Ch 6.

DML(Data Manipulation Language)

6.1 | UPDATE

6.2 | INSERT

6.3 | DELETE

6.4 | 트랜잭션Transaction

6.5 | 잠금Lock

Objective

- ❖ 테이블에 포함된 행을 갱신할 수 있다
- ❖ 테이블에 새로운 행을 삽입할 수 있다
- ❖ 테이블에 포함된 행을 삭제할 수 있다
- ❖ Commit, Rollback 개념을 설명할 수 있다
- ❖ Transaction 개념을 설명할 수 있다
- ❖ Lock 개념을 설명할 수 있다

Ch 6.

DML(Data Manipulation Language)

6.1.1 의미

- 테이블에 포함된 기존 데이터를 수정
- 전체 데이터 건 수(행 수)는 달라지지 않음

기존 데이터

DEPT_ID	DEPT_NAME	LOC_ID
20	회계팀	A1
10	본사 인사팀	A1
50	해외영업1팀	U1
60	기술지원팀	OT
80	해외영업2팀	A2
90	해외영업3팀	A3
30	마케팅팀	A1

갱신 데이터

DEPT_ID	DEPT_NAME	LOC_ID
20	회계팀	A1
10	인사팀	A1
50	해외영업1팀	U1
60	기술지원팀	OT
80	해외영업2팀	A2
90	해외영업3팀	A3
30	홍보팀	A1

6.1.2 데이터 갱신 구문

조건에 맞는 행(또는 여러 행)의 컬럼 값을 갱신할 수 있음

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

[구문 설명]

- WHERE 절이 생략되면 전체 행이 갱신
- SET 절
  - 서브쿼리 사용 가능
  - DEFAULT 옵션 사용 가능

6.1.3 UPDATE 사용

```
UPDATE DEPARTMENT
SET    DEPT_NAME = '전략기획팀'
WHERE  DEPT_ID = '90';

SELECT * FROM DEPARTMENT;
```

DEPT_ID	DEPT_NAME	LOC_ID
20	회계팀	A1
10	본사 인사팀	A1
50	해외영업1팀	U1
60	기술지원팀	OT
80	해외영업2팀	A2
90	전략기획팀	A3
30	마케팅팀	A1

```
UPDATE DEPARTMENT
SET    DEPT_NAME = '전략기획팀';

SELECT * FROM DEPARTMENT;
```

WHERE 절을 생략하면 전체 행이 변경됨

DEPT_ID	DEPT_NAME	LOC_ID
20	전략기획팀	A1
10	전략기획팀	A1
50	전략기획팀	U1
60	전략기획팀	OT
80	전략기획팀	A2
90	전략기획팀	A3
30	전략기획팀	A1

6.1.3 UPDATE 사용

```
UPDATE EMPLOYEE
SET    JOB_ID = (SELECT JOB_ID
                 FROM   EMPLOYEE
                 WHERE  EMP_NAME = '성해교'),
       SALARY = (SELECT SALARY
                 FROM   EMPLOYEE
                 WHERE  EMP_NAME = '성해교')
WHERE  EMP_NAME = '심하균';
```

또는

```
UPDATE EMPLOYEE
SET    (JOB_ID, SALARY) = (SELECT JOB_ID, SALARY
                           FROM   EMPLOYEE
                           WHERE  EMP_NAME = '성해교')
WHERE  EMP_NAME = '심하균';
```

EMP_NAME	JOB_ID	SALARY
성해교	J7	1900000
심하균		2300000

EMP_NAME	JOB_ID	SALARY
성해교	J7	1900000
심하균	J7	1900000

- SET 절에 서브쿼리를 사용할 수 있음
- SET 절에서 여러 개 컬럼을 갱신할 수 있음

6.1.3 UPDATE 사용

```
UPDATE EMPLOYEE
SET    MARRIAGE = DEFAULT
WHERE  EMP_ID = '210';
```

EMP_NAME	MARRIAGE
감우섭	Y

EMP_NAME	MARRIAGE
감우섭	N

EMPLOYEE 테이블

Name	Type	Nullable	Default
EMP_ID	CHAR(3)		
EMP_NAME	VARCHAR2(20)		
EMP_NO	CHAR(14)		
EMAIL	VARCHAR2(25)	Y	
PHONE	VARCHAR2(12)	Y	
HIRE_DATE	DATE	Y	SYSDATE
JOB_ID	CHAR(2)	Y	
SALARY	NUMBER	Y	
BONUS_PCT	NUMBER	Y	
MARRIAGE	CHAR(1)	Y	'N'
MGR_ID	CHAR(3)	Y	
DEPT_ID	CHAR(2)	Y	

- 해당 컬럼에 DEFAULT 값이 설정된 경우 사용 가능
- DEFAULT가 설정되지 않은 경우에는 NULL이 적용 됨

6.1.3 UPDATE 사용

```
UPDATE EMPLOYEE
SET    BONUS_PCT = 0.3
WHERE  DEPT_ID = (SELECT DEPT_ID
                  FROM   DEPARTMENT
                  WHERE  DEPT_NAME = '해외영업2팀');
```

