# SQL 기초

# 데이터 정의어와 무결성 제약조건

- 릴레이션 제거

```
DROP TABLE DEPARTMENT;
```

- ALTER TABLE

```
ALTER TABLE EMPLOYEE ADD PHONE CHAR(13);
```

- 인덱스 생성

```
CREATE INDEX EMPDNO_IDX ON EMPLOYEE(DNO);
```

# 데이터 정의어와 무결성 제약조건(계속)

❑ 제약조건

```
CREATE TABLE EMPLOYEE
   (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 (1,2,3,4,5,6))             ,(5)
    PRIMARY KEY(EMPNO),                                           (6)
    FOREIGN KEY(MANAGER) REFERENCES EMPLOYEE(EMPNO),              (7)
    FOREIGN KEY(DNO) REFERENCES DEPARTMENT(DEPTNO)                (8)
            ON DELETE CASCADE);                                   (9)
```

[그림 4.7] 릴레이션 정의에서 다양한 제약조건을 명시

# 데이터 정의어와 무결성 제약조건(계속)

```
CREATE TABLE EMPLOYEE (
    ID NUMBER,
    NAME CHAR(10),
    SALARY NUMBER,
    MANAGER_SALARY NUMBER,
    CHECK (MANAGER_SALARY > SALARY));
```

# 데이터 정의어와 무결성 제약조건(계속)

- 참조 무결성 제약조건 유지

```
ON DELETE NO ACTION

ON DELETE CASCADE

ON DELETE SET NULL

ON DELETE SET DEFAULT


ON UPDATE NO ACTION
```

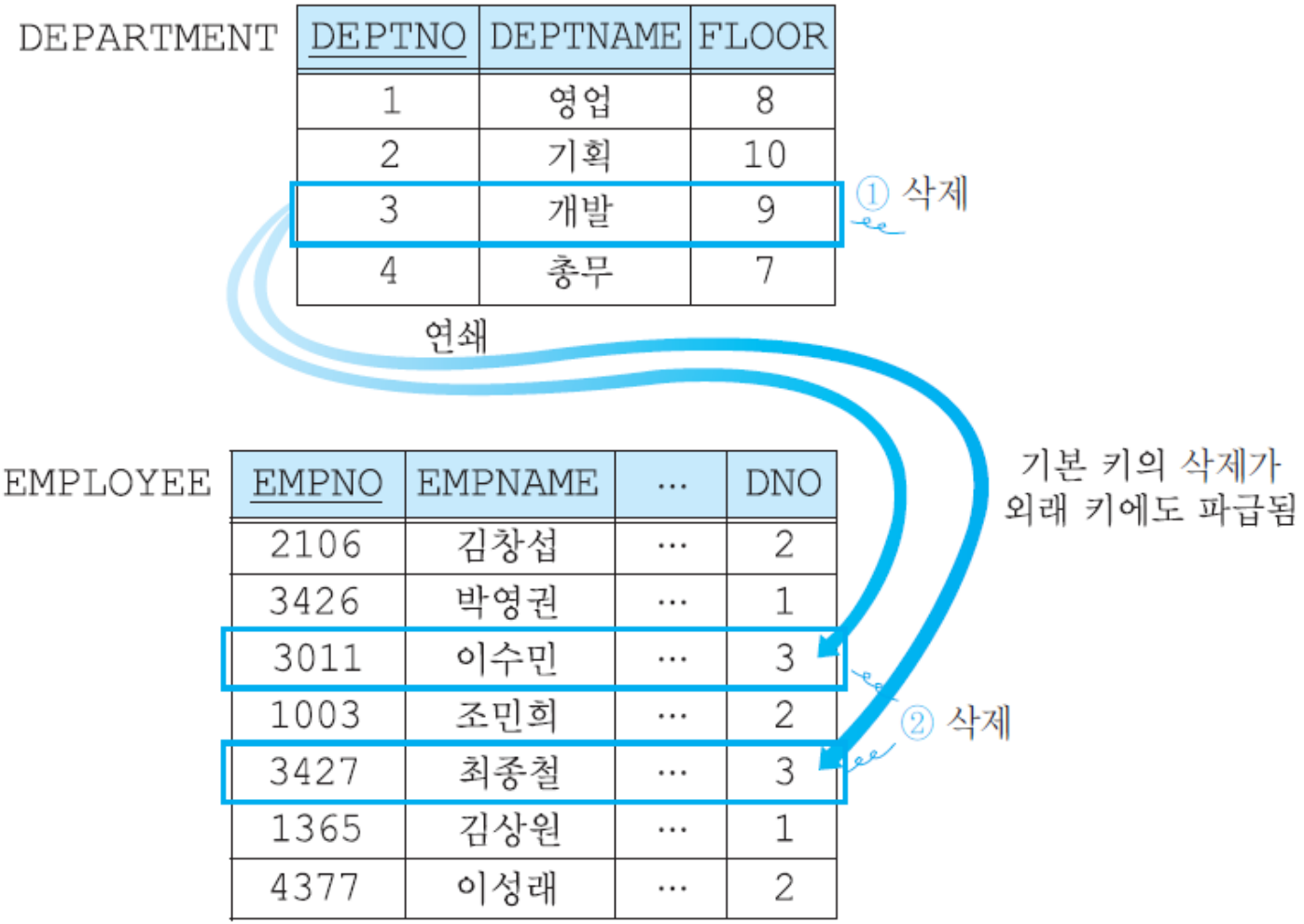
# 데이터 정의어와 무결성 제약조건(계속)

