Chapter 2: Introduction to Relational Model

Revision by Gun-Woo Kim

Dept. of Computer Science and Engineering

Hanyang University

Database System Concepts, 6^{th} Ed.

Contents

2.1 Structure of Relational Databases

2.2 Database Schema

2.3 Key

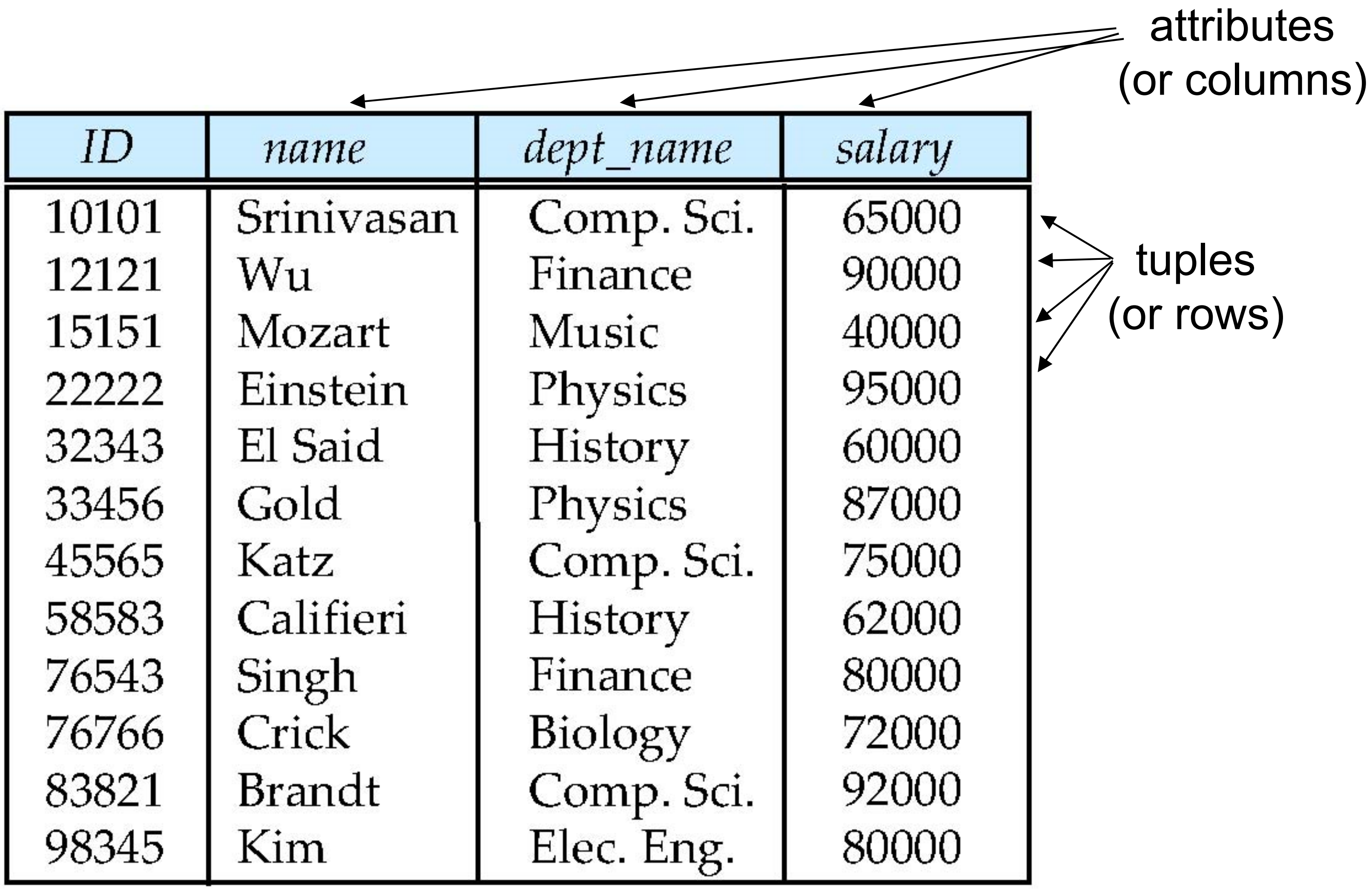
2.4 Schema Diagrams

2.5 Relational Query Languages

2.6 Relational Operations

2.1 Structure of Relational Database

The *Instructor* relation



attributes
(or columns)

tuples
(or rows)

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

2.1 Structure of Relational Database

Other Example (The *course* relation)

attributes
(or columns)

course_id	*title*	*dept_name*	*credit*
BIO-101	*Intro. to Biology*	*Biology*	*4*
BIO-301	*Genetics*	*Biology*	*4*
BIO-399	*Computational Biology*	*Biology*	*3*
CS-101	*Intro. to Computer Science*	*Comp. Sci.*	*4*
CS-190	*Game Design*	*Comp. Sci.*	*4*
CS-315	*Robotics*	*Comp. Sci.*	*3*
CS-319	*Image Processing*	*Comp. Sci.*	*3*
CS-347	*Database System Concepts*	*Comp. Sci.*	*3*

tuples
(or rows)

2.1 Structure of Relational Database

Table == Relation (**Table을 Relation이라고 부름**)

The prereq relation (선행과목)

course_id	*prereq_id*
BIO-301	*BIO-101*
BIO-399	*BIO-101*
CS-190	*CS-101*
CS-319	*CS-101*
CS-347	*CS-101*
EE-181	*PHY-101*

Structure of Relational Database (Cont.)

