**1. PL/SQL로 예외 처리**

PL/SQL 코드를 실행할 때 error 발생하는 경우가 있다. Error는 예외(Exception)를 발생시켜 PL/SQL 블록을 중지시키고 예외 처리기 부분으로 제어가 이동한다. Exception handler는 Exception을 검출하고 조건에 따라 조치 작업을 할 수 있다.

**1.1 예외 처리란**

예외는 PL/SQL 블록의 실행 중에 발생하여 블록의 주요 부분을 중단 시킨다. 항상 PL/SQL 예외가 발생할 때 블록은 항상 종료되지만 마지막 조치 작업을 수행하도록 예외 처리 부분을 작성할 수 있다.

1. 예외란 무엇인가 ?

* PL/SQL을 실행 동안에 발생하는 error 처리를 의미한다.

1. 어떻게 발생되는가 ?

* Oracle 오류가 발생할 때
* 사용자가 직접 발생시킬 수 있다.

1. 처리하는 방법은 무엇인가 ?

* 처리기를 이용한다.
* 실행 환경에 전달한다.

**1.2 예외를 발생시키는 두 가지 방법**

1. Oracle 오류가 발생하면 관련된 예외가 자동적으로 발생한다. 예를 들어, 오류 ORA-014-3는 데이터베이스에서 검색된 행이 전혀 없을 때 발생하며 PL/SQL은 NO\_DATA\_FOUND라 는 예외를 발생시킨다.
2. 블록에 RAISE문을 써서 명시적으로 예외를 발생시킬 수 있다. 발생하는 예외를 사용자가 정의한 것일 수도 있고 미리 정의된 것일 수도 있다.

**1.3 예외 처리**

**1.3.1 예외 트랩(trap)**

만일 예외가 블록의 실행 가능한 섹션에서 발생한다면, 처리는 블록의 예외 섹션에서 해당 예외 처리기로 제어가 넘어 갑니다. PL/SQL 블록이 성공적으로 예외를 처리 한다면 이때 예외는 둘러싸는 블록이나 환경으로 전달 되지 않는다.

**1.3.2 예외 전달**

예외를 처리하는 다른 방법은 실행 환경으로 예외를 전달하도록 하는 것이다. 예외가 블록의 실행부에서 발생하여 해당 예외 처리기가 없다면, PL/SQL 블록의 나머지 부분은 수행되지 못하고 종료된다.

**1.3.3 예외 검출**

예외가 블록의 실행부에서 발생하면 블록의 예외부에 있는 해당 예외 처리부로 제어가 넘어간다.

**1.4 예외의 유형**

실행 중에 ERROR가 발생하면 프로그램이 중단되지 않고 예외에 대한 프로그램을 할 수 있다.

|  |  |  |
| --- | --- | --- |
| 예 외 | 설 명 | 처 리 용 지 시 |
| 정의된 ORACLE  SERVER ERROR | PL/SQL코드에서 자주 발생하는 ERROR을 미리 정의함 | 선언할 수 없고 ORACLE SERVER이 암시적으로 발생 |
| 정의되지 않은  ORACLE SERVER  ERROR | 기타 표준 ORACLE SERVER  ERROR | 사용자가 선언하고 ORACLE SERVER이 그것을 암시적으로 발생 |
| 사용자 정의  ERROR | 프로그래머가 정한 조건이 만족되지 않을 경우 발생 | 사용자가 선언하고 명시적으로 발생한다. |

♣ 참고

Developer/2000에서는 PL/SQL문이 있는 곳에서는 독자적으로 예외를 가진다.

**1.5 예외 정의**

PL/SQL블록의 예외 섹션 내에서 해당 루틴을 포함하므로 모든 에러를 처리할 수 있다. 각각의 에러 처리기는 WHERE절로 구성되는데 그 곳에 에러를 명시하고 WHERE절 뒤에는 예외가 발생했을 때 처리할 문장을 기술한다.

