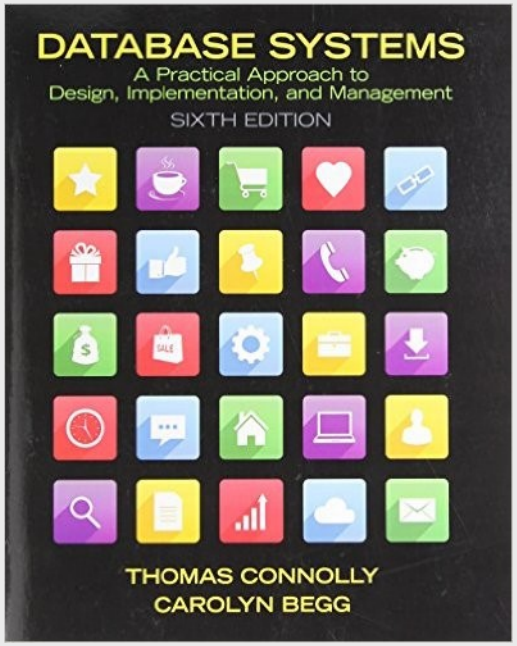
COMP 224 Database Management Systems

Dr. Xu Yang
Room. A323, Chi Un Building

Text book

Thomas Connolly, Carolyn Begg (2015). *DATABASE SYSTEMS: A Practical Approach to Design, Implementation and Management* (6th ed.). Addison Wesley.



Assessment

Assignments	
– One project	15%
– One writing assignment	15%
– In-class exercise	5%
One test (knowledge assessment)	15%
Final examination	50%

COMP 224 Overview

The subject is aimed at introducing students to advanced topics in the design and management of database systems.

- Relational algebra
- Query processing and optimization
- Data Definition Language – Oracle Project (15%)
- Transaction management – Test (15%)
- Distributed database systems– Assignment 2 (15%)
- Big data
- Database Security

How “*Big Data*” will change your life….

“We swim in a sea of data … and the sea level is rising rapidly.”

Data-driven World in the future !

Relational Algebra

Content

- Introduction of relational algebra；
- Five fundamental operations in relational algebra: Selection, Projection, Cartesian product, Union, and Set Difference;
- Join, Intersection, and Division operations which can be expressed in terms of the five basic operations；
- Relational algebra exercises.

Objective:

- How to form queries in relational algebra

Introduction

- Relational algebra is formal language associated with the relational model.

- Relational algebra is used as the basis for other, higher-level Data Manipulation Languages (DMLs) for relational databases.

- It illustrates the basic operations required of any DML.

- It can be used to tell the DBMS how to build a new relation from one or more relations in the database.

Relational Algebra

- Relational algebra operations work on one or more relations to define another relation without changing the original relations.

