SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

COURSE NAME : 19CS402 - DATABASE MANAGEMENT SYSTEMS

II YEAR / III SEMESTER

Unit – 2

Relational Algebra

Relational Query Languages

- *Query languages*: Allow manipulation and retrieval of data from a database.
- Relational model supports simple, powerful QLs:
  - Strong formal foundation based on logic.
  - Allows for much optimization.
- Query Languages != programming languages!
  - QLs not intended to be used for complex calculations.
  - QLs support easy, efficient access to large data sets.

Formal Relational Query Languages

- Two mathematical Query Languages form the basis for "real" languages (e.g. SQL), and for implementation:
- Categories
  - Procedural Language : The user instructs the system to perform a sequence of operations on the database to compute the desired result.
  - Eg: *Relational Algebra*
  - *Non Procedural Language* : The user describes the desired information without giving a specific procedure for obtaining that information.
  - Eg: Relational Calculus

Preliminaries

- A query is applied to *relation instances*, and the result of a query is also a relation instance.
  - *Schemas* of input relations for a query are fixed (but query will run regardless of instance!)
  - The schema for the *result* of a given query is also fixed! Determined by definition of query language constructs.

Example Instances

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

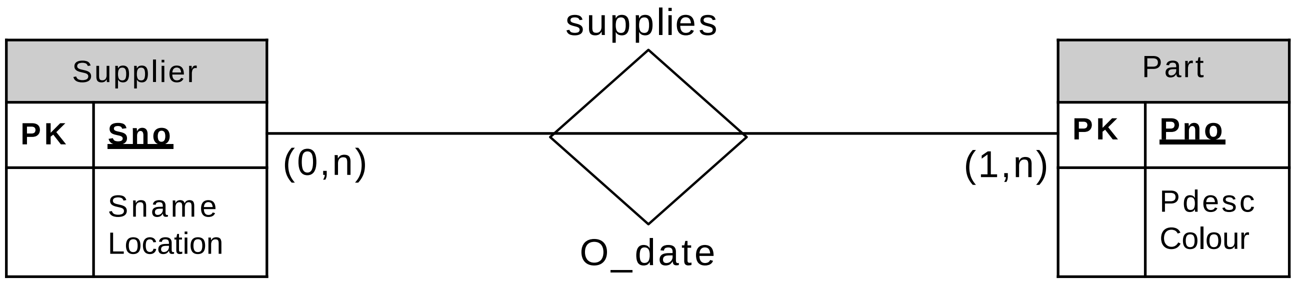
S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

Supplier-Part Example



Supplier

Sno	Sname	Location
s1	Acme	NY
s2	Ajax	Bos
s3	Apex	Chi
s4	Ace	LA
s5	A-1	Phil

Part

Pno	Pdesc	Colour
p1	screw	red
p2	bolt	yellow
p3	nut	green
p4	washer	red

Supplies

Sno	Pno	O_date
s1	p1	nov 3
s2	p2	nov 4
s3	p1	nov 5
s3	p3	nov 6
s4	p1	nov 7
s4	p2	nov 8
s4	p4	nov 9

Relational Algebra

- Basic operations:
  - *Selection* (σ) Selects a subset of rows from relation.
  - *Projection* (π) Deletes unwanted columns from relation.
  - *Cross-product* ($\times$) Allows us to combine two relations.
  - *Set-difference* ($-$) Tuples in reln. 1, but not in reln. 2.
  - *Union* ($\cup$) Tuples in reln. 1 and in reln. 2.
  - *Rename* (ρ) Renaming the reln 1 to reln 2
- Additional operations:
  - *Intersection, join, division, Assignment* Not essential, but (very!) useful.
- Extendend operations:
  - *Aggregate and outerjoin*
- Since each operation returns a relation, operations can be *composed*! (Algebra is "closed".)

The ——————— operation, denoted by −, allows us to find tuples that are in one relation but are not in another.

a) Union
b) Set-difference
c) Difference
d) Intersection

b) Set-difference

Projection

- Deletes attributes that are not in *projection list*.
- *Schema* of result contains exactly the fields in the projection list, with the **same names** that they had in the (only) input relation.
- **Projection operator has to eliminate** *duplicates*!

