데이터베이스

- 순서 : Ch9-1(트랜잭션)
- 학기 : 2018학년도 2학기
- 학과 : 가천대학교 컴퓨터공학과 2학년
- 교수 : 박양재

데이터베이스

- 목차

9장. 트랜잭션

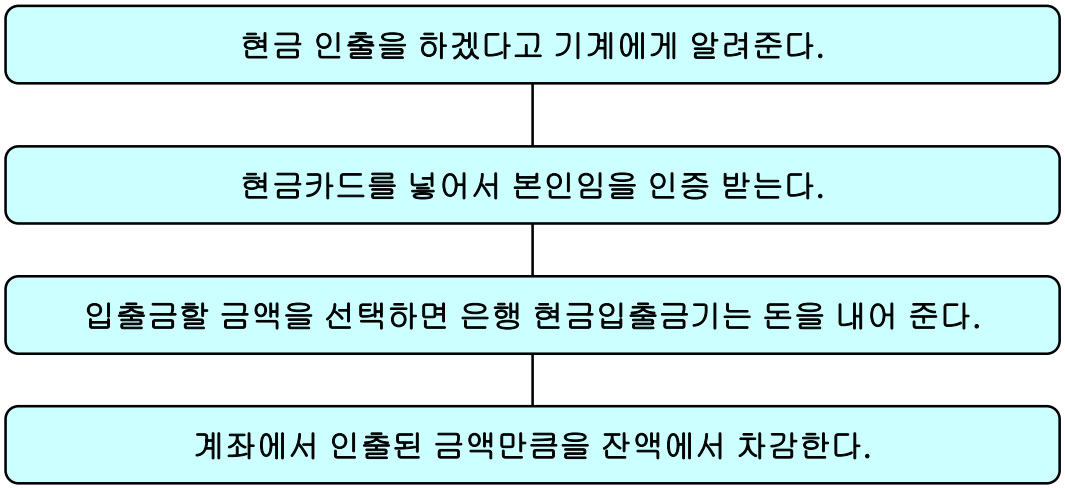
❑ 트랜잭션(transaction)

- ✓ 데이터 처리의 한 단위를 의미
- ✓ 오라클에서 발생하는 여러 개의 SQL명령문을 하나의 논리적인 작업 단위로 처리하는 데 이를 트랜잭션이라고 한다.
- ✓ 하나의 트랜잭션은 ALL-OR-NOTHING 방식으로 처리
- ✓ 데이터베이스에서 작업의 단위를 트랜잭션이라는 개념을 도입한 이유는 데이터의 일관성을 유지하면서 데이터를 안정적으로 복구 시키기 위해서이다.
- ✓ DBMS는 동시에 여러 사용자의 요청을 처리
- ✓ 항공기 예약, 은행, 신용 카드 처리, 대형 할인점 등에서는 대규모 데이터베이스를 수백, 수천 명 이상의 사용자들이 동시에 접근함
- ✓ 많은 사용자들이 동시에 데이터베이스의 서로 다른 부분 또는 동일한 부분을 접근하면서 데이터베이스를 사용함(동일 릴레이션 동일 투플까지)

9장. 트랜잭션(계속)

❑ 트랜잭션(transaction)(계속)

✓ 트랜잭션이 필요한 예



✓ 돈을 인출하려는 일련의 과정이 하나의 묶음으로 처리 되어야 한다. 그리고 처리 도중에 무슨 문제가 발생한다면 진행되던 인출과정 전체를 취소하고 처음부터 시작해야 한다. 이것을 트랜잭션이라고 한다.

✓ 트랜잭션을 제어하기위한 명령어(TRANSACTION CONTROL LANGUAGE)

COMMIT, SAVEPOINT, ROLLBACK이 있다.

9장. 트랜잭션(계속)

❑ COMMIT과 ROLLBACK

01.COMMIT과 ROLLBACK의 개념

COMMIT은 모든 작업을 정상적으로 처리하겠다고 확정하는 명령어로 트랜잭션의 처리과정을 데이터베이스에 반영하기 위해서 변경된 모든 내용을 영구 저장한다. COMMIT 명령어를 수행하면 하나의 트랜잭션 과정이 종료하게 된다.

ROLLBACK은 작업 중 문제가 발생했을 때 트랜잭션의 처리과정에서 발생한 변경사항을 취소하여 트랜잭션 과정을 종료 시킨다. ROLLBACK은 트랜잭션으로 인한 하나의 묶음 처리가 시작되기 이전 상태로 되돌린다.

