

# 테이블

자바 강의실

## 목차

---

I. 테이블스페이스

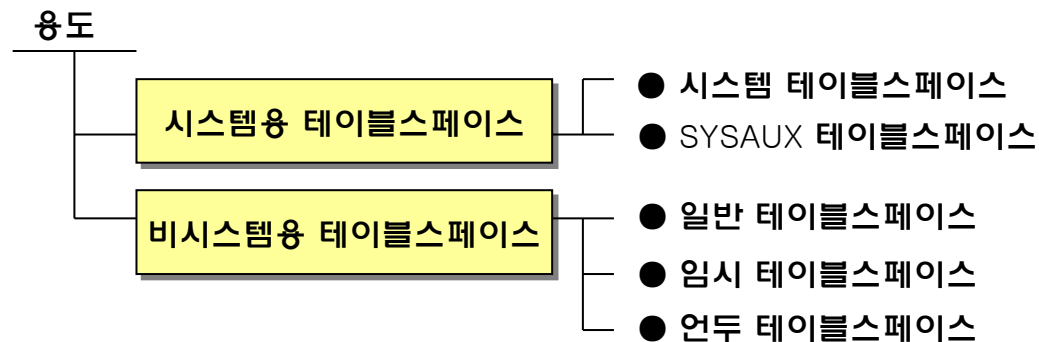
II. 테이블

III. 테이블 무결성

IV. 데이터 관리

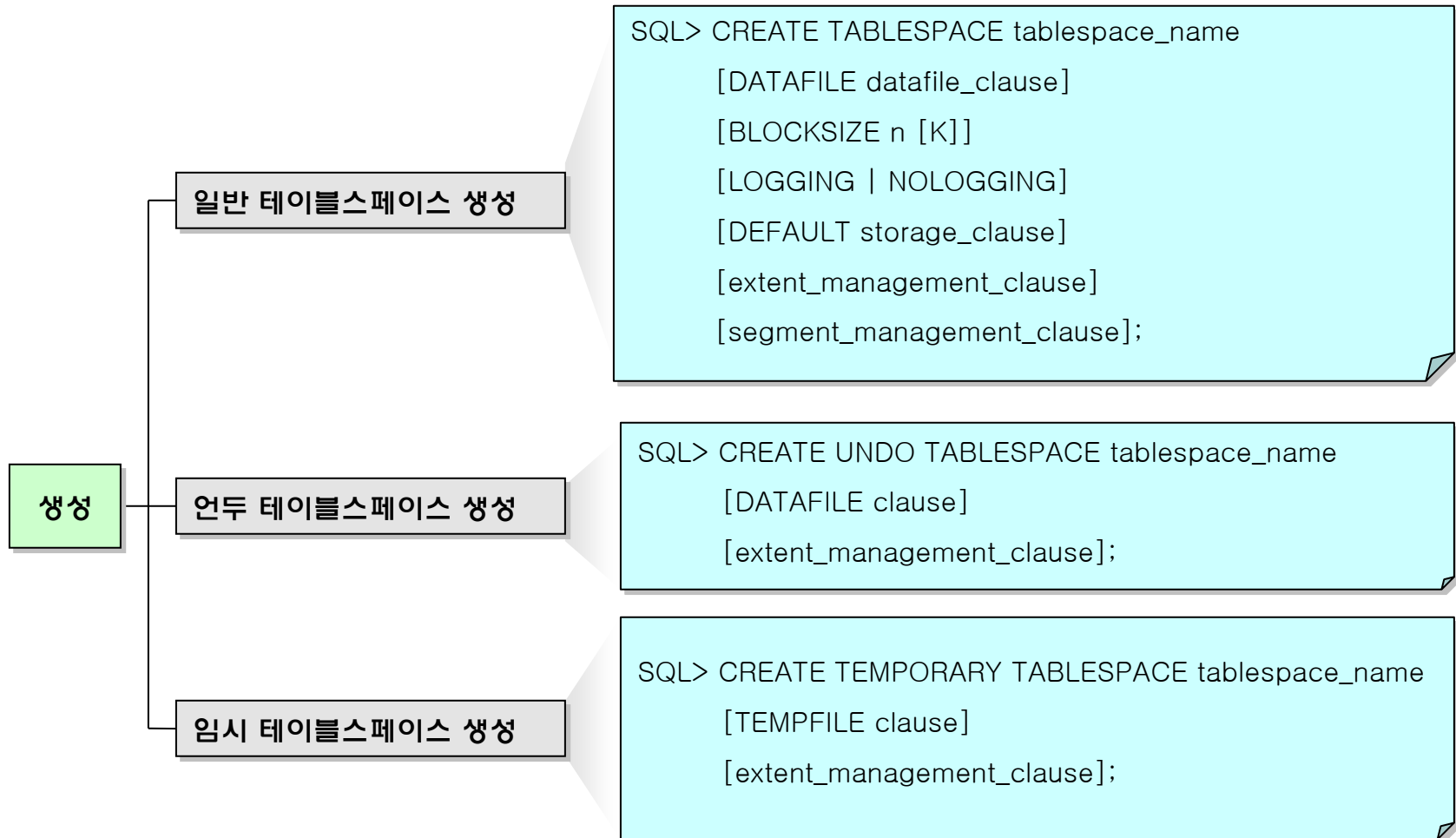
# 1. 테이블 스페이스

- 테이블스페이스란?
  - 데이터의 논리적 저장소
  - 하나 이상의 물리적 파일들로 구성



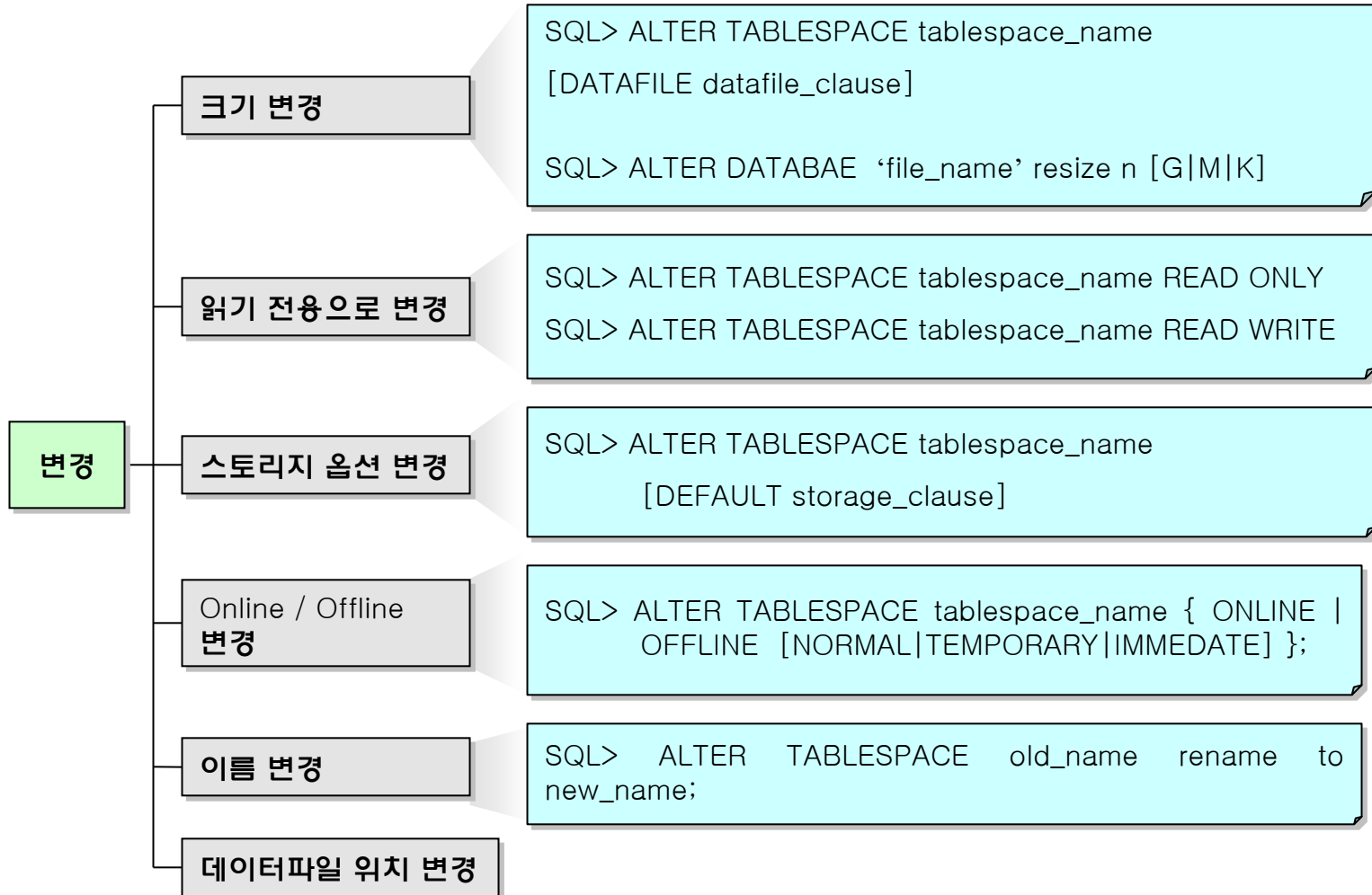
# 1. 테이블 스페이스

## • 테이블스페이스 생성



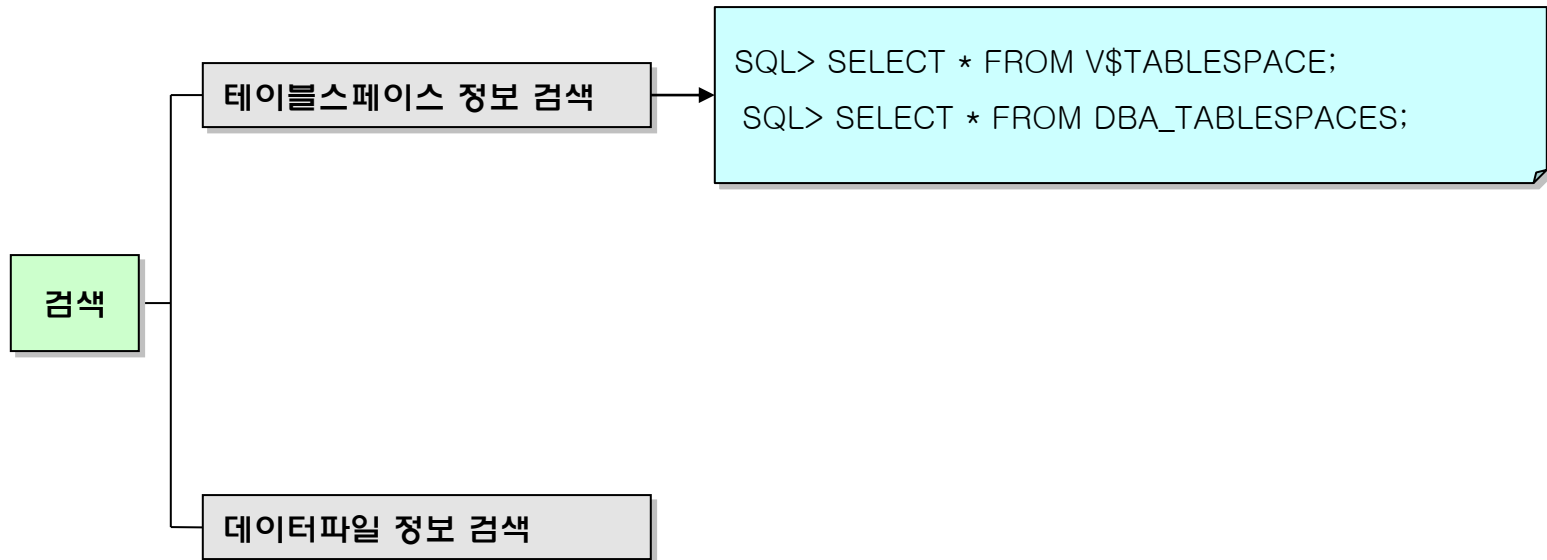
# 1. 테이블 스페이스

## • 테이블스페이스 변경



# 1. 테이블 스페이스

- 테이블스페이스 정보 조회



# 1. 테이블 스페이스

- 테이블스페이스 생성

```
SELECT FILE_NAME, BYTES, STATUS FROM DBA_DATA_FILES;
```

--테이블스페이스 확인

```
CREATE TABLESPACE JSP
```

```
DATAFILE 'D:\ORACLEXE\APP\ORACLE\ORADATA\XE\JSP.DBF'
```

```
SIZE 100M -- 기본사이즈
```

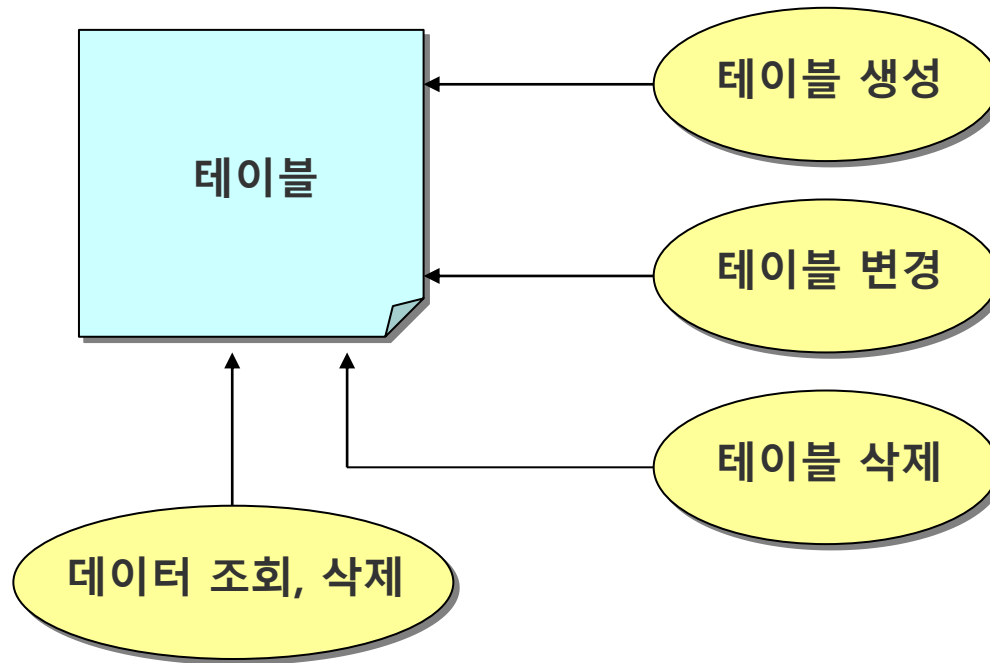
