03. Nested block & Exception

목차

1. NULL

▣ NULL과 BOOLEAN 연산 실습

- 프로그램 개발시 변수 선언후 초기화 하지 않는 경우나 NULL값이 존재하는 경우 연산(논리연산,비교연산)시 예상치 못하는 결과 발생 가능

```
SET SERVEROUTPUT ON
DECLARE
        V_NUM1   NUMBER(4,2);                          -- 변수선언
        V_NUM2   NUMBER(4,2) := 30;                    -- 변수 선언 AND 초기값 할당
BEGIN
        IF (V_NUM1 > 1 AND V_NUM2 < 31) THEN           --  NULL AND TRUE
                DBMS_OUTPUT.PUT_LINE('(V_NUM1 > 1 AND V_NUM2 < 31) IS TRUE ');
        ELSIF NOT (V_NUM1 > 1 AND V_NUM2 < 31) THEN
                DBMS_OUTPUT.PUT_LINE('(V_NUM1 > 1 AND V_NUM2 < 31) IS FALSE ');
        ELSE
                DBMS_OUTPUT.PUT_LINE(' NOT TRUE, NOT FALSE... ???? ');
        END IF;
END;
/
```

1. NULL

Boolean	일반적인 프로그래밍 언어	TRUE, FALSE
	PL/SQL	TRUE, FALSE, NULL

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	FALSE
NULL	TRUE	FALSE	NULL

NOT	
TRUE	FALSE
FALSE	TRUE
NULL	NULL

2. Nested block & Transaction

▣ Nested
사전적 의미: 중첩된 포개진

▣ Nested block
중첩된 Block 으로 Block 안에 또 다른 Block이 내포된 구조
목적: ① **예외처리** ② 모듈화

```
SET SERVEROUTPUT ON

<<MAIN_BLK>>
DECLARE
        V_DNAME                 VARCHAR2(14);
        V_DEPTNO                NUMBER(2);
        V_LOC                   VARCHAR2(13);
BEGIN
        V_DEPTNO := 77;
        V_DNAME  := 'Global variable';
        V_LOC    := 'Main_Block';

        <<LOCAL_BLOCK_1>>
        DECLARE
                V_DNAME     VARCHAR2(14);
                V_DEPTNO    NUMBER(2);
        BEGIN
                V_DEPTNO := 88;
                V_DNAME  := 'Local variable1';
                V_LOC    := 'Nested_Block1';

                INSERT INTO DEPT
                VALUES(V_DEPTNO,MAIN_BLK.V_DNAME,V_LOC);
        END LOAL_BLOCK_1;
```

① <<MAIN_BLK>>
Block Label 이며 3가지 용도
(a) GOTO 분기하는 영역
(b) Source 가독성 (주용도)
(c) Global 변수 참조

Block 시작전에 표기하며 END 뒤에 Label명 표기
(END뒤 Label 생략 가능)

② Global 변수 와 Local 변수
V_DNAME, V_DEPTNO 2개의 변수는
Main Block과 Nested Block에 각각 정의

변수의 Scope과 LifeTime은?
- Global 변수 Main Block 전체 범위
- Local 변수의 해당 Block 범위

?? Global 변수와 Local 변수의 Scope 과 LifeTime 이 겹치는 구간의 우선순위

③ Label을 사용하여 Main Block의 변수 참조

2. Nested block & Transaction

```
        <<LOCAL_BLOCK_2>>
        DECLARE
                V_DNAME     VARCHAR2(14);
                V_DEPTNO    NUMBER(2);
        BEGIN
                V_DEPTNO := 99;
                V_DNAME  := 'Local variable2';
                V_LOC    :=  'Nested_Block2';

                INSERT INTO DEPT VALUES(V_DEPTNO,V_DNAME,V_LOC);

        END;

        INSERT INTO DEPT VALUES(V_DEPTNO,V_DNAME,V_LOC);
END MAIN_BLK;
/

SELECT * FROM  DEPT;

ROLLBACK;
```

<질문> Block내에서 명시적
Transaction 종료자인
commit, rollback 이 없다
Block과 Transaction은 어떤 관계 ?

