Relational Database

Ms. Sonia Tomer
A.P.
CSE Department

Relational Query Languages

- Languages for describing queries on a relational database
- *Structured Query Language* (SQL)
  - Predominant application-level query language
  - Declarative
- *Relational Algebra*
  - Intermediate language used within DBMS
  - Procedural

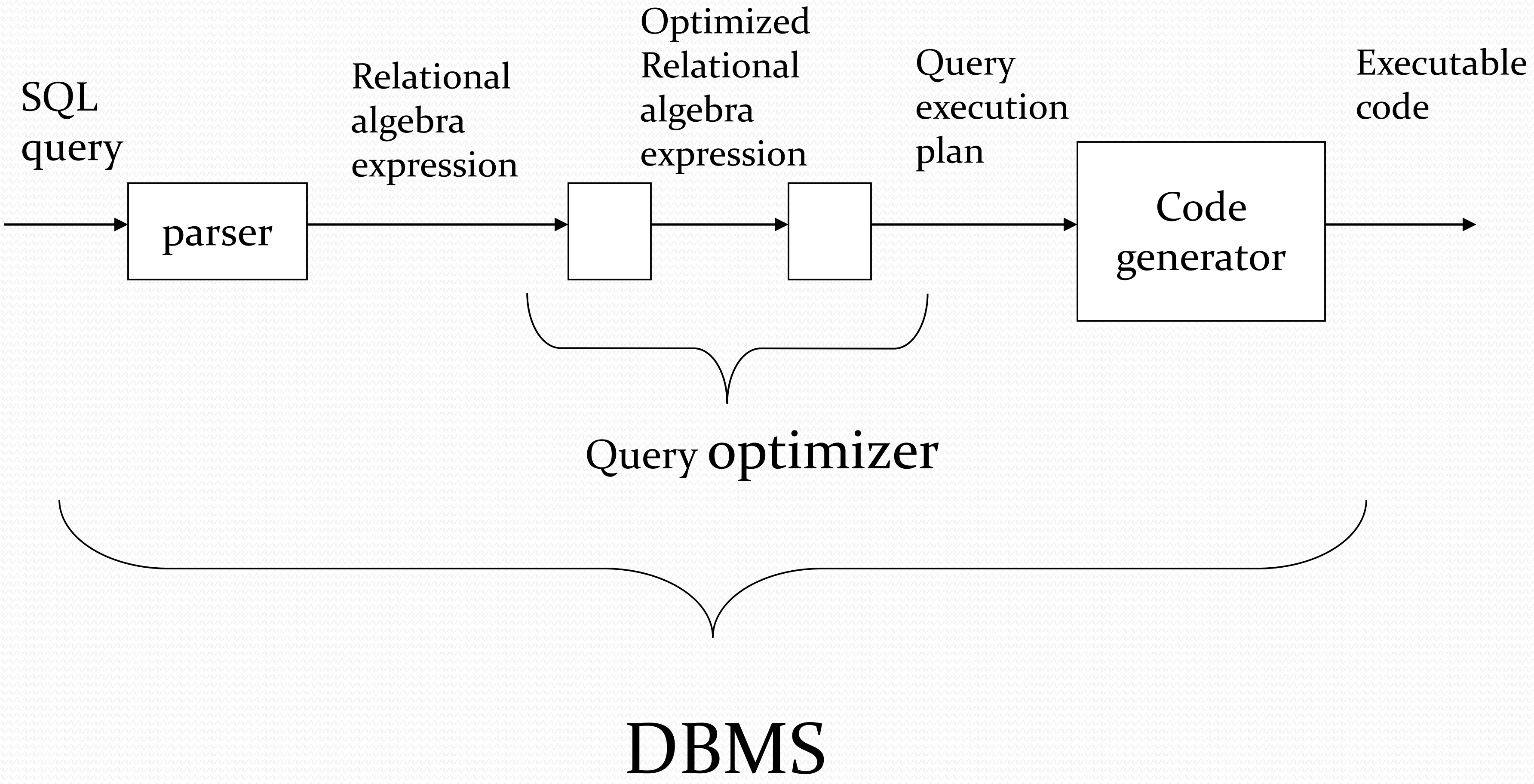
What is an Algebra?

