بسم الله الرحمن الرحيم

إعداد : م/ إسماعيل شبير أحمد

محتويات الكتاب :

م	الموضوع
	الفصل الأول : مدخل إلى لغة PL/SQL
١	• **مقدمة**
٢	• **مميزات اللغة**
٣	• **ماذا أحتاج لتعلم اللغة**
٤	• **كتابة البرنامج الأول**
٥	• **طرق تنفيذ البرنامج**
٦	• **مكونات كتلة الترميز (أجزاء البرنامج)**
٧	• **الكلمة المفتاحية (GOTO) القفز الغير المشروط**
٨	• **التعليقات**
٩	• **تمارين على الفصل الأول**
	الفصل الثاني : المتغيرات والثوابت
١	• **تعريف المتغيرات**
٢	• **أهمية المتغيرات**
٣	• **استخدامات المتغيرات**
٤	• **شروط تسمية المتغيرات**
٥	• **أنواع المتغيرات في اللغة**
	• **كيفية تعريف المتغيرات في اللغة**
	✓ **البيانات الرقمية**
	✓ **البيانات النصية**
٦	✓ **البيانات الأخرى**
	✓ **القيود**
	✓ **الأداة %TYPE**
	✓ **الأداة %ROWTYPE**
٧	• **أمثلة على الفصل الثاني**
٨	• **تمارين على الفصل الثاني**

	الفصل الثالث : جمل التحكم :١- (الجمل الشرطية)
١-	• **مقدمة**
٢-	• **أنواع جمل التحكم**
	• **أولا:الجمل الشرطية**
	• **ماذا تعني الجمل الشرطية**
	• **أشكال الجمل الشرطية**
٣-	• **الشكل الأول : IF-THEN**
	• **الشكل الثاني : IF-THEN-ELSE**
	• **الشكل الثالث : IF-THEN-ELSIF**
	تم دمج النوع الثاني من جمل التحكم :جمل التكرار مع الفصل الرابع .
	الفصل الرابع : المؤشرات
	تم شرح نصف هذا الفصل وقيد الإنشاء . ولكن يوجد في الكتاب ما تم شرحه إلى الآن .

الفصل الأول

مدخل إلى لغة PL/SQL

إعداد : م/ إسماعيل شبير أحمد

المقدمة :

تعتبر اللغة PL/SQL هي لغة البرمجة في نظام أوراكل ، وهي أداة برمجية قوية ، وكلمة PL/SQL اختصاراً لـ (Procedure Language/Structure Query Language)، وتستخدم هذه اللغة لتجهيز نظام Oracle عن طريقة معالجة التسجيلات ، وتستخدم أيضاً في أدوات التطوير المنتجة من قبل شركة Oracle ، وهي تعتبر أساس برنامج بناء النماذج Oracle Forms و Oracle Reports . وتستخدم هذه اللغة أيضاً لتعريف نموذج ما ، والقيام ببعض الحسابات الخاصة في تقرير ما ، ومن أجل التسجيلات .

مزايا لغة PL/SQL :

١- **التكامل :** وهذه اللغة لها دور أساسي بين أجزاء وأدوات أوراكل حيث يكتب بها أكواد (Forms) ويتم بها برمجة أجزاء وأدوات الأوراكل .

٢- **تحسين الأداء :** حيث يمكن لـ PL/SQL تحسين أداء التطبيقات وذلك من خلال :

أ- تجميع جمل SQL معاً في بلوك واحد (كتلة واحدة) وإرسالهما إلى خادم (Data Base) لتنفيذها دفعة واحدة مما يؤدي إلى ارتفاع مستوى الأداء عامة .

ب- يمكن لـ PL/SQL العمل داخل أي جزء من أجزاء وأدوات Oracle وبذلك يضيف قوة المعالجة الإجرائية إلى هذه الأدوات (Oracle Forms)، (Oracle Reports)، ..... ، مما يؤدي إلى تحسين مستوى الأداء .

٣- **تطوير البرنامج** Modularized وذلك بـ :

أ- تجميع منطقي للبيانات داخل كتل (Blocks) البرنامج .

ب- الكتل المتداخلة (Nested Blocks) تتيح العديد من المزايا .

ت- إتاحة تقسيم المشاكل المعقدة إلى مجموعة أبسط من المشاكل يمكن حلّها ببساطة .

ث- الإستفادة من خبرات وأكواد سابقة بجمعها في شكل مكتبات (Libraries) يمكن الإستفادة منها بين أدوات Oracle المختلفة .

ج- يمكن تنفيذ كود PL/SQL من أي أداة من أدوات Oracle المختلفة .

ح- يمكن تعريف المتغيرات والمتحولات (Variables) التي تستقبل العديد من أنواع البيانات المختلفة مثل الأرقام والنصوص والصور والفيديو والبيانات المركبة .... الخ .

خ- وتحتوي أيضاً على المميزات الأخرى مثل أوامر التكرار والتحكم في سير البرنامج ومعالجة الأخطاء والإستثناءات وووووو....الخ .

ماذا أحتاج لتعلّم هذه اللغة :

لا أحتاج إلى شيء إضافي ما دام عندي برنامج SQL*Plus منزّل على جهازي من قبل ،فبرنامج SQL *Plus يفي بالغرض لتعلّم وكتابة هذه اللغة ، الذي أحتاج فقط إلى المعرفة والقدرة على كتابة أوامر SQL وإني اجتزت هذه المادة بشكل جيد .

كتابة البرنامج الأول بلغة PL/SQL :

يمكن كتابة برامج PL/SQL وتنفيذها من محث SQL ضمن برنامج SQL *Plus . ولكن، وبسبب درجة تعقيد وطول البرامج، لا تعتبر آلية مفضلة الإستخدام . إذ يفضل تطوير البرنامج في أي محرر نصوص مثل المفكرة Notepad أو باني الإجرائيات في نظام Oracle . وبعد ذلك يمكن استدعاء ملفات البرامج ضمن SQL *Plus .

هيا نكتب برنامجنا الأول بلغة PL/SQL ، فلنفتح برنامج SQL *Plus وكتابة البرنامج التالي ، وهذا البرنامج لطباعة رسالة (Hello) :

القائمة ١-١

```
SQL> set serveroutput on
SQL> begin
  2  dbms_output.put_line ('Hello');
  3  end;
  4  /
Hello

PL/SQL تم بنجاح إجراء
```

فالبرنامج السابق يتألف من كتلة ترميز بلغة PL/SQL تحتوي على تعليمة واحدة . تُظهر الرسالة (Hello) .

بعض الخصائص الهامة للبرنامج السابق :

- يبدأ بكلمة (Begin) وينتهي بكلمة (End) .
- أمر الطباعة في لغة PL/SQL هو (DBMS_OUTPUT.PUT_LINE) .
- تتكون كتل ترميز PL/SQL من تعليمات ، وكل تعليمة تنتهي بفاصلة منقوطة .
- يتم وضع الرمز (/) في نهاية كتلة ترميز PL/SQL لتنفيذ تعليمات كتلة الترميز .
- تعتبر الكلمة المفتاحية (END) هي الكلمة المفتاحية الوحيدة في كتلة ترميز PL/SQL التي تنتهي بفاصلة منقوطة .

ملحوظة مهمة :

لا حظ أن أول سطر في البرنامج السابق كلمة (SET SERVEROUTPUT ON) ،هي تعليمة SET التابعة لبرنامج SQL *Plus المسؤولة بإظهار الرسائل المرسلة من قبل الإجرائيات الموجودة في الحزمة البرمجية DBMS_OUTPUT ، يجب وضعها مرة واحدة فقط ضمن جلسة العمل ، فإذا لم تكن هناك استخدام للحزمة البرمجية DBMS_OUTPUT فلا داعي إلى استخدام هذه التعليمة .

أعتقد إلى هنا ما توصلنا إلى حاجة مرتبة ومفهومة .. فأنا لست مطالباً بأي حاجة إلى هنا إلاّ جملة SET_SERVEROUTPUT ON : يجب استخدامها مرة واحدة فقط ضمن جلسة العمل إن كان هناك استخدام جملة DBMS_OUTPUT.PUT_LINE المسؤولة لتنفيذ أمر ما .

طرق تنفيذ برنامج PL/SQL :

١- إذا كانت الكتلة البرمجية لترميز PL/SQL مكتوب داخل برنامج SQL *Plus فيمكن تنفيذها بوضع الرمز (/) بعد نهاية كتلة الترميز مباشرة .

٢- وإذا كانت الكتلة البرمجية لترميز PL/SQL في ملف خارجي فيمكن تنفيذها من محث SQL *Plus باستخدام الكلمة المفتاحية (START) أو الرمز (@). وهذا النوع من الكتل البرمجية تسمى (إجرائيات كتل مجهولة) . والشكل العام لاستخدام هذه الكلمة المفتاحية :

START مسار الملف \ اسم الملف. الامتداد

```
SQL> START C:\1.SQL
Hello

تم بنجاح إجراء PL/SQL

SQL>
```

وبعد الانتهاء من تنفيذ كتلة ترميز PL/SQL سيظهر نظام Oracle الرسالة التالية:

PL/SQL تم نجاح إجراء

أو

PL/SQL procedure successfully completed

والتي تخبرنا بأنه تم تنفيذ البرنامج بنجاح ، أما إذا حدث أي خطأ فإن نظام Oracle سيصدر رسالة خطأ .

من هنا نبدأ الجد----------علينا الصبر والمثابرة

مكونات كتلة الترميز(أجزاء البرنامج) :

لكتابة أي برنامج بلغة PL/SQL يجب علينا أن نعرف أن ترميز كتلة PL/SQL يتكون من أربعة مقاطع وهي بالترتيب :

١- **الترويسة :** وهو مقطع اختياري في كتلة الترميز . ويستخدم لتحديد نوع كتلة الترميز واسمها . وأنواع كتل الترميز هي : anonymous procedure أي إجرائيـــــة مجهـــولة الاسم ، و named procedure أي إجرائية لها اسم ، و function أي تابع . وتستخدم الترويسة مع النوعين الأخيرين فقط .

٢- **التصريح :** وهو أيضاً مقطع اختياري في كتلة الترميز. ويحتوي على أسماء الأغراض المحلية التي سيتم استخدامها في كتلة الترميز . وتتضمن المتغيرات وتعاريف المؤشرات والاستثناءات ، ويبدأ هذا المقطع بالكلمة الافتتاحية (DECLARE) .

٣- **التنفيـذ :** وهو المقطع الإجباري الوحيد في كتلة الترميز . ويحتوي كل التعليمات التي سيتم تنفيذها ، والتي تتألف من تعليمات DML ، إجرائيات (كتل ترميز PL/SQL) ، توابع (كتل ترميز PL/SQL تعيد قيمة ما)،وبرامج جزئية مسبقة البناء. ويبدأ هذا المقطع بالكلمة الافتتاحية (BEGIN) .

٤- **الاستثناءات :** وهو مقطع اختياري . ويستخدم لالتقاط ومعالجة أي خطأ يحدث أثناء التعليمات الموجودة في المقطع التنفيذي . ويبدأ هذا المقطع بالكلمة المفتاحية (EXCEPTION) .

