Database

Université Grenoble Alpes

Database Foundations

bahareh.afshinpour@univ-grenoble-alpes.fr

Main reference:
A First Course in Database Systems (and associated material) by
J. Ullman and J. Widom, Prentice-Hall

Content

- Functions of Database Management Systems
- Introduction to data models
  - Relational data model
  - Semi-structured model
- Database design
  - E/R model (UML)
- Database programming
  - Relational algebra + SQL
  - Transactions
  - Embedded SQL

Chapter 1
The words of Database Systems

What is a database?

- **Databases** today are essential to every business.
  - Major Websites (Google, Yahoo, …) or smaller sites that provide information
  - At the core of many scientific investigations
  - …..

What is a database?

A collection of information that exists over a long period of time.

- A **DBMS** is a powerful tool for creating and managing large amounts of data efficiently and allowing it to persist over long periods of time, safely.

The DBMS

- The first commercial database management system appeared in the late 1960s. These systems evolved from **file systems**.

✓Store data over a long period of time

❑Do not guarantee that data can not be lost

❑Do not control access to data from many users

❑.....

The DBMS is expected to:

- DBMSs provide many features that traditional file systems do not.

✓Allow users to create new databases

✓Give users the ability to query data and modify the data

✓Support the storage of very large amounts of data- many terabytes or more

✓Enable durability, the recovery of the database in the face of failures, errors, or intentional misuse

✓Control access to data from many users at once, without allowing unexpected interactions among users

The DBMS

A system for providing efficient, convenient, and safe multi-user storage of and access to massive amounts of persistent data.

- Example: Banking System

Data == Information on branches, accounts, customers, interest rates, transaction histories, etc.

Massive:

-Many gigabytes at least for big banks, more if keep the history of all transactions, even more, if keep images of checks.

Multiuser:

- Many people/programs accessing the same database.

Safe:

– From system failures Example: balance transfer

– From malicious users

Convenient:

– Simple commands to manipulate data: get balance, transfer funds, etc.

Efficient:

-Don’t search all files in order to: get the balance of one account, get all accounts with low balances, etc.

Outline of Database-System Studies

- Part1: Relational Database Modeling
- Part2: Relational Database programming
- Part3: Semi-Structured Data Modeling and Programming
- Part4:Database System Implementation

Chapter 2

The Relational Model of Data

What is a Data Model?

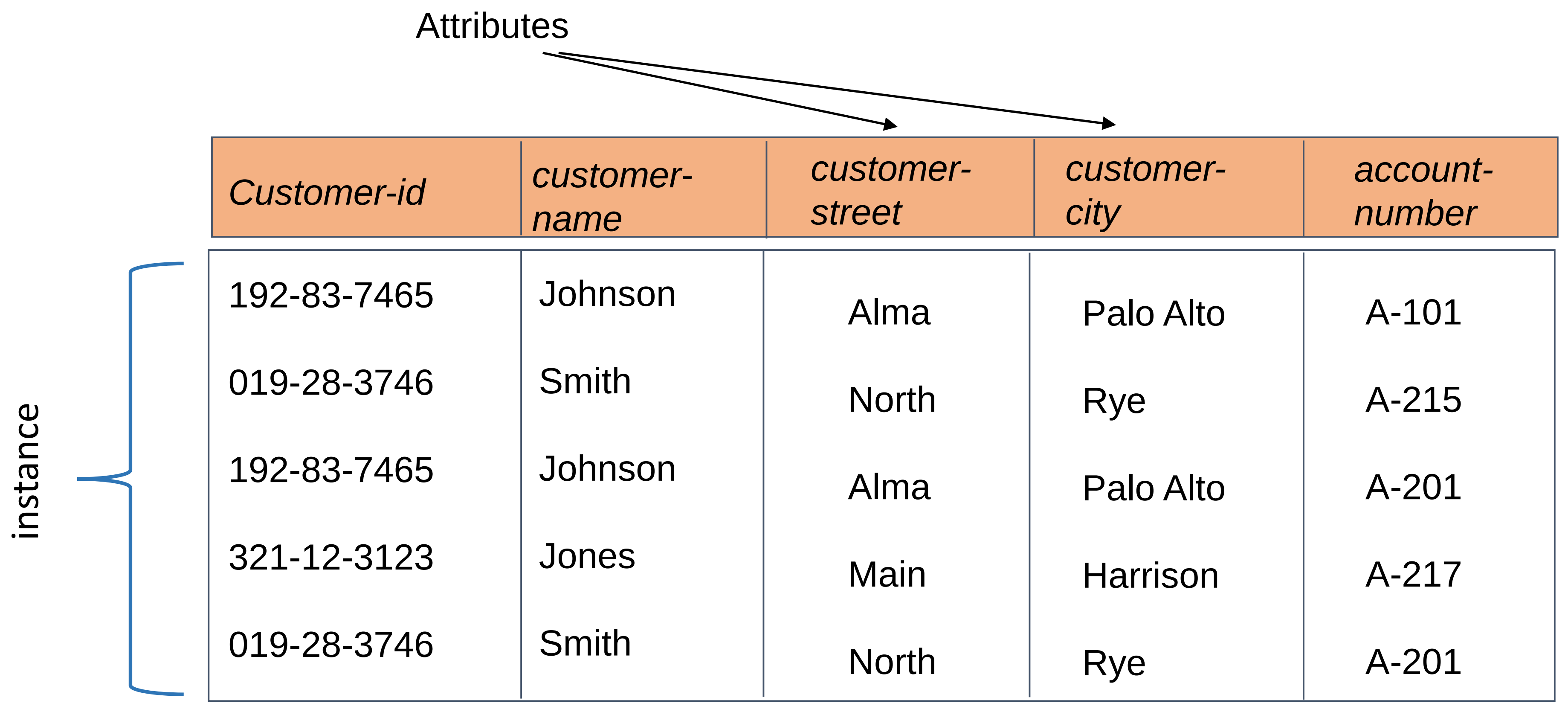
A data model is :

- A notation for describing data or information.
- It describes the conceptual structuring of data stored in the database.

- The two data models of preeminent importance for database systems are:
  - -The relational model
  - -The semi-structured data model

Relational Model

- Example of tabular data in the relational model


Attributes
instance

Customer-id	*customer-name*	*customer-street*	*customer-city*	*account-number*
192-83-7465	Johnson	Alma	Palo Alto	A-101
019-28-3746	Smith	North	Rye	A-215
192-83-7465	Johnson	Alma	Palo Alto	A-201
321-12-3123	Jones	Main	Harrison	A-217
019-28-3746	Smith	North	Rye	A-201

Basics of the Relational Model

- **Attributes**
  - The columns of a relation are named by attributes.
  - Usually, an attribute describes the meaning of entries in the column below.
- **Schemas**
  - The name of a relation and the set of attributes for a relation is called the schema for the relation.

Bank1(customer-name, customer-name, customer-street, customer-city, account-number)

The set of schemas for the relations of a database is called a **relational database schema**, or a **database schema**.

