Database

bahareh.afshinpour@univ-grenoble-alpes.fr

Université Grenoble Alpes

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

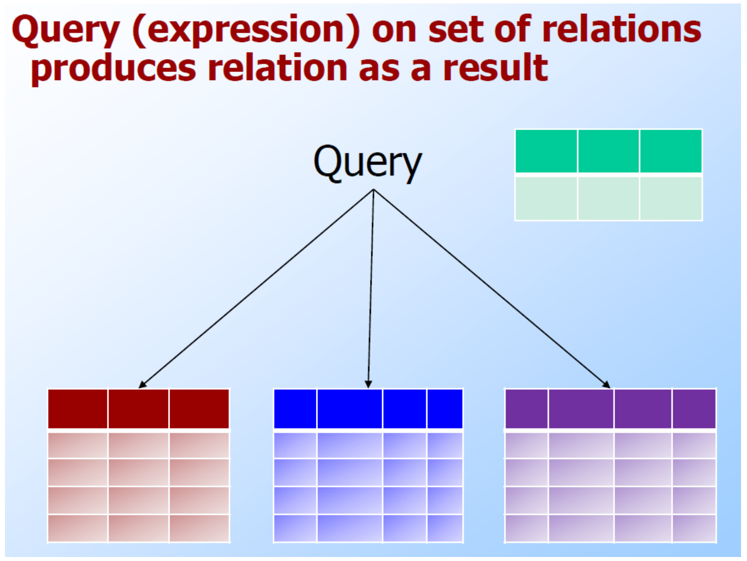
An Algebraic Query Language

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

$r \cup s$

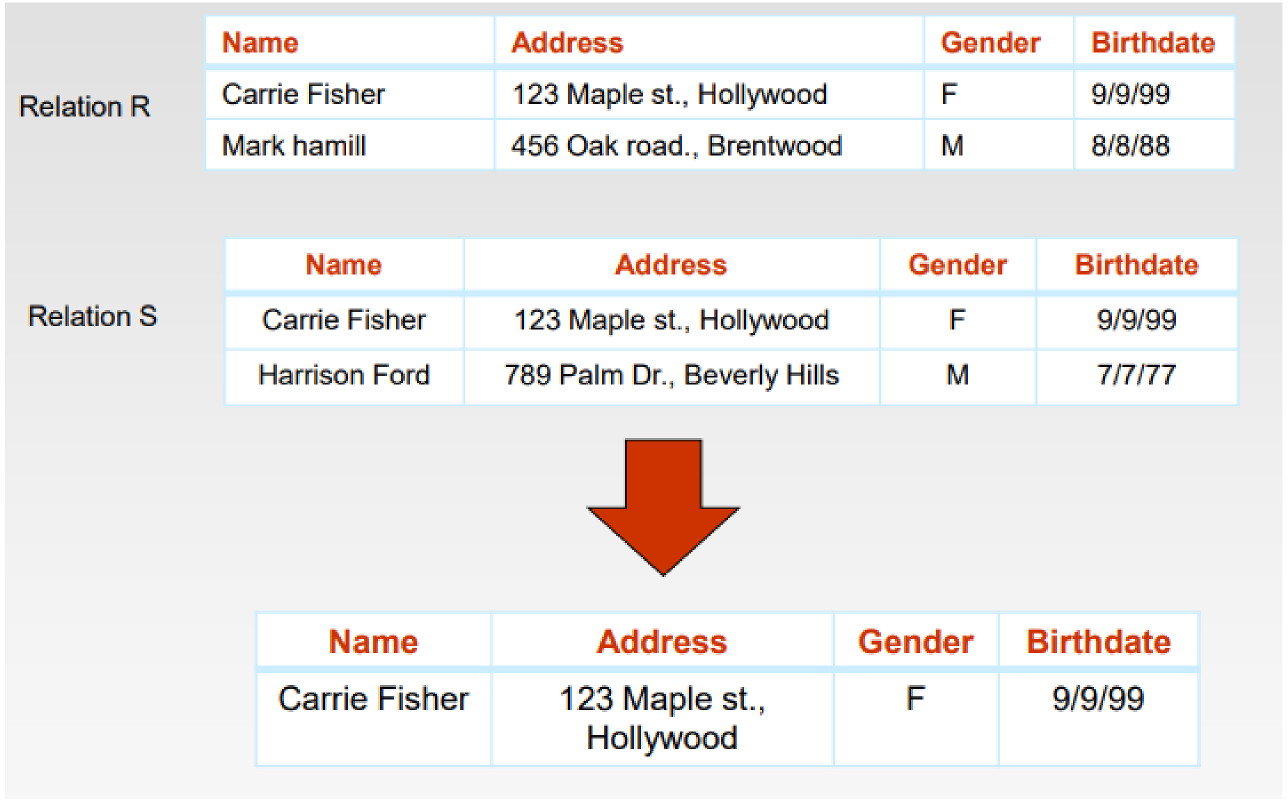
A	B
α	1
α	2
β	1
β	3

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

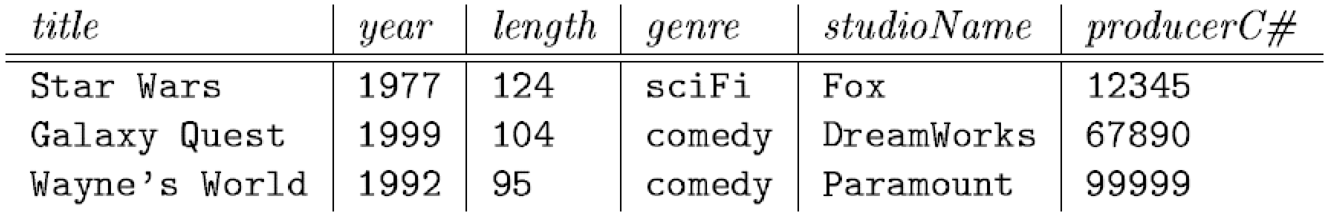
$\pi_{genre}(\texttt{Movies})$

genre
sciFi
comedy

Duplicate tuples are eliminated

Selection