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

$$\pi_{sname,rating}(S2)$$

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S2

age
35.0
55.5

$$\pi_{age}(S2)$$

Relational Algebra is a____________query language that takes two relations as input and produces another relation as an output of the query.

a) Relational
b) Structural
c) Procedural
d) Fundamental

c) Procedural

Projection:

- Projection returns a subset of the columns of a single table.
- Syntax:

$\pi_{\text{<list of columns>}}$ (table_name)

Find all supplier names. Project Supplier over Sname

π_{Sname} (Supplier)

Supplier

Sno	Sname	Location
s1	Acme	NY
s2	Ajax	Bos
s3	Apex	Chi
s4	Ace	LA
s5	A-1	Phil

Answer

Sname
Acme
Ajax
Apex
Ace
A-1

Projection Exercise:

- **Find the addresses of all Cardholders.**

  π_{b_addr} (Cardholder)

Cardholder			
borrower#	b-name	b-address	b-status
1234	john	New Paltz	senior
1345	albert	Rosendale	senior
1325	jo-ann	New Paltz	junior
2653	mike	Modena	senior
7635	john	Kingston	junior
9823	diana	Tilson	senior
5342	susan	Walkill	senior

- Observations:
  - There is only one input table.
  - The schema of the answer table is the **list of columns**
  - If there are many Cardholders living at the same address these are **not duplicated** in the answer table.

Projection:

schema of answer table is the same as the list of columns in the query

Cardholder

borrower#	b-name	b-address	b-status
1234	john	New Paltz	senior
1345	albert	Rosendale	senior
1325	jo-ann	New Paltz	junior
2653	mike	Modena	senior
7635	john	Kingston	junior
9823	diana	Tilson	senior
5342	susan	Walkill	senior

b-address
New Paltz
Rosendale
Modena
Kingston
Tilson

Duplicate 'New Paltz' values in the Cardholder table are dropped from the Answer table

Selection

- Selects rows that satisfy *selection condition*.
- *Schema* of result identical to schema of (only) input relation.
- *Result* relation can be the *input* for another relational algebra operation! (*Operator composition.*)

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S2

sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0

$$\sigma_{rating > 8}(S2)$$

sname	rating
yuppy	9
rusty	10

$$\pi_{sname,rating}(\sigma_{rating > 8}(S2))$$

Selection:

- Selection returns a subset of the rows of a single table.
- Syntax:

$\sigma_{\text{<condition>}}$ **(table_name)**

Find all suppliers from Boston.

$\sigma_{\text{Location = 'Bos'}}$ (Supplier)

Supplier

Sno	Sname	Location
s1	Acme	NY
s2	Ajax	Bos
s3	Apex	Chi
s4	Ace	LA
s5	A-1	Phil

Answer

Sno	Sname	Location
s2	Ajax	Bos

Selection Exercise:

- **Find the Cardholders from Modena.**

$$\sigma_{\text{b\_addr = 'Modena'}} \textbf{(Cardholder)}$$

- Observations:
  - There is only one input table.
  - Both *Cardholder* and the answer table have the same schema (list of columns)
  - Every row in the answer has the value ‘Modena’ in the *b_addr* column.

Cardholder

borrower#	b-name	b-address	b-status
1234	john	New Paltz	senior
1345	albert	Rosendale	senior
1325	jo-ann	New Paltz	junior
2653	mike	Modena	senior
7635	john	Kingston	junior
9823	diana	Tilson	senior
5342	susan	Walkill	senior

Selection:

same schema

Cardholder

borrower#	b-name	b-address	b-status
1234	john	New Paltz	senior
1345	albert	Rosendale	senior
1325	jo-ann	New Paltz	junior
2653	mike	Modena	senior
7635	john	Kingston	junior
9823	diana	Tilson	senior
5342	susan	Walkill	senior

Answer

borrower#	b-name	b-address	b-status
2653	mike	Modena	senior

All rows in the answer have the value 'Modena' in the *b_addr* column

Union, Intersection, Set-Difference

- All of these operations take two input relations, which must be *union-compatible*:
  - Same number of fields.
  - `Corresponding' fields have the same type.