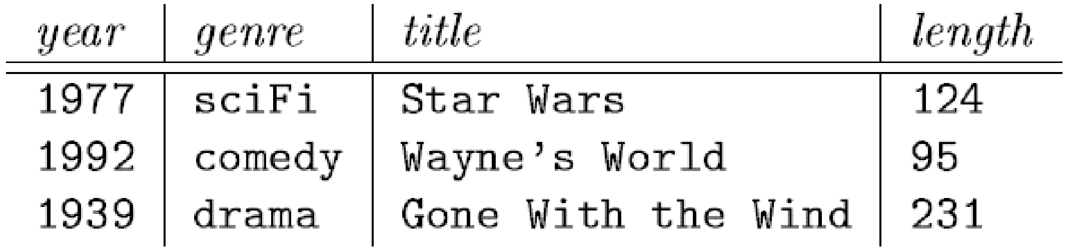
- **Tuples**
  - The rows of a relation are called tuples
  - Relations are a set of tuples not a list of tuples. (the order is immaterial)

Basics of the Relational Model

- **Instances**
  - We shall call a set of tuples for a given relation an instance of the relation.
- **Keys of relations**
  - A set of attributes forms a Key for a relation if we do not allow two tuples in a relation instance to have the same value in all the attributes of the key.
  - We indicate the attribute or attributes that form a key for a relation by underlining the key attribute(s)

➢Ex: Social-security number, Student Id

Movies(title, year, genre,length)



year	*genre*	*title*	*length*
1977	sciFi	Star Wars	124
1992	comedy	Wayne's World	95
1939	drama	Gone With the Wind	231

An example database schema

```
Movies(
    title:string,
    year:integer,
    length:integer,
    genre:string,
    studioName:string,
    producerC#:integer
)
MovieStar(
    name:string,
    address:string,
    gender:char,
    birthdate:date
)
StarsIn(
    movieTitle:string,
    movieYear:integer,
    starName:string
)
```

Exercise

Exercise 2.2.2: In Section 2.2.7 we suggested that there are many examples of attributes that are created for the purpose of serving as keys of relations. Give some additional examples.

!! **Exercise 2.2.3:** How many different ways (considering orders of tuples and attributes) are there to represent a relation instance if that instance has:

a) Three attributes and three tuples

b) Four attributes and five tuples?

c) n attributes and m tuples?

Relations in SQL

- SQL also pronounced (sequel) is the principal language used to describe and manipulate the relational database.
- SQL distinguishes three kinds of relations :
  - **Stored relations (tables):** these are tables that exist in the database we can query and modify them.
  - **Views:** are relations defined by a computation. They are not stored but constructed when needed. We just query them.
  - **Temporary tables:** are constructed by SQL language processors during optimization. These are not stored nor seen by the user.

Data Types

- Char(n): a fixed-length string of up to n characters.
- Varchar(n): a variable-length string of up to n characters
- Bit(n), Varbit(n) fixed, and a variable string of up to n bits.
- Boolean: True False and although it would surprise George Boole Unknown
- Int or Integer: typical integer values
- Float or real: typical real values Decimal(6,2) could be 0123.45
- Date and time: essentially char strings with constraints.

Data Types

Character strings of fixed or varying length. The type `CHAR(n)` denotes a fixed-length string of up to n characters. `VARCHAR(n)` also denotes a string of up to n characters. The difference is implementation-dependent; typically `CHAR` implies that short strings are padded to make n characters, while `VARCHAR` implies that an endmarker or string-length is used. SQL permits reasonable coercions between values of character-string types.

Dates and times can be represented by the data types `DATE` and `TIME`, respectively (see the box on "Dates and Times in SQL"). These values are essentially character strings of a special form. We may, in fact, coerce dates and times to string types, and we may do the reverse if the string "makes sense" as a date or time.

Simple Table Declarations

- The simplest form of declaration of a relation schema consists of the keyword CREATE TABLE followed by the name of the relation and parenthesized, comma-separated list of the attribute names and their types.

```
CREATE TABLE Movies (
    title       CHAR(100),
    year        INT,
    length      INT,
    genre       CHAR(10),
    studioName  CHAR(30),
    producerC#  INT
);
```

Modifying Relation Schemas

- We can delete a table R by the following SQL command

  Drop table R;

- We can modify a table by the command

  Alter Table MovieStar **ADD** phone CHAR(16);

  Alter Table MovieStar **Drop** birthdate;

- Defaults values

  Gender CHAR(1) **DEFAULT** ‘?’,

  Birthdate DATE **DEFAULT** ‘0000-00-00’,

  ALTER TABLE MovieStar ADD phone CHAR (16) **DEFAULT** ‘ unlisted’;

Declaring Keys

- There are two ways to declare an attribute or set of attributes to be a key:

```
CREATE TABLE MovieStar (
    name CHAR(30) PRIMARY KEY,
    address VARCHAR(255),
    gender CHAR(1),
    birthdate DATE
);
```

```
CREATE TABLE MovieStar (
    name CHAR(30),
    address VARCHAR(255),
    gender CHAR(1),
    birthdate DATE,
    PRIMARY KEY (name)
);
```

However, in a situation where the key has more than one attribute, we must use this style

- Replace **primary** with **unique** in examples to get the example with unique

We could also substitute UNIQUE for PRIMARY KEY in this declaration. If we did so, then two or more tuples could have NULL as the value of name, but there could be no other duplicate values for this attribute.

2.3.7 Exercises for Section 2.3

Exercise 2.3.1: In this exercise we introduce one of our running examples of a relational database schema. The database schema consists of four relations, whose schemas are:

```
Product(maker, model, type)
PC(model, speed, ram, hd, price)
Laptop(model, speed, ram, hd, screen, price)
Printer(model, color, type, price)
```

Write the following declarations:

a) A suitable schema for relation `Product`.

b) A suitable schema for relation `PC`.

c) A suitable schema for relation `Laptop`.

d) A suitable schema for relation `Printer`.

e) An alteration to your `Printer` schema from (d) to delete the attribute `color`.

f) An alteration to your `Laptop` schema from (c) to add the attribute `od` (optical-disk type, e.g., cd or dvd). Let the default value for this attribute be `'none'` if the laptop does not have an optical disk.

An Algebraic Query Language

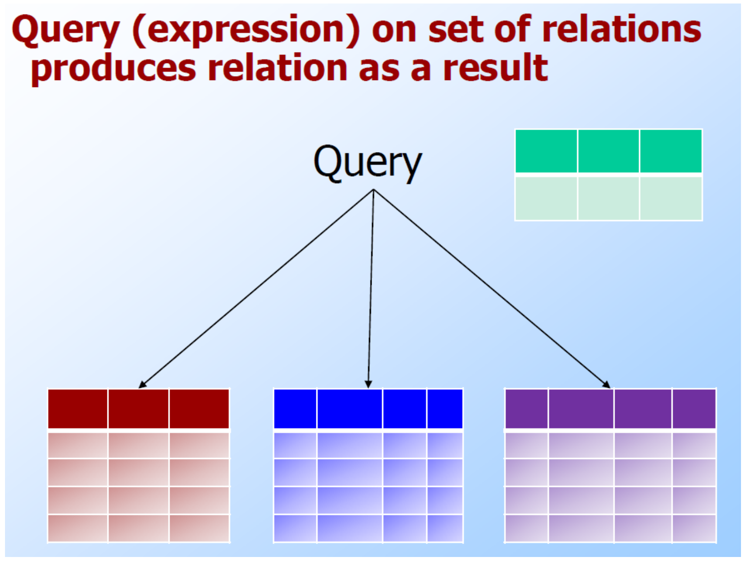
- 2.4.1 -Why Do We Need a Special Query Language?
- 2.4.2 What is Algebra?
- 2.4.3 Overview of Relational Algebra
- 2.4.4 Set Operations on Relations
- 2.4.5 Projection
- 2.4.6 Selection
- 2.4.7 Cartesian Product
- 2.4.8 Natural Joins
- 2.4.9 Theta-Joins
- 2.4.10 Combining Operations to Form Queries
- 2.4.11 Naming and Renaming

Why Do We Need a Special Query Language?

