

제 2장 관계형 데이터베이스

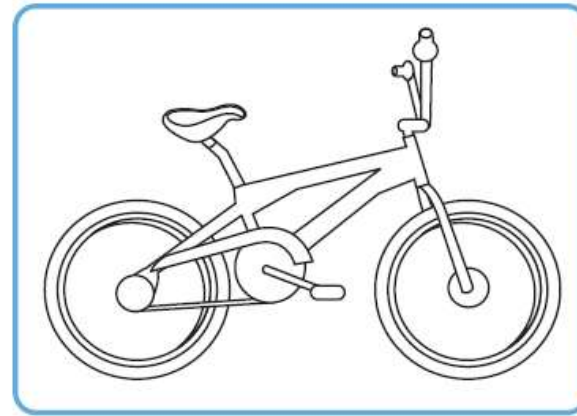
- 관계형 데이터 모델
- 관계형 데이터베이스
- 관계 대수

데이터 모델이란?

- ▶ 물리적 혹은 추상적으로 존재하는 현실세계를 단순화되고 정형화된 형태로 표현하는 하나의 방식 또는 규범



(a) 자전거 사진



(b) 선으로 단순하게 그린 자전거

자전거의 추상화

- ▶ 실제 데이터가 갖는 특성을 살리면서, 목적에 맞게 관심있는 정보만을 단순화 하여 표현하는 방식
 - 데이터에 대한 조작이 가능해야함

릴레이션(relation)의 개념(1)

- ▶ 관계형 데이터 모델(relational data model)
 - ▶ 테이블 형식을 이용하여 데이터들을 정의하고 설명한 모델
 - ▶ 실세계의 데이터를 누구나 직관적으로 이해할 수 있는 형태로 기술할 수 있는 간단한 방식을 제공
 - ▶ 테이블을 릴레이션(relation)이라 부름

이름	전화번호	주소	생일
홍길동	010-1234-5678	서울	3월 15일
이건우	010-2132-2345	서울	8월 23일
이몽룡	010-3245-4368	부산	12월 14일

표 형식으로 기술된 주소록

릴레이션의 개념(2)

- ▶ 릴레이션(relation)

- ▶ 수학적으로, 두 개 이상의 집합으로부터 각 집합을 구성하는 원소들의 순서쌍에 대한 집합을 의미

이름 = {홍길동, 김광식, 박철수, 최용만}

주소 = {서울, 대전, 대구, 부산}

⇒ 순서쌍 : {<홍길동, 서울>, <김광식, 대전>, <박철수, 서울>, <최용만, 부산>}

이름	주소
홍길동	서울
김광식	대구
박철수	서울
최용만	광주

순서쌍을 테이블로 표현한 예

릴레이션의 개념(3)

- ▶ 속성(attribute) – 필드, 컬럼
 - ▶ 릴레이션을 구성하는 각 열(column)의 이름
 - ▶ 예) 주소록 릴레이션을 구성하는 속성
 - 이름, 전화번호, 주소, 생일
- ▶ 튜플(tuple) – 레코드, 행
 - ▶ 릴레이션의 각 행
 - ▶ 예) 주소록 릴레이션의 한 튜플
 - <홍길동, 880-1234, 서울, 3월 15일>
- ▶ 이 책에서는 테이블, 필드, 레코드란 용어를 사용함

릴레이션	테이블
속성	필드(field), 컬럼(column)
튜플	레코드(record), 행(row)

릴레이션의 개념(4)

▶ 도메인(domain)

- ▶ 각 필드에 입력 가능한 값들의 범위, 즉 각 필드가 가질 수 있는 모든 값들의 집합
- ▶ 원자값(atomic value, 더 이상 분리되지 않는 값)이어야 함
- ▶ 예) 주소록의 도메인
 - 이름: 개인 이름들로 구성된 문자열 집합
 - 전화번호: “ddd-ddd-dddd”의 형식으로 구성된 문자열의 집합 (d는 0부터9까지의 숫자)
 - 주소: 도시를 나타내는 문자열의 집합
 - 생일: “dd월dd일”로 구성된 문자열의 집합

▶ 널(null)

- ▶ 특정 필드에 대한 값을 알지 못하거나 아직 정해지지 않아 입력하지 못한 경우의 필드의 값
- ▶ 0이나 공백 문자와는 다름

릴레이션의 개념(5)



테이블 스키마와 테이블 인스턴스

- ▶ 테이블 스키마(table schema, 스키마)
 - ▶ 테이블 정의에 따라 만들어진 데이터 구조
 - $R(A_1, A_2, \dots, A_n)$
 - R : 테이블의 이름
 - A_1, A_2, \dots, A_n : 필드들의 이름
 - 예)
 - 신입생(학번, 주민등록번호, 이름, 주소, 학과명)
- ▶ 차수(degree)
 - ▶ 테이블 스키마에 정의된 필드의 수
 - 차수 = 1 : 단항 테이블(unary relation)
 - 차수 = 2 : 이항 테이블(binary relation)
 - 차수 = n : n 항 테이블(n-ary relation)

테이블 스키마와 테이블 인스턴스(2)

- ▶ 테이블 인스턴스(table instance, 인스턴스)
 - ▶ 테이블 스키마에 현실 세계의 데이터를 레코드로 저장한 형태
 - ▶ 스키마는 한번 정의하면 거의 변함이 없지만 인스턴스는 수시로 바뀔 수 있음 - 레코드의 삽입, 삭제, 수정 등
- ▶ 기수(cardinality)
 - ▶ 테이블 인스턴스의 레코드의 수

