# 데이터베이스 및 설계

## Chap 5. 관계 대수와 관계 해석

2012.04.17.

오 병 우

컴퓨터공학과

kit 국립금오공과대학교
Kumoh National Institute of Technology

# 관계 데이터 연산

- 데이터 모델 (D)

  D = <S, **O**, C>

  - S: 구조, O: 연산, C: 제약 조건
- 연산과 데이터 언어
  - 연산 : 시스템 입장
  - 데이터 언어 : 사용자 입장
- 관계 데이터 언어

  i . 관계 대수(relational algebra)
  - 절차 언어 : how

  ii. 관계 해석(relational calculus)
  - 비 절차 언어 : what
  - 투플 관계해석
  - 도메인 관계해석
- 관계 해석과 관계 대수는 표현이나 기능 면에서 동등

# Relational Algebra & Calculus

- Relational Algebra (관계 대수)
  - ***Algebra*** *is a type of mathematics in which letters are used to represent possible quantities.*
  - Procedural language (절차 언어) : How
- Relational Calculus (관계 해석)
  - ***Calculus*** *is a branch of advanced mathematics which deals with variable quantities.*
  - Nonprocedural language (비절차 언어) : What
  - 뭘 원하는 지만 기술하면 되므로 사용 편이

# Relationally Complete

- 관계 해석으로 표현할 수 있는 것은 모두 관계 대수로도 표현 가능
- a language: relationally complete (완전성)
  - at least as powerful as the relation algebra
    - i.e., its expressions permit the definition of every relation that can be defined by means of expressions of the algebra
  - 어떤 데이터 언어가 relational calculus가 표현할 수 있는 모든 질의를 표현할 수 있을 때 relationally complete 하다고 함 (또는 relational completeness)

# Relational Algebra

kit 국립금오공과대학교
Kumoh National Institute of Technology

# Introduction of Relational Algebra

- Manipulative part of the relational model
  - Relational algebra: a set of operators
  - Assignment operation: named relation := expression of the algebra
- Relational algebra (8 operators)
  - A collection of high-level operators that operate on relations
    - 릴레이션: 투플의 집합
  - One or two relations (input) → a new relation (output)
  - Traditional set operations (집합 연산)
    - Union, intersection, difference, and Cartesian product
  - Special relational operations
    - Restrict (select), project, join, and divide
- 수학의 Closure property (정수의 덧셈 결과는 정수)
  - Relations are closed under the algebra
  - 피연산자와 연산 결과가 모두 릴레이션
  - Nested relational expressions 가능
    - Expressions in which the operands are themselves represented by expressions (not names)

# What is the Algebra for?

- Algebra expression
  - a high-level and symbolic representation of the user's intent
  - basis for query optimization
    - (ex) (( S JOIN SP ) WHERE P#='P2') [SNAME]

      ⇒ (S JOIN (SP WHERE P#='P2')) [SNAME]
      - 줄여 놓은 다음에 Join하여 질의 처리 시간을 줄일 수 있음
- Applications of algebraic expression
  - defining a scope for retrieval
  - defining a scope for update (insert, modify, delete)
  - defining virtual data (i.e., view)
  - defining snapshot data
  - defining access rights (authorization)
  - defining stability requirement (concurrency control)
  - defining integrity constraints

# Traditional Set Operations

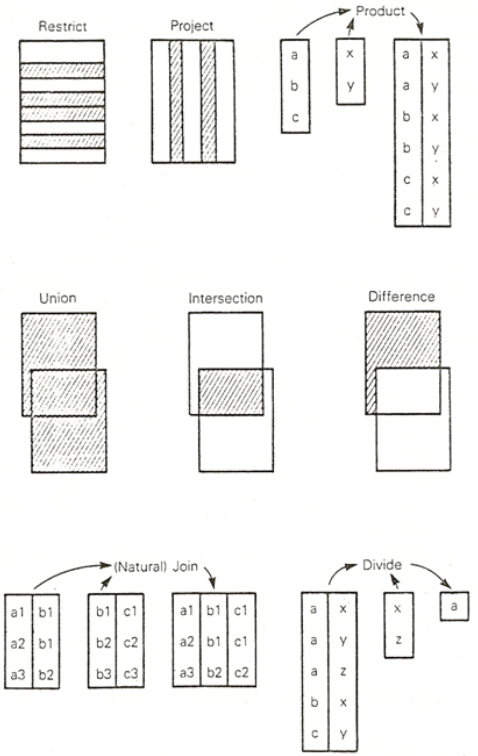
- Two relations: union–compatibility (합병 가능)
  - They have identical (동일한) headings (스키마)
    1. They each have the same set of attribute names (same degree)
    2. Corresponding attributes are defined on the same domain
- Traditional set operations: two operands A, B
  - union(∪) : Union-compatible

    A UNION B = {t: t∈A ∨ t∈B}
  - intersection(∩) : Union-compatible

    A INTERSECT B = {t: t∈A ∧ t∈B}
  - difference ( - ) : Union-compatible

    A MINUS B = {t: t∈A ∧ t∉B}
  - extended Cartesian product (X)
    - Concatenation ( · )

    A TIMES B = {t · s: t∈A ∧ s∈B}



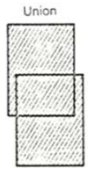
# 일반 집합 연산자

i. 합집합 (union, $\cup$)

$R \cup S = \{ t \mid t \in R \vee t \in S \}$

$|R \cup S| \leq |R| + |S|$

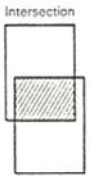
Cardinality (투플의 개수)




ii. 교집합 (intersect, $\cap$)

$R \cap S = \{ t \mid t \in R \wedge t \in S \}$

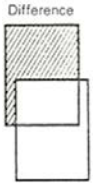
$|R \cap S| \leq \min\{ |R|, |S| \}$




iii. 차집합 (difference, -)

$R - S = \{ t \mid t \in R \wedge t \notin S \}$

$|R - S| \leq |R|$




iv. 카티션 프로덕트 (cartesian product, $\times$)

$R \times S = \{ r \cdot s \mid r \in R \wedge s \in S \}$

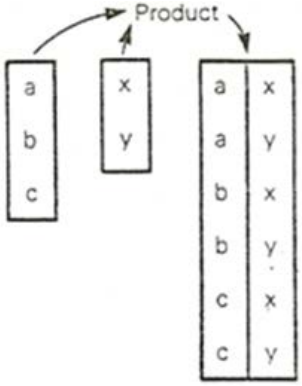
Concatenation of $t=(A_1:a_1, A_2:a_2, \cdots, A_m:a_m)$, $s=(B_1:b_1, B_2:b_2, \cdots, B_n:b_n)$

- $t \cdot s = (A_1:a_1, A_2:a_2, \cdots, A_m:a_m, B_1:b_1, B_2:b_2, \cdots, B_n:b_n)$

(m+n)-th attribute

$|R \times S| = |R| \times |S|$

차수(degree) = R의 차수 + S의 차수



# Associative and Commutative Operations

- UNION, INTERSECT, TIMES (not MINUS)
  - associative

    (A UNION B) UNION C = A UNION (B UNION C)

    = A UNION B UNION C

    – ∪, ∩, × 연산은 결합적(associative)임

    $R \cup S \cup T = (R \cup S) \cup T = R \cup (S \cup T)$

    $R \cap S \cap T = (R \cap S) \cap T = R \cap (S \cap T)$

    $R \times S \times T = (R \times S) \times T = R \times (S \times T)$
  - commutative

    A UNION B = B UNION A

    – ∪, ∩, × 연산은 교환적(commutative)임

    $R \cup S = R \cup S$

    $R \cap S = R \cap S$

    $R \times S = R \times S$

순서에 상관없이 결과가 동일한

# Examples

- Union, intersection, difference and Cartesian product

A

| S# | SNAME | STATUS | CITY |
|---|---|---|---|
| S1 | Smith | 20 | London |
| S4 | Clark | 20 | London |

B

| S# | SNAME | STATUS | CITY |
|---|---|---|---|
| S1 | Smith | 20 | London |
| S2 | Jones | 10 | Paris |

(a) Union (A UNION B)

| S# | SNAME | STATUS | CITY |
|---|---|---|---|
| S1 | Smith | 20 | London |
| S4 | Clark | 20 | London |
| S2 | Jones | 10 | Paris |

