SQL

✓ 원리를 알면 IT가 맛있다

SQL for Beginners

chapter 08.

DML

❑ 학습목표

- DML 개요
- 다양한 INSERT 처리
- UPDATE
- DELETE
- Transaction (commit / rollback)
- lock 경합

❑ 1) DML (Data Manipulation Language)

- DML 용도 및 종류

문장	설명
SELECT	데이터베이스로부터 데이터를 검색
INSERT UPDATE DELETE MERGE	데이터베이스 내의 테이블에 새로운 행을 입력하거나, 기존의 행을 수정 또는 삭제하는 명령어로 데이터 조작어(DML : Data Manipulation Language)라고 함
CREATE ALTER DROP RENAME TRUNCATE	테이블을 생성, 변경, 삭제하는 명령어로 데이터 정의어(DDL : Data Definition Language)라고 함
COMMIT ROLLBACK SAVEPOINT	DML 문장에 의한 변경 사항을 관리하거나, 변경사항을 하나의 논리적 트랜잭션으로 포함시키는 명령어
GRANT REVOKE	데이터베이스와 데이터베이스를 구성하는 구조(테이블, 뷰 등)에 접근 권한을 부여하거나 회수하는 명령어로 데이터 제어어(DCL : Data Control Language)라고 함

❑ 2) INSERT – 단일행 입력

- INSERT 용도

– 테이블에 데이터를 입력하기 위한 데이터 조작어 이다.
– 한번에 하나의 행을 테이블에 입력하는 방법과 서브쿼리를 이용하여 한번에 여러 행을 동시에 입력하는 방법이 있다.

```
INSERT INTO table [(column [, column ...])]
VALUES (value [, value ...])
```

– INTO 절에 명시한 컬럼에 VALUES 절에서 지정한 컬럼값을 입력한다. (일대일 대응)
– INTO 절에 컬럼명을 지정하지 않으면 테이블 생성시 정의한 컬럼순서와 동일한 순서로 입력된다.
– 입력되는 데이터의 타입은 컬럼의 데이터 타입과 같아야 되며, 입력되는 데이터의 크기는 컬럼의 크기보다 작아야 된다. (문자와 날짜는 반드시 ‘’ 지정)

❑ 2) INSERT – 단일행 입력

```
SQL> DESC DEPT;
 이름                                      널?      유형
 ----------------------------------------- -------- ----------------

 DEPTNO                                    NOT NULL NUMBER(2)
 DNAME                                              VARCHAR2(14)
 LOC                                                VARCHAR2(13)
```

```
SQL> INSERT INTO DEPT(DEPTNO,DNAME,LOC)
  2  VALUES( 90,'인사과','서울');

1 개의 행이 만들어졌습니다.
```

```
SQL> INSERT INTO DEPT( LOC, DNAME, DEPTNO)
  2  VALUES( '서울','인사과',70 );

1 개의 행이 만들어졌습니다.
```

```
SQL> INSERT INTO DEPT
  2  VALUES( 80 ,'인사과','서울');

1 개의 행이 만들어졌습니다.
```

❑ 2) INSERT – NULL 값 입력

- NULL 값 입력

: 데이터를 입력하는 시점에 해당 컬럼값을 모르거나 미확정일 때에는 null값을 입력할 수 있다.

1) 묵시적 방법

- INTO 절에서 해당 컬럼명과 값을 생략하면 된다.
  단, 해당 컬럼에 NOT NULL 제약조건이 지정된 경우에는 생략 불가하다.

```
SQL> INSERT INTO DEPT( DEPTNO, DNAME )
  2  VALUES( 91, '인사과' );

1 개의 행이 만들어졌습니다.
```

2) 명시적 방법

- VALUES절의 컬럼값에 null 키워드를 적거나 빈문자열 ‘’을 사용하면 된다.

```
SQL> INSERT INTO DEPT
  2  VALUES( 92, '인사과', NULL );

1 개의 행이 만들어졌습니다.
```

