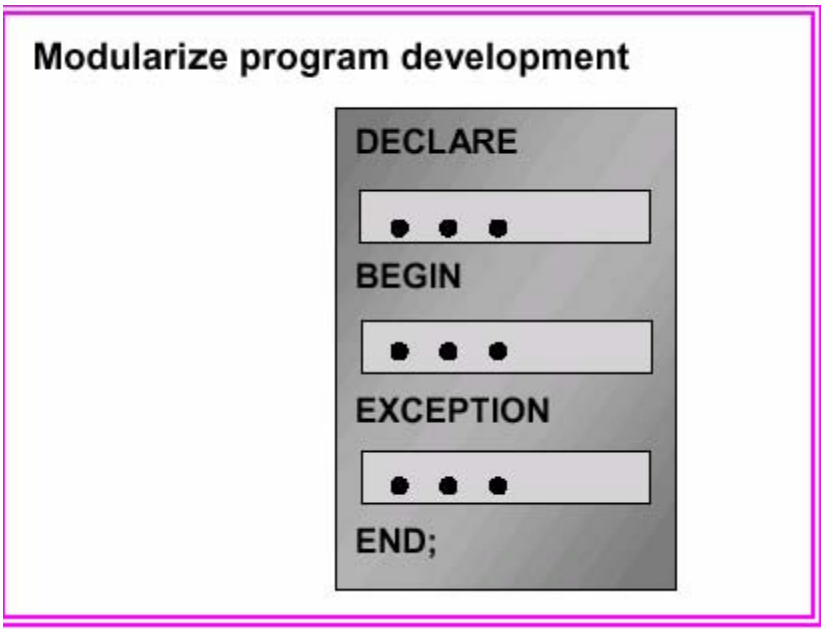
- **وذلك بتجميع منطقى للبيانات داخل كتل (Blocks) البرنامج .**
- **الكتل المتداخلة (Nested blocks) تتيح العديد من المزايا .**
- **اتاحة تقسيم المشاكل المعقدة الى مجموعة أبسط من المشاكل يمكن حلها ببساطة .**
- **الأستفادة من خبرات وأكواد سابقة بجمعها فى شكل مكتبات (Libraries) يمكن الاستفادة منها بين أدوات أوراكل المختلفة .**

د - يمكن تنفيذ الكود PL / SQL من اى أداة من أدوات اوراكل.

هـ- يمكن تعريف المتغيرات (Variables)التى تستقبل العديد من أنواع البيانات المختلفة مثل النصوص والأرقام والصور والفيديو والبيانات المركبة ..... الخ .

و - وتحتوى أيضا على المميزات لأى لغة أجرائية من حيث تواجد أوامر loop والتحكم فى سير البرنامج و معالجة الاخطاء والأستثناءات و غيرها .

- شكل ال(PL / SQL Block) :

