데이터 조작어 (DML)

데이터 조작어 (DML)

1. 데이터 조작어(DML) 개요

- SQL의 핵심요소
- 다음 경우에 DML 문 실행
  - 테이블에 새로운 행을 추가 (INSERT)
  - 테이블로부터 기존의 행을 제거 (DELETE)
  - 테이블에 있는 기존의 행을 변경 (UPDATE)
- 트랜잭션(Transaction)은 논리 작업 단위를 형성하는 DML문의 집합

데이터 조작어 (DML)

2. 데이터 삽입 (INSERT)

2.1 데이터 삽입 개요

50	TESTING	SEOUL

새로운 행

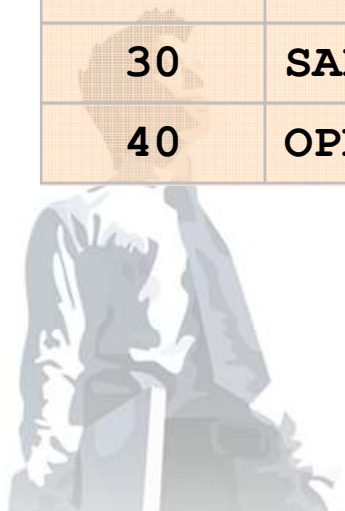
DEPT

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

새로운 행을
DEPT 테이블에 추가

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	TESTING	SEOUL



데이터 조작어 (DML)

2.2 INSERT 구문

- **INSERT** 문을 사용하여 테이블에 새 행을 추가
- 이 구문형식으로는 한번에 오직 하나의 행만이 추가
- 각각의 열에 대한 값을 포함하는 새로운 행을 삽입
- 테이블 열의 기본 순서대로 값을 나열
- **INSERT** 절에서 열을 선택적으로 나열
- 문자와 날짜 값은 작은 따옴표(' ')로 묶음

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

데이터 조작어 (DML)

2.3 INSERT 활용

```
SQL> INSERT INTO dept (deptno, dname, loc)
  2  VALUES (50, 'TESTING', 'SEOUL');

1 개의 행이 만들어졌습니다.
```

```
SQL> SELECT *
  2  FROM   dept;

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

데이터 조작어 (DML)

2.3 INSERT 활용 (계속)

- 널(Null) 값을 가진 새로운 행 삽입

방 법	설 명
암시적	- 열 목록에서 열을 생략
명시적	- VALUES 목록에서 NULL 키워드를 지정 - 문자열 또는 날짜인 경우에만 VALUES 목록에 빈 문자열 (' ') 지정

데이터 조작어 (DML)

2.3 INSERT 활용 (계속)

- 널(Null) 값을 가진 새로운 행 삽입 (계속)

```
SQL> INSERT INTO dept (deptno, dname)
  2  VALUES       (60, 'TESTING01');

1 개의 행이 만들어졌습니다.
```

```
SQL> INSERT INTO dept
  2  VALUES       (70, 'TESTING02', NULL);

1 개의 행이 만들어졌습니다.
```

```
SQL> INSERT INTO dept
  2  VALUES       (80, ' ', ' ');

1 개의 행이 만들어졌습니다.
```

데이터 조작어 (DML)

2.3 INSERT 활용 (계속)

- 함수를 사용한 값 입력

```
SQL> INSERT INTO emp (empno, ename, hiredate, sal,
  2                                          comm, deptno)
  3  VALUES       (1111, USER, SYSDATE, 3000, NULL, 20);