**예 : ON DELETE CASCADE**

4.5절에서 설명할 DELETE문을 사용하여 다음과 같이 DEPARTMENT 릴레이션에서 3번 부서의 투플을 삭제하면, EMPLOYEE 릴레이션에서 3번 부서에 근무하는 모든 사원들의 투플도 자동적으로 삭제된다.

```
DELETE DEPARTMENT
WHERE  DEPTNO = 3;
```

# 데이터 정의어와 무결성 제약조건(계속)



# 데이터 정의어와 무결성 제약조건(계속)

- 무결성 제약조건의 추가 및 삭제

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


ALTER TABLE STUDENT DROP CONSTRAINT STUDENT_PK;
```

# 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 그림 4.8의 관계 데이터베이스 상태를 사용함

# 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

# SELECT문(계속)

## ❑ 기본적인 SQL 질의

✓ SELECT절과 FROM절만 필수적인 절이고, 나머지는 선택 사항

```
SELECT      [DISTINCT] 애트리뷰트(들)          (1)  } 필수
FROM        릴레이션(들)                       (2)  }
[WHERE      조건                               (3)  } 선택
                  [중첩 질의]]                 (4)  }
[GROUP BY   애트리뷰트(들)]                    (5)  }
[HAVING     조건]                              (6)  }
[ORDER BY   애트리뷰트(들) [ASC|DESC]];        (7)  }
```

[그림 4.9] SELECT문의 형식

# SELECT문(계속)

- 별칭(alias)
  - ✓ 서로 다른 릴레이션에 동일한 이름을 가진 애트리뷰트가 속해 있을 때 애트리뷰트의 이름을 구분하는 방법

```
EMPLOYEE.DNO

FROM EMPLOYEE AS E, DEPARTMENT AS D
```

# SELECT문(계속)

❑ 릴레이션의 모든 애트리뷰트나 일부 애트리뷰트들을 검색

**예 : * 를 사용하여 모든 애트리뷰트들을 검색**

질의: 전체 부서의 모든 애트리뷰트들을 검색하라.

[실행 결과]

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

# SELECT문(계속)

**예 : 원하는 애트리뷰트들의 이름을 열거**

질의: 모든 부서의 부서번호와 부서이름을 검색하라.

[실행 결과]

| DEPTNO | DEPTNAME |
|---|---|
| 1 | 영업 |
| 2 | 기획 |
| 3 | 개발 |
| 4 | 총무 |

# SELECT문(계속)

❑ 상이한 값들을 검색

**예 : DISTINCT절을 사용하지 않을 때**

질의: 모든 사원들의 직급을 검색하라.

[실행 결과]

| TITLE |
|---|
| 대리 |
| 과장 |
| 부장 |
| 과장 |
| 사원 |
| 사원 |
| 사장 |

# SELECT문(계속)

## 예 : DISTINCT절을 사용할 때

질의: 모든 사원들의 상이한 직급을 검색하라.

[실행 결과]

| TITLE |
| --- |
| 대리 |
| 과장 |
| 부장 |
| 사원 |
| 사장 |

# SELECT문(계속)

## ❑ 특정한 투플들의 검색

**예 : WHERE절을 사용하여 검색 조건을 명시**

질의: 2번 부서에 근무하는 사원들에 관한 모든 정보를 검색하라.

[실행 결과]

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

# SELECT문(계속)

❑ 문자열 비교

**예 : %를 사용하여 문자열 비교**

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

[실행 결과]

| EMPNAME | TITLE | DNO |
|---|---|---|
| 이수민 | 부장 | 3 |
| 이성래 | 사장 | 2 |

# SELECT문(계속)

❑ 다수의 검색 조건

✓ 아래와 같은 질의는 잘못되었음