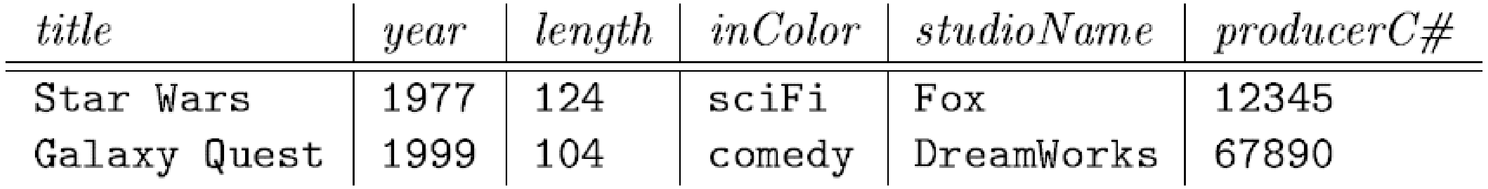
- The selection operator applied to a relation R, produces a new relation with a **subset of R's tuples**.
- The tuples in the resulting relation are those that satisfy some condition C that involves the attributes R.



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

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



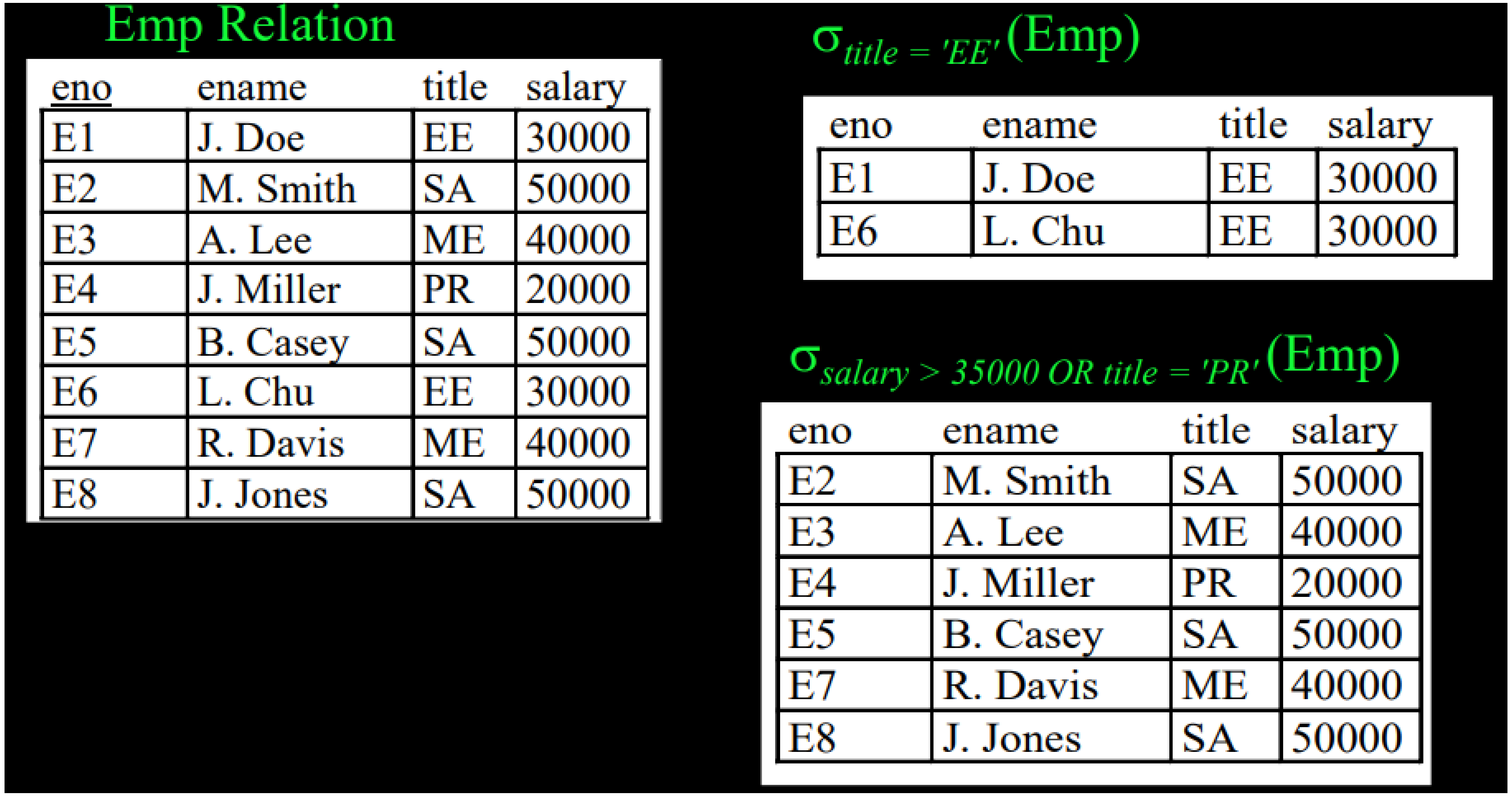
title	*year*	*length*	*inColor*	*studioName*	*producerC#*
Star Wars	1977	124	sciFi	Fox	12345
Galaxy Quest	1999	104	comedy	DreamWorks	67890