트랜잭션은 여러 개의 물리적인 작업(DML 명령어)들이 모여 이루어 진다. 이러한 작업 중 하나라도 문제가 발생하면 모든 작업을 취소해야 하므로 이들을 하나의 논리적인 작업단위(트랜잭션)로 구성한다. 문제 발생시 이 논리적인 작업의 단위를 취소하면 되기 때문이다.

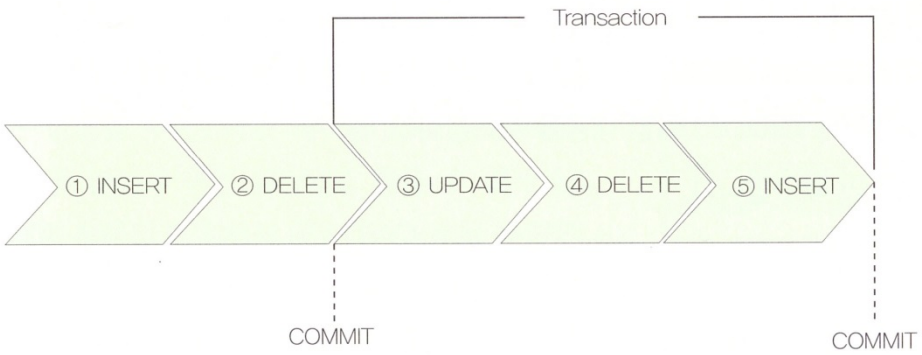
9장. 트랜잭션(계속)

❑ COMMIT과 ROLLBACK

01.COMMIT과 ROLLBACK의 개념

여러 개의 DML 명령어를 어떻게 하나의 논리적인 단위로 트랜잭션으로 묶을 수 있을까?

트랜잭션은 마지막으로 실행한 커밋(혹은 롤백)명령 이후부터 새로운 커밋(혹은 롤백)명령을 실행하는 시점까지 수행된 모든 DML명령들을 의미한다.

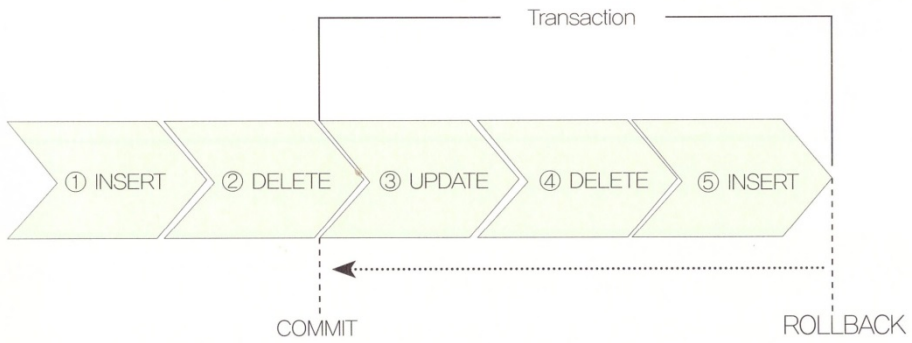


트랜잭션은 DML 명령어 ③, ④, ⑤으로 구성되어 있고, 만약 이 모든 과정이 오류없이 수행되었다면 지금까지 작업한 모든 작업 (하나의 논리적 단위)을 데이터베이스에 영구 저장하라는 명령으로 COMMIT을 수행한다.

9장. 트랜잭션(계속)

❑ COMMIT과 ROLLBACK

01.COMMIT과 ROLLBACK의 개념



트랜잭션은 DML 명령어 ③, ④, ⑤으로 구성되어 있고, 만약 이 모든 과정중에 하나라도 오류가 발생되면 지금까지 작업한 모든 작업 (하나의 논리적 단위)을 취소시켜 이전 상태로 원상복귀 시키는 것을 ROLLBACK이라 한다.

이와 같이 트랜잭션은 ALL-OR-NOTHING방식으로 DML 명령어들을 처리한다.

9장. 트랜잭션(계속)

❑ COMMIT과 ROLLBACK

(1)COMMIT 명령어와 ROLLBACK 명령어의 장점

- 데이터 무결성을 보장한다.
- 영구적인 변경 전에 데이터변경사항들을 확인 할 수 있다.
- 논리적으로 연관된 작업을 그룹화할 수 있다.