(b) Intersection (A INTERSECT B)

| S# | SNAME | STATUS | CITY |
|---|---|---|---|
| S1 | Smith | 20 | London |

(c) Difference (A MINUS B)

| S# | SNAME | STATUS | CITY |
|---|---|---|---|
| S4 | Clark | 20 | London |

Difference (B MINUS A)

| S# | SNAME | STATUS | CITY |
|---|---|---|---|
| S2 | Jones | 10 | Paris |

A

| S# |
|---|
| S1 |
| S2 |
| S3 |
| S4 |
| S5 |

B

| P# |
|---|
| P1 |
| P2 |
| P3 |
| P4 |
| P5 |
| P6 |

Cartesian product (A TIMES B)

| S# | P# |
|---|---|
| S1 | P1 |
| S1 | P2 |
| S1 | P3 |
| S1 | P4 |
| S1 | P5 |
| S1 | P6 |
| S2 | P1 |
| S2 | P2 |
| S2 | P3 |
| S2 | P4 |
| S2 | P5 |
| S2 | P6 |
| S3 | P1 |
| S3 | P2 |
| S3 | P3 |
| S3 | P4 |
| S3 | P5 |
| S3 | P6 |
| S4 | P1 |
| S4 | P2 |
| S4 | P3 |
| S4 | P4 |
| S4 | P5 |
| S4 | P6 |
| S5 | P1 |
| S5 | P2 |
| S5 | P3 |
| S5 | P4 |
| S5 | P5 |
| S5 | P6 |

# 표기법

- 릴레이션 : $R(X) = R(A_1, \ldots, A_n)$
  - R의 투플 : $r = \langle a_1, \ldots, a_n \rangle$

    $R=\{r \mid r = \langle a_1, \ldots, a_n \rangle\}$
  - 투플 r에 대한 애트리뷰트 $A_i$의 값
    - $r.A_i$ 또는 $a_i$
    - $r.A_i = r[A_i] = a_i$
    - 다음 관계가 성립

      $\langle r.A_1, r.A_2, \ldots, r.A_n \rangle = \langle r[A_1], r[A_2], \ldots, r[A_n] \rangle = r[A_1, A_2, \ldots A_n] = r[X]$

# Restriction (select) : σ (sigma)

- Restriction (theta-selection)
  - R(U): a relation
    - A, B ⊆U: attribute defined on the same domain
    - θ (theta) ∈ {<, ≤, >, ≥, =, ≠}
    - v : literal value

$$\sigma_{A\theta\, v}(R) = \{ r \mid r \in R \wedge r.A\ \theta\ v \}$$

$$\sigma_{A\theta B}(R) = \{ r \mid r \in R \wedge r.A\ \theta\ r.B \}$$

  - horizontal subset of R
    - 선택 조건을 만족하는 릴레이션의 수평적 부분집합
  - Boolean combination of simple comparisons
    - R WHERE C1 AND C2 ≡ (R WHERE C1) INTERSECT (R WHERE C2)
    - R WHERE C1 OR C2 ≡ (R WHERE C1) UNION (R WHERE C2)
    - R WHERE NOT C ≡ R MINUS (R WHERE C)


조건식 (predicate)
중간성적 > 30

조건식 (predicate)
중간성적 < 기말성적

데이터 언어식 표현

# Restriction (select)

- Example
  - $\sigma_{\text{학과 = '컴퓨터'}}$ (학생)
  - $\sigma_{\text{학번 = 300 } \wedge \text{ 과목번호='C312'}}$(등록)
  - $\sigma_{\text{중간성적 < 기말성적}}$ (등록)

- 데이터 언어식 표현
  R WHERE 조건식

- $\sigma_{\text{조건2}}(\sigma_{\text{조건1}}(R)) = \sigma_{\text{조건1}}(\sigma_{\text{조건2}}(R))$
  $= \sigma_{\text{조건1} \wedge \text{ 조건2}}(R)$
- 선택도(selectivity) :
  - 선택 조건에 의해 선택된 투플의 비율
  - Query optimization: 선택도가 적은 것부터 수행

# 예제

데이터
언어식 표현

교재 p.86
또는
슬라이드
pp.16-17
참조

$\sigma_{학과='컴퓨터'}(학생)$

학생 WHERE 학과='컴퓨터'

| 학번 | 이름 | 학년 | 학과 |
|---|---|---|---|
| 100 | 나수영 | 4 | 컴퓨터 |
| 300 | 정기태 | 1 | 컴퓨터 |
| 400 | 송병길 | 4 | 컴퓨터 |

$\sigma_{학번=300 \wedge 과목번호='C312'}(등록)$

등록 WHERE 학번=300
AND 과목번호='C312'

| 학번 | 과목번호 | 성적 | 중간성적 | 기말성적 |
|---|---|---|---|---|
| 300 | C312 | A | 90 | 95 |

$\sigma_{중간성적 < 기말성적}(등록)$

등록 WHERE 중간성적 < 기말성적

| 학번 | 과목번호 | 성적 | 중간성적 | 기말성적 |
|---|---|---|---|---|
| 100 | C413 | A | 90 | 95 |
| 300 | C312 | A | 90 | 95 |
| 400 | C312 | A | 90 | 95 |
| 400 | C413 | B | 80 | 85 |
| 400 | E412 | C | 65 | 75 |

# example

- 대학(University) 관계 데이터베이스

학생 (STUDENT)

| 학번 (Sno) | 이름 (Sname) | 학년 (Year) | 학과 (Dept) |
|---|---|---|---|
| 100 | 나 수 영 | 4 | 컴퓨터 |
| 200 | 이 찬 수 | 3 | 전기 |
| 300 | 정 기 태 | 1 | 컴퓨터 |
| 400 | 송 병 길 | 4 | 컴퓨터 |
| 500 | 박 종 화 | 2 | 산공 |

과목 (COURSE)

| 과목번호 (Cno) | 과목이름 (Cname) | 학점 (Credit) | 학과 (Dept) | 담당교수 (PRname) |
|---|---|---|---|---|
| C123 | 프로그래밍 | 3 | 컴퓨터 | 김성국 |
| C312 | 자료 구조 | 3 | 컴퓨터 | 황수관 |
| C324 | 화일 구조 | 3 | 컴퓨터 | 이규찬 |
| C413 | 데이터베이스 | 3 | 컴퓨터 | 이일로 |
| E412 | 반 도 체 | 3 | 전자 | 홍봉진 |

# example

- 대학(University) 관계 데이터베이스(cont'd)

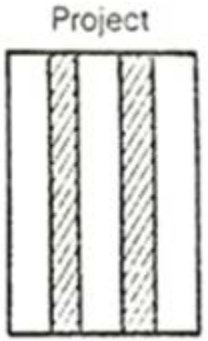
등록 (ENROL)

| 학번 (Sno) | 과목번호 (Cno) | 성적 (Grade) | 중간성적 (Midterm) | 기말성적 (Final) |
|---|---|---|---|---|
| 100 | C413 | A | 90 | 95 |
| 100 | E412 | A | 95 | 95 |
| 200 | C123 | B | 85 | 80 |
| 300 | C312 | A | 90 | 95 |
| 300 | C324 | C | 75 | 75 |
| 300 | C413 | A | 95 | 90 |
| 400 | C312 | A | 90 | 95 |
| 400 | C324 | A | 95 | 90 |
| 400 | C413 | B | 80 | 85 |
| 400 | E412 | C | 65 | 75 |
| 500 | C312 | B | 85 | 80 |

# Projection : Π (pi)

- Projection
  - 릴레이션 R(X)에서 $Y \subseteq X$ 이고 $Y=\{B_1,B_2, \ldots ,B_m\}$ 이면,
    $\Pi_Y(R) = \{ <r.B_1, \ldots , r.B_m> \mid r \in R \}$
  - vertical subset of R
    - 릴레이션의 수직적 부분집합
  - duplicate tuples are eliminated
    - 생성된 중복 투플은 제거
  - Example
    - 학생(학번,이름,학년)에서 $\Pi_{\text{이름}}$(학생)
  - 데이터 언어식 표현