```
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 |

# SELECT문(계속)

**예 : 부울 연산자를 사용한 프레디키트**

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

[실행 결과]

| EMPNAME | SALARY |
|---|---|
| 박영권 | 3000000 |

# SELECT문(계속)

## ❑ 부정 검색 조건

**예 : 부정 연산자**

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

[실행 결과]

| EMPNAME | SALARY |
|---|---|
| 조민희 | 3000000 |

# SELECT문(계속)

## ❑ 범위를 사용한 검색

**예 : 범위 연산자**

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

[실행 결과]

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

# SELECT문(계속)

## ❑ 리스트를 사용한 검색

**예 : IN**

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

[실행 결과]

| 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 |

# SELECT문(계속)

❑ SELECT절에서 산술 연산자(+, -, *, /) 사용

**예 : 산술 연산자**

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

[실행 결과]

| EMPNAME | SALARY | NEWSALARY |
|---|---|---|
| 박영권 | 3000000 | 3300000 |
| 조민희 | 3000000 | 3300000 |

# SELECT문(계속)

- 널값
  - ✓ 널값을 포함한 다른 값과 널값을 +, - 등을 사용하여 연산하면 결과는 널
  - ✓ COUNT(*)를 제외한 집단 함수들은 널값을 무시함
  - ✓ 어떤 애트리뷰트에 들어 있는 값이 널인가 비교하기 위해서 'DNO=NULL'처럼 나타내면 안됨

```
SELECT    EMPNO, EMPNAME
FROM      EMPLOYEE
WHERE     DNO = NULL;
```

# SELECT문(계속)

- 널값(계속)
  - ✓ 다음과 같은 비교 결과는 모두 거짓

```
NULL > 300
NULL = 300
NULL <> 300
NULL = NULL
NULL <> NULL
```

  - ✓ 올바른 표현

```
SELECT    EMPNO, EMPNAME
FROM      EMPLOYEE
WHERE     DNO IS NULL;
```

# SELECT문(계속)

〈표 4.7〉 unknown에 대한 OR 연산

| | true | false | unknown |
|---|---|---|---|
| true | true | true | true |
| false | true | false | unknown |
| unknown | true | unknown | unknown |

〈표 4.8〉 unknown에 대한 AND 연산

| | true | false | unknown |
|---|---|---|---|
| true | true | false | unknown |
| false | false | false | false |
| unknown | unknown | false | unknown |

# SELECT문(계속)

〈표 4.9〉 unknown에 대한 NOT 연산

| | |
|---|---|
| true | false |
| false | true |
| unknown | unknown |

# SELECT문(계속)

## ❑ ORDER BY절

- ✓ 사용자가 SELECT문에서 질의 결과의 순서를 명시하지 않으면 릴레이션에 투플들이 삽입된 순서대로 사용자에게 제시됨
- ✓ ORDER BY절에서 하나 이상의 애트리뷰트를 사용하여 검색 결과를 정렬할 수 있음
- ✓ SELECT문에서 가장 마지막에 사용되는 절
- ✓ 디폴트 정렬 순서는 오름차순(ASC)
- ✓ DESC를 지정하여 정렬 순서를 내림차순으로 지정할 수 있음
- ✓ 널값은 오름차순에서는 가장 마지막에 나타나고, 내림차순에서는 가장 앞에 나타남
- ✓ SELECT절에 명시한 애트리뷰트들을 사용해서 정렬해야 함

# SELECT문(계속)

**예 : ORDER BY**

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

[실행 결과]

