

데이터베이스프로그래밍- PL/SQL Block & Exception(3차시)

3주차: Block 구조 와 Exception

[학습목표]

PL/SQL의 기본단위인 BLOCK , 데이터 처리에 중요한 TRANSACTION , 실행시간에 발생하는 에러를 제어하는 EXCEPTION의 상관 관계를 학습한다.

[학습목차]

- 3-1 Nested Block
- 3-2 Nested Block 과 Transaction
- 3-3 Exception 정의
- 3-4 Exception 발생 및 처리

3-1 Nested Block

Nested는 사전적 의미로 중첩된 포개진 이라는 의미를 가지고 있으며.

Nested Block은 중첩된 Block 으로 Block 안에 또 다른 Block이 내포된 구조

<참고> Nested Block을 사용하는 주요 이유는

- ① 예외처리 ② 모듈화

데이터베이스프로그래밍- PL/SQL Block & Exception(3차시)

3_1_Nested_blk.sql

```
<<MAIN_BLK>>
DECLARE
    U_DNAME          VARCHAR2(14);
    U_DEPTNO         NUMBER(2);
    U_LOC            VARCHAR2(13);
BEGIN
    U_DEPTNO := 77;
    U_DNAME := 'GLOBAL_PART';
    U_LOC := 'Main_BlK';

    <<LOCAL_BLOCK_1>>
    DECLARE
        U_DNAME          VARCHAR2(14);
        U_DEPTNO         NUMBER(2);
    BEGIN
        U_DEPTNO := 88;
        U_DNAME := 'LOCAL_PART_1';
        U_LOC := 'Nested_BlK1';
        INSERT INTO DEPT VALUES(U_DEPTNO,MAIN_BLK.U_DNAME,U_LOC);
    END LOCAL_BLOCK_1;

    <<LOCAL_BLOCK_2>>
    DECLARE
        U_DNAME          VARCHAR2(14);
        U_DEPTNO         NUMBER(2);
    BEGIN
        U_DEPTNO := 99;
        U_DNAME := 'LOCAL_PART_2';
        U_LOC := 'Nested_BlK2';
        INSERT INTO DEPT VALUES(U_DEPTNO,U_DNAME,U_LOC);
    END;

    INSERT INTO DEPT VALUES(U_DEPTNO,U_DNAME,U_LOC);
END MAIN_BLK;
/
```

SELECT * FROM DEPT;

<참고> Main Block = Outer Block(외부에 있는)

Nested Block = Inner Block

Nested Block의 위치는? ① 실행부 ② 예외처리부

① Label << label_name >>

<<MAIN_BLK>> 는 Label 이며 2가지 용도로 사용

(a) GOTO 분기하는 영역

(b) SOURCE 가독성 (주용도)

데이터베이스프로그래밍- PL/SQL Block & Exception(3차시)

(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

Global 변수의 Scope과 LifeTime은 Main Block 전체에 적용되고

Local 변수의 Scope 과 LifeTime은 해당 Block내에서만 적용.

Global 변수와 Local 변수의 Scope 과 LifeTime이 겹치는 구간의 우선순위?

③ MAIN_BLK.V_DNAME

Label을 사용하여 Local Block내에 동일 이름의 변수가 존재하지만 현재의 Scope 밖의 Main Block 에 정의된 변수 값을 참조하는 기법

<참고> 개발언어 공부시 구조적인 프로그래밍에서는 GOTO 문은 스파게티 처럼 프로그램 처리 로직을 뒤죽박죽으로 만들어서 수정 및 변경을 어렵게 만들어 생산성을 떨어트리고 Bug의 온상이 되기에 되도록이면 사용하지 말라고 한다. 위의 예제에서 Global 변수를 Local Block내에서 재할당하고 참조 하는 기법 역시 동일한 부작용(Side Effect)을 발생하기에 되도록이면 개발시 사용하지 않는 것이 좋다. 위의 예제에서 Global 변수 와 Local 변수의 혼용은 개발시 참조할만한 예제가 아니며, 학습 과정에서 언어적인 기능을 설명 하기 위한 학습용 예제 이다.

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

데이터베이스프로그래밍- PL/SQL Block & Exception(3차시)

3-2 Block 과 Transaction

Block 과 Transaction의 관계에 대한 일반적인 오해

- ① Block의 시작(BEGIN)이 Transaction 시작 이고
Block의 종료(END)가 Transaction 종료 이다.
즉 Block의 단위가 Transaction의 단위이다.
- ② Block내 에서 반드시 Transaction을 종료해야 한다.

