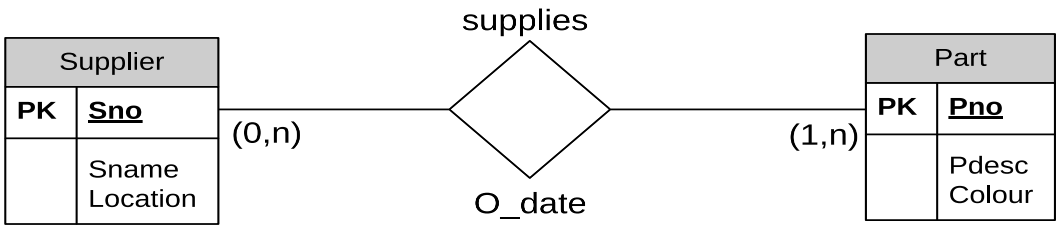
Relational Algebra

What is Relational Algebra?

- It is a language in which we can ask questions (query) of a database.
- Basic premise is that <u>tables are sets</u> (mathematical) and so our query language should manipulate sets with ease.

- Traditional Set Operations:
  - union, intersection, Cartesian product, set difference
- Extended Set Operations:
  - selection, projection, join, quotient

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

SELECTION:

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

select <table_name> where <condition>
/* the <condition> must involve only columns from the indicated table */

alternatively

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

Find all suppliers from Boston.

Supplier

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

Select Supplier
where Location = ‘Bos’

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

Answer

Sno	Sname	Location
s2	Ajax	Bos

SELECTION Exercise:

- Find the Cardholders from Modena.

select Cardholder where b_addr = ‘Modena’

alternatively

$\sigma_{\text{b\_addr = 'Modena'}}$ **(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.

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

PROJECTION:

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

project <table_name> over <list of columns>
/* the columns in <list of columns> must
come from the indicated table */

alternatively

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

Find all supplier names

Supplier

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

Project Supplier over Sname

π_{Sname} (Supplier)

Answer

Sname
Acme
Ajax
Apex
Ace
A-1

PROJECTION Exercise:

- Find the addresses of all Cardholders.

project Cardholder over b_addr

alternatively

π_{b_addr} **(Cardholder)**

- 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

Answer

b-address
New Paltz
Rosendale
Modena
Kingston
Tilson

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

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.

CARTESIAN PRODUCT:

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

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

Info =
project cardholder
over b_name, b_addr

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

How many rows? 36

UNION:

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

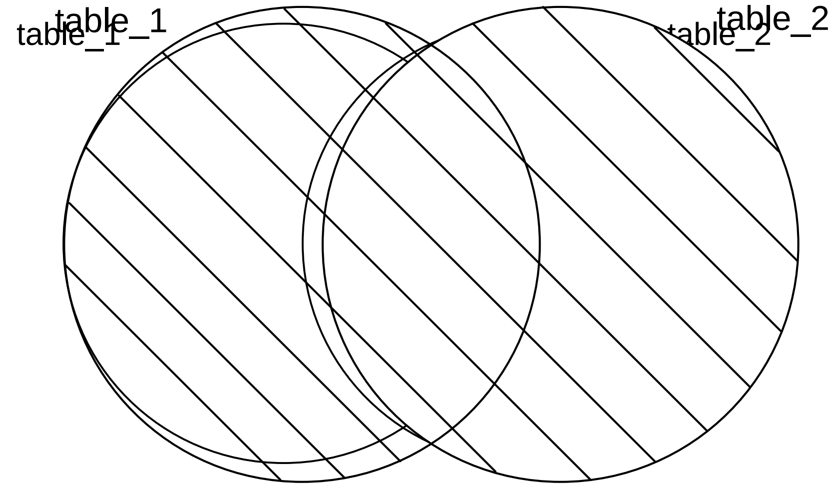


Table1 UNION Table2

alternatively

Table1 $\subset$ **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 = project (select Supplies where Pno = ‘p1’) over Sno
Part2Suppliers = project (select Supplies where Pno = ‘p2’) over Sno

Part1Suppliers UNION Part2Suppliers

alternatively

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

Answer = Part1Suppliers $\cup$ **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 = project Reserves over borrowerid
Borrowers = project Borrows over borrowerid
Answer = Borrowers union Reservers

alternatively

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

INTERSECTION:

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

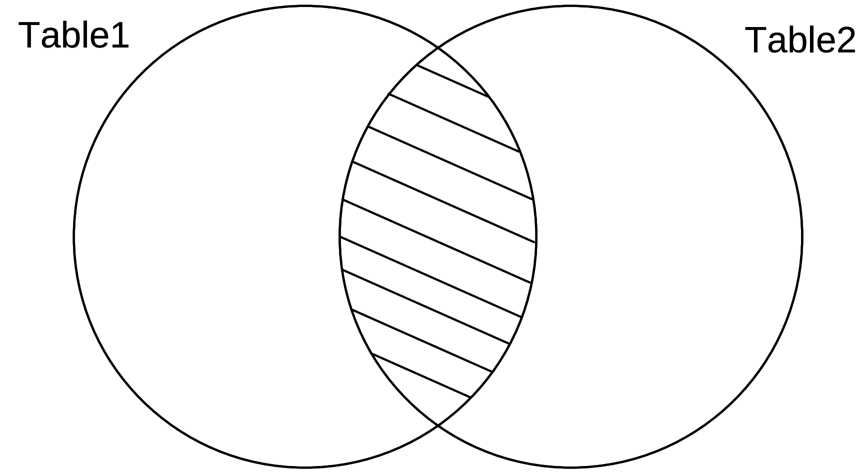


Table1 INTERSECTION Table2

alternatively

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 = project (select Supplies where Pno = ‘p1’) over Sno
Part2Suppliers = project (select Supplies where Pno = ‘p2’) over Sno

Part1Suppliers INTERSECT Part2Suppliers

alternatively

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

Answer = Part1Suppliers $\cap$ 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 = project Reserves over borrowerid
Borrowers = project Borrows over borrowerid

Answer = Borrowers intersect Reservers

alternatively

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

Answer = Borrowers $\cap$ 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:

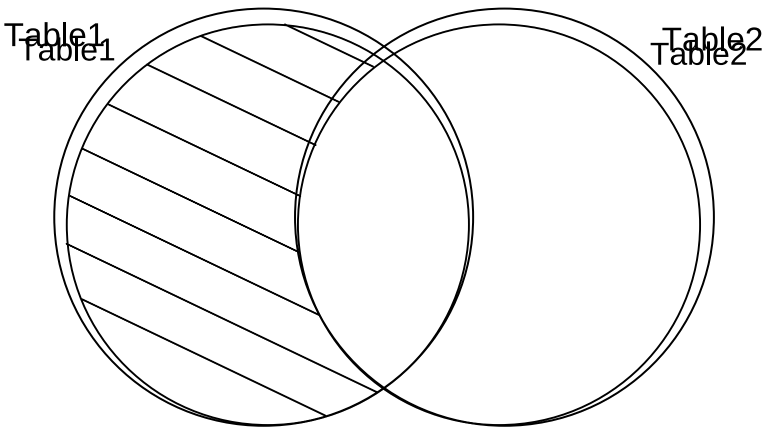


Table1 MINUS Table2

alternatively

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 = project (select Supplies where Pno = ‘p1’) over Sno
Part2Suppliers = project (select Supplies where Pno = ‘p2’) over Sno

Part1Suppliers MINUS Part2Suppliers

alternatively

Part1Suppliers = $\pi_{Sno}(\sigma_{Pno\,=\,\text{‘p1’}}\ (Supplies)\ )$
Part2Suppliers = $\pi_{Sno}(\sigma_{Pno\,=\,\text{‘p2’}}\ (Supplies)\ )$

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 = project Reserves over borrowerid
Borrowers = project Borrows over borrowerid

Answer = Borrowers minus Reservers

alternatively

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

JOIN:

- 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.

related tables

The names we want are here

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:

<u>Supplies x Part</u> = {Sno, Pno, ODate, Pno, PDesc, Colour}

What we know. **What we want**

- We know, from our previous lecture, that a Cartesian Product contains some <u>info</u> rows but lots of <u>noise</u> too.

JOIN Example

- The Cartesian Product has noise 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

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

identical columns

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 JOIN Part
alternatively
Supplies $\bowtie$ Part

Supplies $\bowtie$ 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 = project B over Pdesc

Answer

Pdesc
bolt
screw
nut

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 books purchased for $12.00
  - What we know, *purchase price*, is in the *Copy* table.
  - What we want, *author* and *title*, are in the *Book* table.
  - *Book* and *Copy* share a primary key/foreign key pair (*Book.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**

QUOTIENT

- Although Cartesian Product tables normally contain *noise* rows, sometimes they do not. Sometimes you can even find a Cartesian Product table inside another table.