WHERE 조건에 서브쿼리 사용 가능

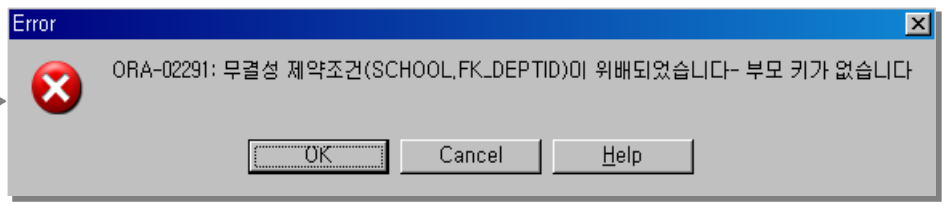
EMP_NAME	BONUS_PCT
전우성	
엄정하	0.2

EMP_NAME	BONUS_PCT
전우성	0.3
엄정하	0.3

6.1.4 UPDATE와 무결성 제약조건

데이터 무결성이 손상되는 변경 작업은 허용되지 않음

```
UPDATE EMPLOYEE
SET    DEPT_ID = '65'
WHERE  DEPT_ID IS NULL;
```



EMPLOYEE 테이블의 제약조건

이름	유형	컬럼	참조
NN_ENAME	C	EMP_NAME	
NN_EMPNO	C	EMP_NO	
CHK_MRIG	C	MARRIAGE	
PK_EMPID	P	EMP_ID	
UNI_EMPNO	U	EMP_NO	
FK_DEPTID	R	DEPT_ID	PK_DEPTID
FK_JOBID	R	JOB_ID	PK_JOBID
FK_MGRID	R	MGR_ID	PK_EMPID

DEPARTMENT 테이블

DEPT_ID
10
20
30
50
60
80
90

Ch 6.

DML(Data Manipulation Language)

6.2.1 의미

테이블에 데이터(새로운 행)를 추가

```
CREATE TABLE DEPT
( DEPT_ID   CHAR(2) ,
  DEPT_NAME VARCHAR2(30) );
SELECT COUNT(*) FROM DEPT;
INSERT INTO DEPT VALUES ('20', '회계팀');
SELECT COUNT(*) FROM DEPT;
INSERT INTO DEPT VALUES ('10', '인사팀');
SELECT COUNT(*) FROM DEPT;
```

COUNT(*)
0

COUNT(*)
1

COUNT(*)
2

6.2.2 구문

```
INSERT INTO table_name [ ( column_name [, column_name ...] ) ]
VALUES ( value1 [, value 2 ...] );
```

[구문 설명]

VALUE 절의 입력 값과 INSERT INTO 절의 컬럼 관계

- 데이터 타입 일치
- 순서 일치
- 개수 일치

6.2.3 INSERT 사용

```
INSERT INTO EMPLOYEE ( EMP_ID, EMP_NO, EMP_NAME, EMAIL,
                       PHONE, HIRE_DATE, JOB_ID, SALARY, BONUS_PCT, MARRIAGE,
                       MGR_ID, DEPT_ID )
VALUES               ( '900', '811126-1484710', '오윤하', 'oyuh@vcc.com',
                       '01012345678', '06/01/01', 'J7', 3000000, 0, 'N',
                       '176', '90' );
```

```
SELECT * FROM EMPLOYEE
WHERE  EMP_ID = '900';
```

EMP_ID	EMP_NAME	EMP_NO	EMAIL	PHONE	HIRE_DATE	JOB_ID	SALARY	BONUS_PCT	MARRIAGE	MGR_ID	DEPT_ID
900	오윤하	811126-1484710	oyuh@vcc.com	01012345678	06/01/01	J7	3000000	0	N	176	90

6.2.3 INSERT 사용

- 모든 컬럼에 값을 입력하는 경우 컬럼 이름 생략 가능
- 컬럼 생성 순서대로 입력 값을 기술해야 함

```
INSERT INTO EMPLOYEE VALUES
('910', '이병언', '781010-1443269', 'TK1@VCC.COM', '01077886655', '04/01/01',
 'J7', 3500000, 0.1, 'N','176', '90');
```

Name	Type
EMP_ID	CHAR(3)
EMP_NAME	VARCHAR2(20)
EMP_NO	CHAR(14)
EMAIL	VARCHAR2(25)
PHONE	VARCHAR2(12)
HIRE_DATE	DATE
JOB_ID	CHAR(2)
SALARY	NUMBER
BONUS_PCT	NUMBER
MARRIAGE	CHAR(1)
MGR_ID	CHAR(3)
DEPT_ID	CHAR(2)

테이블을 생성할 때 기술된
컬럼 순서대로 입력 값 기술

6.2.3 INSERT 사용

NULL 입력

- 암시적 방법 : INSERT INTO 절에서 해당 컬럼 이름 생략
- 명시적 방법 : VALUES 절에서 **NULL** 키워드나 ''Empty String 사용

```
INSERT INTO EMPLOYEE (EMP_ID, EMP_NO, EMP_NAME, EMAIL, PHONE, HIRE_DATE,
                      JOB_ID, SALARY, BONUS_PCT, MARRIAGE, MGR_ID, DEPT_ID)
VALUES               ('880', '860412-2377610', '한채연', '0193382662',
                      '06/01/01', 'J7', 3000000, 0, 'N');
```