    R $[B_1,B_2, \ldots ,B_m]$
    - e.g., (S WHERE CITY = 'Paris') **[ S#, SNAME ]**
  - $\Pi_Y(\Pi_X(R)) = \Pi_Y(R)$



# 예제

데이터
언어식 표현

$\Pi$ 이름,학과(학생)

학생[이름, 학과]

| 이름 | 학과 |
|---|---|
| 나수영 | 컴퓨터 |
| 이찬수 | 전기 |
| 정기태 | 컴퓨터 |
| 송병길 | 컴퓨터 |
| 박종화 | 산공 |

$\Pi$ 과목이름,담당교수(과목)

과목[과목이름, 담당교수]

| 과목이름 | 담당교수 |
|---|---|
| 프로그래밍 | 김성국 |
| 자료구조 | 황수관 |
| 화일구조 | 이규찬 |
| 데이터베이스 | 이일로 |
| 반도체 | 홍봉진 |

# Theta-join : $\bowtie_{A\theta B}$

- Theta-join : not primitive operation
  - R(U), S(V): relations
  - $X \subseteq U$, $Y \subseteq V$: defined on the same domain
  - $\theta \in \{<, \leq, >, \geq, =, \neq\}$
  - R.X theta-join S.Y $= \{r \cdot s: r \in R \wedge s \in S \wedge r[X]\ \theta\ s[Y]\}$

    $\equiv R[X\ \theta\ Y]S$ = (R TIMES S) WHRER X theta Y

    데이터 언어식 표현

    – 예제)
      - ((S RENAME CITY AS SCITY) TIMES (P RENAMES CITY AS PCITY)) WHERE SCITY>PCITY
      - 학생 $\bowtie_{학번=학번}$ 등록의 결과 릴레이션 에서

        학생.학번, 학생.이름, 학생.학년, 학생.학과, 등록.과목번호, 등록.성적, 등록.중간성적, 등록.기말성적
- Equijoin
  - theta($\theta$): “equals”(=)
    - two attributes have identical (동일한) values
  - $R \bowtie_{A=B} S = \{ r \cdot s \mid r \in R \wedge s \in S \wedge ( r.A = s.B ) \}$

# Equijoin 예제

cf.) Natural Join은 1개 (중복 제거)

학생 $\bowtie_{학번=학번}$ 등록

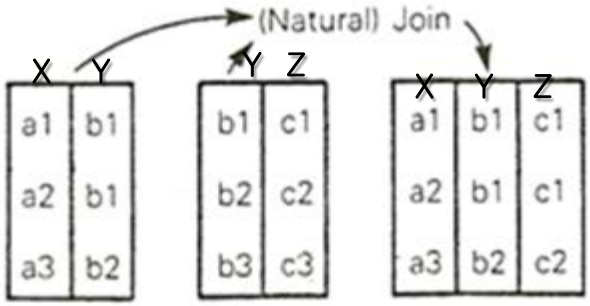
학번이 2개

| 학생.학번 | 이름 | 학년 | 학과 | 등록.학번 | 과목번호 | 성적 | 중간성적 | 기말성적 |
|---|---|---|---|---|---|---|---|---|
| 100 | 나수영 | 4 | 컴퓨터 | 100 | C413 | A | 90 | 95 |
| 100 | 나수영 | 4 | 컴퓨터 | 100 | E412 | A | 95 | 95 |
| 200 | 이찬수 | 3 | 전기 | 200 | C123 | B | 85 | 80 |
| 300 | 정기태 | 1 | 컴퓨터 | 300 | C312 | A | 90 | 95 |
| 300 | 정기태 | 1 | 컴퓨터 | 300 | C324 | C | 75 | 75 |
| 300 | 정기태 | 1 | 컴퓨터 | 300 | C413 | A | 95 | 95 |
| 400 | 송병길 | 4 | 컴퓨터 | 400 | C312 | A | 90 | 95 |
| 400 | 송병길 | 4 | 컴퓨터 | 400 | C324 | A | 95 | 90 |
| 400 | 송병길 | 4 | 컴퓨터 | 400 | C413 | B | 80 | 85 |
| 400 | 송병길 | 4 | 컴퓨터 | 400 | E412 | C | 65 | 75 |
| 500 | 박종화 | 2 | 산공 | 500 | C312 | B | 85 | 80 |

# Natural Join : ⋈ (또는 $\bowtie_N$)

- Natural join
  - R(X, Y) = R($X_1$, $X_2$, ⋯, $X_m$, $Y_1$, $Y_2$, ⋯ , $Y_n$)
  - S(Y, Z) = S($Y_1$, $Y_2$, ⋯ , $Y_n$, $Z_1$, $Z_2$, ⋯ , $Z_p$)
  - R ⋈ S
    - R JOIN S = {x·y·z: r∈R ∧ s∈S ∧ r[X]=x ∧ r[Y] = s[Y] = y ∧ s[Z] = z}

데이터 언어식 표현




  - Associative and commutative
  - R and S: no common attribute names

    ⇒ R JOIN S ≡ R TIMES S

# Natural Join (교재)

- Natural join: $\bowtie_N$
  - R(X), S(Y)의 조인 애트리뷰트를 Z(=X∩Y)라 하면

$R \bowtie_N S$

$= \{ <r \cdot s>[X \cup Y] \mid r \in R \wedge s \in S \wedge r[Z] = s[Z] \}$

$= \Pi_{X \cup Y}(\sigma_{Z=Z}(R \times S))$

$= \Pi_{X \cup Y}(R \bowtie_{Z=Z} S)$

theta-join

  - 즉 equijoin의 결과 릴레이션에서 애트리뷰트의 중복을 제거함
    - 예제) Equijoin: 학생.학번, 등록.학번 (2개)
      Natural Join: 학번 (1개)

# Natural join 예제

학번이 1개

cf.) EquiJoin은 2개 (중복)

학생 $\bowtie_N$ 등록

| 학번 | 이름 | 학년 | 학과 | 과목번호 | 성적 | 중간성적 | 기말성적 |
|---|---|---|---|---|---|---|---|
| 100 | 나수영 | 4 | 컴퓨터 | C413 | A | 90 | 95 |
| 100 | 나수영 | 4 | 컴퓨터 | E412 | A | 95 | 95 |
| 200 | 이찬수 | 3 | 전기 | C123 | B | 85 | 80 |
| 300 | 정기태 | 1 | 컴퓨터 | C312 | A | 90 | 95 |
| 300 | 정기태 | 1 | 컴퓨터 | C324 | C | 75 | 75 |
| 300 | 정기태 | 1 | 컴퓨터 | C413 | A | 95 | 95 |
| 400 | 송병길 | 4 | 컴퓨터 | C312 | A | 90 | 95 |
| 400 | 송병길 | 4 | 컴퓨터 | C324 | A | 95 | 90 |
| 400 | 송병길 | 4 | 컴퓨터 | C413 | B | 80 | 85 |
| 400 | 송병길 | 4 | 컴퓨터 | E412 | C | 65 | 75 |
| 500 | 박종화 | 2 | 산공 | C312 | B | 85 | 80 |

# Division : ÷

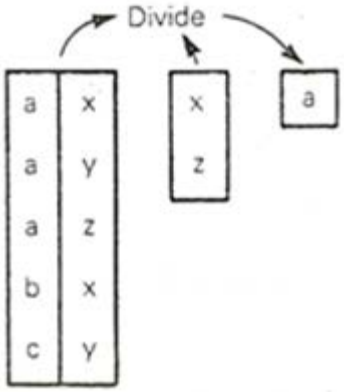
- Division
  - $R(X, Y) = R(X_1, X_2, \cdots, X_m, Y_1, Y_2, \cdots, Y_n)$
  - $S(Y) = S(Y_1, Y_2, \cdots, Y_n)$
  - $R \text{ DIVIDEDBY } S = \{r[X] : r \in R \wedge r[X] \cdot s \in R \text{ for all } s \in S\}$

데이터 언어식 표현

- 교재: 릴레이션 R(X), S(Y) 에 대하여

$Y \subseteq X$이고 $Z = X - Y$이면 $R(X) = R(Z, Y)$

$R \div S = \{ t \mid t \in \Pi_Z(R) \wedge t \cdot s \in R \text{ for all } s \in S \}$