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

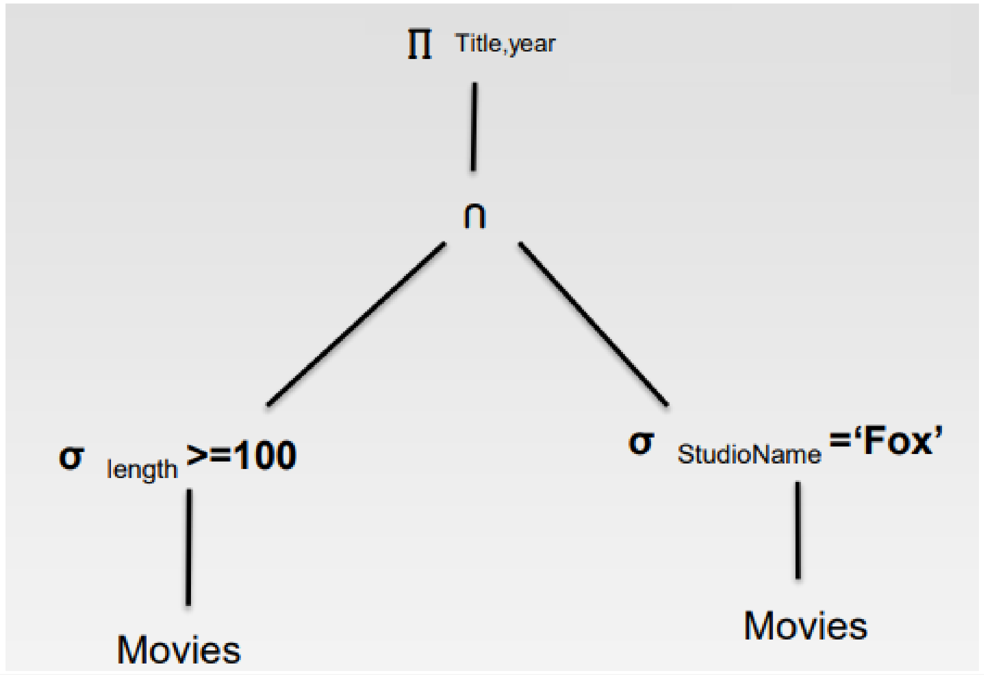
$\sigma_{salary > 35000 \text{ OR } title = 'PR'}(\text{Emp})$

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.