참고INSERT INTO 절에서 흐리게 표현한 컬럼은 생략된 컬럼을 의미

EMP_ID	EMP_NAME	EMP_NO	EMAIL	PHONE	HIRE_DATE	JOB_ID	SALARY	BONUS_PCT	MARRIAGE	MGR_ID	DEPT_ID
880	한채연	860412-2377610		0193382662	06/01/01	J7	3000000	0	N		

6.2.3 INSERT 사용

NULL 입력

- 암시적 방법 : INSERT INTO 절에서 해당 컬럼 이름 생략
- 명시적 방법 : VALUES 절에서 **NULL** 키워드나 ''Empty String 사용

```
INSERT INTO EMPLOYEE VALUES
('840', '하지언', '870115-2253408', 'ju_ha@vcc.com', NULL, '07/06/15',
 'J7', NULL, NULL, 'N', '', '');
```

EMP_ID	EMP_NAME	EMP_NO	EMAIL	PHONE	HIRE_DATE	JOB_ID	SALARY	BONUS_PCT	MARRIAGE	MGR_ID	DEPT_ID
840	하지언	870115-2253408	ju_ha@vcc.com		07/06/15	J7			N		

6.2.3 INSERT 사용

DEFAULT

- DEFAULT 설정된 컬럼 이름을 생략하면 DEFAULT 값이 입력됨
- DEFAULT 키워드를 기술하여 명시적으로 표현(ANSI 표준 권고)
- DEFAULT 값이 설정되지 않은 경우 DEFAULT 키워드를 사용해도 NULL 입력

```
INSERT INTO EMPLOYEE (EMP_ID, EMP_NO, EMP_NAME, SALARY, MARRIAGE)
VALUES                ('860', '810429-2165344', '선예진', DEFAULT, DEFAULT);
```

EMP_ID	EMP_NAME	EMP_NO	EMAIL	PHONE	HIRE_DATE	JOB_ID	SALARY	BONUS_PCT	MARRIAGE	MGR_ID	DEPT_ID
860	선예진	810429-2165344			09/12/29				N		

- SALARY 컬럼에 DEFAULT 키워드를 사용했으나 설정된 값이 없으므로 NULL 입력됨
- MARRIAGE 컬럼에 DEFAULT 키워드를 사용했으므로 'N' 입력됨
- EMAIL, PHONE, HIRE_DATE, JOB_ID, BONUS_PCT, MGR_ID, DEPT_ID 컬럼은 생략되었으므로 NULL이 입력되어야 하지만, HIRE_DATE 컬럼에는 DEFAULT 값이 설정되어 있으므로 DEFAULT 값이 입력됨

6.2.3 INSERT 사용

```
INSERT INTO table_name [ column_name ... ]
Subquery ;
```

```
CREATE TABLE EMP
(EMP_ID    CHAR(3),
 EMP_NAME  VARCHAR2(20),
 DEPT_NAME VARCHAR2(20));
```

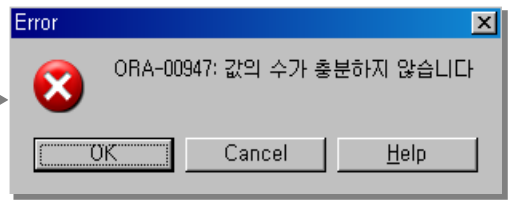
```
INSERT INTO EMP
( SELECT EMP_ID, EMP_NAME, DEPT_NAME
  FROM   EMPLOYEE
  LEFT OUTER JOIN DEPARTMENT USING (DEPT_ID) );

SELECT * FROM EMP;
```

- VALUE 키워드 대신 서브쿼리를 사용할 수 있음
- 서브쿼리 부분의 ()는 생략 가능

EMP_ID	EMP_NAME	DEPT_NAME
210	감우섭	회계팀
208	이중기	회계팀
207	김술오	회계팀
205	임영애	본사 인사팀
202	권상후	본사 인사팀
200	고승우	본사 인사팀
201	박하일	해외영업1팀
149	성해교	해외영업1팀
144	김순이	해외영업1팀
143	나승원	해외영업1팀
141	김예수	해외영업1팀
124	정지현	해외영업1팀
107	조재형	기술지원팀
104	안석규	기술지원팀
103	정도연	기술지원팀
176	엄정하	해외영업2팀
174	전우성	해외영업2팀
102	최만식	해외영업3팀
101	강중훈	해외영업3팀
100	한선기	해외영업3팀
206	염정하	
178	심하균	

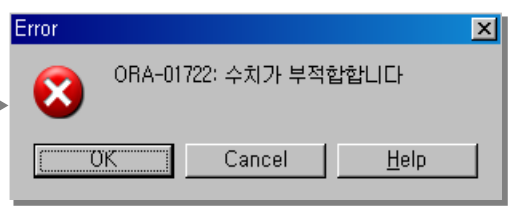
6.2.3 INSERT 사용

```
INSERT INTO EMPLOYEE (EMP_ID, EMP_NO, EMP_NAME,
                      JOB_ID, SALARY)
VALUES               ('870','860610-2412167','김서연',
                      3000000);
```