학번	주민등록번호	이름	주소	학과명
1292001	900424-1825409	김광식	서울	컴퓨터공학과
1292002	900305-1730021	김정현	서울	컴퓨터공학과
1292003	891021-2308302	김현정	대전	컴퓨터공학과
1292301	890902-2704012	김현정	대구	산업공학과
1292303	910715-1524390	박광수	광주	산업공학과
1292305	921011-1809003	김우주	부산	산업공학과
1292501	900825-1506390	박철수	대전	전자공학과
1292502	911011-1809003	백태성	서울	전자공학과

테이블의 특성

- ▶ 중복된 레코드가 존재하지 않음
 - 테이블 인스턴스는 레코드들의 “집합”임
- ▶ 레코드간의 순서는 의미가 없음
 - 테이블 인스턴스는 레코드들의 “집합”임
 - ‘첫번째 레코드’, ‘두번째 레코드’란 표현은 의미 없음
- ▶ 레코드 내에서 필드의 순서는 의미가 없음
 - 테이블 스키마는 필드들의 집합으로 표현됨
 - ‘첫번째 필드’, ‘두번째 필드’란 표현은 의미 없음
- ▶ 모든 필드는 원자값을 가짐



키(key)

▶ 키는 왜 필요한가?

- ▶ 레코드간의 순서가 의미가 없으므로 레코드를 구분하기 위해서는 각 레코드의 값을 이용함
- ▶ 키(key)
 - 필드들의 일부로 각 레코드들을 유일하게 식별해낼 수 있는 식별자(identifier)
 - 일반적으로 하나의 필드를 지정하여 키로 지정하나, 여러 개의 필드들로 키를 구성할 수도 있음
 - 두 개 이상의 필드로 구성된 키를 복합키(composite key)라고 함
 - 예를 들어 신입생 테이블의 학번 또는 주민등록번호 필드는 각 레코드간에 유일하므로 키가 될 수 있음
 - 그러나 학과명은 키가 될 수 없음
- ▶ 관계형 데이터 모델에서 특정 레코드를 구별하거나 탐색하기 위한 유일한 방법

수퍼키, 후보키, 기본키의 개념

▶ 수퍼키(super key)

- 아무런 제약 조건 없이 레코드들을 식별할 수 있는 필드의 집합
- 예) (주민등록번호) (학번, 주민등록번호) (주민등록번호, 이름) (이름, 주소) 등

▶ 후보키(candidate key)

- 최소한의 필드만으로 구성된 키
- 예) (학번) (주민등록번호)(이름, 주소)(이름, 학과명)

▶ 기본키(primary key)

- 후보키 중에서 식별자로 정의한 하나의 키
- 되도록 하나의 필드로 구성된 후보키를 선정하는 것이 유리함
- 예) (학번)
- 그렇다면, (주민등록번호)(이름, 주소)(이름, 학과명) 모두 후보키 자격이 있는가?
 - 모든 가능한 인스턴스에 대해서도 키가 될 수 있어야 함

키가 널(null)이 될 수 있나?

- ▶ 기본키는 식별자의 기능을 함
- ▶ 기본키로 정의된 필드가 널을 갖게 되면 이러한 식별 기능을 상실
 - ▶ 예를 들어 두 개의 레코드에 대한 기본키 값이 동시에 널이면 그 둘은 서로 구별할 수 없음
 - ▶ 따라서 기본키는 널이 될 수 없음

외래 키(foreign key)(1)

학생 (학번, 주민등록번호, 이름, 주소, 학년, 학과번호)
학과 (학과번호, 학과명, 과사무실)

학번	주민등록번호	이름	학년	학과번호
1292001	900424-1825409	김광식	2	920
1292002	900305-1730021	김정현	2	920
1292003	891021-2308302	김현정	2	920
1292301	890902-2704012	김현정	2	923
1292303	910715-1524390	박광수	1	923
1292305	921011-1809003	김우주	2	923
1292501	900825-1506390	박철수	1	925
1292502	911011-1809003	백태성	2	925

(a) 학생 테이블 인스턴스

학과번호	학과명	과사무실
920	컴퓨터공학과	201호
923	산업공학과	207호
925	전자공학과	308호

(b) 학과 테이블 인스턴스

학생 테이블에서 학과번호가 **930**인 레코드를 삽입할 수 있는가?

외래 키(foreign key)(2)

