

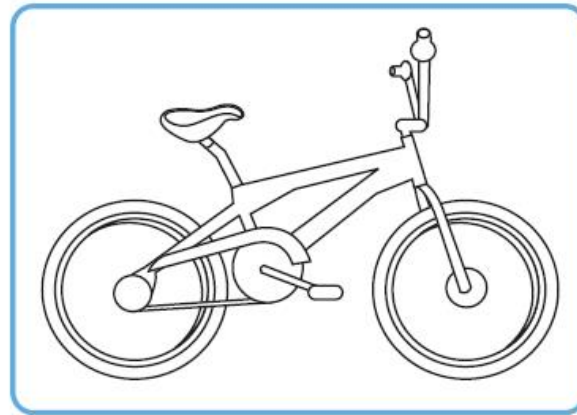
# Relational Model

# Model

- 현실세계를 단순화하고 정형화된 형태로 표현하는 방식 또는 규범
  - 추상화



(a) 자전거 사진



(b) 선으로 단순하게 그린 자전거

- Relational (Data) Model

# Relation

## ❑ Set

- A set is an unordered collection of objects.

## ❑ Cartesian Product

- Let  $A = \{a_1, a_2, \dots, a_k\}$  and  $B = \{b_1, b_2, \dots, b_m\}$
- The Cartesian product  $A \times B$  is defined by a set of pairs  
 $\{(a_1, b_1), (a_1, b_2), \dots, (a_1, b_m), \dots, (a_k, b_m), \}$
- Cartesian product defines a product set, or a set of all ordered arrangements of elements in sets in the Cartesian product

## ❑ A relation R on a set A is a set of ordered pairs of elements from A

- Relation  $P = \{(\text{grandfather}, \text{mother}), (\text{mother}, \text{me}), (\text{mother}, \text{sister}), \dots\}$
- Relation  $S = \{(\text{sister}, \text{me}), (\text{me}, \text{sister}), (\text{brother}, \text{me}), (\text{me}, \text{brother}), \dots\}$

## ❑ Relation Properties

- Reflexive, Symmetric, Transitive, ...

# Relation

□ Formally, given sets  $D_1, D_2, \dots, D_n$

a *relation*  $r$  is a **subset** of  $D_1 \times D_2 \times \dots \times D_n$

$r \subseteq D_1 \times \dots \times D_n$  ( $n$ : degree of  $r$ ) or

$r = \{ \langle d_1, \dots, d_n \rangle \mid d_1 \in D_1, \dots, d_n \in D_n \}$  (set of tuples)

- A set is an unordered collection of objects.

□ A relation is a mathematical tool for describing association between elements of sets.

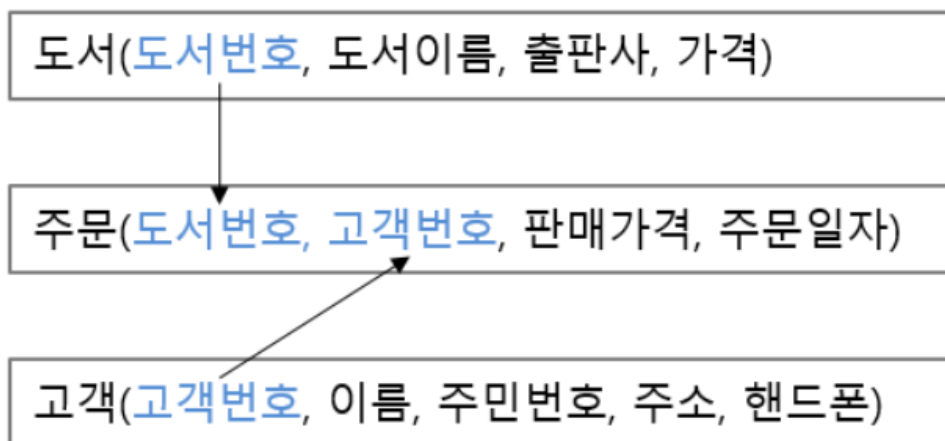
- Widely used in computer science, especially in database and scheduling applications

attributes  
(or columns)

tuples  
(or rows)

