

# SQL 실습

## 4.4 SELECT문

### □ SELECT문

- ✓ 관계 데이터베이스에서 정보를 검색하는 SQL문
- ✓ 관계 대수의 실렉션과 의미가 완전히 다름
- ✓ 관계 대수의 실렉션, 프로젝션, 조인, 카티션 곱 등을 결합한 것
- ✓ 관계 데이터베이스에서 가장 자주 사용됨
- ✓ 여러 가지 질의들의 결과를 보이기 위해서 다음 슬라이드의 그림 4.8의 관계 데이터베이스 상태를 사용함

## 4.4 SELECT문(계속)

EMPLOYEE

EMPNO	EMPNAME	TITLE	MANAGER	SALARY	DNO
2106	김창섭	대리	1003	2500000	2
3426	박영권	과장	4377	3000000	1
3011	이수민	부장	4377	4000000	3
1003	조민희	과장	4377	3000000	2
3427	최종철	사원	3011	1500000	3
1365	김상원	사원	3426	1500000	1
4377	이성래	사장	^	5000000	2

DEPARTMENT

DEPTNO	DEPTNAME	FLOOR
1	영업	8
2	기획	10
3	개발	9
4	총무	7

[그림 4.8] 관계 데이터베이스 상태

## 4.4 SELECT문(계속)

### □ 기본적인 SQL 질의

- ✓ SELECT절과 FROM절만 필수적인 절이고, 나머지는 선택 사항

<b>SELECT</b>	<b>[DISTINCT]</b> 애틀리뷰트(들)	(1)
<b>FROM</b>	릴레이션(들)	(2)
<b>[WHERE</b>	조건	(3)
	<b>[중첩 질의]</b>	(4)
<b>[GROUP BY</b>	애틀리뷰트(들)	(5)
<b>[HAVING</b>	조건	(6)
<b>[ORDER BY</b>	애틀리뷰트(들) <b>[ASC DESC]</b> ];	(7)

[그림 4.9] SELECT문의 형식

## 4.4 SELECT문(계속)

□ 릴레이션의 모든 애트리뷰트나 일부 애트리뷰트들을 검색

예 : \*를 사용하여 모든 애트리뷰트들을 검색

질의: 전체 부서의 모든 애트리뷰트들을 검색하라.

```
SELECT      *  
FROM        DEPARTMENT;
```

DEPTNO	DEPTNAME	FLOOR
1	영업	8
2	기획	10
3	개발	9
4	총무	7

## 4.4 SELECT문(계속)

예 : 원하는 애트리뷰트들의 이름을 열거

질의: 모든 부서의 부서번호와 부서이름을 검색하라.

```
SELECT      DEPTNO, DEPTNAME  
FROM        DEPARTMENT;
```

DEPTNO	DEPTNAME
1	영업
2	기획
3	개발
4	총무

## 4.4 SELECT문(계속)

### □ 상이한 값들을 검색

예 : DISTINCT절을 사용하지 않을 때

질의: 모든 사원들의 직급을 검색하라.

```
SELECT    TITLE
FROM      EMPLOYEE;
```

TITLE
대리
과장
부장
과장
사원
사원
사장

10.5만 가지는 7 개위인 예!

## 4.4 SELECT문(계속)

예 : DISTINCT절을 사용할 때

질의: 모든 사원들의 상이한 직급을 검색하라.

```
SELECT DISTINCT TITLE
FROM EMPLOYEE;
```

TITLE
대리
과장
부장
사원
사장

하나의 에러가  
결과 값에 차원  
있어 중요!!

헤라 에러부터 내  
공백 뺐어 제거.



## 4.4 SELECT문(계속)

### □ 특정한 튜플들의 검색

예 : WHERE절을 사용하여 검색 조건을 명시

질의: 2번 부서에 근무하는 직원들에 관한 모든 정보를 검색하라.

**③ SELECT** \*  
**① FROM** EMPLOYEE  
**② WHERE** DNO = 2;

↑ 검색 조건을 진다.