기술한 컬럼 수와 입력 값 수가 일치하지 않으면 오류 발생

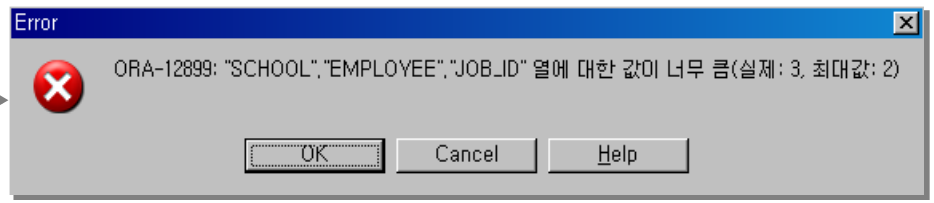
```
INSERT INTO EMPLOYEE (EMP_ID, EMP_NO, EMP_NAME,
                      JOB_ID, SALARY)
VALUES               ('870','860610-2412167','김서연',
                      'J7', '많이받음');
```



- 컬럼 타입과 입력 값 타입이 일치하지 않으면 오류 발생
- SALARY 컬럼(NUBER 타입)에 문자열(CHAR 타입)을 입력하는 경우

6.2.3 INSERT 사용

```
INSERT INTO EMPLOYEE (EMP_ID, EMP_NAME,
                      JOB_ID, EMP_NO)
VALUES ('870', '김서연', 'J10',
        '860610-2412167');
```



컬럼 타입에 맞지 않는 경우 오류 발생

```
INSERT INTO EMPLOYEE (EMP_ID, EMP_NAME, JOB_ID, EMP_NO)
VALUES ('870', '860610-2412167', 'J7', '김서연');
```

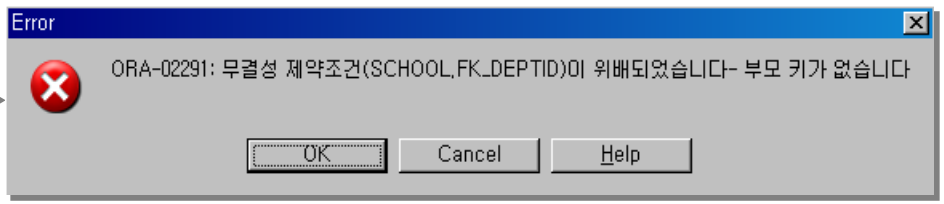
EMP_ID	EMP_NAME	EMP_NO	EMAIL	PHONE	HIRE_DATE	JOB_ID	SALARY	BONUS_PCT	MARRIAGE	MGR_ID	DEPT_ID
870	860610-2412167	김서연			09/12/29	J7			N		

- 컬럼 타입이 일치하면 의미와 무관하게 입력 가능
- 구문 오류는 아니지만 논리적인 오류 발생

6.2.4 INSERT와 무결성 제약조건

데이터 무결성이 손상되는 입력 작업은 허용되지 않음

```
INSERT INTO EMPLOYEE (EMP_ID, EMP_NO,
                      EMP_NAME, DEPT_ID)
VALUES ('990', '810116-2154219',
        '김태휘', '45');
```



EMPLOYEE 테이블의 제약조건

이름	유형	컬럼	참조
NN_ENAME	C	EMP_NAME	
NN_EMPNO	C	EMP_NO	
CHK_MRIG	C	MARRIAGE	
PK_EMPID	P	EMP_ID	
UNI_EMPNO	U	EMP_NO	
FK_DEPTID	R	DEPT_ID	PK_DEPTID
FK_JOBID	R	JOB_ID	PK_JOBID
FK_MGRID	R	MGR_ID	PK_EMPID

DEPARTMENT 테이블

DEPT_ID
10
20
30
50
60
80
90

Ch 6.

DML(Data Manipulation Language)

6.3.1 의미

- 테이블에 포함된 기존 데이터를 삭제
- 행 단위로 삭제되므로 전체 행 수가 달라짐

기존 데이터

DEPT_ID	DEPT_NAME	LOC_ID
20	회계팀	A1
10	본사 인사팀	A1
50	해외영업1팀	U1
60	기술지원팀	OT
80	해외영업2팀	A2
90	해외영업3팀	A3
30	마케팅팀	A1

삭제 후 데이터

DEPT_ID	DEPT_NAME	LOC_ID
20	회계팀	A1
50	해외영업1팀	U1
60	기술지원팀	OT
80	해외영업2팀	A2
90	해외영업3팀	A3

6.3.2 구문

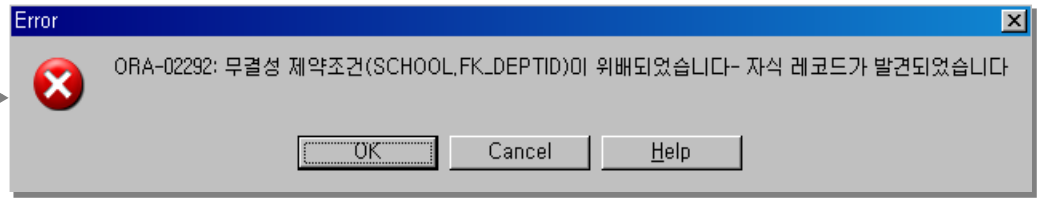
```
DELETE [FROM] table_name
[WHERE  condition ] ;
```

[구문 설명]