[요구사항]
실습예제 에러 각자 수정 하십시요

<참고> Main Block = Outer Block (외부에 있는)
Nested Block = Innner Block (외부에 있는)
Nested Block의 위치는? ① 실행부 ② 예외처리부

2. Nested block & Transaction

▣ Block 과 Transaction

Block 과 Transaction의 관계에 대한 일반적인 오해

① Block의 시작(BEGIN)이 Transaction 시작 이고 Block의 종료(END)가 Transaction 종료.
즉 Block의 단위가 Transaction의 단위이다.

② Block내 에서 반드시 Transaction을 종료해야 한다.

```
INSERT INTO DEPT VALUES(66,'Outer block','Outlander');

<<MAIN_BLK>>
DECLARE
        V_DNAME     VARCHAR2(14);
        V_DEPTNO    NUMBER(2);
        V_LOC       VARCHAR2(13);
BEGIN
        V_DEPTNO := 77;
        V_DNAME  := 'Global var';
        V_LOC    := 'Main_Block';

        <<LOCAL_BLOCK_1>>
        DECLARE
                V_DNAME     VARCHAR2(14);
                V_DEPTNO    NUMBER(2);
        BEGIN
                V_DEPTNO := 88;
                V_DNAME  := 'Local var';
                V_LOC    := 'Nested_Block1';

                INSERT INTO DEPT
                VALUES(V_DEPTNO,MAIN_BLK.V_DNAME,V_LOC);
                COMMIT;
        END LOAL_BLOCK_1;
```

[질문] Transaction의 시작은 어디인가?

[요구] 위 실습에서 몇 개의 Transaction이 실행 되었는가?
Transaction의 시작과 종료를 구분 하십시요

2. Nested block & Transaction

```
        <<LOCAL_BLOCK_2>>
        DECLARE
                V_DNAME     VARCHAR2(14);
                V_DEPTNO    NUMBER(2);
        BEGIN
                V_DEPTNO := 99;
                V_DNAME  := 'Local var';
                V_LOC    := 'Nested_Block2';

                INSERT INTO DEPT VALUES(V_DEPTNO,V_DNAME,V_LOC);

                ROLLBACK;
        END;

        INSERT INTO DEPT VALUES(V_DEPTNO,V_DNAME,V_LOC);
END MAIN_BLK;
/

SELECT * FROM  DEPT;
DELETE FROM DEPT WHERE  DEPTNO IN (66,77,88);
COMMIT;
```

3. Exception

▣ **Error 유형**

발생하는 시점에 따라 2가지 유형으로 구분

(1) **컴파일 시점 에러 (Compile Time Error)**

(2) **실행시점 에러 (Run Time Error)** **-- Exception**

* PL/SQL Block은 (1) 컴파일(Parsing) (2) 실행.

▣ **컴파일 시점 에러 (Compile Time Error)**

```
INSERT INTO DEPT VALUES(66,'Outer block','Outlander');

<<MAIN_BLK>>
DECLARE
        V_DNAME     VARCHAR2(14);
        V_DEPTNO    NUMBER(200);
        V_LOC       VARCHAR2(13);
BEGIN
        V_DEPTNO :=  77;
        V_DNAME  :=  'Global var';
        V_LOC    :=  'Main_Block';
```

[실습] 이전 실습 예제 일부를 수정하여 실습

[확인] NUMBER type은 38자 유효자리

[질문] 위의 예제 실행시 컴파일 시점에 에러가 발생하게 되면 첫번째 INSERT는 어떻게 되는가?