- A language based on operators and a domain of values
- Operators map values taken from the domain into other domain values
- Hence, an expression involving operators and arguments produces a value in the domain
- When the domain is a set of all relations (and the operators are as described later), we get the *relational algebra*
- We refer to the expression as a *query* and the value produced as the *query result*

Relational Algebra

- *Domain*: set of relations
- *Basic operators*: select, project, union, set difference, Cartesian product
- *Derived operators*: set intersection, division, join
- *Procedural*: Relational expression specifies query by describing an algorithm (the sequence in which operators are applied) for determining the result of an expression

Relational Algebra in a DBMS



SQL query

parser

Relational algebra expression

Optimized Relational algebra expression

Query execution plan

Code generator

Executable code

Query optimizer

DBMS

1. Select Operator

- Produce table containing subset of rows of argument table satisfying condition

$$\sigma_{condition}\ relation$$

- Example:

Person

Id	*Name*	*Address*	*Hobby*
1123	John	123 Main	stamps
1123	John	123 Main	coins
5556	Mary	7 Lake Dr	hiking
9876	Bart	5 Pine St	stamps

$\sigma_{Hobby=\text{'stamps'}}(\text{Person})$

Id	*Name*	*Address*	*Hobby*
1123	John	123 Main	stamps
9876	Bart	5 Pine St	stamps

Selection Condition

- Operators: <, ≤, ≥, >, =, ≠
- Simple selection condition:
  - *<attribute> operator <constant>*
  - *<attribute> operator <attribute>*
- <*condition*> AND <*condition*>
- <*condition*> OR <*condition*>
- NOT <*condition*>

Selection Condition - Examples

- $\sigma_{Id>3000 \text{ Or } Hobby=\text{‘}hiking\text{’}}$ (Person)
- $\sigma_{Id>3000 \text{ AND } Id<3999}$ (Person)
- $\sigma_{\text{NOT}(Hobby=\text{‘}hiking\text{’})}$ (Person)
- $\sigma_{Hobby\neq\text{‘}hiking\text{’}}$ (Person)

2. Project Operator

- Produces table containing subset of columns of argument table

$$\Pi_{\textit{attribute list}}(\textit{relation})$$

- Example:

Person

Id	Name	Address	Hobby
1123	John	123 Main	stamps
1123	John	123 Main	coins
5556	Mary	7 Lake Dr	hiking
9876	Bart	5 Pine St	stamps

$\Pi_{Name,Hobby}(\text{Person})$

Name	Hobby
John	stamps
John	coins
Mary	hiking
Bart	stamps

Project Operator

- Example:

Person

$\Pi_{Name,Address}(Person)$

Id	*Name*	*Address*	*Hobby*
1123	John	123 Main	stamps
1123	John	123 Main	coins
5556	Mary	7 Lake Dr	hiking
9876	Bart	5 Pine St	stamps

Name	*Address*
John	123 Main
Mary	7 Lake Dr
Bart	5 Pine St

Result is a table (no duplicates)

Expressions

$$\Pi_{\mathit{Id,\ Name}}\,(\sigma_{\mathit{Hobby='stamps'}\ \mathrm{OR}\ \mathit{Hobby='coins'}}\ (\mathrm{Person})\ )$$

Id	*Name*	*Address*	*Hobby*
1123	John	123 Main	stamps
1123	John	123 Main	coins
5556	Mary	7 Lake Dr	hiking
9876	Bart	5 Pine St	stamps

Person

Id	*Name*
1123	John
9876	Bart

Result

3. Cartesian Product

- Each tuple in R1 with each tuple in R2
- Notation: R1 × R2
- Example:
  - Employee × Dependents
- Very rare in practice; mainly used to express joins

Cartesian Product Example

Employee

Name	SSN
John	999999999
Tony	777777777

Dependents

EmployeeSSN	Dname
999999999	Emily
777777777	Joe

Employee x Dependents

Name	SSN	EmployeeSSN	Dname
John	999999999	999999999	Emily
John	999999999	777777777	Joe
Tony	777777777	999999999	Emily
Tony	777777777	777777777	Joe