ملحوظة :

* يلي ذلك كله الكلمة المفتاحية (END) لإنهاء كتلة الترميز ، وهي الكلمة المفتاحية الوحيدة التي تنتهي بفاصلة منقوطة .

* ولتنفيذ كتلة الترميز يتم وضع الرمز (/) slash بعد نهاية كتلة الترميز .

والشكل العام لكتلة الترميز :

DECLARE
(جزء التصريح والتعريف) هنا يتم وضع المتغيرات وتعاريف المؤشرات والإستثناءات

BEGIN
(جزء التنفيذ) هنا يتم وضع التعليمات التي سيتم تنفيذها

EXCEPTION
(جزء الاستثناءات) هنا يتم وضع الاستثناءات

END; **(نهاية الكتلة البرمجية)** هذه الكلمة يجب وضعها في نهاية كتلة الترميز لإنهاء الكتلة

/ **(تنفيذ الكتلة البرمجية)** وهذا الرمز يتم وضعه لتنفيذ كتلة الترميز

سوف نأخذ المثال التالي :

كتلة برمجية يسترجع اسم الموظف الذي يحمل الرقم (7782) من جدول الموظفين (EMP) ، ثم طباعته،وإذا كان هناك حدوث أي خطأ على التعليمات التنفيذية سيظهر رسالة (ERROR OCCURED) .

```
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
  2   X VARCHAR2(30);
  3  BEGIN
  4   SELECT ENAME
  5    INTO X
  6     FROM EMP
  7      WHERE EMPNO = 7782;
  8  DBMS_OUTPUT.PUT_LINE (X);
  9  EXCEPTION
 10   WHEN OTHERS THEN
 11  DBMS_OUTPUT.PUT_LINE ('ERROR OCCURED');
 12  END;
 13  /
CLARK

PL/SQL تم بنجاح إجراء

SQL>
```

تعريف متغير من النوع الحرفي طوله ٣٠ حرفاً

هنا جملة الاستعلام عن اسم الموظف الذي يحمل الرقم 7782 وطباعته

تعليمات الإستثناء

هذا اسم الموظف المسترجع من جملة الاستعلام

الكلمة المفتاحية (GOTO) القفز الغير المشروط :

تستخدم الكلمة المفتاحية (GOTO) لإجراء قفز غير مشروط والانتقال من مقطع من كتلة الترميز إلى مقطع آخر. ولاستخدام هذه الكلمة المفتاحية هناك قواعد وتعليمات يجب إتباعها وهي بالترتيب :

١- تعريف العناوين : وذلك بوضع إشارتي (<<) قبل اسم العنوان وإشارتي (<<) بعد اسم العنوان .

٢- يتم استخدام الكلمة المفتاحية (GOTO) ثم وضع اسم العنوان المراد القفز إلى هناك بعد (GOTO) .

ملحوظة :

س/ ما المقصود بتعريف العناوين الذي أشرنا إليه في الفقرة رقم واحد ؟

ج/ تعرف العناوين بأنها وسائل تستخدم لوضع علامات لمقاطع كتلة الترميز .

سنأخذ المثال التالي :

كتلة برمجية تستخدم الكلمة المفتاحية (GOTO) لقفز غير مشروط من مقطع إلى مقطع آخر ، تحتوي ثلاث تعليمات تنفيذية ، يتم تنفيذ التعليمة الوسطى ثم الأولى ثم التعليمة الأخيرة :

```
SQL> Begin <<b_lable>>
  2  goto middle;
  3
  4  <<top>>
  5  DBMS_OUTPUT.PUT_LINE ('Top Statement');
  6  goto bottom;
  7
  8  <<middle>>
  9  DBMS_OUTPUT.PUT_LINE ('Middle Statement');
 10  goto top;
 11
 12  <<bottom>>
 13  DBMS_OUTPUT.PUT_LINE ('Bottom Statement');
 14
 15  END;
 16  /
Middle Statement
Top Statement
Bottom Statement

PL/SQL تم بنجاح إجراء

SQL> |
```

الشرح :
عند بداية تنفيذ البرنامج يحصل الجملة التالية (goto middle) فيذهب إلى المقطع (middle) وينفذ التعليمات ثم يحصل الجملة التالية (goto top) فيذهب إلى المقطع (top) وينفذ التعليمات ثم يحصل الجملة التالية (goto bottom) فيذهب إلى المقطع (bottom) ثم يحصل كلمة (end) فينهي البرنامج .

ملحوظة :

يوصي العديد من الخبراء في هذا المجال بعدم استخدام التعليمة (GOTO) ، فهي تجعل فقدان السيطرة على التطبيقات أمراً سهلاً . وهي للاستخدام في بعض الحالات النادرة التي يمكن أن تجعل البرنامج أسهل وأبسط .

التعليقات :

كأي لغات البرمجة الأخرى يمكن إدخال التعليقات ضمن كتلة الترميز ، ويوضع المبرمج مثل هذه التعليقات لتسهيل أمره وقت مراجعته وتطويره للبرنامج بعد فترة زمنية ، ولا تؤثر هذه التعليقات في حجم البرنامج أبداً . ويوجد أداتان لكتابة هذه التعليقات :

١- لكتابة تعليق سطراً واحدا :ً يتم وضع إشارتي ناقص (--) في بداية السطر الذي نرغب بوضعه كتعليق .

٢- ولكتابة تعليق عدة أسطر : يتم وضع الرمز (*/) في بداية التعليق ، ووضع الرمز (/*) في نهاية التعليق .

تمارين على الفصل الأول

السؤال الأول :

ضع علامة صح () أما العبارة الصحيحة وعلامة () خطأ أمام العبارة الخاطئة :

١- لغة PL/SQL هي أساس البرمجة في نظام أوراكل . ( )

٢- من مزايا لغة PL/SQL التكامل فقط . ( )

٣- لا يمكن تنفيذ أكواد PL/SQL من أي أداة من أدوات أوراكل المختلفة . ( )

٤- يمكن كتابة أكواد PL/SQL في محرر نصوص ثم استدعاؤه من محث SQL *Plus . ( )

٥- الكلمة المفتاحية الوحيدة التي تنتهي بفاصلة منقوطة هي (End) . ( )

٦- ولتنفيذ كتلة الترميز يتم وضع الرمز (*) star بعد نهاية كتلة الترميز . ( )

٧- المقطع الإجباري الوحيد هو المقطع التنفيذي (Begin). ( )

٨- الكلمة (GOTO)تشير إلى القفز المشروط . ( )

السؤال الثاني :

أجب عما يأتي :

١- إذا كان ملف كتلة الترميز مكتوب في ملف خارجي . فكيف يتم استدعاؤه من محث SQL *Plus ؟

_____

_____

٢- ما هو الرمز المستخدم في كتابة تعليق لسطر واحد فقط ؟ وما هو الرمز المستخدم في كتابة عدة أسطر؟

_____

_____

٣- أكتب كتلة برمجية يتم فيه طباعة النص التالي (Welcome To PL/SQL) ؟

الفصل الثاني

المتغيرات والثوابت

المتغيرات والثوابت :

تعريف المتغيرات :

إن المتغير هو موقع تخزين - من أجل حفظ قيمة - يمكن أن يتم إسناد قيمة إليه ، وسمي متغبر لأنه يمكن أن تتغير قيمته أثناء تنفيذ البرنامج .

أهمية المتغيرات :

١- حجز مكان في الذاكرة للمتغير .

٢- إعلام المترجم بنوع البيانات التي يمكن أن تخزن في المتغير .

فوائد استخدام المتغيرات :

١- تخزين مؤقت للبيانات .

٢- التعامل مع قيم مخزنة .

٣- إعادة استخدام البيانات نتيجة تغيرات داخل وأثناء البرنامج .

٤- الإختصار وسهولة التعديل والصيانة وذلك من استخدام (%type) و (%rowtype) سيأتي شرحها لاحقاً ، و من خلالهما يمكن تعريف متغير حسب نوع عمود أو صف في قاعدة البيانات مما يتيح قدر كبير من المرونة دون التقيد بنوع معين من البيانات .

شروط تسمية المتغيرات :

١- اسم المتغير يجب أن يبدأ بحرف .

٢- يجب أن لا يكون اسماً لجدول أو عمود سيستخدم في الكتلة البرمجية .

٣- أن لا يزيد طول الاسم عن ٣٠ حرفاً .

٤- يمكن أن يحتوي اسم المتغير على حروف وأعداد أو شرطة سفلية أو الحرف $ أو الرمز # .

٥- لا يحتوي اسم المتغير على رموز خاصة .

٦- لا يحتوي اسم المتغير على مسافات .

٧- لا يكون اسم المتغير ضمن الأسماء المحجوزة في اللغة مثل Declare, for, if ... الخ.

٨- لغة PL/SQL لا تفرق بين الحروف الكبيرة والصغير فمثلاً x لا يختلف عن X .

٩- تعريف متغير واحد في كل جملة .

١٠- يفضل أن يكون اسم المتغير ذو معنى .

أنواع البيانات للمتغيرات في لغة PL/SQL :

تنقسم المتغيرات إلى قسمين أساسين :

١- متغيرات PL/SQL :

وتحتوي على عدة أنواع منها :

أ- المفردة Scalar.

ب- المركبة(المعقدة) Composite .

ت- ذات الأحجام الكبيرة LOB(Large Object) .

٢- متغيرات ليست PL/SQL :

مثل متغيرات : host , Bind .

فما علينا في هذا المنهج إلا أنواع البيانات التي تحتوي على قيم مفردة .

المتغيرات المفردة :

كيفية تعريف المتغيرات في لغة PL/SQL :

يتم تعريف المتغيرات في لغة PL/SQL في جزء التصريح والشكل العام للتعريف هو :

Variable_name [CONSTANT] datatype (precision) ;

اسم المتغير [قيـــد] نوع البيانات (الدقة أو الطول);

ملحوظة مهمة : ١- يجب أن ينتهي التعريف بفاصلة منقوطة . ٢- اسم المتغير ونوعه إجباريان عند تعريف أي متغير .

مثل ما قلنا أن من أنواع المتغيرات في هذه اللغة (المفردة) فما معنى المفردة ؟

المفردة المقصود منها هنا : أن لا يحتوي قيمة المتغير سوى قيمة واحدة فقط .

وهذا الجدول يوضح بالتفصيل أنواع البيانات التي تحتوي على قيم مفردة :

النوع	الوصف
VARCHAR2(size)	البيانات الرمزية متغيرة الطول ويمثل size أكبر عدد من الرموز التي يمكن تخزينها في المتغير.ويجب تحديد الطول عند عملية التعريف . أكبر حجم هو ٣٢,٦٧٦ Byte
CHAR[(SIZE)]	البيانات الرمزية ثابتة الطول ويمثل size أكبر عدد من الرموز التي يمكن تخزينها في المتغير .وإذا لم يتم تحديد الطول تكون القيمة الافتراضية له ١ أكبر حجم هو ٣٢,٦٧٦ Byte
NUMBER (precision,scale)	البيانات الصحيحة والكسرية ويمثل Precision الحجم الكلي للمتغير و scale يمثل عدد المنازل العشرية
DATE	ويمثل نوع البيانات التي تكون على شكل تاريخ (وقت و تاريخ) والقيم التي يمكن أن يحتويها ما بين ٤٧١٢ قبل الميلاد و ٩٩٩٩ بعد الميلاد .
LONG	البيانات الرمزية متغيرة الطول ويمثل size أكبر عدد من الرموز التي يمكن تخزينها في المتغير.ويجب تحديد الطول عند عملية التعريف . أكبر حجم هو ٣٢,٦٧٠ Byte وأكبر حجم للعمود في الجدول من نوع LONG هو ٢,١٤٧,٤٨٣,٦٤٧Byte
LONG RAW	البيانات الممثلة ثنائيا (Binary) مثل الصور .
BOOLEAN	البيانات المنطقية مثل TRUE,FLASE
BINARY_INTEGER	أعداد صحيحة بين ٢,١٤٧,٤٨٣,٦٤٧ -و٢,١٤٧,٤٨٣,٦٤٧
PLS_INTEGR	أعداد صحيحة بين٢,١٤٧,٤٨٣,٦٤٧ -و٢,١٤٧,٤٨٣,٦٤٧ ولكن بحجم أقل من NUMBER و BINARY_INTEGER

سوف نأخذ بعض أنواع البيانات السابقة وكيفية التعريف بالتفصيل .