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

# SELECT문(계속)

❑ 집단 함수

- ✓ 데이터베이스에서 검색된 여러 투플들의 집단에 적용되는 함수
- ✓ 한 릴레이션의 한 개의 애트리뷰트에 적용되어 단일 값을 반환함
- ✓ SELECT절과 HAVING절에만 나타날 수 있음
- ✓ COUNT(*)를 제외하고는 널값을 제거한 후 남아 있는 값들에 대해서 집단 함수의 값을 구함
- ✓ COUNT(*)는 결과 릴레이션의 모든 행들의 총 개수를 구하는 반면에 COUNT(애트리뷰트)는 해당 애트리뷰트에서 널값이 아닌 값들의 개수를 구함
- ✓ 키워드 DISTINCT가 집단 함수 앞에 사용되면 집단 함수가 적용되기 전에 먼저 중복을 제거함

# SELECT문(계속)

〈표 4.10〉 집단 함수의 기능

| 집단 함수 | 기능 |
|---|---|
| COUNT | 투플이나 값들의 개수 |
| SUM | 값들의 합 |
| AVG | 값들의 평균값 |
| MAX | 값들의 최대값 |
| MIN | 값들의 최소값 |

# SELECT문(계속)

**예 : 집단 함수**

질의: 모든 사원들의 평균 급여와 최대 급여를 검색하라.

[실행 결과]

| AVGSAL | MAXSAL |
|---|---|
| 2928571 | 5000000 |

# SELECT문(계속)

## ❑ 그룹화

- ✓ GROUP BY절에 사용된 애트리뷰트에 동일한 값을 갖는 투플들이 각각 하나의 그룹으로 묶임
- ✓ 이 애트리뷰트를 그룹화 애트리뷰트(grouping attribute)라고 함
- ✓ 각 그룹에 대하여 결과 릴레이션에 하나의 투플이 생성됨
- ✓ SELECT절에는 각 그룹마다 하나의 값을 갖는 애트리뷰트, 집단 함수, 그룹화에 사용된 애트리뷰트들만 나타날 수 있음
- ✓ 다음 질의는 그룹화를 하지 않은 채 EMPLOYEE 릴레이션의 모든 투플에 대해서 사원번호와 모든 사원들의 평균 급여를 검색하므로 잘못됨

```
SELECT  EMPNO, AVG(SALARY)
FROM    EMPLOYEE;
```

# SELECT문(계속)

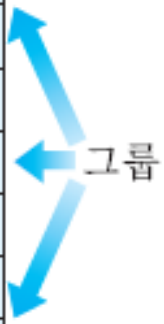
**예 : 그룹화**

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

# 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 |




[실행 결과]

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

# SELECT문(계속)

## ❑ HAVING절

- ✓ 어떤 조건을 만족하는 그룹들에 대해서만 집단 함수를 적용할 수 있음
- ✓ 각 그룹마다 하나의 값을 갖는 애트리뷰트를 사용하여 각 그룹이 만족해야 하는 조건을 명시함
- ✓ 그룹화 애트리뷰트에 같은 값을 갖는 투플들의 그룹에 대한 조건을 나타내고, 이 조건을 만족하는 그룹들만 질의 결과에 나타남
- ✓ HAVING절에 나타나는 애트리뷰트는 반드시 GROUP BY절에 나타나거나 집단 함수에 포함되어야 함

# SELECT문(계속)

**예 : 그룹화**

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

# 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 |

[실행 결과]

HAVING →

| DNO | AVGSAL | MAXSAL |
|---|---|---|
| 2 | 3500000 | 5000000 |
| 3 | 2750000 | 4000000 |

# SELECT문(계속)

❑ 집합 연산

- ✓ 집합 연산을 적용하려면 두 릴레이션이 합집합 호환성을 가져야 함
- ✓ UNION(합집합), EXCEPT(차집합), INTERSECT(교집합), UNION ALL(합집합), EXCEPT ALL(차집합), INTERSECT ALL(교집합)

# SELECT문(계속)

**예 : 합집합**

질의: 김창섭이 속한 부서이거나 개발 부서의 부서번호를 검색하라.