Union, Intersection, Set-Difference

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$S1 - S2$

sid	sname	rating	age
22	dustin	7	45.0

$S1 \cup S2$

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0
44	guppy	5	35.0
28	yuppy	9	35.0

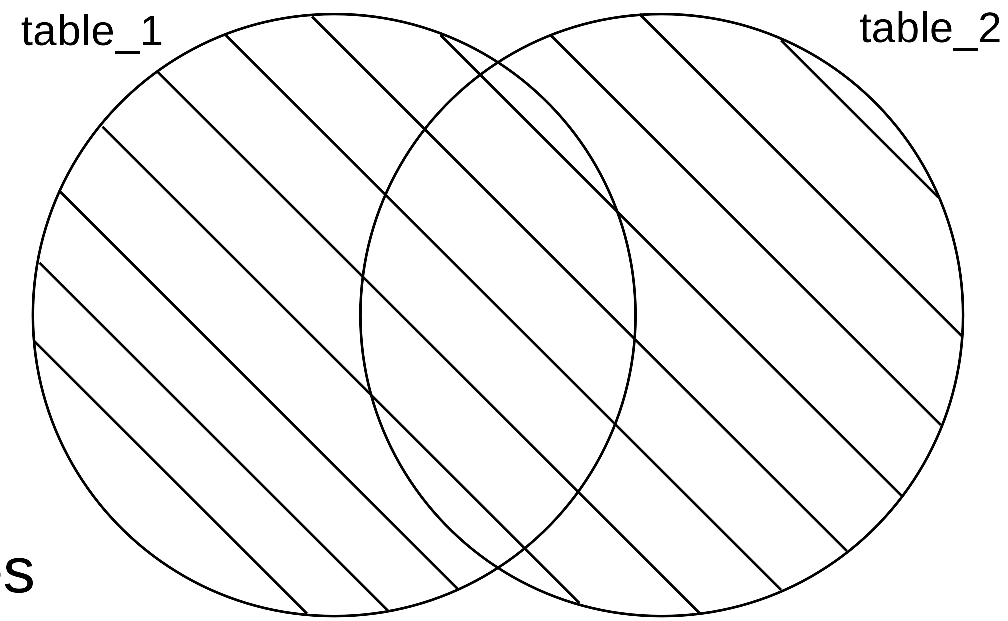
$S1 \cap S2$

sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35.0

Union:

- Treat two tables as sets and perform a set union
- Syntax:

Table1 ∪ Table2



- Observations:
  - This operation is impossible unless both tables involved have the same schemas. Why?
  - Because rows from both tables must fit into a single answer table; hence they must “look alike”.
  - Because some rows might already belong to both tables

Union Example:

Part1Suppliers = $\pi_{Sno}(\sigma_{Pno = 'p1'}$ (Supplies)

Part2Suppliers = $\pi_{Sno}(\sigma_{Pno = 'p2'}$ (Supplies)

Supplies

Sno	Pno	O_date
s1	p1	nov 3
s2	p2	nov 4
s3	p1	nov 5
s3	p3	nov 6
s4	p1	nov 7
s4	p2	nov 8
s4	p4	nov 9

Answer = Part1Suppliers ⋃ Part2Suppliers

Part1Suppliers

Sno
s1
s3
s4

Part2Suppliers

Sno
s2
s4

Part1Suppliers union Part2Suppliers

Sno
s1
s2
s3
s4

Union Exercise:

Find the borrower numbers of all borrowers who have either borrowed <u>or</u> reserved a book (any book).