3_NESTED_BLOCK_TRANS.SQL

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

DECLARE
    U_DNAME          VARCHAR2(14);
    U_DEPTNO         NUMBER(2);
    U_LOC            VARCHAR2(13);
BEGIN
    U_DEPTNO := 77;
    U_DNAME  := 'GLOBAL_PART';
    U_LOC    := 'Main_BlK';

    <<LOCAL_BLOCK_1>>
    DECLARE
        U_DNAME          VARCHAR2(14);
        U_DEPTNO         NUMBER(2);
    BEGIN
        U_DEPTNO := 88;
        U_DNAME  := 'LOCAL_PART_1';
        U_LOC    := 'Nested_BlK1';
        INSERT INTO DEPT VALUES(U_DEPTNO,U_DNAME,U_LOC);
        COMMIT;
    END LOAL_BLOCK_1;

    <<LOCAL_BLOCK_2>>
    DECLARE
        U_DNAME          VARCHAR2(14);
        U_DEPTNO         NUMBER(2);
    BEGIN
        U_DEPTNO := 99;
        U_DNAME  := 'LOCAL_PART_2';
        U_LOC    := 'Nested_BlK2';
        INSERT INTO DEPT VALUES(U_DEPTNO,U_DNAME,U_LOC);
        COMMIT;
    END LOAL_BLOCK_2;

    INSERT INTO DEPT VALUES(U_DEPTNO,U_DNAME,U_LOC);
END;
/

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

[실습] PK 제약사항 위반시 이전 실습 데이터를 처리한후 실습 진행

[질문] 3개의 Transaction을 찾아서 Transaction의 시작과 종료를 구분 하십시오

[질문] Block과 Transaction의 관계는 ?

데이터베이스프로그래밍- PL/SQL Block & Exception(3차시)

3-3 Exception 정의

3-3-1 Error 유형

에러는 발생하는 시점에 따라 2가지 유형으로 구분

- (1) 컴파일 시점 에러 (Compile Time Error)
- (2) 실행시점 에러 (Run Time Error)

PL/SQL Block은 2단계로 처리 (1) 컴파일 → (2) 실행.

(1) 컴파일 시점 에러(Compile Time Error)

3_NESTED_BLOCK_EXCEPT_1.SQL

```
SET SERVEROUTPUT ON
INSERT INTO DEPT VALUES(66,'OUTER_BLK_PART','Outlander');
DECLARE
    V_DNAME      VARCHAR2(14);
    V_DEPTNO      NUMBER(200);      -- Compile Time Error 유발 요인 및 발생
    --V_DEPTNO      NUMBER(2);
    V_LOC         VARCHAR2(13);
BEGIN
    V_DEPTNO := 77;
    V_DNAME  := 'GLOBAL_PART';
    V_LOC    := 'Main_Blkl';

    <<Nested_BLOCK_1>>
    DECLARE
        V_DNAME      VARCHAR2(14);
        V_DEPTNO      NUMBER(2);
    BEGIN
        V_DEPTNO := 88;
        V_DNAME  := 'LOCAL_PART_1';
        V_LOC    := 'Nested_Blkl1';
        INSERT INTO DEPT VALUES(V_DEPTNO,V_DNAME,V_LOC);
        COMMIT;
    END Nested_BLOCK_1;
END Nested_BLOCK_1;
```

데이터베이스프로그래밍- PL/SQL Block & Exception[3차시]

```
<<Nested_BLOCK_2>>
DECLARE
    V_DNAME      VARCHAR2(14);
    V_DEPTNO     NUMBER(2);

BEGIN
    V_DEPTNO := 99;
    V_DNAME  := 'LOCAL_PART_2';
    V_LOC    := 'Nested_Blk2';
    INSERT INTO DEPT VALUES(V_DEPTNO,V_DNAME,V_LOC);
    COMMIT;

END Nested_BLOCK_2;

INSERT INTO DEPT VALUES(V_DEPTNO,V_DNAME,V_LOC);
END;
/
SELECT * FROM DEPT WHERE DEPTNO IN (66,77,88,99);
DELETE FROM DEPT WHERE DEPTNO IN (66,77,88,99);
COMMIT;
```

[확인] NUMBER는 38자 유효자리

[질문] 실행시 예제는 컴파일 시점에 에러가 발생하게 되면 첫번째 INSERT는 어떻게 되는지?

[실습] 컴파일 시점 에러 확인후 수정하여 실습

데이터베이스프로그래밍- PL/SQL Block & Exception(3차시)

(2) 실행시점 에러 (Run Time Error)

3_NESTED_BLOCK_EXCEPT_2.SQL

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

    <<Nested_BLOCK_1>>
    BEGIN
        INSERT INTO DEPT VALUES(76,'LOCAL_PART_1','Nested_Blk1');
        INSERT INTO DEPT VALUES(777,'LOCAL_PART_1','Nested_Blk1'); -- Run Time Error 발생
        --INSERT INTO DEPT VALUES(77,'LOCAL_PART_1','Nested_Blk1');
        INSERT INTO DEPT VALUES(78,'LOCAL_PART_1','Nested_Blk1');
        COMMIT;
    END Nested_BLOCK_1;