❑ 2) INSERT – 특정 데이터 입력

- INSERT 의 특수한 형태

1. 특수 값 입력 (SYSDATE , USER)

```
SQL> INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
  2  VALUES(9000, USER, '연구원', 7839, SYSDATE, 5000, NULL, 90);

1 개의 행이 만들어졌습니다.
```

2. 특정 데이터 타입으로 입력 (RR/MM/DD 형식 , TO_DATE() 사용)

```
SQL> INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
  2  VALUES (9001, '홍길동', 'MANAGER', 7839, '2000/01/01', 2000, NULL, 30 );

1 개의 행이 만들어졌습니다.


SQL> INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
  2  VALUES (9001, '임꺽정', 'MANAGER', 7839, TO_DATE('1999-12-13','YYYY/MM/DD')
, 2000, NULL, 30 );

1 개의 행이 만들어졌습니다.
```

❑ 3) INSERT – 다중행 입력

- 다중 행 입력

– 서브쿼리 절을 이용하여 하나의 INSERT 명령문에 여러 행을 동시에 입력 가능하다.

```
SQL> CREATE TABLE copy_emp
  2  AS
  3  SELECT empno,ename FROM emp
  4  WHERE 1=2;

테이블이 생성되었습니다.

SQL> INSERT INTO copy_emp ( empno, ename )
  2  SELECT empno, ename FROM emp;

14 개의 행이 만들어졌습니다.
```

❑ 4) INSERT – 다중 테이블 다중 행 입력

■ 복수 테이블에 대한 INSERT 문

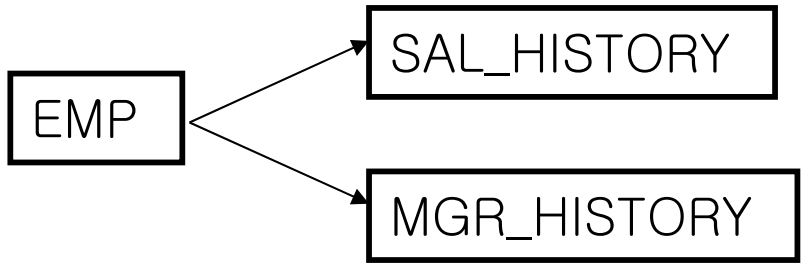
– 한번의 INSERT 시 복수 테이블에 데이터 저장.
– 데이터웨어 하우스를 구축 시 원천데이터(OLTP)로부터 여러 목표 테이블로 데이터 전송에 사용될 수 있다.

1) 무조건 INSERT ALL
2) 조건 INSERT ALL
3) 조건 INSERT FIRST

•OLTP는 웹처럼 다수의 사용자가 온라인으로 데이터를 조회하기 위한 업무.
•데이터웨어하우스는 경영진단이나 사업성과 예측등을 위해 장기간 축적된 대량의 데이터를 일괄처리식으로 분석하기 위한 업무.

❑ 4) INSERT – 다중 테이블 다중 행 입력

- 무조건 INSERT ALL




```
SQL> CREATE TABLE  sal_history
  2  AS
  3  SELECT empno , hiredate , sal
  4  FROM emp
  5  WHERE 1 = 2;

SQL> CREATE TABLE  mgr_history
  2  AS
  3  SELECT  empno, mgr , sal
  4  FROM emp
  5  WHERE 1= 2;
```