- WHERE 조건 생략하면 전체 행이 삭제됨
- 데이터는 삭제되고 테이블 구조는 유지됨

6.3.3 DELETE 사용

```
DELETE FROM DEPARTMENT
WHERE  LOC_ID NOT LIKE 'A%';
```



참조 데이터가 있는 경우 삭제 불가

```
DELETE FROM DEPARTMENT
WHERE  LOC_ID NOT LIKE 'A%';

SELECT * FROM DEPARTMENT;
```

삭제 작업을 수행하기 위해 제약조건을 비활성화 시켰음

DEPT_ID	DEPT_NAME	LOC_ID
20	회계팀	A1
10	본사 인사팀	A1
80	해외영업2팀	A2
90	해외영업3팀	A3
30	마케팅팀	A1

DEPT_ID	DEPT_NAME	LOC_ID
20	회계팀	A1
10	본사 인사팀	A1
50	해외영업1팀	U1
60	기술지원팀	OT
80	해외영업2팀	A2
90	해외영업3팀	A3
30	마케팅팀	A1

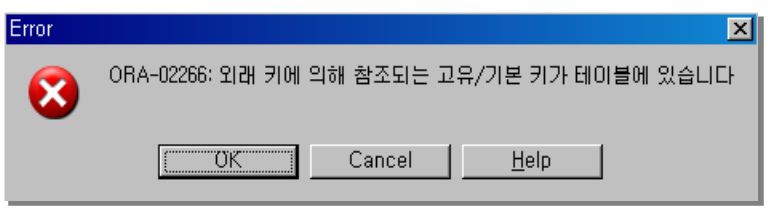
WHERE 조건이 없으면 전체 행 삭제

6.3.3 DELETE 사용 - 테이블 전체 데이터 삭제

- DELETE 명령 사용 : WHERE 조건 없는 경우
- TRUNCATE 명령 사용
  - DELETE 명령보다 수행 속도 빠름
  - 테이블 전체 데이터를 삭제하는 경우에만 사용 가능
  - 롤백 불가능, 제약 조건 있는 경우 사용 불가
  - 구문

```
TRUNCATE TABLE table_name ;
```

```
ALTER TABLE EMPLOYEE
DISABLE CONSTRAINTS FK_DEPTID;

TRUNCATE TABLE DEPARTMENT;
```



삭제 작업을 수행하기 위해 제약조건을 비활성화 시켰음

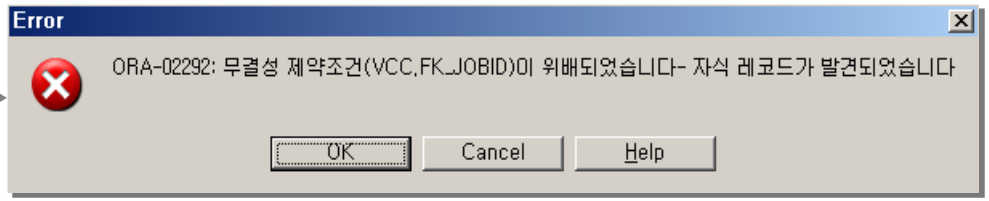
6.3.4 Deletion Rule

데이터가 삭제될 때, 무결성이 손상되는 것을 방지하기 위한 방법

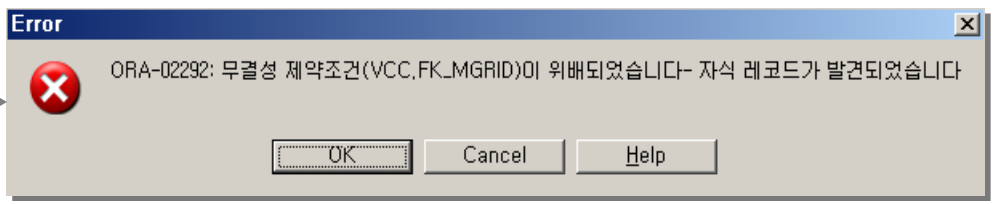
- RESTRICTED
  삭제 대상 데이터를 참조하는 데이터가 존재하면 삭제할 수 없도록 하는 규칙(기본 값)
- SET NULL
  대상 데이터를 삭제하고, 해당 데이터를 참조하는 데이터를 NULL로 바꾸는 규칙
- CASCADE
  대상 데이터를 삭제하고, 해당 데이터를 참조하는 데이터를 삭제하는 규칙

6.3.4 Deletion Rule - RESTRICTED

```
DELETE FROM JOB
WHERE  JOB_ID = 'J2';
```




```
DELETE FROM EMPLOYEE
WHERE  EMP_ID = '141';
```



EMPLOYEE 테이블의 제약조건

이름	유형	컬럼	참조
NN_ENAME	C	EMP_NAME	
NN_EMPNO	C	EMP_NO	
CHK_MRIG	C	MARRIAGE	
PK_EMPID	P	EMP_ID	
UNI_EMPNO	U	EMP_NO	
FK_DEPTID	R	DEPT_ID	PK_DEPTID
FK_JOBID	R	JOB_ID	PK_JOBID
FK_MGRID	R	MGR_ID	PK_EMPID

6.3.4 Deletion Rule - SET NULL

```
ALTER TABLE EMPLOYEE DROP CONSTRAINTS FK_MGRID;

ALTER TABLE EMPLOYEE ADD CONSTRAINTS FK_MGRID FOREIGN KEY (MGR_ID)
REFERENCES EMPLOYEE ON DELETE SET NULL;

DELETE FROM EMPLOYEE WHERE  EMP_ID = '141';
```