Note : $(R \div S) \times S \subseteq R$

# Examples of Division

학과목(SC)

| 학번 (Sno) | 과목번호 (Cno) |
|---|---|
| 100 | C413 |
| 100 | E412 |
| 200 | C123 |
| 300 | C312 |
| 300 | C324 |
| 300 | C413 |
| 400 | C312 |
| 400 | C324 |
| 400 | C413 |
| 400 | E412 |
| 500 | C312 |

과목1(C1)

| 과목번호 (Cno) |
|---|
| C413 |

과목2(C2)

| 과목번호 (Cno) |
|---|
| C312 |
| C413 |

과목3(C3)

| 과목번호 (Cno) |
|---|
| C312 |
| C413 |
| E412 |

SC ÷ C1

| 학번 (Sno) |
|---|
| 100 |
| 300 |
| 400 |

SC ÷ C2

| 학번 (Sno) |
|---|
| 300 |
| 400 |

SC ÷ C3

| 학번 (Sno) |
|---|
| 400 |

# RENAME: ρ (Rho)

- 중간 결과 릴레이션에 이름을 지정(예제 1)하거나 애트리뷰트 이름을 변경(예제 2)할 때 사용
  ① $\rho_S(E)$
  관계 대수식 E의 결과 릴레이션의 이름을 S로 지정
  ② $\rho_{S(B_1,B_2,\ldots,B_m)}(E)$
  관계 대수식 E의 결과 릴레이션의 이름을 S로 지정하면서
  애트리뷰트 이름을 각각 $B_1,B_2,\ldots,B_m$ 으로 변경
  ③ $\rho_{(B_1,B_2,\ldots,B_m)}(R)$
  릴레이션 R의 애트리뷰트 이름을 각각 $B_1,B_2,\ldots,B_m$ 으로 변경
- 예제 1: 학생 중에서 컴퓨터 학과 학생의 이름 선택
  - $\Pi_{이름}(\sigma_{학과='컴퓨터'}(학생))$
    - Temp ← $\sigma_{학과='컴퓨터'}(학생)$
    - 컴학생 ← $\Pi_{이름}$(Temp)
- 예제 2: 이름 애트리뷰트를 성명으로 변경
  - 컴학생 (성명)← $\Pi_{이름}$(Temp)

데이터 언어식 표현

# Relational Assignment

- Relational assignment operation
  - to “remember” the value of some algebraic expression
  - to change the database state
    - insert

      S := S UNION { (S# : ‘S6’, SNAME: ‘Baker’, STATUS: 50, CITY: ‘Madrid’) }

    - delete

      SP := SP MINUS { (S# : ‘S1’, P# : ‘P1’, QTY : 300) }

데이터 언어식 표현

데이터 언어식 표현

# Primitive and Composite Operations

- Primitive operations (근원 연산)
  - can not be defined in terms of the others
    - 하나의 논리적 기능을 수행, 다른 연산을 이용하여 표현할 수 없음
  - restriction, projection, product, union, difference
- Composite operations (복합 연산)
  - 근원 연산을 이용하여 표현할 수 있음 (intersection, join, division)

$$R \cap S = R - (R - S) = S - (S - R)$$
$$= (R \cup S) - ((R - S) \cup (S - R))$$

$$R \bowtie_{A\theta B} S = \sigma_{A\theta B}(R \times S)$$

$$R(Z,Y) \div S(Y) = R[Z] - ((R[Z] \times S) - R)[Z]$$

  - 연산력 보다는 표현력 증대
    - 간단하게 표현 가능

# Relational Algebra의 확장

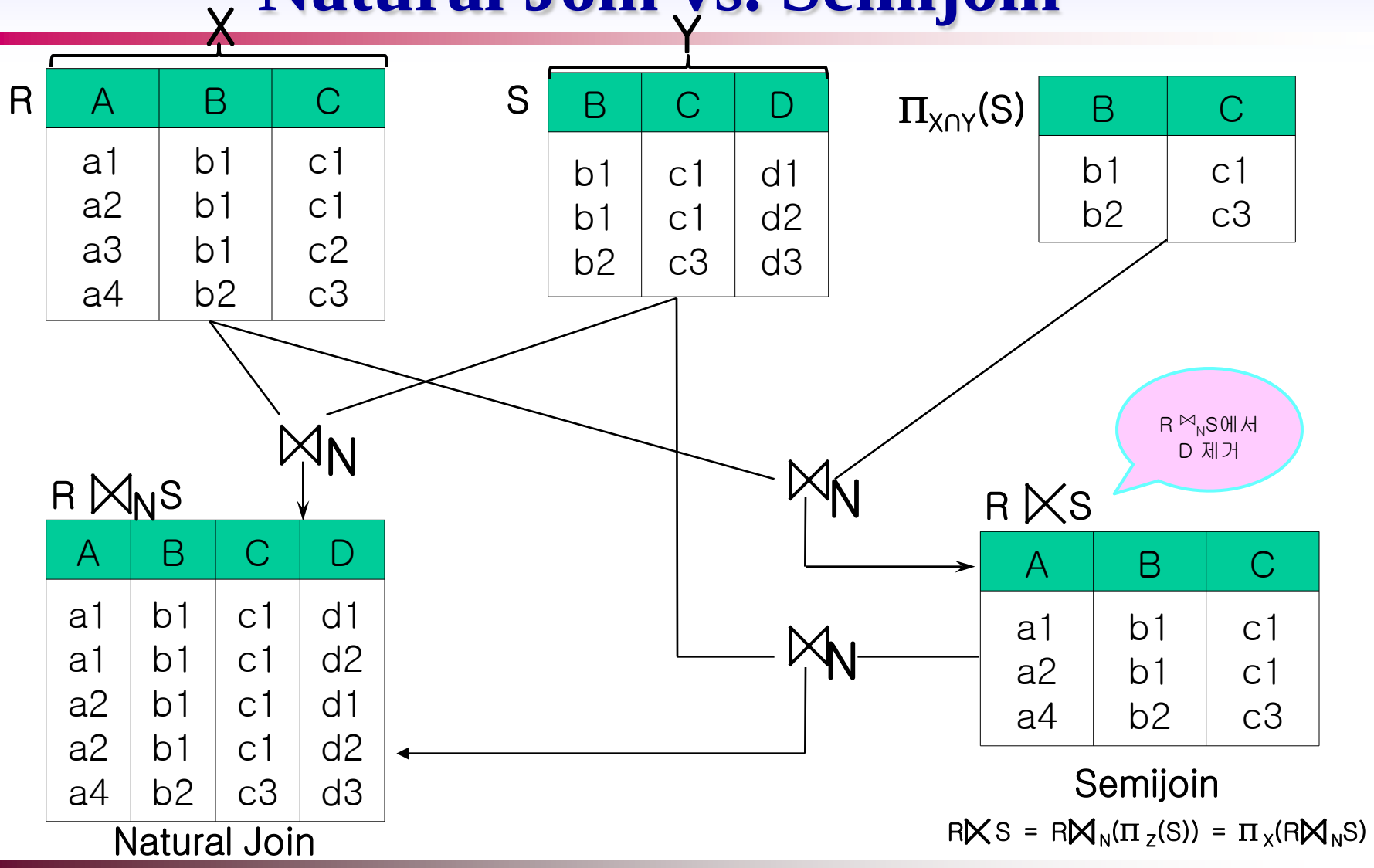
- Semijoin
- Outerjoin
- Outer-union
- 수학적인 집계 연산
  - SUM
  - AVG
  - MAX, MIN
  - COUNT
  - GROUP