3. Exception

▣ 실행시점 에러 (Run Time Error)

```
SET  SERVEROUTPUT ON

BEGIN
        ①INSERT INTO DEPT VALUES(66,'OUTER_BLK','Main_Blk');

          <<Nested_BLOCK_1>>
          BEGIN
                    INSERT INTO DEPT VALUES(76,'LOCAL_PART1','Nested_Blk1');
                  ②INSERT INTO DEPT VALUES(777,'LOCAL_PART1','Nested_Blk1');      -- Run Time Error 발생
                  --INSERT INTO DEPT VALUES(77,'LOCAL_PART1','Nested_Blk1');
                    INSERT INTO DEPT VALUES(78,'LOCAL_PART1','Nested_Blk1');
                    COMMIT;
          ③                                                                       -- Exception section
          END Nested_BLOCK_1;

          <<Nested_BLOCK_2>>
          BEGIN
                    INSERT INTO DEPT VALUES(86,'LOCAL_PART2','Nested_Blk2');
                    COMMIT;
          END Nested_BLOCK_2;

          INSERT INTO DEPT VALUES(96,'OUTER_BLK_PART','Main_Blk');
END;
/

SELECT * FROM  DEPT WHERE  DEPTNO IN (66,76,77,78,88,99);
DELETE FROM DEPT WHERE  DEPTNO IN (66,76,77,78,88,99);
COMMIT;
```

3. Exception

▣ 실행시점 에러 (Run Time Error)

① 첫번째 Insert시 Transaction 시작

② Nested Block에서 실행시점 에러(Run Time Error)가 발생
- 컬럼 허용 자리수를 초과하는 실행 에러가 발생하여 Statement Rollback 자동 실행

[중요] Exception이 발생하면 발생한 위치 이하의 작업을 수행하지 않고 Block내의 예외처리 영역으로 처리흐름(Control Flow) 전달

[Statement Level Rollback]
Update 연산이 총 10개의 Row를 수정해야 하는데 8번째에서 에러가 발생하면 1~8 Row 까지 수정 사항을 자동으로 Rollback 처리. 해당 Statement 내에 변경된 사항을 Rollback 하는것이 Statement Level Rollback

③ **예외처리부에서 Excpetion을 처리 할수 있는 경우**
Exception을 처리한후 해당 Block을 종료한후 Block의 다음 Statement 실행

④ **예외처리부에서 Excpetion을 처리 할수 없는 경우**
(a) 예외 처리부가 없는경우
(b) 예외 처리부가 있지만 해당 Exception을 제어하지 못하는경우

해당 Block을 종료한후 Exception을 외부에 전달(Excpetion Propagation)

[질문] 예제상에서는 Nested_BLOCK_1 내에서 예외를 처리할수 없기 때문에 외부 Block으로 Exception을 던지게 된다. Main Block내에서도 예외처리부가 없기 때문에 Main Block을 호출한 외부 프로그램(실습환경에서는 SQLDEV)으로 에러를 전달한다. 진행중인 트랜잭션은 어떻게 될까? **실행결과를 조회한후 각자 유추**

3. Exception

▣ Exception 정의

PL/SQL **실행시 발생하는 RUN TIME ERROR.**

▣ Exception 종류

ORACLE defined exception	Predefined	**NO_DATA_FOUND** **TOO_MANY_ROWS** TIMEOUT_ON_RESOURCE
	Non-Predefined (Internally defined)	Name이 없는 Exception ORA-00001 :무결성 제약 조건 (SCOTT.DEPT_DEPTNO_PK)에 위배됩니다
USER defined exception	비즈니스룰에 따라 개발자가 정의	SAL < 0 , E_MINUS_SAL

▣ Exception 발생

ORACLE defined exception	**automatic raised by oracle DBMS**
USER defined exception	**raise command** by user

▣ Exception 처리 장소

① 해당BLOCK내에서 처리

② OUTER BLOCK(CALLING ENVIRONMENT) 에서 처리

3. Exception

<참고>
SQL,PL/SQL에서 정의된 수많은 에러들 중에 Exception 처리부(section) 빈번히 사용되는
에러에게는 사용성 및 식별성을 위해서 Exception Name을 부여하여 Exception Name이
있는 경우 PREDEFINED-ORACLE- EXCEPTION 이라 하며 Exception Name이 없는 경우
Non-PREDEFINED-ORACLE-EXCEPTION 이라 한다.