(2)COMMIT 명령어

- 트랜잭션(INSERT, UPDATE, DELETE)작업 내용을 실제 데에터베이스에 저장한다.
- 이전 데이터가 완전히 UPDATE 된다.
- 모든 사용자가 변경된 데이터 결과를 볼 수 있다.

(3)ROLLBACK 명령어

- 트랜잭션(INSERT, UPDATE, DELETE)작업 내용을 취소한다.
- 이전 COMMIT한 곳까지만 복구한다.

9장. 트랜잭션(계속)

❑ COMMIT과 ROLLBACK

(4)자동 COMMIT 명령과 자동 ROLLBACK 명령이 되는 경우

데이터베이스 사용자가 COMMIT이나 ROLLBACK명령어를 명시적으로 수행시키지 않더라도 다음과 같은 경우에는 자동 COMMIT과 자동 ROLLBACK이 발생한다.

- SQLP*PLUS가 정상 종료되었다면 자동으로 COMMIT되지만 비정상 종료되었다면 자동으로 ROLLBACK된다.
- DDL과 DCL(GRANT, REVOKE문) 명령문이 수행된 경우 자동으로 COMMIT 된다.
- 정전이 발생했거나 컴퓨터가 다운 되었을 경우 자동으로 ROLLBACK된다.

9장. 트랜잭션(계속)

❑ COMMIT과 ROLLBACK

실습 : ROLLBACK문을 사용한 복구와 커밋으로 영구 저장하기

부서번호 20번인 부서에 대해서만 삭제하려고 했는데 테이블 내의 모든 행이 삭제되어 아무런 데이터도 찾을 수 없게 되었더라도 이전 상태로 되돌릴 수 있다. 그런 다음 부서번호가 20번인 부서만 삭제한 후 커밋으로 영구 저장하기

```
SQL> DROP TABLE DEPT01;

테이블이 삭제되었습니다.

SQL>
SQL> CREATE TABLE DEPT01
  2  AS
  3  SELECT * FROM DEPT ;

테이블이 생성되었습니다.

SQL> SELECT * FROM DEPT01 ;

    DEPTNO DNAME          LOC
---------- -------------- ------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

9장. 트랜잭션(계속)

❑ COMMIT과 ROLLBACK

실습 : ROLLBACK문을 사용한 복구와 커밋으로 영구 저장하기

부서번호 20번인 부서에 대해서만 삭제하려고 했는데 테이블 내의 모든 행이 삭제되어 아무런 데이터도 찾을 수 없게 되었더라도 이전 상태로 되돌릴 수 있다. 그런 다음 부서번호가 20번인 부서만 삭제한 후 커밋으로 영구 저장하기

```
SQL> DELETE FROM DEPT01 ;

4 행이 삭제되었습니다.

SQL> SELECT * FROM DEPT01 ;

선택된 레코드가 없습니다.

SQL> ROLLBACK ;

롤백이 완료되었습니다.

SQL> SELECT * FROM DEPT01 ;

    DEPTNO DNAME          LOC
---------- -------------- ----------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

9장. 트랜잭션(계속)

❑ COMMIT과 ROLLBACK

실습 : ROLLBACK문을 사용한 복구와 커밋으로 영구 저장하기

부서번호 20번인 부서에 대해서만 삭제하려고 했는데 테이블 내의 모든 행이 삭제되어 아무런 데이터도 찾을 수 없게 되었더라도 이전 상태로 되돌릴 수 있다.

그런 다음 부서번호가 20번인 부서만 삭제한 후 커밋으로 영구 저장하기

ROLLBACK해도 영구저장 된다.

```
SQL> DELETE FROM DEPT01
  2  WHERE DEPTNO=20 ;

1 행이 삭제되었습니다.

SQL> SELECT * FROM DEPT01 ;

    DEPTNO DNAME          LOC
---------- -------------- ----------
        10 ACCOUNTING     NEW YORK
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

SQL> COMMIT;

커밋이 완료되었습니다.
```

9장. 트랜잭션(계속)