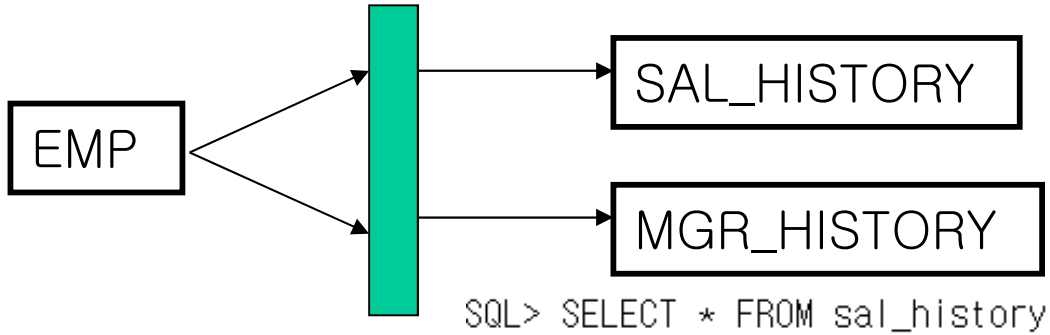
```
SQL> INSERT ALL
       INTO sal_history VALUES ( empno , hiredate, sal )
       INTO mgr_history VALUES ( empno , mgr , sal )
       SELECT empno , hiredate , sal , mgr
       FROM emp ;
```

```
SQL> SELECT * FROM sal_history;

     EMPNO HIREDATE         SAL
---------- -------- ----------
      7369 80/12/17         800
      7499 81/02/20        1600

SQL> SELECT * FROM mgr_history;

     EMPNO        MGR        SAL
---------- ---------- ----------
      7369       7902        800
      7499       7698       1600
```

*학생테이블에서 height 테이블과 weight 테이블로 분리.

❑ 4) INSERT – 다중 테이블 다중 행 입력

- 조건 INSERT ALL
  – 서브쿼리 조건이 만족하면 WHEN모두 실행.




```
SQL> CREATE TABLE  sal_history
  2  AS
  3  SELECT empno , hiredate , sal
  4  FROM emp
  5  WHERE 1 = 2;

SQL> CREATE TABLE  mgr_history
  2  AS
  3  SELECT  empno, mgr , sal
  4  FROM emp
  5  WHERE 1= 2;
```

```
SQL>INSERT ALL
      WHEN  sal < 2500    THEN
                  INTO  sal_history  VALUES ( empno , hiredate, sal )
      WHEN  sal > 2500    THEN
                  INTO  mgr_history VALUES ( empno , mgr , sal )
       SELECT empno , hiredate , sal , mgr
       FROM emp ;
```

```
SQL> SELECT * FROM sal_history;

     EMPNO HIREDATE          SAL
---------- -------- ----------
      7369 80/12/17         800
      7499 81/02/20        1600
      7521 81/02/22        1250
      7654 81/09/28        1250
      7782 81/06/09        2450
      7844 81/09/08        1500
      7876 87/05/23        1100
      7900 81/12/03         950
      7934 82/01/23        1300
```

```
SQL> SELECT * FROM mgr_history;

     EMPNO        MGR        SAL
---------- ---------- ----------
      7566       7839       2975
      7698       7839       2850
      7788       7566       3000
      7839                  5000
      7902       7566       3000
```

❑ 4) INSERT – 다중 테이블 다중 행 입력

- 조건 INSERT FIRST

```
SQL> CREATE TABLE  sal_history
  2  AS
  3  SELECT empno , hiredate , sal
  4  FROM emp
  5  WHERE 1 = 2;

SQL> CREATE TABLE  mgr_history
  2  AS
  3  SELECT  empno, mgr , sal
  4  FROM emp
  5  WHERE 1= 2;

SQL> create table test_history
  2  as
  3  select empno , sal
  4  from emp
  5  where 1 = 2;
```

```
SQL>INSERT FIRST
      WHEN  sal = 800  THEN
                  INTO  sal_history  VALUES ( empno , hiredate, sal )
      WHEN  sal < 2500          THEN
                  INTO  mgr_history VALUES ( empno , mgr , sal )
      ELSE
                  INTO  test_history  VALUES ( empno , sal )
       SELECT empno , hiredate , sal , mgr
       FROM emp ;
```

```
SQL> select * from sal_history;

     EMPNO HIREDATE        SAL
---------- -------- ----------
      7369 80/12/17        800
```

```
SQL> select * from mgr_history;

     EMPNO        MGR        SAL
---------- ---------- ----------
      7499       7698       1600
      7521       7698       1250
      7654       7698       1250
      7782       7839       2450
      7844       7698       1500
      7876       7788       1100
      7900       7698        950
      7934       7782       1300
```

```
SQL> select * from test_history;

     EMPNO        SAL
---------- ----------
      7566       2975
      7698       2850
      7788       3000
      7839       5000
      7902       3000
```