# Semijoin: $\ltimes$

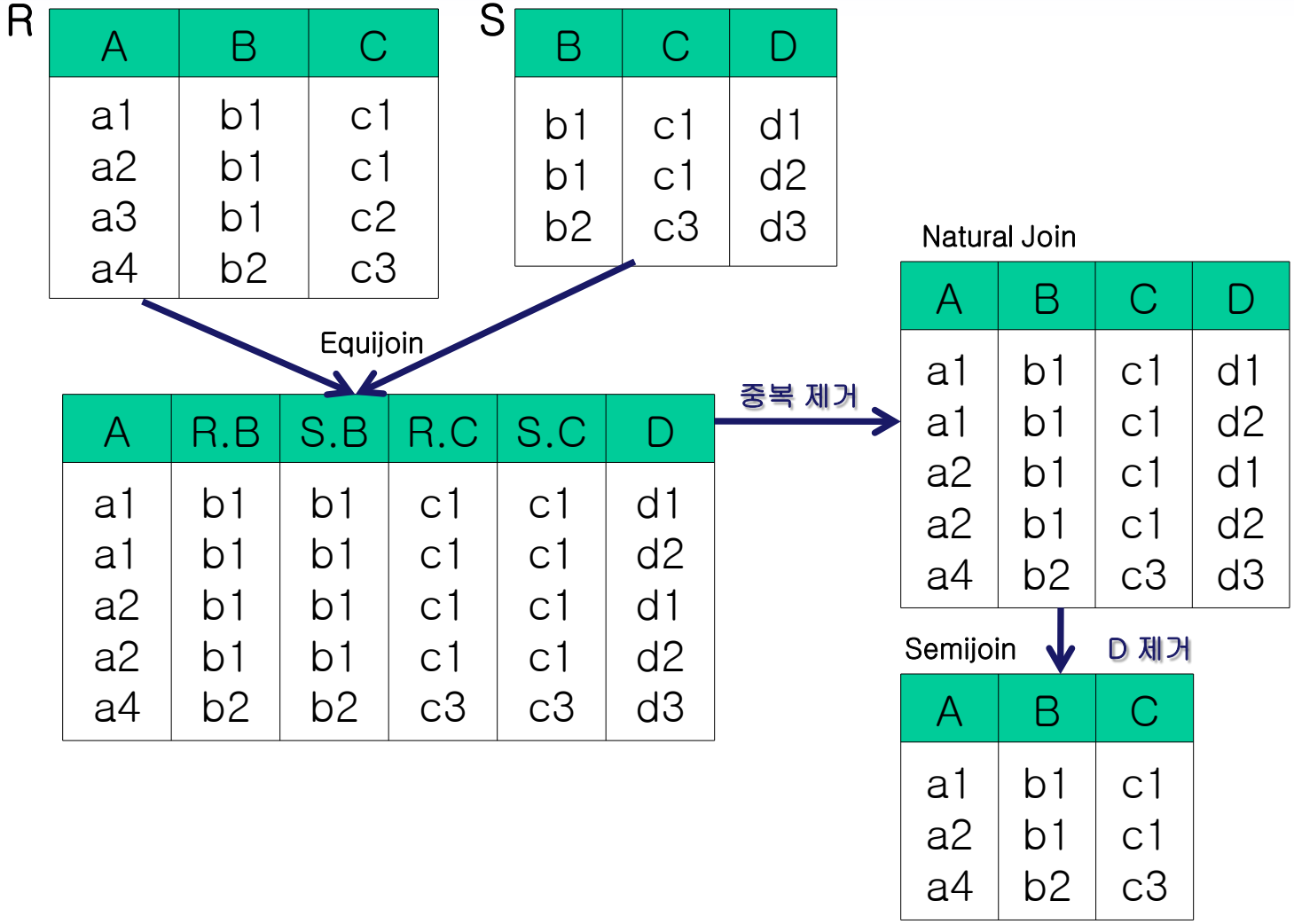
- R(X), S(Y)의 조인 애트리뷰트를 Z=X∩Y라 하면
  $R \ltimes S = R \bowtie_N (\Pi_Z(S)) = \Pi_X(R \bowtie_N S)$
  - S와 natural join 할 수 있는 R의 투플만을 선택

- 특징
  - $R \ltimes S \neq S \ltimes R$
  - $R \bowtie_N S = (R \ltimes S) \bowtie_N S = (S \ltimes R) \bowtie_N R$
    - 처리해야 할 데이터의 양이 다름
      - Query Optimization

# Natural Join vs. Semijoin



# Equijoin, Natural Join and Semijoin



# Outerjoin: $\bowtie^{+}$

- 한 릴레이션에 있는 투플이 조인할 상대 릴레이션에 대응되는 투플이 없을 경우, 상대를 널(null) 투플로 만들어 결과 릴레이션에 포함
  - 두 조인 릴레이션의 모든 투플들이 결과 릴레이션에 포함됨

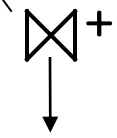
# Natural Join vs. Outerjoin

R

| A | B | C |
|---|---|---|
| a1 | b1 | c1 |
| a2 | b1 | c1 |
| a3 | b1 | c2 |
| a4 | b2 | c3 |

S

| B | C | D |
|---|---|---|
| b1 | c1 | d1 |
| b1 | c1 | d2 |
| b2 | c3 | d3 |
| b3 | c3 | d3 |



$R \overset{+}{\bowtie} S$

| A | B | C | D |
|---|---|---|---|
| a1 | b1 | c1 | d1 |
| a1 | b1 | c1 | d2 |
| a2 | b1 | c1 | d1 |
| a2 | b1 | c1 | d2 |
| a3 | b1 | c2 | |
| a4 | b2 | c3 | d3 |
| | b3 | c3 | d3 |

outerjoin

$R \overset{N}{\bowtie} S$

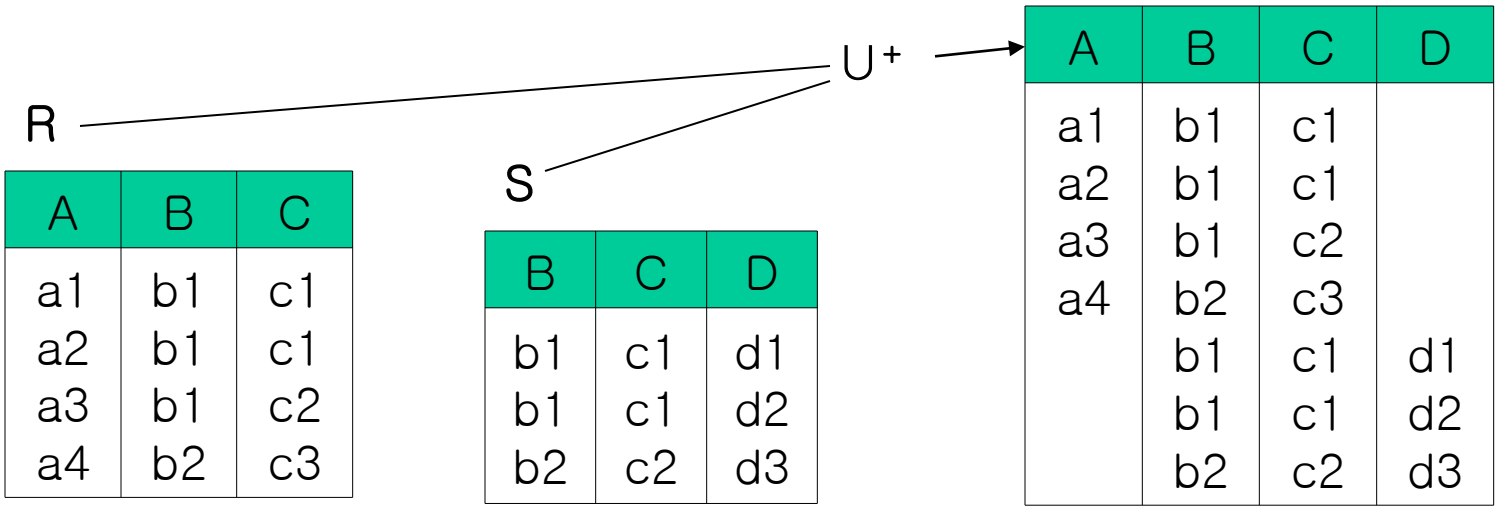
| A | B | C | D |
|---|---|---|---|
| a1 | b1 | c1 | d1 |
| a1 | b1 | c1 | d2 |
| a2 | b1 | c1 | d1 |
| a2 | b1 | c1 | d2 |
| a4 | b2 | c3 | d3 |

Natural join

# Outer-union, $\cup^+$

- 외부 합집합
  - 합병 가능하지 않은(부분적으로 합병 가능한) 두 릴레이션을 degree(차수)를 확장시켜 합집합 생성
    - R(U): degree n, S(V): degree m, RS = R JOIN S
    - R OJOIN S

