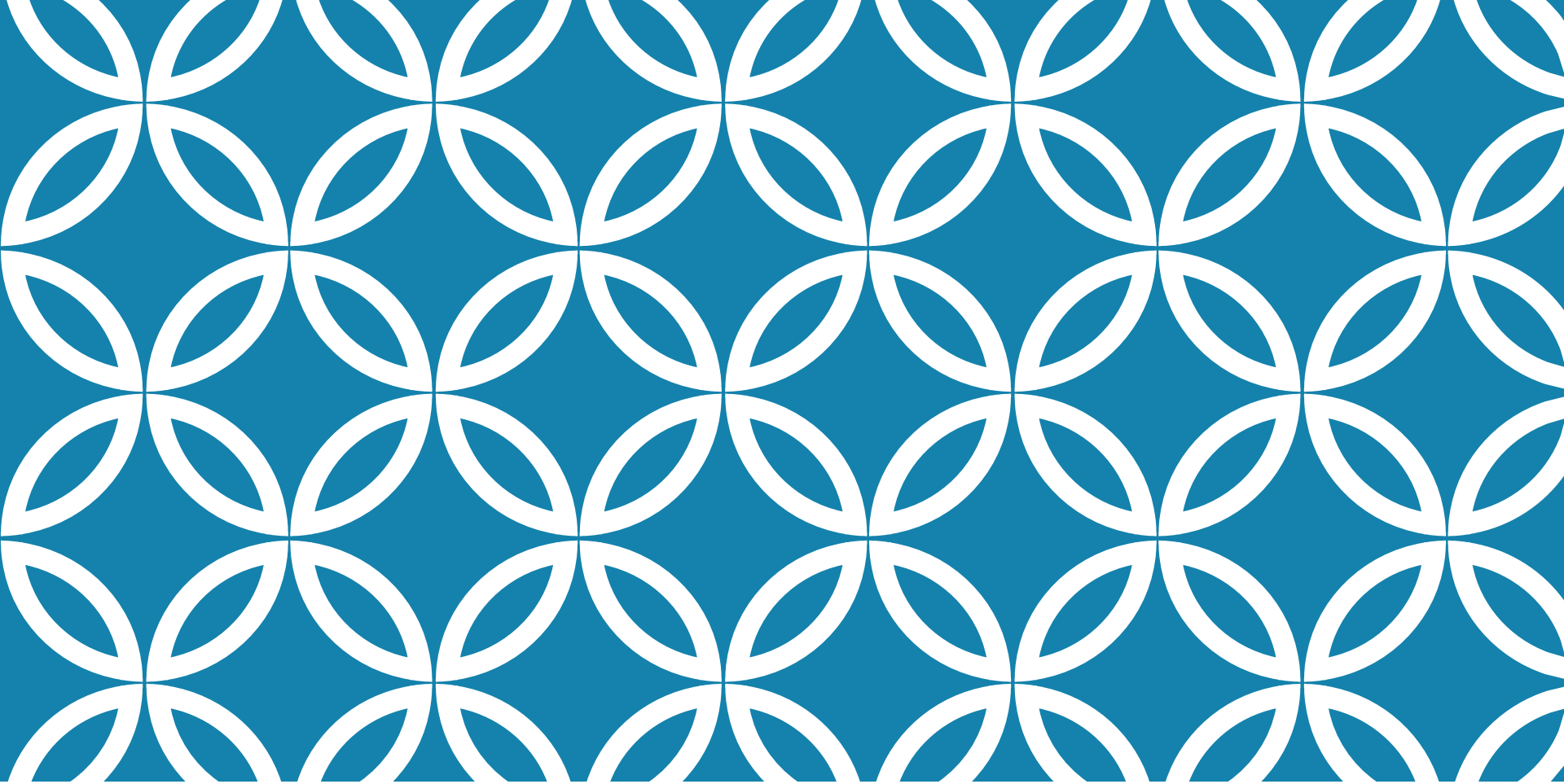


QUERY PROCESSING AND RELATIONAL ALGEBRA

CT230 Database Systems I

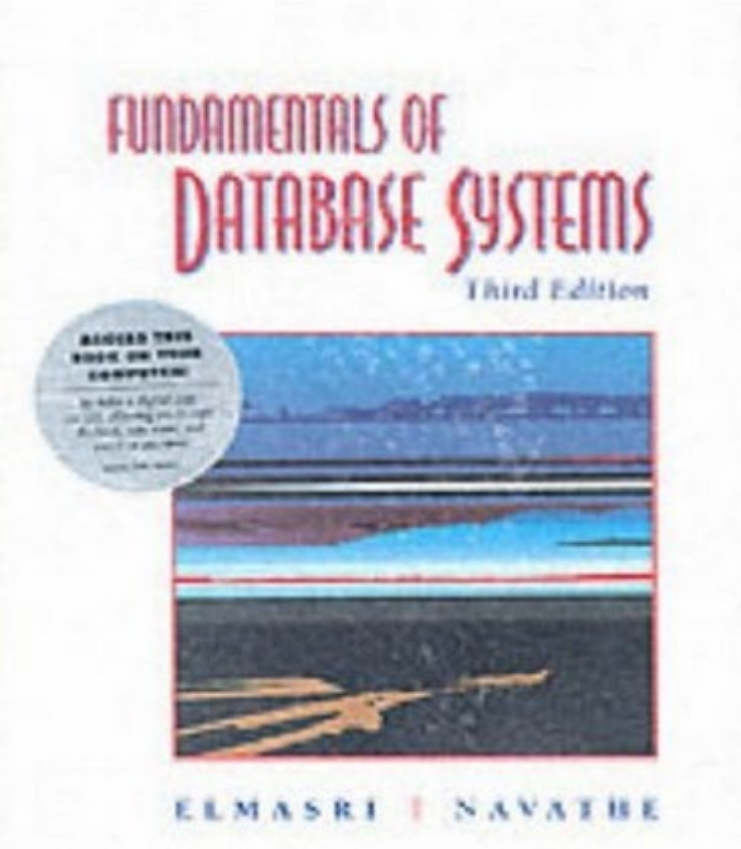
RECOMMENDED TEXT:

See:

Chapter 18

Elmasri & Navathe

(3rd Edition)



DEFINITION: Query Processing

Transforms SQL (high level language) in to a **correct** and **efficient** low level language representation of relational algebra.

Each relational algebra operator has code associated with it (a program) which, when run, performs the operation on the data specified, allowing the specified data to be output as the result.

Steps Involved in Processing a SQL Query:

- Process (Parse and Translate) and create an internal representation of the query – may be an Operator Tree, **Query tree** or Query graph (for more complicated queries).
- Optimise.
- Execute/Evaluate returning results.

How to Translate SQL to Relational Algebra?

Must have:

- a meaningful set of relational algebra operators (today's lecture).
- a mapping (translation) between SQL code and relational algebra expressions.

RELATIONAL ALGEBRA

Two formal languages exist for the relational model:

- Relational algebra (procedural)
- Relational calculus (non-procedural)

Both are logically equivalent

Note: the *practical/implementation* language of the relational model is SQL (as we have seen)

Relational Algebra Operations

π σ ρ ← → τ γ ∧ ∨ ¬ = ≠ ≥ ≤ ∩ ∪ ÷ - × ⋈ ⋈ ⋈ ⋈ ⋉ ⋊ ▷

- A basic set of operations exist for the relational model.
- These allow for the specification of basic retrieval requests.
- A sequence of relational algebra (RA) operations forms a relational algebra expression.
- RA operations are divided into two groups:
  - operations based on mathematical set theory (e.g., union, product etc.)
  - specific relational database operations.

RELATIONAL ALGEBRA versus SQL

The core operations and functions (i.e., programs) in the internal modules of most relational database systems are based on relational algebra.

SQL is a declarative language …. It allows you specify the results you require … not the order of the operations to retrieve those results.

Relational Algebra is *procedural* - must specify exactly *how* to retrieve results when using relational algebra.