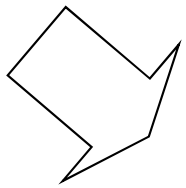
❑ 5) UPDATE

- UPDATE 용도
  : 테이블에 저장된 행들을 변경하는 문장이다.
  : 한 번에 여러 개의 행들을 변경할 수 있다.

```
UPDATE table
SET column = value [, column = value, ... ]
[WHERE condition];
```

```
SQL> SELECT *
  2  FROM DEPT;

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

```
SQL> UPDATE DEPT
  2  SET DNAME = '경리과', LOC ='부산'
  3  WHERE DEPTNO = 90;

1 행이 갱신되었습니다.
```



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

❑ 5) UPDATE

- UPDATE 의 특수한 형태

  -서브쿼리를 이용한 복수 컬럼 변경

```
SQL> UPDATE EMP
  2  SET JOB = ( SELECT JOB
  3              FROM EMP
  4              WHERE EMPNO = 7900 ),
  5      SAL = ( SELECT SAL
  6              FROM EMP
  7              WHERE EMPNO = 7844 )
  8  WHERE EMPNO = 9001;

1 행이 갱신되었습니다.
```

❑ 6) DELETE

- DELETE 용도
  : 테이블에 저장된 행들을 삭제한다.
  : 한 번에 여러 개의 행들을 삭제할 수 있다.

```
DELETE [FROM] table
[WHERE condition];
```

```
SQL> DELETE FROM DEPT
  2  WHERE DEPTNO = 91;

1 행이 삭제되었습니다.
```

- DELETE 의 특수한 형태

  - 서브쿼리 및 다른 테이블 기반를 이용한 데이터 삭제

```
SQL> DELETE FROM EMP
  2  WHERE DEPTNO = ( SELECT DEPTNO
  3                   FROM DEPT
  4                   WHERE DNAME = '경리과' );

1 행이 삭제되었습니다.
```

- MERGE 용도

- 구조가 같은 **2개의 테이블을 비교하여 하나의 테이블로 합치기** 위한 데이터 조작어.
- WHEN 절의 조건절에서 대상 테이블에 해당 행이 이미 존재하면 UPDATE 가 실행되고 존재하지 않으면 INSERT가 실행된다.

- 데이터웨어하우스에서 주로 사용.
예> 전자상거래 (하나의 판매데이터 테이블 → 월별로 관리하고 연말에 MERGE)

```
MERGE INTO table_name table_alias
  USING (table|view|sub_query) alias
  ON (join condition)
  WHEN MATCHED THEN
    UPDATE SET
    col1 = col1_val,
    col2 = col2_val
  WHEN NOT MATCHED THEN
    INSERT (column_list)
    VALUES (column_values);
```

❑ 7) MERGE

```
SQL> CREATE TABLE pt_01
  2  ( 판매번호 VARCHAR2(8),
  3    제품번호 NUMBER,
  4    수량     NUMBER,
  5    금액     NUMBER);

SQL> CREATE TABLE pt_02
  2  ( 판매번호 VARCHAR2(8),
  3    제품번호 NUMBER,
  4    수량     NUMBER,
  5    금액     NUMBER);

SQL> CREATE TABLE p_total
  2  ( 판매번호 VARCHAR2(8),
  3    제품번호 NUMBER,
  4    수량     NUMBER,
  5    금액     NUMBER);
```