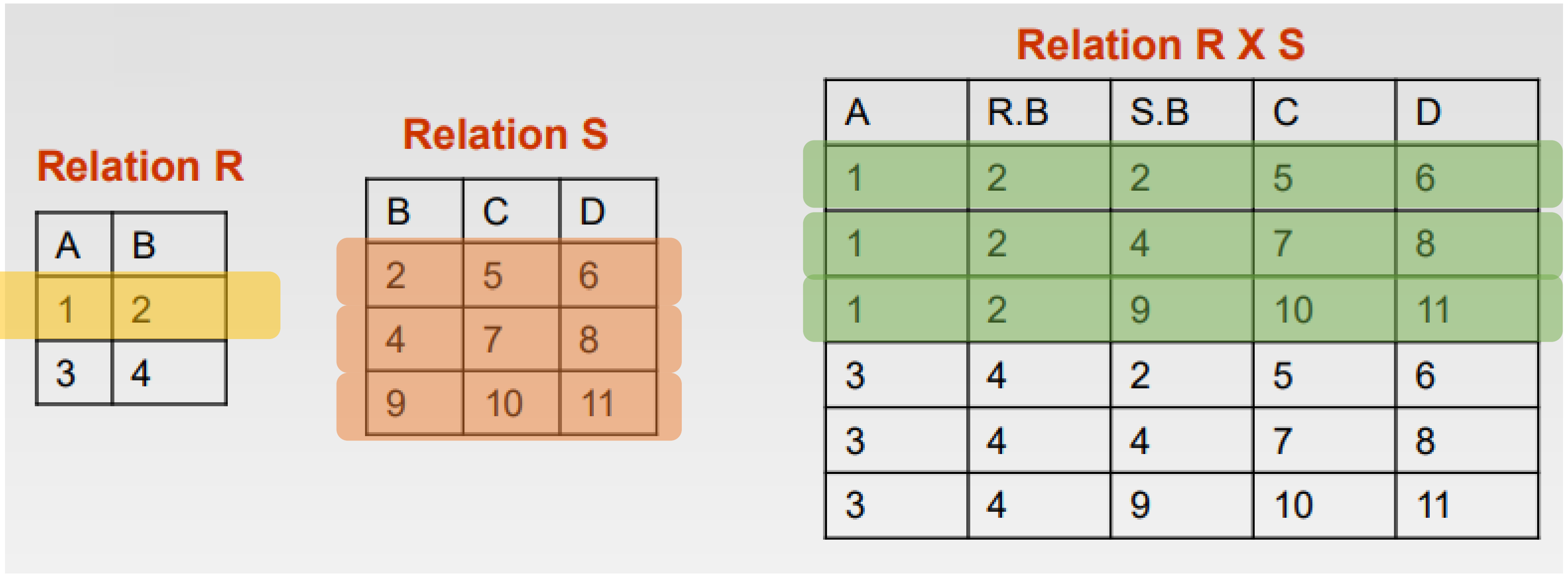
1- select those Movie tuples that have length>=100
2- select those Movies tuples that have studioName='fax'
3- compute the intersection of (1) and (2)
4-project the relation from (3) onto attributes title and year.