RELATIONAL ALGEBRA EXPRESSIONS

- A valid relational algebra expression is built by connecting tables or expressions with defined unary and binary operators and their arguments (if applicable)
- Temporary relations resulting from a relational algebra expression can be used as input to a new relational algebra expression
- Expressions in brackets are evaluated first
- Relational Algebra operators are either Unary or Binary

Relational Algebra: UNARY OPERATORS

- Selection
- Projection
- Rename
- Order
- Group

Each operation:

- takes one relation (table) or expression as input
- gives a new relation as a result

Selection operator

σ (sigma)

Used to select certain tuples (rows) from a relation R

Notation: $\sigma_p R$

where:

p: selection predicate i.e., *a condition*

R: relation/table name

NOTE:

The Selection (σ) operator in relational algebra is NOT the same as the `SELECT` clause in an SQL query.

A SQL `SELECT` query could be equivalent to a combination of relational algebra operators (σ, π and `JOIN`)

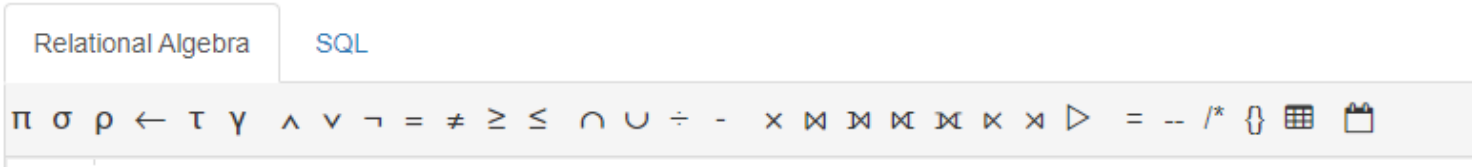
EXAMPLE 1 (using company schema):
Find the projects with pno = 10 and hours worked < 20

$\sigma_{(\texttt{hours < 20 AND pno = 10})}\texttt{works\_on}$

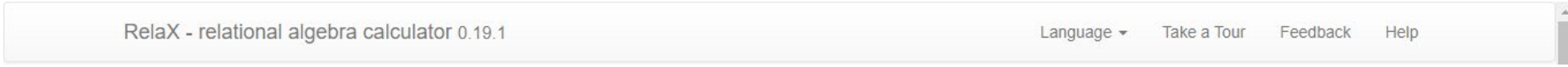
$\texttt{sigma}_{(\texttt{hours < 20 AND pno = 10})}\ \texttt{works\_on}$

Returns the set:

```
{ (333445555, 10, 10.0 ), (999887777, 10, 10.0)}
```



WORKING WITH THE RelaX CALCULATOR



There is no standard language for relational algebra like there is for SQL.

One University group have developed a calculator that supports a *fairly* common standard.

Note that it is CASE SENSITIVE.

Provides a number of datasets with the option of also using your own dataset.

We will load in a version of the COMPANY schema

LOAD A DATASET:

Calculator: https://dbis-uibk.github.io/relax/calc/local/uibk/local/0

Go to “”Group Editor” Tab

Copy text from file on Blackboard and add

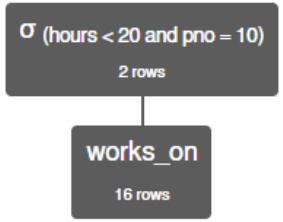
Then choose “Preview”

Then choose “Use group in Editor”

*Note: only stored temporarily

Example 1 in RelaX calculator:

Find the projects with pno = 10 and hours worked < 20



$\sigma_{(hours < 20 \text{ and } pno = 10)}$ works_on

Execution time: 0 ms

works_on.essn	works_on.pno	works_on.hours
333445555	10	10
999887777	10	10

NOTE:

- The **degree** of the relation resulting from a selection of table R is the same as the degree of R, e.g., *same number of attributes/columns*

The operation is **commutative**, i.e. a sequence of selects can be applied in any order,

e.g.