# Relation and Relationship

용어	한글 용어	비고
relation	<u>릴레이션</u> , 테이블	“관계”라고 하지 않음
relational data model	관계 데이터 모델	
relational database	관계 데이터베이스	
relational algebra	관계대수	
relationship	관계	



# Attribute Types



- ❑ The set of allowed values for each attribute is called the **domain** of the attribute
- ❑ Attribute values are (normally) required to be **atomic**; that is, indivisible
- ❑ The special value ***null*** is a member of every domain. Indicated that the value is “unknown”
- ❑ The null value causes complications in the definition of many operations

# Relation Schema and Instance

- ❑  $A_1, A_2, \dots, A_n$  are *attributes*
- ❑  $R = (A_1, A_2, \dots, A_n)$  is a *relation schema*

Example:

*instructor = (ID, name, dept\_name, salary)*

- ❑ Formally, given sets  $D_1, D_2, \dots, D_n$  a **relation**  $r$  is a subset of  
 $D_1 \times D_2 \times \dots \times D_n$

Thus, a relation is a set of  $n$ -tuples  $(a_1, a_2, \dots, a_n)$  where each  $a_i \in D_i$

- The current values (**relation instance**) of a relation are specified by a table
- An element  $t$  of  $r$  is a *tuple*, represented by a *row* in a table



# Relations are Unordered

- ❑ Order of tuples is irrelevant (tuples may be stored in an arbitrary order)

Example: *instructor* relation with unordered tuples

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

# A (theoretical) summary on *Relation*

- ❑ 속성은 단일 값을 가진다
  - 각 속성의 값은 도메인에 정의된 값만을 가지며 그 값은 모두 단일 값이어야 함.
- ❑ 속성은 서로 다른 이름을 가진다
  - 속성은 한 릴레이션에서 서로 다른 이름을 가져야만 함.
- ❑ 한 속성의 값은 모두 같은 도메인 값을 가진다
  - 한 속성에 속한 열은 모두 그 속성에서 정의한 도메인 값만 가질 수 있음.
- ❑ 속성의 순서는 상관없다
  - 속성의 순서가 달라도 릴레이션 스키마는 같음.
  - 예) 릴레이션 스키마에서 (이름, 주소) 순으로 속성을 표시하거나 (주소, 이름) 순으로 표시하여도 상관없음.
- ❑ 릴레이션 내의 중복된 튜플은 허용하지 않는다
  - 하나의 릴레이션 인스턴스 내에서는 서로 중복된 값을 가질 수 없음. 즉 모든 튜플은 서로 값이 달라야 함.
- ❑ 튜플의 순서는 상관없다
  - 튜플의 순서가 달라도 같은 릴레이션임. 관계 데이터 모델의 튜플은 실제적인 값을 가지고 있으며 이 값은 시간이 지남에 따라 데이터의 삭제, 수정, 삽입에 따라 순서가 바뀔 수 있음.

# Keys

- ❑ Let  $K \subseteq R$ ,  $K$  is a **superkey** of  $R$  if values for  $K$  are sufficient to identify a unique tuple of each possible relation  $r(R)$ 
  - No two distinct tuples in any state  $r$  of  $R$  can have the same value for superkey
  - Example:  $\{ID\}$  and  $\{ID, name\}$  are both superkeys of *instructor*.
- ❑ Superkey  $K$  is a **candidate key** if  $K$  is minimal
  - Removing any attribute  $A$  from  $K$  leaves a set of attributes  $K$  that is not a superkey of  $R$  any more
  - Cannot remove any attributes and still have uniqueness constraint in above condition hold
  - Example:  $\{ID\}$  is a candidate key for *Instructor*
- ❑ One of the candidate keys is selected to be the **primary key**.
  - which one?
- ❑ **Foreign key** constraint: Value in one relation must appear in another
  - **Referencing** relation
  - **Referenced** relation
  - Example – *dept\_name* in *instructor* is a foreign key from *instructor* referencing *department*

# Keys



- ☐ *Key*
  - == *superkey*
- ☐ *Surrogate key*
  - == *artificially generated key*
- ☐ *Alternate key*
  - *Non-candidate key*