```
AUTOEXTEND ON NEXT 10M; --확장사이즈
```

```
select name from v$datafile;
```

--테이블스페이스 파일 확인

## 2. 테이블

- 테이블 사용





## 2. 테이블

- 테이블 생성하는 CREATE 문

```
CREATE 테이블명 또는 릴레이션 (  
    컬럼이름1 컬럼타입,  
    컬럼이름2 컬럼타입,  
    컬럼이름3 컬럼타입,  
);
```

## 2. 테이블

- 테이블 데이터 삭제 및 테이블 삭제하는 DROP 문

```
TRUNCATE TABLE 테이블 명;
```

```
DROP TABLE 테이블 명;
```

```
DROP TABLE 테이블명 purge; -- 복원 안됨
```

## 2. 테이블

- 테이블의 구조를 변경하는 ALTER 문

### 컬럼수정

```
ALTER TABLE 테이블 명 MODIFY (컬럼이름 컬럼타입);
```

### 컬럼 추가

```
ALTER TABLE 테이블 명  
ADD (컬럼이름 컬럼타입);
```

## 2. 테이블

- 테이블의 구조를 변경하는 ALTER 문

### 컬럼삭제

```
ALTER TABLE 테이블 명 DROP COLUMN 컬럼이름 ;
```

### 컬럼이름 변경

```
ALTER TABLE 테이블 명  