      = RS $\cup$ ((R - RS[U]) $\times$ (null, $\cdots$ , null)m-1) $\cup$ ((null, $\cdots$ , null)n-1 $\times$ (S - RS[V]))
    - 문제점: nulls in the primary key position




| A | B | C |
|---|---|---|
| a1 | b1 | c1 |
| a2 | b1 | c1 |
| a3 | b1 | c2 |
| a4 | b2 | c3 |

| B | C | D |
|---|---|---|
| b1 | c1 | d1 |
| b1 | c1 | d2 |
| b2 | c2 | d3 |

| A | B | C | D |
|---|---|---|---|
| a1 | b1 | c1 | |
| a2 | b1 | c1 | |
| a3 | b1 | c2 | |
| a4 | b2 | c3 | |
| | b1 | c1 | d1 |
| | b1 | c1 | d2 |
| | b2 | c2 | d3 |

# 집계 연산

- 집계 연산
  - SUM : 합계, AVG: 평균값, MAX: 최대값, MIN: 최소값
  - COUNT: 집합 내에 속한 값의 개수
  - GROUP: 지정된 애트리뷰트 값에 따라 투플들을 그룹핑
    - SUMMARIZE SP GROUPBY (P#) ADD SUM (QTY) AS TOTQTY

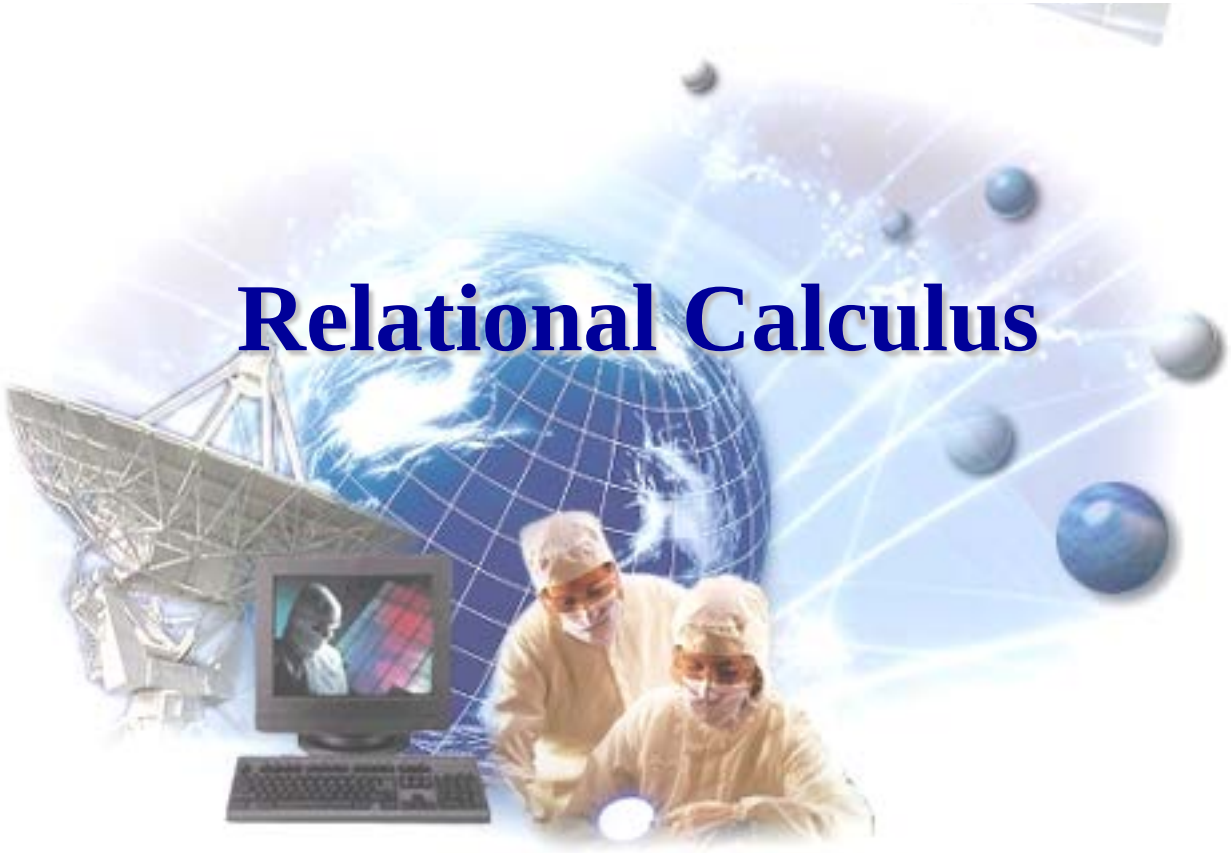
데이터 언어식 표현

  - $AVG_{성적}$(등록)
    - 등록 릴레이션에 있는 성적 애트리뷰트 값들에 대해 평균값 계산
  - $GROUP_{학년}$(학생)
    - 학생 릴레이션의 투플들을 학년 값에 따라 그룹 짓게 함
  - $GROUP_{과목번호}AVG_{성적}$(등록)
    - 과목별 그룹에 대한 평균성적
- 일반 형식 : $G_AF_B(E)$
  - G : 그룹 함수 GROUP (GROUPBY)
  - A : 그룹 함수가 적용할 애트리뷰트
  - F : 집단 함수 ( SUM, AVG, MAX, MIN, COUNT)
  - B : 집단 함수의 적용 대상 애트리뷰트
  - E : Relational Algebra Expression (관계 대수 식)

# Relational Algebra 질의

교재 p.111, p.86 참조

- 시험문제: 다음 질의를 (1) Relational Algebra, (2) 데이터 언어식 표현으로 바꾸고 (3) 결과 릴레이션을 구하시오.
  - 모든 학생의 이름과 학과를 검색하라.
  - 과목번호가 C413인 과목에 등록한 학생의 이름과 성적은 무엇인가?
  - '화일구조' 과목을 가르치는 교수의 이름을 검색하라.
  - 모든 과목에 수강하고 있는 학생의 학번과 이름을 검색하라.
  - 학번이 600, 이름이 '김영호', 학년이 4, 학과가 컴퓨터인 학생을 삽입하라.
  - 과목 '데이터베이스'를 삭제하라.



# Relational Calculus

kit 국립금오공과대학교
Kumoh National Institute of Technology

# Introduction of Relational Calculus

- Relational calculus
  - an applied predicate calculus specially tailored to relational databases
  - first-order predicate calculus for query expression
- Relational algebra vs. relational calculus
  - similarity
    - a formal basis for the manipulative part
      - 관계 데이터 모델의 연산 표현 방법
    - precisely equivalent to one another
  - differences

| **relational algebra** | **relational calculus** |
|---|---|
| **- a collection of explicit operations (join, union, projection, etc)** | **-a notation for formulating the definition of the desired relation** |
| **-procedural (how)** | **-descriptive, nonprocedural (what)** |
| **-1. join, 2. selection 3. projection** | -원하는 정보가 무엇이라는 것만 선언 |

# Introduction

- Two kinds of relational calculus
  - Tuple calculus (TRC: Tuple Relational Calculus)
    - Tuple (range) variables: range over tuples
      - 지정된 릴레이션의 한 투플만을 그 값으로 취할 수 있는 변수
  - Domain calculus (DRC: Domain Relational Calculus)
    - domain variables: range over domain elements ( = field values)
      - 지정된 애트리뷰트의 도메인의 한 원소값만을 값으로 취하는 변수
      - QBE(Query-By-Example)

# Tuple Variables and Qualified Attribute

- Tuple Variables
  - 지정된 릴레이션의 한 투플만을 그 값으로 취할 수 있는 변수
  - 투플 변수(tuple variable) 또는 범위변수(range variable): t
  - 범위식 (range formula) : R(t)
    - t는 R의 투플 변수
  - R : t의 범위 릴레이션 (range relation)

- 한정 애트리뷰트(qualified attribute) : t.A 또는 t[A]
  - 투플 변수 t가 나타내는 투플의 어떤 애트리뷰트 A의 값
  - Student(s)

    s.Sno

# First-order Predicate Calculus

이해 안되면 skip

- Predicate (서술, 술어, 서술어)
  - a function whose value is true or false
    - a function that maps object arguments into TRUE or FALSE
- first-order predicate calculus
  - forbid variables that represent predicates (i.e., objects only)
  - 문장의 주어가 개별 개체 (소크라테스는 죽는다)
  - Quantifier가 변수에만 적용되고 predicate나 함수에 대해서는 허용되지 않음
    - $(\forall x)P(x)$
  - 서술어, 함수, 변수, 상수, 한정자, 논리적 연결자 등으로 구성된 Symbol로 표현
- second-order 또는 higher order predicate calculus
  - permit variables that represent predicates
  - 주어가 술어(predicate)로 구성
    - 죽는다는 것은 비극이다, $(\forall P)P(x)$