$\sigma_{(\texttt{hours < 20 and pno = 10})}$ works_on

$\sigma_{(\texttt{pno = 10 and hours < 20})}$ works_on

EXAMPLE 2: (Using company database):

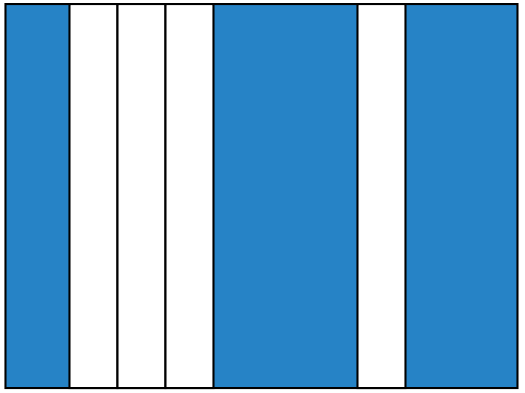
List the department numbers of departments located in Houston

```
σ (dlocation = 'Houston') dept_locations
```

or can write as:

```
sigma (dlocation = 'Houston') dept_locations
```

PROJECTION OPERATOR
π *Pi*



Used to return certain attributes/columns

Notation: $\pi_{A1, A2, \ldots, Ak}(\mathbf{R})$

where:

$A_1 \ldots A_k$ attribute names

R: relation/table name

Result is a relation with the *k* attributes listed in same order as they appear in list. Duplicate tuples are removed from the result.

** NOTE: Commutativity does *not* hold.

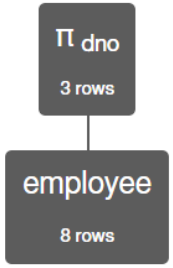
EXAMPLE 3: (Company schema):

List all the department numbers where employees work

π dno employee

or can write as:

Pi dno employee



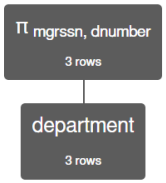
π_{dno} employee

employee.dno
5
4
1

Returns:{5, 4, 1}

EXAMPLE 4: List all managers (ssn) and the departments (number) they manage

π mgrssn, dnumber department



π mgrssn, dnumber department

department.mgrssn	department.dnumber
333445555	5
987654321	4
888665555	1

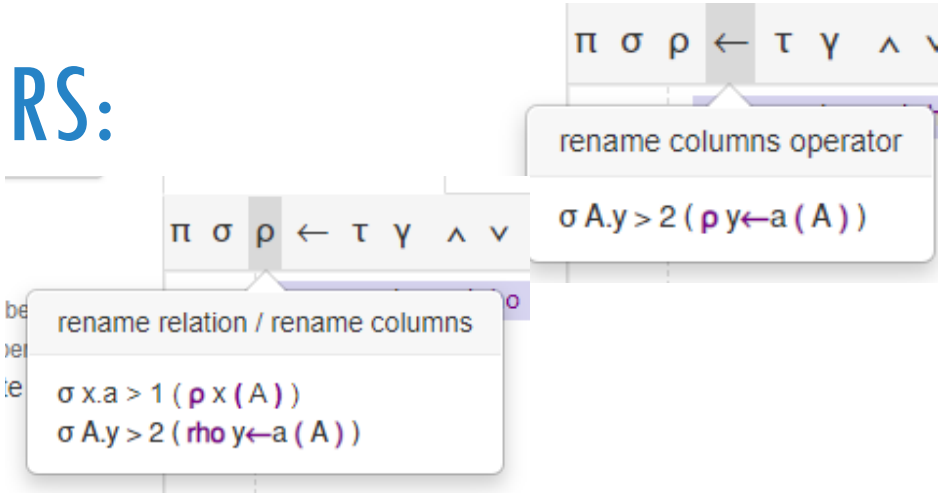
YOU TRY . . .