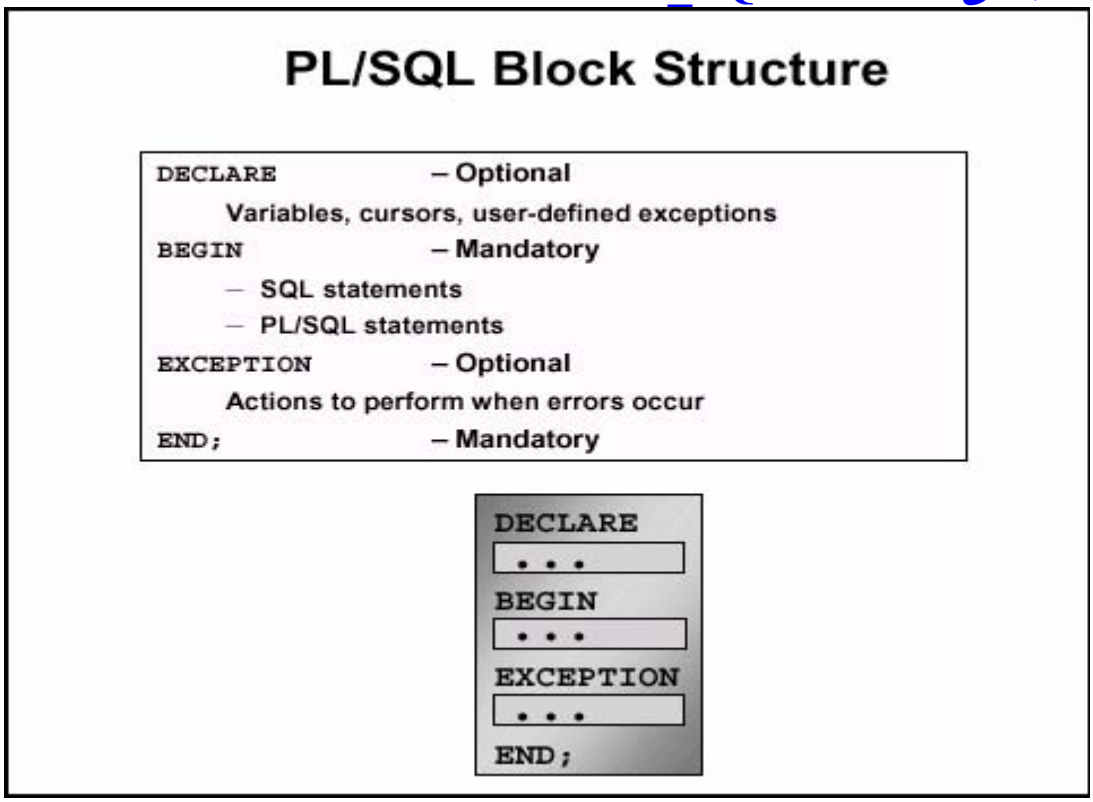


يتكون بلوك او كتلة PL / SQL من:

- جزء تعريفى للمتغيرات وهو جزء أختيارى ليس اساسى(Optional) يبدأ بـ Declare .
- جزء أخر أساسى (Mandatory) ويحتوى على جمل البرمجة يبدأ من (Begin) وينتهى بـ (End;)
- جزء أخر أختيارى (Optional) هو الـ (Exception) يحتوى على جمل المعالجة والتعامل مع الأخطاء والأستثناءات المختلفة ويبدأ من Exception .

الفصل الاول
تعريف المتغيرات

الهيكل البنائى لـ PL_SQL Block :



PL / SQL :

هى لغة هيكلية وتعنى أن البرنامج يمكن تقسيمه الى اجزاء وكتل منطقية

ويتكون برنامج PL / SQL من (Block). ويحتوى البلوك على ثلاثة أجزاء:

أ- جزء تعريف : (أختيارى) وهو يضم تعريفات المتغيرات(Variables) ، الـ Cursors، الاستثناءات المستخدمة داخل البلوك .

ب- جزء تنفيذى : (أساسى) يحتوى على جمل وعبارات اللغة لمعالجة وتناول البيانات.

جـ- جزء للأستثناءات : (أختيارى) تحدد الأعمال وماذا سيتم أذا حدث خطأ ما أو استثناء أثناء تنفيذ البلوك.

تنفيذ جمل وعبارات PL / SQL:

- **يتم وضع علامة (;) فاصلة منقوطة فى نهاية كل جملة أو عبارة من جمل لغة PL / SQL**
- **عند تنفيذ البلوك بدون اخطاء يظهر الجملة التالية لتوضيح تمام وصحة التنفيذ**

PL /SQL procedure successfully completed.

- **لاحظ عدم وجود علامة فاصلة منقوطة فى نهاية كلمة Declare ،او Exception،او Begin.**
- **لاحظ وجود علامة فاصلة منقوطة فى نهاية جملة (END ;)**
- **يمكن كتابة أكثر من جملة (عبارة) على نفس السطر والفصل بينهم بالعلامة الفاصلة المنقوطة (;) ولكن لا يحبذ ذلك لجعل قراءة البرنامج أسهل وكذلك التعديل فيه.**

- **انواع بلوكات وكتل لغة PL / SQL :**

لاحظ أن انواع الـ Blocks فى لغة PL / SQL يمكن أن تكون منفصلة كليا أو متداخلة (Nested) ضمن بعضها وتنقسم هذه الكتل الى قسمين :

1- كتلة مجهولة " Anonymous block " :

وهو برنامج ليس له أسم ويتم تعريفه عند نقطة التطبيق وتنفذ ساعتها ويتم أرسالها الى معالج أوراكل لترجمتها وتنفيذها .

2- " Subprograms" وحدات برمجية :

وهى وحدات او Blocks تقبل معاملات (Parameters) يمكن أن تستند عليها فى تنفيذ البرنامج .

