PL/SQL

Lecture 2 : Exceptions

Exceptions (01/11)

- Reminder : PL/SQL block structure

```
DECLARE
    -- declarations section
BEGIN
    -- executable command(s)
EXCEPTION
    -- exception handling
END;
```

- What we already know
  - The "declare" part (variables)
  - The "begin... end" part (core program)
- Today, we'll see the "Exception" part

Exceptions (02/11)

- Why ?
  - A program runs well, *except* in some critical cases
    - A division by x, which can equal to zero
    - An instruction that loads a file, which can be not found
    - An instruction SELECT... INTO that returns no values
    - A program that updates a database by inserting values, which can be inappropriate in some cases
    - ...
  - A good programmer never forgets to ancitipate such critical cases
  - A good program can catch these cases...
  - ... and treat them apart the rest of the code

Exceptions (03/11)

- What and how ?
  - A specific error, with a name (optional), a code and an associated message
  - 4 categories
    - Identified and pre-defined by Oracle
    - User-identified and user-defined
    - Anonymous and pre-defined by Oracle
    - Anonymous and user-defined
  - Can be raised by the user or triggered by Oracle
  - Is written in the "Exception" sub-block
  - When caught, the program is interrupted and the exception's associated instructions are executed

Exceptions (04/11)

- General syntax

```
BEGIN
    ... /* normal program */
EXCEPTION
    WHEN E1 THEN /* instructions associated to E1 */
    WHEN E2 THEN /* instructions associated to E2 */
    ...
    WHEN OTHERS THEN /* instructions for any nonlisted
                        exceptions */
END;
/
```

- E1, E2... identifiers are either pre-defined by Oracle or defined by the user

Exceptions (05/11)

- First example : pre-defined by Oracle

```
DECLARE
    num NUMBER;
    name VARCHAR2(30) := 'Barbie Doll';
BEGIN
    SELECT numprod INTO num FROM Product
    WHERE nameprod = name;
    DBMS_OUTPUT.put_line ('Article ' || name || ' has ' || num || ' as
    number');
EXCEPTION
    WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.put_line('No
    article has such a name');
    WHEN TOO_MANY_ROWS THEN DBMS_OUTPUT.put_line('Too
    many articles carry the name' || name);
    WHEN OTHERS THEN DBMS_OUTPUT.put_line('We''re gonna
    have a problem here');
END;
/
```

Exceptions (06/11)

- Examples of Oracle's pre-defined exceptions
  - NO_DATA_FOUND

    A request SELECT... INTO doesn't return any line
  - TOO_MANY_ROWS

    A request SELECT... INTO returns several lines
  - DUP_VAL_ON_INDEX

    A request INSERT INTO is refused because of unicity problems
  - ROWTYPE_MISMATCH

    Incompatible type parameters
  - LOGON_DENIED

    Bad login/password during connection
- ... and many more !

Exceptions (07/11)

- Second example : user-defined

```
BEGIN
EXCEPTION
    WHEN My_exc THEN
        DBMS_OUTPUT('My_Exc has been raised !");
    WHEN OTHERS THEN
        DBMS_OUTPUT.put_line('SQLCODE = ' || SQLCODE);
        DBMS_OUTPUT.put_line('SQLERRM = ' || SQLERRM);
END;
```

- Global variables associated to exceptions
  - SQLCODE : Oracle error code
  - SQLERRM : error message corresponding to the latest raised exception

Exceptions (08/11)

- Fully user-defined : declared, raised and caught
  - Exception is a type, therefore it can be declared
  - It can also be activated in code block, with RAISE
- Example

```
DECLARE
    My_exc EXCEPTION;
BEGIN
    ...
    IF condition THEN RAISE My_exc END IF;
EXCEPTION
    WHEN My_exc THEN /* instructions */
END;
```

Exceptions (09/11)

- Anonymous, pre-defined
  - They have no identifier, only a SQLCODE
  - Check Oracle's documentation to know more
- Example

```
BEGIN
    ...
EXCEPTION
    ...
    WHEN OTHERS THEN
        IF SQLCODE=CODE1 THEN /* instructions */
        ELSIF SQLCODE=CODE2 THEN /* instructions */
        ELSE DBMS_OUTPUT.put_line('I dunno');
END;
```

Exceptions (10/11)

- Anonymous, user-defined
  - They have no name, no SQLCODE
  - They are activated by the programmer with the RAISE APPLICATION_ERROR instruction
- Example

```
BEGIN
    IF condition THEN RAISE APPLICATION_ERROR END IF;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.put_line('Generic error has been raised');
END;
```

Exceptions (11/11)

- Transactions
  - Sometimes, the program needs to update database
  - Problem : how to deal with critical code ?
  - Solution : transaction instructions
    - COMMIT : records all modifications permanently
    - ROLLBACK : cancels all modifications since last commit
- Example and use with exceptions

```
BEGIN
    ... COMMIT;
EXCEPTION
    WHEN My_exc THEN ROLLBACK;
END;
```