Example 5 Return all project locations which are in dept 5

Example 6 Return the names of all employees in department 5

Example 7.List the names of all employees whose salary is greater than 45000

RENAME OPERATORS: RHO *P AND* ←



Rename Operation (ρ)

Notation – $\rho_x(E)$

Where the result of expression **E** is saved with name of **x**

You might want to do this to save typing a table name,

e.g., for table dependent might want to rename it as dep as follows:

```
π dep.bdate (rho dep (dependent))
```

NOTE: ASSIGNMENT ALSO AVAILABLE

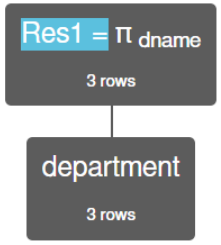
BUT NOT A RELATIONAL ALGEBRA OPERATOR

-- definition

```
Res1 = π dname department
```

-- execution

```
Res1
```



π dname department

department.dname
'Research'
'Administration'
'Headquarters'

Order operator

τ (tau)

Used to order by certain columns from a relation R

Notation: $\tau_{A1,\ A2,\ \ldots,\ Ak}\ R$

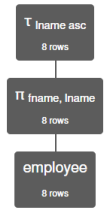
where:

`A1, A2, …, Ak` : are attributes with either asc or desc

R: relation/table name

EXAMPLE 8: (Company schema):
List all the employee first names and surnames, ordered by surname (asc)

τ lname asc (π fname, lname employee)



τ lname asc (π fname, lname employee)

employee.fname	employee.lname
'James'	'Borg'
'Joyce'	'English'
'Ahmad'	'Jabbar'
'Ramesh'	'Narayan'
'John'	'Smith'
'Jennifer'	'Wallace'
'Franklin'	'Wong'
'Alicia'	'Zelaya'

or can write as:

τau lname asc (π fname, lname employee)

asc is default ordering

Group By operator

γ (gamma)

Used to group by certain columns from a relation R

AGGREGATE FUNCTIONS SUPPORTED
(THOUGH NOT PART OF RELATIONAL ALGEBRA)

COUNT(*)

COUNT(column)

MIN(column)

MAX(column)

SUM(column)

AVG(column)

BINARY OPERATORS

General Syntax:

(child_expression) function argument (child_expression)

UNION OPERATOR: ∪

Notation: (R) ∪ (S)

where R and S are relations/tables

Returns all tuples from R and all tuples from S

Notes:

- No duplicates will be returned.

INTERSECTION OPERATOR: n

Notation: (R) n (S)

where R and S are relations/tables

Result: returns all tuples from R that are also in S.

SET DIFFERENCE: -

Notation: (R) – (S)

where R and S are relations/tables

Result:

returns tuples that are in relation R but not in S

Note: (R) – (S) and (S) – (R) are not the same

UNION COMPATIBILITY

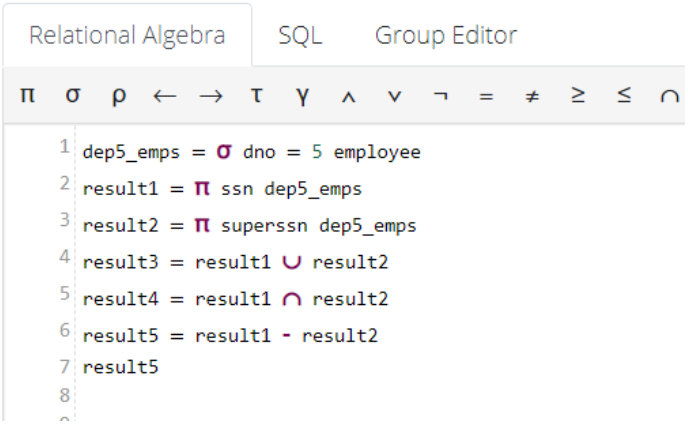
For union, intersection and minus, relations must be **union compatible**, that is:

- schemas of relations must match, i.e., same number of attributes and each corresponding attributes have the same domain

EXAMPLE 9:

What is displayed in the results relation following these operations? (using ReLaX schema)

dep5_emps = σ dno = 5 employee
result1 = π ssn dep5_emps
result2 = π superssn dep5_emps
result3 = result1 ∪ result2
result4 = result1 ∩ result2
result5 = result1 - result2
result5