결과물이 나올까 걱정:

① 'EMPLOYEE' 테이블을 가져온다.

② 'DNO' 열 내 값이 2인 튜플만 남긴다.

③ 남겨진 튜플에서 SELECT 절에 기입된 열에

EMPNO	EMPNAME	TITLE	MANAGER	SALARY	DNO
1003	조민희	과장	4377	3000000	2
2016	김창섭	대리	1003	2500000	2
4377	이성래	사장	^	5000000	2

해당하는 값들만 최종적으로 출력해낸다.

## 4.4 SELECT문(계속)

### □ 문자열 비교

예 : %를 사용하여 문자열 비교

질의: 이씨 성을 가진 직원들의 이름, 직급, 소속 부서번호를 검색하라.

```
SELECT      EMPNAME, TITLE, DNO
FROM        EMPLOYEE
WHERE       EMPNAME LIKE '이%';
```

↑ access이전 \*  
SMS이전 %

EMPNAME	TITLE	DNO
이수민	부장	3
이성래	사장	2

## 4.4 SELECT문(계속)

### □ 다수의 검색 조건

- ✓ 아래와 같은 질의는 잘못되었음 (부서 이름은 단일 값을 갖기에 AND 연산을 만족하는 튜플 없음)

```
SELECT      FLOOR
FROM        DEPARTMENT
WHERE       DEPTNAME= '영업' AND DEPTNAME= '개발' ;
```

〈표 4.6〉 연산자들의 우선 순위

연산자	우선순위
비교 연산자	1
NOT	2
AND	3
OR	4

## 4.4 SELECT문(계속)

### 예 : 부울 연산자를 사용한 프레디키트

질의: 직급이 과장이면서 1번 부서에서 근무하는 직원들의 이름과 급여를 검색하라.

```
SELECT      EMPNAME, SALARY
FROM        EMPLOYEE
WHERE        TITLE = '과장' AND DNO = 1;
```

EMPNAME	SALARY
박영권	3000000

## 4.4 SELECT문(계속)

### □ 부정 검색 조건

#### 예 : 부정 연산자

질의: 직급이 과장이면서 1번 부서에 속하지 않은 직원들의 이름과 급여를 검색하라.

```
SELECT      EMPNAME, SALARY
FROM        EMPLOYEE
WHERE       TITLE = '과장' AND DNO <> 1;
```

EMPNAME	SALARY
조민희	3000000

~!~를 SQL에선  
'<>'로 나타낸다.

## 4.4 SELECT문(계속)

### □ 범위를 사용한 검색

#### 예 : 범위 연산자

질의: 급여가 3000000원 이상이고, 4500000원 이하인 직원들의 이름, 직급, 급여를 검색하라.

```
SELECT      EMPNAME, TITLE, SALARY
FROM        EMPLOYEE
WHERE        SALARY BETWEEN 3000000 AND 4500000;
```

BETWEEN은 양쪽의 경계값을 포함하므로 이 질의는 아래의 질의와 동등하다.

```
SELECT      EMPNAME, TITLE, SALARY
FROM        EMPLOYEE
WHERE        SALARY >= 3000000 AND SALARY <= 4500000;
```

EMPNAME	TITLE	SALARY
박영권	과장	3000000
이수민	부장	4000000
조민희	과장	3000000

## 4.4 SELECT문(계속)

### □ 리스트를 사용한 검색

예 : IN

질의: 1번 부서나 3번 부서에 소속된 직원들에 관한 모든 정보를 검색하라.

```
SELECT      *  
FROM        EMPLOYEE  
WHERE       DNO IN (1, 3);
```

EMPNO	EMPNAME	TITLE	MANAGER	SALARY	DNO
1365	김상원	사원	3426	1500000	1
3011	이수민	부장	4377	4000000	3
3426	박영권	과장	4377	3000000	1
3427	최종철	사원	3011	1500000	3