Relation

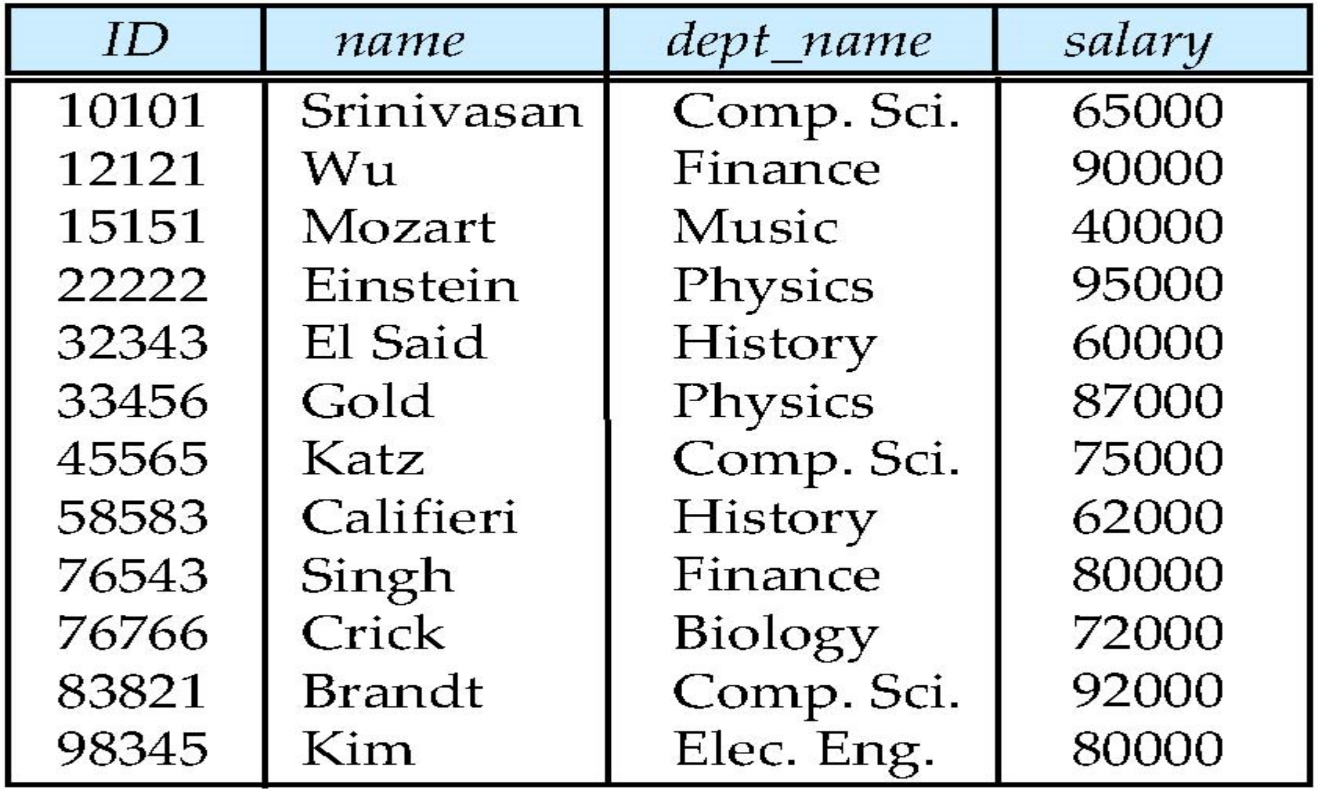
Refer to a table

Attribute

Refer to a column of a table

Tuple (or record)

Refer to a row in a table



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

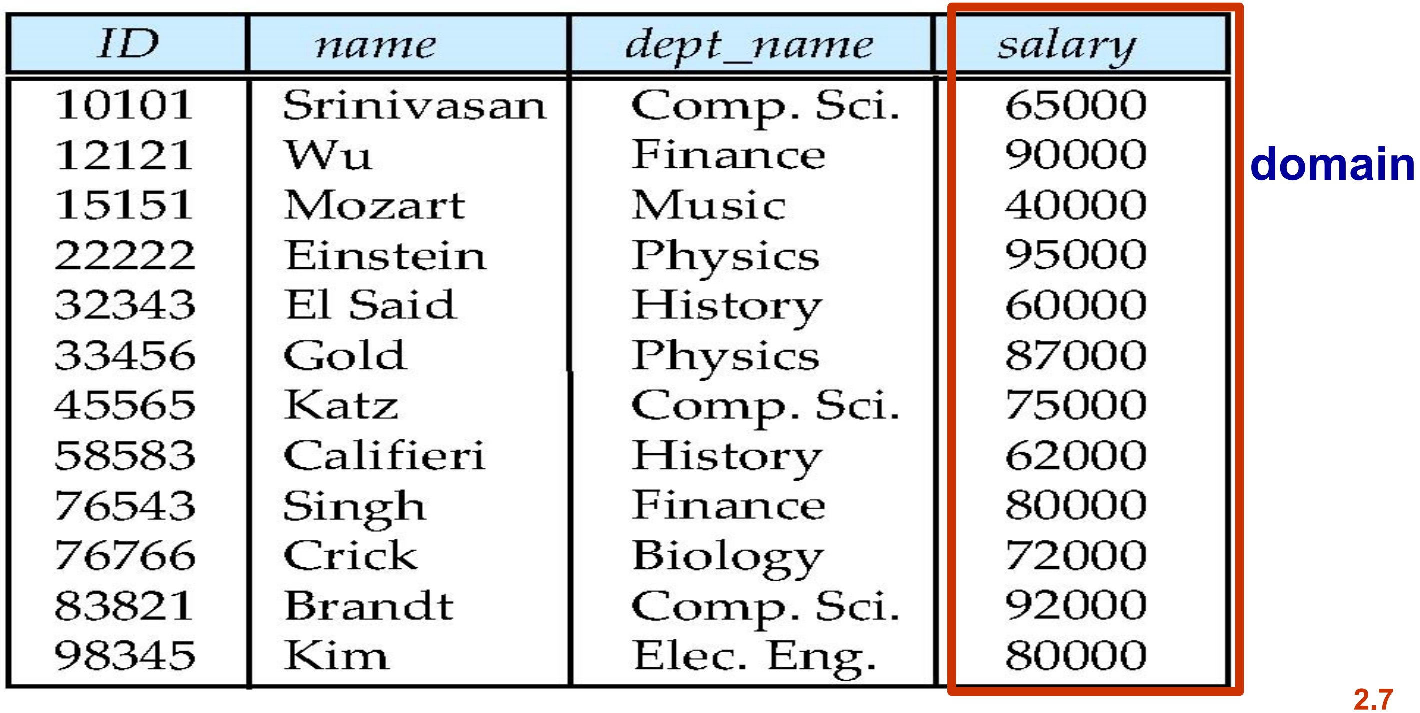
Structure of Relational Database (Cont.)