4. OPERATORS FROM SET THEORY

UNION

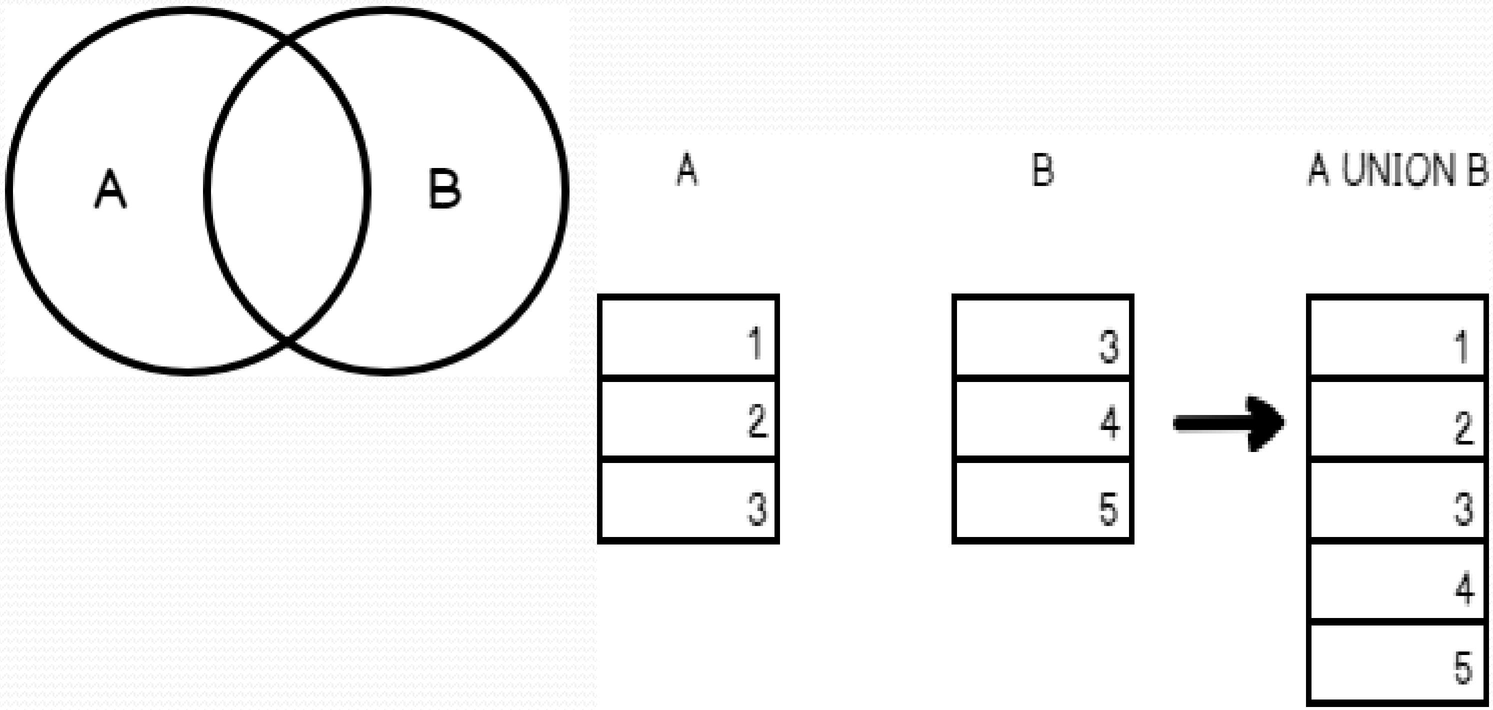
The UNION operator is used to combine the result-set of two or more SELECT statements.