- The surprising answer is that Relational algebra is useful because it is less powerful than other languages like C and Java.
- Being less powerful is helpful because
    - Ease of programming
    - Ease of compilation

What is relational Algebra?

- An algebra whose **operands** are relations or variables that represent relations.
- **Operators** are designed to do the most common things that we need to do with relations in a database.
- The result is an algebra that can be used as a query language for relations.


Query (expression) on set of relations produces relation as a result
Query

What is relational Algebra?

- Relational algebra is another example of algebra.
- The operation of traditional relational algebra falls into four classes:
    1. Set operation: union, intersection, differences
    2. Operation that removes part of the relation: selection, projection
    3. Operation that combines the tuples of two relations: cartesian product, join
    4. Renaming

When we apply **set operation** to relations, we need to put some conditions on R and S:

- Domains (types) for each attribute must be the same
- The columns R and S must be ordered

Example

r

A	B
α	1
α	2
β	1

s

A	B
α	2
β	3

$r \cup s$

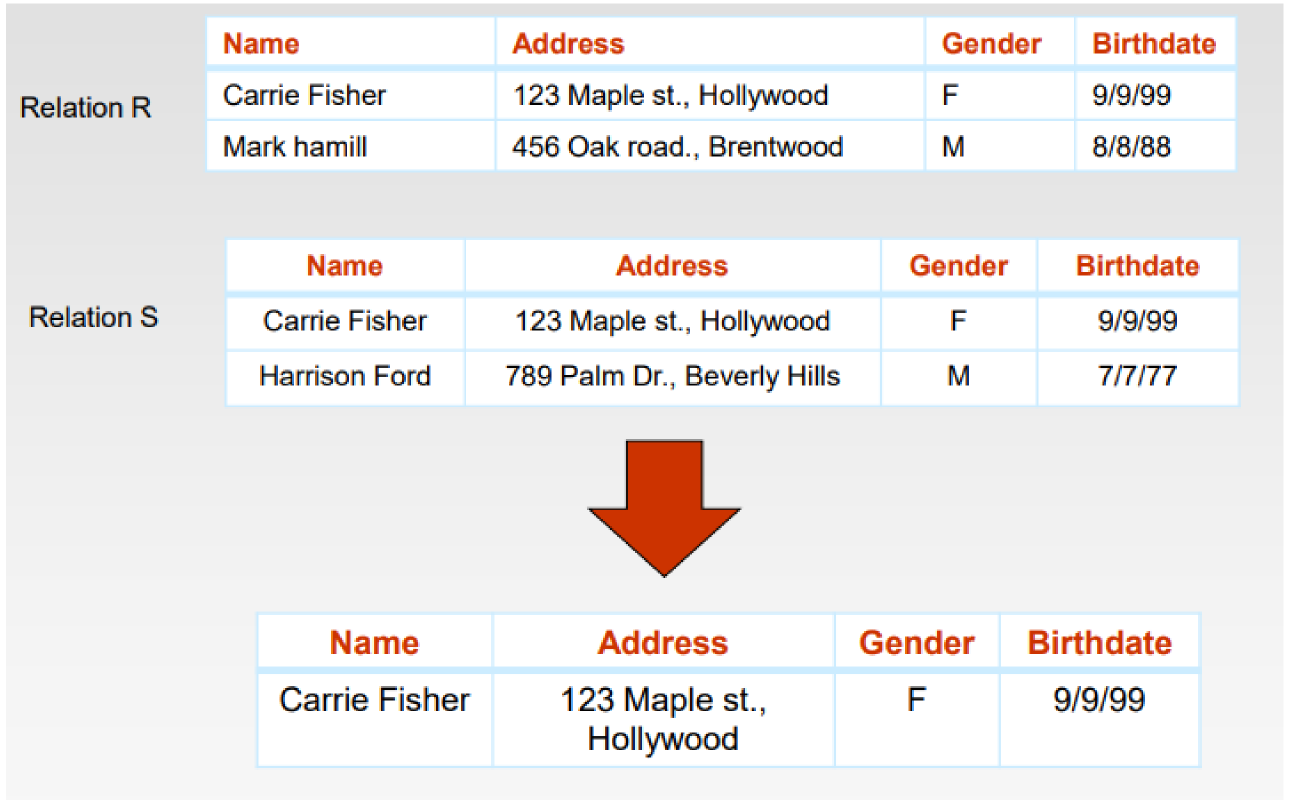
A	B
α	1
α	2
β	1
β	3

For r ∪ s to be applicable,

1. r, s must have the same number of attributes
2. Attribute domains must be compatible
3. attributes do not need to have the same name.
4. Result has attribute names of first relation.

Example

- Intersection (R S): the set of elements that are in both R and S. Appears only once in the intersection.



Relation R

Name	Address	Gender	Birthdate
Carrie Fisher	123 Maple st., Hollywood	F	9/9/99
Mark hamill	456 Oak road., Brentwood	M	8/8/88

Relation S

Name	Address	Gender	Birthdate
Carrie Fisher	123 Maple st., Hollywood	F	9/9/99
Harrison Ford	789 Palm Dr., Beverly Hills	M	7/7/77

Name	Address	Gender	Birthdate
Carrie Fisher	123 Maple st., Hollywood	F	9/9/99

Example

- Denoted by **R – S**
  (Illegal if R & S have different numbers of attributes or if respective domains mismatch!)

R:

A	B
Jane	Toy
Jim	Toy
June	Complaint

S:

A	C
Jane	Toy
John	Complaint

R - S =

A	**B**
Jim	Toy
June	Complaint

- Note attributes in resulting relation take the name from the first relation
- **R – S ≠ S – R**

Projection

- The Projection operator applied to a relation R, produces a new relation with a subset of R's columns.

title	*year*	*length*	*genre*	*studioName*	*producerC#*
Star Wars	1977	124	sciFi	Fox	12345
Galaxy Quest	1999	104	comedy	DreamWorks	67890
Wayne's World	1992	95	comedy	Paramount	99999

The relation Movies

$\pi_{title,year,length}(\texttt{Movies})$

title	*year*	*length*
Star Wars	1977	124
Galaxy Quest	1999	104
Wayne's World	1992	95

$\pi_{genre}(\texttt{Movies})$

genre
sciFi
comedy

Duplicate tuples are eliminated

Selection

- The selection operator applied to a relation R, produces a new relation with a **subset of R’s tuples**.
- The tuples in the resulting relation are those that satisfy some condition C that involves the attributes R.

title	*year*	*length*	*genre*	*studioName*	*producerC#*
Star Wars	1977	124	sciFi	Fox	12345
Galaxy Quest	1999	104	comedy	DreamWorks	67890
Wayne’s World	1992	95	comedy	Paramount	99999