EXAMPLE 9: *ctd.*

result1

ssn
123456789
333445555
666884444
453453453

result2

superssn
333445555
888665555

result1 u result2

ssn
123456789
333445555
666884444
453453453
888665555

EXAMPLE 9 *ctd.*

`result1`

ssn
123456789
333445555
666884444
453453453

`result2`

superssn
333445555
888665555

`result1 n result2`

ssn
333445555

EXAMPLE 8 *ctd.*

`result1`

ssn
123456789
333445555
666884444
453453453

`result2`

superssn
333445555
888665555

`result1-result2`

ssn
123456789
666884444
453453453

CARTESIAN PRODUCT OPERATOR: X (cross join)

Notation: (R) X (S) where R and S are relations/tables

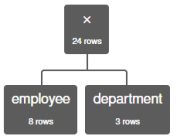
Returns: tuples comprising the concatenation (combination) of *every tuple* in R with every tuple in S

Note:

No condition is specified

Example:

```
employee x department
```



employee × department

Execution time: 2 ms

employee.fname	employee.minit	employee.lname	employee.ssn	employee.bdate	employee.address	employee.gender	employee.salary	empl
'John'	'B'	'Smith'	123456789	'1975-Jan-09'	'731 Fondren, Houston, TX'	'Man'	55250	
'John'	'B'	'Smith'	123456789	'1975-Jan-09'	'731 Fondren, Houston, TX'	'Man'	55250	
'John'	'B'	'Smith'	123456789	'1975-Jan-09'	'731 Fondren, Houston, TX'	'Man'	55250	
'Franklin'	'T'	'Wong'	333445555	'1980-Dec-08'	'638 Voss, Houston, TX'	'Man'	65000	
'Franklin'	'T'	'Wong'	333445555	'1980-Dec-08'	'638 Voss	'Man'	65000	

EXAMPLE 10:

Given relations: **R**(A, B) and **S**(C, D, E):

A	B
1	2
3	4

C	D	E
22	55	66
44	77	88
99	10	11

Then R x S is?

R

A	B
1	2
3	4

S

C	D	E
22	55	66
44	77	88
99	10	11

R x S =

A	B	C	D	E
1	2	22	55	66
1	2	44	77	88
1	2	99	10	11
3	4	22	55	66
3	4	44	77	88
3	4	99	10	11

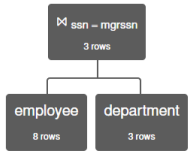
JOIN OPERATOR: ⋈

The Join operator is a hybrid operator – it is a combination of the Cartesian product operator (x) and a select operator (σ)

Tables are joined together based on the **condition** specified

Example:

`employee ⋈ ssn = mgrssn department`



employee ⋈ ssn = mgrssn department

employee.lname	employee.ssn	employee.bdate	employee.addre
'Wong'	333445555	'1955-Dec-08'	'638 Voss, Houston, TX'
'Wallace'	987654321	'1941-Jun-20'	'291 Berry, Bellaire, TX'

Cartesian product versus Join?

The main difference between a Cartesian product operator and a join operator is that with a join, only tuples **satisfying a condition** appear in the result (as we have already seen)

In a Cartesian product operator, all combinations of tuples are included in the result.

EQUI AND THETA JOINS

Notation: (**R1**) ⋈ p (**R2**)
where:
p: Join condition
R1 and R2: relations/tables
Result: The JOIN operation returns all combinations of tuples from relation R1 and relation R2 satisfying the join condition p

Note:
EQUI JOINS use only equality comparisons (=) in the join condition p

EXTRA EXAMPLES ….

11. Write the relational algebra expression to find the names of the employees in the Research department
12. Find the name(s) of Jennifer Wallace's dependents
13. Find the name(s) of employees who work on projects which are located in Houston

SUMMARY

Important to know:

- Unary relational algebra operators and how they work – especially, σ and π
- Binary relational algebra operators and how they work – especially x and $\bowtie$
- How to combine binary operators (where order is significant) to answer a question
- Using the ReLaX calculator

VERY Important not to confuse SQL and Relational Algebra