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

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

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

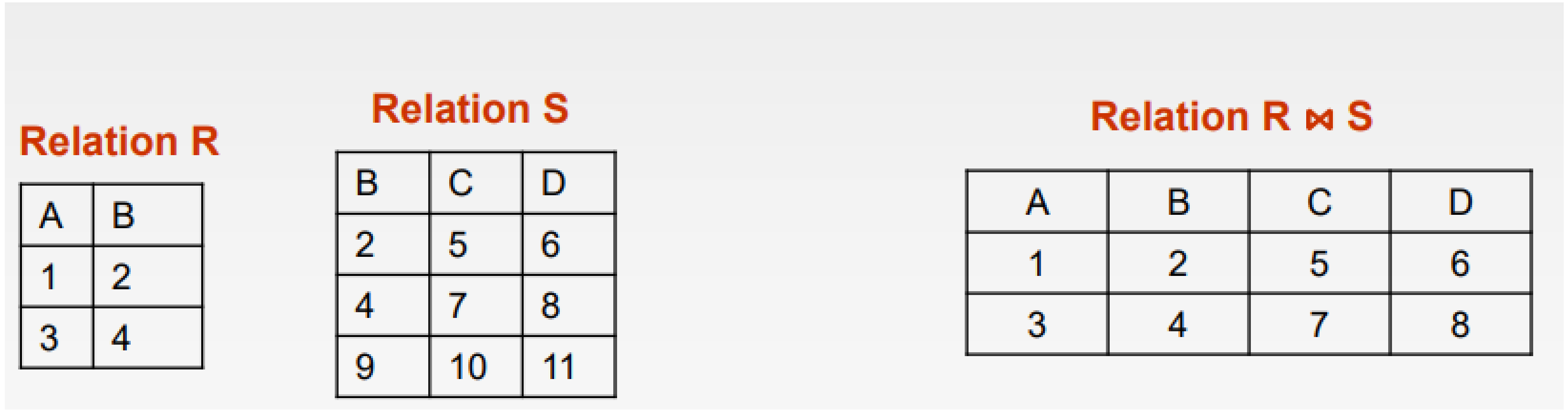
B	C	D
2	5	6
4	7	8
9	10	11

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

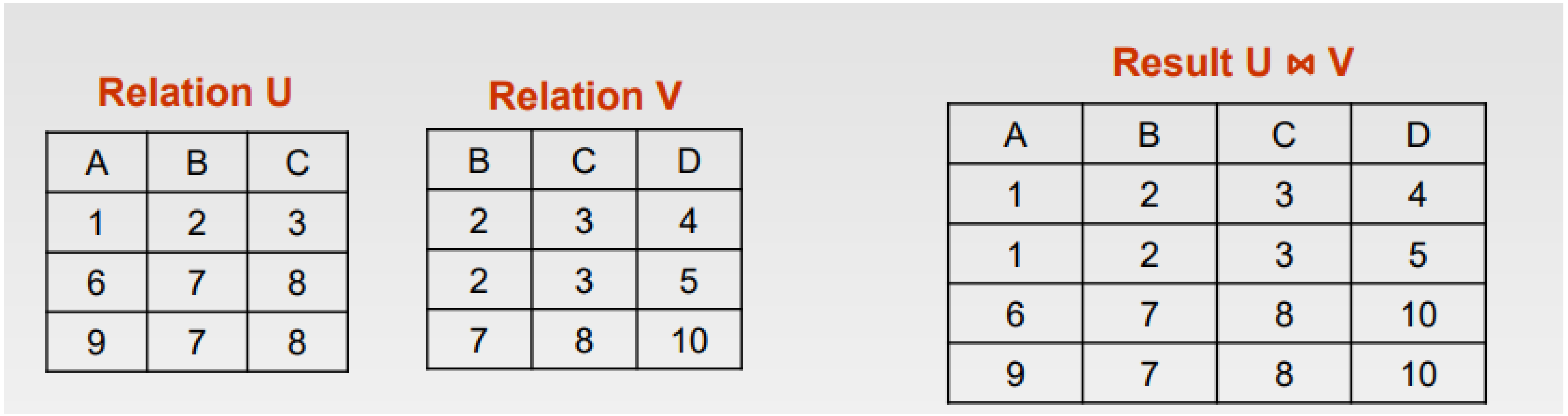