# Examples of Keys

고객

고객번호	이름	주민번호	주소	핸드폰
1	박지성	810101-1111111	영국 맨체스터	000-5000-0001
2	김연아	900101-2222222	대한민국 서울	000-6000-0001
3	장미란	830101-2333333	대한민국 강원도	000-7000-0001
4	추신수	820101-1444444	미국 클리블랜드	000-8000-0001

도서

도서번호	도서이름	출판사	가격
1	축구의 역사	굿스포츠	7000
2	축구하는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000

주문

고객번호	도서번호	판매가격	주문일자
1	1	7000	2014-07-01
1	2	13000	2014-07-03
2	5	8000	2014-07-03
3	2	13000	2014-07-04
4	4	35000	2014-07-05
1	3	22000	2014-07-07
4	3	22000	2014-07-07

# Examples of Keys

고객

고객번호	이름	주민번호	주소	핸드폰
1	박지성	810101-1111111	영국 맨체스타	000-5000-0001
2	김연아	900101-2222222	대한민국 서울	000-6000-0001
3	장미란	830101-2333333	대한민국 강원도	000-7000-0001
4	<u>추신수</u>	820101-1444444	미국 <u>클리블랜드</u>	000-8000-0001

기본키

도서

도서번호	도서이름	출판사	가격
1	축구의 역사	<u>굿스포츠</u>	7000
2	<u>축구하는 여자</u>	<u>나무수</u>	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	<u>굿스포츠</u>	8000

기본키

참조

주문

주문번호	고객번호	도서번호	판매가격	주문일자
1	1	1	7000	2014-07-01
2	1	2	13000	2014-07-03
3	2	5	8000	2014-07-03
4	3	2	13000	2014-07-04
5	4	4	35000	2014-07-05
6	1	3	22000	2014-07-07
7	4	3	22000	2014-07-07

외래키

참조

기본키

# Examples of Keys




Diagram illustrating a table with keys:

The table has four columns: 선수번호 (Player Number), 이름 (Name), 주소 (Address), and 멘토번호 (Mentor Number).

Annotations:

- 기본키 (Primary Key) points to the 선수번호 column.
- 참조 (Reference) points to the 멘토번호 column.
- 외래키 (Foreign Key) points to the 멘토번호 column.

<u>선수번호</u>	<u>이름</u>	<u>주소</u>	<u>멘토번호</u>
1	박지성	영국 <u>맨체스터</u>	NULL
2	김연아	대한민국 서울	3
3	장미란	대한민국 강원도	4
4	<u>추신수</u>	미국 <u>클리블랜드</u>	NULL

# Constraints



## ☐ Constraints

- Restrictions on the permitted values in a database state
- Derived from the rules in the miniworld that the database represents

## ☐ Inherent model-based constraints or implicit constraints

- Inherent in the data model
- e.g., duplicate tuples are not allowed in a relation

## ☐ Schema-based constraints or explicit constraints

- Can be directly expressed in schemas of the data model
- e.g., films have only one director

## ☐ Application-based or semantic constraints

- Also called business rules
- Not directly expressed in schemas
- Expressed and enforced by application program
- e.g., this year's salary increase can be no more than last year's



# Integrity Constraints: Domain Constraints



❑ Declared by specifying the data type for each attribute:

- Numeric data types for integers and real numbers
- Characters
- Booleans
- Fixed-length strings
- Variable-length strings
- Date, time, timestamp
- Money
- Other special data types

# Integrity Constraints: Key Constraints

- ❑ Uniqueness constraints on tuples
- ❑  $SK \subseteq \{A_1, A_2, \dots, A_n\}$  is a **superkey** of  $R(A_1, A_2, \dots, A_n)$  if
  - In any relation state  $r$  of  $R$ , no two distinct tuples can have the same values for  $SK$ 
    - $t_1[SK] = t_2[SK] \Rightarrow t_1 = t_2$
- ❑  $K$  is a **key** of  $R$  if
  1.  $K$  is a superkey of  $R$
  2. Removing any attribute from  $K$  leaves a set of attributes that is not a superkey of  $R$  any more
    - No proper subset of  $K$  is a superkey of  $R$
- ❑ If  $K$  is a key, it satisfies two properties
  1. No two distinct tuples have the same values across all attributes in  $K$  (i.e., it is a superkey)
  2. It is a minimal superkey (i.e., no subset of  $K$  has this uniqueness constraint)