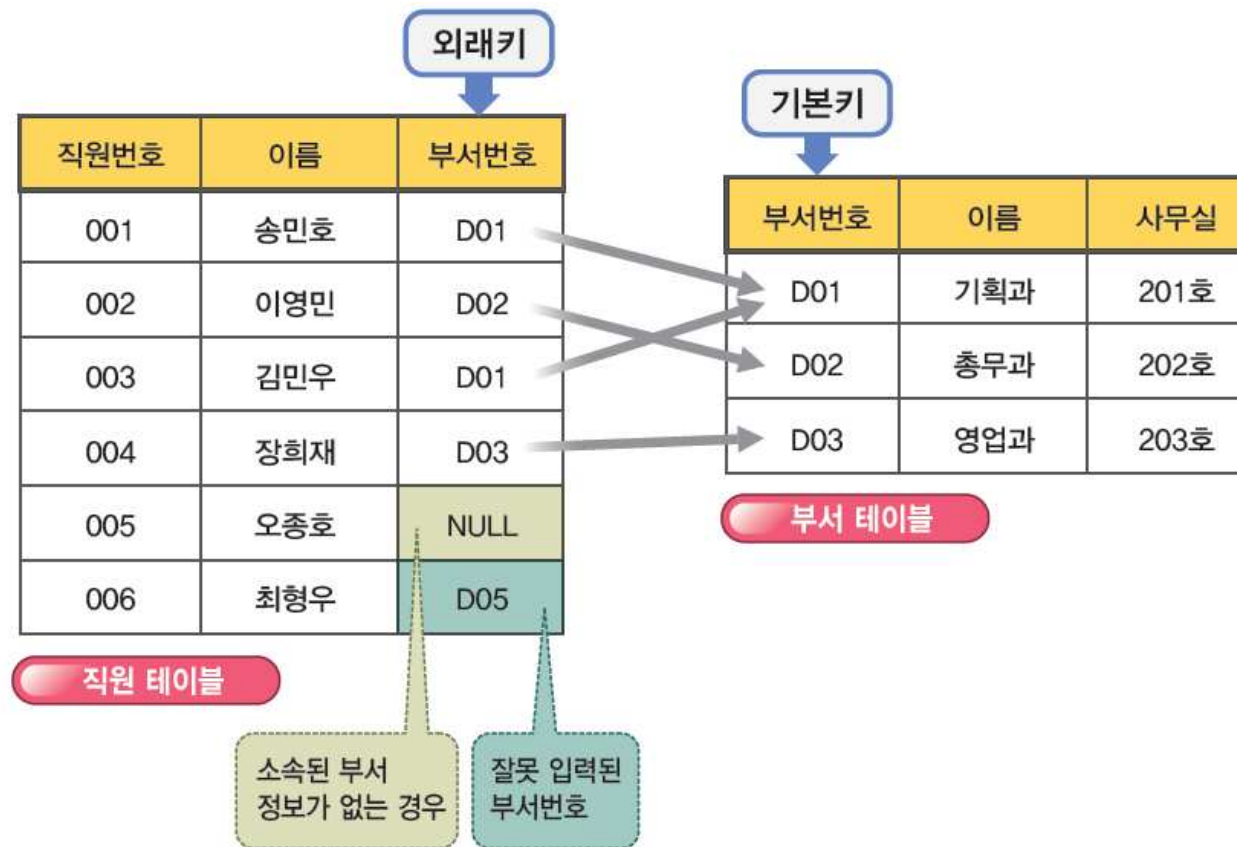
- ▶ 다른 테이블의 기본 키를 참조하는 필드집합
- ▶ 두 테이블 스키마 R_1, R_2 에 대해,
 - ▶ R_1 의 어떤 필드집합 FK가 다음 두 조건을 만족하면, FK는 R_2 의 기본키인 PK를 참조하는 R_1 의 외래키임
 - FK의 필드들은 테이블 스키마 R_2 의 기본 키 PK와 동일한 도메인을 가짐
 - R_1 의 각 레코드의 FK값은 R_2 의 레코드 중 하나의 PK값과 일치하거나 널이 됨
 - ▶ 여기서 R_1 레코드의 FK값이 널이 된다는 것은 알지 못하거나 아직 결정되지 않았다는 것을 의미함

이때, R_1 : 참조하는 테이블(referencing table)

R_2 : 참조되는 테이블(referenced table)

외래 키 (foreign key)(3)

직원(직원번호, 이름, 부서번호)
부서(부서번호, 부서명, 사무실)



외래 키(foreign key)(4)

교수 (교수번호, 주민등록번호, 이름, 학과명, 학과장)



관계형 데이터베이스(relational database)

▶ 정의

- ▶ 관계형 데이터 모델에 기반하여 하나 이상의 테이블로 실세계를 표현한 데이터베이스
 - 실세계를 관계형 데이터 모델이라는 추상적인 도구를 이용하여 표현한 것
 - 테이블들을 컴퓨터의 기억 장치에 어떠한 방법으로 저장할 것인가에 대한 물리적인 구조까지 정의한 것은 아님
- ▶ 관계형 데이터베이스가 하나 이상의 테이블로 구성되어 있을 때
 - 데이터베이스 스키마(database schema)
:테이블 스키마의 집합
 - 데이터베이스 인스턴스(database instance)
:테이블 스키마들에 대한 테이블 인스턴스의 집합

예제: 학사 데이터베이스 (1)

기본키

외래키

student (stu_id, resident_id, name, address, year, dept_id)

department (dept_id, dept_name, office)

professor (prof_id, resident_id, name, dept_id, position, year_emp)

course (course_id, title, credit)

class (class_id, course_id, year, semester, division, prof_id, classroom, enroll)

takes (stu_id, class_id, grade)

예제: 학사 데이터베이스 (2)

stu_id	resident_id	name	year	dept_id
1292001	900424-1825409	김광식	2	920
1292002	900305-1730021	김정현	2	920
1292003	891021-2308302	김현정	2	920
1292301	890902-2704012	김현정	2	923
1292303	910715-1524390	박광수	1	923
1292305	921011-1809003	김우주	2	923
1292501	900825-1506390	박철수	1	925
1292502	911011-1809003	백태성	2	925