وتنقسم هذه الوحدات الى (Procedure) و (Function)

وهذه الوحدات يتم حفظها داخل الأوراكل ويتم أستدعاءها عند الحاجة اليها .

Block Types

Anonymous	Procedure	Function

[DECLARE]
 	
BEGIN
--statements
 	
[EXCEPTION]
 	
END;
	PROCEDURE name
	IS
	
	BEGIN
	--statements
	
	[EXCEPTION]
	
	END;
		FUNCTION name
		RETURN datatype
		IS
		BEGIN
		--statements
		RETURN value;
		[EXCEPTION]
		
		END;

اولا الـ " Anonymous block "

- **أستخدام المتغيرات(Variables) :-**

يمكن أستخدام المتغيرات فى :

- **تخزين مؤقت للبيانات .**
- **التعامل مع قيم مخزنة .**
- **أعادة أستخدام البيانات نتيجة تغيرات داخل وأثناء سير البرنامج .**
- **سهولة التعديل والصيانة وذلك من خلال أستخدام(% Rowtype)، (% type) التى سوف تشرح لاحقا ، ومن خلالهما يمكن تعريف متغير حسب نوع عمود أو صف فى قاعدة البيانات مما يتيح قدر كبير من المرونة دون التقيد بنوع معين من البيانات .**

أنواع المتغيرات

Types of variables

تنقسم المتغيرات الى نوعين أساسيين :-

1- متغيرات PL / SQL (PL/SQL Varriables):

وتحتوى على عدة أنواع منها

- Scalar المفردة
- Composite المركبة (المعقدة)
- Reference المشار بها (عناوين)
- LOB - ذات الأحجام الكبيرة "large objects"

2- متغيرات ليست PL / SQL (Non-PL/SQL Variables):

مثل متغيرات Bind ، host

أولا المتغيرات الخاصة PL / SQL :

- المفردة Scalar : وتحتوى على قيم مفردة ولا يمكن أن تجزأ الى قيم مفردة أصغر فهى انوع لا يمكن ان يحتوى المتغير سوى على قيمه واحدة . مثل Number او Varchar2 او Data او Boolean و غيرها.
- المركبة (المعقدة التركيب) Composite :

وتحتوى على مجموعة من الأجزاء كل جزء ذات تركيب محددة ويمكن أن يختلف جزء عن جزء أخر ويتم التعامل مع كل جزء على حدى مثل Record فهو يمكن أن يحتوى على جزء من النوع Number وجزء أخر من النوع Data وهكذا ، وسيتم شرحها لاحقا .

- المشار بها (عناوين)" Reference " :

وهذه المتغيرات تحتوى قيم تشير الى برامج وتطبيقات أخرى وليست قيم بذاتها يمكن أستخدامها وهذا النوع لن يتم تغطية فى هذا المنهج .

- متغيرات ذات الأحجام الكبيرة " LOB" :

وهى تحتوى على أنواع من البيانات التى تحتاج الى مساحة كبيرة مثل الصور ، الفيديو ، الكتب ، رسومات وسوف يتم شرحها لاحقا .

ا- المتغيرات المفردة Scalar Variables

- التعامل مع المتغيرات :

- اولا يتم تعريف المتغيرات وذلك فى الجزء التعريفى "Declaration" وكذلك يمكن أن يتم وضع قيم أبتدائية لهذه المتغيرات .
- تعيين وأدخال قيم جديدة للمتغيرات فى الجزء التنفيذى .
- عرض النتائج من خلال المتغيرات .

تعريف المتغيرات المفردة Scalar:

الصيغة

Identifier (Constant) data type (not null) {:=/ default expr}

أمثلة

```
V_ Hiredate      Date;
V_deptno         number (2) not null := 10;
V_location       Varchar2 (15) :='DALLAS';
C_ comm          Contrast number: = 1600;
```

شرح الصيغة :

Identifier: هو أسم المتغير يجب أن يلتزم بقواعد التسمية.

Constant: أن المتغير المراد تعريفه هو ثابت لا تتغير قيمته التى بدء بها.

Data type: نوع بيانات المتغير سواء كان مفرد أو معقد او ذات حجم كبير(LOB) .

Not null: لا يسمح له بأخذ قيمة Null أثناء التنفيذ ويجب أن يبدأ بقيمة معينة.