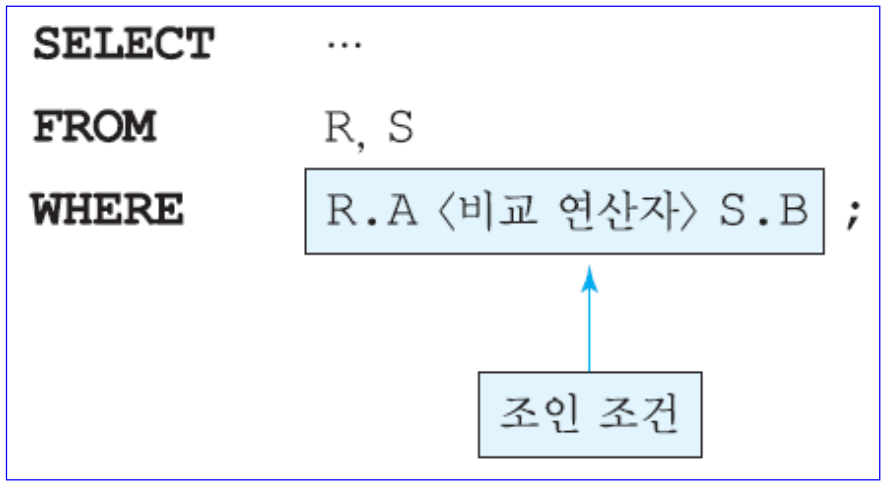
[실행 결과]

| DNO |
|---|
| 2 |
| 3 |

# SELECT문(계속)

## ❑ 조인

- ✓ 두 개 이상의 릴레이션으로부터 연관된 투플들을 결합
- ✓ 일반적인 형식은 아래의 SELECT문과 같이 FROM절에 두 개 이상의 릴레이션들이 열거되고, 두 릴레이션에 속하는 애트리뷰트들을 비교하는 조인 조건이 WHERE절에 포함됨
- ✓ 조인 조건은 두 릴레이션 사이에 속하는 애트리뷰트 값들을 비교 연산자로 연결한 것
- ✓ 가장 흔히 사용되는 비교 연산자는 =



# SELECT문(계속)

❑ 조인(계속)