The relation Movies

$\sigma_{length \geq 100}(\texttt{Movies})$

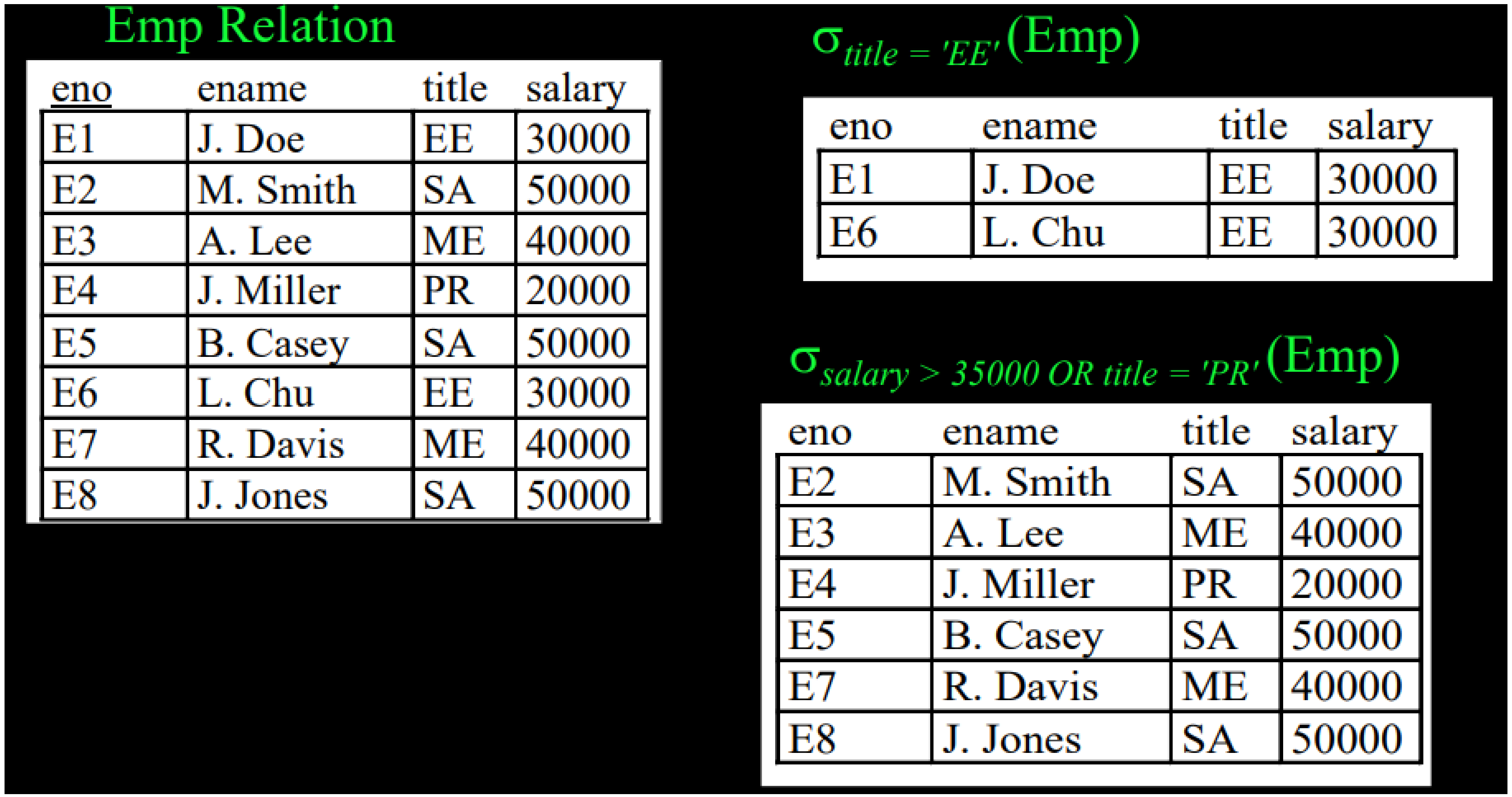
title	*year*	*length*	*inColor*	*studioName*	*producerC#*
Star Wars	1977	124	sciFi	Fox	12345
Galaxy Quest	1999	104	comedy	DreamWorks	67890

Example 2.11 : Suppose we want the set of tuples in the relation `Movies` that represent Fox movies at least 100 minutes long. We can get these tuples with a more complicated condition, involving the `AND` of two subconditions. The expression is

$$\sigma_{length \geq 100 \texttt{ AND } studioName = \texttt{'Fox'}}(\texttt{Movies})$$

title	*year*	*length*	*inColor*	*studioName*	*producerC#*
Star Wars	1977	124	true	Fox	12345

Example



Emp Relation

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

$\sigma_{title\ =\ 'EE'}(\text{Emp})$

eno	ename	title	salary
E1	J. Doe	EE	30000
E6	L. Chu	EE	30000

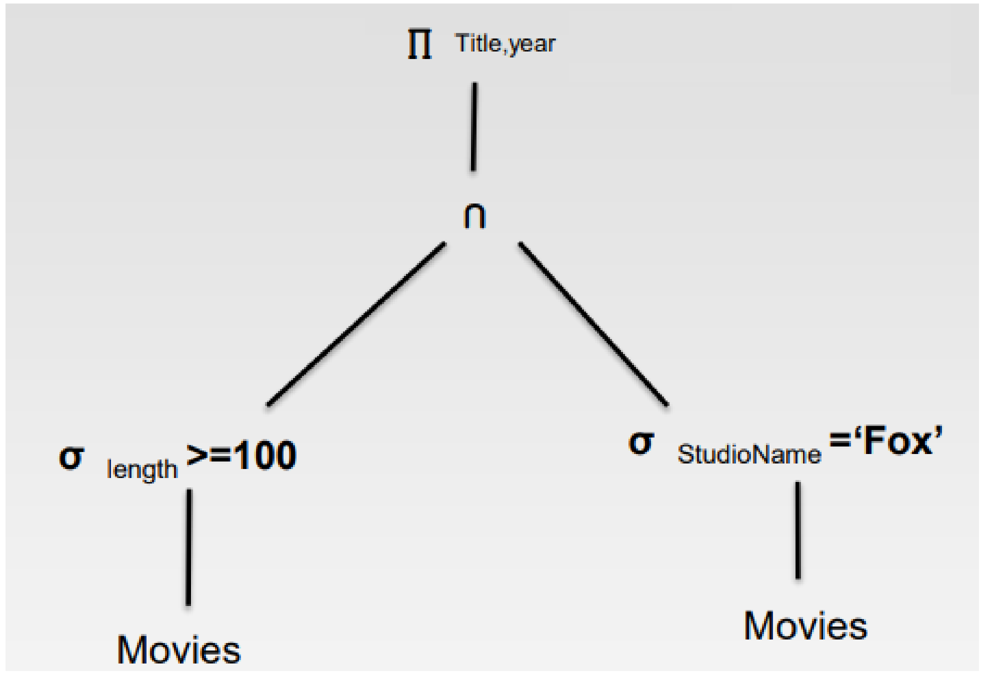
$\sigma_{salary\ >\ 35000\ OR\ title\ =\ 'PR'}(\text{Emp})$

eno	ename	title	salary
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

Combining Operations to Form Queries

- Example: “ What are the titles and years of movies made by Fox that are at least 100 minutes long”

If all we could do was to write single operations on one or two relations as queries, then relational algebra would not be nearly as useful as it is.