개발시 PREDEFINED-ORACLE- EXCEPTION은 빈번하게 사용 하고 Non-PREDEFINED-ORACLE- EXCEPTION는 예외처리 명료성을 위해 간혹 사용.

<참고>
EXCEPTION 처리가 된경우 마무리 작업을 EXCEPTION-ROUTINE내에서 처리후 해당 BLOCK을 정상 종료 하지만
EXCEPTION 처리가 안된경우 해당 BLOCK을 비정상 종료 후 PROPAGATE EXCEPTION(외부에 해당 ERROR를 전파) 발생.

[과제]
Oracle 공식 error manual(error message reference) 에서
ora-01403 원인(cause) 및 조치(action)방안을 검색하는 방법 시현 및 설명

3. Exception

▣ Exception 발생 및 처리

```
SET  SERVEROUTPUT ON

BEGIN
        INSERT INTO DEPT VALUES(66,'OUTER_BLK','Main_Blk');

        <<Nested_BLOCK_1>>
        BEGIN
                INSERT INTO DEPT VALUES(76,'LOCAL_PART1','Nested_Blk1');
              ① INSERT INTO DEPT VALUES(777,'LOCAL_PART1','Nested_Blk1');      -- Run Time Error 발생
               --INSERT INTO DEPT VALUES(77,'LOCAL_PART1','Nested_Blk1');
                INSERT INTO DEPT VALUES(78,'LOCAL_PART1','Nested_Blk1');
                COMMIT;
        EXCEPTION                                                              -- Exception section
                WHEN OTHERS THEN
                        NULL;
        END Nested_BLOCK_1;

        <<Nested_BLOCK_2>>
        BEGIN
                INSERT INTO DEPT VALUES(86,'LOCAL_PART2','Nested_Blk2');
                COMMIT;
        END Nested_BLOCK_2;

        INSERT INTO DEPT VALUES(96,'OUTER_BLK_PART','Main_Blk');
END;
/
SELECT * FROM  DEPT WHERE  DEPTNO IN (66,76,77,78,88,99);                      -- 결과로 유추
DELETE FROM DEPT WHERE  DEPTNO IN (66,76,77,78,88,99);
COMMIT;
```

3. Exception

① Exception이 발생 → 해당 Statement를 Rollback처리

② 다음 Statement를 처리하지 않고 해당 Block내의 예외처리부로 제어를 이동.
OTHERS는 IF 조건문의 ELSE 역할과 동일한 기능으로 기타 나머지 모든 예외를 처리하겠다는 의미,

③ NULL 명령어
아무런 처리도 하지 않는 구문으로 기능적으로는 아무런 역할을 하지 않아도 되지만 문법적으로 Statement가 필요한 경우 사용하는 명령어

[질문] Exception을 처리한다는 의미 설명 하십시요

① Block내에서 발생한 Exception이 **외부로 전파되지 않도록 잡기만(Catch) 하는 행위**

② Exception을 잡아서(Catch) ~~오류의 원인을 처리한후~~ **재작업**을 하거나 **오류를 기록하는 행위(logging)**

```
INSERT INTO DEPT VALUES(66,'OUTER_BLK','Main_Blk');
dbms_output.put_line('1111111111111111');
<<Nested_BLOCK_1>>
BEGIN
        INSERT INTO DEPT VALUES(76,'LOCAL_PART1','Nested_Blk1');
        INSERT INTO DEPT VALUES(777,'LOCAL_PART1','Nested_Blk1');
      --INSERT INTO DEPT VALUES(77,'LOCAL_PART1','Nested_Blk1');
        INSERT INTO DEPT VALUES(78,'LOCAL_PART1','Nested_Blk1');
        dbms_output.put_line('before commit');
        COMMIT;
~

dbms_output.put_line('22222222222222');
```