- *R U S*

➢Each SELECT statement within UNION must have the same number of columns

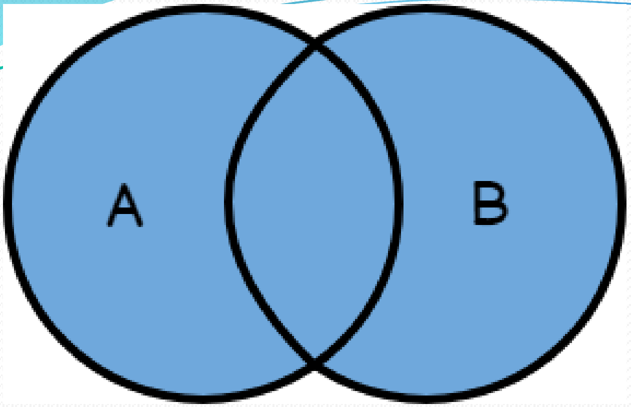
➢The columns must also have similar data types

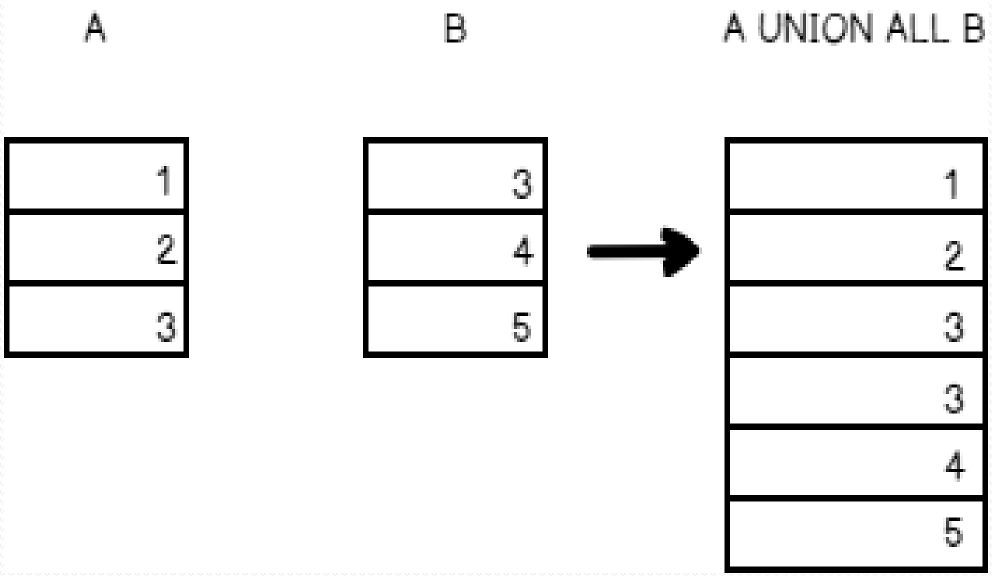
➢The columns in each SELECT statement must also be in the same order



```
SELECT column_name(s) FROM table 1
UNION
SELECT column_name(s) FROM table 2;
```

Note: If some customers or suppliers have the same city, each city will only be listed once, because UNION selects only distinct values. Use UNION ALL to also select duplicate values!





A	B	A UNION ALL B
1	3	1
2	4	2
3	5	3
		3
		4
		5

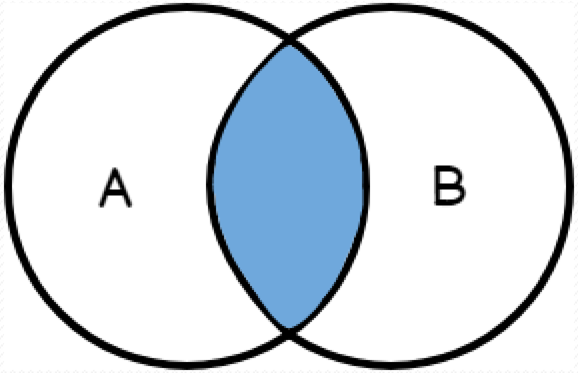
```
SELECT City FROM Customers
UNION ALL
SELECT City FROM Suppliers
ORDER BY City;
```