# Integrity Constraints: Entity Integrity Constraints



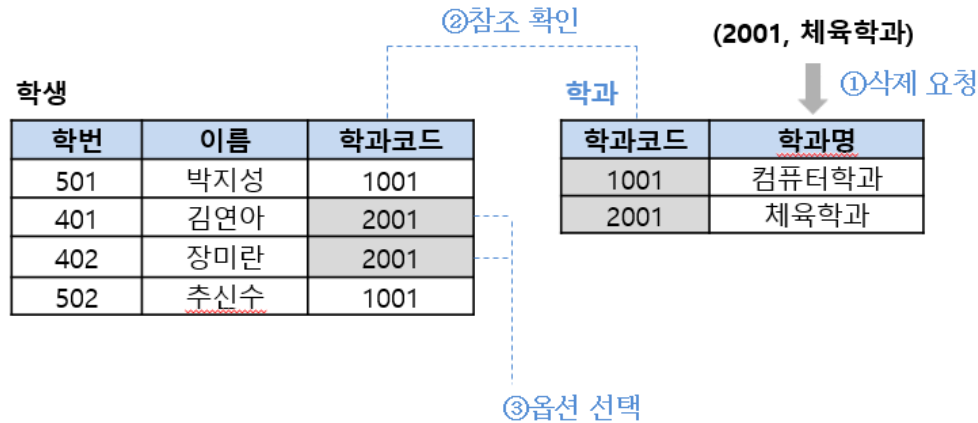
## ❑ Integrity constraint

- No primary key value can be NULL

## ❑ Referential integrity constraint

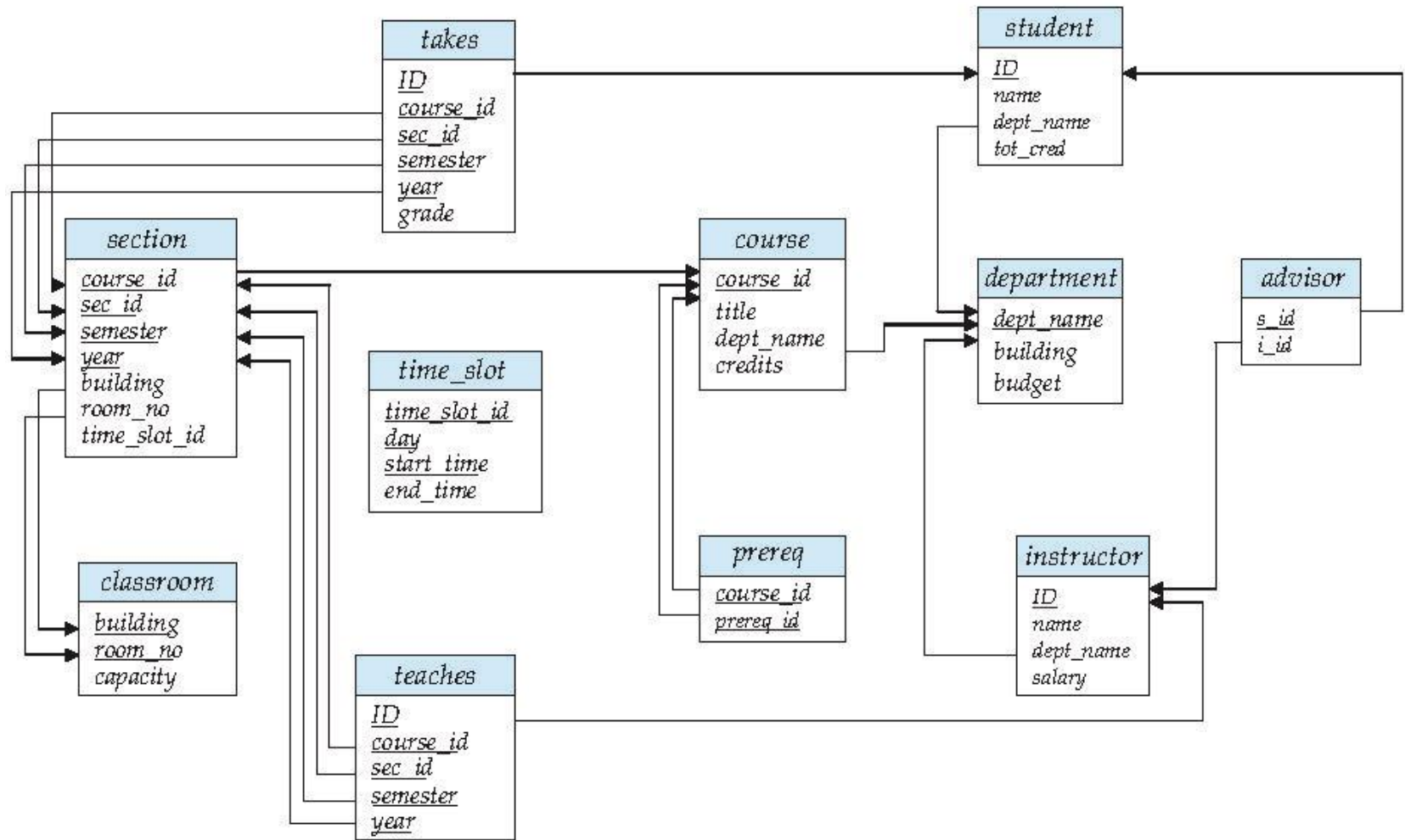
- Specified between two relations
  - Allows tuples in one relation to *refer to* tuples in another
- Maintains consistency among tuples in two relations
- **Foreign key rules:**
  - Let PK be the primary key in one relation  $R_1$  (set of attributes in its relational schema declared to be primary key)
  - Let FK be a set of attributes for another relation  $R_2$
  - The attribute(s) FK have the same domain(s) as the attribute(s) PK
  - Value of FK in a tuple  $t_2$  of the current state  $r_2(R_2)$  either occurs as a value of PK for some tuple  $t_1$  in the current state  $r_1(R_1)$  or it is NULL