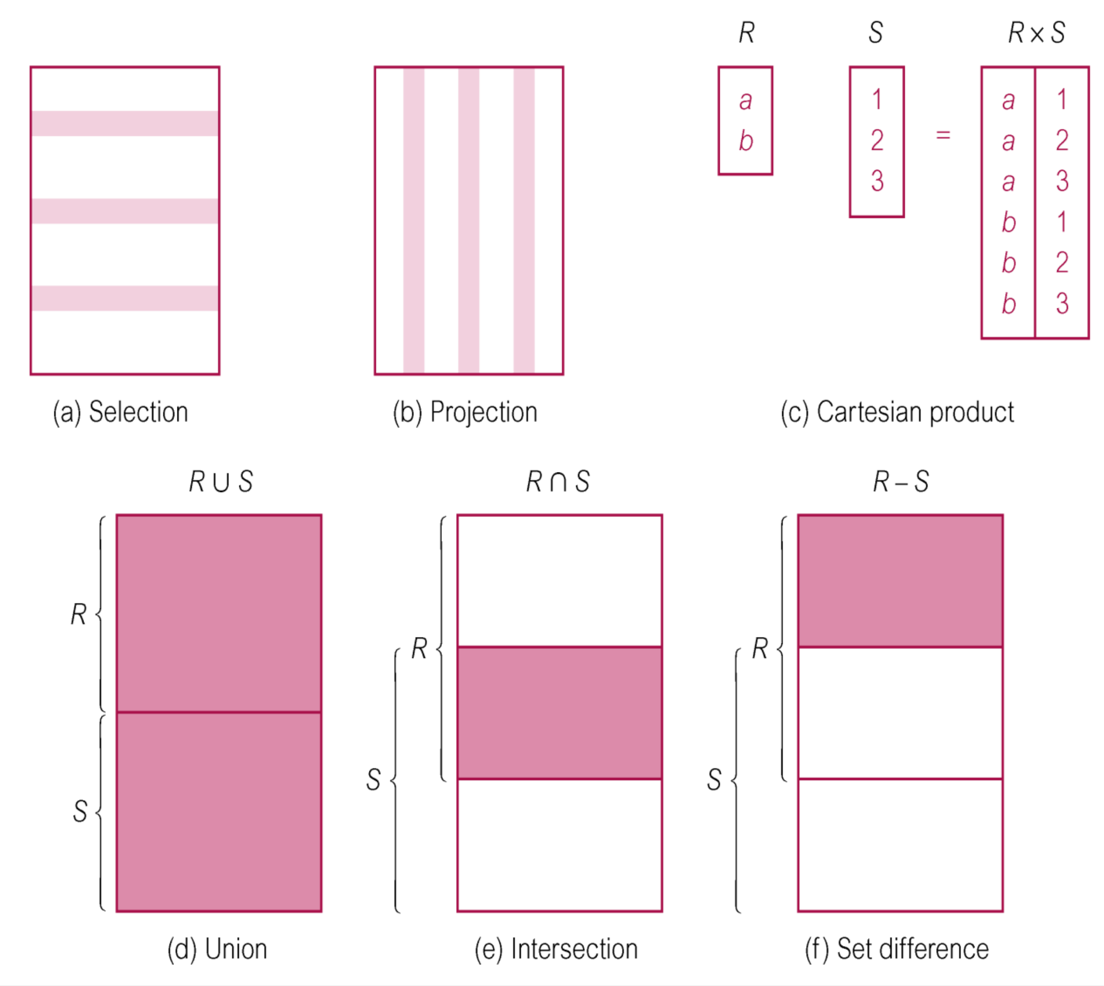
- Both operands and result are relations, so output from one operation can become input to another operation.

- Allows expressions to be nested, just as in arithmetic. This property is called <u>closure</u>.

Relational Algebra

- Five basic operations in relational algebra:
  - Selection
  - Projection
  - Cartesian product
  - Union
  - Set Difference
- These perform most of the data retrieval operations needed.
- Also have Join, Intersection, and Division operations, which can be expressed in terms of 5 basic operations.

Relational Algebra Operations



Selection and **Projection** are unary operations.

The other operations are binary operations.

Selection (or Restriction) σ

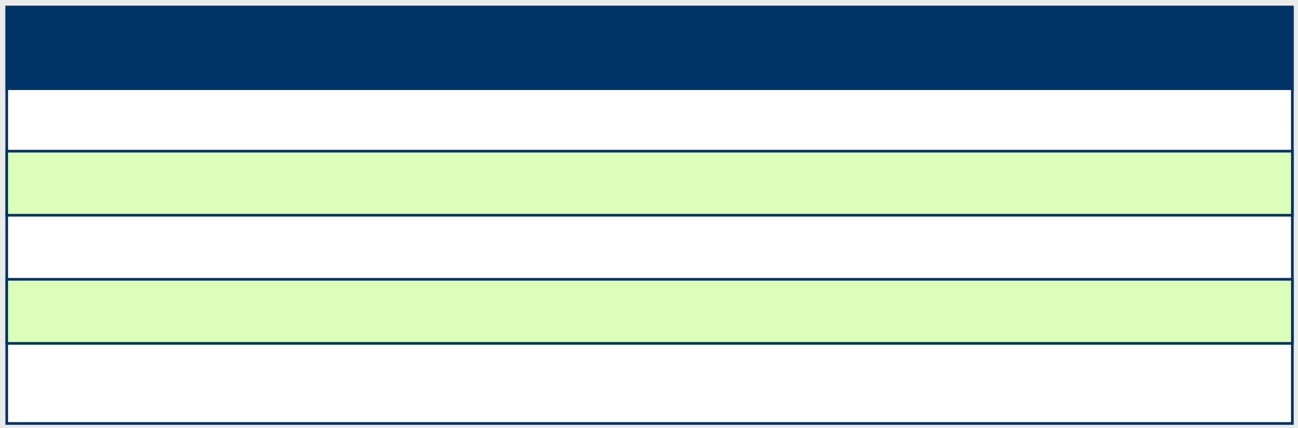
- $\sigma_{predicate}$ **(R)** [*Greek letter called **sigma***]
  - **Works on a single relation R and defines a relation that contains only those tuples (rows) of R that satisfy the specified condition (*predicate*).**

Example of selection:

$\sigma_{branch\text{-}name=\text{'Macau'}}(account)$

Selection

A



The restriction of relation A on the condition:

```
A WHERE condition
```

is a relation with the same heading as A and with a body consisting of the set of all tuples t of A such that the condition is true for t.