**1.5.1 Syntax**

|  |
| --- |
| **EXCEPTION**  **WHEN** exception1 [OR exception2, . . . .] **THEN**  statement1;  statement2;  . . . . . .  [**WHEN** exception2 [OR exception3, . . . .] **THEN**  statement3;  statement4;  . . . . . .]  [**WHEN OTHERS THEN**  statement5;  statement6;  . . . . . .]  END; |

exception 선언섹션에서 선언된 미리 정의된 예외의 표준 이름 이거나 사용자

정의예외의 이름입니다.

Statement 하나 이상의 PL/SQL또는 SQL문장입니다.

OTHERS 명시적으로 선언되지 않은 모든 예외를 트랩하는 예외 처리 절입니다.

**1.5.2 WHEN OTHERS 예외 처리기**

예외 처리 섹션은 지정된 예외만 트랩(trap)합니다. OTHERS예외 처리기를 사용하지 않으면 다른 예외들은 트랩(trap) 되지 않습니다. 이것은 아직 처리되지 않은 모든 예외를 트랩합니다. 그러므로 OTHERS는 마지막에 정의되는 예외 처리기입니다. 일부 ORACLE 툴들은 어플리케이션에서 이벤트를 발생시키기 위해 일으키는 개별적인 미리 정의된 예외들을 가지고 있습니다. OTHERS는 또한 이 예외들도 트랩합니다.

☞ Guidelines

1. EXCEPTION키워드로 블록의 예외 처리 섹션을 시작합니다.
2. 블록에서 개별적인 작업에 대해 여러 예외 처리기를 정의합니다.
3. 예외가 발생할 때 블록 종료 전에 PL/SQL은 하나의 처리기만 프로세스 합니다. 다른 모든 예외 처리 절 후에 OTHERS절을 넣습니다.
4. 최대 하나의 OTHERS절을 가질 수 있습니다.
5. 예외는 지정(assignment) 문장 또는 SQL문장에서 쓰일 수 없습니다.

**1.6 미리 정의된 ORACLE SERVER 에러**

해당 예외 처리 루틴에서 표준 이름을 참조함으로써 미리 정의된 ORACLE SERVER 에러를 트랩(trap)합니다.

|  |  |  |
| --- | --- | --- |
| 예외 이름 | 에러 번호 | 설 명 |
| ACCESS\_INTO\_NULL | ORA-06530 | 초기화 되지않은 객체의 속성에 대해 값을 지정하는 것을 시도합니다. |
| COLLECTION\_IS\_NULL | ORA-06531 | 초기화되지 않은 중첩 테이블 대해 EXISTS를 제외한 메쏘드 모음의 적용을 시도합니다. |
| CURSOR\_ALREADY\_OPEN | ORA-06511 | 이미 열린 CURSOR의 열기를 시도합니다. |
| DUP\_VAL\_ON\_INDEX | ORA-00001 | 중복 값의 삽입을 시도합니다. |
| INVALID\_CURSOR | ORA-01001 | 잘못된 CURSOR연산이 발생합니다. |
| INVALID\_NUMBER | ORA-01722 | 수의 문자열 전환은 실패입니다. |
| LOGIN\_DENIED | ORA-01017 | 잘못된 사용자명과 비밀 번호로 ORACLE에 로그온합니다. |
| NO\_DATA\_FOUND | ORA-01403 | 데이터를 RETURN하지 않는 SELECT문장 |
| NOT\_LOGGED ON | ORA-01012 | PL/SQL프로그램은 ORACLE에 연결하지 않고 데이터베이스 호출을 발생합니다. |
| PROGRAM\_ERROR | ORA-06501 | PL/SQL은 내부 문제를 가지고 있습니다. |
| ROWTYPE\_MISMATCH | ORA-06504 | 지정문에 포함된 호스트CURSOR변수와 PL/SQL CURSOR변수는 RETURN 유형이 다릅니다. |
| STORAGE\_ERROR | ORA-06500 | PL/SQL이 메모리를 다 써버리거나 또는 메모리가 훼손되었습니다. |
| SUBSCRIPT\_BEYOND\_COUNT | ORA-06533 | 모음의 요소 개수보다 더 큰 인덱스 개수를 사용하는 중첩 테이블 참조합니다. |
| SUBSCRIPT\_OUTSIDE\_LIMIT | ORA-06532 | 범위 밖의 인덱스 번호를 사용하여 중첩 테이블 참조 합니다. |
| TIMEOUT\_ON\_RESOURCE | ORA-00051 | ORACLE이 리소스를 대기하는 동안 시간 초과가 발생합니다. |
| TOO\_MANY\_ROWS | ORA-01422 | 단일 행 SELECT는 하나 이상의 행을 RETURN합니다. |
| VALUE\_ERROR | ORA-06502 | 계산,변환,절단,또는 크기 제약 오류가 발생합니다. |
| ZERO\_DIVIDE | ORA-01476 | 0으로 배분을 시도합니다. |