(a) student

dept_id	dept_name	office
920	컴퓨터공학과	201호
923	산업공학과	207호
925	전자공학과	308호

(b) department

예제: 학사 데이터베이스 (3)

prof_id	resident_id	name	dept_id	position	year_emp
92001	590327-1839240	이태규	920	교수	1997
92002	690702-1350026	고희석	920	부교수	2003
92301	741011-2765501	최성희	923	부교수	2005
92302	750728-1102458	김태석	923	교수	1999
92501	620505-1200546	박철재	925	조교수	2007
92502	740101-1830264	장민석	925	부교수	2005

(c) professor

course_id	title	credit
C101	전산개론	3
C102	자료구조	3
C103	데이터베이스	4
C301	운영체제	3
C302	컴퓨터구조	3
C303	이산수학	4
C304	객체지향언어	4
C501	인공지능	3
C502	알고리즘	2

(d) course

예제: 학사 데이터베이스 (4)

class_id	course_id	year	semester	division	prof_id	classroom	enroll
C101-01	C101	2012	1	A	92301	301호	40
C102-01	C102	2012	1	A	92001	209호	30
C103-01	C103	2012	1	A	92501	208호	30
C103-02	C103	2012	1	B	92301	301호	30
C501-01	C501	2012	1	A	92501	103호	45
C501-02	C501	2012	1	B	92502	204호	25
C301-01	C301	2012	2	A	92502	301호	30
C302-01	C302	2012	2	A	92501	209호	45
C502-01	C502	2012	2	A	92001	209호	30
C502-02	C502	2012	2	B	92301	103호	26

(e) class

예제: 학사 데이터베이스 (5)

(f) takes

stu_id	class_id	grade
1292001	C101-01	B+
1292001	C103-01	A+
1292001	C301-01	A
1292002	C102-01	A
1292002	C103-01	B+
1292002	C502-01	C+
1292003	C103-02	B
1292003	C501-02	A+
1292301	C102-01	C+
1292303	C102-01	C
1292303	C103-02	B+
1292303	C501-01	A+
1292303	C502-01	B
1292305	C102-01	B
1292305	C103-01	C+
1292305	C501-02	A
1292305	C301-01	A+
1292305	C502-01	A+
1292501	C101-01	B
1292501	C102-01	B
1292501	C501-02	B
1292502	C501-01	C
1292502	C501-02	B

관계 대수 (Relational Algebra)

▶ 질의어(query language)

- ▶ 삽입, 삭제, 수정, 검색 등의 데이터 조작을 위한 연산들을 표현하기 위한 언어

▶ 절차적 언어(procedural language)

- 사용자가 원하는 결과를 얻기 위해 수행되어야 할 일련의 절차를 명시해야 하는 언어
- 예: C, C++와 같은 대부분의 프로그래밍 언어

▶ 비절차적 언어(non-procedural language)

- 수행 절차는 기술하지 않고 사용자가 원하는 결과만을 형식적으로 명시하는 언어
- 실질적 수행절차는 시스템 내부적으로 결정해야 함

관계 대수 (Relational Algebra)(2)

▶ 관계형 데이터베이스에서의 대표적 질의어

- ▶ 관계 대수(relational algebra)
 - 절차적 언어
 - 수학에서의 수식구조와 유사
 - ✓ 피연산자(operand) : 테이블
 - ✓ 연산자(operator)
 - 단항 연산자(unary operator)
 - 이항 연산자(binary operator)
- ▶ 관계 해석(relational calculus)
 - 비절차적 언어
 - 이 책에서는 다루지 않음

관계대수의 연산 종류

▶ 기본 연산

- ▶ 선택 연산
- ▶ 추출 연산
- ▶ 재명명 연산
- ▶ 집합 연산
- ▶ 카티션 프로덕트

▶ 추가연산

- ▶ 조인
- ▶ 자연 조인
- ▶ 외부 조인
- ▶ 지정 연산
- ▶ (나누기 연산)

선택(selection)

- ▶ 하나의 테이블에서 주어진 조건을 만족하는 레코드들을 검색하는 기능

형식 σ <조건식>(<테이블이름>)

- ▶ <테이블이름>
 - <테이블이름>은 연산의 대상이 되는 테이블의 이름
- ▶ <조건식>
 - 비교연산자(<, >, <=, >=, <>)와 부울 연산자(\vee , \wedge , NOT)의 조합

선택(selection)

- ▶ student 테이블에서 address가 '서울'인 레코드를 검색

질의 1

$\sigma_{address = '서울'}(student)$

연산 결과

stu_id	resident_id	name	year	address	dept_id
1292001	900424-1825409	김광식	3	서울	920
1292002	900305-1730021	김정현	3	서울	920
1292502	911011-1809003	백태성	3	서울	925

선택(selection)

- ▶ 2000년 이후에 임용된 교수들에 대한 레코드를 검색

질의 2

$\sigma_{year_emp \geq 2000}(\text{professor})$

연산 결과

prof_id	resident_id	name	dept_id	position	year_emp
92002	690702-1350026	고희석	920	부교수	2003
92301	741011-2765501	최성희	923	부교수	2005
92501	620505-1200546	박철재	925	조교수	2007
92502	740101-1830264	장민석	925	부교수	2005