```
INSERT INTO pt_01 VALUES ( '20150101','1000', 10, 500 );
INSERT INTO pt_01 VALUES ( '20150102','1001', 10, 400 );
INSERT INTO pt_01 VALUES ( '20150103','1002', 10, 300 );

INSERT INTO pt_02 VALUES ( '20150201','1003', 5, 500 );
INSERT INTO pt_02 VALUES ( '20150202','1004', 5, 400 );
INSERT INTO pt_02 VALUES ( '20150203','1005', 5, 300 );
commit;
```

❑ 7) MERGE

```
SQL> MERGE INTO  p_total total
  2  USING pt_01 p01
  3  ON ( total.판매번호 = p01.판매번호)
  4  WHEN  MATCHED THEN
  5    UPDATE SET  total.제품번호 = p01.제품번호
  6  WHEN NOT MATCHED THEN
  7    INSERT VALUES ( p01.판매번호,p01.제품번호,p01.수량, p01.금액);
```

```
SQL> MERGE INTO  p_total total
  2  USING pt_02 p02
  3  ON ( total.판매번호 = p02.판매번호)
  4  WHEN  MATCHED THEN
  5    UPDATE SET  total.제품번호 = p02.제품번호
  6  WHEN NOT MATCHED THEN
  7    INSERT VALUES ( p02.판매번호,p02.제품번호,p02.수량, p02.금액);
```

COMMIT;

```
SQL> SELECT * FROM p_total;

판매번호   제품번호         수량         금액
-------- ---------- ---------- ----------
20150102       1001         10        400
20150101       1000         10        500
20150103       1002         10        300
20150203       1005          5        300
20150202       1004          5        400
20150201       1003          5        500
```

❑ 7) MERGE – 조건 추가 및 DELETE 추가

- 표준 MERGE에 조건구문을 추가 가능하다.
- MERGE.. UPDATE문에 DELETE 구문을 포함 시킬 수 있다.

```
SQL> CREATE TABLE EMP_M2
  2  AS
  3  SELECT EMPNO,JOB,SAL
  4  FROM EMP
  5  WHERE 1=2;

SQL> MERGE INTO emp_m2  m2
  2  USING emp b
  3  ON ( m2.empno = b.empno )
  4  WHEN MATCHED THEN
  5    UPDATE SET
  6    m2.job = b.job,
  7    m2.sal = b.sal
  8    WHERE  b.job='CLERK'
  9  WHEN NOT MATCHED THEN
 10    INSERT ( m2.empno, m2.job, m2.sal )
 11    VALUES ( b.empno, b.job, b.sal )
 12      WHERE  b.job='CLERK';
SQL> SELECT * FROM EMP_M2;

     EMPNO JOB                 SAL
---------- ----------------- ----------
      7369 CLERK                    800
      7876 CLERK                   1100
      7900 CLERK                    950
      7934 CLERK                   1300
```

```
SQL> CREATE TABLE emp_m3
  2  as
  3  SELECT empno, job, sal
  4  FROM emp;

SQL> MERGE INTO emp_m3 m3
  2  USING emp b
  3  ON ( m3.empno = b.empno )
  4  WHEN MATCHED THEN
  5    UPDATE SET
  6    m3.job = b.job,
  7    m3.sal = b.sal + 1000
  8    DELETE WHERE ( m3.job = 'CLERK' )
  9  WHEN NOT MATCHED THEN
 10    INSERT ( m3.empno , m3.job, m3.sal )
 11    VALUES ( b.empno , b.job, b.sal+500 );

SQL> SELECT * FROM EMP_M3;