1- select those Movie tuples that have length>=100

2- select those Movies tuples that have studioName=‘fax’

3- compute the intersection of (1) and (2)

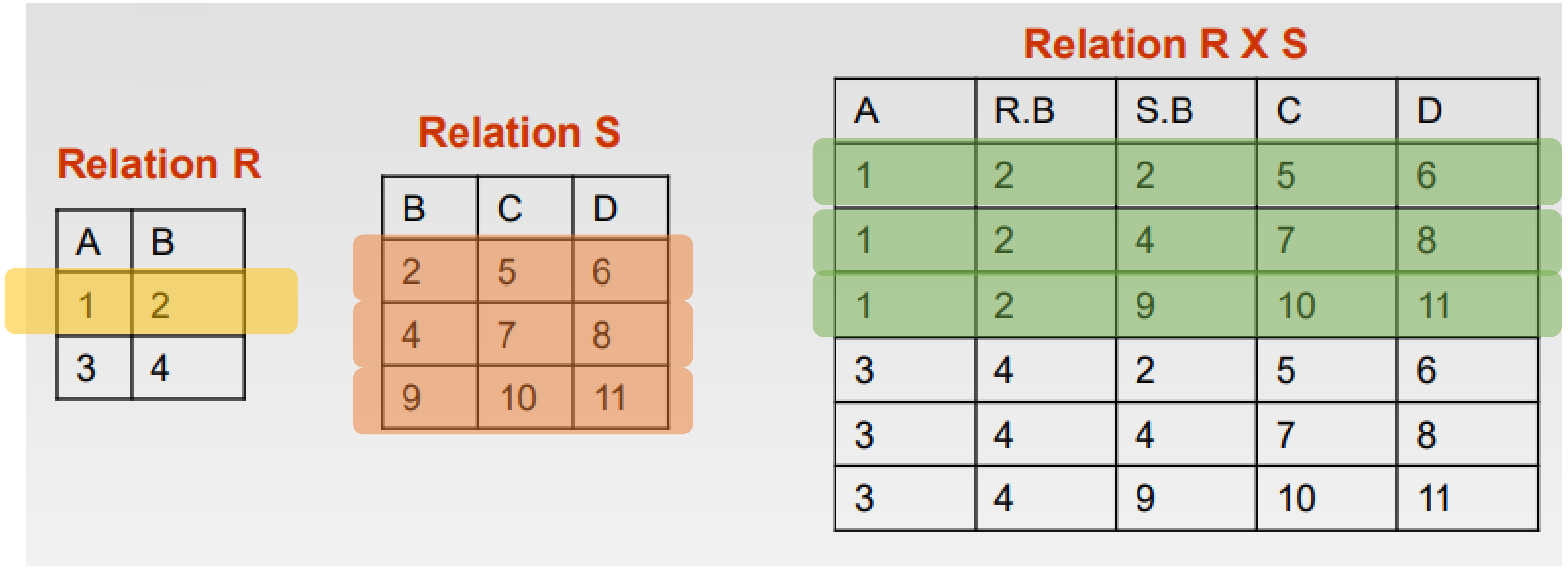
4-project the relation from (3) onto attributes title and year.

$$\Pi_{\text{Title,year}} (\sigma_{\text{length} >=100} (\text{Movies}) \cap \sigma_{\text{StudioName} =\text{'Fox'}} (\text{Movies})$$

$$\Pi_{\text{Title,year}} (\sigma_{\text{length} >=100 \textbf{ AND } \text{StudioName} =\text{'Fox'}} (\text{Movies})$$

Cartesian Product (cross-product)

- The Cartesian Product of two sets R and S is the set of pairs that can be formed by choosing the first element from R and the second from S.
- If R and S have some attributes in common, we need to invent a new name for the identical attributes.



Relation R

A	B
1	2
3	4

Relation S

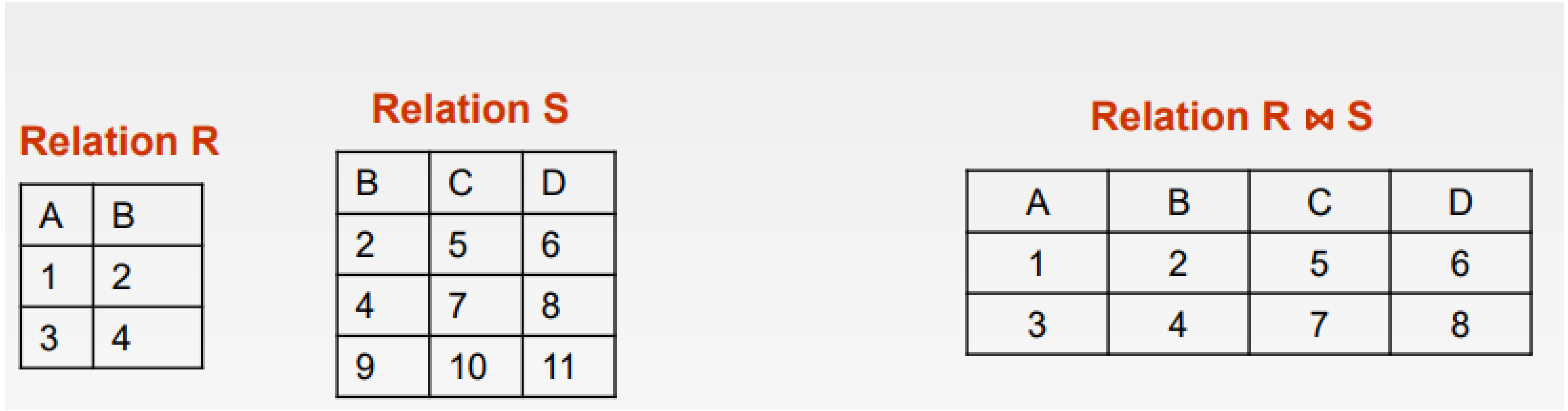
B	C	D
2	5	6
4	7	8
9	10	11

Relation R X S

A	R.B	S.B	C	D
1	2	2	5	6
1	2	4	7	8
1	2	9	10	11
3	4	2	5	6
3	4	4	7	8
3	4	9	10	11

Natural Joins

- The Natural join of two sets R and S is the set of pairs that agree in whatever attributes are common to the schemas of R and S.