## 4.4 SELECT문(계속)

□ SELECT절에서 산술 연산자(+, -, \*, /) 사용

예 : 산술 연산자

질의: 직급이 과장인 직원들에 대하여 이름과, 현재의 급여, 급여가 10% 인상됐을 때의 값을 검색하라.

```
SELECT      EMPNAME, SALARY, SALARY * 1.1 AS NEWSALARY
FROM        EMPLOYEE
WHERE       TITLE = '과장';
```

EMPNAME	SALARY	NEWSALARY
박영권	3000000	3300000
조민희	3000000	3300000

산술 연산자는 SELECT 문에 들어간다!

AS 키워드로  
'별칭' 사용 가능!



## 4.4 SELECT문(계속)

예 : ORDER BY

질의: 2번 부서에 근무하는 직원들의 급여, 직급, 이름을 검색하여 급여의 오름차순으로 정렬하라.

```
SELECT SALARY, TITLE, EMPNAME
FROM EMPLOYEE
WHERE DNO = 2
ORDER BY SALARY DESC
```

← 내림차순이 대한 키를  
대기에 적는다!  
DESC

순서대로 정렬이 없는 데이터  
ORDER BY 가능!

SALARY	TITLE	EMPNAME
2500000	대리	김창섭
3000000	과장	조민희
5000000	사장	이성래

## 4.4 SELECT문(계속)

〈표 4.10〉 집단 함수의 기능

집단 함수	기능
COUNT	튜플이나 값들의 개수
SUM	값들의 합
AVG	값들의 평균값
MAX	값들의 최대값
MIN	값들의 최소값

특정 열에 속한 튜플 값  
전체(집단)에 대해

적용하는 함수  
⇒ SELECT 문에서  
사용해야 하는  
함수이다!

## 4.4 SELECT문(계속)

예 : 집단 함수

질의: 모든 직원들의 평균 급여와 최대 급여를 검색하라.

```
SELECT    AVG (SALARY) AS AVGSAL, MAX (SALARY) AS MAXSAL
FROM      EMPLOYEE;
```

AVGSAL	MAXSAL
2928571	5000000

SELECT 절에서 사용!!  
선택된 에트리뷰트 내 모든 특점에 해당 함수를 적용하는 것이다.

## 4.4 SELECT문(계속)

같은 묶음끼리 묶는 것이다.

예 : 그룹화

질의: 모든 사원들에 대해서 사원들이 속한 부서번호별로 그룹화하고, 각 부서마다 부서번호, 평균 급여, 최대 급여를 검색하라.

```
SELECT DNO, AVG(SALARY) AS AVGSAL, MAX(SALARY) AS MAXSAL
FROM EMPLOYEE
GROUP BY DNO;
```

② ← 각 DNO에 대한 평균 SALARY와 최대 SALARY를 구한다.

· GROUP BY절이 존재하면, SELECT절에는 GROUP BY절 내 명시된 애트리뷰트와 그룹함수가 명시되어야 한다.

## 4.4 SELECT문(계속)

EMPLOYEE

EMPNO	EMPNAME	TITLE	MANAGER	SALARY	DNO
3426	박영권	과장	4377	3000000	1
1365	김상원	사원	3426	1500000	1
2106	김창섭	대리	1003	2500000	2
1003	조민희	과장	4377	3000000	2
4377	이성래	사장	^	5000000	2
3011	이수민	부장	4377	4000000	3
3427	최종철	사원	3011	1500000	3



DNO	AVGSAL	MAXSAL
1	2250000	3000000
2	3500000	5000000
3	2750000	4000000

## 4.4 SELECT문(계속)

### 예 : 그룹화

질의: 모든 직원들에 대해서 직원들이 속한 부서번호별로 그룹화하고, 평균 급여가 2500000원 이상인 부서에 대해서 부서번호, 평균 급여, 최대 급여를 검색하라.

```
SELECT      DNO, ②AVG (SALARY) AS AVGSAL, MAX (SALARY) AS MAXSAL
FROM        EMPLOYEE
①GROUP BY   DNO
②HAVING     AVG (SALARY) >= 2500000;
```

← <변칙을 having 절에서 사용할 수 없다!>

↑ '그룹화'가 반영된 결과물이 나오난 뒤, 해당 결과물에 대한 조건을 적용함.