선택(selection)

- ▶ 2000년 이후에 임용된 '부교수'들의 레코드를 검색

질의 3

$\sigma_{year_emp \geq 2000 \wedge position = '부교수'}(professor)$

연산 결과

prof_id	resident_id	name	dept_id	position	year_emp
92002	690702-1350026	고희석	920	부교수	2003
92301	741011-2765501	최성희	923	부교수	2005
92502	740101-1830264	장민석	925	부교수	2005

선택 연산에서 널(null)의 처리

- ▶ Professor 테이블에 null이 입력된 예

prof_id	resident_id	name	dept_id	position	year_emp
92001	590327-1839240	이태규	920	교수	1997
92002	690702-1350026	고희석	NULL	부교수	2003
92301	741011-2765501	최성희	NULL	부교수	2005
92302	750728-1102458	김태석	923	교수	1999

$\sigma_{dept_id = '920'} \text{ (professor)}$

연산 결과

prof_id	resident_id	name	dept_id	position	year_emp
92001	590327-1839240	이태규	920	교수	1997

$\sigma_{dept_id \neq '920'} \text{ (professor)}$

연산 결과

prof_id	resident_id	name	dept_id	position	year_emp
92302	750728-1102458	김태석	923	교수	1999

- dept_id가 '920'인지 그렇지 않은지 알지 못함 따라서 검색 결과에서 배제해야 함

추출(project)

- ▶ 테이블에서 사용자가 원하는 필드만을 결과로 출력하는 연산

형식

π <필드리스트>(<테이블이름>)

- ▶ <테이블이름>
 - <테이블이름>은 연산의 대상이 되는 테이블의 이름
- ▶ <필드리스트>
 - 테이블에서 추출하고자 하는 필드들의 리스트

추출(project)

- ▶ student 테이블에서 stu_id와 name만을 추출

질의 4 $\pi_{stu_id, name}(student)$

연산 결과

stu_id	name
1292001	김광식
1292002	김정현
1292003	김현정
1292301	김현정
1292303	박광수
1292305	김우주
1292501	박철수
1292502	백태성

추출(project)

- ▶ student 테이블에서 dept_id를 추출

질의 5 $\pi_{dept_id} (student)$

- ▶ 중복을 제거해야 함
 - ▶ 관계형 모델은 중복된 레코드들을 허용하지 않음

연산 결과

dept_id
920
923
925

참고:

관계 대수에서는 중복 레코드를 허용하지 않으나 실제 DBMS에서는 대부분 허용함

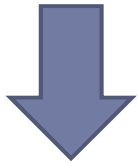
연산자들의 조합

- ▶ 관계 대수 연산자들은 상호 중첩하여 사용 가능
- ▶ 2000년 이후에 임용된 '부교수'들의 레코드를 검색

질의 6

$\sigma_{position = '부교수'}(\sigma_{year_emp \geq 2000}(professor))$

중간 결과



최종 결과

prof_id	resident_id	name	dept_id	position	year_emp
92002	690702-1350026	고희석	920	부교수	2003
92301	741011-2765501	최성희	923	부교수	2005
92501	620505-1200546	박철재	925	조교수	2007
92502	740101-1830264	장민석	925	부교수	2005

prof_id	resident_id	name	dept_id	position	year_emp
92002	690702-1350026	고희석	920	부교수	2003
92301	741011-2765501	최성희	923	부교수	2005
92502	740101-1830264	장민석	925	부교수	2005

연산자들의 조합

- ▶ 선택 연산은 교환 법칙이 성립

$$\sigma_{\langle \text{조건식1} \rangle}(\sigma_{\langle \text{조건식2} \rangle}(\langle \text{테이블이름} \rangle)) \equiv \sigma_{\langle \text{조건식2} \rangle}(\sigma_{\langle \text{조건식1} \rangle}(\langle \text{테이블이름} \rangle))$$

예)

$$\sigma_{\text{year_emp} > 2000}(\sigma_{\text{position} = \text{'부교수'}}(\text{professor})) \equiv \sigma_{\text{position} = \text{'부교수'}}(\sigma_{\text{year_emp} > 2000}(\text{professor}))$$

연산자들의 조합

- ▶ 추출 연산에 대해서 다음의 두 질의는 동일한 결과

$$\pi_{name, position}(\pi_{prof_id, name, position}(professor))$$
$$\pi_{name, position}(professor)$$

- ▶ 다음은 잘못된 질의임

$$\pi_{prof_id, name, position}(\pi_{name, position}(professor))$$

- ▶ 다음의 두 질의가 동일하기 위한 조건은?

$$\pi_{\langle 필드리스트_1 \rangle}(\pi_{\langle 필드리스트_2 \rangle}(\langle 테이블이름 \rangle)) \equiv \pi_{\langle 필드리스트_1 \rangle}(\langle 테이블이름 \rangle)$$

연산자들의 조합

- ▶ 일반적으로는 선택과 추출 연산의 조합으로 질의를 표현
- ▶ 2000년 이후에 임용된 '부교수'들의 이름을 검색

질의 10

$\pi_{name}(\sigma_{year_emp \geq 2000 \wedge position = '부교수' (professor)})$

연산 결과

name
고희석
최성희
장민석

- ▶ 위의 연산에서 선택과 추출 연산의 순서를 바꿀 수 있는가?