- EMPLOYEE 테이블의 MGR_ID 컬럼은 EMP_ID 컬럼을 참조
- 참조하는 EMP_ID 컬럼이 삭제되면 참조하고 있는 값들을 NULL로 변경하도록 함

EMP_ID	EMP_NAME	MGR_ID
124	정지현	141
141	김예수	100
143	나승원	141
144	김순이	141
149	성해교	141

EMP_ID	EMP_NAME	MGR_ID
124	정지현	
143	나승원	
144	김순이	
149	성해교	

6.3.4 Deletion Rule - CASCADE

```
ALTER TABLE EMPLOYEE DROP CONSTRAINTS FK_JOBID;

ALTER TABLE EMPLOYEE ADD CONSTRAINTS FK_JOBID FOREIGN KEY (JOB_ID)
REFERENCES JOB ON DELETE CASCADE;

DELETE FROM JOB WHERE JOB_ID = 'J2';
```

- EMPLOYEE 테이블의 JOB_ID 컬럼은 JOB 테이블의 JOB_ID 컬럼을 참조
- 참조하는 JOB_ID 컬럼이 삭제되면 참조하고 있는 값들을 삭제하도록 함

EMP_ID	EMP_NAME	JOB_ID
101	강중훈	J2
102	최만식	J2

COUNT(*)
2

EMP_ID	EMP_NAME	JOB_ID

COUNT(*)
0

- 'J2'를 참조하는 EMPLOYEE 테이블의 데이터는 2건
- JOB 테이블에서 'J2' 값을 삭제하면 EMPLOYEE 테이블에도 'J2'를 참조하는 데이터는 삭제됨

Ch 6.

DML(Data Manipulation Language)

6.4.1 개념

- 데이터 일관성을 유지시키려는 목적으로 사용하는 논리적으로 연관된 작업들의 집합
- 구성
  - 하나 이상의 연관된 DML 구문
  - 하나 이상의 DDL 구문

[계좌 이체 트랜잭션 예]

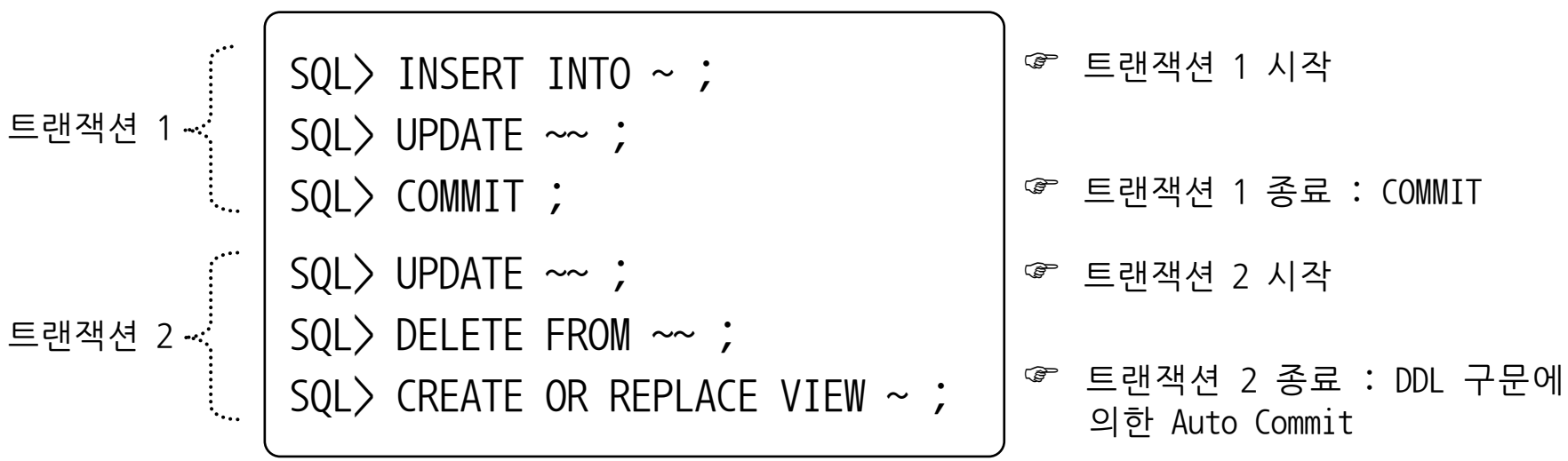
- 계좌이체 트랜잭션은 급여계좌에서 출금하는 작업과 결제계좌로 입금하는 작업으로 구성
- 급여계좌의 출금 금액 = 결제계좌의 입금 금액
- 출금 작업과 입금 작업은 동시에 처리되어야 함
- 출금 작업과 입금 작업은 양쪽 모두 성공하거나 양쪽 모두 실패하여 취소되는 경우만 의미 있음

6.4.2 범위

- 트랜잭션 시작

  첫번째 DML 구문이 실행될 때 시작됨

- 트랜잭션 종료
  - COMMIT/ROLLBACK 명령이 실행될 때 종료
  - DDL 구문이 실행될 때 종료 → Auto Commit
  - SQL*Plus 또는 DBMS Server가 비정상적으로 종료되는 경우 → Auto Rollback