تعريف المتغيرات الرقمية :

يمكن أن تتضمن البيانات الرقمية وسيطين اثنين : الدقة (الطول الكامل للقيمة) و القيمة العشرية (عدد الأرقام الممكن وضعها إلى يسار أو يمين الفاصلة العشرية .

البيانات الرقمية تنقسم إلى قسمين :

١- البيانات الرقمية الصحيحة ويمكن تعريفه بالشكل التالي :

Salary integer(3);

٢- البيانات الرقمية ذات الفاصلة العشرية ويمكن تعريفه بالشكل التالي:

age number(3);

فهذا المتغير قيمته عددا دون تحديد القيمة العشرية ويمكن أن تحتوي هذه القيمة ثلاث أرقام عشرية وثلاث أرقام عشرية إلى اليسار ، بحيث لا يتجاوز الطول الأعظمي أكثر من ثلاثة مواضع .

summary number(3,2);

فهذا المتغير طوله ٣ منها عددان إلى اليمين بعد الفاصلة العشرية

ملحوظة :

* يتم إعطاء قيمة افتراضية للدقة مقدارها ٣٨ .

تعريف المتغيرات النصية :

يمكن تعريف البيانات النصية بشكل عادي ، ويتكون البيانات النصية من نمطين :

١- CHAR ٢- VARCHAR2 . والشكل العام لتعريف البيانات النصية :

Job char(3);

Name varchar2(30);

X char();

ملحوظة :

١- تحتوي التعاريف من النوع char فراغات في المواضع غير المشغولة .

٢- لا يمكن اعتبار الفراغات والقيم الفارغة نفس الشيء وحتى لا يمكن إجراء أي مقارنة بينهما .

٣- يعتبر الطول الافتراضي عندما لا يتم تحديد الطول لنمط البيانات char هو ١ ، والطول الأعظمي هو ٣٢٧٦٧.

٤- يجب تحديد الطول عند تعريف متغير لنمط البيانات varchar2 ، والطول الأعظمي هو ٣٢٧٦٧ .

٥- تسمح اللغة بتعريف متغيرات من الأنماط char و varchar2 بأطوال أعظمية أكبر .

تعريف المتغيرات من أنواع أخرى :

١- Boolean : يستخدم لتسجيل حالة ما ، ويمكن أن يأخذ القيمة True أو False أو Null .

مثل: Yes Boolean ;

٢- Date : يستخدم لتسجيل قيم التاريخ .

مثل: s_day date ;

٣- Exception : يستخدم لتعريف استثناء مخصص أو مقبض للخطأ .

مثل: e_error exception;

تعاريف القيود :

يمكن وضع القيود على المتغيرات معرّفة في كتلة الترميز ، ويُعرف القيد بأنه شرط يتم وضعه على المتغير. ويوجد نوعان شائعان من القيود وهما :

١- CONSTANT : وهذا القيد يتأكد من أن القيمة لم تتغير بعد نسب قيمة أولية للمتغير . فإذا حاولت تعليمة ما تغيير القيمة ، سيحدث خطأ .

٢- NOT NULL : هذا القيد يتأكد من أن المتغير يحتوى على دائماً على قيمة ، فإذا حاولت تعليمة ما نسب قيمة فارغة إلى المتغير ، سيحدث خطأ .

نسب قيم للمتغيرات :

توجد طريقتان لنسب القيم للمتغيرات في لغة PL/SQL :

١- معامل النسب (:=) مثل :

```
Salary number := 15;
```

٢- الكلمة المفتاحية (INTO) **وتستخدم في** تعليمة (SELECT) و (FETCH) ،مثل :

```
SELECT  ename
 INTO  v_name
  FROM  emp
   WHERE empno = 7782 ;
```

ملحوظة :

إذا استخدم INTO في جملة SELECT فإنه يسترجع قيمة واحدة أو سجل واحد فقط ، وإذا استرجع أكثر من قيمة واحدة أو سجل واحد ، سيحدث خطأ .

تعريف متغيرات من نوع تسجيلة PL/SQL ومتغيرات المجموعات :

ميزة أضيفت في لغة PL/SQL وهي تعريف متغيرات من نوع تسجيلة ومتغيرات المجموعات بمعنى : يمكن تعريف متغير على أساس تعريف عمود أو صف بأكمله في جدول بقاعدة البيانات أو متغير سبق تعريفه ، فالمتغير الجديد يأخذ نفس نوع بيانات عمود جدول في قاعدة البيانات ، وبذلك تتيح قدر كبير من المرونة في تعريف المتغيرات ، وأيضاً يوفر الكثير من الوقت للمبرمج ، وأيضاً اختصار البرنامج .

الفوائد الهامة من هذه الميزة :

١- يستطيع المطوّر أن يعرف بشكل آلي متغير بنفس مواصفات بيانات عمود جدول أو متغير من نوع مؤشر ، وبدون معرفة المواصفات لبيانات العمود أو المؤشر .

٢- يستطيع المطوّر إعداد المتغيرات لمؤشر ما أو تسجيلة جدول بتعليمة واحدة . وستمتلك المتغيرات نفس المواصفات للجدول أو متغيرات المؤشر .

*** الخاصية أو الأداة (%TYPE) :**

تعتبر الأداة (%TYPE) أول أداة لتعريف متغير مجموعة ، والتي تسمح بتعريف متغير بنفس مواصفات عمود جدول بقاعدة البيانات ، والشكل العام للتعريف بهذه الأداة :

VARIABLE_NAME TABLE_CURSOR_NAME.COLUMN_NAME%TYPE ;

اسم المتغير اسم الجدول أو المؤشر اسم العمود%TYPE ;

وفي ما يلي نأخذ مثالاً على ذلك :

كتلة برمجية لعرض اسم الموظف وراتبه من جدول الموظفين (EMP) ، بحيث يكون رقم الموظف يساوي 7782 .

```
SQL> Declare
  2   name emp.ename%type;
  3   salary emp.sal%type;
  4  Begin
  5   select ename,sal
  6    into name,salary
  7     from emp
  8      where empno = 7782;
  9  DBMS_OUTPUT.PUT_LINE ('Name:' || name || 'Sal Is:' || salary);
 10  End;
 11  /
Name:CLARKSal Is:2450

تم بنجاح إجراء PL/SQL

SQL> |
```

دعنا الآن نشرح المثال السابق بالتفصيل :

السطر الأول : كلمة مفتاحية (Declare) تحدد جزء التصريح والتعريف .

السطر الثاني : تعريف متغير واسمه (name)ونوع بياناته نفس نوع بيانات عمود(ename)من جدول(emp).

السطر الثالث : تعريف متغير واسمه (salary)ونوع بياناته نفس نوع بيانات عمود(sal)من جدول(emp).

السطر الرابع : كلمة مفتاحية (Begin) تحدد جزء التنفيذ .

السطر الخامس : جملة استعلام (select) يسترجع الاسم والراتب (ename) و(sal) .

السطر السادس : كلمة مفتاحية (into) لإسناد قيم للمتغيرات التي تم تعريفها في جزء التصريح (ename) و (salary) .

السطر السابع : أشرنا إلى استرجاع السجل يكون من جدول الموظفين (emp) .

السطر الثامن : جملة الشرط لاسترجاع السجل بحيث يكون رقم الموظف (empno) يساوي (7782) .

السطر التاسع : جملة الطباعة بحيث يطبع اسم الموظف وراتبه المسترجع من جملة الإستعلام السابقة .

السطر العاشر : كلمة مفتاحية (End) لإنهاء البرنامج أو كتلة الترميز وتنتهي بفاصلة منقوطة .

السطر الحادي عشر : يدل الرمز (/) على تنفيذ البرنامج أو كتلة الترميز السابقة .

ملحوظة :

١. السطر التاسع : نرى الرمز (||) ، ويستخدم لطباعة قيمة أكثر من متغير أو جملة في نفس جملة الطباعة .
٢. لطباعة نص يجب وضعه بين علامتي تنصيص مفردة .

*** الخاصية أو الأداة (%ROWTYPE) :**

الأداة () ، وهي ثاني أداة لتعريف متغير مجموعة ، وتستخدم لتأسيس مصفوفة من المتغيرات مبنية على الأعمدة الموجودة في مؤشر ما أو جدول ما . والشكل العام للتعريف بهذه الأداة :

ARRAY_NAME TABLE/CURSOR_NAME%ROWTYPE ;

اسم المتغير اسم الجدول أو المؤشر%ROWTYPE ;

وفي ما يلي نأخذ مثالاً على ذلك :

كتلة برمجية لعرض اسم الموظف وراتبه ووظيفته من جدول الموظفين (EMP) ، بحيث يكون رقم الموظف يساوي 7782 .(باستخدام الأداة %ROWTYPE) :

```
SQL> Declare
  2   x emp%rowtype;
  3  Begin
  4   select *
  5    into x
  6     from emp
  7      where empno = 7782;
  8  DBMS_OUTPUT.PUT_LINE ('Name: ' || x.ename || ' Sal Is:' || x.sal || ' Job Is:' || x.job);
  9  End;
 10  /
Name: CLARK Sal Is:2450 Job Is:MANAGER

تم بنجاح إجراء PL/SQL

SQL>
```

شرح البرنامج السابق :

هو نفس البرنامج الذي كتبنا في الأداة (%TYPE) ولكن يختلف هنا في بعض الفقرات عن البرنامج السابق من حيث الكتابة :

١- تم تعريف متغير (x) تأخذ نفس نوع أعمدة جدول الموظفين (emp) .

٢- جملة الطباعة في السطر الثامن يختلف عن السابق فعند طباعة قيمة متغير يجب وضع اسم المتغير ثم نقطة ثم اسم العمود في الجدول كالشكل التالي مثلاً :

DBMS_OUTPUT.PUT_LINE ('Name: ' || x.ename);

ملحوظة :

كل ما سبق تسمى بالمؤشرات الضمنية ، وتسترجع سجلاً واحداً فقط ، وإذا استرجع أكثر من سجل سيحدث خطأ .