Reservers = $\pi_{borrowerid}$ (Reserves)
Borrowers = $\pi_{borrowerid}$(Borrows)

Answer = Borrowers ∪ Reservers

Borrowers

borrowerid
1234
1325
2653
7635
9823
5342

Reservers

borrowerid
1345
1325
9823
2653
7635

not duplicated

Borrowers union Reservers

borrowerid
1234
1325
2653
7635
9823
5342
1345

Which of the following is used to denote the selection operation in relational algebra?

a) Pi (Greek)
b) Sigma (Greek)
c) Lambda (Greek)
d) Omega (Greek)

b) Sigma (Greek)

For select operation the__________appear in the subscript and the _______________argument appears in the parenthesis after the sigma.

a) Predicates, relation

b) Relation, Predicates

c) Operation, Predicates

d) Relation, Operation

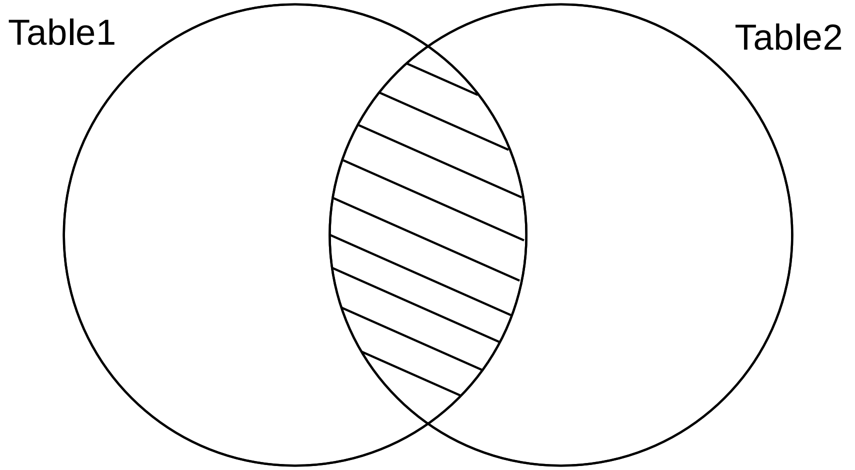
Predicates, relation

Intersection:

- Treat two tables as sets and perform a set intersection

- **Syntax:**

Table1 ∩ Table2



- Observations:
  - This operation is impossible unless both tables involved have the same schemas. Why?
  - Because rows from both tables must fit into a single answer table; hence they must “look alike”.

Intersection Example:

Part1Suppliers = $\pi_{Sno}(\sigma_{Pno = 'p1'}$ (Supplies))
Part2Suppliers = $\pi_{Sno}(\sigma_{Pno = 'p2'}$ (Supplies))

Supplies

Sno	Pno	O_date
s1	p1	nov 3
s2	p2	nov 4
s3	p1	nov 5
s3	p3	nov 6
s4	p1	nov 7
s4	p2	nov 8
s4	p4	nov 9

Answer = Part1Suppliers ∩ Part2Suppliers

Part1Suppliers

Sno
s1
s3
s4

Part2Suppliers

Sno
s2
s4

Part1Suppliers intersect Part2Suppliers

Sno
s4

Intersection Exercise:

- **Find the borrower numbers of all borrowers who have borrowed <u>and</u> reserved a book.**

Reservers = $\pi_{borrowerid}$ (Reserves)

Borrowers = $\pi_{borrowerid}$(Borrows)