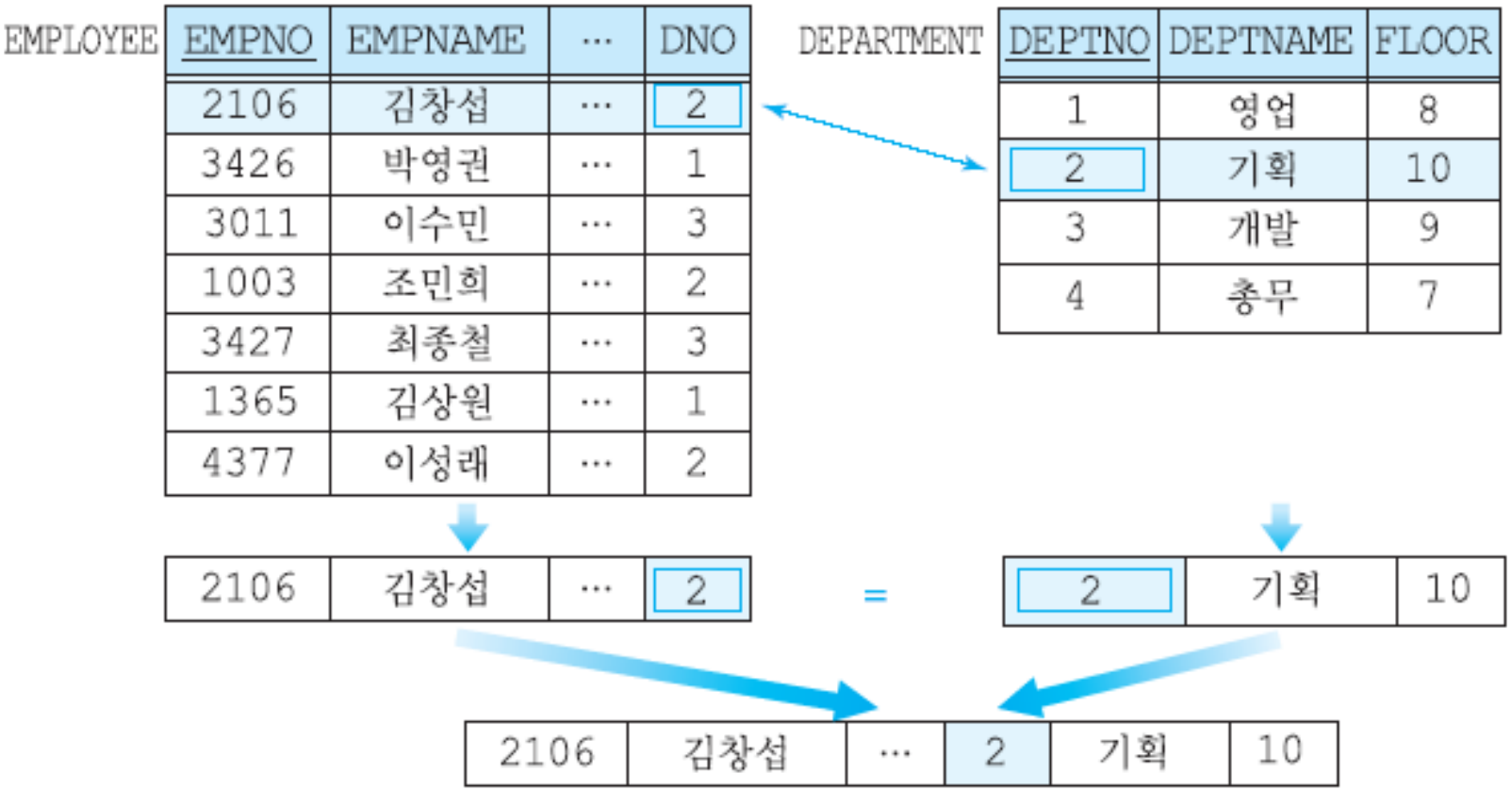
- ✓ 조인 조건을 생략했을 때와 조인 조건을 틀리게 표현했을 때는 카티션 곱이 생성됨
- ✓ 조인 질의가 수행되는 과정을 개념적으로 살펴보면 먼저 조인 조건을 만족하는 투플들을 찾고, 이 투플들로부터 SELECT절에 명시된 애트리뷰트들만 프로젝트하고, 필요하다면 중복을 배제하는 순서로 진행됨
- ✓ 조인 조건이 명확해지도록 애트리뷰트 이름 앞에 릴레이션 이름이나 투플 변수를 사용하는 것이 바람직
- ✓ 두 릴레이션의 조인 애트리뷰트 이름이 동일하다면 반드시 애트리뷰트 이름 앞에 릴레이션 이름이나 투플 변수를 사용해야 함