A	B
1	2
3	4

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

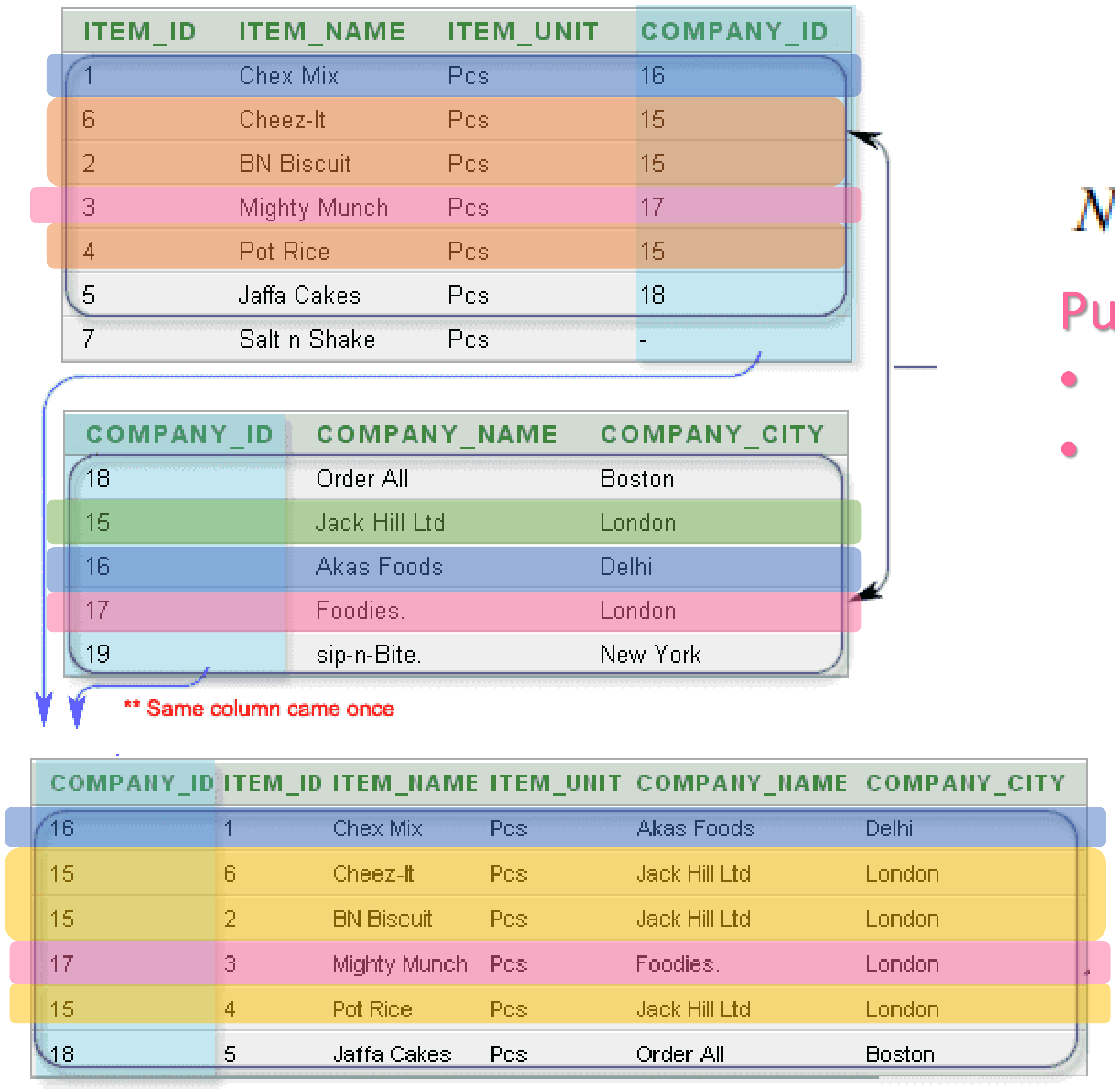
A	B	C
1	2	3
6	7	8
9	7	8

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


** Same column came once

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

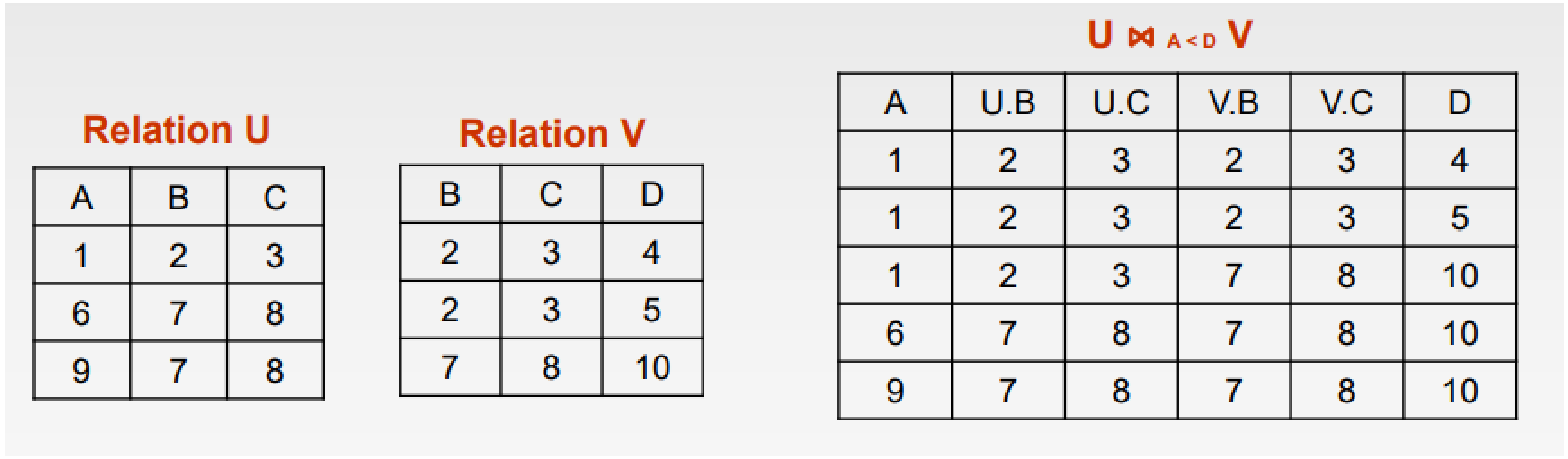