أمثلة الفصل الثاني :

١- كتلة برمجية تطبع نص وقيمة المتغير (x) = 1500 : بهذا الشكل (x= 1500)

```
SQL> Declare
  2   x number(4) := 1500;
  3  Begin
  4  DBMS_OUTPUT.PUT_LINE ('x= ' || x);
  5  End;
  6  /
x= 1500

تم بنجاح إجراء PL/SQL

SQL>
```

٢- كتلة برمجية تسترجع اسم الموظف وراتبه واسم القسم الذي يشتغل فيه وطباعته ، بحيث يكون رقم الموظف يساوي (7782) : باستخدام الأداة (%TYPE) :

```
SQL> Declare
  2   name emp.ename%type;
  3   salary emp.sal%type;
  4   de_name dept.dname%type;
  5  Begin
  6  select ename,sal,dname
  7     into name,salary,de_name
  8      from emp,dept
  9       where emp.empno = 7782
 10        and emp.deptno = dept.deptno;
 11   DBMS_OUTPUT.PUT_LINE ('Name=' || name || ' Sal Is:' || salary || ' dname' || de_name);
 12   End;
 13  /
Name=CLARK Sal Is:2450 dnameACCOUNTING

تم بنجاح إجراء PL/SQL

SQL> |
```

٣- كتلة برمجية يتم فيه طباعة اسم الموظف ووظيفته الذي يحمل الرقم (7782) . ولكن باستخدام الأداة (%ROWTYPE) :

```
SQL> Declare
  2   x emp%rowtype;
  3  Begin
  4   select *
  5    into x
  6     from emp
  7      where emp.empno = 7782 ;
  8  DBMS_OUTPUT.PUT_LINE ('Name=' || x.ename || ' Job Is:' || x.job);
  9  End;
 10  /
Name=CLARK Job Is:MANAGER

تم بنجاح إجراء PL/SQL

SQL> |
```

تمارين الفصل الثاني :

السؤال الأول :

أجب عما يأتي :

١- ماذا تعني كلمة المتغير ؟

٢- ما هي أهمية المتغيرات ؟

أ -

ب-

٣- أذكر شروط تسمية المتغيرات ؟

أ -

ب -

ج -

د -

هـ -

و-

ز-

ح-

ط-

ي-

٤ - ما هي أنواع المتغيرات في لغة PL/SQL ؟

أ-

ب -

السؤال الثاني :

املأ الفراغات التالية :

١. من أنماط البيانات النصية char و ..... .

٢. يمكن تعريف المتغيرات في جزء ..... .

٣. تستخدم الأداة (%ROWTYPE) لـــ ..... .

٤. القيمة الافتراضية من نمط البيانات (CHAR) عندما لا يتم تحديد الطول هو ..... .

٥. ..... الطول الكامل للقيمة ، و ..... عدد الأرقام الممكن وضعها إلى يمين أو يسار الفاصلة العشرية .

٦. ..... يستخدم للبيانات البولية أو المنطقية .

٧. DATE يستخدم لـــ ..... .

السؤال الثالث :

١- اكتب كتلة برمجية لاستعراض تاريخ التعيين للموظف الذي اسمه (KING) ؟

٢- اكتب كتلة برمجية لحساب عدد الموظفين في جدول الموظفين (EMP) وطباعته ؟

٣- اكتب كتلة برمجية تقوم فيه تعريف متغيرين من النوع الرقمي وتنسب قيمة للأول (50) والثاني (30) ، ثم تقوم بجمع العددين وطباعة الناتج ؟

الفصل الثالث

جمل التحكم

إعداد : م/ إسماعيل شبير أحمد

جمل التحكم : ١- الجمل الشرطية

مقدمة :

نادراً ما نجد برنامجاً يسير بالتسلسل من بدايته حتى النهاية ، حتى وإن كان البرنامج صغيراً ، سنجد في البرنامج يتم فيه تكرار بعض الأجزاء أو تخطي بعض الأجزاء من البرنامج أو غير ذلك .... وسف نأخذ في هذا الفصل جمل وعبارات التحكم التي في لغة PL/SQL :

تنقسم جمل التحكم إلى قسمين رئيسين هما :

١- جمل الشرطية مثل (IF,ELSIF,ELSE) .

٢- جمل التكرار مثل (LOOP,FOR) .

أولاً : الجمل الشرطية :

ماذا تعني الجمل الشرطية ؟

أن تقوم بتنفيذ جزء معين من البرنامج إذا تحقق شرط ما ، فإذا لم تتحقق الشرط تتخطى هذا الجزء .

وما هي هذه الجمل الشرطية ؟

١- IF

٢- ELSIF

٣- ELSE

والشكل العام لهذه الجمل بالترتيب :

```
IF الشرط THEN
   الجمل التي يتم تنفيذها
ELSIF الشرط THEN
   الجمل التي يتم تنفيذها
ELSE
   الجمل التي يتم تنفيذها
END IF;
```

وسوف نأخذ الآن الأشكال السابقة بالتفضيل ...

الشكل الأول : IF :

هذا الشكل الأول من أشكال الجمل الشرطية ويستخدم لتنفيذ جمل أو جزء من البرنـامج إذا تحقق الشرط التي تلي كلمة(IF) .ويسمى هذا الشكل (الجمل الشرطية البسيطة) .

والشكل العام لهذا الشكل :

```
IF  الشرط  THEN
   الجمل التي يتم تنفيذها
END IF ;
```

دعنا الآن نأخذ مثالاً على ذلك :

المثال الأول :

كتلة برمجية تستعرض اسم الموظف وراتبه الذي يحمل الرقم (7782) ، بحيث تظهر الرسـالة (IS Greater Than 2000) إذا كان راتب الموظف أكبر من ٢٠٠٠ .

```
SQL> DECLARE
  2   X EMP%ROWTYPE;
  3  BEGIN
  4   SELECT *
  5    INTO X
  6     FROM EMP
  7      WHERE EMPNO = 7782;
  8  DBMS_OUTPUT.PUT_LINE ('ENAME: ' || X.ENAME);
  9  DBMS_OUTPUT.PUT_LINE ('SALARY: ' || X.SAL);
 10  IF X.SAL > 2000 THEN
 11  DBMS_OUTPUT.PUT_LINE ('IS Greater Than 2000');
 12  END IF;
 13  END;
 14  /
ENAME: CLARK
SALARY: 2450
IS Greater Than 2000

PL/SQL تم بنجاح إجراء

SQL>
```

هنا وضعنا الشرط

كان راتب الموظف أكبر من ٢٠٠٠ ، ولأن الشرط تحقق في الجملة الشرطية التي وضعناها ، نفذ الجملة التي تلي (IF) . وهي هذه

ملحوظة :

١- يجب أن تبدأ الجملة الشرطية بكلمة (IF) وتنتهي بكلمة (END IF) متبوعة بفاصلة منقوطة .

٢- تلي الشرط كلمة (THEN) في جملة (IF) ولا تتبع بفاصلة منقوطة .

٣- يتم تنفيذ الجمل التي تلي جملة (IF) إذا تحقق الشرط ، وإلا يتخطى هذا الجزء . ويكمّل باقي أجزاء البرنامج .

دعنا نتأمل في المثال التالي :

```
DECLARE
 x emp%Rowtype;
BEGIN
 select *
   into x
     from emp
       where empno = 7782;
DBMS_OUTPUT.PUT_LINE ('ENAME: ' || x.ename);
DBMS_OUTPUT.PUT_LINE ('SALARY: ' || x.sal);
IF X.SAL > 3000 THEN
  DBMS_OUTPUT.PUT_LINE ('IS Greater Than 3000');
  DBMS_OUTPUT.PUT_LINE ('Good');
END IF;
DBMS_OUTPUT.PUT_LINE ('SUMAA');
END;
/
```

ونتساءل ماهي مخرجات البرنامج ؟

والجواب هو :

ENAME: CLARK

SALARY: 2450

SUMAA

بعد ما تعرّفنا على مخرجات البرنامج يمكن أن نتساءلَ سؤالاً آخراً وهو : لماذا لم يتم طباعة الجملتين التاليتين (IS Greater Than 3000) ، (Good) . وتم طباعة الجملة التالية (SUMAA) .

والجواب هو : لم يتم طباعة الجملتين التاليتين (IS Greater Than 3000) ، (Good)لأنه لم يتحقق الشرط وتخطى هذا الجزء. وتم طباعة الجملة (SUMAA) وكما قلنا إذا لم يتحقق الشرط فإنه يتخطى جزء الجملة الشرطية ويكمّل في تنفيذ باقي أجزاء البرنامج إلى الأخير .

الشكل الثاني : IF-THEN-ELSE :

الشكل الثاني من أشكال الجمل الشرطية (IF-THEN-ELSE) ويستخدم في حال كان هناك اختيارين ليتم تنفيذ أحدهما ، مثلاً عندما نرغب في اختبار شرط معين بحيث إذا كان صحيحاً ، فإننا ننفذ الجمل التي تلي IF وإلا يتم تنفيذ الجمل التي تلي ELSE . والشكل العام لهذا الشكل :

```
IF الشرط THEN
    مجموعة الأوامر التي يتم تنفيذها
ELSE
    مجموعة الأوامر التي يتم تنفيذها
END IF;
```

والآن سنأخذ المثال التالي :

كتلة برمجية تستعرض اسم الموظف وراتبه الذي يحمل الرقم (7782) ، بحيث تظهر الرسالة (IS Greater Than 2000) إذا كان راتب الموظف أكبر من ٣٠٠٠ . وإلا تظهر الرسالة (IS Less Than 3000) .

```
SQL> DECLARE
  2   X EMP%ROWTYPE;
  3  BEGIN
  4   SELECT *
  5    INTO X
  6     FROM EMP
  7      WHERE EMPNO = 7782;
  8  DBMS_OUTPUT.PUT_LINE ('ENAME: ' || X.ENAME);
  9  DBMS_OUTPUT.PUT_LINE ('SALARY: ' || X.SAL);
 10   IF X.SAL > 3000 THEN
 11     DBMS_OUTPUT.PUT_LINE ('IS Greater Than 3000');
 12   ELSE
 13     DBMS_OUTPUT.PUT_LINE ('IS Less Than 3000');
 14  END IF;
 15  END;
 16  /
ENAME: CLARK
SALARY: 2450
IS Less Than 3000

PL/SQL تم بنجاح إجراء
```

الشرح :
هنا، لأنه لم يتحقق الشرط فإنه تخطى ما يلي IF ، ونفذ ما يلي ELSE .