1 개의 행이 만들어졌습니다.
```

- 특정 날짜 값 입력

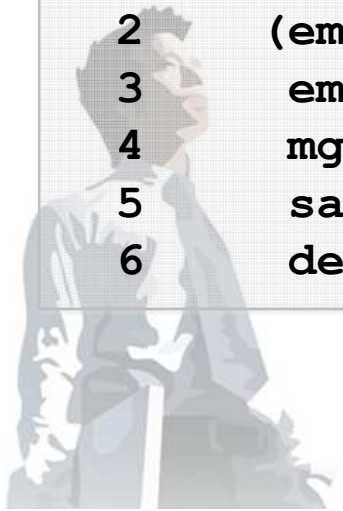
```
SQL> INSERT INTO emp (empno, ename, hiredate)
  2  VALUES       (3333, 'LEE',
  3                TO_DATE('5월/05 2010', 'MON/DD YYYY'));

1 개의 행이 만들어졌습니다.
```

데이터 조작어 (DML)

2.3 INSERT 활용 (계속)

- 다른 테이블에서 행 복사
  - 서브쿼리로 **INSERT** 문 작성
  - **VALUES** 절 사용하지 않음
  - 서브쿼리의 열 수와 **INSERT** 절의 열 수 일치

```
SQL> CREATE TABLE s_emp
  2    (empid      NUMBER(5),
  3     empname    VARCHAR2(10),
  4     mgr        NUMBER(5),
  5     sal        NUMBER(7,2),
  6     deptid     NUMBER(2));
```



데이터 조작어 (DML)

2.3 INSERT 활용 (계속)

- 다른 테이블에서 행 복사 (계속)

```
SQL> INSERT INTO s_emp
  2     SELECT empno, ename, mgr, sal, deptno
  3     FROM   emp;

16 개의 행이 만들어졌습니다.
```

```
SQL> COMMIT;
```

데이터 조작어 (DML)

3. 데이터 삭제 (DELETE)

3.1 데이터 삭제 개요

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	TESTING	SEOUL
•••		

DEPT 테이블로부터 행 삭제

DEPT

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

데이터 조작어 (DML)

3.2 DELETE 구문

- **WHERE** 절을 지정하여 특정 행 삭제
- **WHERE** 절을 생략하면 테이블의 모든 행이 삭제

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

데이터 조작어 (DML)

3.3 DELETE 활용

- WHERE 절 있는 DELETE 문

```
SQL> DELETE dept
  2  WHERE  deptno = 50;

1 행이 삭제되었습니다.
```

- WHERE 절 없는 DELETE 문

```
SQL> DELETE dept;

7 행이 삭제되었습니다.
```

```
SQL> ROLLBACK;
```

데이터 조작어 (DML)

3.3 DELETE 활용 (계속)

- 다른 테이블을 기반으로 행 삭제
  - **DELETE** 문에 서브쿼리를 사용하여 테이블에서 다른 테이블의 값을 기반으로 하는 행을 제거

```
SQL> DELETE emp
  2  WHERE  deptno = (SELECT deptno
  3                   FROM   dept
  4                   WHERE  dname = 'TESTING02');

1 행이 삭제되었습니다.
```

```
SQL> ROLLBACK;
```

데이터 조작어 (DML)

3.3 DELETE 활용 (계속)

- 무결성 제약조건 오류
  - 다른 테이블에서 **FOREIGN KEY**로 사용되는 **PRIMARY KEY**를 포함하는 행은 삭제할 수 없음

```
SQL> DELETE dept
  2  WHERE  deptno = 10;
```

```
DELETE dept
*
1행에 오류:
ORA-02292: 무결성 제약조건(SCOTT.SYS_C004426)이
           위배되었습니다- 자식 레코드가 발견되었습니다
```

데이터 조작어 (DML)

4. 데이터 변경 (UPDATE)

4.1 데이터 변경 개요

EMP

EMPNO	ENAME	JOB	•••	DEPTNO
7839	KING	PRESIDENT		10
7566	JONES	MANAGER		20
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		10
•••				

EMP 테이블의 행을 갱신

EMP

EMPNO	ENAME	JOB	•••	DEPTNO
7839	KING	PRESIDENT		10
7566	JONES	MANAGER		20
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		30
•••				

데이터 조작어 (DML)

4.2 UPDATE 구문

- 필요한 경우 한 번에 여러 행을 갱신할 수 있음
- **WHERE** 절을 지정하여 특정 행 수정
- **WHERE** 절을 생략하면 테이블의 모든 행이 수정

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

데이터 조작어 (DML)

4.3 UPDATE 활용

- WHERE 절 있는 UPDATE 문