RENAME COLUMN 컬럼이름 to 새 컬럼이름;
```

## 2. 테이블

- 테이블이름을 변경하는 ALTER 문

**테이블명 변경**

```
ALTER TABLE 테이블명 RENAME to new테이블명
```

## 2. 테이블

- 릴레이션의 키
  - 각 튜플을 고유하게 식별할 수 있는 하나 이상의 애트리뷰트들의 모임
  - 후보 키(candidate key), 기본 키(primary key), 대체 키(alternate key), 외래 키(foreign key)

## 2. 테이블

### • 후보 키

- 각 튜플을 고유하게 식별하는 최소한의 애트리뷰트들의 모임
- 예: (신용카드번호, 주소)는 신용카드 회사의 고객 릴레이션의 후보 키가 아니지만 (신용카드번호)는 후보 키
- 모든 릴레이션에는 최소한 한 개 이상의 후보 키가 있음
- 후보 키도 두 개 이상의 애트리뷰트로 이루어질 수 있으며 이런 경우에 복합 키(composite key)라고 부름
- 예: (학번, 과목번호)가 후보 키

수강	학번	과목번호	학점
	11002	CS310	A0
	11002	CS313	B+
	24036	CS345	B0
	24036	CS310	A+

[그림] 수강 릴레이션

## 2. 테이블

### 릴레이션의 인스턴스와 키

그림의 학생 릴레이션에서 이름이 후보 키가 될 수 있는가?