     EMPNO JOB                 SAL
---------- ----------------- ----------
      7499 SALESMAN                2600
      7521 SALESMAN                2250
      7566 MANAGER                 3975
      7654 SALESMAN                2250
      7698 MANAGER                 3850
      7782 MANAGER                 3450
      7788 ANALYST                 4000
      7839 PRESIDENT               6000
      7844 SALESMAN                2500
      7902 ANALYST                 4000
```

- Transaction 정의

  -트랜잭션은 데이터베이스의 **논리적인 단위**이다.
  -트랜잭션은 밀접히 관련되어 분리될 수 없는 한 개 이상의 데이터베이스 조작을 가리킨다.
  -하나의 트랜잭션에는 하나 이상의 SQL문장이 포함되며, 분할 할 수 없는 최소의 단위이다. 그렇기 때문에 전부 적용하거나 전부 취소된다.
  즉, 트랜잭션은 'All or Nothing'이다.

  -트랜잭션의 대상이 되는 SQL문은 DML문이다. SELECT문장은 직접적인 트랜잭션의 대상이 아니지만, SELECT FOR UPDATE등 배타적 lock을 요구하는 SELECT문장은 트랜잭션의 대상이 될 수 있다.

  예> select * from emp FOR UPDATE;

- Transaction 시작 및 종료

1. Transaction 시작

: DML 명령을 시작한 경우에 Transaction 는 자동으로 시작된다.

2. Transaction 종료 (명시적/무시적)

: 사용자가 COMMIT 또는 ROLLBACK 명령을 명시적으로 실행한 경우.
: DDL문장을 실행한 경우
: DCL 문장을 실행한 경우
: 사용자가 SQL*PLUS 또는 iSQL*PLUS 를 종료한 경우
: 하드웨어 고장 또는 시스템 오류시.

❑ 8) 트랜잭션 (transaction)

- 명시적인 COMMIT / ROLLBACK 장점

  : 데이터의 일관성을 보장해 준다.
  : 데이터의 변경사항을 데이터베이스에 영구히 반영하기 전에 데이터 변경사항을 미리 볼 수 있다.
  : 논리적으로 연관된 작업을 그룹화 할 수 있다.

- Transaction 종료 전의 데이터 상태

  : 트랜잭션내의 모든 데이터 변경 사항은 트랜잭션이 종료되기 전까지는 모두 임시적이다. 따라서 데이터의 변경 전 데이터로 복구 될 수 있다.
  : 변경된 행은 Lock(잠금)이 걸리며, 해당 사용자를 제외한 나머지 사용자는 해당 행들을 변경할 수 없다.
  : 현재 사용자는 SELECT 문장을 이용하여 데이터 변경 후의 결과를 확인 할 수 있다.
  : 다른 사용자는 현재 사용자에 의해 변경된 데이터 결과를 확인할 수 없다.
  (데이터 일관성)

- COMMIT

: 모든 데이터 변경사항을 데이터베이스에 영구히 반영시키는 명령어.
: 변경전의 데이터는 모든 잃게 된다.
: 모든 사용자들이 트랜잭션 종료 후의 결과를 확인 할 수 있다.
: 트랜잭션이 진행 중이었던 행들에 대한 잠금이 모두 해소되며, 다른 사용자에 의해서 변경이 가능해진다.

```
SQL> SELECT * FROM DEPT;

    DEPTNO DNAME      LOC
---------- ---------- ----------
        90 경리과     부산
        92 인사과
        10 ACCOUNTING NEW YORK
        20 RESEARCH   DALLAS
        30 SALES      CHICAGO
        40 OPERATIONS BOSTON