Selection Condition (predicate)

may have Boolean conditions **AND** （∧）, **OR** （ ∨ ）, and **NOT** （ ~ ）

- has clauses of the form:
  - » <attribute name> <comparison op> <constant value>

    or
  - » <attribute name> <comparison op> <attribute name>

$\sigma_{(Dno=4 \text{ AND } Salary>2500) \text{ OR } (Dno=5 \text{ AND } Salary>30000)}(\text{EMPLOYEE})$

Relational model is set-based (no duplicate tuples), Relation R has no duplicates, therefore selection cannot produce duplicates.

Selection

Do not confuse this with SQL's `SELECT` statement!

Correspondence

- Relational algebra

  $\sigma_{<selection\ condition>}(R)$

- SQL

```
SELECT *
FROM R
WHERE <selection condition>
```

Example - Selection (or Restriction) σ

List all staff information with a salary greater than £10,000.

$$\sigma_{salary > 10000}\ (Staff)$$

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24- Mar-58	18000	B003
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003

Example - Selection (or Restriction)

- Relation *r*

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

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

A	*B*	*C*	*D*
α	α	*1*	*7*
β	β	*23*	*10*

What is the equivalent relational algebra expression?

Employee

ID	Name	S	Dept	JobType
12	Chen	F	CS	Faculty
13	Wang	M	MATH	Secretary
14	Lin	F	CS	Technician
15	Liu	M	ECE	Faculty

```
SELECT  *
FROM  Employee
WHERE JobType = 'Faculty';
```

Projection Operation

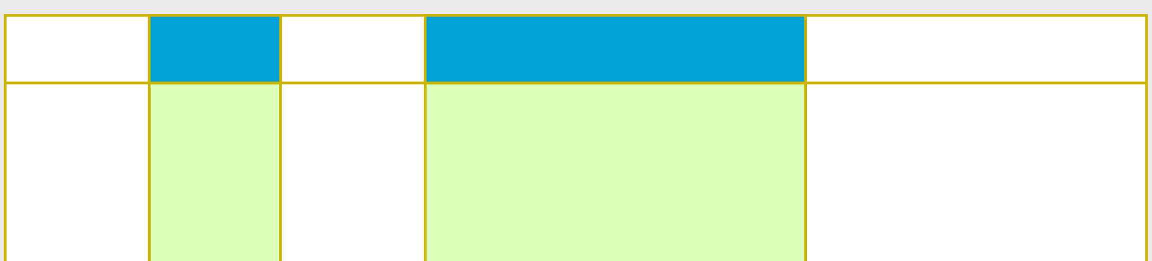
- $\Pi_{\mathbf{col1}, \ldots, \mathbf{coln}}(\mathbf{R})$ [*Greek letter called* ***pi***]

  where *col1, col2* are attribute names and *R* is a relation name.

  - **Works on a single relation R and defines a relation that contains a vertical subset of R, extracting the values of specified attributes and eliminating duplicates.**

- The projection operator produces the output table by
  - eliminating all attributes not specified, and
  - eliminating any duplicated tuples

R



Example - Projection

- Produce a list of salaries for all staff, showing only staffNo, fName, lName, and salary details.

$$\Pi_{\text{staffNo, fName, lName, salary}}(\text{Staff})$$

staffNo	fName	lName	salary
SL21	John	White	30000
SG37	Ann	Beech	12000
SG14	David	Ford	18000
SA9	Mary	Howe	9000
SG5	Susan	Brand	24000
SL41	Julie	Lee	9000

Example - Projection

Airports

airport_name	country_name
Naples	Italy
London/Gk	England
London/Hw	England
Pisa	Italy
Venice/MP	Italy
Manchester	England
Kai Tak	Hong Kong
Changi	Singapore

$$\Pi_{country\_name}(Airports)$$

country_name
Italy
England
Hong Kong
Singapore

Projection

Correspondence

- Relational algebra

  $\pi_{<attribute\ list>}(R)$

- SQL

```
SELECT DISTINCT <attribute list>
FROM R
```

- Note the need for `DISTINCT` in SQL

What is the equivalent relational algebra expression?