    <<Nested_BLOCK_2>>
    BEGIN
        INSERT INTO DEPT VALUES(88,'LOCAL_PART_2','Nested_Blk2');
        COMMIT;
    END Nested_BLOCK_2;

    INSERT INTO DEPT VALUES(99,'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;
```

- ① 첫번째 Insert 시점에 Transaction이 시작.
 - ② 두번째 Nested Block에서 컬럼 허용 자리수를 초과하는 실행시점 에러(Run Time Error)가 발생하고 Statement Rollback이 실행된다.
- [중요] Exception이 발생하게 되면 발생한 위치 이하의 작업을 수행하지 않고 Block내의 예외처리 영역으로 처리흐름(Control Flow)이 넘어 가게 된다.

[참고] Update 연산이 총 10개의 Row를 수정해야 하는데 8번째에서 에러가 발생하면 1~8 Row 까지 수정되었던 사항을 Rollback 처리를 한다.
해당 Statement 내에 변경된 사항을 Rollback 하는것이 Statement Level Rollback 이다.

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

데이터베이스프로그래밍- PL/SQL Block & Exception(3차시)

④ 예외처리부에서 Exception을 처리 할수 없는 경우

(a) 예외 처리부가 없는경우

(b) 예외 처리부가 있지만 해당 Exception을 제어하지 못하는경우

해당 Block을 종료한후 Exception을 외부에 전달(Exception 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 SQL*Plus)으로 에러를 전달한다.
진행중인 트랜잭션은 어떻게 될까?

3-3-1 Exception 정의

[정의] PL/SQL Block 실행시 발생하는 RUN TIME ERROR.

[종류]

ORACLE Defined Exception	PREDEFINED	NO_DATA_FOUND TOO_MANY_ROWS TIMEOUT_ON_RESOURCE
	NON-PREDEFINED	Name이 없는 Exception ORA-00001 :무결성 제약 조건 (SCOTT.DEPT_DEPTNO_PK)에 위배됩니다
USER Defined Exception	비즈니스룰에 따라 정의	SAL < 0 , E_MINUS_SAL

<참고> SQL,PL/SQL 에 정의된 수많은 에러들 중에 Exception 처리에 빈번히 사용되는 Exception들에게는 사용성 및 식별성을 위해서 Exception Name 부여
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 한다.

데이터베이스프로그래밍- PL/SQL Block & Exception(3차시)

- [발생] ① AUTOMATIC RAISED BY ORACLE DBMS
② RAISE COMMAND BY USER

- [처리장소] ① 해당BLOCK내에서 처리
② OUTER BLOCK(CALLING ENVIRONMENT) 에서 처리

<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)을 하게 된다.

3-4 Exception 발생 및 처리


3_NESTED_BLOCK_EXCEPT_3.SQL

```
BEGIN
    INSERT INTO DEPT VALUES(66,'OUTER_BLK_PART','Main_Bl1k');

    <<Nested_BLOCK_1>>
    BEGIN
        INSERT INTO DEPT VALUES(76,'LOCAL_PART_1','Nested_Bl1k1');
        INSERT INTO DEPT VALUES(77,'LOCAL_PART_1','Nested_Bl1k1'); -- Run Time Error 발생
        --INSERT INTO DEPT VALUES(77,'LOCAL_PART_1','Nested_Bl1k1');
        INSERT INTO DEPT VALUES(78,'LOCAL_PART_1','Nested_Bl1k1');
        COMMIT;
    EXCEPTION
        WHEN OTHERS THEN
            NULL;
    END Nested_BLOCK_1;

    <<Nested_BLOCK_2>>
    BEGIN
        INSERT INTO DEPT VALUES(88,'LOCAL_PART_2','Nested_Bl1k2');
        COMMIT;
    END Nested_BLOCK_2;

    INSERT INTO DEPT VALUES(99,'OUTER_BLK_PART','Main_Bl1k');
END;
/
```



SELECT * FROM DEPT;

- 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 예외
를 처리하겠다는 의미,

데이터베이스프로그래밍- PL/SQL Block & Exception(3차시)

NULL 명령어로 아무런 처리도 하지 않는 구문

기능적으로는 아무런 역할을 하지 않아도 되지만 문법적으로 명령 Statement가 필요한 경우 사용하는 명령어 이다.

[질문] Exception을 처리한다는 의미를 설명 하십시오?

- ① Block내에서 발생한 Exception이 외부로 전파되지 않도록 잡기만(Catch) 하는 행위
- ② Exception을 잡아서(Catch) 오류의 원인을 처리한후 재작업을 하거나 오류를 기록하는 행위

3_NESTED_BLOCK_EXCEPT_4.SQL

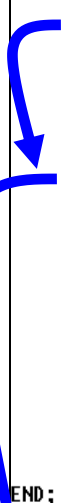
```
BEGIN
    INSERT INTO DEPT VALUES(66,'OUTER_BLK_PART','Main_Blk');