# Referential integrity constraint




명령어	의미	예
RESTRICTED	자식 릴레이션에서 참조하고 있을 경우 부모 릴레이션의 삭제 작업을 거부함	학과 릴레이션의 튜플 삭제 거부
CASCADE	자식 릴레이션의 관련 튜플을 같이 삭제 처리함	학생 릴레이션의 관련 튜플을 삭제
DEFAULT	자식 릴레이션의 관련 튜플을 미리 설정해둔 값으로 변경함	학생 릴레이션의 학과가 다른 학과로 자동 배정
NULL	자식 릴레이션의 관련 튜플을 NULL 값으로 설정함(NULL 값을 허가한 경우)	학과 릴레이션의 학과가 NULL 값으로 변경

# Schema Diagram for University Database



# Relational Query Languages

- 
- ☐ Procedural vs .non-procedural, or declarative
  - ☐ “Pure” languages:
    - **Relational algebra**
    - Tuple relational calculus
    - Domain relational calculus
  - ☐ The above 3 pure languages are equivalent in computing power

# Relational Algebra



- ❑ Procedural language
- ❑ A system of operand and operator
  - operands: relations
  - operators: basic operator taking two or more relations as inputs and giving a new relation (or tuple sets) as a result
    - Ex) select, project, union, set difference, rename, Cartesian product, ...
    - Ex) join, natural join, assignment, division, ...
- ❑ Relational Calculus
  - Non-procedural language
  - Not covered in this course

# Select Operation – selection of rows (tuples)

Relation r

A	B	C	D
$\alpha$	$\alpha$	1	7
$\alpha$	$\beta$	5	7
$\beta$	$\beta$	12	3
$\beta$	$\beta$	23	10

$\sigma_{A=B \wedge D > 5}(r)$

A	B	C	D
$\alpha$	$\alpha$	1	7
$\beta$	$\beta$	23	10



# Project Operation – selection of columns (Attributes)

□ Relation  $r$ :

A	B	C
$\alpha$	10	1
$\alpha$	20	1
$\beta$	30	1
$\beta$	40	2

$\Pi_{A,C}(r)$

A	C
$\alpha$	1
$\alpha$	1
$\beta$	1
$\beta$	2

=

A	C
$\alpha$	1
$\beta$	1
$\beta$	2

# Union of two relations

□ Relations  $r, s$ :