## 4.4 SELECT문(계속)

EMPLOYEE

EMPNO	EMPNAME	TITLE	MANAGER	SALARY	DNO
3426	박영권	과장	4377	3000000	1
1365	김상원	사원	3426	1500000	1
2106	김창섭	대리	1003	2500000	2
1003	조민희	과장	4377	3000000	2
4377	이성래	사장	^	5000000	2
3011	이수민	부장	4377	4000000	3
3427	최종철	사원	3011	1500000	3

그룹

GROUP BY

DNO	AVGSAL	MAXSAL
1	2250000	3000000
2	3500000	5000000
3	2750000	4000000

HAVING

DNO	AVGSAL	MAXSAL
2	3500000	5000000
3	2750000	4000000

SQL 이전 연산에 대한 것은  
'결과물'을 사용해야 한다.

## 4.4 SELECT문(계속)

예: 합집합

: 두 SQL 쿼리문의 결과를 합치는 연산자. (레이블  
상에서, 서로 '별'로  
합쳐진 결과물이  
도출됨)

질의. 김창섭이 속한 부서이거나 개발 부서의 부서번호를 검색하라.

```
(SELECT      DNO
FROM      EMPLOYEE
WHERE      EMPNAME = '김창섭')
UNION
(SELECT      DEPTNO
FROM      DEPARTMENT
WHERE      DEPTNAME = '개발');
```

이러한 것은  
'합집합'의  
개념!!

해당 괄호는  
없어도 된다!

UNION: 행의 모든 값이 일치하는 행은  
1개만 출력됨

DNO
2
3

UNIONALL: 모든 행을 다 출력함.



## 4.4 SELECT문(계속)

### 예 : 조인 질의

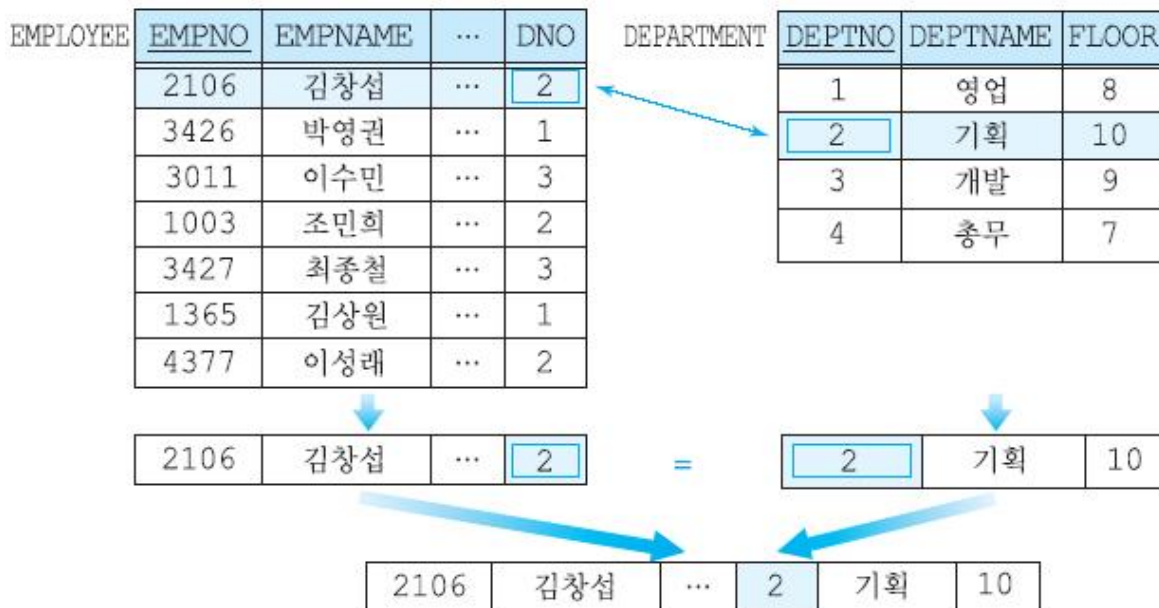
질의: 모든 사원의 이름과 이 사원이 속한 부서 이름을 검색하라.

```
SELECT      EMPNAME, DEPTNAME
FROM        EMPLOYEE AS E, DEPARTMENT AS D
WHERE       E.DNO = D.DEPTNO;
```