Employee

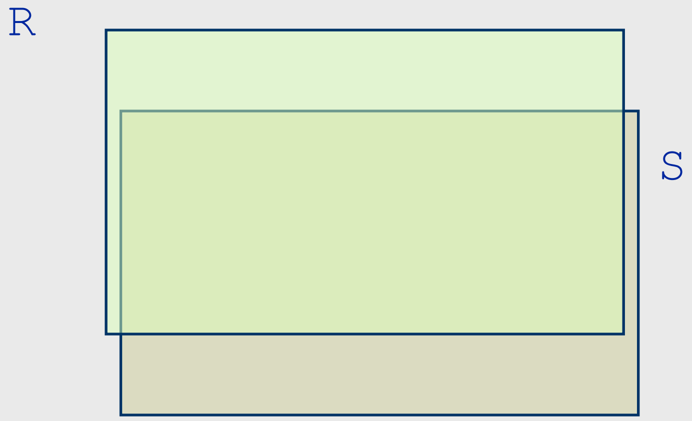
ID	Name	S	Dept	JobType
12	Chen	F	CS	Faculty
13	Wang	M	MATH	Secretary
14	Lin	F	CS	Technician
15	Liu	M	ECE	Faculty

```
SELECT  DISTINCT Name, S, Dept
FROM Employee;
```

Union Operation

- R ∪ S
  - Union of two relations R and S defines a relation that contains all the tuples of R, or S, or both R and S, **duplicate tuples being eliminated**.
  - R and S must be **union-compatible**.
- If R and S have *I* and *J* tuples, respectively, union is obtained by concatenating them into one relation with a **maximum of (*I* + *J*) tuples**.

Two relations are union-compatible if they have the *same set of attributes* which are defined *on the same domains*.



Example - Union

Produce a list of all staff that work in either of two departments *(each department has a separate database)*, showing only their staffNo, and date of birth.

$$\Pi_{staffNo,\ dob}(Staff\_DepA) \cup \Pi_{staffNo,\ dob}(Staff\_DepB)$$

Staff_DepA

staffNo	dob
SL10	14-02-64
SA51	21-11-82
DS40	01-01-40

$\cup$

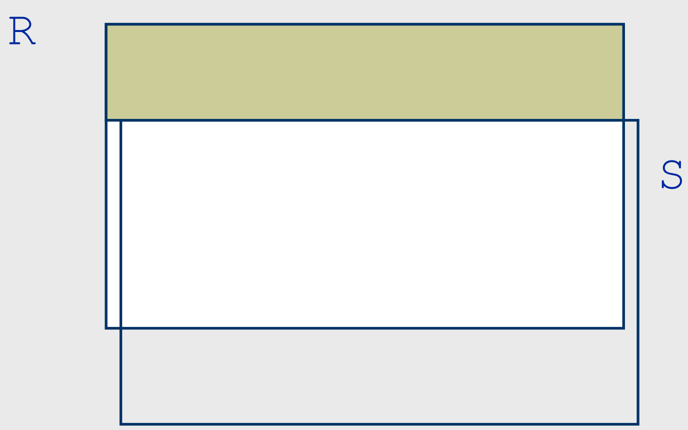
Staff_DepB

staffNo	dob
CC15	11-03-66
SA51	21-11-82

staffNo	dob
SL10	14-02-64
SA51	21-11-82
DS40	01-01-40
CC15	11-03-66

Set Difference

- **R – S**
  - **Defines a relation consisting of the tuples that are in relation R, but not in S.**
  - **R and S must be union-compatible.**



Example - Difference

Produce a list of all staff that only work in department A *(each department has a separate database)*, showing only their staffNo, and date of birth.