Answer = Borrowers ∩ Reservers

Borrowers

borrowerid
1234
1325
2653
7635
9823
5342

Reservers

borrowerid
1345
1325
9823
2653
7635

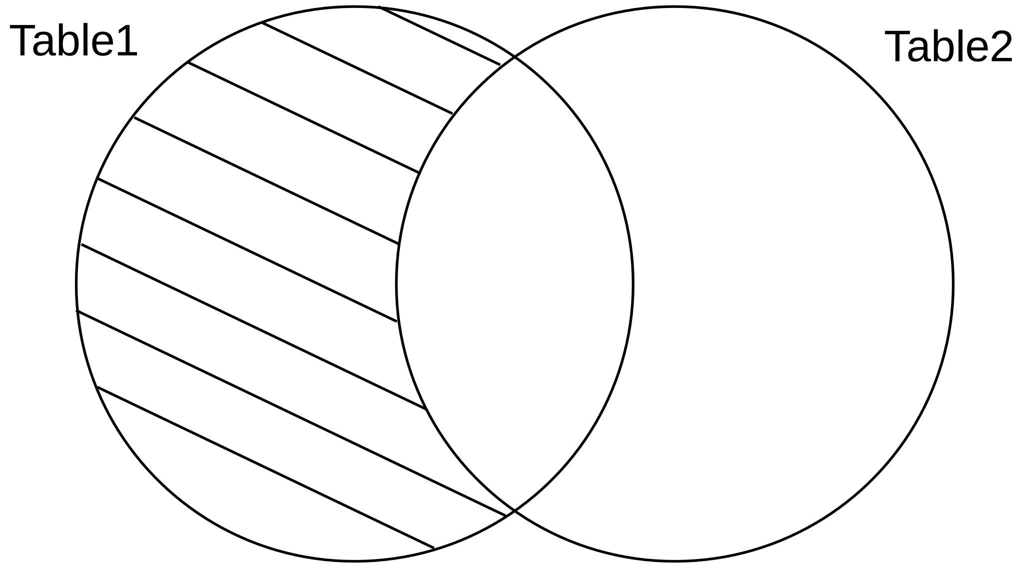
Borrowers intesect Reservers

borrowerid
1325
2653
7635
9823

Set Difference:

- Treat two tables as sets and perform a set intersection
- **Syntax:**

Table1 - Table2



- Observations:
  - This operation is impossible unless both tables involved have the same schemas. Why?
  - Because it only makes sense to calculate the set difference if the two sets have elements in common.

Set Difference Example:

Part1Suppliers = $\pi_{Sno}(\sigma_{Pno\ =\ 'p1'}$ (Supplies))
Part2Suppliers = $\pi_{Sno}(\sigma_{Pno\ =\ 'p2'}$ (Supplies))

Supplies

Sno	Pno	O_date
s1	p1	nov 3
s2	p2	nov 4
s3	p1	nov 5
s3	p3	nov 6
s4	p1	nov 7
s4	p2	nov 8
s4	p4	nov 9

Answer = Part1Suppliers - Part2Suppliers

Part1Suppliers

Sno
s1
s3
s4

Part2Suppliers

Sno
s2
s4

Part1Suppliers minus Part2Suppliers

Sno
s1
s3

Set Difference Exercise:

- **Find the borrower numbers of all borrowers who have borrowed something and reserved nothing.**

Reservers = $\pi_{borrowerid}$ (Reserves)

Borrowers = $\pi_{borrowerid}$(Borrows)

Answer = Borrowers - Reservers

Borrowers

borrowerid
1234
1325
2653
7635
9823
5342

Reservers

borrowerid
1345
1325
9823
2653
7635

Borrowers minus Reservers

borrowerid
1234
5342

The ______________ operation, denoted by -, allows us to find tuples that are in one relation but are not in another.

a.	Union
b.	Set-difference
c.	Difference
d.	Intersection

b) Set Difference

If E1 and E2 are relational algebra expressions, then which of the following is NOT a relational algebra expression ?

a.	E1 U E2
b.	E1 / E2
c.	E1 - E2
d.	E1 x E2

b) E1 / E2

The operation of a relation X, produces Y, such that Y contains only selected attributes of X. Such an operation is :

a.	Projection
b.	Intersection
c.	Union
d.	Difference

a) Projection

Relational algebra is :

a.	Data Definition Language
b.	Meta Language
c.	Procedural query language
d.	Non procedural language

c) Procedural query language

The result of the UNION operation between R1 and R2 is a relation that includes

a.	all the tuples of R1
b.	all the tuples of R2
c.	all the tuples of R1 and R2
d.	all the tuples of R1 and R2 which have common columns

D) all the tuples of R1 and R2 which have common columns

Cross-Product/Cartesian Product

- **Each row of S1 is paired with each row of R1.**
- *Result schema* has one field per field of S1 and R1, with field names `inherited' if possible.
  - *Conflict*: Both S1 and R1 have a field called *sid*.