# SELECT문(계속)

**예 : 조인 질의**

질의: 모든 사원의 이름과 이 사원이 속한 부서 이름을 검색하라.

# SELECT문(계속)


EMPLOYEE
EMPNO
EMPNAME
…
DNO
2106 김창섭 … 2
3426 박영권 … 1
3011 이수민 … 3
1003 조민희 … 2
3427 최종철 … 3
1365 김상원 … 1
4377 이성래 … 2
DEPARTMENT
DEPTNO
DEPTNAME
FLOOR
1 영업 8
2 기획 10
3 개발 9
4 총무 7
2106 김창섭 … 2
=
2 기획 10
2106 김창섭 … 2 기획 10

# 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 |

# SELECT문(계속)

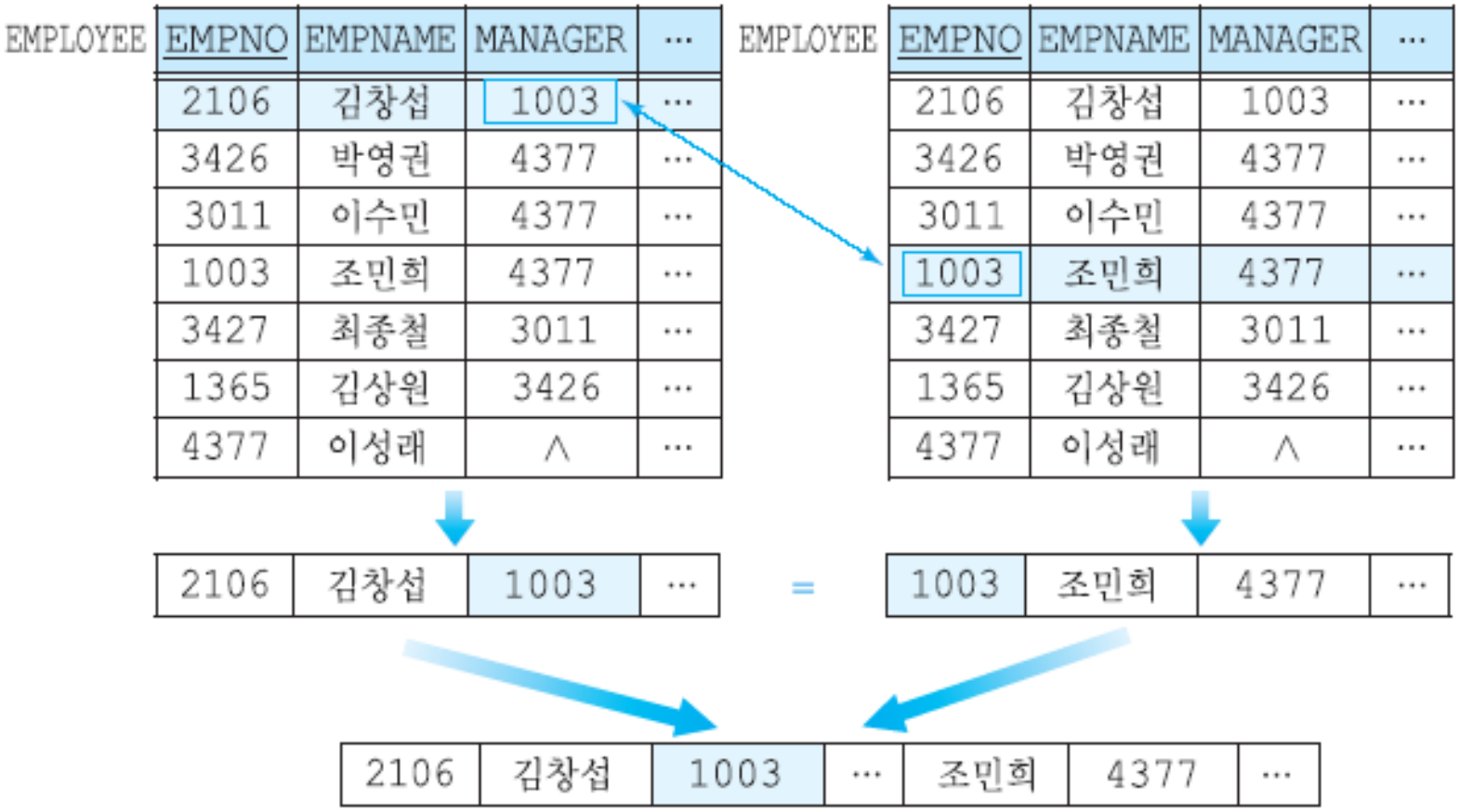
❑ 자체 조인(self join)

- ✓ 한 릴레이션에 속하는 투플을 동일한 릴레이션에 속하는 투플들과 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.

# SELECT문(계속)


EMPLOYEE
EMPNO
EMPNAME
MANAGER
…
2106 김창섭 1003 …
3426 박영권 4377 …
3011 이수민 4377 …
1003 조민희 4377 …
3427 최종철 3011 …
1365 김상원 3426 …
4377 이성래 ∧ …
EMPLOYEE
EMPNO
EMPNAME
MANAGER
…
2106 김창섭 1003 …
3426 박영권 4377 …
3011 이수민 4377 …
1003 조민희 4377 …
3427 최종철 3011 …
1365 김상원 3426 …
4377 이성래 ∧ …
2106 김창섭 1003 …
=
1003 조민희 4377 …
2106 김창섭 1003 … 조민희 4377 …

# SELECT문(계속)

최종 결과 릴레이션은 아래와 같다.

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

# 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.

[실행 결과]

오름차순

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

내림차순 (개발)

내림차순 (기획)

내림차순 (영업)

# SELECT문(계속)

- 중첩 질의(nested query)
  - ✓ 외부 질의의 WHERE절에 다시 SELECT ... FROM ... WHERE 형태로 포함된 SELECT문
  - ✓ 부질의(subquery)라고 함
  - ✓ INSERT, DELETE, UPDATE문에도 사용될 수 있음
  - ✓ 중첩 질의의 결과로 한 개의 스칼라값(단일 값), 한 개의 애트리뷰트로 이루어진 릴레이션, 여러 애트리뷰트로 이루어진 릴레이션이 반환될 수 있음