$$\Pi_{staffNo,\ dob}(Staff\_DepA) - \Pi_{staffNo,\ dob}(Staff\_DepB)$$

Staff_DepA

staffNo	dob
SL10	14-02-64
SA51	*21-11-82*
DS40	01-01-40

—

Staff_DepB

staffNo	dob
CC15	11-03-66
SA51	*21-11-82*

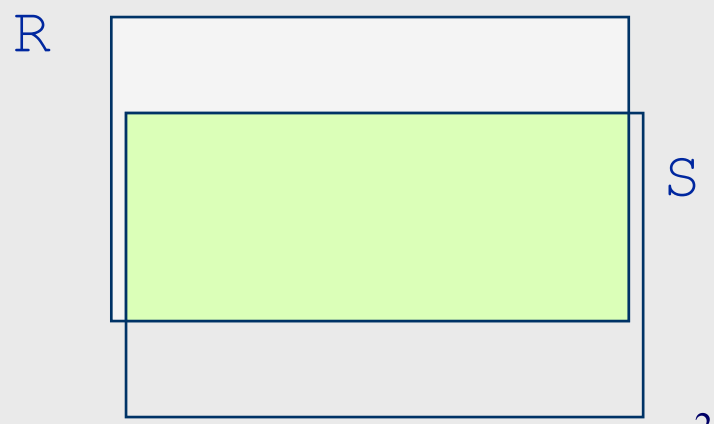
staffNo	dob
SL10	14-02-64
DS40	01-01-40

Intersection

- $R \cap S$
  - Defines a relation consisting of the set of all tuples that are **in both R and S**.
  - R and S must be **union-compatible**.

- Expressed using basic operations:

$$R \cap S = R - (R - S)$$



Example - Intersection

Produce a list of staff that work in both department A and department B, showing only their staffNo, and date of birth.

$$(\Pi_{staffNo,\ dob}(Staff\_DepA)) \cap (\Pi_{staffNo,\ dob}(Staff\_DepB))$$

Staff_DepA

staffNo	dob
SL10	14-02-64
SA51	*21-11-82*
DS40	01-01-40

$\cap$

Staff_DepB

staffNo	dob
CC15	11-03-66
SA51	*21-11-82*

$\Rightarrow$

staffNo	dob
SA51	21-11-82

Example of Union/Intersection/Difference

A

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S4	Clark	20	London

B

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris

a. *Union*
(A UNION B)

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S4	Clark	20	London
S2	Jones	10	Paris

b. *Intersection*
(A INTERSECT B)

S#	SNAME	STATUS	CITY
S1	Smith	20	London

c. *Difference*
(A MINUS B)

S#	SNAME	STATUS	CITY
S4	Clark	20	London

d. *Difference*
(B MINUS A)

S#	SNAME	STATUS	CITY
S2	Jones	10	Paris

Cartesian product

- **R X S**
  - **Defines a relation that is the concatenation of every tuple of relation R with every tuple of relation S.**
- **The Cartesian product operation multiplies two relations to define another relation consisting of all possible pairs of tuples from the two relation.**

a1
a2

X

b1
b2

=

a1	b1
a1	b2
a2	b1
a2	b2

More about product operation

- If one relation has I tuples, and N attributes; the other has J tuples and M attributes
  - The Cartesian product relation will contain (I*J) tuples with (N+M) attributes.
  - If the two relations have attributes with the same name, the attribute names are prefixed with the relation name to maintain the uniqueness of attributes name within a relation.
- The result is often large and space-costly but produces no more information than there was in the input.

Example - 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

Example - Cartesian product

Combine details of staff (S) and the departments (D).

$\Pi_{\text{staffNo, job, dept}}(S) \times \Pi_{\text{dept, name}}(D)$

S

staffNo	job	dept
SL10	Salesman	10
SA51	Manager	20
DS40	Clerk	20

X

D

dept	name
10	Stratford
20	Barking

staffNo	job	S.dept	D.dept	name
SL10	Salesman	10	10	Stratford
SA51	Manager	20	10	Stratford
DS40	Clerk	20	10	Stratford
SL10	Salesman	10	20	Barking
SA51	Manager	20	20	Barking
DS40	Clerk	20	20	Barking

