목차

VIII.예외처리와 트랜잭션

1. 예외처리

- ✓ 프로그램을 개발하다 보면 다양한 경우의 수를 산정해서 오류 확인 및 예외처리를 한다.
- ✓ PL/SQL코드를 작성할 때 발생할 수 있는 오류에는 크게 두 가지가 있다.
  - ① 문법오류로 객체(테이블,뷰,함수,프로시저 등) 나 키워드 이름을 잘못 참조하거나 함수나 프로시저의 매개변수를 잘못 명시했을 때 발생되는 오류 이는 컴파일할 때 걸러진다.
  - ② 실행, 즉 런타임 때 로직을 처리하면서 발생하는 오류인데 이를 **예외(Exception)**라고 한다.
- ✓ 예외에는 크게 두가지로 **시스템 예외** 오라클에서 발생시키는 것과 **사용자 정의 예외로** 구분할 수 있다.

-예외처리 구문

```
EXCEPTION WHEN 예외명1 THEN 예외처리 구문1
          WHEN 예외명2 THEN 예외처리 구문2
          ....
          WHEN OTHERS THEN 예외처리 구문n;
```

1. 예외처리

-SQLCODE, SQLERRM을 이용한 예외정보 참조

- ✓ 단순히 메시지만 출력 하는 것이 아니라 실제로 어떤 오류가 발생했는지 알고 싶을 때 참조하기 위한 방법.
- ✓ SQLCODE : 실행부에서 발생한 예외에 해당하는 코드를 반환
- ✓ SQLERRM : 발생한 예외에 대한 오류 메시지 반환.
- ✓ DBMS_UTILTY.FORMAT_ERROR_BACKTRACE)

```
EXCEPTION WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('오류가 발생했습니다.');
        DBMS_OUTPUT.PUT_LINE('SQL ERROR CODE:' || SQLCODE);
        DBMS_OUTPUT.PUT_LINE('SQL ERROR CODE:' || SQLERRM); -- 매개변수 없는
        DBMS_OUTPUT.PUT_LINE(SQLERRM(SQLCODE));             -- 매개변수 있는
        DBMS_OUTPUT.PUT_LINE(DBMS_UTILITY.FORMAT_ERROR_BACKTRACE);
```

1. 예외처리

-시스템 예외

✓ 예외처리 시 OTHERS 외에도 시스템 예외명을 사용할 수 있다. 이를 미리 정의된 예외 (Predefined)

예외명	예외 코드	설명
ACCESS_INTO_NULL	ORA-06530	LOB과 같은 객체 초기화 되지 않은 상태에서 사용
CASE_NOT_FOUND	ORA-06592	CASE문 사용시 구문 오류
CURSOR_ALREADY_OPEN	ORA-06511	커서가 이미 OPEN된 상태인데 OPEN 하려고 시도
DUP_VAL_ON_INDEX	ORA-00001	유일 인덱스가 있는 컬럼에 중복값으로 INSERT, UPDATE 수행
INVALID_CURSOR	ORA-01001	존재하지 않는 커서를 참조
INVALID_NUMBER	ORA-01722	문자를 숫자로 변환할 때 실패할 경우
NO_DATA_FOUND	ORA-01403	SELECT INTO 시 데이터가 한 건도 없을 경우
VALUE_ERROR	ORA-06502	수치 또는 값 오류
STORAGE_ERROR	ORA-06500	프로그램 수행시 메모리 부족
ZERO_DIVIDE	ORA-01476	0으로 나눌 때

1. 예외처리

-시스템 예외

✓ 미리 정의된 예외는 그 수가 제한되어 있는 반면, 오라클 내부에서 처리하는 예외의 수는 매우 많다.
따라서 보통 예외처리를 할 때 미리 정의된 예외를 먼저, 맨 마지막에 OTHERS를 명시하는 형태로 사용한다.
여러 개의 예외를 명시할 때 OTHERS는 반드시 맨 끝에 명시해야 한다.

2개 이상의 예외명

```
EXCEPTION
WHEN ZERO_DIVIDE THEN
        DBMS_OUTPUT.PUT_LINE('오류1');
        DBMS_OUTPUT.PUT_LINE('SQL ERROR MESSAGE1:' || SQLERRM);
WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('오류2');
        DBMS_OUTPUT.PUT_LINE('SQL ERROR MESSAGE1:' || SQLERRM);
```

✓ 예외처리는 EXCEPTION 절에 명시한 순서대로 처리된다.
즉 처음 명시한 예외가 발생하면 해당되는 로직을 처리하고 이후 예외는 무시하고 프로시저는 종료 된다.

1. 예외처리

-사용자 정의 예외

✓ 개발자가 직접 예외를 정의하는 방법.

① 예외 정의 : 사용자_정의_예외명 EXCEPTION;

② 예외발생시키기 : RAISE 사용자_정의_예외명;
시스템 예외는 해당 예외가 자동으로 검출 되지만, 사용자 정의 예외는 직접 예외를 발생시켜야 한다.
RAISE 예외명 형태로 사용한다.

③ 발생된 예외 처리 : EXCEPTION WHEN 사용자_정의_예외명 THEN ..

2개 이상의 예외명

```
IS
 사용자_정의_예외명 EXCEPTION;
BEGIN
 IF 조건 THEN
    RAISE 사용자_정의_예외명;
EXCEPTION
WHEN 사용자_정의_예외명 THEN
                DBMS_OUTPUT.PUT_LINE('오류내용');
WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('다른오류');
```

2. 트랜잭션

- ✓ 트랜잭션(Transaction) '거래' 라는 뜻으로 은행에서 사용하는 입금과 출금을 하는 거래를 말한다.
  A란 은행에서 출금하여 B란 은행으로 송금을 할때 송금 되는 도중 알 수 없는 오류가 발생하여
  A란 은행에서는 돈이 빠져 나갔는데 B은행 계좌에 입금이 되지 않았다면 !?
- ✓ 이런 경우거래가 성공적으로 모두 끝난 후에 이를 완전한 거래로 승인하고, 거래 도중 오류가 발생하면
  아예 없던 거래로 되돌리는 것이다. 이러한 거래의 안정성을 확보하는 방법이 트랜잭션이다.

-COMMIT과 ROLLBACK

```
 COMMIT;
EXCEPTION WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
        ROLLBACK;
END
```

감사합니다!