Expr: وهو قيمة أبتدائية للمتغير سواء كانت قيمة ثابتة مثل '22-May-oo' او 'c' او 5 أو عملية حسابية ما.

خطوط عامة لتسمية المتغيرات

- يجب أتباع قواعد التسمية الدالة على محتوى المتغير
- يجب وضع قيم أبتدائية للمتغيرات ذات طبيعة not null أو constant (ثابت) داخل الـDeclaration.
- تعريف متغير واحد فى كل سطر
- أستخدام (:=) أو تعبير (Default) لوضع قيم أبتدائية داخل المتغيرات .

أمثلة لتعريف وأستخدام المتغيرات :

مثال 1:

```
Declare
  V_hiredate   Date;
Begin
  V_hiredate:= '15-may – 1999';
End;
```

مثال 2:

```
Declare
  V_mgr number (4) Default 100;
Begin
  V_mgr := 120;
End;
```

مثال 3:

```
Declare
  V_city varchar2 (30)    not null :='oxford';
Begin
  V_City := 'Dallas';
End;
```

مثال 1- يتم فيه تعريف متغير أسمه (V_hiredate) من النوع تاريخ (Date) وذلك فى الجزء التعريفى ، ويتم وضع قيمة داخل هذا المتغير فى الجزء التنفيذى وهذه القيمة هى تاريخ (15-مايو 1999).

مثال 2: يتم تعريف متغير من النوع الرقمى (العددى) مساحته أربع خانات وذات قيمة أتوماتيكية هى(100) وذلك فى الجزء التعريفى.

وفى الجزء التنفيذى يتم وضع قيمة جديدة داخله (120)

مثال 3 : فى الجزء التعريفى :

تعريف متغير (V-city) من النوع الحرفى سعته 30 حرف من النوع (not null) (لا يأخذ قيم فارغة) ذات قيمة أبتدائية (oxford). فى الجزء التنفيذى :

يتم وضع قيمة ('Dallas') فى هذا المتغير .

الأنواع المفردة للمتغيرات : Scalar Data types

- تحجز قيمة مفردة .
- لا تحتوى على أجزاء داخلية أو تراكيب .

أمثلة على الانواع المختلفة للـ (Data types) :

Char (length): متغير حرفى ثابت السعة سواء تم ملاءها أو تركت فارغة وهذا النوع مضر فى المساحة لكنه أسرع فى التعامل.

Varchar2 (length): متغير حرفى ذات سعة معينة لكن هذه السعة متغيرة بحد أقصى ويتم ملاء المتغير بسعة النص فقط بحد أقصى سعة هذا المتغير فمثلا متغير حرفى ذات سعة 30 حرف ولم يوضع سوى 6 حروف يملأ بالحروف الستة ويتم توفير الباقى وهذا النوع مفيد فى المساحة لكن ابطئ من النوع السابق (Char).

Long: هو النوع الاساسى للبيانات النصية ذات سعة بحد أقصى 32760 بايت

Long row: وهو مثل long لكنه لا يتم التعامل به ولا يفهمه PL / SQL.

Number (p, s): وهو متغير رقمى يأخذ نطاق من خان واحدة الى 38 خانة وكذلك من كسر عشرى من - 84 الى 127

P: عدد خانات الرقم الصحيح .

S: عدد خانات الكسر العشرى .

تابع أنواع المفردة للمتغيرات :

- binary_integer : وهو النوع الرقمى الصحيح (لايأخذ كسور) ويأخذ قيم خلال ±2147483 .
- Pls_integer : وهو مثل النوع السابق لكنه أسرع ويأخذ مساحة أقل .
- Boolean : وهو نوع يمكن أن يأخذ ثلاث قيم فقط هى (true ، false ، null)
  ويستخدم فى حالات الشروط والمقارنات المنطقية فقط .
- Data : نوع المتغيرات التاريخية (الوقت) يحتوى على بيانات تاريخ أو وقت أو زمن
  وهو يبدأ من 4712 قبل الميلاد الى 9999 ميلادية .