그림의 학생 릴레이션에서 이메일이 후보 키가 될 수 있는가?

학생

학번	이름	이메일
11002	이홍근	sea@hanmail.net
24036	김순미	smkim@iweb.cwunet.ac.kr
13427	박상웅	blue@hanmir.com

[그림] 학생 릴레이션



## 2. 테이블

- 기본 키

- 한 릴레이션에 후보 키가 두 개 이상 있으면 설계자 또는 데이터베이스 관리자가 이들 중에서 하나를 기본 키로 선정함
- 예: 신용카드 회사의 고객 릴레이션에서 신용카드번호와 주민등록번호가 후보 키가 될 수 있음. 이 중에서 신용카드 번호를 기본 키로 선정
- 자연스러운 기본 키를 찾을 수 없는 경우에는 레코드 번호와 같이 종종 인위적인 키 애트리뷰트를 릴레이션에 추가할 수 있음

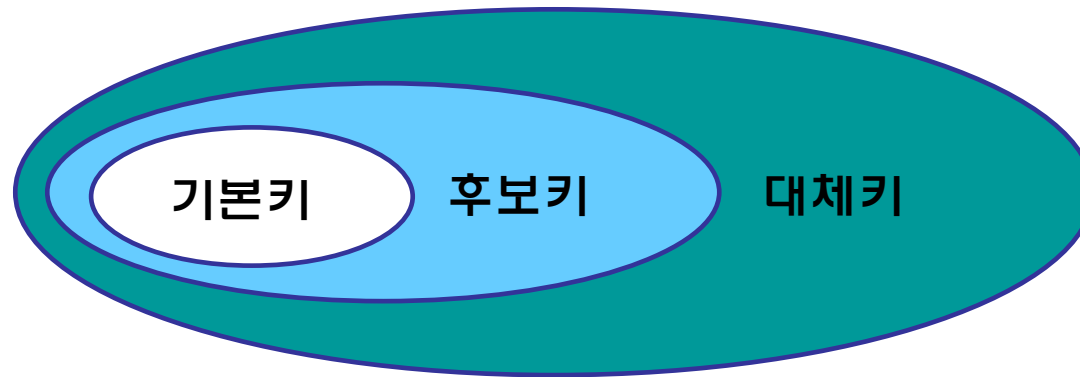
## 2. 테이블

- 기본 키 선정 시 고려사항
  - 애트리뷰트가 항상 고유한 값을 가질 것인가
  - 애트리뷰트가 확실하게 널값을 갖지 않을 것인가
  - 애트리뷰트의 값이 변경될 가능성이 높은 애트리뷰트는 기본 키로 선정하지 말 것
  - 가능하면 작은 정수 값이나 짧은 문자열을 갖는 애트리뷰트
  - 가능하면 복합 기본 키를 피할것

## 2. 테이블

- 대체 키
  - 기본 키가 아닌 후보 키
  - 예: 신용카드 회사의 고객 릴레이션에서 신용카드번호를 기본 키로 선정하면 주민등록번호는 대체 키
- 외래 키
  - 어떤 릴레이션의 기본 키를 참조하는 애트리뷰트
  - 관계 데이터베이스에서 릴레이션들 간의 관계를 나타내기 위해서 사용됨
  - 외래 키 애트리뷰트는 참조되는 릴레이션의 기본 키와 동일한 도메인을 가져야 함
  - 자신이 속한 릴레이션의 기본 키의 구성요소가 되거나 되지 않을 수 있음