6 개의 행이 선택되었습니다.
```

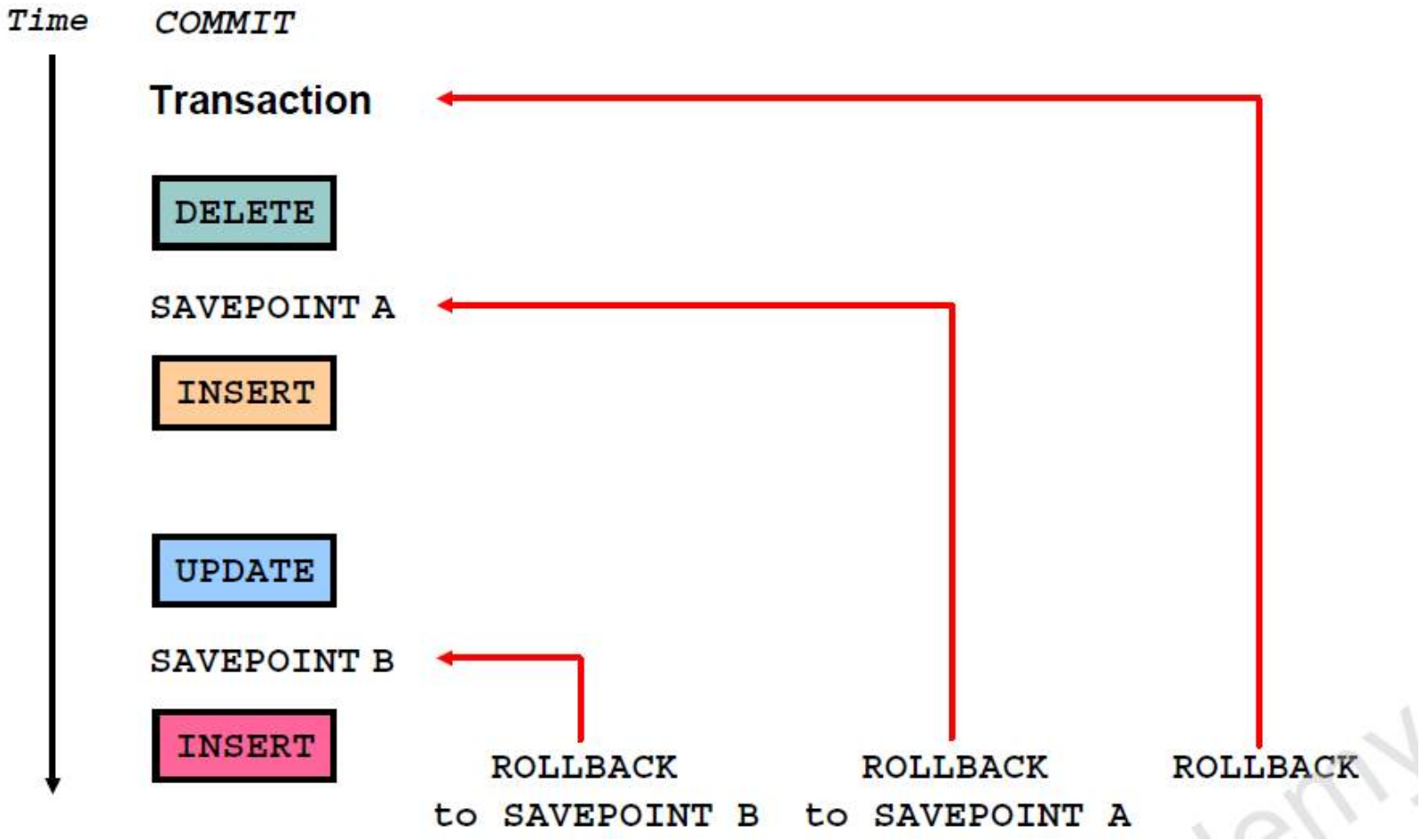
```
SQL> DELETE FROM DEPT
  2  WHERE DEPTNO = 92;

1 행이 삭제되었습니다.

SQL> COMMIT;

커밋이 완료되었습니다.
```

- ROLLBACK

: 모든 데이터 변경사항을 취소하는 명령어.
: 변경전의 데이터가 복원된다.
: 모든 사용자들이 트랜잭션 종료 후의 결과를 확인 할 수 있다.
: 트랜잭션이 진행 중이었던 행들에 대한 잠금이 모두 해소되며, 다른 사용자에 의해서 변경이 가능해진다.