5 attributes, 2 dept

Relation Algebra Operations

T

A	B
a	1
b	2

U

B	C
1	x
1	y
3	z

$T \bowtie U$

A	B	C
a	1	x
a	1	y

(g) Natural join

$T \rhd_B U$

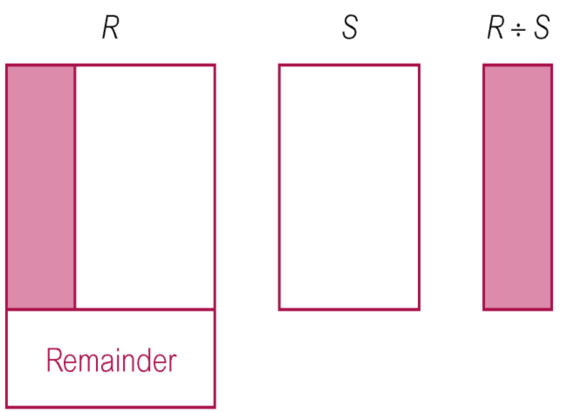
A	B
a	1

(h) Semijoin

$T ⟕_C U$

A	B	C
a	1	x
a	1	y
b	2	

(i) Left Outer join



(j) Division (shaded area)

V

A	B
a	1
a	2
b	1
b	2
c	1

W

B
1
2

$V \div W$

A
a
b

Example of division

Join Operations

- Join is a derivative of Cartesian product.
- Equivalent to performing a Selection, using join predicate as selection formula, over Cartesian product of the two operand relations.
- One of the most difficult operations to implement efficiently in an RDBMS and one reason why RDBMSs have intrinsic performance problems.

Join Operations

- Various forms of join operation
  - Theta join
  - Equijoin (a particular type of Theta join)
  - Natural join
  - Outer join
  - Semijoin

Theta join (θ-join) $\bowtie_{\text{<join condition>}}$

$\mathbf{R} \bowtie_{\text{<join condition>}} \mathbf{S}$

- Defines a relation that contains tuples satisfying the predicate F from the Cartesian product of R and S.
- The predicate F is of the form $R.a_i\ \theta\ S.b_i$ where θ may be one of the comparison operators ($<$, $\leq$, $>$, $\geq$, $=$, $\neq$).
- Can rewrite Theta join using basic Selection and Cartesian product operations.

$$\mathbf{R \bowtie_F S = \sigma_F(R \times S)}$$

- Degree of a Theta join is sum of degrees of the operand relations R and S. If predicate F contains only equality (=), the term *Equijoin* is used.

Theta join (θ-join)

Produce a list of staff and the departments they work in.

$$(\Pi_{\text{staffNo, job, dept}}(\text{Staff})) \bowtie_{\text{Staff.dept} = \text{Dept.dept}} (\Pi_{\text{dept, name}}(\text{Dept}))$$

Staff

staffNo	job	dept
SL10	Salesman	10
SA51	Manager	20
DS40	Clerk	20

$\bowtie$

Dept

dept	name
10	Stratford
20	Barking

⇒

staffNo	job	S.dept	D.dept	name
SL10	Salesman	10	10	Stratford
SA51	Manager	20	20	Barking
DS40	Clerk	20	20	Barking

Because the predicate operator is an '=' this is known as an ***Equijoin***

Natural join $\bowtie$

- $R \bowtie S$
  - An Equijoin of the two relations R and S **over all common attributes** *x*. One occurrence of each common attribute **is eliminated** from the result.

Produce a list of staff and the departments they work in.

$(\Pi_{staffNo,\ job,\ dept}(Staff)) \bowtie (\Pi_{dept,\ name}(Dept))$

Staff

staffNo	job	dept
SL10	Salesman	10
SA51	Manager	20
DS40	Clerk	20

$\bowtie$

Dept

dept	name
10	Stratford
20	Barking

➡

staffNo	job	dept	name
SL10	Salesman	10	Stratford
SA51	Manager	20	Barking
DS40	Clerk	20	Barking

Example- Natural Join

Airports

airport_name	country_name
Naples	Italy
London/Gk	England
London/Hw	England
Pisa	Italy
Venice/MP	Italy
Manchester	England
Kai Tak	Hong Kong
Changi	Singapore

Stop

flight_number	airport_name	stop_number
BA383	Kai Tak	1
BA019	Changi	2

Stops (natural) JOIN Airports

flight_number	airport_name	stop_number	country_name
BA383	Kai Tak	1	Hong Kong
BA019	Changi	2	Singapore

Example- Natural Join

Relations r, s:

A	*B*	*C*	*D*
m	*1*	*s*	a
n	*2*	*t*	a
o	*4*	*t*	b
p	*1*	*u*	a
q	*2*	*v*	b

r

B	*D*	*E*
1	a	*f*
3	a	*f*
1	a	*g*
2	b	*h*
3	b	*h*

s

$r \bowtie s$

A	*B*	*C*	*D*	*E*
m	*1*	*s*	a	*f*
m	*1*	*s*	a	*g*
p	*1*	*u*	a	*f*
p	*1*	*u*	a	*g*
q	*2*	*v*	b	*h*

Only tuples that have r.B=s.B and r.D=s.D can stay in the results relation.