```
SQL> UPDATE emp
  2  SET    deptno = 30
  3  WHERE  deptno = 20;
```

- WHERE 절 없는 UPDATE 문

```
SQL> UPDATE emp
  2  SET    deptno = 30;
```

```
SQL> ROLLBACK;
```

데이터 조작어 (DML)

4.3 UPDATE 활용 (계속)

- 다른 테이블을 기반으로 행 갱신

```
SQL> UPDATE emp
  2  SET    job    = (SELECT job
  3                   FROM   emp
  4                   WHERE  ename = 'SMITH')
  5  WHERE  deptno = (SELECT deptno
  6                   FROM   dept
  7                   WHERE  dname = 'SALES');

4 행이 갱신되었습니다.
```

```
SQL> ROLLBACK;
```

데이터 조작어 (DML)

4.3 UPDATE 활용 (계속)

- 무결성 제약조건 오류

```
SQL> UPDATE emp
  2  SET    deptno = 22
  3  WHERE  deptno = 10;
```

```
UPDATE emp
*
1행에 오류:
ORA-02291: 무결성 제약조건(SCOTT.SYS_C004426)이
           위배되었습니다- 부모 키가 없습니다
```

트랜잭션 제어

트랜잭션 제어

1. 데이터베이스 트랜잭션 개요

- 트랜잭션은 데이터 변경시 융통성 및 제어 기능을 제공
- 트랜잭션은 사용자 프로세스가 중단되거나 시스템 장애가 발생한 경우 데이터 일관성을 보장
- 트랜잭션은 다음 명령문 중 하나로 구성
  - 데이터를 일관성 있게 변경하는 **DML** 문
  - 하나의 **DDL** 문
  - 하나의 **DCL** 문

트랜잭션 제어

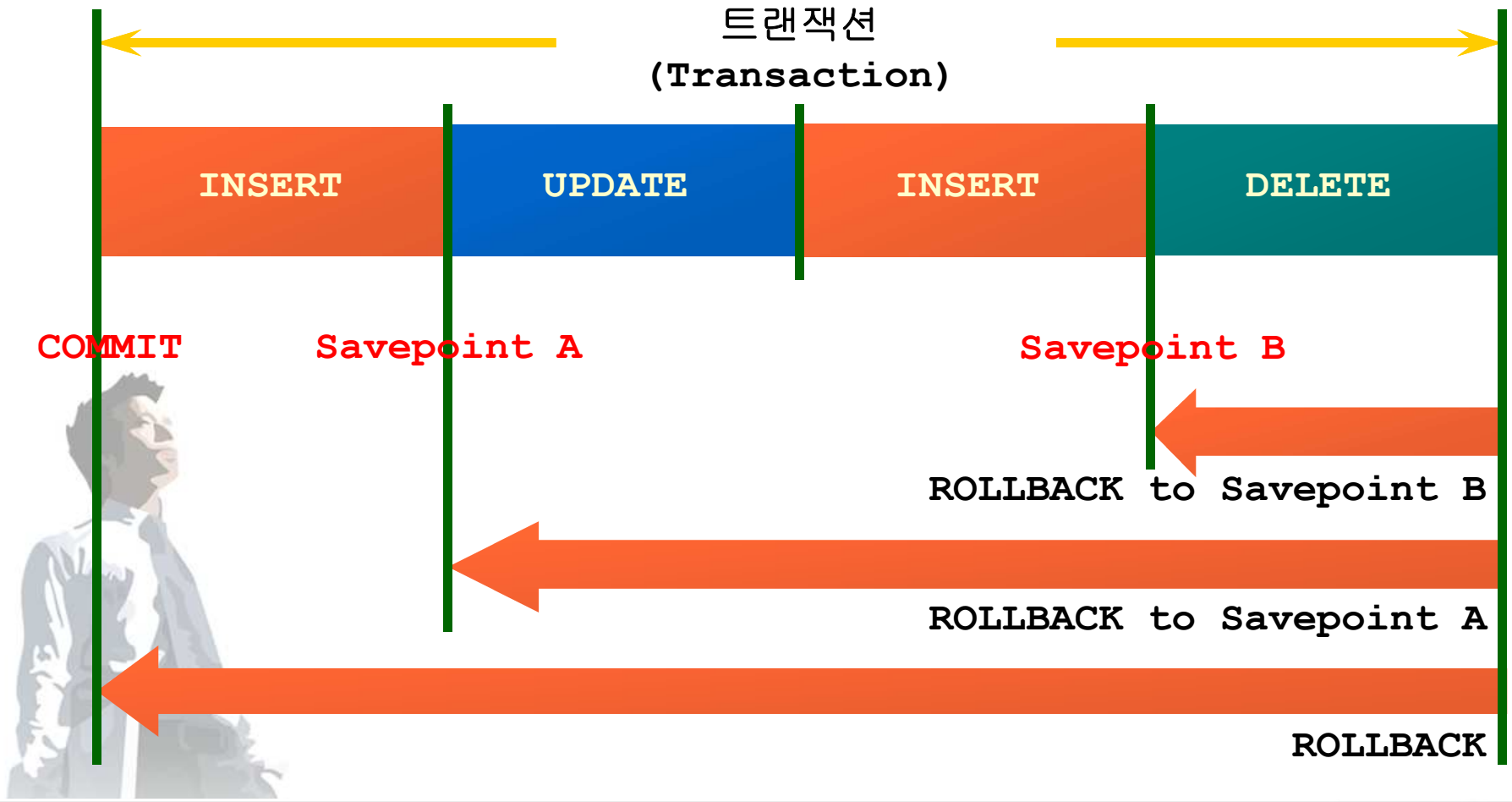
1. 데이터베이스 트랜잭션 개요 (계속)

- 트랜잭션은 실행 가능한 첫번째 **SQL** 문이 실행되면 시작
- 트랜잭션은 다음 이벤트가 발생하면 종료
  - **COMMIT** 또는 **ROLLBACK** 실행
  - **DDL** 또는 **DCL** 문장 실행 **(**자동 커밋**)**
  - 사용자 종료 **(SQL*Plus** 종료**)**
  - 시스템 장애가 있거나 시스템이 고장인 경우 **(crash)**

트랜잭션 제어

2. 트랜잭션 제어 (Transaction Control)

2.1 트랜잭션 제어 개요