```
SQL> DELETE FROM EMP;

14 행이 삭제되었습니다.

SQL> ROLLBACK;

롤백이 완료되었습니다.
```

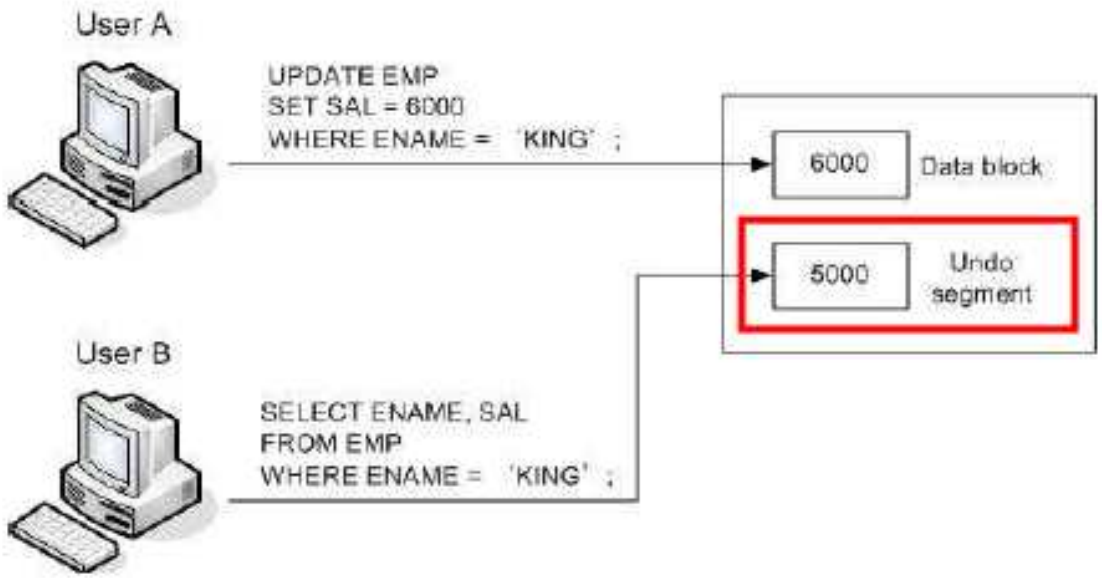
- SAVEPOINT


Time
COMMIT
Transaction
DELETE
SAVEPOINT A
INSERT
UPDATE
SAVEPOINT B
INSERT
ROLLBACK
to SAVEPOINT B
ROLLBACK
to SAVEPOINT A
ROLLBACK

- 읽기 일관성 (Read Consistency)

  : **사용자들에게 가장 최근에 커밋된 데이터를 보여주는 것.**
  : 데이터를 검색하는 사용자들과 변경하는 사용자들 사이에 일관적인 관점을 제공한다.
  즉, 다른 사용자들이 변경중인 데이터를 볼 수 없게 한다.
  : 일관적인 데이터베이스 변경방법을 제공함으로써 동일한 데이터를 동시에 변경할 때 발생할 수 있는 혼란을 방지한다.

- 읽기 일관성 구현 원리



- LOCK 경합

① SQL> SELECT * FROM DEPT;

DEPTNO	DNAME	LOC
90	경리과	부산
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

② SQL> UPDATE DEPT
2 SET DNAME ='인사과'
3 WHERE DEPTNO = 90;

1 행이 갱신되었습니다.

③ SQL> SELECT * FROM DEPT;

DEPTNO	DNAME	LOC
90	인사과	부산
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

⑥ SQL> ROLLBACK;

롤백이 완료되었습니다.

④ SQL> SELECT * FROM DEPT;

DEPTNO	DNAME	LOC
90	경리과	부산
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

⑤ SQL> UPDATE DEPT
2 SET LOC = '서울'
3 WHERE DEPTNO = 90; Wait!!

⑦ 1 행이 갱신되었습니다.

Thank you