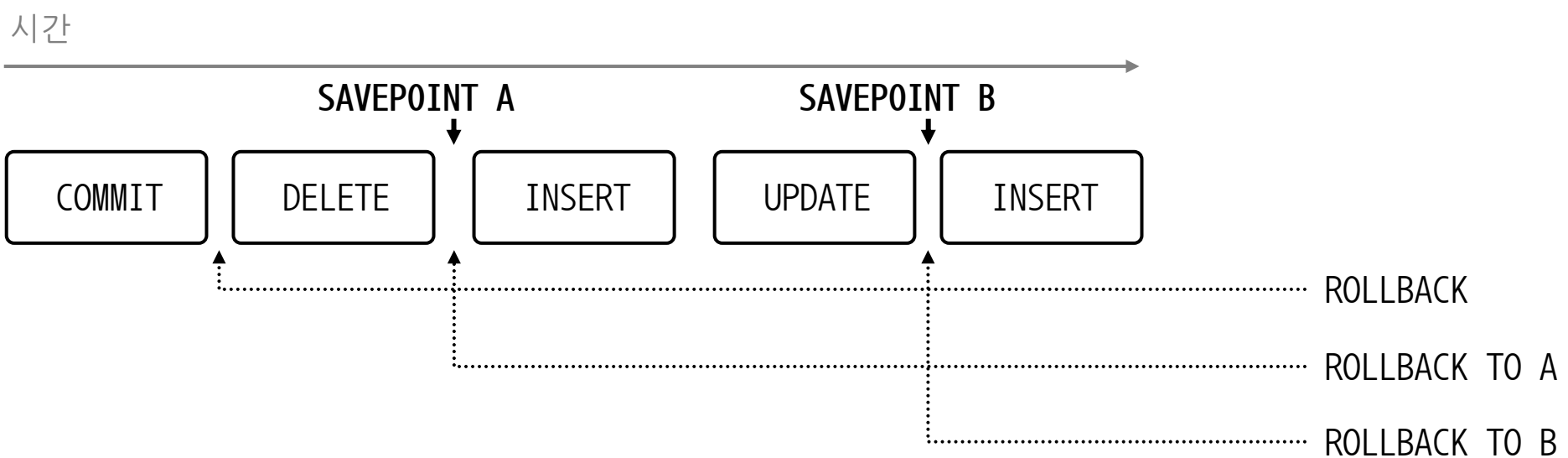


6.4.3 트랜잭션 제어

- COMMIT

  변경된 데이터를 저장하고 트랜잭션을 종료하는 명령

- ROLLBACK
  - 변경 작업을 취소하고 트랜잭션을 종료하는 명령
  - 기본적으로 데이터 상태를 트랜잭션의 시작 시점으로 되돌림
  - SAVEPOINT *savepoint_name* : 트랜잭션의 특정 시점을 기록하는 명령
  - ROLLBACK TO *savepoint_name* : 지정한 특정 시점으로 데이터 상태를 되돌릴 수 있음



6.4.4 트랜잭션과 데이터 상태

- 트랜잭션에 포함된 DML 구문에 의해 데이터 상태가 변경됨
- COMMIT/ROLLBACK 명령을 통해 변경된 데이터 상태를 확정시켜야 함

트랜잭션

- 다른 사용자(다른 세션)는 현재 작업중인 DML 결과를 볼 수 없음
- 작업 중인 행에 대해 다른 변경 작업을 할 수 없음

COMMIT

↓

- 변경 내용이 데이터베이스(파일)에 저장/반영
- 모든 사용자는 변경된 동일한 결과를 볼 수 있음
- 동일한 행에 대해 다른 변경 작업 가능
- 지금까지 설정한 모든 SAVEPOINT는 사라짐

ROLLBACK

↓

- 데이터는 이전 상태로 복구
- 동일한 행에 대해 다른 변경 작업 가능
- 지금까지 설정한 모든 SAVEPOINT는 사라짐

Appendix [참고 8] : 언두 세그먼트와 Multi-Version Read

6.4.4 트랜잭션과 데이터 상태

```
ALTER TABLE EMPLOYEE
DISABLE CONSTRAINTS FK_MGRID;

SAVEPOINT S0;

INSERT INTO DEPARTMENT
VALUES ('40','기획전략팀','A1');

SAVEPOINT S1;

UPDATE EMPLOYEE
SET DEPT_ID = '40'
WHERE DEPT_ID IS NULL;

SAVEPOINT S2;

DELETE FROM EMPLOYEE;
```

```
ROLLBACK TO S2;

SELECT COUNT(*)
FROM   EMPLOYEE;
```