문제1) 이름을 입력받아 부서번호에 따라 급여를 갱신한다. 단 10번이면25%, 20번 이면 20%, 30번이면 15%을 적용한다.

|  |
| --- |
| SET VERIFY OFF  SET SERVEROUTPUT ON  ACCEPT p\_ename PROMPT ' 이름을 입력하시오 : '  DECLARE  TYPE emp\_record\_type IS RECORD(  v\_empno emp.empno%TYPE,  v\_ename emp.ename%TYPE,  v\_sal emp.sal%TYPE,  v\_deptno emp.deptno%TYPE);  emp\_record emp\_record\_type;  g\_ename emp.ename%TYPE := UPPER('&p\_ename');  BEGIN  SELECT empno,ename,sal,deptno  INTO emp\_record  FROM emp  WHERE ename = g\_ename;  IF emp\_record.v\_deptno = 10 THEN  UPDATE emp  SET sal = TRUNC(emp\_record.v\_sal \* 1.25,-1)  WHERE empno = emp\_record.v\_empno;  ELSIF emp\_record.v\_deptno = 20 THEN  UPDATE emp  SET sal = TRUNC(emp\_record.v\_sal \* 1.20,-1)  WHERE empno = emp\_record.v\_empno;  ELSIF emp\_record.v\_deptno = 30 THEN  UPDATE emp  SET sal = TRUNC(emp\_record.v\_sal \* 1.15,-1)  WHERE empno = emp\_record.v\_empno;  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('&p\_ename' || '는 자료가 없습니다.');  WHEN TOO\_MANY\_ROWS THEN  DBMS\_OUTPUT.PUT\_LINE('&p\_ename' || '는 자료가 여러개 있습니다.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('기타 에러입니다.');  END;  /  SET VERIFY ON  SET SERVEROUTPUT OFF |

**1.7 미리 정의되지 않은 ORACLE SERVER 에러**

우선 에러를 선언하고 나서 OTHERS에서 미리 정의되지 않은 ORACLE SERVER 에러를 처리(에러 번호 확인)합니다. 선언된 예외는 암시적으로 발생합니다. PL/SQL에서 PARAGMA와 EXCEPTION\_INT는 ORACLE 에러 번호와 예외 이름을 관련시키기 위해 컴파일러에게 알려줍니다. PARAGMA는 PL/SQL블록이 실행될 때 처리되지 않는 컴파일러 명령문임을 의미하는 키워드입니다. 블록 내에서 예외 이름이 발생되면 그것을 관련된 ORACLE SERVER 에러번호로 해독하기 위해 PL/SQL컴파일러에게 지시합니다.

**1.7.1 선언 절차**

**가) 선언 부분에서 예외 이름을 선언**

|  |
| --- |
| exception\_name **EXCEPTION**; |

exception\_name 예외 이름을 정의한다.

**나) PRAGMA EXCEPTION\_INIT문장을 사용하여 표준 에러 번호와 선언된 예외를 연결한다.**

|  |
| --- |
| **PRAGMA EXCEPTION\_INIT**(exception\_name, error\_number); |

exception\_name 앞에서 선언 된 예외 이름을 기술한다.

error\_number 표준 ORACLE SERVER의 에러 번호를 기술한다.