↑ 이처럼, 관계 맺어진 두 테이블에 대한 카라전 쿼리를  
실행하면 유용하다!

★ < FROM 절에 두 테이블을 넣으면, 두 테이블에 대한  
'카라전 쿼리'를 실행한다. >

## 4.4 SELECT문(계속)



## 4.4 SELECT문(계속)

최종 결과 릴레이션은 아래의 릴레이션에서 EMPNAME과 DEPTNAME을 프로젝션한 것이다.

EMPNO	EMPNAME	TITLE	MANAGER	SALARY	DNO	DEPTNAME	FLOOR
1003	조민희	과장	4377	3000000	2	기획	10
1365	김상원	사원	3426	1500000	1	영업	8
2106	김창섭	대리	1003	2500000	2	기획	10
3011	이수민	부장	4377	4000000	3	개발	9
3426	박영권	과장	4377	3000000	1	영업	8
3427	최종철	사원	3011	1500000	3	개발	9
4377	이성래	사장	∧	5000000	2	기획	10

## 4.4 SELECT문(계속)

### □ 자체 조인(self join)

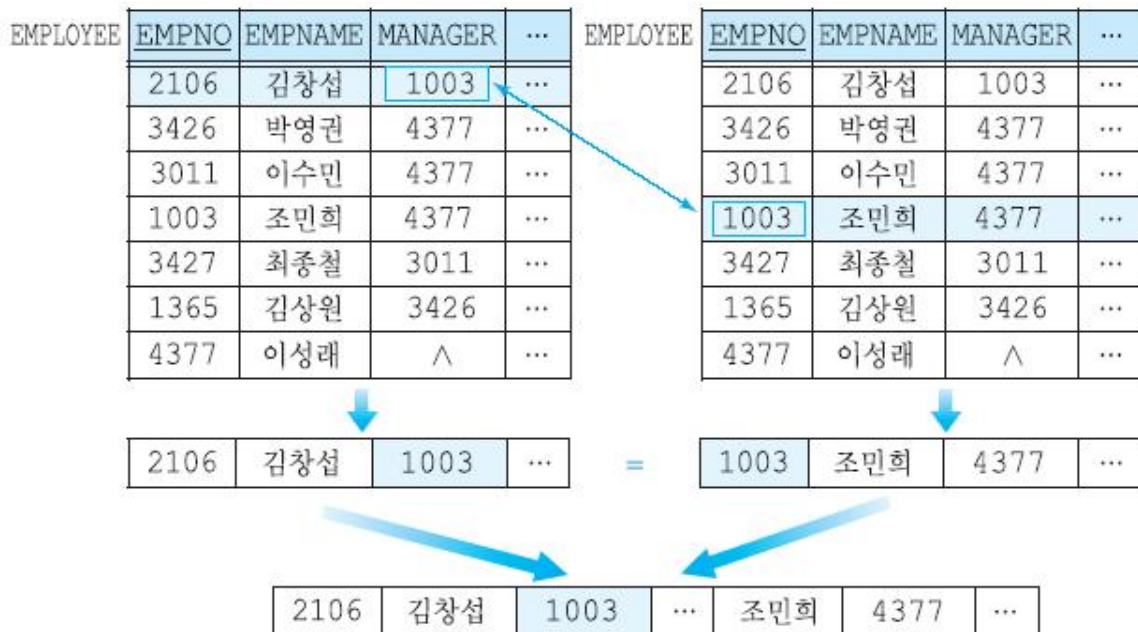
- ✓ 한 릴레이션에 속하는 튜플을 동일한 릴레이션에 속하는 튜플들과 조인하는 것
- ✓ 실제로는 한 릴레이션이 접근되지만 FROM절에 두 릴레이션이 참조되는 것처럼 나타내기 위해서 그 릴레이션에 대한 별칭을 두 개 지정해야 함