❑ 자동 COMMIT

실습 : CREATE문에 의한 자동 COMMIT

```
SQL> DROP TABLE DEPT02;

테이블이 삭제되었습니다.

SQL>
SQL> CREATE TABLE DEPT02
  2  AS
  3  SELECT * FROM DEPT;

테이블이 생성되었습니다.

SQL>
SQL> DELETE FROM DEPT02
  2  WHERE DEPTNO=40;

1 행이 삭제되었습니다.

SQL> SELECT * FROM DEPT02;

    DEPTNO DNAME          LOC
---------- -------------- -----------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
```

```
SQL> CREATE TABLE DEPT03
  2  AS
  3  SELECT * FROM DEPT;

테이블이 생성되었습니다.

SQL> ROLLBACK ;

롤백이 완료되었습니다.

SQL> SELECT * FROM DEPT02;

    DEPTNO DNAME          LOC
---------- -------------- -----------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
```

#.ROLLBACK을 수행하더라도 CREATE TABLE DEPT03이 수행되어 자동 COMMIT되어 ROLLBACK이 수행되지 않고 자동 COMMIT 되었다.

9장. 트랜잭션(계속)

❑ 자동 COMMIT

실습 : TRUNCATE문의 실패에 의한 자동 COMMIT

TRUNCATE문이 실패했더라도 이전에 수행했던 DML명령어가 자동으로 COMMIT되는것을 확인하기

```
SQL> DROP TABLE DEPT03;

테이블이 삭제되었습니다.

SQL>
SQL>
SQL> CREATE TABLE DEPT03
  2  AS
  3  SELECT * FROM DEPT ;

테이블이 생성되었습니다.

SQL> SELECT * FROM DEPT03 ;

    DEPTNO DNAME          LOC
---------- -------------- ----------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

```
SQL> DELETE FROM DEPT03
  2  WHERE DEPTNO=20 ;

1 행이 삭제되었습니다.

SQL> SELECT * FROM DEPT03;

    DEPTNO DNAME          LOC
---------- -------------- ----------
        10 ACCOUNTING     NEW YORK
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

9장. 트랜잭션(계속)