**다) 해당 예외 처리 부분에서 선언된 예외를 참조한다.**

문제2) 삭제하고자 하는 사원의 이름을 입력하여 자료를 삭제하여라. EXCEPTION절을 이용하여 각종 에러를 처리하여라.

|  |
| --- |
| SET VERIFY OFF  SET SERVEROUTPUT ON  ACCEPT p\_ename PROMPT '삭제하고자 하는 사원의 이름을 입력하시오 : '  DECLARE  v\_ename emp.ename%TYPE := '&p\_ename';  v\_empno emp.empno%TYPE;  emp\_constraint EXCEPTION;  PRAGMA EXCEPTION\_INIT (emp\_constraint, -2292);  BEGIN  SELECT empno  INTO v\_empno  FROM emp  WHERE ename = UPPER(v\_ename);  DELETE emp  WHERE empno = v\_empno;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('&p\_ename' || '는 자료가 없습니다.');  WHEN TOO\_MANY\_ROWS THEN  DBMS\_OUTPUT.PUT\_LINE('&p\_ename' || '는 자료가 여러개 있습니다.');  WHEN emp\_constraint THEN  DBMS\_OUTPUT.PUT\_LINE('&p\_ename' || '는 삭제할 수 없습니다.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('기타 에러입니다.');  END;  /  SET VERIFY ON  SET SERVEROUTPUT OFF |

**1.8 사용자 정의 예외**

PL/SQL에서는 개별적으로 예외를 정의할 수 있습니다. 사용자 정의 PL/SQL 예외는PL/SQL 블록의 선언 섹션에서 선언 하고RAISE 문장으로 명시적으로 발생시킨다.

**1.8.1 선언 절차**

**가) 선언 섹션에서 사용자가 선언한다.**

|  |
| --- |
| Exception\_name **EXCEPTION**; |

Exception\_name 예외 이름을 정의

**나) 실행 섹션에서 명시적으로 예외를 발생하기 위해 RAISE문장을 사용한다.**

|  |
| --- |
| **RAISE** exception\_name; |

exception\_name 앞에서 선언된 예외 이름을 기술한다.

**다) 해당 예외 처리기 안에 선언된 예외를 참조한다.**

문제3) 조회하고자 하는 부서번호를 입력받아 사원번호,이름,담당업무,급여를 출력하여라. 단 가능한 모든 에러를 EXCEPTION에서 처리한다.

|  |
| --- |
| SET VERIFY OFF  SET SERVEROUTPUT ON  ACCEPT p\_deptno PROMPT '조회하고자 하는 부서번호를 입력하시오 : '  DECLARE  v\_deptno emp.deptno%TYPE := &p\_deptno;  CURSOR emp\_cursor IS  SELECT empno,ename,job,sal  FROM emp  WHERE deptno = v\_deptno;  emp\_deptno\_ck EXCEPTION;  BEGIN  IF v\_deptno NOT IN (10,20,30) THEN  RAISE emp\_deptno\_ck;  ELSE  DBMS\_OUTPUT.PUT\_LINE('사번 이 름 담당업무 급 여');  DBMS\_OUTPUT.PUT\_LINE('---- ---------- --------- ------------');  FOR emp\_record IN emp\_cursor LOOP  DBMS\_OUTPUT.PUT\_LINE(RPAD(emp\_record.empno,4) || ' ' ||  RPAD(emp\_record.ename,11) || RPAD(emp\_record.job,10) ||  RPAD(TO\_CHAR(emp\_record.sal,'$999,990.00'),12));  END LOOP;  END IF;  EXCEPTION  WHEN emp\_deptno\_ck THEN  DBMS\_OUTPUT.PUT\_LINE('&p\_deptno' || '는 자료가 없습니다.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('기타 에러입니다.');  END;  /  SET VERIFY ON  SET SERVEROUTPUT OFF |

**1.9 예외 트래핑 함수**

에러가 발생 했을 때 두 함수를 사용하여 관련된 에러 코드 또는 메시지를 확인할 수 있습니다. 코드 또는 메시지에 따라 에러에 대해 취할 작업을 정할 수 있습니다.

|  |  |
| --- | --- |
| 함 수 | 설 명 |
| SQLCODE | 에러 코드에 대한 숫자를 RETURN한다. |
| SQLERRM | 에러 번호에 해당하는 MESSAGE를 RETURN한다. |

**1.9.1 SQL CODE 값**

|  |  |
| --- | --- |
| SQL CODE 값 | 설 명 |
| 0 | 예외가 없습니다.(NO ERROR) |
| 1 | 사용자 정의 ERROR NUMBER |
| +100 | NO\_DATA\_FOUND 예외 |
| 양의 정수 | 표준 에러 번호 |