Outer join

- To display rows in the result that do not have matching values in the join column, use **Outer join.**

- **R ⟕ S**
  - **(Left) outer join is join in which tuples from R that do not have matching values in common columns of S are also included in result relation.**

Outer Join

- An extension of the join operation that avoids loss of information by other types of join.
- Computes the join and then adds tuples from one relation that does not match tuples in the other relation to the result of the join.
- **Uses *null* values:**
  - *null* signifies that the value is unknown or does not exist
  - All comparisons involving *null* are (roughly speaking) **false** by definition.

Example - Left Outer join

Produce a list of all departments and associated staff that work in them.

$$(\Pi_{\text{dept, name}}(\text{Dept})) ⟕ (\Pi_{\text{staffNo, job, dept}}(\text{Staff}))$$

Dept

dept	name
10	Stratford
20	Barking
30	Watford

⟕

Staff

staffNo	job	dept
SL10	Salesman	10
SA51	Manager	20
DS40	Clerk	20

dept	name	staffNo	job
10	Stratford	SL10	Salesman
20	Barking	SA51	Manager
20	Barking	DS40	Clerk
30	*Watford*	*null*	*null*

Example

- Relation *loan*

loan-number	*branch-name*	*amount*
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

- Relation *borrower*

customer-name	*loan-number*
Jones	L-170
Smith	L-230
Hayes	L-155

Example

- **Natural Join**

  loan ⋈ *Borrower*

loan-number	*branch-name*	*amount*	*customer-name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith

- **Left Outer Join**

  loan ⟕ *borrower*

loan-number	*branch-name*	*amount*	*customer-name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*

Semijoin

$R \rhd_F S$

– **Defines a relation that contains the tuples of R that participate in the join of R with S.**

Can rewrite Semijoin using Projection and Join:

$$R \rhd_F S = \Pi_A(R \bowtie_F S)$$

- **A is the set of all attributes for R**

Example - Semijoin

Employee

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Production

Dept

DeptName	Manager
Sales	Bob
Sales	Thomas
Production	Katie
Production	Mark

$$\textbf{Employee} \rhd_{\textit{Employee.DeptName} = \textit{Dept.DeptName}} \textbf{Dept}$$

Name	EmpId	DeptName
Sally	2241	Sales
Harriet	2202	Production

Division Operation

- **R ÷ S**
  - **Relation R has attribute set A; relation S has attribute set B; and B⊆ A. Let C=A-B.**
  - **Defines a relation over the attributes C that consists of set of tuples from R that match combination of *every* tuple in S.**
  - **Suited to queries that include the phrase "for all".**
  - **Let *r* and *s* be relations on schemas R and S respectively where**
    - » $R = (A_1, \ldots, A_m, B_1, \ldots, B_n)$
    - » $S = (B_1, \ldots, B_n)$

    The result of r ÷ s is a relation on schema

    $R - S = (A_1, \ldots, A_m)$

Example - Division

Show all staff that use all the company's programming languages.

Staff_Prog ÷ Prog

Staff_Prog

staffNo	language
SL10	COBOL
SA51	BASIC
SA51	COBOL
SE14	BASIC
SE18	BASIC

÷

Prog

language
COBOL
BASIC

⇒

staffNo
SA51

Summary

The fundamental intent of the algebra is to allow the writing of the relational expressions.

Selection	$\sigma_{predicate}$ **(R)**
Projection	$\Pi_{col1, \ldots, coln}$**(R)**
Union	**R** $\cup$ **S**
Set difference	**R – S**
Intersection	**R** $\cap$ **S**
Cartesian product	**R X S**

Summary

Theta join $\mathbf{R} \bowtie_{\mathbf{F}} \mathbf{S}$

Equijoin $\mathbf{R} \bowtie_{\mathbf{F}} \mathbf{S}$

Natural join $\mathbf{R} \bowtie \mathbf{S}$

(Left) Outer join $\mathbf{R} ⟕ \mathbf{S}$

Semijoin $\mathbf{R} \rhd_{\mathbf{F}} \mathbf{S}$

Division $\mathbf{R} \div \mathbf{S}$