Relation R

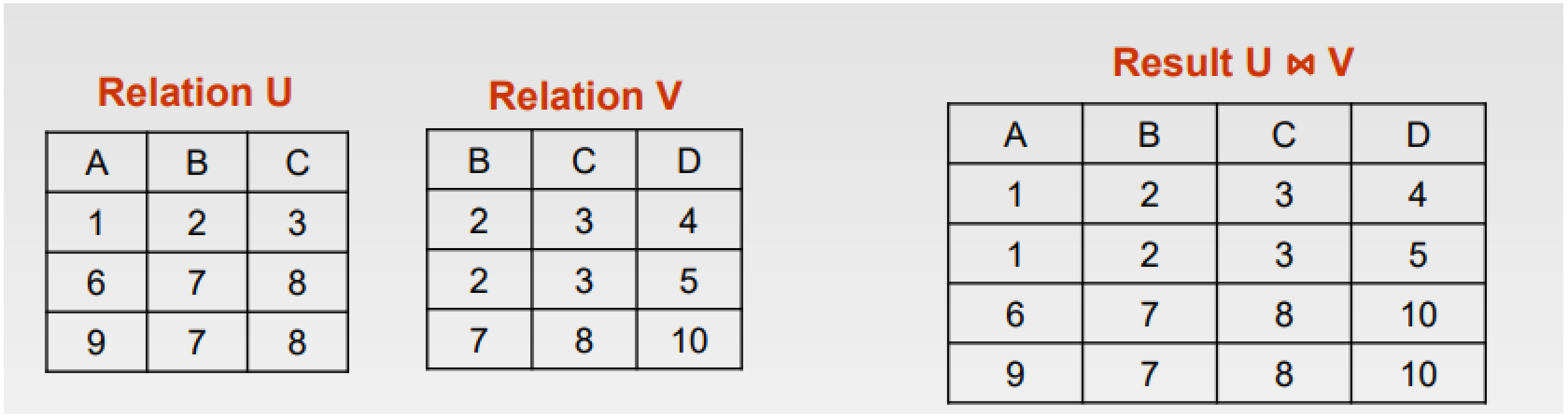
A	B
1	2
3	4

Relation S

B	C	D
2	5	6
4	7	8
9	10	11

Relation R ⋈ S

A	B	C	D
1	2	5	6
3	4	7	8



Relation U

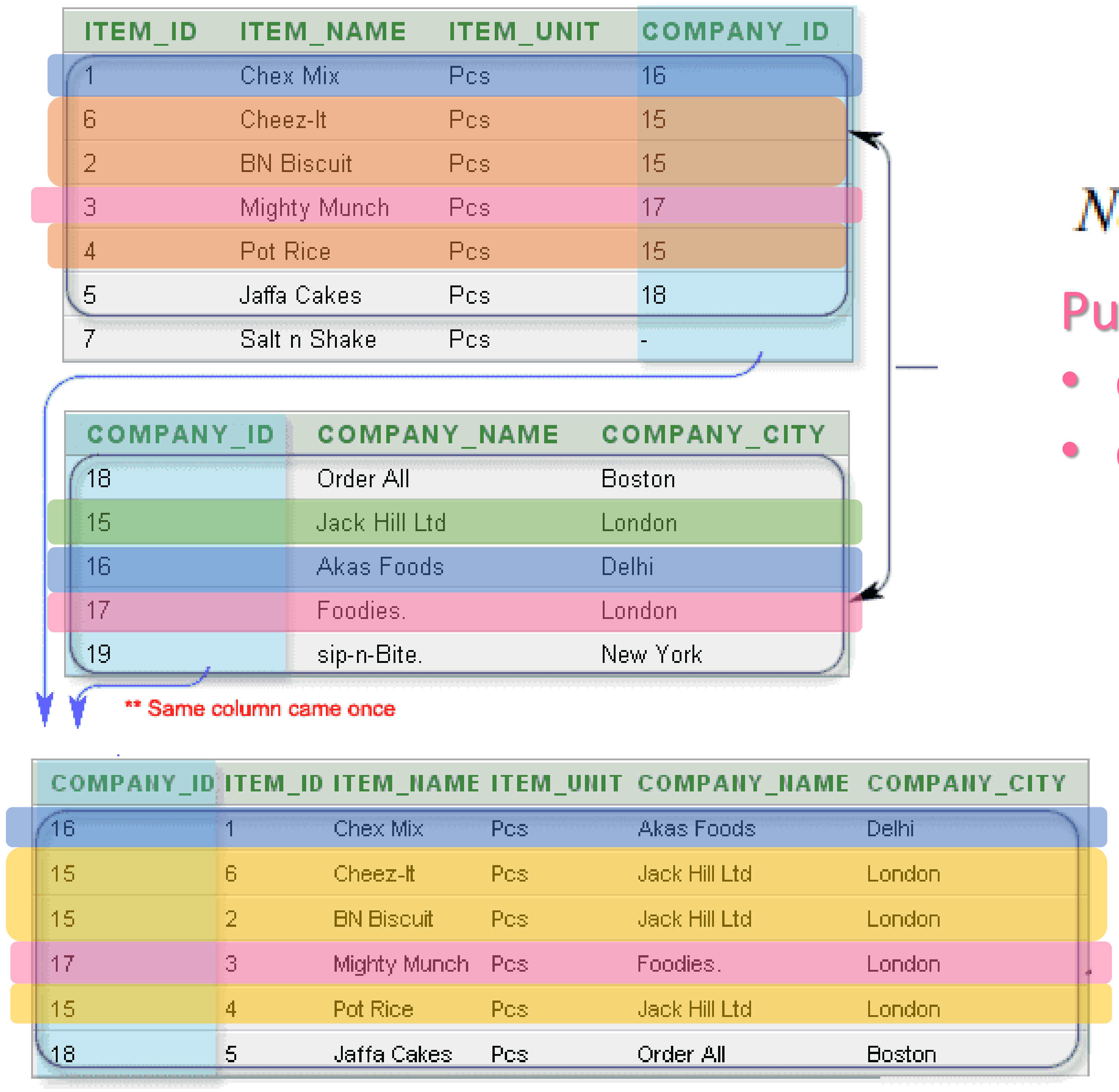
A	B	C
1	2	3
6	7	8
9	7	8

Relation V

B	C	D
2	3	4
2	3	5
7	8	10

Result U ⋈ V

A	B	C	D
1	2	3	4
1	2	3	5
6	7	8	10
9	7	8	10



ITEM_ID	ITEM_NAME	ITEM_UNIT	COMPANY_ID
1	Chex Mix	Pcs	16
6	Cheez-It	Pcs	15
2	BN Biscuit	Pcs	15
3	Mighty Munch	Pcs	17
4	Pot Rice	Pcs	15
5	Jaffa Cakes	Pcs	18
7	Salt n Shake	Pcs	-

COMPANY_ID	COMPANY_NAME	COMPANY_CITY
18	Order All	Boston
15	Jack Hill Ltd	London
16	Akas Foods	Delhi
17	Foodies.	London
19	sip-n-Bite.	New York

COMPANY_ID	ITEM_ID	ITEM_NAME	ITEM_UNIT	COMPANY_NAME	COMPANY_CITY
16	1	Chex Mix	Pcs	Akas Foods	Delhi
15	6	Cheez-It	Pcs	Jack Hill Ltd	London
15	2	BN Biscuit	Pcs	Jack Hill Ltd	London
17	3	Mighty Munch	Pcs	Foodies.	London
15	4	Pot Rice	Pcs	Jack Hill Ltd	London
18	5	Jaffa Cakes	Pcs	Order All	Boston