# SELECT문(계속)


[그림 4.10] 중첩 질의의 구조

# SELECT문(계속)

❑ 한 개의 스칼라값이 반환되는 경우

**예 : 한개의 스칼라 값이 반환되는 경우**

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

[실행 결과]

| EMPNAME | TITLE |
|---|---|
| 박영권 | 과장 |
| 조민희 | 과장 |

# SELECT문(계속)

- 한 개의 애트리뷰트로 이루어진 릴레이션이 반환되는 경우
  - ✓ 중첩 질의의 결과로 한 개의 애트리뷰트로 이루어진 다수의 투플들이 반환될 수 있음
  - ✓ 외부 질의의 WHERE절에서 IN, ANY(SOME), ALL, EXISTS와 같은 연산자를 사용해야 함
  - ✓ 키워드 IN은 한 애트리뷰트가 값들의 집합에 속하는가를 테스트할 때 사용됨
  - ✓ 한 애트리뷰트가 값들의 집합에 속하는 하나 이상의 값들과 어떤 관계를 갖는가를 테스트하는 경우에는 ANY를 사용
  - ✓ 한 애트리뷰트가 값들의 집합에 속하는 모든 값들과 어떤 관계를 갖는가를 테스트하는 경우에는 ALL을 사용

# SELECT문(계속)

예 : IN

(3426 IN
| 2106 |
| --- |
| 3426 |
| 3011 |
)은 참이다.

(1365 IN
| 2106 |
| --- |
| 3426 |
| 3011 |
)은 거짓이다.

(1365 NOT IN
| 2106 |
| --- |
| 3426 |
| 3011 |
)은 참이다.

# SELECT문(계속)

**예 : ANY**

| |
|---|
| 2500000 |
| 3000000 |
| 4000000 |

(3000000 < ANY ) 은 참이다.

| |
|---|
| 2500000 |
| 3000000 |
| 4000000 |

(4000000 < ANY ) 은 거짓이다.

# SELECT문(계속)

예 : ALL

(3000000 < ALL | 2500000<br>3000000<br>4000000 | )은 거짓이다.

(1500000 < ALL | 2500000<br>3000000<br>4000000 | )은 참이다.

(3000000 = ALL | 2500000<br>3000000<br>4000000 | )은 거짓이다.

(1500000 <> ALL | 2500000<br>3000000<br>4000000 | )은 참이다.

# SELECT문(계속)

**예 : IN을 사용한 질의**

질의: 영업부나 개발부에 근무하는 사원들의 이름을 검색하라.

# 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.

[실행 결과]

| EMPNAME |
| --- |
| 박영권 |
| 이수민 |
| 최종철 |
| 김상원 |

# SELECT문(계속)

- 여러 애트리뷰트들로 이루어진 릴레이션이 반환되는 경우
  - ✓ 중첩 질의의 결과로 여러 애트리뷰트들로 이루어진 릴레이션이 반환되는 경우에는 EXISTS 연산자를 사용하여 중첩 질의의 결과가 빈 릴레이션인지 여부를 검사함
  - ✓ 중첩 질의의 결과가 빈 릴레이션이 아니면 참이 되고, 그렇지 않으면 거짓

# SELECT문(계속)

**예 : EXISTS를 사용한 질의**

질의: 영업부나 개발부에 근무하는 사원들의 이름을 검색하라.

[실행 결과]

| EMPNAME |
|---|
| 박영권 |
| 이수민 |
| 최종철 |
| 김상원 |

# SELECT문(계속)

- 상관 중첩 질의(correlated nested query)
  - ✓ 중첩 질의의 WHERE절에 있는 프레디키트에서 외부 질의에 선언된 릴레이션의 일부 애트리뷰트를 참조하는 질의
  - ✓ 중첩 질의의 수행 결과가 단일 값이든, 하나 이상의 애트리뷰트로 이루어진 릴레이션이든 외부 질의로 한 번만 결과를 반환하면 상관 중첩 질의가 아님
  - ✓ 상관 중첩 질의에서는 외부 질의를 만족하는 각 투플이 구해진 후에 중첩 질의가 수행되므로 상관 중첩 질의는 외부 질의를 만족하는 투플 수만큼 여러 번 수행될 수 있음

# SELECT문(계속)

**예 : 상관 중첩 질의**

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

[실행 결과]

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