## 2. 테이블

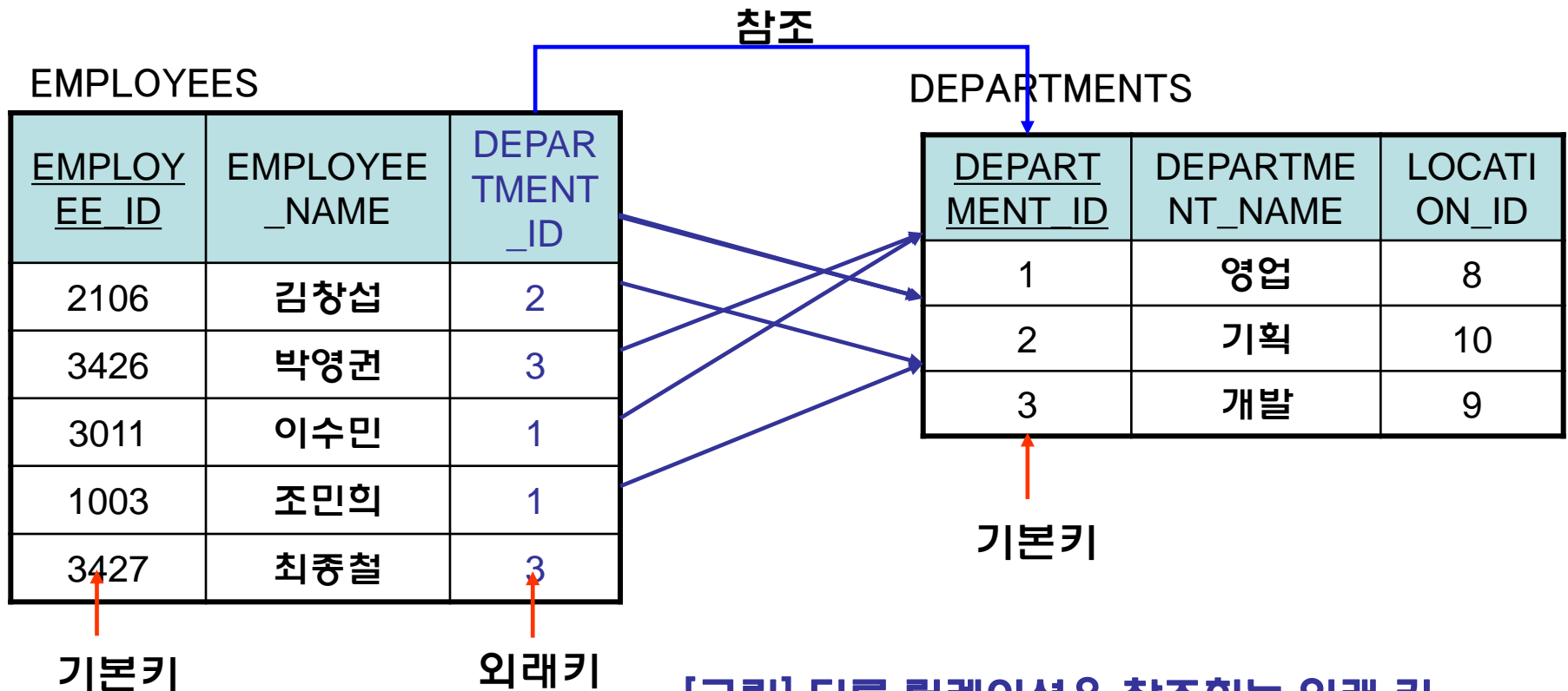


[그림] 키들의 포함 관계

## 2. 테이블

- 외래 키의 유형


- 다른 릴레이션의 기본 키를 참조하는 외래 키



## 2. 테이블

- 외래 키의 유형(계속)
  - 자체 릴레이션의 기본 키를 참조하는 외래 키

참조

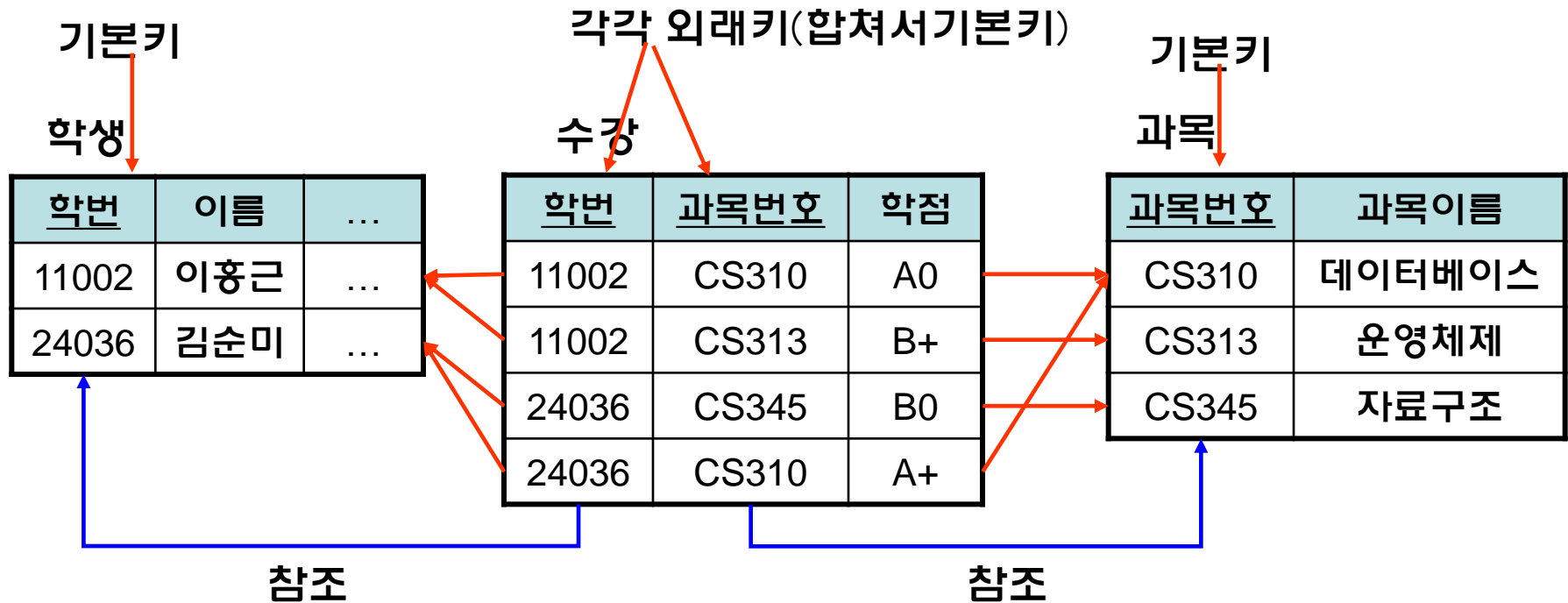


<u>EMPLOYEE_ID</u>	EMPLOYEE_NAME	MANAGER_ID	DEPARTMENT_ID
2106	김창섭	3426	2
3426	박영권	3011	3
3011	이수민	^	1
1003	조민희	3011	1
3427	최종철	2106	3

[그림] 자체 릴레이션을 참조하는 외래 키

## 2. 테이블

- 외래 키의 유형(계속)
  - 기본 키의 구성요소가 되는 외래 키



[그림] 기본 키의 구성요소가 되는 외래 키

### 3. 테이블 무결성

- 데이터 무결성(data integrity)
  - 데이터의 정확성 또는 유효성을 의미
  - 일관된 데이터베이스 상태를 정의하는 규칙들을 묵시적으로 또는 명시적으로 정의함
  - 무결성 제약조건은 데이터베이스 상태가 만족시켜야 하는 조건이다. 사용자에 의한 데이터베이스 갱신이 데이터베이스의 일관성을 깨지 않도록 보장하는 수단이다. 일반적으로 데이터베이스 상태가 실세계에 허용되는 상태만 나타낼 수 있도록 제한한다.
  - 데이터베이스가 갱신될 때 DBMS가 자동적으로 일관성 조건을 검사하므로 응용 프로그램들은 일관성 조건을 검사할 필요가 없음