# Tuple Relational Calculus

- 원하는 릴레이션을 tuple calculus expression으로 정의할 수 있는 표기법
- Query : { t | P(t) }으로 표현
  - P(t)는 tuple variable t에 대한 formula
    - Expressions in the calculus are called formulas
- Answer
  - the set of all tuples t for which the formula P(t) evaluates to TRUE
- Formula
  - Recursively defined (nested formula)
    - Start with simple atomic formulas
      - Get tuples from relations or making comparisons of values
    - Build bigger and better formulas using the logical connectives ($\neg$, $\wedge$, $\vee$, $\Rightarrow$, $=$)

# Formulas

- Atomic formula, atomic expression, proposition (명제), atom
  - ① R(t)
    t : 투플 변수, R : t의 범위 릴레이션
  - ② t.A θ u.B
    t, u : 투플 변수, A, B : t와 u에 대한 한정 애트리뷰트
    θ : 비교 연산자(=, ≠, <, ≤, >,≥)
  - ③ t.A θ c
    A : 투플 변수 t에 대한 한정 애트리뷰트, c : 상수
  - Atom의 결과는 반드시 참(True) 또는 거짓(False)
- A formula can be:
  - An atomic formula
  - ¬ p, p ∧ q, p ∨ q where p and q are formulas
  - (∃ t)P(t) where variable t is a tuple variable
  - (∀ t)P(t) where variable t is a tuple variable

# Free and Bound Variables

- Quantifiers (정량자)
  - ∀ : 전칭 정량자(Universal quantifier) – “for all”
  - ∃ : 존재 정량자(Existential quantifier) – “there exists”
- Bound Variable (속박 변수)
  - Formula에 ∃x 및 ∀x를 포함하고 있다면 x는 bound variable
    - cf.) bound variable이 아닌 것은 free variable
- Free Variable (자유 변수)
  - Quantifiers (∀ and ∃)로 한정되지 않는 tuple variable
- There is an important restriction
  - The variable t that appears to the left of ‘|’ must be the only **free variable** in the formula P(t)
  - 즉, 다른 모든 tuple variable들은 quantifier를 사용한 bound variable이어야 함

# WFF: Well-Formed Formula

- 정형식(WFF, Well-formed formula)
  - Atom, Boolean operator (∧, ∨, ¬), quantifier (∀, ∃)가 다음 규칙에 따라 결합된 식
    - ① 모든 atomic formula(atom)는 WFF
    - ② F가 WFF이면, (F)와 ¬F도 WFF
    - ③ F와 G가 WFF이면, F∧G와 F∨G도 WFF
    - ④ 투플 변수 t가 free variable로 사용된 F(t)가 WFF이면, ∀t(F(t))와 ∃t(F(t))도 WFF
    - ⑤ 위의 규칙만을 반복 적용해서 만들어진 식은 WFF
  - WFF의 예제

    s.Sno = 100

    c.Cno ≠ e.Cno

    s.Sno = e.Sno ∧ e.Cno ≠ c.Cno

    (∃e)(e.Sno = s.Sno ∧ e.Cno = 'C413')

# Tuple Calculus Expression

- 형식

  Target list

  WFF

  $\{ t_1.A_1, t_2.A_2, \ldots, t_n.A_n \mid F(t_1, \ldots t_n, t_{n+1}, \ldots, t_{n+m}) \}$

  - $t_i$ : 투플 변수
  - $F(t_1,\ldots, t_n, t_{n+1},\ldots, t_{n+m})$: $t_i$가 연관된 WFF로 조건을 명세
  - Target list
    - 막대 ( | ) 왼편에 나온 qualified attribute들
      - a list of “target items” separated by commas
    - 막대( | ) 오른편에 명세된 조건을 만족하는 결과로 추출 됨
- example
  - {s.Sname | STUDENT(s) }
  - {s.Sname | STUDENT(s) ∧ s.Dept=‘컴퓨터’ }
  - {s.Sname, s.Dept | STUDENT(s) ∧
    (∃e)(ENROL(e) ∧ s.Sno=e.Sno ∧ e.Grade='A')}

# 예제

- 과목 C413에서 성적이 A인 학생의 학번을 모두 검색하라

  {e.Sno | ENROL(e) ∧ e.Cno='C413'∧ e.Grade='A' }

- 과목 C413을 등록한 학생의 이름과 학과를 모두 검색하라

  { s.Sname, s.Dept | STUDENT(s) ∧ ∃e(ENROL(e) ∧ s.Sno=e.Sno ∧ e.Cno='C413') }

- 모든 과목에 등록한 학생의 이름을 전부 검색하라.

  { s.Sname | STUDENT(s) ∧ (∀c)(∃e)(COURSE(c) ∧ENROL(e)∧ e.Sno=s.Sno ∧ e.Cno=c.Cno) }

- 과목 C413에 등록하지 않은 학생의 이름 전부를 검색하라.

  {s.Sname | STUDENT(s) ∧ (¬ ∃e)(ENROL(e) ∧ s.Sno=e.Sno ∧ e.Cno=‘C413’) }

# Domain Relational Calculus

- 원하는 릴레이션을 domain calculus expression으로 표현하는 방법
- Domain variable
  - 지정된 애트리뷰트의 도메인의 한 원소만을 값으로 취하는 변수
    - 편의상 애트리뷰트 이름 앞에 x 붙임: xSno, xSname, …
  - 범위식을 사용하여 도메인 선언
    - STUDENT(xSno, xSname, xDept, xYear)
- Atomic formula

① $R(x_1,x_2,\ldots,x_n)$
- $x_i$ : 도메인 변수, R : $x_i$의 range relation (범위 릴레이션)
- $<x_1,x_2,\ldots,x_n>$에 해당하는 값의 리스트는 릴레이션 R의 투플

② $x\ \theta\ y$
- x, y : 도메인 변수, θ: 비교 연산자(=, ≠, <, ≤, >,≥)

③ $x\ \theta\ c$
- x : 도메인 변수, θ: 비교 연산자, c : x가 정의된 도메인 값의 상수

  - 원자의 실행 결과는 반드시 참(True) 또는 거짓(False)

# Domain Calculus Expression

- 형식

Target list

WFF

$\{ x_1,x_2,\ldots,x_n \mid F(x_1,\ldots,x_n,x_{n+1},\ldots,x_{n+m}) \}$

  - $x_i$: 도메인변수
  - $F(x_1,\ldots,x_n,x_{n+1},\ldots,x_{n+m})$ : $x_i$에 대한 WFF

◆ Target list

  - 막대 ( | ) 왼편에 나온 domain variable들
  - 막대( | ) 오른편에 명세된 조건을 만족하는 domain 값으로 만들어지는 tuple

◆ example

① { xSname | STUDENT(xSno, xSname, xYear, xDept)}

② { xSname | (∃xDept)(STUDENT(xSno, xSname, xYear, xDept)∧ xDept='컴퓨터') }

③ { xSno, xDept | STUDENT(xSno, xSname, xYear, xDept) ∧ (∃xxSno)(∃xGrade)(ENROL(xxSno, xCno, xGrade, xMidterm, xFinal) ∧xSno=xxSno∧xGrade='A') }

# 예제

- 컴퓨터학과 3,4 학년의 이름을 검색하라.

  {xSname | (∃xYear)(∃xDept)(STUDENT(xSno, xSname, xYear, xDept)∧
  xYear ≥ 3∧ xDept='컴퓨터' ) }

- 과목 C413에서 성적이 A인 학생의 학번을 모두 검색하라

  { xSno | (∃xCno)(∃xGrade)(ENROL(xSno, xCno, xGrade, xMidterm, xFinal)
  ∧ xCno= 'C413' ∧ xGrade='A') }

- 기말 성적이 90점 이상인 학생의 학번과 이름을 검색하라.