#### 예 : 자체 조인

질의: 모든 사원에 대해서 사원의 이름과 직속 상사의 이름을 검색하라.

```
SELECT      E.EMPNAME, M.EMPNAME
FROM        EMPLOYEE E, EMPLOYEE M
WHERE       E.MANAGER = M.EMPNO;
```

## 4.4 SELECT문(계속)



## 4.4 SELECT문(계속)

최종 결과 릴레이션은 아래와 같다.

E.EMPNAME	M.EMPNAME
김창섭	조민희
박영권	이성래
이수민	이성래
조민희	이성래
최종철	이수민
김상원	박영권

## 4.4 SELECT문(계속)

### 예 : 조인과 ORDER BY의 결합

질의: 모든 사원에 대해서 소속 부서이름, 사원의 이름, 직급, 급여를 검색하라. 부서 이름에 대해서 오름차순, 부서이름이 같은 경우에는 SALARY에 대해서 내림차순으로 정렬하라.

```
SELECT  DEPTNAME, EMPNAME, TITLE, SALARY
FROM    EMPLOYEE E, DEPARTMENT D
WHERE   E.DNO = D.DEPTNO
ORDER BY DEPTNAME, SALARY DESC;
```

↑ 정렬 기준을 이렇게 두개 설정함  
 (DEPTNAME 오름차순, SALARY 내림차순)  
 '내림차순' (DESC)

DEPTNAME	EMPNAME	TITLE	SALARY	
개발	이수민	부장	4000000	↓ 내림차순
개발	최종철	사원	1500000	
기획	이성래	사장	5000000	↓ 내림차순
기획	조민희	과장	3000000	
기획	김창섭	대리	2500000	↓ 내림차순
영업	박영권	과장	3000000	
영업	김상원	사장	1500000	↓ 내림차순

오름차순  
↓

↓ 내림차순

↓ 내림차순

↓ 내림차순

↑ 해당 명령어를  
기입하는 위치!

## 4.4 SELECT문(계속)

← 한 개의 에트리뷰트에 한 개의 튜플이 존재하는 것

- 한 개의 스칼라값이 반환되는 경우

예 : 한 개의 스칼라 값이 반환되는 경우

질의: 박영권과 같은 직급을 갖는 모든 사원들의 이름과 직급을 검색하라.

```
SELECT  EMPNAME, TITLE
FROM    EMPLOYEE
WHERE   TITLE =
```

과장

```
(SELECT  TITLE
FROM    EMPLOYEE
WHERE   EMPNAME = '박영권') ;
```

중첩 질의

EMPNAME	TITLE
박영권	과장
조민희	과장



## 4.4 SELECT문(계속)

### 예 : IN을 사용한 질의

질의: 영업부나 개발부에 근무하는 직원들의 이름을 검색하라.

```
SELECT EMPNAME  
FROM EMPLOYEE  
WHERE DNO IN
```

(1, 3)

```
(SELECT DEPTNO  
FROM DEPARTMENT  
WHERE DEPTNAME = '영업' OR DEPTNAME = '개발') ;
```

```
SELECT E.EMPNAME, D.DEPTNAME  
FROM DEPARTMENT AS D, EMPLOYEE AS E  
WHERE E.DNO = D.DEPTNO AND (D.DEPTNAME = '영업' OR D.DEPTNAME = '개발')
```

↑ 이렇게 간단히 DML을 작성해도 무관함.