الشكل الثالث والأخير: IF-THEN-ELSIF :

والشكل الثالث والأخير من أشكال الجمل الشرطية هو تركيب IF المتداخلة ، ويستخدم عندما نرغب في إجراء عدة اختيارات متتالية ، والشكل العام لهذا الشكل :

```
IF  الشرط  THEN
    مجموعة الأوامر التي يتم تنفيذها
ELSIF  الشرط  THEN
    مجموعة الأوامر التي يتم تنفيذها
ELSE
    مجموعة الأوامر التي يتم تنفيذها
END IF;
```

وهذا المثال التالي يوضح لنا أكثر تطبيقياً :

كتلة برمجية تستعرض اسم الموظف وراتبه الذي يحمل الرقم (7782) ، بحيث تظهر الرسالة (IS Greater Than 2450) إذا كان راتب الموظف أكبر من ٢٤٥٠ ، وتظهر الرسالة (IS Equal To 2450) في حال كان راتب الموظف يساوي (2450) ، وإلا تظهر الرسالة (IS Less Than 2450)

```
SQL> DECLARE
  2   X EMP%ROWTYPE;
  3  BEGIN
  4   SELECT *
  5    INTO X
  6     FROM EMP
  7      WHERE EMPNO = 7782;
  8  DBMS_OUTPUT.PUT_LINE ('ENAME: ' || X.ENAME);
  9  DBMS_OUTPUT.PUT_LINE ('SALARY: ' || X.SAL);
 10   IF X.SAL > 2450 THEN
 11     DBMS_OUTPUT.PUT_LINE ('IS Greater Than 2450');
 12   ELSIF X.SAL = 2450 THEN
 13     DBMS_OUTPUT.PUT_LINE ('IS Equal To 2450');
 14   ELSE
 15     DBMS_OUTPUT.PUT_LINE ('IS Less Than 2450');
 16  END IF;
 17  END;
 18  /
ENAME: CLARK
SALARY: 2450
IS Equal To 2450

PL/SQL تم بنجاح إجراء
```

الشرح :
هنا، لم يتحقق الشرط الأول الذي في if فتخطى هذا الجزء ، ثم اختبر الشرط الثاني فتحقق الشرط ونفذ الجمل التي بعدها ، فبعد تنفيذ الجملة الشرطية التي تلي elsif خرج من الجملة الشرطية ولم يتابع في تكملة الجملة الشرطية else.

ملحوظة :

إذا تحقق شرط ما في الجملة الشرطية فإنه يتم تنفيذ الجمل التي تلي الشرط ويتخطى الشروط الباقية .

وبهكذا انتهينا من النوع الأول من جمل التحكم وهي الجمل الشرطية ... فإننا نؤخر النوع الثاني من جمل التحكم وهي جمل التكرار وندمج مع فصل المؤشرات الصريحة .. لأن ذلك يؤدي فهم الموضوع بشكل أكبر وبسهولة .

الفصل الرابع
المؤشرات الصريحة

إعداد : م/ إسماعيل شبير أحمد

المؤشرات :

ما هو المؤشر؟

المؤشر هو : أداة تستخدم لاسترجاع سجل/ سجلات من جدول / جدول اعتباري إلى الذاكرة . وتسمح المؤشرات بقراءة كل تسجيلة من التسجيلات من قبل كتلة الترميز ثم معالجتها تسجيلة تسجيلة .

أنواع المؤشرات في لغة PL/SQL :

١- مؤشرات ضمنية .

٢- مؤشرات صريحة .

والجدول التالي يوضح الفرق بين المؤشرات والضمنية والصريحة .

المؤشرات الضمنية	المؤشرات الصريحة
تستخدم لاسترجاع سجلاً واحداً فقط .	تستخدم لاسترجاع سجلاً واحداً أو أكثر.
يتم تعريفها في المقطع التنفيذي	يتم تعريفها في مقطع التصريح .
يتم استخدامها مع جملة : SELECT ....INTO	يتم استخدامها مع جملة : FETCH ....INTO
يجب استرجاع سجلاً واحدا .	يجب اتباع القواعد التالية بالترتيب لاستخدامها : ١- تعريف المؤشر . ٢- فتح المؤشر . ٣- استرجاع السجلات . ٤- إغلاق المؤشر .

مثال :
DECLARE
x EMP%ROWTYPE;
BEGIN
SELECT ename
INTO x
FROM emp
WHERE empno = 7782;
END;
	مثال :
	DECLARE
	CURSOR abc IS SELECT ename FROM emp;
	x EMP%ROWTYPE;
	BEGIN
	OPEN abc;
	FETCH INTO x ;
	CLOSE abc ;
	END;

كل ما سبق في الفصول السابقة كان استخدامنا في المؤشرات الضمنية وتسترجع سجلا واحداً فقط . وهنا نأتي لشرح المؤشرات الصريحة بالتفصيل .

المؤشرات الصريحة :

تستخدم لاسترجاع مجموعة السجلات من جدول إلى الذاكرة ثم قراءة كل سجل من السجلات من قبل ترميز ثم معالجتها سجلاً سجلاً .

التصريح عن المؤشر :

يتم تعريف والإعلان عن المؤشر في جزء التصريح من كتلة الترميز . والشكل العام للتعريف :

CURSOR اسم المؤشر IS جملة الاستعلام(SELECT) ;

ولاستخدام المؤشر في كتلة الترميز يجب اتباع القواعد التالية :

١- تعريف المؤشر والشكل العام لتعريف المؤشر :

CURSOR اسم المؤشر IS جملة الاستعلام(SELECT) ;

٢- فتح المؤشر والشكل العام لفتح المؤشر :

OPEN اسم المؤشر ;

٣- استرجاع البيانات (السجلات) والشكل العام للإسترجاع :

FETCH اسم المؤشر INTO المتغيرات التي تم تعريفها في جزء التصريح ;

٤- إغلاق المؤشر والشكل لإغلاق المؤشر :

CLOSE اسم المؤشر ;

المثال التالي يوضح عمل المؤشرات الصريحة :

كتلة برمجية تطبع بيانات ثلاث موظفين من الاسم والراتب من جدول الموظفين (EMP) .

```
SQL> Declare
  2   CURSOR abc IS SELECT ename,sal FROM emp;
  3   x emp.ename%type;
  4   y emp.sal%type;
  5  Begin
  6   OPEN abc;
  7   FETCH abc INTO x,y;
  8     DBMS_OUTPUT.PUT_LINE ('Name:' || x || ' Sal Is:' || y);
  9   FETCH abc INTO x,y;
 10     DBMS_OUTPUT.PUT_LINE ('Name:' || x || ' Sal Is:' || y);
 11   FETCH abc INTO x,y;
 12     DBMS_OUTPUT.PUT_LINE ('Name:' || x || ' Sal Is:' || y);
 13  CLOSE abc;
 14  End;
 15  /
Name:SMITH Sal Is:800
Name:ALLEN Sal Is:1600
Name:WARD Sal Is:1250

تم بنجاح إجراء PL/SQL
```

دعنا الآن نشرح المثال السابق :

السطر الثاني : عرّفنا مؤشر صريح يبدأ بكلمة (CURSOR) ثم افترضنا اسماً للمؤشر (abc) ثم كلمة (IS) وبعدها جملة الإستعلام .

السطر الثالث والرابع : عرّفنا متغيرين لتخزين القيم . الأول (x) لتخزين اسم الموظف ، والثاني (y) لتخزين الراتب .

السطر السادس : جملة فتح المؤشر وتبدأ بكلمة (OPEN) ثم اسم المؤشر .

السطر السابع والتاسع والحادي عشر : جملة استرجاع البيانات وتخزينها إلى المتغيرات . تبدأ بكلمة (FETCH) ثم اسم المؤشر ثم كلمة (INTO) وبعدها المتغيرات .

السطر الثامن والعاشر والثاني عشر : جملة طباعة البيانات المسترجعة من جملة الإستعلام .

السطر الثالث عشر : جملة إغلاق المؤشر وتبدأ بكلمة (CLOSE) ثم اسم المؤشر .

السطر الرابع عشر : كما نعرف لإنهاء كتلة الترميز (البرنامج) .

السطر الخامس عشر : لتنفيذ كتلة الترميز (البرنامج) .

يمكن أن نسأل سؤالاً مهمّاً هنا : ماذا نفعل لو يكون لدينا مثلاً خمسون سجلاً ونريد طباعة كل السجلات بالطريقة السابقة فهل نضع خمسون جملة (FETCH) وخمسون جملة طباعة .. يبدو أن الأمر كذا صعبة وقلقة جداً .. طيب إذا كان الأمر كذلك .. فماذا يفعل الذين لا يدرون كم سجلاً عندهم في قاعدة البيانات؟ وماذا يفعل الذين لديهم آلاف وملايين السجلات ويريدون طباعة كل السجلات ؟ .. طبعاً نقول لهم (لا تقلقوا فاللغة ما قصّرت معنا ووضعت لنا طرق عديدة نستفيد منها لاختصار برامجنا) ،هيّا فلنتعلّم كيف نختصر برامجنا في مثل هذه الحالات .. ولاختصار برامجنا يجب علينا أن نتعلّم جمل التكرار ... وفي ما يلي سنشرح جمل التكرار وكيفية استخدامها مع المؤشرات .

جمل التحكم : ٢- جمل التكرار

مقدمة :

في بعض الأحيان نحتاج إلى تكرار جزء من البرنامج عدد من المرات لأغراض معينة . إذن :

ما هو التكرار ؟

هو أن يقوم بتكرار جزء من البرنامج عدد معين من المرات أو حسب شرط معين . وهناك ثلاثة أنواع من البنى التكرارية في لغة PL/SQL تسمح لنا بأداء هذه المهمة .

البنى التكرارية :

١- LOOP ٢- WHILE .. LOOP ٣- FOR .. LOOP .

البنية الأولى : LOOP :

تسمى هذه البنية بالبنية البسيطة ، فهي تبدأ بـ(LOOP) وتنتهي بـ(END LOOP) متبوعة بفاصلة منقوطة . وتحتوي هذه البنية على جملة شرطية(IF-THEN-ELSE) أو (WHEN) والتي تحتوي الأمر (EXIT) لتوقيف عملية التكرار والشكل العام لها :

```
LOOP
  EXIT  WHEN  شرط التوقف  ;
  مجموعة الأوامر التي يتم تنفيذها وتكرارها ;
END  LOOP ;
```

أو الشكل :

```
LOOP
  مجموعة الأوامر التي يتم تنفيذها وتكرارها ;
  IF  شرط التوقف  THEN
   EXIT;
  END IF;
END  LOOP ;
```

مثال باستخدام LOOP مع WHEN :

```
SQL> DECLARE
  2    X NUMBER ;
  3  BEGIN
  4  X := 100;
  5  LOOP
  6  EXIT WHEN X > 1000 ;
  7  X := X + 10;
  8   END LOOP;
  9  DBMS_OUTPUT.PUT_LINE (X);
 10  end;
 11  /
1010

PL/SQL تم بنجاح إجراء
```

الشرح :
عرّفنا متغير في جزء التصريح ، ثم في جزء التنفيذ وضعنا قيمة ابتدائية للمتغير x := 100
ثم وضعنا كلمة LOOP أي بداية جملة التكرار ثم وضعنا شرطاً لتوقيف عملية التكرار أو(شرط الإستمرارية للتكرار) ووضعنا x := x+10 بمعنى يكرر في كل مرة يتحقق شرط الإستمرارية بمقدار ١٠ ، ثم أنهينا جملة التكرار بـEND LOOP ثم بعدها طبعنا قيمة أخيرة للمتغير ثم أنهينا البرنامج ، وتنفيذه .

مثال باستخدام LOOP مع IF :

نفس البرنامج السابق ولكن مع IF :

```
SQL> DECLARE
  2    X NUMBER ;
  3  BEGIN
  4    X := 100;
  5   LOOP
  6    X := X + 10;
  7     IF X > 1000 THEN
  8      EXIT;
  9     END IF;
 10   END LOOP;
 11  DBMS_OUTPUT.PUT_LINE (X);
 12  END;
 13  /
1010

PL/SQL تم بنجاح إجراء
```

الفرق بين البرنامجين السابقين :

لا فرق بين البرنامجين السابقين إلاّ في كيفية الاستخدام وطريقة كتابته ، فالأولى استخدمنا LOOP مع WHEN بينما الثاني استخدمنا LOOP مع IF .