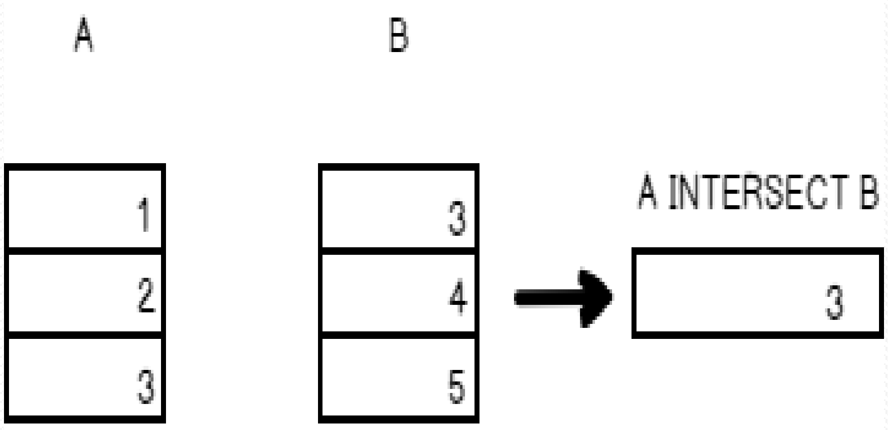
INTERSECTION

The interest operator keeps the rows that are common to all the queries

- *R* ∩ *S*

Includes all tuples that are in both *R and S*





A
1
2
3

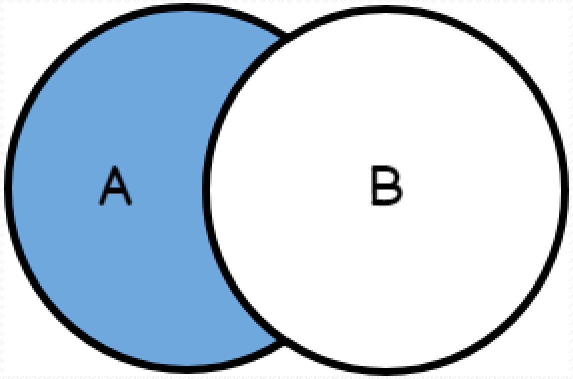
B
3
4
5

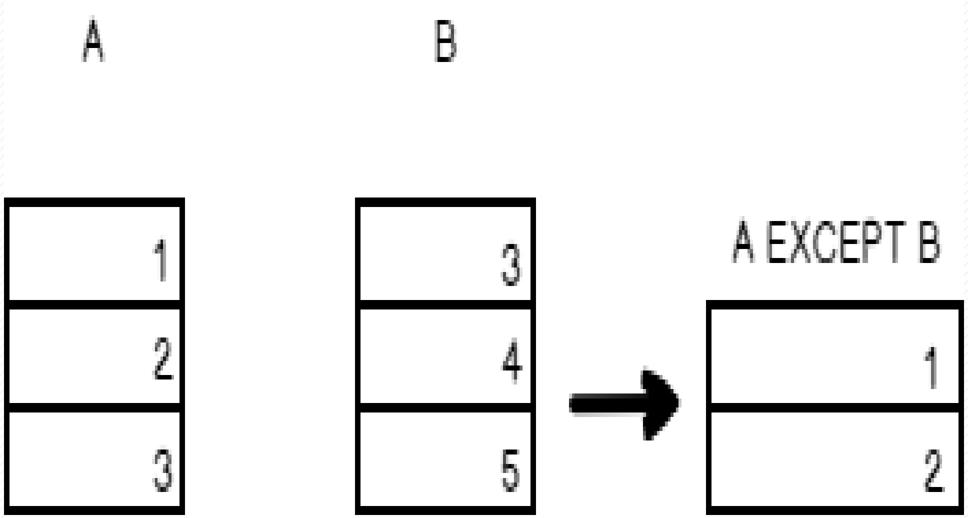
A INTERSECT B
3

DIFFERENCE (or MINUS)

The EXCEPT operator lists the rows in the first that are not in the second.

- *R – S*
- Includes all tuples that are in *R but not in S*





A
1
2
3

B
3
4
5

A EXCEPT B
1
2

5. RENAMING RELATIONS & ATTRIBUTES

- Unary RENAME operator
- Rename relation

$$\rho_S R$$

- Rename attributes

$$\rho_{(B1,B2,...Bn)} R$$

- Rename relation and its attributes

$$\rho_{S(B1,B2,...,Bn)} R$$