Supplier4

Sno
s4

x

RedParts

Pno
p1
p4

=

Supplier4 x Redparts

Sno	Pno
s4	p1
s4	p4

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

Shows *s4* supplies all red parts {p1, p4}

- This often happens when we are interested in answering a query like

Find the suppliers who supply <u>all</u> red parts

QUOTIENT

- In fact, *QUOTIENT* is used precisely when the query contains the words *“all”* or *“every”* in the query condition.
- CARTESIAN PRODUCT contains this quality of *“all”.*
- In a CARTESIAN PRODUCT the elements of one set are combined with <u>all</u> elements of another.
- In the following slides we construct the answer to the query without using quotient just to show it can be done:

Find the suppliers who supply <u>all</u> red parts

QUOTIENT

SuppliedParts = project Supplies over Sno, Pno

7 rows

SuppliedParts

Sno	Pno
s1	p1
s2	p2
s3	p1
s3	p3
s4	p1
s4	p2
s4	p4

RedParts = project (select Part where Colour = 'Red') over Pno

RedParts

Pno
p1
p4

AllSuppliers = project Supplier over Sno

AllSuppliers

Sno
s1
s2
s3
s4
s5

QUOTIENT

AllSuppliers x RedParts

10 rows

AllSuppliers x RedParts

Sno	Pno	
s1	p1	**i**
s2	p1	**n**
s3	p1	**i**
s4	p1	**i**
s5	p1	**n**
s1	p4	**n**
s2	p4	**n**
s3	p4	**n**
s4	p4	**i**
s5	p4	**n**

Note: Like most Cartesian Products this table contains a few rows of info and the rest is noise

- Compare *AllSuppliers x RedParts* with *SuppliedParts*
  - they have the same schema – {Sno, Pno}.
  - *SuppliedParts* contains <u>only</u> info
  - *AllSuppliers x RedParts* contains some info (4 rows) and some noise (6 rows)
  - The rows they have in common are the info rows of *AllSuppliers x RedParts*

QUOTIENT

AllSuppliers x RedParts

Sno	Pno
s1	p1
s2	p1
s3	p1
s4	p1
s5	p1
s1	p4
s2	p4
s3	p4
s4	p4
s5	p4

SuppliedParts

Sno	Pno
s1	p1
s2	p2
s3	p1
s3	p3
s4	p1
s4	p2
s4	p4

- Next calculate:

NonSuppliedRedParts =
(AllSuppliers x RedParts) \ SuppliedParts

AllSuppliers x RedParts \ RedParts

Sno	Pno
s2	p1
s5	p1
s1	p4
s2	p4
s3	p4
s5	p4

NOTE: These are the "noise" rows of the Cartesian Product. We know that for every row in this table, the supplier mentioned did NOT supply the red part mentioned.

QUOTIENT

- The list of suppliers in *NonSuppliedRedParts* is important to us – this is a list of all suppliers who are *NOT* in our final answer.

NonAnswer =
project NonSuppliedRedParts over Sno

NonAnswer

Sno
s2
s5
s1
s3

- So the final answer is the suppliers in:

FinalAnswer =
AllSuppliers \ NonAnswer

FinalAnswer

Sno
s4

QUOTIENT

- This large amount of work is performed by the QUOTIENT operator (see JOIN motivation).
- **Definition**: If *R* and *S* are tables such that $\underline{S} \subseteq \underline{R}$, then the *QUOTIENT* of *R* by *S* (written *R/S*) is defined to be the largest table (call it *Q*) such that $Q \times S \subseteq R$.

SuppliedParts

Sno	Pno
s1	p1
s2	p2
s3	p1
s3	p3
s4	p1
s4	p2
s4	p4

/

RedParts

Pno
p1
p4

=

FinalAnswer = SuppliedParts / RedParts

FinalAnswer

Sno
s4

How to Use QUOTIENT

- Consider the query you are trying to answer; one that contains "all" in the condition.

Find the suppliers of all red parts

- We have to create three tables here – *R, S* and *Q*.
- We know that $\underline{Q} \cup \underline{S} = \underline{R}$.
- *S* contains whatever is described in the "all" phrase. In this case, *S = {all red parts}* and $\underline{S}$ = *{Pno}.*
- *Q* is the answer table so in this case, $\underline{Q}$ = *{Sno}.*
- Hence $\underline{R}$ = *{Sno,Pno}*, since $\underline{Q \times S} = \underline{R}$.

How to Use QUOTIENT

- Our problem becomes build a table *R* that is easy to build, has the correct schema and data related to what we are trying to find.
- In our example, we are asking to find suppliers who supply all red parts and so *R* must be about supplying parts; red or otherwise. Thus

R = project Supplies over Sno, Pno

- There is no choice for S. It must be

S = project (select Part where Colour = ‘Red’) over Pno

- Given *R* and *S, Q* <u>must</u> be the answer to the query.