### 3. 테이블 무결성

- 기본 키와 엔티티 무결성 제약조건(entity integrity constraint)
  - 릴레이션의 기본 키를 구성하는 어떤 애트리뷰트도 널값을 가질 수 없음
  - 대체 키에는 적용되지 않음
  - 사용자가 릴레이션을 생성하는 데이터 정의문에서 어떤 애트리뷰트가 릴레이션의 기본 키의 구성요소인가를 DBMS에게 알려줌

### 3. 테이블 무결성

- 외래 키와 참조 무결성 제약조건(referential integrity constraint)
  - 참조 무결성 제약조건은 두 릴레이션의 연관된 튜플들 사이의 일관성을 유지하는데 사용됨
  - 관계 데이터베이스가 포인터 없이 오직 릴레이션들만으로 이루어지고, 릴레이션 사이의 관계들이 다른 릴레이션의 기본 키를 참조하는 것을 기반으로 하여 묵시적으로 표현되기 때문에 외래 키의 개념이 중요
  - 릴레이션 R2의 외래 키가 릴레이션 R1의 기본 키를 참조할 때 참조 무결성 제약조건은 아래의 두 조건 중 하나가 성립되면 만족됨
    - 외래 키의 값은 R1의 어떤 튜플의 기본 키 값과 같다
    - 외래 키가 자신을 포함하고 있는 릴레이션의 기본 키를 구성하고 있지 않으면 널값을 가진다



### 3. 테이블 무결성

- 무결성 제약조건의 유지
  - 데이터베이스에 대한 갱신 연산의 수행 결과에 따라서는 무결성 제약조건이 위배될 수 있음
  - 데이터베이스에 대한 갱신 연산은 삽입 연산, 삭제 연산, 수정 연산으로 구분함
  - DBMS는 각각의 갱신 연산에 대하여 데이터베이스가 무결성 제약조건들을 만족하도록 필요한 조치를 취함
  - DBMS는 외래 키가 갱신되거나, 참조된 기본 키가 갱신되었을 때 참조 무결성 제약조건이 위배되지 않도록 해야 함
  - EMPLOYEE 릴레이션의 DNO 애트리뷰트가 DEPARTMENT 릴레이션의 기본 키인 DEPTNO를 참조하는 외래 키이므로, DEPARTMENT를 참조된 릴레이션, EMPLOYEE를 참조하는 릴레이션으로 부르기로 함

### 3. 테이블 무결성

EMPLOYEES

EMPLOYEE_ID	EMPLOYEE_NAME	DEPARTMENT_ID
2106	김창섭	2
3426	박영권	3
3011	이수민	1
1003	조민희	1
3427	최종철	3

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	영업	8
2	기획	10
3	개발	9
4	홍보	8

[그림] 관계 데이터베이스 인스턴스

### 3. 테이블 무결성

- 삽입

- 참조되는 릴레이션에 새로운 튜플이 삽입되면 참조 무결성 제약 조건은 위배되지 않음
- DEPARTMENT에 새로 삽입되는 튜플의 기본 키 애트리뷰트의 값에 따라서는 도메인 제약조건, 키 제약조건, 엔티티 무결성 제약 조건 등을 위배할 수 있음
- 참조하는 릴레이션에 새로운 튜플을 삽입할 때는 도메인 제약조건, 키 제약조건, 엔티티 무결성 제약조건 외에 참조 무결성 제약 조건도 위배할 수 있음
- 예: EMPLOYEE 릴레이션에 (4325, 오혜원, 6)라는 튜플을 삽입하면 참조 무결성 제약조건을 위배하게 됨

### 3. 테이블 무결성

- 삭제

- 참조하는 릴레이션에서 튜플이 삭제되면 도메인 제약조건, 키 제약조건, 엔티티 무결성 제약조건, 참조 무결성 제약조건 등 모든 제약조건을 위배하지 않음
- 참조되는 릴레이션에서 튜플이 삭제되면 참조 무결성 제약조건을 위배하는 경우가 생기거나 생기지 않을 수 있음
- 예1: DEPARTMENTS 릴레이션에서 네 번째 튜플인 (4, 홍보, 8)을 삭제하더라도 참조 무결성 제약조건을 위배하지 않음
- 예2: DEPARTMENTS 릴레이션에서 세 번째 튜플인 (3, 개발, 9)를 삭제하면 참조 무결성 제약조건을 위배하게 됨

### 3. 테이블 무결성

- 참조 무결성 제약조건을 만족시키기 위해서 DBMS가 제공하는 옵션
  - 제한(restricted)
    - 위반을 야기한 연산을 단순히 거절
    - 예: DEPARTMENTS 릴레이션에서 (3, 개발, 9)를 삭제하면 참조 무결성 제약조건을 위반하게 되므로 삭제 연산을 거절
  - 연쇄(cascade)
    - 참조되는 릴레이션에서 튜플을 삭제하고, 참조하는 릴레이션에서 이 튜플을 참조하는 튜플들도 함께 삭제
    - 예: DEPARTMENTS 릴레이션에서 (3, 개발, 9)를 삭제하면 EMPLOYEES 릴레이션에서 부서번호 3을 참조하는 두 번째 튜플과 다섯 번째 튜플도 함께 삭제

### 3. 테이블 무결성

EMPLOYEES

EMPLOY EE_ID	EMPLOYEE _NAME	DEPAR TMENT _ID
2106	김창섭	2
3426	박영권	3
3011	이수민	1
1003	조민희	1
3427	최종철	3