    <<Nested_BLOCK_1>>
    BEGIN
        INSERT INTO DEPT VALUES(76,'LOCAL_PART_1','Nested_Blk1');
        INSERT INTO DEPT VALUES(777,'LOCAL_PART_1','Nested_Blk1'); -- Run Time Error 발생
        --INSERT INTO DEPT VALUES(77,'LOCAL_PART_1','Nested_Blk1');
        INSERT INTO DEPT VALUES(78,'LOCAL_PART_1','Nested_Blk1');
        COMMIT;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            NULL;
    END Nested_BLOCK_1;

    <<Nested_BLOCK_2>>
    BEGIN
        INSERT INTO DEPT VALUES(88,'LOCAL_PART_2','Nested_Blk2');
        COMMIT;
    END Nested_BLOCK_2;

    INSERT INTO DEPT VALUES(99,'OUTER_BLK_PART','Main_Blk');

END;
```



- ① Exception 발생 → Statement Level Rollback 발생
- ② 다음 Statement를 처리하지 않고 해당 Block내의 예외처리부로 제어 이동
ORA-01438: 이 열에 대해 지정된 전체 자릿수보다 큰 값이 허용됩니다.
ORA-06512: 줄 7에서

발생한 Exception은 자리수 초과 오류 이지만 예외 처리부에서는 NO_DATA_FOUND Exception(Fetch된 Row 가 없음 의미)을 처리하도록 되어 있기 때문에 Nested Block내에서 Exception을 처리하지 못하고 외부(Outer Block, 호출 Application)로 exception 전파.

데이터베이스프로그래밍- PL/SQL Block & Exception(3차시)

3_NESTED_BLOCK_EXCEPT_5.SQL

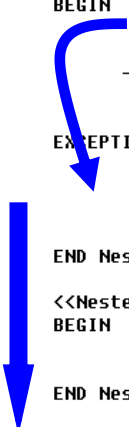
```
BEGIN
    INSERT INTO DEPT VALUES(66,'OUTER_BLK_PART','Main_Blk');

    <<Nested_BLOCK_1>>
    BEGIN
        INSERT INTO DEPT VALUES(76,'LOCAL_PART_1','Nested_Blk1');
        INSERT INTO DEPT VALUES(777,'LOCAL_PART_1','Nested_Blk1'); -- Run Time Error 발생
        --INSERT INTO DEPT VALUES(77,'LOCAL_PART_1','Nested_Blk1');
        INSERT INTO DEPT VALUES(78,'LOCAL_PART_1','Nested_Blk1');
        COMMIT;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            NULL;
        WHEN OTHERS THEN
            NULL;
    END Nested_BLOCK_1;

    <<Nested_BLOCK_2>>
    BEGIN
        INSERT INTO DEPT VALUES(88,'LOCAL_PART_2','Nested_Blk2');
        COMMIT;
    END Nested_BLOCK_2;

    INSERT INTO DEPT VALUES(99,'OUTER_BLK_PART','Main_Blk');

END;
/
```



- ① Exception 발생 → Statement Level Rollback 발생
- ② 다음 Statement를 처리하지 않고 해당 Block내의 예외처리부로 제어를 이동합니다.

ORA-01438: 이 열에 대해 지정된 전체 자릿수보다 큰 값이 허용됩니다.

ORA-06512: 줄 7에서

위의 Exception에 대한 예외처리는 없지만 OTHERS 구문으로 인해 정의되지 않은 모든 예외에 대한 처리를 수행 할수 있게 된다.

3-4-1 Exception 구문

EXCEPTION

```
WHEN exception1 [OR exception 2 ....] THEN
    statement1;
    statement2;
```

```
[ WHEN exception3 [OR exception 4.. ] THEN
    statement1;
    statement2;
```

]

```
[WHEN OTHERS THEN
    statement1;
    statement2;
```

]

데이터베이스프로그래밍- PL/SQL Block & Exception(3차시)

- ① EXCEPTION 구문은 예외처리 ROUTINE의 시작 표시
- ② WHEN 절에서 처리할 EXCEPTION을 명기
 - 공통의 처리를 해야 하는 경우 여러 개의 EXCEPTION을 OR 로 연결 가능
- ③ Exception을 처리하기 위해 여러 Statement 사용 가능
 - 이 위치에 Nested Block을 정의 할수 있다.
- ④ OTHERS는
 - 예외처리부내에서 1번만 정의 됩니다.
 - 예외처리부에서 명시되지 않은 모든 다른 EXCEPTION을 처리합니다.
 - 맨마지막에 사용됩니다. (IF 문장의 ELSE와 유사한 개념)