재명명 연산

- ▶ 테이블에 이름을 부여하거나 변경하는 연산

형식 ρ <테이블명1>(<테이블명2>)

- ▶ <테이블명2>의 이름을 <테이블명1>로 변경하라는 의미

형식 ρ <테이블명1>(<필드리스트>)(<테이블명2>)

- ▶ <테이블명2>의 이름을 <테이블명1>로 변경하는 동시에 <테이블명2>에 정의된 필드명들을 모두 <필드리스트>로 변경

재명명 연산

- ▶ professor 테이블에서 dept_id가 '920'인 교수들의 이름을 검색

질의 13 $\pi_{com_dept.name}(\rho_{com_dept}(\sigma_{dept_id='920'}(professor)))$

- ▶ 강의실이 '301호'인 class의 prof_id와 enroll을 검색

질의 14 $\rho_{class301(id, number)}(\pi_{prof_id, enroll}(\sigma_{classroom='301'}(class)))$

연산 결과

id	number
92301	40
92301	30
92502	30

- ▶ 주의
 - ▶ 재명명 연산은 중간 결과나 최종 결과에 대한 테이블명이 변경됨
 - ▶ 본래 데이터베이스에 저장된 테이블명까지 변경되는 것은 아님

집합 연산

- ▶ 수학적 집합 이론에서 정의된 연산
 - ▶ 합집합(union)
 - ▶ 차집합(minus)
 - ▶ 카티션 프로덕트(Cartesian product)
 - ▶ 교집합(intersection) → 차집합으로 정의할 수 있음

- ▶ 호환 가능한 테이블들(compatible relations)
 - 합집합, 차집합, 교집합 연산에서 두 피연산자의 차수와 필드 이름들이 동일해야 함
 - 같은 이름의 필드들이라 하더라도 도메인이 일치해야 함

합집합(\cup)

- ▶ 학생 또는 교수들의 이름과 소속 학과번호를 모두 검색

질의 17

$\pi_{name, dept_id}(student) \cup \pi_{name, dept_id}(professor)$

연산 결과

name	dept_id
김광식	920
김정현	920
김현정	920
김현정	923
박광수	923
김우주	923
박철수	925
백태성	925
이태규	920
고희석	920
최성희	923
김태석	923
박철재	925
장민석	925

교집합

- ▶ 교과목 중에서 한번 이상 개설된 과목에 대한 교과목번호를 검색

질의 18 $\pi_{course_id}(course) \cap \pi_{course_id}(class)$

- ▶ 다음 질의와 결과가 동일함

$\pi_{course_id} (class)$

연산 결과

course_id
C101
C102
C103
C301
C302
C501
C502

차집합(-)

- 강좌가 개설되지 않은 과목에 대한 교과목번호를 검색

질의 20

$\pi_{course_id}(course) - \pi_{course_id}(class)$

연산 결과

course_id
C303
C304

카티션 프로덕트(Cartesian product)

- ▶ 두 개의 테이블에서 각각의 레코드들을 서로 결합하여 하나의 레코드로 구성하면서 가능한 모든 조합의 레코드들로 테이블을 생성

형식 <테이블명1> × <테이블명2>

A_1	A_2
a	1
b	2
c	3
d	4

×

A_3	A_4
e	5
f	6
g	7
h	8

=

A_1	A_2	A_3	A_4
a	1	e	5
b	2	e	5
c	3	e	5
d	4	e	5
a	1	f	6
b	2	f	6
c	3	f	6
d	4	f	6
a	1	g	7
b	2	g	7
c	3	g	7
d	4	g	7
a	1	h	8
b	2	h	8
c	3	h	8
d	4	h	8

카티션 프로덕트(Cartesian product)

- ▶ 두 개 이상의 테이블이 필요한 질의의 표현이 가능
- ▶ 교수들의 이름과 소속된 학과 이름을 검색

질의 21

$\pi_{\text{professor.name, department.dept_name}}(\sigma_{\text{professor.dept_id} = \text{department.dept_id}}(\text{professor} \times \text{department}))$

카티션 프로덕트(Cartesian product)

professor × department의 결과

같은 이름의 필드가 있으므로
앞에 소속 테이블 이름을 명시

professor. prof_id	professor. resident_id	professor. name	professor. dept_id	professor. position	professor. year_emp	department. dept_id	department. dept_name	department. office
92001	590327-1839240	이태규	920	교수	1997	920	컴퓨터공학과	201호
92001	590327-1839240	이태규	920	교수	1997	923	산업공학과	207호
92001	590327-1839240	이태규	920	교수	1997	925	전자공학과	308호
92002	690702-1350026	고희석	920	부교수	2003	920	컴퓨터공학과	201호
92002	690702-1350026	고희석	920	부교수	2003	923	산업공학과	207호
92002	690702-1350026	고희석	920	부교수	2003	925	전자공학과	308호
92301	741011-2765501	최성희	923	부교수	2005	920	컴퓨터공학과	201호
92301	741011-2765501	최성희	923	부교수	2005	923	산업공학과	207호
92301	741011-2765501	최성희	923	부교수	2005	925	전자공학과	308호
92302	750728-1102458	김태석	923	교수	1999	920	컴퓨터공학과	201호
92302	750728-1102458	김태석	923	교수	1999	923	산업공학과	207호
92302	750728-1102458	김태석	923	교수	1999	925	전자공학과	308호
92501	620505-1200546	박철재	925	조교수	2007	920	컴퓨터공학과	201호
92501	620505-1200546	박철재	925	조교수	2007	923	산업공학과	207호
92501	620505-1200546	박철재	925	조교수	2007	925	전자공학과	308호
92502	740101-1830264	장민석	925	부교수	2005	920	컴퓨터공학과	201호
92502	740101-1830264	장민석	925	부교수	2005	923	산업공학과	207호
9250	740101-1830264	장민석	925	부교수	2005	925	전자공학과	308호