ملحوظة :

إذا تحقق شرط التوقيف لعملية التكرار بالقيمة TRUE فإن البرنامج يخرج من عملية التكرار ويتابع في تنفيذ باقي أجزاء البرنامج حتى النهاية .

البنية التكرارية الثانية : WHILE ... LOOP :

هذه البنية تقريباً مشابهة للبنية السابقة البسيطة LOOP ، إلا أن الاختلاف في كيفية ومكان وضع شرط التوقيف ، فيتم وضعه هنا في هذه البنية في أعلى بنية الحلقة التكرارية LOOP عوضاً عن وضعه ضمن الحلقة التكرارية ، وتستخدم عندما نرغب فحص الشرط قبل الدخول إلى الحلقة التكرارية ، فإذا كان شرط التوقيف غير صحيح FALSE فإنه لم يتم الدخول إلى الحلقة التكرارية ، والشكل العام لها :

```
WHILE    شرط التوقيف    LOOP
  مجموعة الأوامر التي يتم تنفيذها ;
END LOOP;
```

مثال ذلك : (نفس البرنامج السابق ولكن باستخدام WHILE) :

```
SQL> DECLARE
  2    X NUMBER ;
  3  BEGIN
  4  X := 100;
  5  WHILE  X <= 1000  LOOP
  6  X := X + 10;
  7   END LOOP;
  8  DBMS_OUTPUT.PUT_LINE (X);
  9  end;
 10  /
1010

تم بنجاح إجراء PL/SQL
```

الشرح :
تم استخدام WHILE لعملية التكرار وتم وضع الشرط بعد هذه الكلمة مباشرة، وعملية سير البرنامج هو أن يفحص الشرط أوّلاً فإذا تحقق هذا الشرط بالقيمة TRUE فإنه يدخل إلى الحلقة التكرارية ، أما إذا لم يتحقق الشرط وكانت القيمة FALSE فإنه لم يتم الدخول إلى الحلقة التكرارية ويكمل في تنفيذ باقي أجزاء البرنامج حتى النهاية .

الفرق بين WHILE ... LOOP و LOOP البسيطة :

١- الفرق الأساسي هو أن البنية البسيطة LOOP يتم الدخول إلى الحلقة التكرارية ثم يتم فحص شرط التوقيف ، بينما WHILE...LOOP يتم فحص شرط التوقيف أوّلاً ، ثم إذا تحقق الشرط يتم الدخول إلى الحلقة التكرارية ، وإلاّ لن يتم الدخول إلى الحلقة التكرارية .

٢- باستخدام البنية البسيطة LOOP يجب استخدام الأمر EXIT ، بينما باستخدام WHILE...LOOP لا يتم استخدامه .

البنية التكرارية الثالثة والأخيرة : FOR...LOOP :

تعتبر هذه البنية الأكثر استخداماً مع المؤشرات وقوة عند الاستخدام ، وتسمى هذه البنية بالبنية الرقمية وهي الأبسط جداً ، وهي جملة دوران محدد لها بداية ونهاية ويتم تحديدهم أول جملة التكرار ، والشكل العام لها :

```
FOR متغير العداد IN [REVERSE] بداية العداد .. نهاية العداد LOOP
    مجموعة الأوامر التي يتم تنفيذها ;
END LOOP ;
```

ملحوظة :

١- لا يتم تعريف متغير العداد ولكن يتم التعرّف عليه من قبل جملة التكرار تلقائياً .

٢- يجب أن يكون بداية العداد أصغر من نهايته .

٣- تستخدم كلمة (REVERSE) بعد كلمة (IN) لتحويل العداد تنازلياً (من الأكبر إلى الأصغر) ، والافتراضي دائماً تصاعديا (من الأصغر إلى الأكبر) بدون استخدام (REVERSE) .

والمثال التالي: يقوم بطباعة العداد Y (تصاعدياً من الأصغر إلى الأكبر) وزيادة المتغير X بعشرة وذلك لعشر مرات :

```
SQL> DECLARE
  2   X NUMBER;
  3  BEGIN
  4   X := 100;
  5    FOR Y IN 1..10 LOOP
  6     X := X + 10;
  7     DBMS_OUTPUT.PUT_LINE ('Y=' ||Y ||'  X=' ||X);
  8    END LOOP;
  9  end;
 10  /
Y=1  X=110
Y=2  X=120
Y=3  X=130
Y=4  X=140
Y=5  X=150
Y=6  X=160
Y=7  X=170
Y=8  X=180
Y=9  X=190
Y=10  X=200

PL/SQL تم بنجاح إجراء
```

لاحظ قيمة العداد Y بدأت من ١ إلى ١٠ .

بحيث Y قيمة العداد و X قيمة المتغير في كل مرة يكرر الجملة .

و هنا نفس المثال السابق ولكن بترتيب (تنازلياً من الأكبر إلى الأصغر):

```
SQL> DECLARE
  2   X NUMBER;
  3  BEGIN
  4   X := 100;
  5    FOR Y IN REVERSE 1..10 LOOP
  6     X := X + 10;
  7     DBMS_OUTPUT.PUT_LINE ('Y=' ||Y ||'  X=' ||X);
  8    END LOOP;
  9  end;
 10  /
Y=10  X=110
Y=9  X=120
Y=8  X=130
Y=7  X=140
Y=6  X=150
Y=5  X=160
Y=4  X=170
Y=3  X=180
Y=2  X=190
Y=1  X=200

PL/SQL تم بنجاح إجراء
```

لاحظ قيمة العداد Y بدأت من ١٠ إلى ١

بزيادة كلمة (REVERSE) بعد كلمة (IN) .

بعدما تعرّفنا على كل أشكال البنى التكرارية سنشرح في ما يلي كيفية استخدامها مع المؤشرات الصريحة و خصائص المؤشرات .

خصائص المؤشرات:

هناك خصائص للمؤشرات يمكن أن نستفيد منها عند استخدامنا للمؤشرات الصريحة :

١- **%FOUND :** وتأخذ القيمة TRUE إذا أعادت آخر تعليمة FETCH سجلاً أو تسجيلة . وتأخذ القيمة FALSE إذا لم تعيد آخر تعليمة FETCH أي سجلاً أو تسجيلة .

٢- **%NOTFOUND :** وتأخذ القيمة TRUE إذا لم تعيد آخر تعليمة FETCH سجلاً أو تسجيلة . وتأخذ القيمة FALSE إذا أعادت آخر تعليمة FETCH سجلاً أو تسجيلة .

٣- **%ROWCOUNT :** وتستخدم هذه الخاصية لتعيد عدد أوامر FETCH التي تم تنفيذها على المؤشر.

٤- **%ISOPEN :** وتأخذ القيمة TRUE إذا كان المؤشر المحدد مفتوحاً . وتأخذ القيمة FALSE إذا كان المؤشر المحدد مغلقاً .

حيث تستخدم هذه الخصائص في شرط ضمن الإجرائية وتستخدم لتقييم حالة المؤشر . وتعتبر هذه الخصائص تعابيراً وليست تعليمات ، ويعني ذلك بأنها جزء من تعليمة وأنه لا يليها فاصلة منقوطة .

لماذا نستخدم هذه الخصائص ضمن الإجرائية ؟

ونستخدم هذه الخصائص مثلاً عندما نخشى أن تقوم إجرائية ما بفتح المؤشر وهو في الأصل مفتوحاً ، أو أن تقوم إجرائية ما بإغلاق المؤشر وهو في الأصل غير مفتوح . وباستخدام ذلك سيمنع من حدوث أخطاء .

دعنا نأخذ مثالاً بسيطاً لنتعرف كيفية استخدام هذه الخصائص ضمن الإجرائية :

```
SQL> declare
  2   cursor abc is select * from emp;
  3   x emp%rowtype;
  4  begin
  5   IF NOT abc%ISOPEN THEN
  6    open abc;
  7   END IF;
  8  fetch abc into x;
  9  dbms_output.put_line('Name:' || x.ename);
 10  dbms_output.put_line('Number OF Rows:' || abc%rowcount);
 11   IF abc%ISOPEN THEN
 12    close abc;
 13   END IF;
 14  end;
 15  /
Name:SMITH
Number OF Rows:1

تم بنجاح إجراء PL/SQL
```

الشرح :
في بداية جزء التنفيذ وضعنا شرطاً لفحص حالة المؤشر باستخدام if not abc%isopen then كأننا هنا قلنا إذا كان المؤشر ليس مفتوحا افتح المؤشر بجملة open abc; وأغلقنا الجملة الشرطية. ونلاحظ في السطر العاشر جملة طباعة وتحديداً abc%rowcount هنا طبعنا عدد السجلات المسترجعة من المؤشر abc . وبعد ذلك وضعنا شرطاً لفحص حالة المؤشر هل المؤشر مفتوحاً إذا كان كذلك أغلقه بجملة close abc; .

وإن شاء الله في نهاية هذا الفصل سوف نأخذ مثالاً شاملاً يحتوي فيه كل هذه الخصائص .

استخدام البنى التكرارية مع المؤشرات الصريحة :

أوّلاً : استخدام LOOP البسيطة مع المؤشرات الصريحة :

فكما قلنا سابقاً أن استخدام البنى التكرارية مع المؤشرات الصريحة تعطي قوة في البرنامج واختصاراً أكثر للبرنامج ، دعنا نتأمل في المثال التالي :

كتلة برمجية تستعرض جميع بيانات الموظفين من الاسم(ENAME) والراتب(SAL) من جدول الموظفين (EMP) .

```
SQL> Declare
  2   CURSOR abc IS SELECT * FROM emp;
  3   x emp%rowtype;
  4  Begin
  5   OPEN abc;
  6  LOOP
  7   EXIT WHEN abc%NOTFOUND;
  8   FETCH abc INTO x;
  9     DBMS_OUTPUT.PUT_LINE ('Name:' || x.ENAME || ' ' || ' Sal Is:' || x.SAL);
 10  END LOOP;
 11  CLOSE abc;
 12  End;
 13  /
Name:SMITH  Sal Is:800
Name:ALLEN  Sal Is:1600
Name:WARD  Sal Is:1250
Name:JONES  Sal Is:2975
Name:MARTIN  Sal Is:1250
Name:BLAKE  Sal Is:2850
Name:CLARK  Sal Is:2450
Name:SCOTT  Sal Is:3000
Name:KING  Sal Is:5000
Name:TURNER  Sal Is:1500
Name:ADAMS  Sal Is:1100
Name:JAMES  Sal Is:950
Name:FORD  Sal Is:3000
Name:MILLER  Sal Is:1300
Name:MILLER  Sal Is:1300

PL/SQL تم بنجاح إجراء
```

الشرح :
عرّفنا مؤشر في جزء التصريح ، ثم عرّفنا متغير ليأخذ حقول الجدول بأنواعه ، وفي جزء التنفيذ فتحنا المؤشر ثم استخدمنا جملة التكرار LOOP وأعطينا شرط التوقف EXIT WHEN ABC%NOTFOUND ; أي عندما لا يسترجع أي سجل آخر بعد استرجاع السجل الأخير الخروج من التكرار وكلمة (%NOTFOUND) هي خاصية من خواص المؤشرات بمعنى إذا لم يسترجع ، وبعد ذلك استخدمنا جملة FETCH لاسترجاع السجلات ثم جملة الطباعة ثم نهاية الجملة التكرارية ثم إغلاق المؤشر وبعدها نهاية البرنامج وتنفيذه.