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

- *Renaming operator*: $\rho\ (C(1 \rightarrow sid1, 5 \rightarrow sid2),\ S1 \times R1)$

Cross-Product/Cartesian Product

- The Cartesian product of **two sets** is a **set of pairs of elements (tuples), one from each set.**
- If the original sets are already sets of tuples then the tuples in the Cartesian product are all that bigger.
- Syntax:

<table_name> x <table_name>

- As we have seen, Cartesian products are usually **unrelated to a real-world** thing. They normally contain some ***noise*** **tuples.**
- However they may be useful as a first step.

Cross-Product/Cartesian Product Example

5 rows

Supplier

Sno	Sname	Location
s1	Acme	NY
s2	Ajax	Bos
s3	Apex	Chi
s4	Ace	LA
s5	A-1	Phil

4 rows

Part

Pno	Pdesc	Colour
p1	screw	red
p2	bolt	yellow
p3	nut	green
p4	washer	red

Supplier x Part **20 rows**

Sno	Sname	Location	Pno	Pdesc	Color
s1	Acme	NY	p1	screw	red
s2	Ajax	Bos	p1	screw	red
s3	Apex	Chi	p1	screw	red
s4	Ace	LA	p1	screw	red
s5	A-1	Phil	p1	screw	red
s1	Acme	NY	p2	bolt	yellow
. . .					
s5	A-1	Phil	p4	washer	red

info: 7 rows in total

noise: 13 rows in total

Cross-Product/Cartesian Product Exercise:

Names = Project Cardholder over b_name
Addresses = Project Cardholder over b_addr

Names x Addresses

Names

b_name
john
albert
jo-ann
mike
diana
susan

Addresses

b_addr
New Paltz
Rosendale
Modena
Kingston
Tilson
Wallkill

Names x Addresses

Names x Addresses

b_name	b_addr
john	New Paltz
albert	New Paltz
jo-ann	New Paltz
mike	New Paltz
diana	New Paltz
susan	New Paltz
john	Rosendale
. . .	
susan	Wallkill

. . .

noise

Info = project cardholder over b_name, b_addr

How many rows? **36**

Joins

- *Condition Join*:

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	58	103	11/12/96

$$S1 \bowtie_{S1.sid < R1.sid} R1$$

- *Result schema* same as that of cross-product.
- **Fewer tuples than cross-product.**
- Filters tuples not satisfying the join condition.
- Sometimes called a *theta-join*.

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Joins

- *Equi-Join*: A special case of condition join where the condition *c* contains only ***equalities***.

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/96
58	rusty	10	35.0	103	11/12/96

- *Result schema* similar to cross-product, but only one copy of fields for which equality is specified.
- *Natural Join*: Equijoin on *all* common fields.

$$\pi_{sid,..,age,bid,..}(S1 \bowtie_{sid} R1)$$

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Joins

- The most useful and most common operation.
- Tables are “related” by having columns in common; ***primary key*** on one table appears as a ***“foreign” key*** in another.
- *Join* uses this ***relatedness to combine the two tables into one***.
- *Join* is usually needed when a ***database query involves knowing something found in one table but wanting to know something found in a different table***.
- *Join* is useful because ***both Select and Project work on only one table at a time.***

Join Example:

- Suppose we want to know the names of all parts ordered between Nov 4 and Nov 6.

The names we want are here

related tables

Supplier

Sno	Sname	Location
s1	Acme	NY
s2	Ajax	Bos
s3	Apex	Chi
s4	Ace	LA
s5	A-1	Phil

Part

Pno	Pdesc	Colour
p1	screw	red
p2	bolt	yellow
p3	nut	green
p4	washer	red

Supplies

Sno	Pno	O_date
s1	p1	nov 3
s2	p2	nov 4
s3	p1	nov 5
s3	p3	nov 6
s4	p1	nov 7
s4	p2	nov 8
s4	p4	nov 9

What we know is here?