$A$	$B$
$\alpha$	1
$\alpha$	2
$\beta$	1

$r$

$A$	$B$
$\alpha$	2
$\beta$	3

$s$

□  $r \cup s$ :

$A$	$B$
$\alpha$	1
$\alpha$	2
$\beta$	1
$\beta$	3

# Set difference of two relations

□ Relations  $r, s$ :

$A$	$B$
$\alpha$	1
$\alpha$	2
$\beta$	1

$r$

$A$	$B$
$\alpha$	2
$\beta$	3

$s$

$r - s$ :

$A$	$B$
$\alpha$	1
$\beta$	1

# Set intersection of two relations

□ Relation  $r, s$ :

$A$	$B$
$\alpha$	1
$\alpha$	2
$\beta$	1

$r$

$A$	$B$
$\alpha$	2
$\beta$	3

$s$

□  $r \cap s$

$A$	$B$
$\alpha$	2

Note:  $r \cap s = r - (r - s)$

# joining two relations -- Cartesian-product

■ Relations  $r$ ,  $s$ :

$A$	$B$
$\alpha$	1
$\beta$	2

$r$

$C$	$D$	$E$
$\alpha$	10	a
$\beta$	10	a
$\beta$	20	b
$\gamma$	10	b

$s$

$r \times s$ :

$A$	$B$	$C$	$D$	$E$
$\alpha$	1	$\alpha$	10	a
$\alpha$	1	$\beta$	10	a
$\alpha$	1	$\beta$	20	b
$\alpha$	1	$\gamma$	10	b
$\beta$	2	$\alpha$	10	a
$\beta$	2	$\beta$	10	a
$\beta$	2	$\beta$	20	b
$\beta$	2	$\gamma$	10	b

# Cartesian-product – naming issue

- Relations  $r, s$ :

$A$	$B$
$\alpha$	1
$\beta$	2

$r$

$C$	$D$	$E$
$\alpha$	10	a
$\beta$	10	a
$\beta$	20	b
$\gamma$	10	b

$s$

- $r \times s$ :

$A$	$B$	$C$	$D$	$E$
$\alpha$	1	$\alpha$	10	a
$\alpha$	1	$\beta$	10	a
$\alpha$	1	$\beta$	20	b
$\alpha$	1	$\gamma$	10	b
$\beta$	2	$\alpha$	10	a
$\beta$	2	$\beta$	10	a
$\beta$	2	$\beta$	20	b
$\beta$	2	$\gamma$	10	b

# Renaming a Table

- Allows us to refer to a relation, (say  $E$ ) by more than one name.

$$\rho_x(E)$$

returns the expression  $E$  under the name  $x$

- Relations  $r$

$A$	$B$
$\alpha$	1
$\beta$	2

$r$

- $r \bowtie \rho_s(r)$

$r.A$	$r.B$	$s.A$	$s.B$
$\alpha$	1	$\alpha$	1
$\alpha$	1	$\beta$	2
$\beta$	2	$\alpha$	1
$\beta$	2	$\beta$	2

# Composition of Operations

❑ Can build expressions using multiple operations

❑ Example:  $\sigma_{A=C}(r \times s)$

❑  $r \times s$

A	B	C	D	E
$\alpha$	1	$\alpha$	10	a
$\alpha$	1	$\beta$	10	a
$\alpha$	1	$\beta$	20	b
$\alpha$	1	$\gamma$	10	b
$\beta$	2	$\alpha$	10	a
$\beta$	2	$\beta$	10	a
$\beta$	2	$\beta$	20	b
$\beta$	2	$\gamma$	10	b

❑  $\sigma_{A=C}(r \times s)$

A	B	C	D	E
$\alpha$	1	$\alpha$	10	a
$\beta$	2	$\beta$	10	a
$\beta$	2	$\beta$	20	b



# Joining two relations – Natural Join



- Let  $r$  and  $s$  be relations on schemas  $R$  and  $S$  respectively.  
Then, the “natural join” of relations  $R$  and  $S$  is a relation on schema  $R \cup S$  obtained as follows:
- Consider each pair of tuples  $t_r$  from  $r$  and  $t_s$  from  $s$ .
  - If  $t_r$  and  $t_s$  have the same value on each of the attributes in  $R \cap S$ , add a tuple  $t$  to the result, where
    - $t$  has the same value as  $t_r$  on  $r$
    - $t$  has the same value as  $t_s$  on  $s$