ثانياً : استخدام WHILE .. LOOP مع المؤشرات الصريحة :

في حال استخدامنا للمؤشرات الصريحة مع البنى التكرارية WHILE .. LOOP يجب علينا أن نعرف بعض النقاط المهمة وهي :

١- استخدام جملة FETCH مرتين ، مرة قبل جملة التكرار WHILE ومرة أخرى قبل نهاية جملة التكرار .

نفس البرنامج السابق ولكن باستخدام WHILE .. LOOP :

```
SQL> Declare
  2   CURSOR abc IS SELECT * FROM emp;
  3   x emp%rowtype;
  4  Begin
  5   OPEN abc;
  6   FETCH abc INTO x;
  7  WHILE abc%FOUND LOOP
  8     DBMS_OUTPUT.PUT_LINE ('Name:' || x.ENAME || ' ' || ' Sal Is:' || x.SAL);
  9   FETCH abc INTO x;
 10  END LOOP;
 11  CLOSE abc;
 12  End;
 13  /
Name:SMITH  Sal Is:800
Name:ALLEN  Sal Is:1600
Name:WARD  Sal Is:1250
Name:JONES  Sal Is:2975
Name:MARTIN  Sal Is:1250
Name:BLAKE  Sal Is:2850
Name:CLARK  Sal Is:2450
Name:SCOTT  Sal Is:3000
Name:KING  Sal Is:5000
Name:TURNER  Sal Is:1500
Name:ADAMS  Sal Is:1100
Name:JAMES  Sal Is:950
Name:FORD  Sal Is:3000
Name:MILLER  Sal Is:1300

PL/SQL تم بنجاح إجراء
```

الشرح :

لاحظ في البرنامج أننا استخدمنا جملة FETCH مرتين ، مرة قبل جملة التكرار WHILE ، ومرة أخرى قبل نهاية جملة التكرار .

وطريقة عمل البرنامج كالتالي :

هنا في بداية تنفيذ البرنامج لأول مرة يسترجع سجلاً ويضعه في الذاكرة فهذا بجملة FETCH التي قبل جملة التكرار، ثم يبدأ بفحص الشرط إذا تحقق يدخل إلى جملة التكرار ويطبع السجل ثم يسترجع سجلاً آخر بجملة FETCH التي قبل نهاية الجملة التكرارية فيصادف جملة فينقل المؤشر لجملة فحص الشرط وهكذا في كل مرة إلى أن لم يتحقق الشرط فينهي البرنامج .

ملحوظة :

أن جملة FETCH التي قبل جملة التكرار تنفذ مرة واحدة في بداية تنفيذ البرنامج ، وأما جملة FETCH التي قبل نهاية جملة التكرار فتنفذ في كل مرة يتم التكرار .

ثالثاً : استخدام FOR .. LOOP مع المؤشرات الصريحة :

تعتبر استخدام FOR...LOOP مع المؤشرات الصريحة هي الأكثر استخداماً والأقوى والأبسط اختصاراً ، حيث يسمح للمطوّر تحديد موقع التعليمة التي تحدد نقطة توقف البنية التكرارية .

فكما تعلّمنا سابقاً أن استخدام المؤشرات الصريحة يجب أن نعمل كالتالي :

١- فتح المؤشر . -٢- بدأ التكرار . -٣- استرجاع البيانات . -٤- اختبار إن كان هناك سجل تم استرجاعه .

٥- المعالجة . -٦- إنهاء التكرار . -٧- إغلاق المؤشر .

بينما هنا باستخدام تركيبة FOR...LOOP سيتم ضمنياً فتح وإغلاق المؤشر واسترجاع البيانات ، والشكل العام لهذه التركيبة :

```
FOR   record_name   IN
  [ Cursor_name [(parameter [,parameter])]
  | (Query_definition)]
LOOP
   Statement;
END LOOP;
```

كما في المثال التالي :

```
SQL> declare
  2    cursor abc is select ename,sal
  3     from emp
  4      where comm is not null;
  5  begin
  6    for x in abc loop
  7      DBMS_OUTPUT.PUT_LINE ('Name:' || x.ENAME || ' ' || ' Sal Is:' || x.SAL);
  8    end loop;
  9  end;
 10  /
Name:ALLEN  Sal Is:1600
Name:WARD  Sal Is:1250
Name:MARTIN  Sal Is:1250
Name:TURNER  Sal Is:1500

PL/SQL تم بنجاح إجراء
```

الشرح :

لاحظ في هذا البرنامج أنه لا يوجد فتح وإغلاق المؤشر ولا يوجد استرجاع البيانات ، لأنه باستخدام المؤشرات الصريحة مع تركيبة for ... loop سيتم ذلك كله ضمنياً ضمن نطاق حلقة التكرار . ففي بداية جزء التصريح استخدمنا مؤشر صريح لاسترجاع الاسم والراتب من جدول الموظفين بحيث تكون العمولة للموظف غير فارغ ، ثم في بداية جزء التنفيذ استخدمنا جملة التكرار for ووضعنا متغير (لا يجب تعريفه في جزء التصريح ولكن جملة التكرار يتعرّف تلقائياً للمتغير) ثم كلمة in ، فيليها اسم المؤشر الذي عرّفنا في جزء التصريح ثم كلمة loop فبعد هذا السطر نجد جملة طباعة لطباعة السجلات ثم نهاية جملة التكرار end loop ،وهنا ينقل المؤشر إلى جملة for لإعادة التكرار فيتحقق الشرط إذا يوجد سجل تم استرجاعه في المؤشر يطبع وإلا ينهي البرنامج .

فيمكن اختصار البرنامج السابق ، في حال إذا لم نكن نريد تعريف المؤشر في جزء التصريح ، فنستطيع وضع جملة الاستعلام (select) داخل تركيبة for التكرارية كما في المثال التالي :

```
SQL> begin
  2    for x in (select ename,sal from emp where comm is not null )
  3    loop
  4      DBMS_OUTPUT.PUT_LINE ('Name:' || x.ENAME || ' ' || ' Sal Is:' || x.SAL);
  5    end loop;
  6  end;
  7  /
Name:ALLEN  Sal Is:1600
Name:WARD  Sal Is:1250
Name:MARTIN  Sal Is:1250
Name:TURNER  Sal Is:1500

PL/SQL تم بنجاح إجراء
```

الشرح :
لاحظ في هذا البرنامج لا يوجد جزء التصريح بل اكتفينا بجزء التنفيذ ، فوضعنا جملة الاستعلام (select) داخل تركيبة for التكرارية .
أليس هذا جميلاً ومختصراً ؟ ما أروع هذه اللغة ! قوة ... بساطة ... اختصار ... الخ .

قفل السجلات مع الخيار FOR UPDATE :

عندما يتم وضع علامة على سجل ما أنه قد أُحضر من أجل التعديل أو من أجل الحذف يقوم نظام Oracle بوضع قفل على السجل ، بمعنى أنه لا يمكن لمستخدم آخر تعديل أو حذف هذا السجل قبل أن يتم حفظ السجل بشكل دائم (وتتم إزالة القفل) . وتعبر هذه الخاصية من خصائص قاعدة البيانات الهامة للتحكم بعمليات الوصول المتزامن للسجلات . (وخاصة إذا كان البرنامج مستخدم على الشبكة) . فإذا رغبنا بأن لا يستطيع مستخدم آخر التعديل في السجل بينما يقوم مستخدم ثان بإجراء التعديل فعلياً على السجل . وسيقوم مدير قاعدة البيانات بجعل المستخدم الثاني ينتظر حتى ينتهي المستخدم الأول من إجراء تعديلاته وتحريره للسجل إما بتعليمة التثبيت Commit أو بتعليمة التراجع Rollback . فدعنا الآن نأخذ النقاط المهمة في الموضوع :

١- تستخدم جملة (FOR UPDATE) بعد جملة (SELECT) لقفل سجل معين أو سجلات معينة ومنع الآخرين من تغيير أو مسح السجلات المقفلة .

٢- عند قفل السجلات يسمح للآخرين عرض (استرجاع) السجلات ولكن لا يسمح بتغيير أو قفل السجلات .

٣- تركيبة الجملة فيتم وضعها في جملة SELECT حيث يضاف كلمة FOR UPDATE في نهاية جملة SELECT.

٤- لا يتم تحرير (إزالة القفل ،فتح) السجلات حتى يتم إنهاء العملية باستخدام تعليمة التثبيت COMMIT أو باستخدام تعليمة التراجع ROLLBACK .

مثال ذلك :

```
SQL> Declare
  2   CURSOR abc IS SELECT * FROM emp WHERE empno=7782 FOR UPDATE;
  3   x emp%rowtype;
  4  Begin
  5   OPEN abc;
  6   FETCH abc INTO x;
  7  WHILE abc%FOUND LOOP
  8      DBMS_OUTPUT.PUT_LINE ('Name:' || x.ENAME);
  9   FETCH abc INTO x;
 10  END LOOP;
 11  CLOSE abc;
 12  End;
 13  /
Name:CLARK

PL/SQL تم بنجاح إجراء
```

الشرح :
في هذا المثال لا مكننا رؤية تأثير البرنامج إلا إذا كان برنامج فعلي مستخدم على الشبكة ، فعموماً أننا استخدمنا جملة for update في نهاية جملة select لقفل السجلات المسترجعة من قبل المؤشر .

الخيار FOR UPDATE OF :

يمكن أن يتبع جملة FOR UPDATE كلمة أخرى بعدها هي OF وأسماء الأعمدة المطلوبة . وعندما يتم وضع هذه الكلمة (OF) ، سيقوم المؤشر بقفل السجلات فقط إذا احتوى جملة SELECT الموجودة في تعريف المؤشر اسم أحد الأعمدة المبينة بعد الكلمة (OF) .

مثال ذلك :

```
SQL> Declare
  2   CURSOR abc IS SELECT ename FROM emp WHERE empno=7782 FOR UPDATE OF sal;
  3   x emp.ename%type;
  4  Begin
  5   OPEN abc;
  6   FETCH abc INTO x;
  7  WHILE abc%FOUND LOOP
  8     DBMS_OUTPUT.PUT_LINE ('Name:' || x);
  9   FETCH abc INTO x;
 10  END LOOP;
 11  CLOSE abc;
 12  End;
 13  /
Name:CLARK

تم بنجاح إجراء PL/SQL
```

الشرح :
لن تقفل التعليمة for update السجلات لأن العمود sal الموجود بعد كلمة of غير موجود في عبارة select .

النقاط المهمة :

١- يلي كلمة OF أسماء الأعمدة المطلوبة .

٢- سيقوم المؤشر بقفل السجلات فقط إذا احتوى تعريف جملة SELECT نفس اسم العمود (الأعمدة) بعد كلمة OF .