카티션 프로덕트(Cartesian product)

$\sigma_{\text{professor.dept_id} = \text{department.dept_id}}$ (professor × department)의 결과

professor. prof_id	professor. resident_id	professor. name	professor. dept_id	professor. position	professor. year_emp	department. dept_id	department. dept_name	department. office
92001	590327-1839240	이태규	920	교수	1997	920	컴퓨터공학과	201호
92002	690702-1350026	고희석	920	부교수	2003	920	컴퓨터공학과	201호
92301	741011-2765501	최성희	923	부교수	2005	923	산업공학과	207호
92302	750728-1102458	김태석	923	교수	1999	923	산업공학과	207호
92501	620505-1200546	박철재	925	조교수	2007	925	전자공학과	308호
92502	740101-1830264	장민석	925	부교수	2005	925	전자공학과	308호

카티션 프로덕트(Cartesian product)

최종 연산 결과

professor.name	department.dept_name
이태규	컴퓨터공학과
고희석	컴퓨터공학과
최성희	산업공학과
김태석	산업공학과
박철재	전자공학과
장민석	전자공학과



카티션 프로덕트(Cartesian product)

- ▶ '전산개론'을 수강한 학생들의 학번과 성적을 검색

질의 22

$\pi_{takes.stu_id, takes.grade}$
 $(\sigma_{course.course_id = class.course_id \wedge class.class_id = takes.class_id \wedge course.title = '전산개론'}$
 $((course \times class) \times takes))$

최종 연산 결과

stu_id	grade
1292001	B+
1292501	B

카티션 프로덕트(Cartesian product)

- ▶ 교환법칙과 결합법칙이 성립함

$$R_1 \times R_2 \equiv R_2 \times R_1$$

$$\begin{aligned} R_1 \times (R_2 \times R_3) &\equiv (R_1 \times R_2) \times R_3 \\ &\equiv R_1 \times R_2 \times R_3 \end{aligned}$$

추가연산

- ▶ 조인(세타 조인)
- ▶ 자연 조인
- ▶ 외부 조인
- ▶ 지정 연산

조인(join)

- ▶ 두 테이블로 부터 특정 조건을 만족하는 레코드들을 하나의 레코드로 결합하는 연산
- ▶ 카티션 프로덕트는 모든 가능한 조합에 의해 레코드들을 생성하지만 조인은 특정 조건을 만족하는 레코드만을 선택
- ▶ 세타(theta: Θ) 조인이라고도 함

형식 $\langle \text{테이블이름1} \rangle \bowtie_{\langle \text{조건식} \rangle} \langle \text{테이블이름2} \rangle$

- ▶ <조건식>
- ▶ 조인조건(join condition)이라 부름
 - 조인 조건은 필드간의 동등비교(=)가 대부분이며 이를 동등조인(equijoin)이라 함
- ▶ 다음과 같이 카티션 프로덕트로 표현가능

$$\langle \text{테이블이름1} \rangle \bowtie_{\langle \text{조건식} \rangle} \langle \text{테이블이름2} \rangle \equiv \sigma_{\langle \text{조건식} \rangle} (\langle \text{테이블이름1} \rangle \times \langle \text{테이블이름2} \rangle)$$

조인(join)

- ▶ 학생과 학생이 소속된 학과에 관한 정보를 검색

질의 23

student ⋈_{student.dept_id = department.dept_id} department

연산 결과

student. stu_id	student. resident_id	student. name	student. year	student. address	student. dept_id	department. dept_id	department. dept_name	department. office
1292001	900424-1825409	김광식	3	서울	920	920	컴퓨터공학과	201호
1292002	900305-1730021	김정현	3	서울	920	920	컴퓨터공학과	201호
1292003	891021-2308302	김현정	4	대전	920	920	컴퓨터공학과	201호
1292301	890902-2704012	김현정	2	대구	923	923	산업공학과	207호
1292303	910715-1524390	박광수	3	광주	923	923	산업공학과	207호
1292305	921011-1809003	김우주	4	부산	923	923	산업공학과	207호
1292501	900825-1506390	박철수	3	대전	925	925	전자공학과	308호
1292502	911011-1809003	백태성	3	서울	925	925	전자공학과	308호

조인(join)

- ▶ 교수들의 이름과 소속된 학과 이름을 검색

질의 21 $\pi_{professor.name, department.dept_name}(\sigma_{professor.dept_id = department.dept_id} (professor \times department))$

질의 24 $\pi_{professor.name, department.dept_name}(professor \bowtie_{professor.dept_id = department.dept_id} department)$

- ▶ '전산개론'을 수강한 학생들의 학번과 성적을 검색

질의 22 $\pi_{takes.stu_id, takes.grade}$
 $(\sigma_{course.course_id = class.course_id \wedge class.class_id = takes.class_id \wedge course.title = '전산개론'}$
 $((course \times class) \times takes))$

