4. 데이터 조작어(DML)(5)[트랜잭션(commit,rollback)]

4.5 트랜잭션 (Commit과 Rollback)

**** 데이터베이스 TRANSACTION ****

- 트랜잭션은 데이터 처리의 한 단위 입니다.
- 오라클 서버에서 발생하는 SQL문들을 하나의 논리적인 작업단위로써 성공하거나 실패하는 일련의 SQL문을 트랜잭션이라 보시면 됩니다.
- ORACLE SERVER는 TRANSACTION을 근거로 데이터의 일관성을 보증 합니다.
- TRANSACTION은 데이터를 일관되게 변경하는 DML문장으로 구성됩니다
  (COMMIT, ROLLBACK, SAVEPOINT)

1. TRANSACTION의 시작
   - 실행 가능한 SQL문장이 제일 처음 실행될 때

2. TRANSACTION의 종료
   - COMMIT이나 ROLLBACK
   - DDL이나 DCL문장의 실행(자동 COMMIT)
   - 기계 장애 또는 시스템 충돌(crash)
   - deadlock 발생
   - 사용자가 정상 종료

**** COMMIT과 ROLLBACK ****

AUTOCOMMIT 확인하기

	@@autocommit
▶	1

1: TRUE(자동커밋), 2:FALSE

COMMIT : 변경사항 저장

ROLLBACK : 변경사항 취소

1. AUTOCOMMIT해제

SQL> SET AUTOCOMMIT=0;

**** Commit 과 Rollback 예제 ****

이전의 커밋(COMMIT)이 일어난 뒤부터 다음의 커밋(COMMIT)전 까지의 작업이 하나의 트랜잭션 이며, 커밋과 롤백(ROLLBACK)은 이러한 트랜잭션 단위로 데이터 베이스에서 발생한 작업을 저장, 삭제하는 일입니다.

- **Automatic commit** : DDL(Create, Alter, Drop), DCL(Grant, Revoke)
- **Automatic Rollback** : 비정상적인 종료, system failure

**** 예제 ****

SQL> DELETE FROM emp **WHERE** empno = 7521;

-> 한 개의 행이 삭제 되었습니다.

SQL> COMMIT;

-> 커밋이 완료 되었습니다.

한 개의 행을 삭재하고, COMMIT 문으로 데이터를 영구히 저장했습니다. 이 것은 하나의 트랜잭션이 여기서 종료되고 새로운 트랜잭션이 발생하는 것을 말합니다.

SQL> SELECT empno **FROM** emp **WHERE** empno = 7521;

-> 선택된 레코드가 없습니다.

SQL> INSERT INTO emp(empno, ename, hiredate) **VALUES**(9000, 'test', sysdate);

-> 한 개의 행이 작성되었습니다.

SQL> COMMIT;

-> 커밋이 완료 되었습니다.

SQL> DELETE FROM emp **WHERE** empno = 9000;

-> 한개의 행이 삭제 되었습니다.

SQL> SELECT empno **FROM** emp **WHERE** empno = 9000;

-> 선택된 레코드가 없습니다.

위의 예제를 보면 empno가 9000번인 데이터를 Insert한후 Commit으로 데이터를 저장한 다음에 데이터를 다시 삭제한 후 Select를 해보면 데이터가 검색되지 않는 것을 알 수 있습니다.

하지만 다른 유저에서는 커밋이나 롤백을 하기 전까지 이전에 Insert한 empno가 9000번인 데이터를 조회하면 데이터가 검색 됩니다.

데이터베이스에서의 이런 기능을 읽기 일관성이라고 합니다.

SQL> ROLLBACK;

-> 롤백이 완료 되었습니다.

(이전에 트랜잭션(커밋)이 발생하고나서 지금 발생한 ROLLBACK문 전까지의 작업의 취소를 말합니다.)

검색을 해보면 커밋이 완료된 시점의 레코드 하나가 검색 됩니다.

SQL> SELECT empno FROM emp WHERE empno = 9000;

```
   EMPNO
----------
      9000
```

-> 한 개의 행이 선택되었습니다.

**** SAVEPOINT 와 ROLLBACK TO ****

SAVEPOINT는 사용자가 트랜잭션의 작업을 여러개의 세그먼트로 분할할 수 있도록 하는 특별한 작업입니다.

SAVEPOINT는 **부분적인 롤백을** 가능하게 하기 위해 트랜잭션에 대한 중간점을 정의 합니다.

SQL> INSERT INTO emp(empno, ename, hiredate) **VALUES**(10000, 'test2', sysdate);

-> 한 개의 행이 작성되었습니다.

SQL> SAVEPOINT A;

-> 저장점이 생성 되었습니다. (여기서 SAVEPOINT를 생성했습니다.)

SQL> INSERT INTO emp(empno, ename, hiredate) VALUES(10001, 'test3', sysdate);

-> 한 개의 행이 작성되었습니다.

SQL> INSERT INTO emp(empno, ename, hiredate) VALUES(10002, 'test4', sysdate);

-> 한 개의 행이 작성되었습니다.

SQL> DELETE FROM emp WHERE empno IN(10000, 10001, 10002);

-> 세 개의 행이 삭제 되었습니다.

SQL> SELECT empno, ename FROM emp WHERE empno IN(10000, 10001, 10002);

-> 선택된 행이 없습니다.

SQL> ROLLBACK TO SAVEPOINT A;

-> 롤백이 완료 되었습니다. (SAVEPOINT까지만 롤백이 시행 됩니다.)

SQL> SELECT empno, ename **FROM** emp **WHERE** empno **IN**(10000, 10001, 10002);

-> 한 개의 행이 선택되었습니다.

```
     EMPNO ENAME
---------- ---------------
     10000 test2
```

SAVEPOINT까지만 롤백이 실행되었습니다. 그 결과 첫 번째 데이터는 그대로 남고, SAVEPOINT 후에 실행된 데이터 입력은 삭제되었습니다.