  {xSno,xSname | (STUDENT(xSno,xSname,xYear,xDept) ∧
  (∃xFinal)(∃xxSno) (ENROL(xxSno, xCno, xGrade, xMidterm, xFinal)∧
  xSno=xxSno ∧ xFinal ≥ 90) }

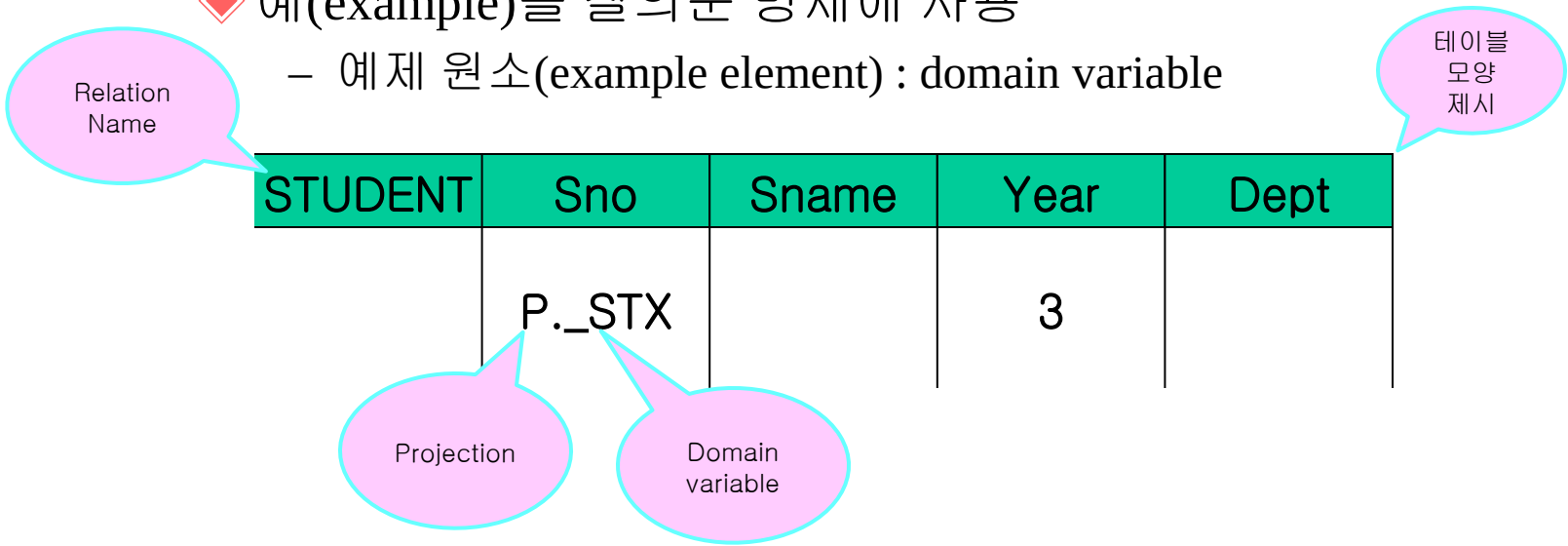
- 과목 C324에 등록하지 않은 학생의 이름을 검색하라.

  { xSname | (∃xSno)((STUDENT(xSname, xSno, xYear, xDept) ∧ (¬ ∃xxSno)
  (∃xCno) (ENROL(xxSno, xCno, xGrade, xMidterm, xFinal) ∧ xSno=xxSno
  ∧ xCno='C324')) }

# Query by Example

# QBE

- Domain relational calculus 사용
- QBE (Query By Example)
  - 1975, IBM
  - 그래픽 디스플레이 단말기 사용
  - 이차원 구문(two-dimensional syntax) 언어
  - 예(example)를 질의문 명세에 사용
    - 예제 원소(example element) : domain variable



# 데이터 검색(1)

- 단순 조건 검색
  - 컴퓨터학과 4학년 학생의 학번과 이름을 검색하라

| STUDENT | Sno | Sname | Year | Dept |
|---|---|---|---|---|
| | P. | P. | 4 | 컴퓨터 |

  - 중복되는 것은 자동적으로 제거됨
  - 'ALL'을 삽입하면 중복을 허용

| STUDENT | Sno | Sname | Year | Dept |
|---|---|---|---|---|
| | | P.ALL | 4 | 컴퓨터 |

# 데이터 검색(2)

- 테이블 전체의 검색
  - 학생 테이블의 모든 사항을 검색하라

| STUDENT | Sno | Sname | Year | Dept |
|---|---|---|---|---|
| | P. | P. | P. | P. |

  - 간단한 방법 : 테이블 이름 밑에 P. 명세

| STUDENT | Sno | Sname | Year | Dept |
|---|---|---|---|---|
| P. | | | | |

# 데이터 검색(3)

- 복수 조건 검색
  - 'OR' 조건 : 두 개의 행, 다른 예제 원소
    - 기말성적이 85점 이상이거나 과목번호 'C413'에 등록한 학생의 학번을 검색하라

| ENROL | Sno | Cno | Final | Midterm |
|---|---|---|---|---|
| | P. | | ≥85 | |
| | P. | C413 | | |

  - 'AND' 조건 : 하나의 행, 같은 예제 원소
    - 과목번호가 'C413'이고 기말성적이 85점 이상인 학생의 학번

| ENROL | Sno | Cno | Final | Midterm |
|---|---|---|---|---|
| | P. | C413 | ≥85 | |

# 데이터 검색(4)

- 복수 조건 검색(con't)
  - 조건 상자(condition box)의 사용

| ENROL | Sno | Cno | Final | Midterm |
|---|---|---|---|---|
| | P. | _EC | _EF | |

| CONDITIONS |
|---|
| _EC=C413 AND _EF ≥85 |

# 데이터 검색(5)

- 복수 테이블에서 검색
  - 기말성적이 85점 이상이거나 과목 'C413'을 등록한 학생의 이름

| ENROL | Sno | Cno | Final | Midterm |
|---|---|---|---|---|
| Link 역할 | _STX | | ≥85 | |
| | _STY | C413 | | |

| STUDENT | Sno | Sname | Year | Dept |
|---|---|---|---|---|
| | _STX | P. | | |
| | _STY | P. | | |

# 데이터의 삽입

- 단순 레코드의 삽입
  - 학번이 100이고 과목번호가 'C413'인 투플 삽입

| ENROL | Sno | Cno | Grade | Midterm | Final |
|---|---|---|---|---|---|
| I. | 100 | C413 | | | |

insert

Note : Primary key가 null이어서는 안됨

- 투플 검색을 이용한 삽입
  - 4학년 학생의 학번을 학생테이블로부터 검색해서 SENIOR 테이블에 삽입하라.

| SENIOR | Sno |
|---|---|
| I. | _STX |

| STUDENT | Sno | Sname | Year | Dept |
|---|---|---|---|---|
| | _STX | | 4 | |

# 데이터의 삭제

- 한 테이블에서의 삭제
  - 학번이 100인 학생을 학생 테이블에서 삭제

delete

| STUDENT | Sno | Sname | Year | Dept |
|---|---|---|---|---|
| D. | 100 | | | |

- 복수 테이블에서의 레코드 삭제
  - 기말성적이 60점 미만인 학생을 등록 테이블과 학생테이블에서 삭제

| ENROL | Sno | Cno | Grade | Midterm | Final |
|---|---|---|---|---|---|
| | _STX | | | | <60 |
| D. | _STX | | | | |

| STUDENT | Sno | Sname | Year | Dept |
|---|---|---|---|---|
| D. | _STX | | | |

# 데이터의 갱신

- 필드 값의 단순 갱신
  - 학번이 300인 학생의 학년을 2로 변경

| STUDENT | Sno | Sname | Year | Dept |
|---|---|---|---|---|
| | 300 | | U.2 | |

update

| STUDENT | Sno | Sname | Year | Dept |
|---|---|---|---|---|
| U. | 300 | | 2 | |

update

- 산술식을 이용한 갱신
  - 과목 'C413'에 등록한 학생의 기말 성적(Final)에 5점을 가산

| ENROL | Sno | Cno | Final | Midterm |
|---|---|---|---|---|
| U. | | | _G+5 | |
| | | C413 | _G | |