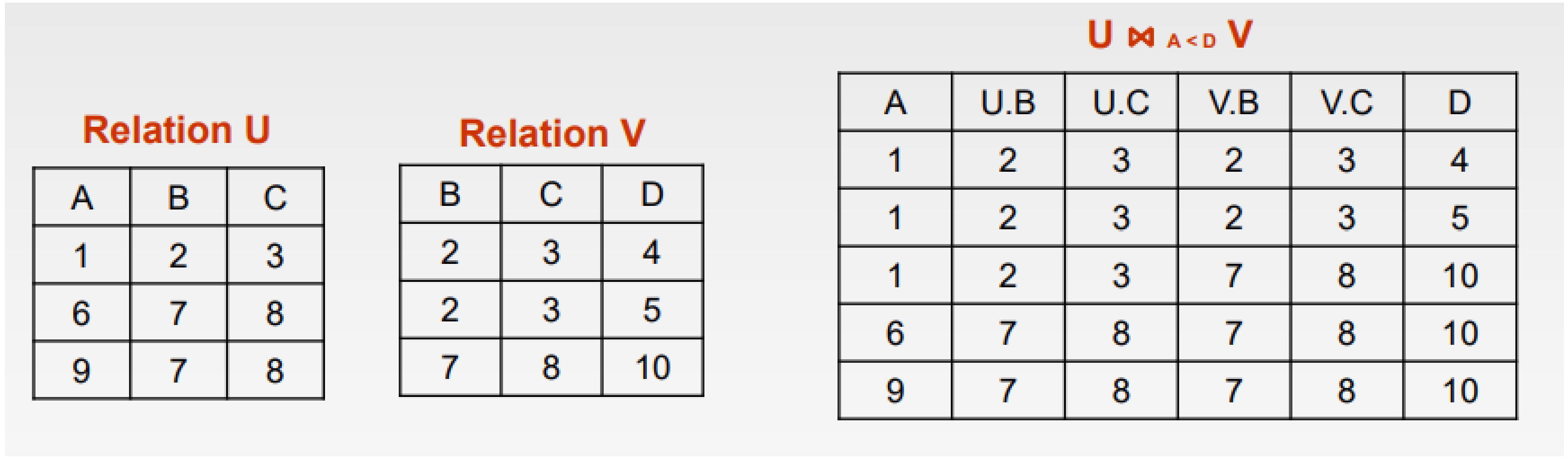
Notation: $R \bowtie S$

Purpose: relate rows from two tables, and

- enforce equality on all common attributes
- eliminate one copy of common attributes

Theta-Joins

- It is sometimes desirable to pair tuples on other conditions except for all the common attributes being equal.
- The notation for a theta-join of relation R and S based on condition C is $R \bowtie_C S$
- The result is constructed as follows:
  - – Take the product of R and S
  - – Select tuples that satisfy C



Relation U

A	B	C
1	2	3
6	7	8
9	7	8

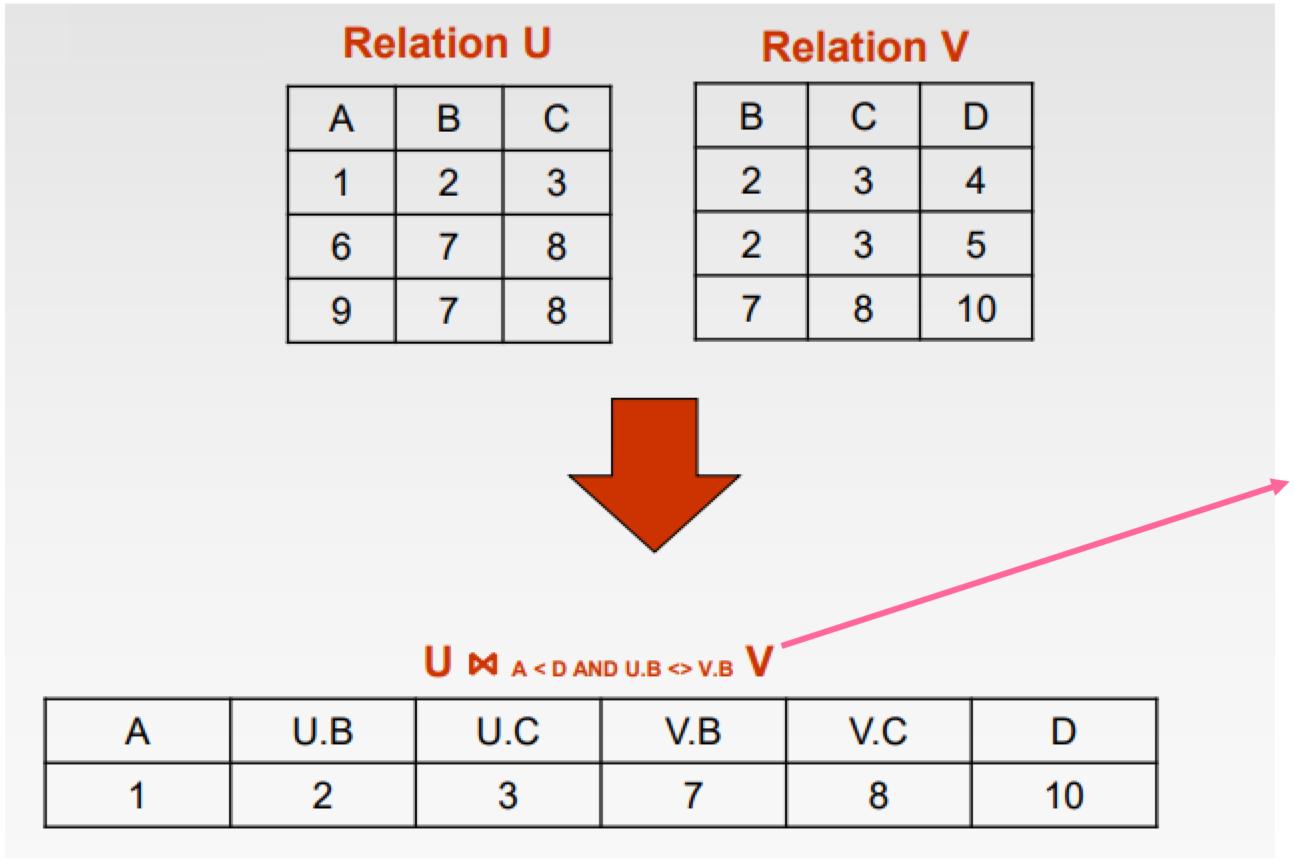
Relation V

B	C	D
2	3	4
2	3	5
7	8	10

$U \bowtie_{A<D} V$

A	U.B	U.C	V.B	V.C	D
1	2	3	2	3	4
1	2	3	2	3	5
1	2	3	7	8	10
6	7	8	7	8	10
9	7	8	7	8	10

Example



Relation U

A	B	C
1	2	3
6	7	8
9	7	8

Relation V

B	C	D
2	3	4
2	3	5
7	8	10

$U \bowtie_{A < D \text{ AND } U.B <> V.B} V$

A	U.B	U.C	V.B	V.C	D
1	2	3	7	8	10

Not only must the A component of the U-tuple be less than the D component of the V-tuple for pairing to be successful, but the two tuples must also disagree on their respective B components

Renaming

- Operator to explicitly rename attributes in relations.

$$\rho_{S(A_1, A_2, \ldots, A_n)}(R)$$

- The resulting relation has exactly the same tuples as R, but the name of the relation is S. Moreover, the attributes of the result relation S are named A1, A2,…, An.

If we only want to change the name of the relation to S and leave the attributes as they are in R, we can just say $\rho_S(R)$.