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

더이상 쪼갤수 없는(분류할수 없는) 최소한의 원자단위로 이루어 져야함
예로 전화번호는 원자단의 값인것 같지만 지역별로 분류할수도 있음

- The special value ***null*** is a member of every domain
- The null value causes complications in the definition of many operations



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

domain

참고: NULL 값

- 널(null) 값
    - 값이 무엇인지 정의할 수 없는 경우 사용될 수 있다.
  - 알려지지 않은 값 (value unknown)
    - 값이 무엇인지 알 수 없는 경우
      - (예) 어떤 영화 스타의 알려지지 않은 생일
  - 적용 불가능한 값 (value inapplicable)
    - 적합한 값이 존재하지 않는 경우
      - (예) 미혼인 영화 스타의 배우자
  - 보류된 값(value withheld)
    - 값을 알 수 있는 자격이 없는 경우
      - (예) 공개되지 않은 전화번호

2.2 Database Schema

- A database consists of multiple relations
- Database schema
  - Logical design of the database
- Database instance
  - A snapshot of the data in the database at a given instant in time
- Relation schema
  - Logical design of a table
  - A list of attributes and their corresponding domains
- Relation instance
  - A snapshot of the data in a table at a given instant in time

Database Schema (Cont.)

$A_1, A_2, \ldots, A_n$ are *attributes*

$R = (A_1, A_2, \ldots, A_n)$ is a *relation schema*

Example:

instructor = (*ID*, *name*, *dept_name*, *salary*)

Formally, given sets $D_1, D_2, \ldots. D_n$ a **relation *r*** is a subset of

$D_1 \times D_2 \times \ldots \times D_n$

Thus, a relation is a set of *n*-tuples $(a_1, a_2, \ldots, a_n)$ 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

2.3 Keys

Key <= Relation

Let K ⊆ R

K is a **superkey** of *R* if values for *K* are sufficient to identify a unique tuple of each possible relation *r(R)* 유일성을 만족하나 최고성은 만족못한다는 단점이 있음

Example: {*ID*} and {ID,name} are both superkeys of *instructor.*

Candidate key is minimal superkey for which no proper subset is a superkey

최소성을 만족하는 슈퍼키의 부분집합

즉, 최소의 갯수의 키로 슈퍼키와 상응하는 권한을 가지는 슈퍼키의 집합

Example: {*ID*} is a candidate key for *Instructor*

Is {ID, name} a candidate key for Instructor ? No !!

Is {name, dept_name} a candidate key for Instructor ? Yes!!

One of the candidate keys is selected to be the **primary key**.

which one 주키로 선정할때는 고유성과 불변성을 가진 후보키를 선택해야한다

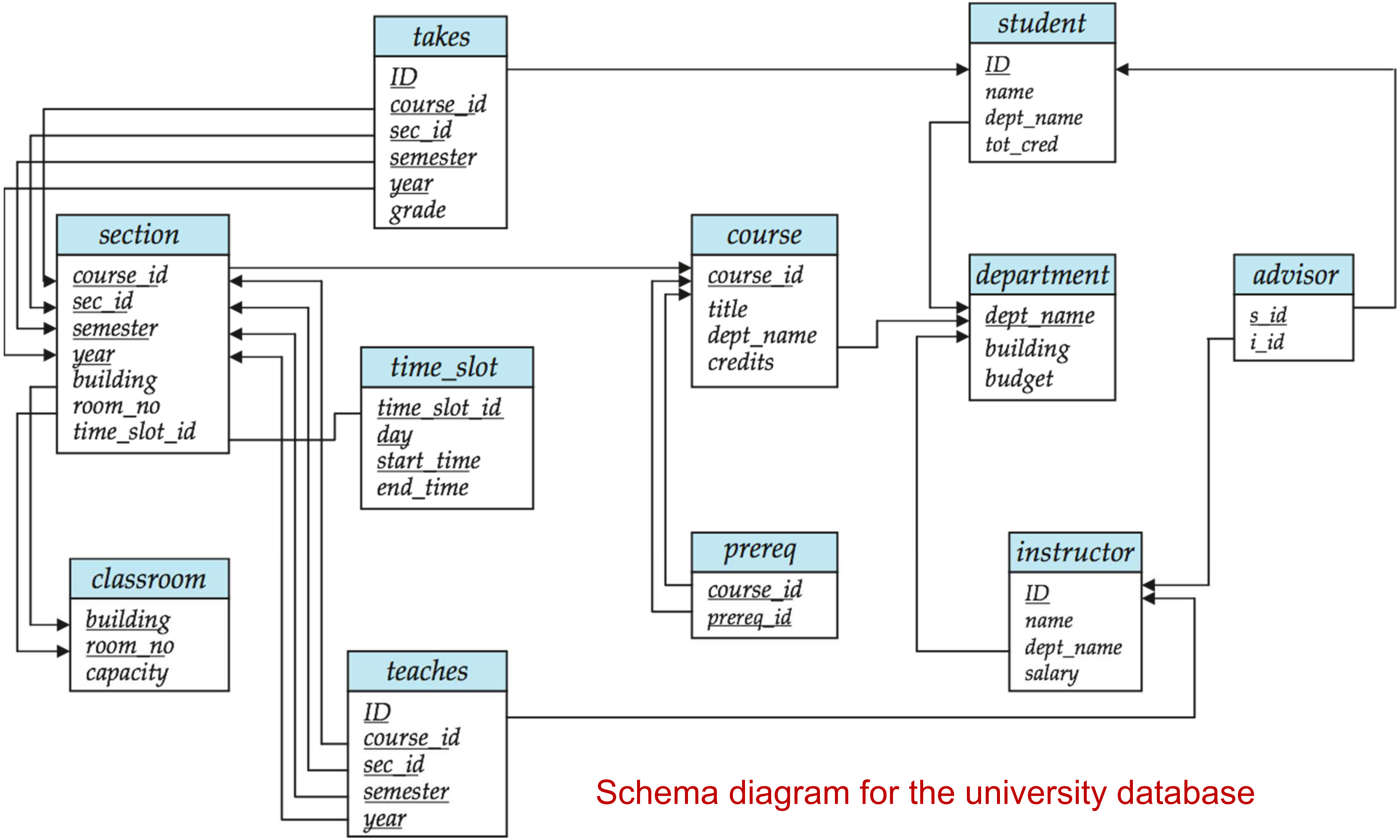
Foreign key constraint: Value in one relation must appear in another