Notation: $R \bowtie S$

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

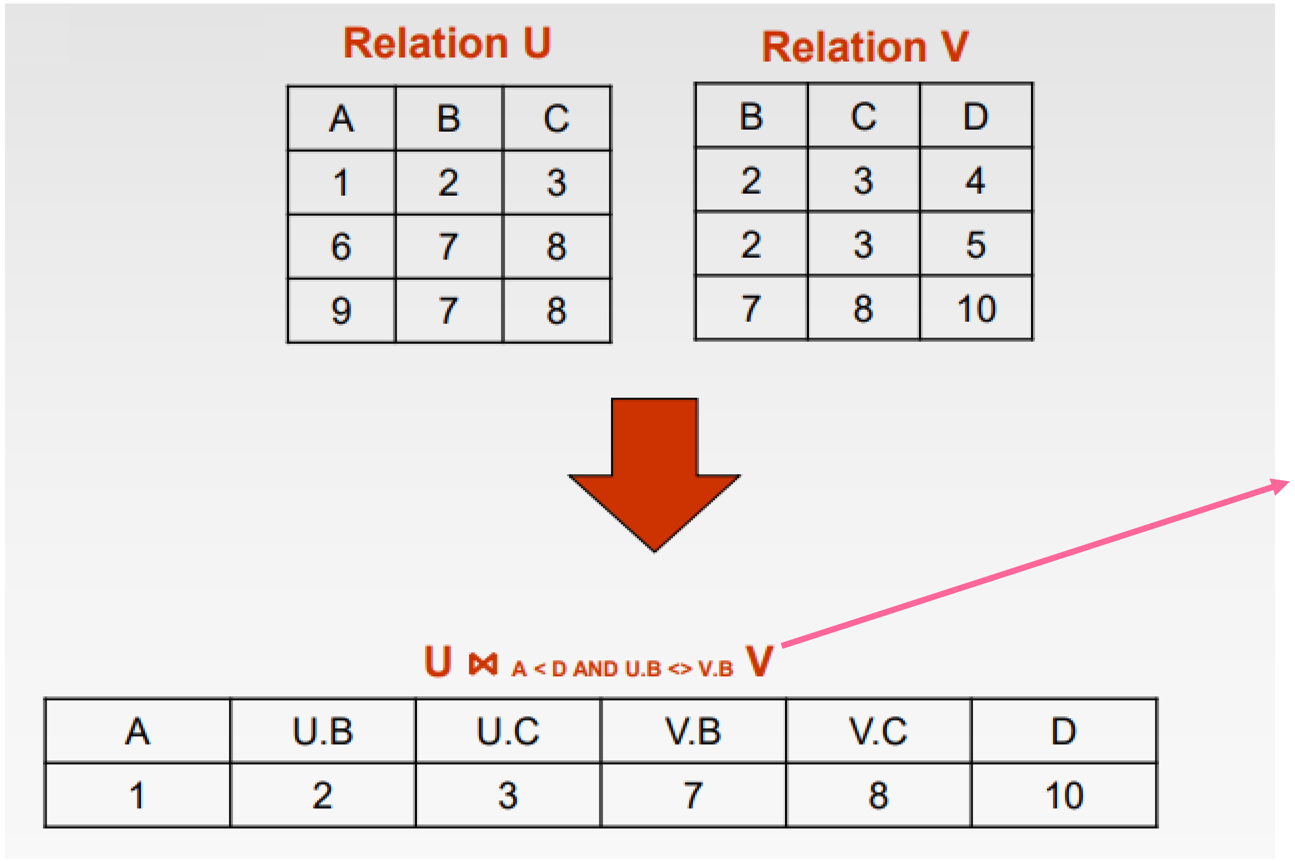
Relation V

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

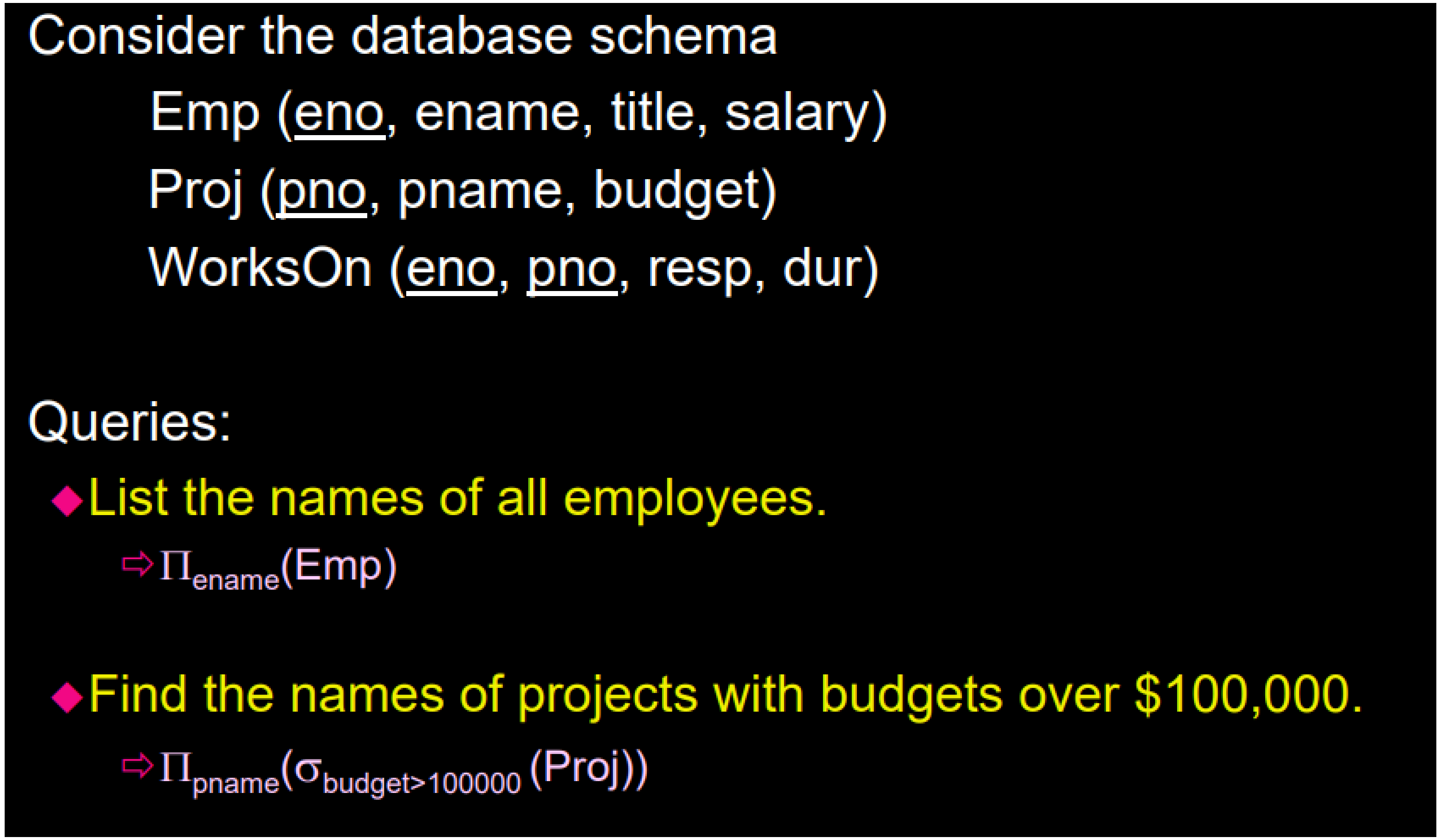
Complete Set of Relational Algebra Operators