❑ 자동 COMMIT

실습 : TRUNCATE문의 실패에 의한 자동 COMMIT

TRUNCATE문이 실패했더라도 이전에 수행했던DML명령어가 자동으로 COMMIT되는것을 확인하기(고의로 TRUNCATE문 에러발생 시키면 ROLLBACK을 수행해도 20번 부서 행이 살아나지 못한다.)

```
SQL> TRUNCATE TABLE DEPTAAAA ;
TRUNCATE TABLE DEPTAAAA
               *
1행에 오류:
ORA-00942: 테이블 또는 뷰가 존재하지 않습니다
```

```
SQL> ROLLBACK ;

롤백이 완료되었습니다.

SQL> SELECT * FROM DEPT03 ;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

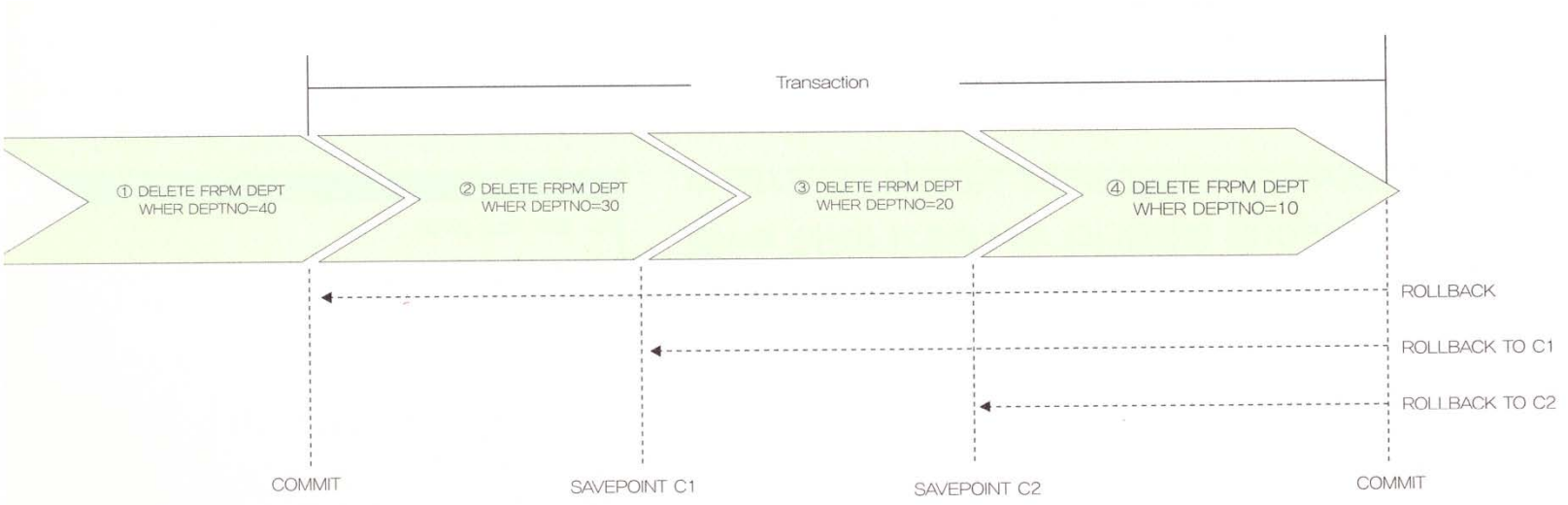
*.TRUNCATE문은 테이블 전체를 삭제하면서 복구가 불가능,DELETE는 일부 삭제 복구가능

*.TRUNCATE문은 DELETE보다 속도가 빠르다.

9장. 트랜잭션(계속)

❑ 트랜잭션을 작게 분할하는 SAVEPOINT

SAVEPOINT 명령을 사용해서 현재의 트랜잭션을 작게 분할 할 수 있다. 저장된 SAVEPOINT는 ROLLBACK TO SAVEPOINT문을 사용하여 표시한 것까지 롤백할 수 있다.



COMMIT명령 후 다음 COMMIT명령이 나타날 때까지가 하나의 트랜잭션으로 구성되므로 ②~④까지가 하나의 트랜잭션이된다. 이렇게 트랜잭션의 중간 중간에 SAVEPOINT명령으로 위치를 지정해 놓으면 ROLLBACK TO C까지 ROLLBACK할수 있다.

9장. 트랜잭션(계속)

❑ 트랜잭션을 작게 분할하는 SAVEPOINT

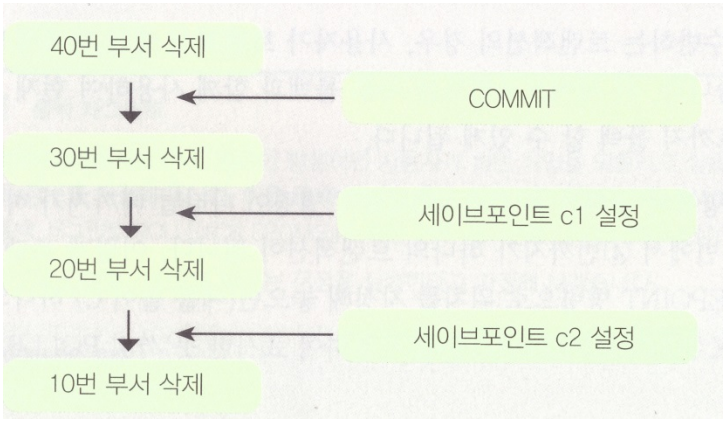
SAVEPOINT로 특정 위치지정하기 위한 형식

[형식1] SAVEPOINT LABEL_NAME ;

SAVEPOINT로 지정해 놓은 특정 위치로 되돌아가지 위한 형식

[형식 2] ROLLBACK TO LABEL_NAME ;

실습: 트랜잭션 중간 단계에서 SAVEPOINT 지정하기

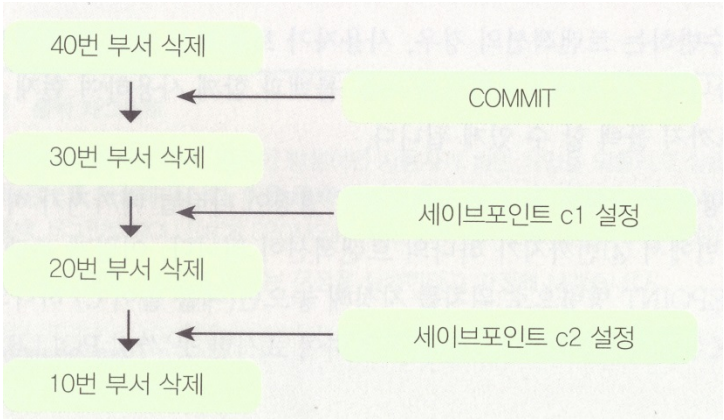


9장. 트랜잭션(계속)

❑ 트랜잭션을 작게 분할하는 SAVEPOINT

실습: 트랜잭션 중간 단계에서 SAVEPOINT 지정하기

```
SQL> DROP TABLE DEPT01;

테이블이 삭제되었습니다.