② 삭제

DEPARTMENTS

DEPART MENT_ID	DEPARTME NT_NAME	LOCATI ON_ID
1	영업	8
2	기획	10
3	개발	9
4	홍보	8 ①

연쇄

① 삭제

[그림] 연쇄 삭제



### 3. 테이블 무결성

- 참조 무결성 제약조건을 만족시키기 위해서 DBMS가 제공하는 옵션(계속)
  - 널값(nullify)
    - 참조되는 릴레이션에서 튜플을 삭제하고, 참조하는 릴레이션에서 이 튜플을 참조하는 튜플들의 외래 키에 널값을 삽입
    - 예: DEPARTMENT 릴레이션에서 (3, 개발, 9)를 삭제하면 EMPLOYEE 릴레이션에서 부서번호 3을 참조하는 두 번째 튜플과 다섯 번째 튜플의 부서번호에 널값을 삽입
  - 디폴트값
    - 널값을 넣는 대신에 디폴트값을 넣는다는 것을 제외하고는 바로 위의 옵션과 비슷함

### 3. 테이블 무결성

- 수정

- DBMS는 수정하는 애트리뷰트가 기본 키인지 외래 키인지 검사함
- 수정하려는 애트리뷰트가 기본 키도 아니고 외래 키도 아니면 수정 연산이 참조 무결성 제약조건을 위반하지 않음
- DBMS는 수정하려는 애트리뷰트의 새로운 값이 올바른 데이터 타입과 도메인을 만족하는지 확인하기만 하면 됨
- 기본 키나 외래 키를 수정하는 것은 하나의 튜플을 삭제하고 새로운 튜플을 그 자리에 삽입하는 것과 유사하므로, 삽입 및 삭제에서 설명한 제한, 연쇄, 널값, 디폴트값 규칙이 수정 연산에도 적용됨

### 3. 테이블 무결성

- 릴레이션 정의

```
CREATE TABLE DEPARTMENTS2
  (DEPTNO                NUMBER NOT NULL,
   DEPTNAME              CHAR(10),
   FLOOR                 NUMBER,
   PRIMARY KEY (DEPTNO) );

CREATE TABLE EMPLOYEES2
  (EMPNO                NUMBER PRIMARY KEY,
   EMPNAME              CHAR(10),
   TITLE                CHAR(10),
   MANAGER              NUMBER,
   SALARY               NUMBER,
   DNO                  NUMBER, CONSTRAINT FK_DNO
   FOREIGN KEY (DNO) REFERENCES DEPARTMENTS2 (DEPTNO) );
```

[그림] EMPLOYEES2 릴레이션과 DEPARTMENTS2 릴레이션의 생성

### 3. 테이블 무결성

- 릴레이션 제거(DROP TABLE)
  - 릴레이션의 정의와 튜플 모두 삭제
  - 삭제 옵션
    - RESTRICT : 다른 릴레이션에서 참조되지 않는 릴레이션만 제거
    - CASCADE : 릴레이션을 참조하는 뷰, 인덱스, 제약조건, 외래 키 모두 삭제

**DROP TABLE** DEPARTMENTS2 **RESTRICT**

**DROP TABLE** DEPARTMENTS2 **CASCADE**

### 3. 테이블 무결성

- 제약조건

```
CREATE TABLE EMPLOYEES2
    EMPNO          NUMBER NOT NULL,                (1)
    EMPNAME        CHAR(10) UNIQUE,                (2)
    TITLE          CHAR(10) DEFAULT '사원' ,       (3)
    MANAGER        NUMBER,
    SALARY NUMBER CHECK (SALARY < 6000000),         (4)
    DNO            NUMBER
CHECK (DNO IN (1, 2, 3, 4))                        (5)
DEFAULT 1,
CONSTRAINT PK_EMPNO PRIMARY KEY (EMPNO),           (6)
CONSTRAINT FK_MANAGER FOREIGN KEY (MANAGER)
REFERENCES EMPLOYEES2 (EMPNO)                      (7)
CONSTRAINT FK_DNO FOREIGN KEY (DNO)
REFERENCES DEPARTMENTS2 (DEPTNO)                   (8)
ON DELETE SET DEFAULT ON UPDATE CASCADE);          (9)
```

[그림] 릴레이션 정의에서 다양한 제약조건을 명시

### 3. 테이블 무결성

- **애트리뷰트 제약조건**
  - NOT NULL(1)
    - 널 값을 허용하지 않을때
  - UNIQUE(2)
    - 동일한 애트리뷰트값을 갖는 튜플이 두개 이상 존재하지 않도록 보장
  - DEFAULT(3)
    - 널 값 대신 특정 값을 지정
  - CHECK(4, 5)
    - 애트리뷰트가 가질수 있는 값들의 범위 지정
- **기본키 제약 조건(6)**
  - 각 릴레이션마다 최대 한 개의 기본 키 지정
- **참조 무결성 제약 조건(7,8)**
  - 외래 키의 무결성 보장

### 3. 테이블 무결성

- 참조 무결성 제약조건 유지
  - ON DELETE NO ACTION
  - ON DELETE CASCADE
  - ON DELETE SET NULL
  - ON DELETE SET DEFAULT
  
  - ON UPDATE NO ACTION
  - ON UPDATE CASCADE
  - ON UPDATE SET NULL
  - ON UPDATE SET DEFAULT

### 3. 테이블 무결성

#### 예: ON UPDATE CASCADE

UPDATE문을 사용하여 DEPARTMENTS 릴레이션의 3번 부서의 부서번호를 6번으로 수정하면 EMPLOYEES 릴레이션에서 3번 부서에 근무하는 모든 직원들의 소속 부서번호가 자동적으로 6으로 수정된다

```
UPDATE DEPARTMENTS
SET DEPARTMENT_ID = 6
WHERE DEPARTMENT_ID = 3;
```