Referencing relation

Referenced relation

2.4 Schema Diagram

A database schema , along with primary key and foreign key dependencies, can be depicted by **schema diagrams** 이런 다이어그램에서 주키는 속성에 밑줄이 가있음



Schema diagram for the university database

2.5 Relational Query Languages

절차식: 원하는 결과를 얻기 위해 사용자가 시스템에 어떤 명령을 수행해야할지 알려줌
비절차식: 사용자가 정보를 얻기 위한 구체적 수행 과정을 제공하지 않고
어떤 정보를 원하는 지에 대한 기술만함

- Procedural vs. non-procedural, or declarative
- "Pure" languages:
  - Relational algebra 절차식
  - Tuple relational calculus 비절차식
  - Domain relational calculus 비절차식
- Relational operators (Symbol)
  - Selection (σ)
  - Project (Π)
  - Union (∪)
  - Difference (–)
  - Intersection (∩)
  - Join
    - Cartesian product (x)
    - Natural Join (⋈)

2.6 Relational Operations - Selection of tuples

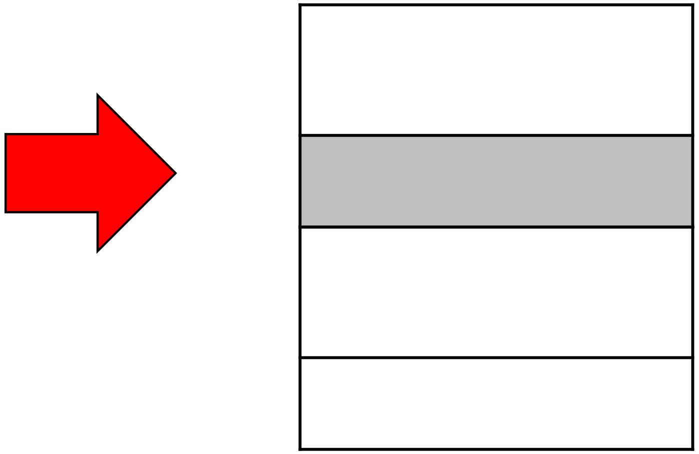
Selection

Extract data horizontally (i.e. as a row)

Chooses the tuples in a relation which satisfy certain conditions

(i.e. it selects rows of the table)

Product Name	***Unit Price***
Melon	*5,000*
Strawberry	*3,000*

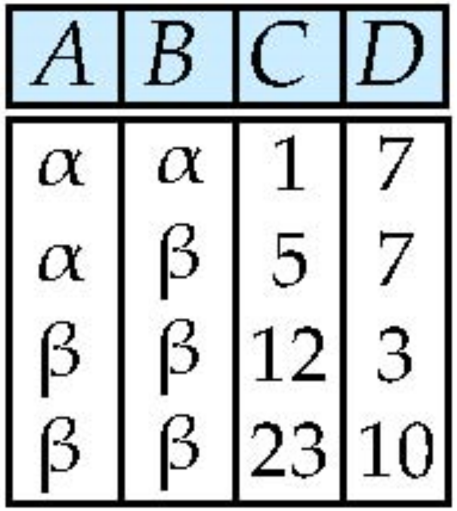


SELECT Product Name
where Unit Price > 4,000

Melon

2.6 Relational Operations - Selection of tuples

Relation r

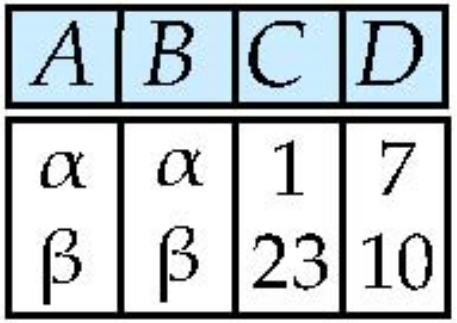


A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

Select tuples with A=B and D > 5

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

select * from table where A=B and D>5;



A	B	C	D
α	α	1	7
β	β	23	10

Selection of Columns (Attributes)

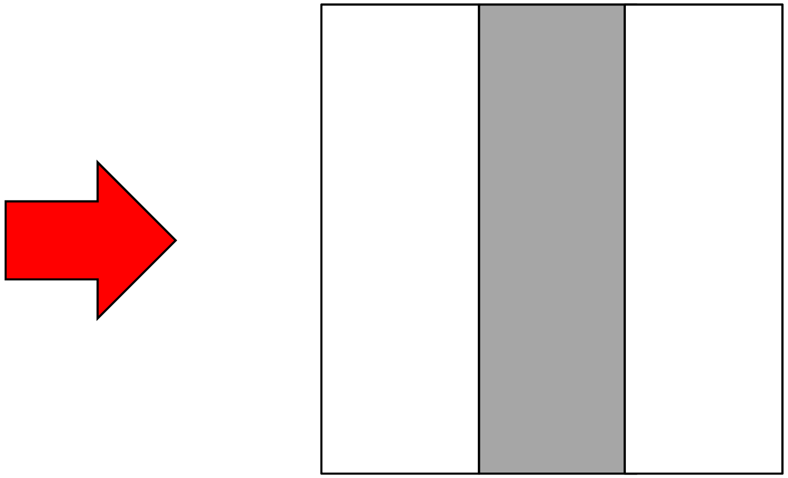
Projection 데이터를 속성별(수직적) 으로 가져옴

Extract data Vertically (i.e. as a column) 값을 집합적(중복x)로 가져옴

Chooses some of the attributes of the relation

(i.e. columns of the table)

Product Name	*Unit Price*
Melon	*5,000*
Strawberry	*3,000*
Apple	*2,000*
Lemon	*1,500*



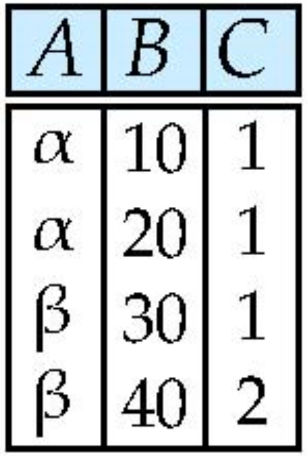
distinct select attr from table;