SQL> CREATE TABLE DEPT01
  2  AS
  3  SELECT * FROM DEPT ;

테이블이 생성되었습니다.

SQL> DELETE FROM DEPT01 WHERE DEPTNO=40;

1 행이 삭제되었습니다.

SQL> COMMIT ;

커밋이 완료되었습니다.

SQL> SELECT * FROM DEPT01 ;

    DEPTNO DNAME          LOC
---------- -------------- ---------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
```

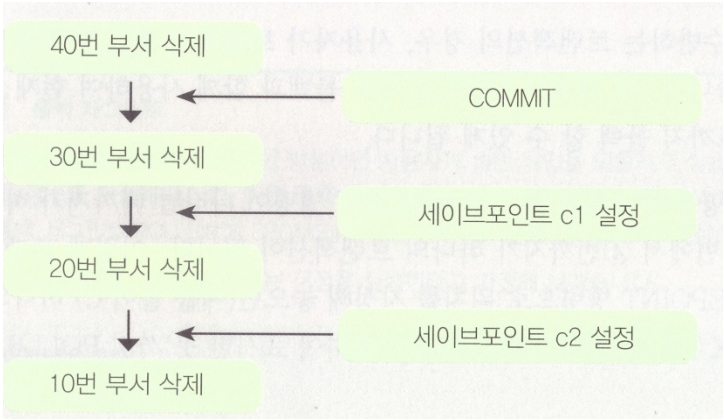


9장. 트랜잭션(계속)

❑ 트랜잭션을 작게 분할하는 SAVEPOINT

실습: 트랜잭션 중간 단계에서 SAVEPOINT 지정하기

```
SQL> DELETE FROM DEPT01 WHERE DEPTNO=30 ;

1 행이 삭제되었습니다.
SQL> SAVEPOINT C1 ;

저장점이 생성되었습니다.

SQL> DELETE FROM DEPT01 WHERE DEPTNO=20 ;

1 행이 삭제되었습니다.

SQL> SELECT * FROM DEPT01 ;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
SQL> SAVEPOINT C2 ;

저장점이 생성되었습니다.

SQL> DELETE FROM DEPT01 WHERE DEPTNO=10 ;

1 행이 삭제되었습니다.

SQL> SELECT * FROM DEPT01 ;

선택된 레코드가 없습니다.
```



9장. 트랜잭션(계속)

❑ 트랜잭션을 작게 분할하는 SAVEPOINT

실습: 트랜잭션 중간 단계로 되돌리기(부서번호가 10번인 사원이 삭제되기 전으로 되돌리기)

①지금 ROLLBACK명령 실행하면 이전 COMMIT 지점으로 되돌아 가므로 10, 20, 30부서의 삭제가 모두 취소된다. 따라서 10번 부서의 삭제 이전으로 되돌리려면 다시 30, 20번 부서를 삭제해야 한다.10번 부서를 삭제하기 전에 SAVEPOINT C2를 설정해 두었으므로 이곳까지만 롤백하면 된다.

```
SQL> ROLLBACK TO C2 ;
롤백이 완료되었습니다.
SQL> SELECT * FROM DEPT01 ;

    DEPTNO DNAME          LOC
---------- -------------- -----------
        10 ACCOUNTING     NEW YORK
```

9장. 트랜잭션(계속)

❑ 트랜잭션을 작게 분할하는 SAVEPOINT

실습: 트랜잭션 중간 단계로 되돌리기(부서번호가 10번인 사원이 삭제되기 전으로 되돌리기)

② 부서번호가 20번인 사원을 삭제하기 바로 직전으로 되돌아가려면 SAVEPOINT까지만 롤백하면 된다.

```
SQL> ROLLBACK TO C1 ;

롤백이 완료되었습니다.

SQL> SELECT * FROM DEPT01 ;

    DEPTNO DNAME          LOC
---------- -------------- ----------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
```

9장. 트랜잭션(계속)

❑ 트랜잭션을 작게 분할하는 SAVEPOINT

실습: 트랜잭션 중간 단계로 되돌리기(부서번호가 10번인 사원이 삭제되기 전으로 되돌리기)

③ 마지막으로 이전 트랜잭션까지 롤백하면 전체가 롤백된다.

```
SQL> ROLLBACK ;

롤백이 완료되었습니다.

SQL> SELECT * FROM DEPT01 ;

    DEPTNO DNAME          LOC
---------- -------------- ----------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
```