# Natural Join Example

Relations r, s:

A	B	C	D
$\alpha$	1	$\alpha$	a
$\beta$	2	$\gamma$	a
$\gamma$	4	$\beta$	b
$\alpha$	1	$\gamma$	a
$\delta$	2	$\beta$	b

*r*

B	D	E
1	a	$\alpha$
3	a	$\beta$
1	a	$\gamma$
2	b	$\delta$
3	b	$\epsilon$

*s*

■ Natural Join

■  $r \bowtie s$

A	B	C	D	E
$\alpha$	1	$\alpha$	a	$\alpha$
$\alpha$	1	$\alpha$	a	$\gamma$
$\alpha$	1	$\gamma$	a	$\alpha$
$\alpha$	1	$\gamma$	a	$\gamma$
$\delta$	2	$\beta$	b	$\delta$

$$\square \Pi_{A, r.B, C, r.D, E}(\sigma_{r.B = s.B \wedge r.D = s.D} (r \times s))$$

# Notes about Relational Languages



- ☐ Each Query input is a table (or set of tables)
- ☐ Each query output is a table.
- ☐ All data in the output table appears in one of the input tables
- ☐ Relational Algebra is not Turing complete
- ☐ Can we compute:
  - SUM
  - AVG
  - MAX
  - MIN

# Summary of Relational Algebra Operators

Symbol (Name)	Example of Use
$\sigma$ (Selection)	$\sigma \text{ salary} \geq 85000 \text{ (instructor)}$ Return rows of the input relation that satisfy the predicate.
$\Pi$ (Projection)	$\Pi ID, salary \text{ (instructor)}$ Output specified attributes from all rows of the input relation. Remove duplicate tuples from the output.
$\times$ (Cartesian Product)	$instructor \times department$ Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.
$\cup$ (Union)	$\Pi name \text{ (instructor)} \cup \Pi name \text{ (student)}$ Output the union of tuples from the <i>two</i> input relations.
$-$ (Set Difference)	$\Pi name \text{ (instructor)} - \Pi name \text{ (student)}$ Output the set difference of tuples from the two input relations.
$\bowtie$ (Natural Join)	$instructor \bowtie department$ Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.

# Composition of Relational Operations

- ❑ The result of a relational-algebra operation is relation and therefore of relational-algebra operations can be composed together into a **relational-algebra expression**.
- ❑ Consider the query -- Find the names of all instructors in the Physics department.

$$\Pi_{name}(\sigma_{dept\_name = "Physics"}(instructor))$$

- ❑ Instead of giving the name of a relation as the argument of the projection operation, we give an expression that evaluates to a relation.

# Cartesian-Product Operation

- ❑ The Cartesian-product operation (denoted by  $\times$ ) allows us to combine information from any two relations.
- ❑ Example: the Cartesian product of the relations *instructor* and *teaches* is written as:

*instructor*  $\times$  *teaches*

- ❑ We construct a tuple of the result out of each possible pair of tuples: one from the *instructor* relation and one from the *teaches* relation (see next slide)
- ❑ Since the instructor *ID* appears in both relations we distinguish between these attribute by attaching to the attribute the name of the relation from which the attribute originally came.
  - *instructor.ID*
  - *teaches.ID*

# The *instructor X teaches* table

<i>Instructor.ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>	<i>teaches.ID</i>	<i>course_id</i>	<i>sec_id</i>	<i>semester</i>	<i>year</i>
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2017
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2018
12121	Wu	Finance	90000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
12121	Wu	Finance	90000	15151	MU-199	1	Spring	2018
12121	Wu	Finance	90000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
15151	Mozart	Music	40000	10101	CS-101	1	Fall	2017
15151	Mozart	Music	40000	10101	CS-315	1	Spring	2018
15151	Mozart	Music	40000	10101	CS-347	1	Fall	2017
15151	Mozart	Music	40000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
15151	Mozart	Music	40000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
22222	Einstein	Physics	95000	10101	CS-101	1	Fall	2017
22222	Einstein	Physics	95000	10101	CS-315	1	Spring	2018
22222	Einstein	Physics	95000	10101	CS-347	1	Fall	2017
22222	Einstein	Physics	95000	12121	FIN-201	1	Spring	2018
22222	Einstein	Physics	95000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...