أمثلة أخرى على تعريف المتغيرات :

```
Declare
  v_job  varchar2 (15);                    متغير حرفى سعة 15 حرف
  v_count    binary-integer:=o;            متغير رقمى صحيح يأخذ قيمة أبتدائية صفر
v_total_sal    number (7.2):= 3.17;        متغير رقمى من سبع خانات منهم أثنان كسر عشرى
v_order_date  date:= sysdate +7;           متغير زمنى يبدأ فى الأسبوع القادم
C-TAX-RATIO CONSTANT NUMBER (4.2):=17.25;
متغير ثابت رقمى مكون من أربع خانات منهم خانتان وقيمته 17.25
  V_Flag      Boolean   Not Null :=True;
متغير منطقى  " BOOLEAN" لا يأخذ فارغ "null" ويأخذ قيمة افتراضية true
```

الخاصية %type :

يمكن تعريف متغير على أساس تعريف عمود فى جدول بقاعدة البيانات او نفس تعريف متغير سبق تعريفه . فالمتغير الجديد هنا يأخذ نفس نوع البيانات "data type" للعمود او للمتغير القديم دون أخذ ما به من قيمة وهذه الخاصية تتيح قدر كبير من المرونة فى تعريف المتغيرات لأن تعريف متغير رقمى مثلا ليكون له نفس الـData Type لعمود ما (Empno فى جدول الـEmp مثلا) واذا ما دعت الحاجة لتغيير نوع البيانات فى الجدول من رقمى الى حرفى فأنه سوف يؤدى الى تعطل البرنامج المبنى على اساس انه (عمود) رقمى وليس (عمود) حرفى لذلك نستخرج خاصية type % لتلاقى هذه المشكلة . حيث يأخذ المتغير نفس الـData type لعمود معين مهما تغيرت الـ Data type لهذا العمود.

مثال

```
Declare
  V_name    emp.Ename%type;
  V_id        emp.Empno%type;
  V_sal       emp.sal%type := 1200;
```

الصيغة العامة : Identifier table.column_name% type

أستخدام : DBMS_output . put_line

- وهى Package داخلية فى الأوراكل الغرض منهما أظهار قيم ونصوص وعرض البيانات داخل بلوكات PL / SQL.
- يجب أن يكون أختيار (set serveroutput on) للعرض.

مثال:

```
Set serveroutput on
Declare
   V varchar2(50);
Begin
    V :='He is the only one can give us the help';
    DBMS_output.put_line (v);
End;
```

```
He is the only one can give us the help
PL/SQL procedure successfully completed.
```

ملاحظة هامه جدا:

لاحظ ان المتغيرات من نوع Scalar او ال PL/SQL Variables بصورة عامة.

لها نطاق (Scope) اى لها حدود لاستعمالها حيث لا يمكن استعمالها الا فى حدود نفس البلوك التى عرفت فيه.

لذلك لا يمكن التعامل مع متغير خارج البلوك الخاص به.

أستخدام المتغيرات host , Bind

لاحظ ان هذا النوع له نطاق (Scope) اوسع من ال(PL/SQL Variables) يمكن استخدامه داخل اى بلوك داخل نفس الـ(Session)

تعريف هذا النوع يكون خارج PL / SQL Block كما يأتى :

```
VARIABLE G_SAL NUMBER
BEGIN
        :G_SAL := 1200 ;
END ;
/
PRINT G_SAL ;
```

الشرح:

لاحظ عدم وجود علامة (;) فاصلة منقوطة فى أخر جملة تعريف المتغير (G_SAL) لأنها ليست جملة PL/SQL .

عند أستخدام هذا النوع نضع أمام المتغير (:) وذلك داخل كود PL / SQL .

يتم طبع واظهار قيم هذه المتغيرات من خارج PL / SQL وذلك من خلال جملة PRINT .

او من داخل الـ BLOCK باستخدام DBMS_OUTPUT.PUT_LINE(:G_SAL);

يمكن ايضا استخدام Define لعمل متغيرمن خارج البلوك و لكن فى هذا النوع يأخذ قيمه ثابته عند تعريفه لا يمكن تغيرها ويستخدم هذا النوع فى عمل متغيرات لها قيمه ثابتة.

مثال:

```
SET SERVEROUTPUT ON
DEFINE p_annual_sal = 60000
DECLARE
  v_sal NUMBER(9,2) := &p_annual_sal;
BEGIN
  v_sal := v_sal/12;
  DBMS_OUTPUT.PUT_LINE ('The monthly salary is ' ||
                        TO_CHAR(v_sal));
END;
/
```