질의 25 $\pi_{takes.stu_id, takes.grade}$
 $(\sigma_{course.title = '전산개론'}$
 $((course \bowtie_{course.course_id = class.course_id} class) \bowtie_{class.class_id = takes.class_id} takes))$

자연 조인(natural join)

- ▶ 서로 다른 테이블에서 같은 이름을 갖는 두 필드에 대한 동등 조인 중 하나의 필드를 제거하여 단순히 표현한 연산

형식 <테이블이름1> ⋈ <테이블이름2>

- ▶ 다음이 성립함

$$R_1 \bowtie R_2 \equiv \pi_{R_1 \cup R_2} (\sigma_{R_1.A_1 = R_2.A_1 \wedge R_1.A_2 = R_2.A_2 \wedge \dots \wedge R_1.A_n = R_2.A_n} (R_1 \times R_2))$$

- ▶ $R_1 \cup R_2$: 필드들의 합집합
- ▶ A_1, A_2, \dots, A_n : 공통 필드

- ▶ 공통되는 필드가 없으면 카티션 프로덕트와 같음

자연 조인(natural join)

▶ 조인의 예

student ⋈ department

전체적으로 동일한 필드가 없으므로
필드명 앞에 소속 테이블을 명시할
필요가 없음

연산 결과

공통 필드가 하나만 나타남

stu_id	resident_id	name	year	address	dept_id	dept_name	office
1292001	900424-1825409	김광식	3	서울	920	컴퓨터공학과	201호
1292002	900305-1730021	김정현	3	서울	920	컴퓨터공학과	201호
1292003	891021-2308302	김현정	4	대전	920	컴퓨터공학과	201호
1292301	890902-2704012	김현정	2	대구	923	산업공학과	207호
1292303	910715-1524390	박광수	3	광주	923	산업공학과	207호
1292305	921011-1809003	김우주	4	부산	923	산업공학과	207호
1292501	900825-1506390	박철수	3	대전	925	전자공학과	308호
1292502	911011-1809003	백태성	3	서울	925	전자공학과	308호

자연 조인(natural join)

- ▶ '데이터베이스' 과목이 개설된 연도, 학기, 분반 정보를 검색

질의 27 $\pi_{year, semester, division} (\sigma_{title = '데이터베이스'} (course \bowtie class))$

- ▶ 다음 질의들과 동일

질의 28 $\pi_{year, semester, division} (\sigma_{course.course_id = class.course_id \wedge title = '데이터베이스'} (course \times class))$

질의 29 $\pi_{year, semester, division} (\sigma_{title = '데이터베이스'} (course \bowtie_{course.course_id = class.course_id} class))$

연산 결과

year	semester	division
2012	1	A
2012	1	B

외부 조인(outer join)

▶ 자연 조인의 예

name	address		name	dept_name	=	name	address	dept_name
김광식	서울		김광식	컴퓨터공학과		김광식	서울	컴퓨터공학과
김현정	대전		김현정	산업공학과		김현정	대전	산업공학과
조영수	대전		이진영	전자공학과				

- ▶ '조영수'와 '이진영'은 서로 일치되는 레코드가 없어 검색 결과에서 배제됨

▶ 외부 조인

- ▶ 조인 조건에 만족되지 않은 레코드까지 검색 결과에 포함시키기 위한 방법
- ▶ 서로 매치되지 않는 필드에 대해서는 NULL을 입력함
- ▶ 종류

- ▶ 왼쪽 외부조인(left outer join)




- ▶ 오른쪽 외부조인(right outer join)



- ▶ 완전 외부조인(full outer join)



외부 조인(outer join)

freshmen  member

freshmen  member

freshmen  member

연산 결과

name	address	dept_name
김광식	서울	컴퓨터공학과
김현정	대전	산업공학과
조영수	대전	NULL

name	address	dept_name
김광식	서울	컴퓨터공학과
김현정	대전	산업공학과
이진영	NULL	전자공학과

name	address	dept_name
김광식	서울	컴퓨터공학과
김현정	대전	산업공학과
조영수	대전	NULL
이진영	NULL	전자공학과

지정 (assignment) 연산

- ▶ 복잡한 질의를 여러 개의 질의로 분리하거나 중간 결과에 이름을 부여
- ▶ 최종 질의를 결과에 이름을 부여
- ▶ 연산 기호로는 \leftarrow 를 사용
- ▶ 예) student 테이블에서 3학년인 학생을 선택해서 그 결과 테이블을 junior이라는 이름으로 지정

질의 33 $\text{junior} \leftarrow \sigma_{\text{year} = 3} (\text{student})$

- ▶ 다음의 두 질의는 동일

질의 29 $\pi_{\text{year}, \text{semester}, \text{division}} (\sigma_{\text{title} = \text{'데이터베이스'}} (\text{course} \bowtie_{\text{course.course_id} = \text{class.course_id}} \text{class}))$

질의 34 $\text{temp1} \leftarrow \text{course} \bowtie_{\text{course.course_id} = \text{class.course_id}} \text{class}$
 $\text{temp2} \leftarrow \sigma_{\text{title} = \text{'데이터베이스'}} (\text{temp1})$
 $\pi_{\text{year}, \text{semester}, \text{division}} (\text{temp2})$