- 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:

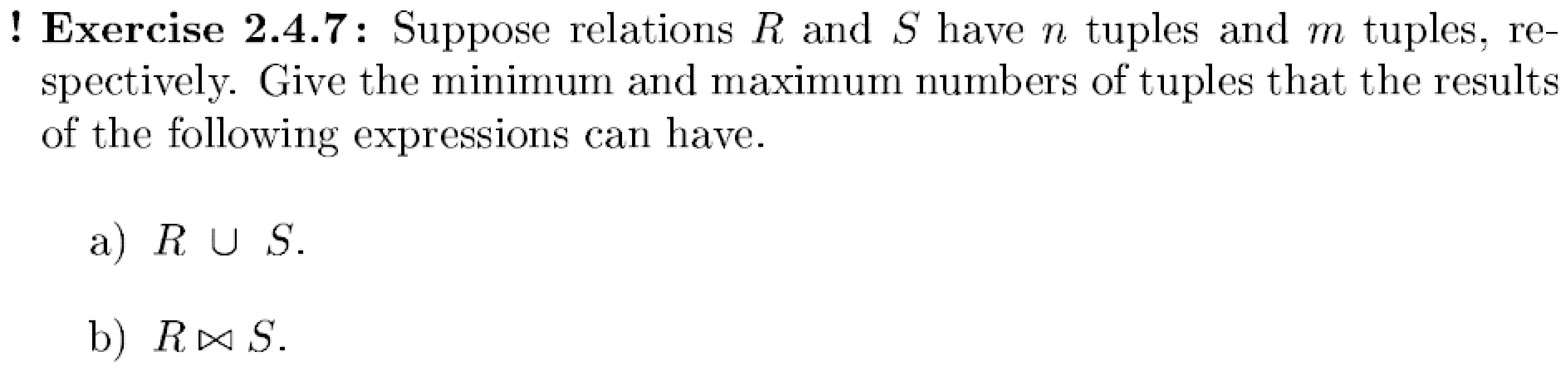
- List the names of all employees.
  - ⇨ $\Pi_{ename}(Emp)$
- Find the names of projects with budgets over $100,000.
  - ⇨ $\Pi_{pname}(\sigma_{budget>100000}(Proj))$

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) $R \bowtie S$.