Join Example:

- **Step 1:** Without the join operator we would start by combining the two tables using Cartesian Product.

Part x Supplies

- The table, *Supplies x Part*, now contains both
  - **What we know** (*OrderDate*) and
  - **What we want** (*PartDescription*)
- The schema of *Supplies x Part* is:

Supplies x Part = {Sno, Pno, ODate, Pno, PDesc, Colour}

What we know. (ODate) **What we want** (PDesc)

- We know, that a Cartesian Product contains some info rows but lots of noise too.

Join Example:

- The Cartesian Product has <u>noise</u> rows we need to get rid of

Supplies.Pno = Part.Pno

Supplies.Pno != Part.Pno

Supplies x Part

Sno	Pno	O_date	Pno	Pdesc	Colour
s1	p1	nov 3	p1	screw	red
s1	p1	nov 3	p2	bolt	yellow
s1	p1	nov 3	p3	nut	green
s1	p1	nov 3	p4	washer	red
s2	p2	nov 4	p1	screw	red
. . .					
s4	p4	nov 9	p4	washer	red

info

noise

Join Example:

- **Step 2:** Let's get rid of all the noise rows from the Cartesian Product.

A = select (Supplies x Part) where Supplies.PNo = Part.PNo

identical columns

- The table, *A*, now contains both
  - **What we know** (*OrderDate*) and
  - **What we want** (*PartDescription*)
- – And no noise rows!

Select (Supplies x Part) where Supplies.Pno = Part.Pno

Sno	Pno	O_date	Pno	Pdesc	Colour
s1	p1	nov 3	p1	screw	red
s2	p2	nov 4	p2	bolt	yellow
s3	p1	nov 5	p1	screw	red
s3	p3	nov 6	p3	nut	green
s4	p1	nov 7	p1	screw	red
s4	p2	nov 8	p2	bolt	yellow
s4	p4	nov 9	p4	washer	red

Join Example:

- **Step 3:** We now have **two identical columns**
  - *Supplies.Pno* and *Part.Pno*
- We can ***safely get rid of one of these***

project(select (Supplies x Part) where Supplies.Pno = Part.Pno)
over Sno, Supplies.Pno, O_date, Pdesc, Colour

Sno	Pno	O_date	Pdesc	Colour
s1	p1	nov 3	screw	red
s2	p2	nov 4	bolt	yellow
s3	p1	nov 5	screw	red
s3	p3	nov 6	nut	green
s4	p1	nov 7	screw	red
s4	p2	nov 8	bolt	yellow
s4	p4	nov 9	washer	red

Join Example:

- Because the idea of:

1. ***taking the Cartesian Product of two tables with a common column,***
2. then ***select getting rid*** of the ***noise rows*** and finally
3. ***project getting rid of the duplicate column***

   is so common we give it a name - *JOIN*.

Project (Select (Supplies x Part) where Supplies.Pno = Part.Pno) over Sno, Supplies.Sno, O_date, Pdesc, Colour

Join Example:

- SYNTAX:

Supplies ⋈ Part

Supplies ⋈ Part =

project(select (Supplies x Part) where Supplies.Pno = Part.Pno) over Sno, Supplies.Pno, O_date, Pdesc, Colour

Sno	Pno	O_date	Pdesc	Colour
s1	p1	nov 3	screw	red
s2	p2	nov 4	bolt	yellow
s3	p1	nov 5	screw	red
s3	p3	nov 6	nut	green
s4	p1	nov 7	screw	red
s4	p2	nov 8	bolt	yellow
s4	p4	nov 9	washer	red

Join Example:

- Summary:
  - Used when **two tables are to be combined into one**
  - Most often, the **two tables share a column**
  - The **shared column is often a *primary key*** in one of the tables
  - ***Because it is a primary key in one table the shared column is called a foreign key in any other table*** that contains it
  - *JOIN* is a combination of
    - ***Cartesian Product (to combine 2 tables in 1)***
    - ***Select (rows with identical key values)***
    - ***Project (out one copy of duplicate column)***

Join Example: (Finishing Up):

- Let's finish up our query.