3. Exception

▣ Exception 발생 및 처리

```
SET  SERVEROUTPUT ON
BEGIN
        INSERT INTO DEPT VALUES(66,'OUTER_BLK','Main_Blk');

        <<Nested_BLOCK_1>>
        BEGIN
                INSERT INTO DEPT VALUES(76,'LOCAL_PART1','Nested_Blk1');
                INSERT INTO DEPT VALUES(777,'LOCAL_PART1','Nested_Blk1');      -- Run Time Error 발생
              --INSERT INTO DEPT VALUES(77,'LOCAL_PART1','Nested_Blk1');
                INSERT INTO DEPT VALUES(78,'LOCAL_PART1','Nested_Blk1');
                COMMIT;
        EXCEPTION                                                              -- Exception section
                WHEN NO_DATA_FOUND THEN
                        NULL;
        END Nested_BLOCK_1;

        <<Nested_BLOCK_2>>
        BEGIN
                INSERT INTO DEPT VALUES(86,'LOCAL_PART2','Nested_Blk2');
                COMMIT;
        END Nested_BLOCK_2;

        INSERT INTO DEPT VALUES(96,'OUTER_BLK_PART','Main_Blk');
END;
/
SELECT * FROM  DEPT WHERE  DEPTNO IN (66,76,77,78,86,96);
DELETE FROM DEPT WHERE  DEPTNO IN (66,76,77,78,86,96);
COMMIT;
```

3. Exception

▣ Exception 발생 및 처리

```
SET  SERVEROUTPUT ON
BEGIN
        INSERT INTO DEPT VALUES(66,'OUTER_BLK','Main_Blk');
        <<Nested_BLOCK_1>>
        BEGIN
                INSERT INTO DEPT VALUES(76,'LOCAL_PART1','Nested_Blk1');
                INSERT INTO DEPT VALUES(777,'LOCAL_PART1','Nested_Blk1');      -- Run Time Error 발생
              --INSERT INTO DEPT VALUES(77,'LOCAL_PART1','Nested_Blk1');
                INSERT INTO DEPT VALUES(78,'LOCAL_PART1','Nested_Blk1');
                COMMIT;
        EXCEPTION                                                              -- Exception section
                WHEN NO_DATA_FOUND THEN
                        NULL;

                WHEN  OTHERS THEN
                        NULL;
        END Nested_BLOCK_1;

        <<Nested_BLOCK_2>>
        BEGIN
                INSERT INTO DEPT VALUES(86,'LOCAL_PART2','Nested_Blk2');
                COMMIT;
        END Nested_BLOCK_2;

        INSERT INTO DEPT VALUES(96,'OUTER_BLK_PART','Main_Blk');
END;
/
SELECT * FROM  DEPT WHERE  DEPTNO IN (66,76,77,78,86,96);
DELETE FROM DEPT WHERE  DEPTNO IN (66,76,77,78,86,96);
COMMIT;
```

3. Exception

▣ **Exception 구문**

```
EXCEPTION
  WHEN ex_name_1 THEN
                statements_1                -- Exception handler
                statements_11
                statements_111
  [WHEN ex_name_2 OR ex_name_3 THEN statements_2  ]        -- Exception handler
  [WHEN OTHERS THEN statements_3                  ]        -- Exception handler
END;
```

① Exception 구문은 예외처리 영역(section) 시작 표시

② WHEN 절에서 처리할 Exception 을 명기
공통의 처리를 해야 하는 경우 여러 개의 Exception 을 OR 로 연결 가능

③ Exception을 처리하기 위해 여러 Statement 사용 가능하며 Nested Block을 정의 할수 있다.

④ **OTHERS**는
- 예외처리부내에서 1번만 정의.
- **예외처리부에서 명시되지 않은 모든 다른 EXCEPTION 처리.**
- **맨마지막에 사용** (IF절의 ELSE와 유사)