트랜잭션 제어

2.2 명시적 트랜잭션 제어

- **COMMIT** 및 **ROLLBACK** 문으로 다음을 수행
  - 데이터 일관성 보장
  - 데이터 변경 내용을 영구히 저장하기 전에 미리 볼 수 있음
  - 논리적으로 관련된 작업을 묶음

명령문	설 명
COMMIT	- 보류 중인 데이터 변경 사항을 모두 커밋(**COMMIT**) 하여 현재 트랜잭션(**Transaction**)을 종료
SAVEPOINT ***name***	- 현재 트랜잭션 내에 저장점(**SAVEPOINT**)을 표시
ROLLBACK [TO SAVEPOINT ***name*****]**	- **ROLLBACK**은 보류 중인 데이터 변경사항을 모두 되돌려서 현재 트랜잭션을 종료 - **ROLLBACK TO SAVEPOINT**는 현재 트랜잭션을 지정된 저장점(**SAVEPOINT**)으로 롤백(**ROLLBACK**)하여 저장점 이후의 변경 내용을 취소

트랜잭션 제어

2.3 암시적 트랜잭션 제어

상 태	설 명
자동 커밋 **(Automatic Commit)**	- **DDL** 문이 실행된 경우 - **DCL** 문이 실행된 경우 - 명시적인 **COMMIT** 또는 **ROLLBACK**이 실행되지 않은 채 **SQL*Plus**에서 정상 종료한 경우
자동 롤백 **(Automatic Rollback)**	- **SQL*Plus**의 비정상 종료 시 또는 시스템 장애가 발생한 경우

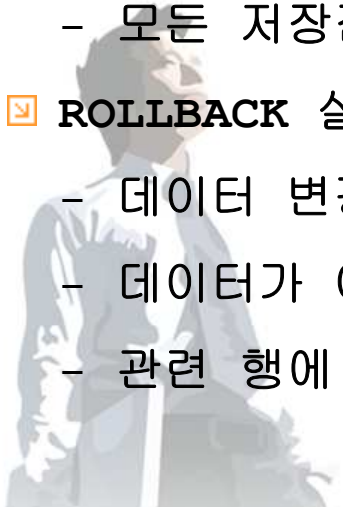
2.4 트랜잭션 제어 활용

- **COMMIT** 또는 **ROLLBACK** 실행 이전 데이터 상태
  - 데이터의 이전 상태로 복구 가능
  - 현재 사용자는 **SELECT** 문을 사용하여 **DML** 작업 결과 검토 가능
  - 현재 사용자가 사용 중인 **DML** 문의 결과를 다른 사용자가 볼 수 없음
  - 관련 행이 잠겨 있으므로 다른 사용자는 관련행의 데이터를 변경할 수 없음

트랜잭션 제어

2.4 트랜잭션 제어 활용 (계속)

- **COMMIT** 실행 이후 데이터 상태
  - 데이터 변경 내용이 데이터베이스에 영구히 저장
  - 데이터의 이전 상태는 완전히 상실
  - 모든 사용자가 결과를 볼 수 있음
  - 관련 행 잠금이 해제되어 다른 사용자가 행을 조작할 수 있음
  - 모든 저장점(**SAVEPOINT**)이 제거
- **ROLLBACK** 실행 이후 데이터 상태
  - 데이터 변경이 취소
  - 데이터가 이전 상태로 복구
  - 관련 행에 대한 잠금 해제



트랜잭션 제어

2.4 트랜잭션 제어 활용 (계속)