- **Step 4**: We know that the only rows that really interest us are those for Nov 4, 5 and 6.

A = Supplies JOIN Part

B = select A where O_date between 'Nov 4' and 'Nov 6'

B

Sno	Pno	O_date	Pdesc	Colour
s2	p2	nov 4	bolt	yellow
s3	p1	nov 5	screw	red
s3	p3	nov 6	nut	green

Join Example (Finishing Up):

- **Step 5:** What we wanted to know in the first place was the list of parts ordered on certain days.

B

Sno	Pno	O_date	Pdesc	Colour
s2	p2	nov 4	bolt	yellow
s3	p1	nov 5	screw	red
s3	p3	nov 6	nut	green

we want the values in this column

- **Final Answer:**

Answer

Pdesc
bolt
screw
nut

Answer = project B over Pdesc

Join Summary:

- *JOIN* is the operation most ***often used to combine two tables into one.***
- The kind of *JOIN* we ***performed where we compare two columns*** using the = operator is called the *natural equi-join*.
- It is also possible to compare columns using other operators such as **<, >, <=, !=** etc. Such joins are called *theta-joins*.
- These are expressed with a subscripted condition

$$\bowtie_{R.A\ \theta\ S.B}$$

where θ is any comparison operator except =

Join Exercise:

- Find the author and title of boo 2.0
  - **What we know**, ***purchase price***, is
  - **What we want**, ***author*** and ***title***, ar
  - ***Book*** and ***Copy*** share a primary k .ISBN ,

Copy.ISBN)

Copy

accession#	p-price	ISBN
qt-76.4c1	19.00	1-23
qt-78.2c1	30.00	4-76
qt-78.2c2	30.00	4-76
qs-77.3c1	11.00	6-99
qs-77.3c2	11.00	6-99
qs-77.3c3	11.00	6-99
qs-77.3c4	11.00	6-99
qs-77.3c5	12.00	6-99
qs-77.3c6	12.00	6-99
qp-91.2c1	21.00	3-56
qt-76.5c1	28.00	1-52
qt-76.5c2	28.00	1-52
qt-76.5c3	28.00	1-52
qt-75.5c1	35.00	7-45
qt-75.3c1	30.00	2-34
qt-75.3c2	37.00	2-34

purchase price of $12.00

info we want

Book

ISBN	title	author	pub-date	c-price	pub-name
1-23	DB	Ullman	1982	23.00	CSP
2-34	Netw	T'baum	1981	37.00	PH
3-56	Queue	K'rock	1978	25.00	Wiley
4-76	SysD	J'son	1981	32.00	PH
1-52	DB	Date	1984	28.00	AW
6-99	MMM	Br'kes	1978	12.00	AW
7-45	Arch	Baer	1981	35.00	CSP

Join Exercise:

- **Step 1:** *JOIN Copy* and *Book* — **A = Copy JOIN Book**
- **Step 2:** Find the copies that cost $12.00 — **B = Select A where p_price = 12.00**
- **Step 3:** Find the author and title of those books.

Answer = project B over author, title

Answer

author	title
Brookes	MMM

Division

- **Not supported as a primitive operator**, but useful for expressing queries like:
- *Find sailors who have reserved **<u>all</u>** boats.*
- **Precondition:** in A/B, the attributes in B must be included in the schema for A. Also, the result has attributes A-B.
  - **SALES(supId, prodId);**
  - **PRODUCTS(prodId);**
  - Relations SALES and PRODUCTS must be **built using projections**.
  - SALES/PRODUCTS: the ids of the suppliers supplying ALL products.

Examples of Division A/B

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

A

pno
p2

B1

pno
p2
p4

B2

pno
p1
p2
p4

B3

sno
s1
s2
s3
s4

A/B1

sno
s1
s4

A/B2

sno
s1

A/B3