문제4) 삭제하고자 하는 사원의 이름을 입력하여 삭제하여라.단 가능한 모든 에러를 처리하여라.

|  |
| --- |
| SET VERIFY OFF  SET VERIFY OFF  SET SERVEROUTPUT ON  ACCEPT p\_ename PROMPT '삭제하고자 하는 사원의 이름을 입력하시오 : '  DECLARE  v\_ename emp.ename%TYPE := '&p\_ename';  v\_empno emp.empno%TYPE;  v\_err\_code NUMBER;  v\_err\_msg VARCHAR2(255);  BEGIN  SELECT empno  INTO v\_empno  FROM emp  WHERE ename = UPPER(v\_ename);  DELETE emp  WHERE empno = v\_empno;  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  v\_err\_code := SQLCODE;  v\_err\_msg := SQLERRM;  DBMS\_OUTPUT.PUT\_LINE('에러 번호 : ' || TO\_CHAR(v\_err\_code));  DBMS\_OUTPUT.PUT\_LINE('에러 내용 : ' || v\_err\_msg);  END;  /  SET VERIFY ON  SET SERVEROUTPUT OFF |

**1.10 RAISE\_APPLICATION\_ERROR**

표준화 되지 않은 에러 코드와 에러 MESSAGE를 RETURN하는 RAISE\_APPLICATION\_ERROR 프로시저를 사용합니다. RAISE\_APPLICATION\_ERROR로 어플리케이션에 대한 에러를 제어할 수 있고 처리되지 않은 에러가 RETURN되지 않도록 합니다.

**1.10.1 Syntax**

|  |
| --- |
| raise\_application\_error (error\_number, message[,{**TRUE**|**FALSE**}]); |

error\_number -20000과 20999사이의 예외에 대해 지정된 번호

message 예외에 대한 사용자 지정 MESSAGE

TRUE|FALSE 선택적 BOOLEAN 매개변수로 TRUE면 에러는 이전의 에러 스택에

의치하고 FALSE(DEFAULT)면 에러는 모든 이전의 에러를 대치합니다.

문제5) 삭제하고자 하는 사원의 이름을 입력하여 삭제하여라.단 가능한 모든 에러를 처리하여라.

|  |
| --- |
| SET VERIFY OFF  SET VERIFY OFF  SET SERVEROUTPUT ON  ACCEPT p\_ename PROMPT '삭제하고자 하는 사원의 이름을 입력하시오 : '  DECLARE  v\_ename emp.ename%TYPE := '&p\_ename';  v\_err\_code NUMBER;  v\_err\_msg VARCHAR2(255);  BEGIN  DELETE emp  WHERE ename = v\_ename;  IF SQL%NOTFOUND THEN  RAISE\_APPLICATION\_ERROR(-20100,'no data found');  END IF;  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  v\_err\_code := SQLCODE;  v\_err\_msg := SQLERRM;  DBMS\_OUTPUT.PUT\_LINE('에러 번호 : ' || TO\_CHAR(v\_err\_code));  DBMS\_OUTPUT.PUT\_LINE('에러 내용 : ' || v\_err\_msg);  END;  /  SET VERIFY ON  SET SERVEROUTPUT OFF |

**◈ 연 습 문 제 ◈**

1. 급여를 입력 받아 하나 이상의 행을 RETURN하면 예외 처리기에서 “사원이 한명 이상입니다”을 출력하고, 행이 없으면 “사원이 없습니다”를 출력하고 한명 있으면 이름,업무,급여를 출력하여라.

2. 급여를 입력 받아 -100부터 +100사이의 모든 사원을 출력하여라.

3. 사용자가 필요한 에러를 정의하여 EXCEPTION에서 사용하는 절차를 설명하여라.

4. PRAGMA는 언제 사용하는가.

5. DEPT TABLE에 행을 삽입하는 SCRIPT를 작성하여라. 단 ERROR 발생시 SQL ERROR CODE와 SQL ERROR MESSAGE를 출력하여라.

6. WHEN OTHERS절을 설명하여라.