```
SQL> DELETE s_emp
  2  WHERE  empid = 1111;

1 행이 삭제되었습니다.
```

```
SQL> COMMIT;

커밋이 완료되었습니다.
```

```
SQL> DELETE s_emp;
```

```
SQL> ROLLBACK;

롤백이 완료되었습니다.
```

트랜잭션 제어

2.4 트랜잭션 제어 활용 (계속)

```
SQL> @ C:\skjdb.sql

SQL> DELETE emp
  2  WHERE  deptno = 10;

SQL> SAVEPOINT sp1;

저장점이 생성되었습니다.

SQL> DELETE emp
  2  WHERE  deptno = 20;

SQL> ROLLBACK TO sp1;

롤백이 완료되었습니다.

SQL> COMMIT;

커밋이 완료되었습니다.
```

2.4 트랜잭션 제어 활용 (계속)

- 명령문 레벨 롤백
  - 트랜잭션 실행 중 단일 **DML** 문 실행에 실패하면 해당 명령문만 롤백
  - **Oracle Server**는 암시적 저장점**(Savepoint)**을 구현
  - 다른 모든 변경 내용은 유지
  - **DDL** 문의 경우 성공적으로 실행되지 않아도 **DDL** 문 전후에서 암시적 커밋**(Commit)**이 실행되므로 명령문은 롤백**(Rollback)** 할 수 없음
  - 사용자는 **COMMIT** 또는 **ROLLBACK** 문을 실행하여 트랜잭션을 명시적으로 종료해야 함