## 4.4 SELECT문(계속)

이 질의를 중첩 질의를 사용하지 않은 다음과 같은 조인 질의로 나타낼 수 있다. 실제로, 중첩 질의를 사용하여 표현된 대부분의 질의를 중첩 질의가 없는 조인 질의로 표현할 수 있다.

```
SELECT    EMPNAME
FROM      EMPLOYEE E, DEPARTMENT D
WHERE      E.DNO = D.DEPTNO
            AND (D.DEPTNAME = '영업' OR D.DEPTNAME = '개발');
```

EMPNAME
박영권
이수민
최종철
김상원

## 4.4 SELECT문(계속)

### 예 : EXISTS를 사용한 질의

질의: 영업부나 개발부에 근무하는 직원들의 이름을 검색하라.

```
SELECT EMPNAME
FROM EMPLOYEE E
WHERE EXISTS
  (SELECT
    FROM DEPARTMENT D
    WHERE E.DNO = D.DEPTNO
    AND (DEPTNAME = '영업' OR DEPTNAME = '개발'));
```

EMPNAME
박영권
이수민
최종철
김상원

외부질의에서 가져온  
튜플이 내부질의의 결과  
레이블에 존재하는지  
확인하는 질의

여기는 아무거나 와도 된다.

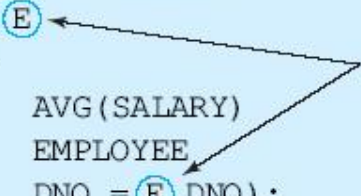
상관 내부 질의

## 4.4 SELECT문(계속)

### 예 : 상관 중첩 질의

질의: 자신이 속한 부서의 직원들의 평균 급여보다 많은 급여를 받는 직원들에 대해서 이름, 부서번호, 급여를 검색하라.

```
SELECT EMPNAME, DNO, SALARY
FROM EMPLOYEE E
WHERE SALARY >
      (SELECT AVG (SALARY)
       FROM EMPLOYEE
       WHERE DNO = E.DNO);
```



EMPNAME	DNO	SALARY
박영권	1	3000000
이수민	3	4000000
이성래	2	5000000

## 4.5 INSERT, DELETE, UPDATE문(계속)

### 예 : 한 개의 튜플을 삽입

질의: DEPARTMENT 릴레이션에 (5, 연구, 0) 튜플을 삽입하는 INSERT문은 아래와 같다.

```
INSERT INTO DEPARTMENT  
VALUES (5, '연구', 0);
```

DEPARTMENT	DEPTNO	DEPTNAME	FLOOR
	1	영업	8
	2	기획	10
	3	개발	9
	4	총무	7
	5	연구	0

## 4.5 INSERT, DELETE, UPDATE문(계속)

### □ INSERT문(계속)

- ✓ 릴레이션에 한 번에 여러 개의 튜플들을 삽입하는 INSERT문

```
INSERT  
INTO    릴레이션 (애틀리뷰트1, ..., 애틀리뷰트n)  
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 있다고 가정한다.

```
INSERT    INTO HIGH_SALARY (ENAME, TITLE, SAL)  
SELECT    EMPNAME, TITLE, SALARY  
FROM      EMPLOYEE  
WHERE      SALARY >= 3000000;
```

## 4.5 INSERT, DELETE, UPDATE문(계속)

### 예 : DELETE문

질의: DEPARTMENT 릴레이션에서 4번 부서를 삭제하라.

```
DELETE FROM DEPARTMENT  
WHERE DEPTNO = 4;
```

## 4.5 INSERT, DELETE, UPDATE문(계속)

### □ 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;

#### 예 : UPDATE문

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

```
UPDATE EMPLOYEE
SET     DNO = 3, SALARY = SALARY * 1.05
WHERE   EMPNO = 2106;
```