New name

$$\rho_{\text{result(EmployeeName, ProjectNum, Duration)}}(\text{empwo})$$

New attribute name

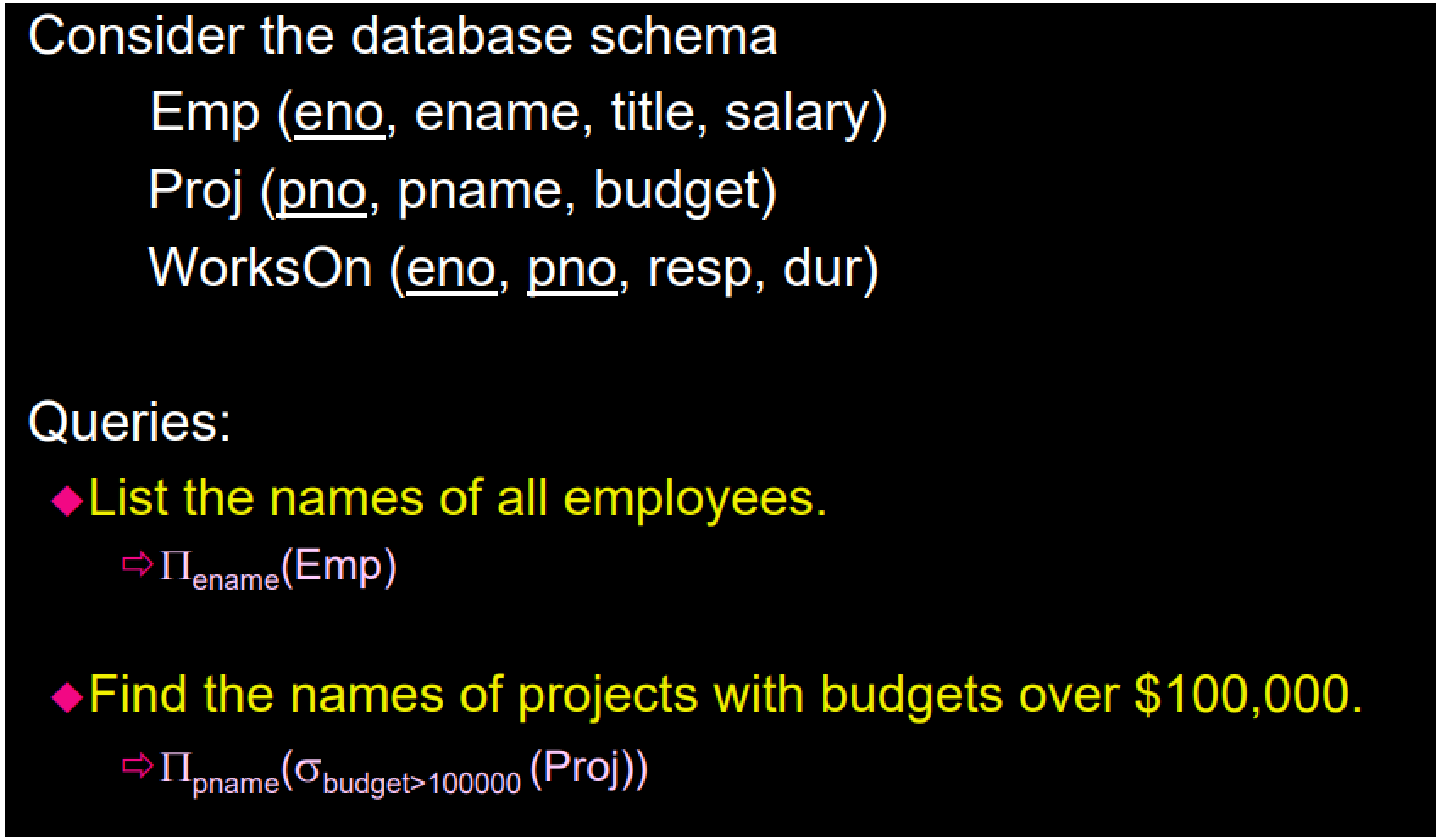
Complete Set of Relational Algebra Operators

- It has been shown that the relational operators {σ, Π, ×, ∪, -} form a complete set of operators.
- That is, any of the other operators can be derived from a combination of these 5 basic operators. Examples:
  - Intersection - $R \cap S \equiv R \cup S - ((R - S) \cup (S - R))$
  - We have also seen how a join is a combination of a Cartesian product followed by a selection.

Database Assignment

- Objective: work on database design and querying
- Size of groups: 3 students.
- **Part one**
- **1.** Choose an application domain to work on: each group has to create its own application! Write a text describing the data to be stored in the database. This text should be a maximum 3 / 4 pages long (i.e less than one page). Your application has to be so as to lead to, at least, 6 classes and to require sub-typing.
- 2. Design a UML diagram for your data. Use the notation presented in class (mandatory).

Relational Algebra Query Examples



Consider the database schema

Emp (eno, ename, title, salary)

Proj (pno, pname, budget)

WorksOn (eno, pno, resp, dur)

Queries:

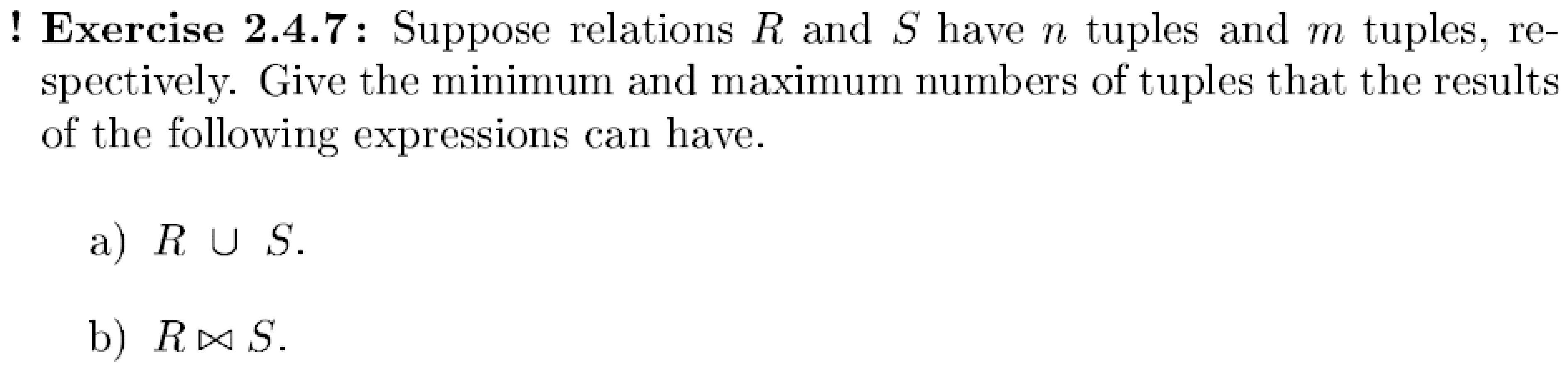
- List the names of all employees.
  - ⇨ $\Pi_{ename}(Emp)$
- Find the names of projects with budgets over $100,000.
  - ⇨ $\Pi_{pname}(\sigma_{budget>100000}(Proj))$

Practice Questions

branch (bname, address, city, assets)
customer (cname, street, city)
deposit (accnum, cname, bname, balance)
borrow (accnum, cname, bname, amount)

- Relational database schema:
- 1) List the names of all branches of the bank.
- 2) List the names of all deposit customers together with their account numbers.
- 3) Find all cities where at least one customer lives.
- 4) Find all cities with at least one branch.
- 5) Find all cities with at least one branch or customer.
- 6) Find all cities that have a branch but no customers who live in that city.

Practice Questions



! **Exercise 2.4.7:** Suppose relations R and S have n tuples and m tuples, respectively. Give the minimum and maximum numbers of tuples that the results of the following expressions can have.

a) $R \cup S$.

b) $R \bowtie S$.