المحاضرة السابعة

التعامل مع الأخطاء

Handling Errors in PL / SQL

أحيانا قد يتسبب ملقم الأوراكل **Oracle Server** أو تطبيقات المستخدمين أو حتى الشبكة في ظهور أخطاء. وهذه الأخطاء **Errors** تعرف في الأوراكل بـ **Exception**. و كما سبق وعرفنا فإن الوحدة النمطية **PL / SQL Block** تتكون من ثلاثة أقسام من ضمنها قسم للتعامل مع الأخطاء الواردة في البرنامج وهو قسم **Exception** ، والأخطاء في الأوراكل يتم تقسيمها لنوعين هما :-

- أخطاء معرفة من قبل الأوراكل . **Pre-defined application Errors**
- أخطاء معرفة بواسطة المستخدم . **User-defined application Errors**

النوع الأول Pre-defined application Error :-

هي عبارة عن مجموعة من الأخطاء العامة معرفة من قبل الأوراكل حيث يعطي لكل خطاء اسماً يعرفه، ويتم مراجعة كل خطاء باسمه.

Syntax: -

```
DECLARE
        Some variables declaration;
BEGIN
        …
        …
        Exception
        When  exception_1  Then
           Statements;
        When  exception_2  Then
           Statements;
        …
        …
        When Others Then
           Statements;
END;
```

في الصيغة أعلاه نقوم باستبدال **exception_1** باسم الخطاء المعرف.

الأخطاء المعرفة من قبل الأوراكل :-

على الرغم من أن الأوراكل يعطي رقماً لكل خطاء إلا أن الأخطاء المعرفة من قبل الأوراكل **Predefined** يجب أن يتم التعامل معها بأسمائها. ويوفر لنا الأوراكل مجموعة من الأخطاء المعرفة، والجدول التالي يوضح لنا جزء من هذه الأخطاء الشائعة :-

الخطاء	تعريفه
no_data_found	عندما نريد سجل واحد و لا ترجع جملة **Select** أي سجل.
too_many_rows	عندما نريد سجل واحد وترجع جملة **Select** أكثر من سجل.
invalid_cursor	قامت الـ **Cursor** بأداء عملية غير صحيحة.
value_error	عند حدوث خطاء في تحويل نوع البيانات أو بيانات أطول من الحقل.
invalid_number	عند عدم القدرة على التحويل من رقم إلى نص.
zero_devide	عند القسمة على صفر.
dub_val_on_index	عند إدخال قيمة مكررة في حقل فريد.
cursor_already_open	عند فتح **Cursor** تم فتحها مسبقاً في البرنامج ومازالت مفتوحة.
not_logged_on	عندما يتم التعامل مع قاعدة البيانات دون تسجيل الدخول للأوراكل.
login_denied	لم يتم تسجيل الدخول بسبب خطاء في الاسم أو كلمة المرور.
program_error	عند حدوث خطاء داخلي في **PL / SQL**.
storage_error	إذا كانت الذاكرة لا تكفي أو بها عطب.
others	تستخدم للتعامل مع كل الأخطاء.

مثال: اختبار خطاء عدم وجود بيانات :-

```
SQL> DECLARE
  2  mem_empno   integer;
  3  BEGIN
  4  dbms_output.enable;
  5  select empno into mem_empno
  6  from emp
  7  where ename = 'AAAAAAAAA';
  8  exception
  9  when no_data_found then
 10   dbms_output.put_line('NO Employee');
 11  when too_many_rows then
 12  dbms_output.put_line('Too Many Employee Found');
 13  END;
 14  /
```

NO Employee

PL/SQL procedure successfully completed.

مثال: اختبار خطاء إرجاع بيانات لأكثر من سجل :-

```
SQL> DECLARE
  2  mem_empno   integer;
  3  BEGIN
  4  dbms_output.enable;
  5  select empno into mem_empno
  6  from emp
  7  where ename like 'S%';
  8  exception
  9  when no_data_found then
 10  dbms_output.put_line('NO Employee');
 11  when too_many_rows then
 12  dbms_output.put_line('Too Many Employee Found');
 13  END;
 14  /
```

Too Many Employee Found

PL/SQL procedure successfully completed.

النوع الثاني User-defined application Errors :-

النوع الثاني هي الأخطاء التي يقوم المبرمج بإدخالها في البرنامج حتى يقوم بالتحكم في البرنامج بالشكل الذي يريده حيث يمكننا مثلاً أن نعرف خطاء ينتج في البرنامج عندما تكون قيمة حقل معين غير مطابقة للمواصفات (مثل وجود موظف يأخذ عمولة أكبر من راتبه الشهري إذا افترضنا أن ذلك لا يجوز).

في هذه الحالة نقوم بتعريف متغير نوعه **Exception** ، مع ملاحظة أن الأوراكل لا يقوم بتنفيذ هذا الخطاء ما لم نطلب منه ذلك باستخدام الكلمة المحجوزة **Raise** .

مثال: إعطاء رسالة خطاء إذا وجد موظف يأخذ عمولة أعلى من راتبه :-

```
SQL> DECLARE
  2  illegal_commission  exception;
  3  emp_name emp.ename%type;
  4
  4  BEGIN
  5
  5  dbms_output.enable;
  6  for emp_rec in (select ename,sal,comm from emp) loop
  7     if emp_rec.comm > emp_rec.sal then
  8        emp_name := emp_rec.ename;
  9        raise illegal_commission;
 10     end if;
 11  end loop;
 12
 12  Exception
 13  when illegal_commission then
 14  dbms_output.put_line(emp_name ||' : ' || 'has an illegal commission');
 15
 15  END;
 16  /
MARTIN : has an illegal commission

PL/SQL procedure successfully completed.
```

استخدام RAISE APPLICATION ERROR :-

لاحظنا في المثال السابق أننا يجب أن نستخدم **DBMS_OUTPUT** لاستعراض الخطاء الناتج، ولكن الأوراكل يوفر لنا طريقة أخرى لاستعراض الخطاء الناتج وذلك باستخدام الإجراء **raise_application_error** وهو إجراء محفوظ ومخزن في **DBMS_STANDARD Package** . ويتم استخدامه كالتالي.

مثال: استخدام raise_application_error :-

```
SQL> DECLARE
  2  illegal_commission  exception;
  3  emp_name emp.ename%type;
  4  BEGIN
  5  for emp_rec in (select ename,sal,comm from emp) loop
  6     if emp_rec.comm > emp_rec.sal then
  7        emp_name := emp_rec.ename;
  8        raise illegal_commission;
  9     end if;
 10  end loop;
 11  Exception
 12  when illegal_commission then
 13  raise_application_error(-20000,emp_name ||' : ' || 'has an illegal commission');
 14  end;
 15  /
DECLARE
*
ERROR at line 1:
ORA-20000: MARTIN : has an illegal commission
ORA-06512: at line 13
```

☑	**هام جداً:-** عند استخدام raise_application_error فإننا يجب أن نستخدم رقم للخطأ يقع بين 20000– إلى 20999- وهي أرقام أخطاء محجوزة للمستخدم