DEPARTMENT			EMPLOYEE	
DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	EMPLOYEE_ID	EMPLOYEE_NAME
1	영업	8	2106	김창섭
2	기획	10	3426	박영권
<del>3</del>	개발	9	3011	이수민
4	총무	7	1003	조민희

EMPLOYEE_ID	EMPLOYEE_NAME	...	DEPARTMENT_ID
2106	김창섭	...	2
3426	박영권	...	1
3011	이수민	...	<del>3</del> 6
1003	조민희	...	2
3427	최종철	...	<del>3</del> 6
1365	김상원	...	1
4377	이성래	...	2

6

기본 키의 수정이  
외래 키에도 파급됨



### 3. 테이블 무결성

- 무결성 제약조건의 추가 및 삭제

```
ALTER TABLE STUDENT ADD CONSTRAINT STUDENT_PK  
PRIMARY KEY (STNO) ;
```

```
ALTER TABLE STUDENT DROP CONSTRAINT STUDENT_PK;
```

## 4. 데이터 관리

- INSERT문

- 기존의 릴레이션에 튜플을 삽입
- 참조되는 릴레이션에 튜플이 삽입되는 경우에는 참조 무결성 제약조건의 위배가 발생하지 않으나 참조하는 릴레이션에 튜플이 삽입되는 경우에는 참조 무결성 제약조건을 위배할 수 있음
- 릴레이션에 한 번에 한 튜플씩 삽입하는 것과 한 번에 여러 개의 튜플들을 삽입할 수 있는 것으로 구분
- 릴레이션에 한 번에 한 튜플씩 삽입하는 INSERT문

```
INSERT
INTO      릴레이션(애트리뷰트1, ..., 애트리뷰트n)
VALUE     (값1, ..., 값n);
```

## 4. 데이터 관리

예: 한 개의 튜플을 삽입

질의: DEPARTMENT 릴레이션에 (5, 연구, ^) 튜플을 삽입하는 INSERT문은 아래와 같다.

```
INSERT INTO DEPARTMENTS
(DEPARTMENT_ID, DEPARTMENT_NAME, LOCATION_ID)
VALUES (5, '연구' ,NULL );
```

DEPARTMENT

<u>DEPARTMENT_ID</u>	DEPARTMENT_NAME	LOCATION_ID
1	영업	8
2	기획	10
3	개발	9
4	총무	7
5	연구	^

## 4. 데이터 관리

- INSERT문(계속)

- 릴레이션에 한 번에 여러 개의 튜플들을 삽입하는 INSERT문

```
INSERT
INTO      릴레이션(애트리뷰트1, ..., 애트리뷰트n)
SELECT ... FROM ... WHERE ... ;
```

### 예: 여러 개의 튜플을 삽입

질의: EMPLOYEE 릴레이션에서 급여가 3000000 이상인 직원들의 이름, 직급, 급여를 검색하여 HIGH\_SALARY라는 릴레이션에 삽입하라. HIGH\_SALARY 릴레이션은 이미 생성되어 있다.

```
INSERT      INTO      HIGH_SALARY
              (FIRST_NAME, JOB_ID, SALARY)
SELECT      FIRST_NAME, JOB_ID, SALARY
FROM        EMPLOYEES
WHERE       SALARY >= 3000000;
```

## 4. 데이터 관리

- DELETE문

- 삭제 연산은 한 릴레이션으로부터 한 개 이상의 튜플들을 삭제함
- 참조되는 릴레이션의 삭제 연산의 결과로 참조 무결성 제약조건이 위배될 수 있으나, 참조하는 릴레이션에서 튜플을 삭제하면 참조 무결성 제약조건을 위배하지 않음
- DELETE문의 구문

```
DELETE
FROM      릴레이션
WHERE     조건 ;
```

## 4. 데이터 관리

### 예: DELETE문

질의: DEPARTMENT 릴레이션에서 4번 부서를 삭제하라.

```
DELETE          FROM  DEPARTMENTS
WHERE           DEMPARTMENT_ID= 4 ;
```

<u>DEMPARTMENT_ID</u>	DEPARTMENT_NAME	LOCATION_ID
1	영업	8
2	기획	10
3	개발	9
4	총무	7

DEPARTMENT

4번 부서  
삭제

<u>DEMPARTMENT_ID</u>	DEPARTMENT_NAME	LOCATION_ID
1	영업	8
2	기획	10
3	개발	9

## 4. 데이터 관리

- UPDATE문

- 한 릴레이션에 들어 있는 튜플들의 애트리뷰트 값들을 수정
- 기본 키나 외래 키에 속하는 애트리뷰트의 값이 수정되면 참조 무결성 제약조건을 위배할 수 있음
- UPDATE문의 구문

UPDATE	릴레이션
SET	애트리뷰트 = 값 또는 식[, ...]
WHERE	조건 ;

## 4. 데이터 관리

### 예: UPDATE문

질의: 사원번호가 2106인 사원의 소속 부서를 3번 부서로 옮기고, 급여를 5% 올려라.

```
UPDATE      EMPLOYEES
SET         DEPARTMENT_ID = 3,
           SALARY = SALARY * 1.05
WHERE      EMPLOYEE_ID = 2106 ;
```

EMPLOYEE

<u>EMPLOYEE_ID</u>	...	SALARY	DEPARTMENT_ID
2106	...	2500000	2
3426	...	3000000	1
3011	...	4000000	3
1003	...	3000000	2
3427	...	1500000	3
1365	...	1500000	1
4377	...	5000000	2

2106  
UPDATE

EMPLOYEE

<u>EMPLOYEE_ID</u>	...	SALARY	DEPARTMENT_ID
2106	...	2625000	3
3426	...	3000000	1
3011	...	4000000	3
1003	...	3000000	2
3427	...	1500000	3
1365	...	1500000	1
4377	...	5000000	2