الخيار WHERE CURRENT OF :

بينما يقوم نظام ORACLE بإحضار السجلات من مؤشرها ، يقوم دائماً بوضع علامة لتحديد السجل الحالي . وإذا رغبنا بتعديل هذا السجل ، يمكننا استخدام هذا الخيار WHERE CURRENT OF . ويتم وضع هذا الخيار في نهاية تعليمات DML مثلاً في تعليمة التعديل أو التحديث UPDATE أو تعليمة الحذف DELETE حيث نظام ORACLE يقوم بتعديل السجل الحالي في المؤشر . ويمتلك هذا الخيار فائدتين هامتين وهما :

١. **فائدة خاصة بالأداء :** حيث يعرف نظام ORACLE دائماً معرفة السطر للسجل الحالي . وعندما يتم تعديل السجل ،يستطيع نظام ORACLE الذهاب إلى السجل بدون الحاجة لتحديد السجل انطلاقاً من الجدول . أما إذا أزلنا هذا الخيار ، ستحتاج تعليمات UPDATE و DELETE عبارة WHERE لتحديد السجل المطلوب ، وسيطلب ذلك بعض عمليات الدخل والخرج .ويمكن أن يزيد الخيار WHERE CURRENT OF بشكل كبير أداء تعديل البيانات من قبل الإجرائية .

٢. **فائدة خاصة بالاختصار :** يلغي هذا الخيار الحاجة إلى إنشاء عبارة WHERE مع تعليمات DML . كما يلغي أيضاً الحاجة إلى متغيرات محلية ، حيث يتم إحضار القيم إلى المتغيرات وتضمينها في عبارة WHERE . وسيقلص هذا الخيار حجم الإجرائية من حيث التعليمات البرمجية أيضاً .

والشكل العام لهذا الخيار :

اسم الجدول ← table_name ، اسم العمود المراد تعديله ← column_name

```
UPDATE  table_name  SET   column_name = value
WHERE CURRENT OF  cursor_name;
```

← اسم المؤشر

مثال ذلك :

كتلة برمجية تقوم بتعديل راتب الموظف الذي رقم 7782 بنسبة ثلاثة بالمائة ، وكما تقوم بعرض بيانات الموظف من حيث الاسم والراتب بعد التعديل .

```
SQL> Declare
  2   CURSOR abc IS SELECT ename,sal FROM emp where empno= 7782 FOR UPDATE;
  3   x emp.ename%type;
  4   y emp.sal%type;
  5  Begin
  6   OPEN abc;
  7   FETCH abc INTO x,y;
  8  WHILE abc%FOUND LOOP
  9     DBMS_OUTPUT.PUT_LINE ('Name:' || x ||'  Sal is: '|| y);
 10    UPDATE emp SET sal = SAL * 1.03
 11    WHERE CURRENT OF abc;
 12   FETCH abc INTO x,y;
 13  END LOOP;
 14  CLOSE abc;
 15  DBMS_OUTPUT.PUT_LINE ('---After Update---');
 16  OPEN abc;
 17   FETCH abc INTO x,y;
 18  WHILE abc%FOUND LOOP
 19     DBMS_OUTPUT.PUT_LINE ('Name:' || x ||'  Sal is: '|| y);
 20   FETCH abc INTO x,y;
 21  END LOOP;
 22  CLOSE abc;
 23  End;
 24  /
Name:CLARK  Sal is: 2523.5
---After Update---
Name:CLARK  Sal is: 2599.21

PL/SQL تم بنجاح إجراء
```

جملة update لتعديل راتب الموظف بنسبة ثلاثة بالمائة ثم قمنا بوضع جملة where current of وبعدها اسم المؤشر الذي قمنا بتعريفه في جزء التصريح كمؤشر . حيث تختبر نظام Oracle تماماً أين توجد السجل الذي يجب تعديله؟.

وهنا استخدمنا المؤشر مرة أخرى لعرض بيانات الموظف بعد تعديل الراتب .

هنا المخرجات قبل وبعد التعديل

النقاط المهمة في استخدام هذا الخيار :

١. تستخدم في Update و Delete لتحديد أحدث سجل تم استرجاعه من الجدول لتعديله أو لحذفه.

٢. يجب تعريف المؤشر باستخدام FOR UPDATE .

وبهذا انتهينا ولله الحمد من هذا الفصل يبدو طويلاً نوعاً ما ، ولكن الفصل الأهم في تعليم هذه اللغة ، فما تبقّى لنا من هذا الفصل أي شيء آخر إلا نأخذ مثالاً شاملاً عما درسناه في هذا الفصل .

هيا إلى هذا المثال التالي ولنتأمّل فيه بدقة ويحتوي فيه أغلب التعليمات الموجودة في هذا الفصل :

```
DECLARE
   Cursor abc is select * from emp;
    x abc%rowtype;
BEGIN
  If not abc%isopen then
    Open abc;
  End if;
Loop
  Fetch abc into x;
  Exit when abc%notfound;
  Dbms_output.put_line('Name is:' || x.ename || '  sal is:' || x.sal);
End Loop;
z:= abc%rowcount;
Dbms_output.put_line( 'Number Of Rows:' || z );

If abc%isopen then
   Close abc;
End If;
End;
```

هنا جزء التصريح ، عرفنا مؤشر صريح ، وعرّفنا متغير من متغيرات مجموعة ويأخذ نوع الأعمدة الموجودة في المؤشر .

هنا فتح المؤشر بطريقة اختبار المؤشر أولا ثم فتح المؤشر.

أما هنا جملة التكرار واسترجاع البيانات وطباعتها .

هنا استرجاع عدد السجلات المسترجعة من المؤشر .

هنا إغلاق المؤشر بطريقة اختبار المؤشر أولا ثم إغلاق المؤشر.

يمكن استبدال المربع أعلاه بما يلي :

```
Fetch abc into x;
While abc%found Loop
   Dbms_output.put_line('Name is:' || x.ename || '  sal is:' || x.sal);
   Fetch abc into x;
End Loop;
```

ويمكن استبدال العبارة

```
Exit when abc%notfound;
```

بما يلي:

```
If abc%notfound
   Exit;
End if;
```

أمثلة على الفصل الثالث والرابع

كتلة برمجية تستعرض الاسم والراتب ورقم القسم الذي يشتغل فيه من جدول الموظفين بحيث يعملون في قسم رقم (١٠) .

```
SQL> declare
  2    cursor abc is select * from emp where deptno = 10;
  3    x abc%rowtype;
  4  begin
  5   if not abc%isopen then
  6     open abc;
  7   end if;
  8
  9  loop
 10   exit when abc%notfound;
 11   fetch abc into x;
 12   dbms_output.put_line('Name is:'||x.ename||'  Sal is:'||x.sal ||'  deptno='||x.deptno);
 13  end loop;
 14
 15   if abc%isopen then
 16     close abc;
 17   end if;
 18  end;
 19  /
Name is:CLARK  Sal is:2450  deptno=10
Name is:KING  Sal is:5000  deptno=10
Name is:MILLER  Sal is:1300  deptno=10
Name is:MILLER  Sal is:1300  deptno=10

PL/SQL تم بنجاح إجراء
```

تمارين الفصل الثالث والرابع

السؤال الأول :

أكمل ما يأتي :

١- تنقسم الجمل التحكم إلى ١- _____ ٢- _____ .

٢- تعني الجمل الشرطية أن تقوم بتنفيذ_____

_____ .

وأشكالها:

أ-

ب-

ت-

٣- تبدأ الجمل الشرطية بالكلمة_____ وتنتهي بالكلمة _____ .

٤- تعني الجمل التكرارية أن تقوم بتكرار_____

_____ .

وأشكالها :

أ-

ب-

ت-

٥- تستخدم المؤشرات الصريحة _____

_____ .

٦- من قواعد استخدام المؤشرات الصريحة:

أ-

ب-

ت-

ث-

٧- يتم التعريف والإعلان عن المؤشر في جزء _____ .

٨- عند استخدام المؤشر مع البنية التكرارية FOR LOOP سيتم فتح وإغلاق المؤشر واسترجاع البيانات _____ .

٩- تستخدم جملة FOR Update بعد جملة _____ وذلك لـ_____

_____ .

١٠- يلي كلمة OF _____ .

السؤال الثاني :

ضع علامة () أمام العبارة الصحيحة وعلامة () أمام العبارة الخاطئة :

١- يجب أن تبدأ الجملة الشرطية بكلمة (IF) وتنتهي بكلمة (END IF) لا تتبع بفاصلة منقوطة ( ).

٢- عند قفل السجلات يسمح للآخرين عرض (استرجاع) السجلات ويسمح بتغيير أو قفل السجلات ( ).

٣- %ROWCOUNT **:** وتستخدم لتعيد عدد أوامر FETCH التي تم تنفيذها على المؤشر ( ).

٤- FOR LOOP : وتسمى هذه البنية بالبنية البسيطة ( ).

٥- لا يتم تحرير (إزالة القفل ،فتح) السجلات حتى يتم إنهاء العملية باستخدام تعليمة التثبيت COMMIT أو باستخدام تعليمة التراجع ROLLBACK ( ).

السؤال الثالث :

اختر الجمل الصحيحة مما يأتي مع تصحيح الخطأ في الجمل الخاطئة :

1:

```
1- select * where emp;
2- select * fom emp
3- select * fom emp where empno = 10;
```

2:

```
1- cursor abc select ename from emp;
2- cursor abc is a select ename fom emp;
3- cursor abc is select * from emp;
```

3:

أي مما يلي تسترجع كل بيانات الموظفين من حيث الاسم والوظيفة :

1-

```
Declare
   Cursor abc is select * from emp ;
   X abc%rowtype ;
Begin
   Open abc ;
While abc%found Loop
   Fetch abc into X ;
   Dbms_output.put_line ( ' Name is:' || X.ename || '  Job is:' || X.job);
End Loop;
   Close abc;
End ;
```

2-

```
Declare
   Cursor abc is select * from emp ;
   X abc%rowtype ;
Begin
   Open abc ;
Loop
   Fetch abc into X ;
   Dbms_output.put_line ( ' Name is:' || X.ename || '  Job is:' || X.job);
End Loop;
   Close abc;
End ;
```

3-

```
Begin
For X IN ( select * from emp )  Loop
   Dbms_output.put_line ( ' Name is:' || X.ename || '  Job is:' || X.job);
End Loop;
End ;
```

السؤال الرابع :

أكتب الكتل البرمجية التالية :

١- كتلة برمجية تستعرض رقم الموظف الذي اسمه (BLAKE) بحيث تظهر الرسالة التالية (Sorry) في حال رقم الموظف المسرتجع من الاستعلام هو (1432) ، وتظهر الرسالة التالية (successfully) في حال كان رقم الموظف هو (7698) ، وإلا تظهر الرسالة التالية (Not Found) .

٢- كتلة برمجية تستعرض اسماء ورواتب الموظفين الذين أسماءهم يبدأ بحرف (M) .راعياً ما يلي:

١- فتح واسترجاع وإغلاق المؤشر مع فحص حالة المؤشر عند استخدامه .

٢- طباعة عدد السجلات المسترجعة من المؤشر .

٣- كتابة الكتلة البرمجية ثلاث مرات ، مرة باستخدام Loop ، ومرة باستخدام While Loop ، ومرة باستخدام For Loop .

الفصل الخامس

معالجة الاستثناءات واستخدام الإجرائيات الاسمية والتوابع