Project Product Name

Melon
Strawberry
Apple
Lemon

Selection of Columns (Attributes)

Relation *r*:

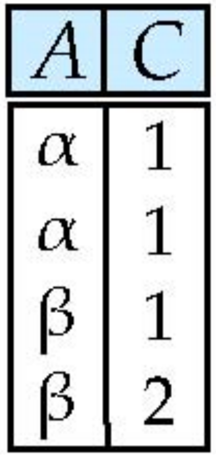


A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

Select A and C

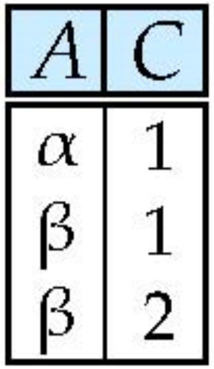
Projection

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



A	C
α	1
α	1
β	1
β	2

=



A	C
α	1
β	1
β	2

Union of two relations

Union

Combine all rows in two relations

Product Name	*Unit Price*
Melon	*5,000*
Strawberry	*3,000*
Apple	*2,000*
Lemon	*1,500*

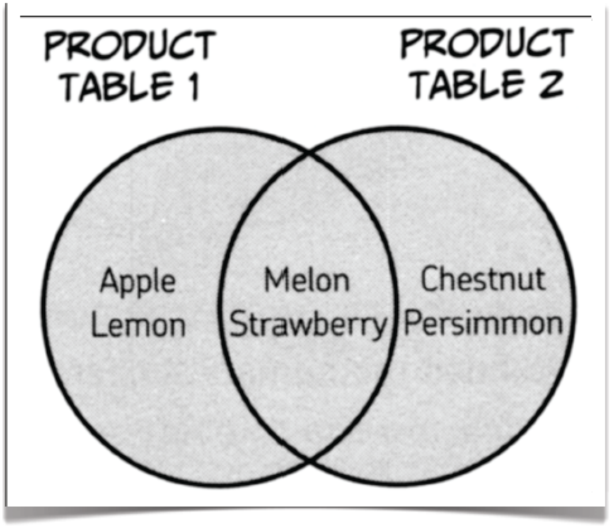
∪

Product Name	*Unit Price*
Melon	*5,000*
Strawberry	*3,000*
Chestnut	*1,000*
Persimmon	*500*

↓

Product Name	*Unit Price*
Melon	*5,000*
Strawberry	*3,000*
Apple	*2,000*
Lemon	*1,500*
Chestnut	*1,000*
Persimmon	*500*

Union



Union of two relations

Relations *r, s:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

동일한 컬럼을 가졌을때 가능

r ∪ s:

A	B
α	1
α	2
β	1
β	3

Set difference of two relations (1/2)

Difference

Extracts rows form one of the tables

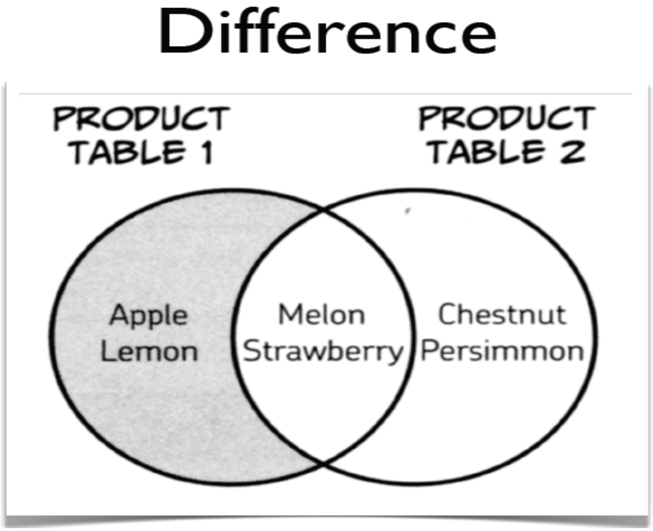
- For Example : All of the products in the first table that don't appear in the second

Product Name	*Unit Price*
Melon	*5,000*
Strawberry	*3,000*
Apple	*2,000*
Lemon	*1,500*

-

Product Name	*Unit Price*
Melon	*5,000*
Strawberry	*3,000*
Chestnut	*1,000*
Persimmon	*500*

Product Name	*Unit Price*
Apple	*2,000*
Lemon	*1,500*



Set difference of two relations (2/2)

Difference

The results depend upon which table contains rows to extract, and which has rows to exclude

- For Example : All of the products in the second table that don't appear in the first

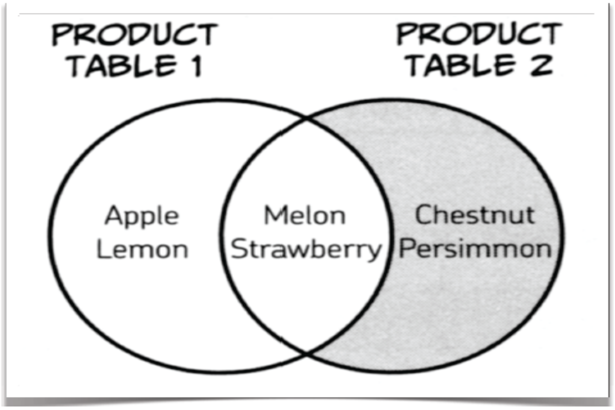
Product Name	*Unit Price*
Melon	*5,000*
Strawberry	*3,000*
Apple	*2,000*
Lemon	*1,500*

-

Product Name	*Unit Price*
Melon	*5,000*
Strawberry	*3,000*
Chestnut	*1,000*
Persimmon	*500*

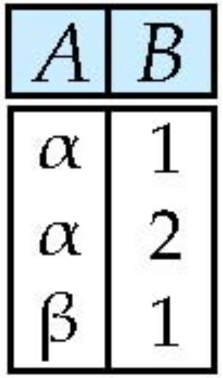
Product Name	*Unit Price*
Chestnut	*1,000*
Persimmon	*500*

Difference (2)



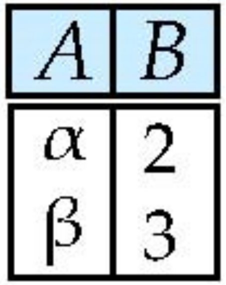
Set difference of two relations

Relations *r*, *s*:



A	*B*
α	1
α	2
β	1

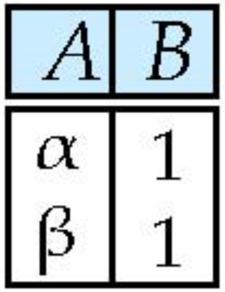
r



A	*B*
α	2
β	3

s

r – *s:*



A	*B*
α	1
β	1

Set Intersection of two relations

Intersection

Extracts products that are includes in both tables

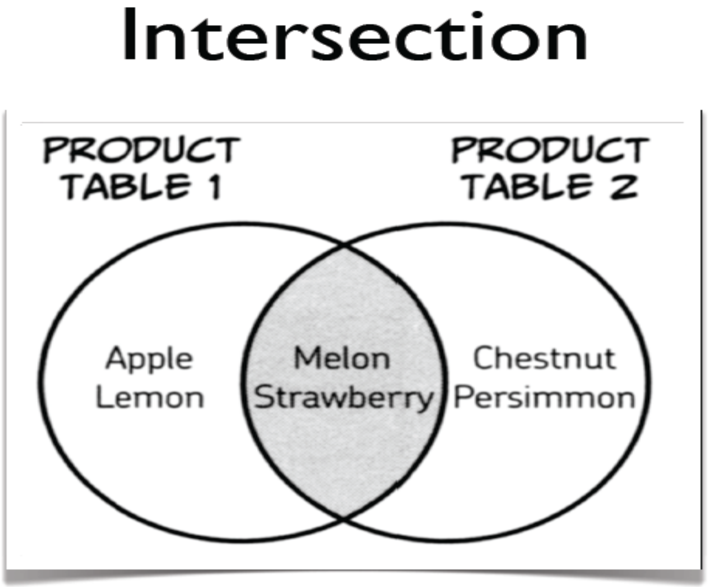
Product Name	*Unit Price*
Melon	*5,000*
Strawberry	*3,000*
Apple	*2,000*
Lemon	*1,500*

∩

Product Name	*Unit Price*
Melon	*5,000*
Strawberry	*3,000*
Chestnut	*1,000*
Persimmon	*500*

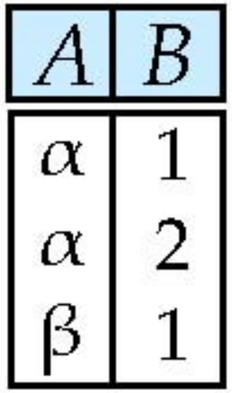
↓

Product Name	*Unit Price*
Melon	*5,000*
Strawberry	*3,000*



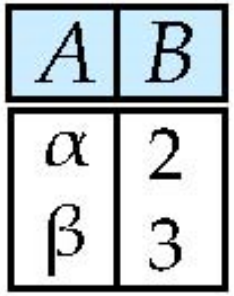
Set Intersection of two relations

Relation *r, s*:



A	B
α	1
α	2
β	1

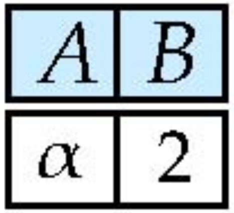
r



A	B
α	2
β	3

s

$r \cap s$

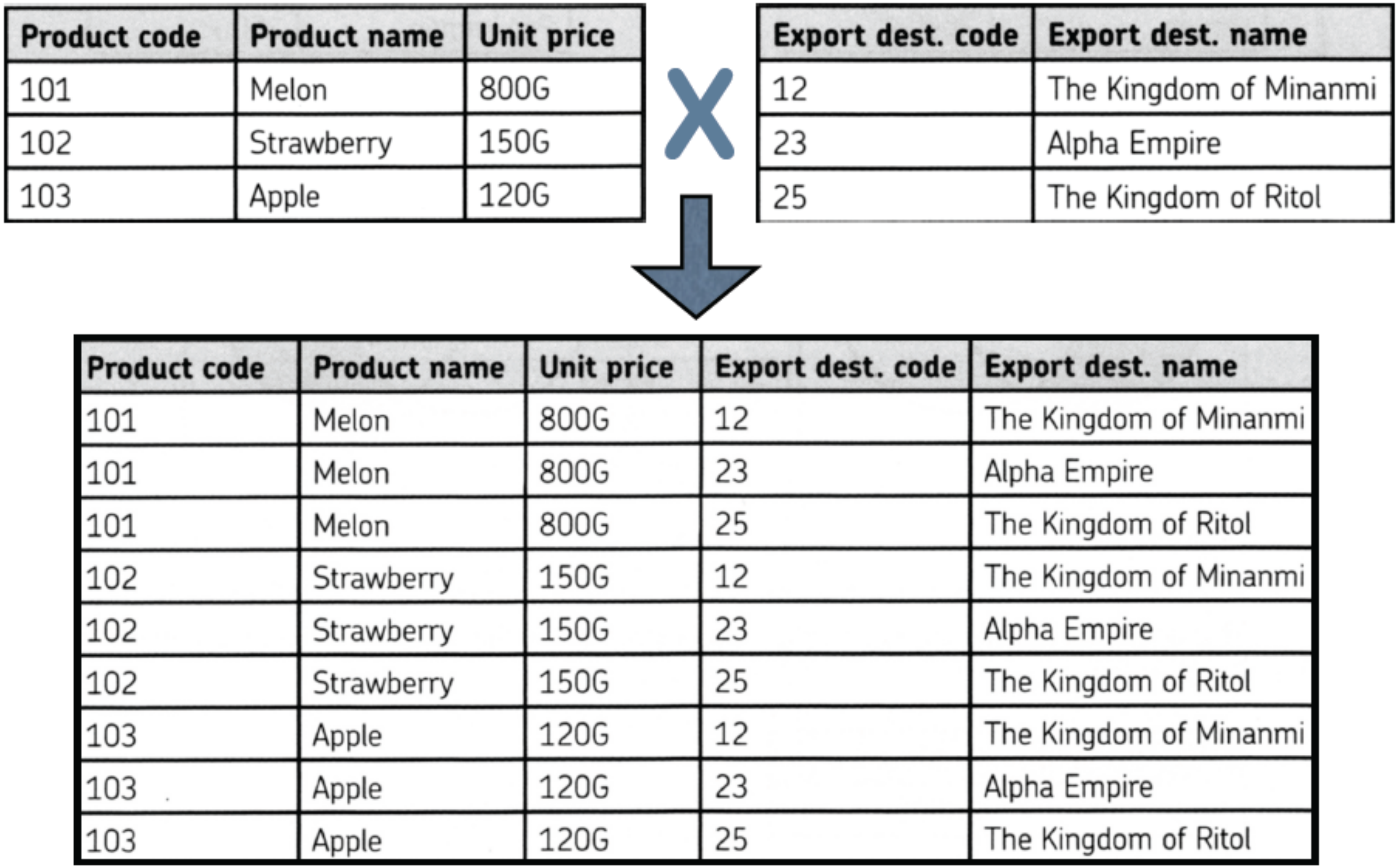


A	B
α	2

Joining two relations – Cartesian Product

Cartesian Product

Combines all rows in the two tables



Product code	Product name	Unit price
101	Melon	800G
102	Strawberry	150G
103	Apple	120G

Export dest. code	Export dest. name
12	The Kingdom of Minanmi
23	Alpha Empire
25	The Kingdom of Ritol

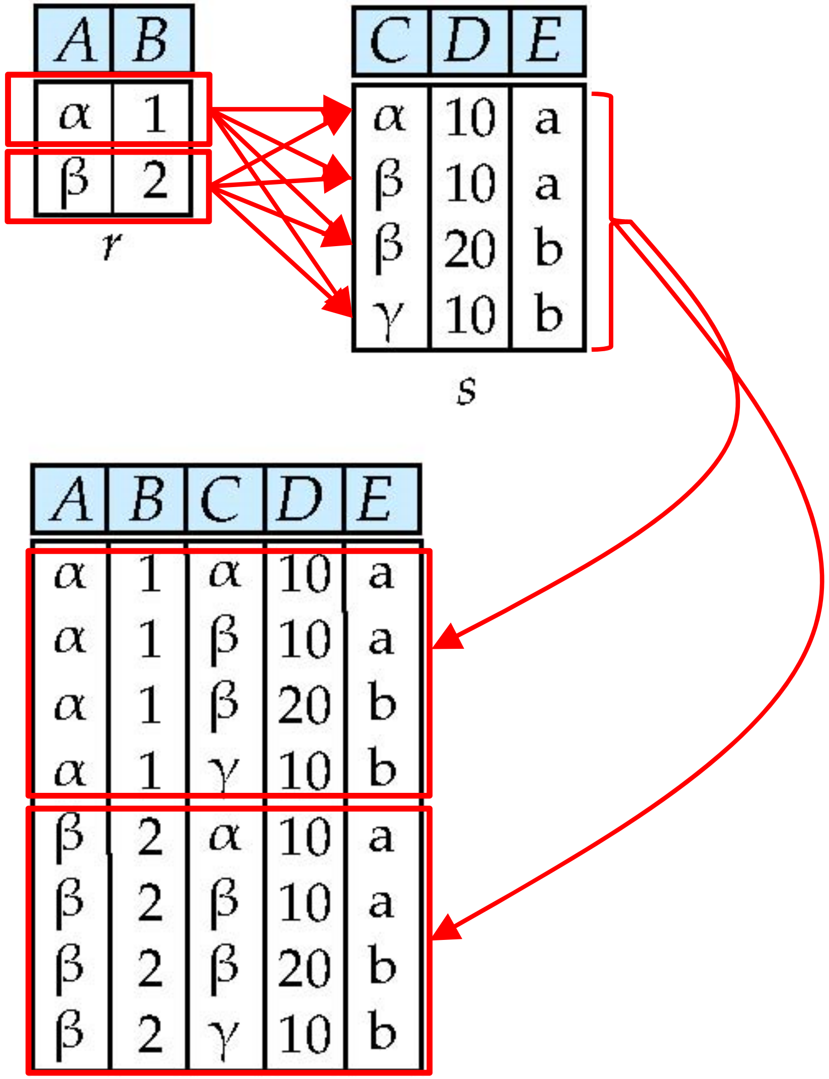
3 rows

Product code	Product name	Unit price	Export dest. code	Export dest. name
101	Melon	800G	12	The Kingdom of Minanmi
101	Melon	800G	23	Alpha Empire
101	Melon	800G	25	The Kingdom of Ritol
102	Strawberry	150G	12	The Kingdom of Minanmi
102	Strawberry	150G	23	Alpha Empire
102	Strawberry	150G	25	The Kingdom of Ritol
103	Apple	120G	12	The Kingdom of Minanmi
103	Apple	120G	23	Alpha Empire
103	Apple	120G	25	The Kingdom of Ritol

3 x 3 =
9 rows

Joining two relations – Cartesian Product

Relations *r, s*:



A	*B*
α	1
β	2

r

C	*D*	*E*
α	10	a
β	10	a
β	20	b
γ	10	b

s

r x *s*:

A	*B*	*C*	*D*	*E*
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Joining two relations – Natural Join

inner join

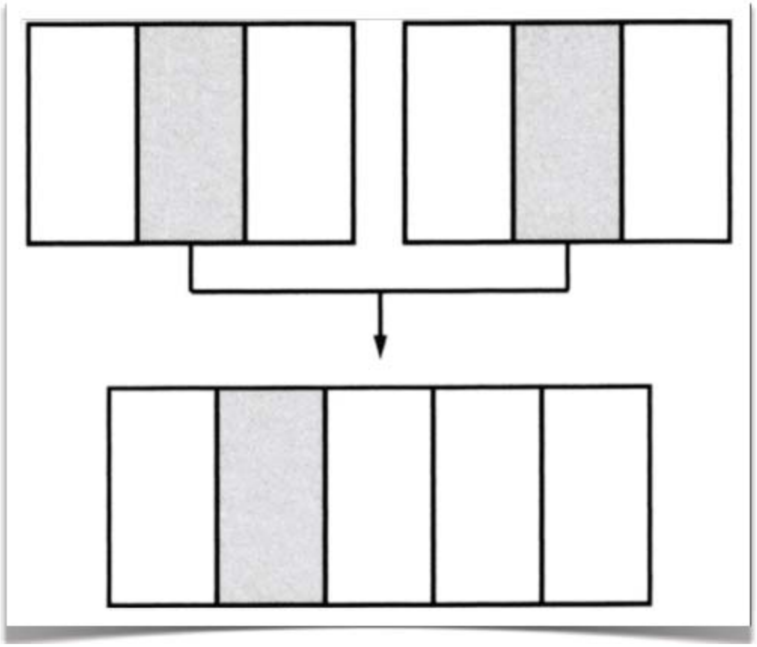
Let *r* and *s* be relations on schemas *R* and *S* respectively. Then, the "natural join" of relations *R* and *S* is a relation on schema $R \cup S$ obtained as follows:

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*

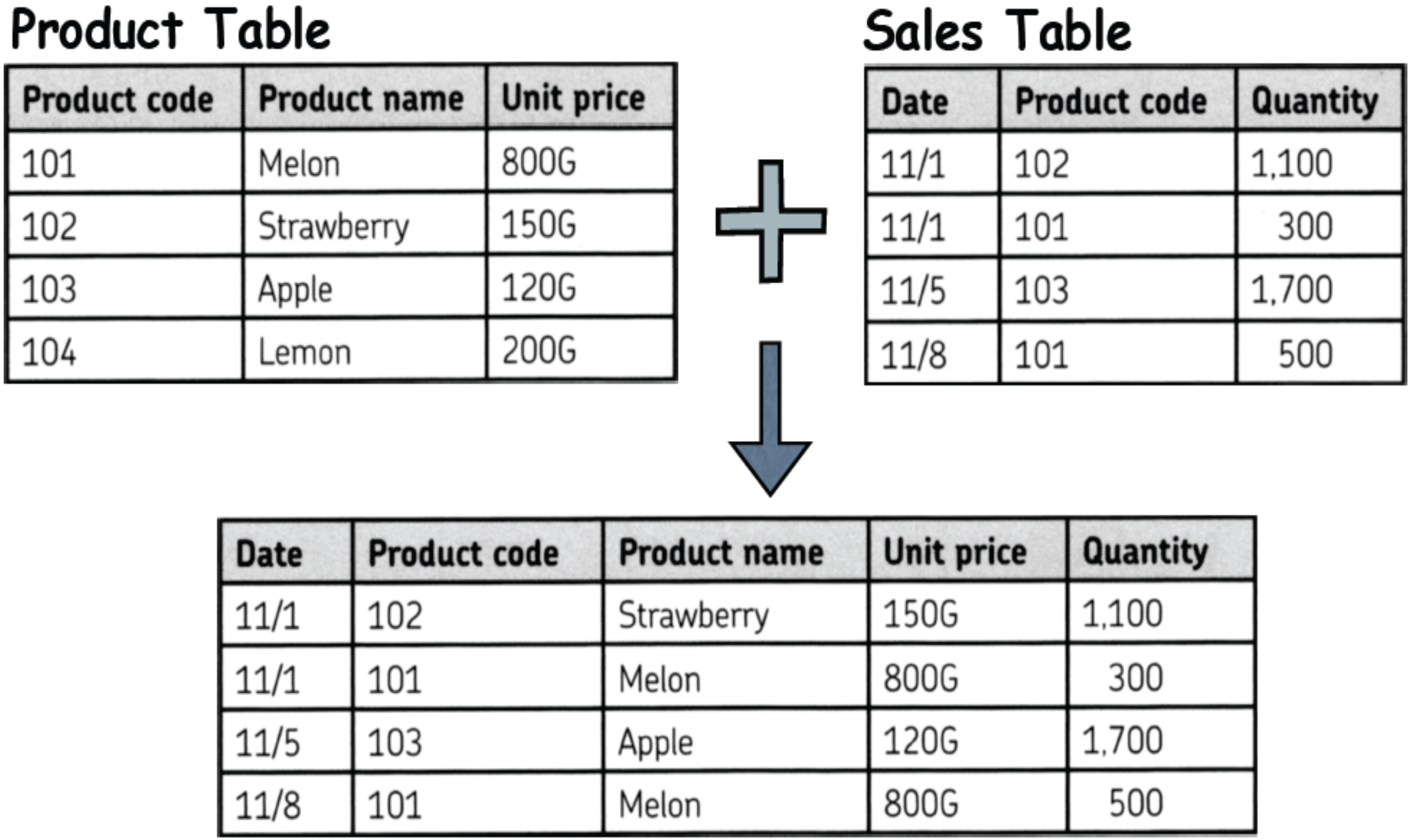
자연조인이란 릴레이션 R과 S에 공통적으로 존재하는 속성들을 이용하여 공통 속성들의 값들이 서로 같은 투플들을 조인하는 것이다.



Natural Join

Joins two tables together using a shared key

(i.e. Primary/Foreign Key)



Product Table

Product code	Product name	Unit price
101	Melon	800G
102	Strawberry	150G
103	Apple	120G
104	Lemon	200G

Sales Table

Date	Product code	Quantity
11/1	102	1,100
11/1	101	300
11/5	103	1,700
11/8	101	500

Date	Product code	Product name	Unit price	Quantity
11/1	102	Strawberry	150G	1,100
11/1	101	Melon	800G	300
11/5	103	Apple	120G	1,700
11/8	101	Melon	800G	500

Natural Join Example

Relations r, s:

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ε

s

Natural Join

r ⋈ s

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

Relational Algebra

Symbol (Name)	Example of Use
σ (Selection)	$\sigma_{\text{salary>=85000}}(\textit{instructor})$
	Return rows of the input relation that satisfy the predicate.
Π (Projection)	$\Pi_{\textit{ID, salary}}(\textit{instructor})$
	Output specified attributes from all rows of the input relation. Remove duplicate tuples from the output.
⋈ (Natural Join)	*instructor* ⋈ *department*
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.
× (Cartesian Product)	*instructor* × *department*
	Output all pairs of rows from the two input relations (regardless of whether or not they have the same values on common attributes)
∪ (Union)	$\Pi_{\textit{name}}(\textit{instructor}) \cup \Pi_{\textit{name}}(\textit{student})$
	Output the union of tuples from the two input relations.