# Join Operation



## ❑ The Cartesian-Product

*instructor X teaches*

associates every tuple of *instructor* with every tuple of *teaches*.

- Most of the resulting rows have information about instructors who did NOT teach a particular course.

## ❑ To get only those tuples of “*instructor X teaches*” that pertain to instructors and the courses that they taught, we write:

$\sigma_{instructor.id = teaches.id} (instructor \times teaches)$

- We get only those tuples of “*instructor X teaches*” that pertain to instructors and the courses that they taught.

## ❑ The result of this expression, shown in the next slide



# Join Operation (Cont.)

❑ The table corresponding to:

$$\sigma_{instructor.id = teaches.id}(instructor \times teaches))$$

<i>Instructor.ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>	<i>teaches.ID</i>	<i>course_id</i>	<i>sec_id</i>	<i>semester</i>	<i>year</i>
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017
32343	El Said	History	60000	32343	HIS-351	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-101	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-319	1	Spring	2018
76766	Crick	Biology	72000	76766	BIO-101	1	Summer	2017
76766	Crick	Biology	72000	76766	BIO-301	1	Summer	2018
83821	Brandt	Comp. Sci.	92000	83821	CS-190	1	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-190	2	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-319	2	Spring	2018
98345	Kim	Elec. Eng.	80000	98345	EE-181	1	Spring	2017

# Join Operation (Cont.)

- ❑ The **join** operation allows us to combine a select operation and a Cartesian-Product operation into a single operation.
- ❑ Consider relations  $r(R)$  and  $s(S)$
- ❑ Let “theta” be a predicate on attributes in the schema  $R \cup S$ . The join operation  $r \bowtie_{\theta} s$  is defined as follows:

$$r \bowtie_{\theta} s = \sigma_{\theta}(r \times s)$$

- ❑ Thus

$$\sigma_{instructor.id = teaches.id}(instructor \times teaches)$$

- ❑ Can equivalently be written as

$$instructor \bowtie_{Instructor.id = teaches.id} teaches.$$

# The Assignment Operation

- ❑ It is convenient at times to write a relational-algebra expression by assigning parts of it to temporary relation variables.
- ❑ The assignment operation is denoted by  $\leftarrow$  and works like assignment in a programming language.
- ❑ Example: Find all instructor in the “Physics” and Music department.

$Physics \leftarrow \sigma_{dept\_name = "Physics"}(instructor)$

$Music \leftarrow \sigma_{dept\_name = "Music"}(instructor)$

$Physics \cup Music$

- ❑ With the assignment operation, a query can be written as a sequential program consisting of a series of assignments followed by an expression whose value is displayed as the result of the query.

# Equivalent Queries

- ❑ There is more than one way to write a query in relational algebra.
- ❑ Example: Find information about courses taught by instructors in the Physics department with salary greater than 90,000
- ❑ Query 1

$$\sigma_{dept\_name = "Physics" \wedge salary > 90,000} (instructor)$$

- ❑ Query 2

$$\sigma_{dept\_name = "Physics"} (\sigma_{salary > 90,000} (instructor))$$

- ❑ The two queries are not identical; they are, however, equivalent -- they give the same result on any database.

# Equivalent Queries

- ❑ There is more than one way to write a query in relational algebra.
- ❑ Example: Find information about courses taught by instructors in the Physics department
- ❑ Query 1

$$\sigma_{dept\_name="Physics"}(instructor \bowtie_{instructor.ID = teaches.ID} teaches)$$

- ❑ Query 2

$$(\sigma_{dept\_name="Physics"}(instructor)) \bowtie_{instructor.ID = teaches.ID} teaches$$

- ❑ The two queries are not identical; they are, however, equivalent -- they give the same result on any database.

# Reference



- ❑ Many slides from Database System Concepts, 6th Ed.
  - See [www.db-book.com](http://www.db-book.com) for conditions on re-use
- ❑ Some slides (figures in Korean) from 한빛아카데미 데이터베이스 개론