COUNT(*)
22

```
SELECT COUNT(*)
FROM   EMPLOYEE
WHERE  DEPT_ID = '40';
```

COUNT(*)
3

```
ROLLBACK TO S1;

SELECT COUNT(*)
FROM   DEPARTMENT
WHERE  DEPT_ID = '40';
```

COUNT(*)
1

```
SELECT COUNT(*)
FROM   EMPLOYEE
WHERE  DEPT_ID = '40';
```

COUNT(*)
0

```
ROLLBACK TO S0;

SELECT COUNT(*)
FROM   DEPARTMENT
WHERE  DEPT_ID = '40';
```

COUNT(*)
0

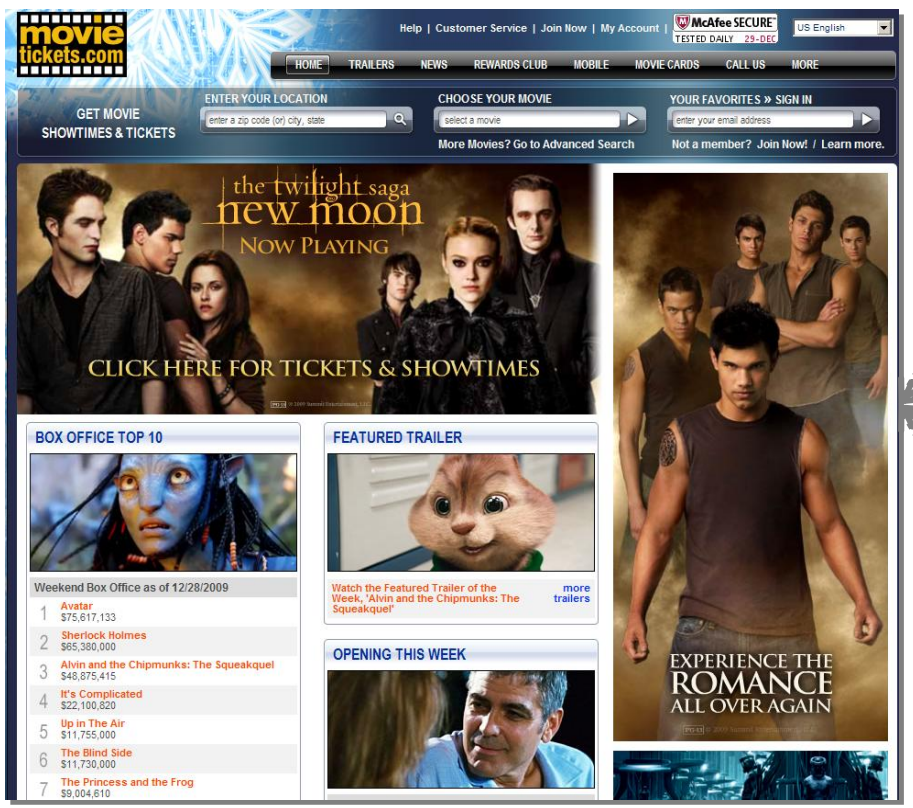
Ch 6.

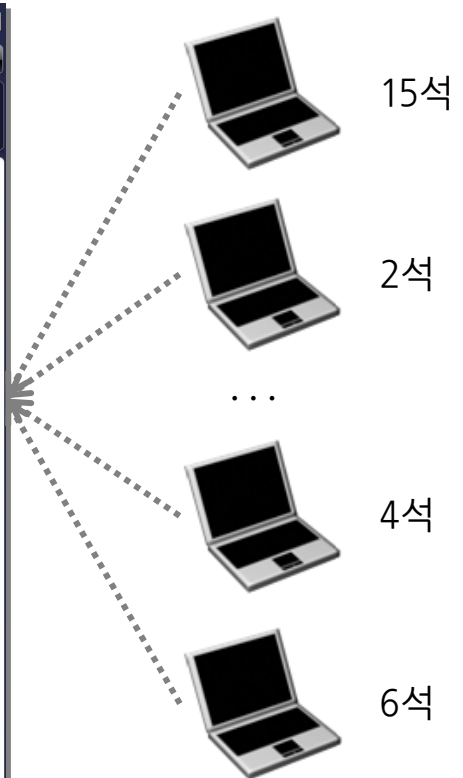
DML(Data Manipulation Language)

6.5.1 동시성Concurrency 제어 개념

- 동시성 : 다수 사용자들이 동시에 동일한 데이터에 접근하여 변경 시도 가능
- 무결성을 보장하기 위해 동시성을 제어하는 것이 필요함

여유 좌석이 10석인 경우, 동시에 1000명이 예약을 하는 상황이라면 어떻게 처리해야 하는가?





6.5.2 잠금 개념

- 가장 일반적인 데이터 동시성 제어 기법
- 특징
  - 서로 다른 사용자(서로 다른 트랜잭션)가 동시에 동일한 행을 변경할 수 없도록 방지
  - 다른 사용자(다른 트랜잭션)가 COMMIT 되지 않은 변경 내용을 Overwrite 할 수 없도록 방지
  - 트랜잭션이 실행되는 동안 자동으로 수행/유지/관리 됨

6.5.2 잠금 예

세션1

SQL Window - New

SQL | Output | Statistics

```
SELECT EMP_NAME,          ❶
       MARRIAGE
FROM   EMPLOYEE
WHERE  EMP_ID = '143';

UPDATE EMPLOYEE           ❷
SET    MARRIAGE = 'N'
WHERE  EMP_ID = '143';
```

EMP_NAME	MARRIAGE
나승원	Y

EMP_NAME	MARRIAGE
나승원	N

세션2

SQL Window - New

SQL | Output | Statistics

```
SELECT EMP_NAME,          ❸
       MARRIAGE
FROM   EMPLOYEE
WHERE  EMP_ID = '143';

UPDATE EMPLOYEE           ❹
SET    MARRIAGE = 'N'
WHERE  EMP_ID = '143';
```

- 변경되고 COMMIT 되지 않은 데이터에 대한 변경 작업 시도
  → '잠금' 상태로 대기

❺ - 이전 세션의 트랜잭션이 종료되는 순간 수행됨