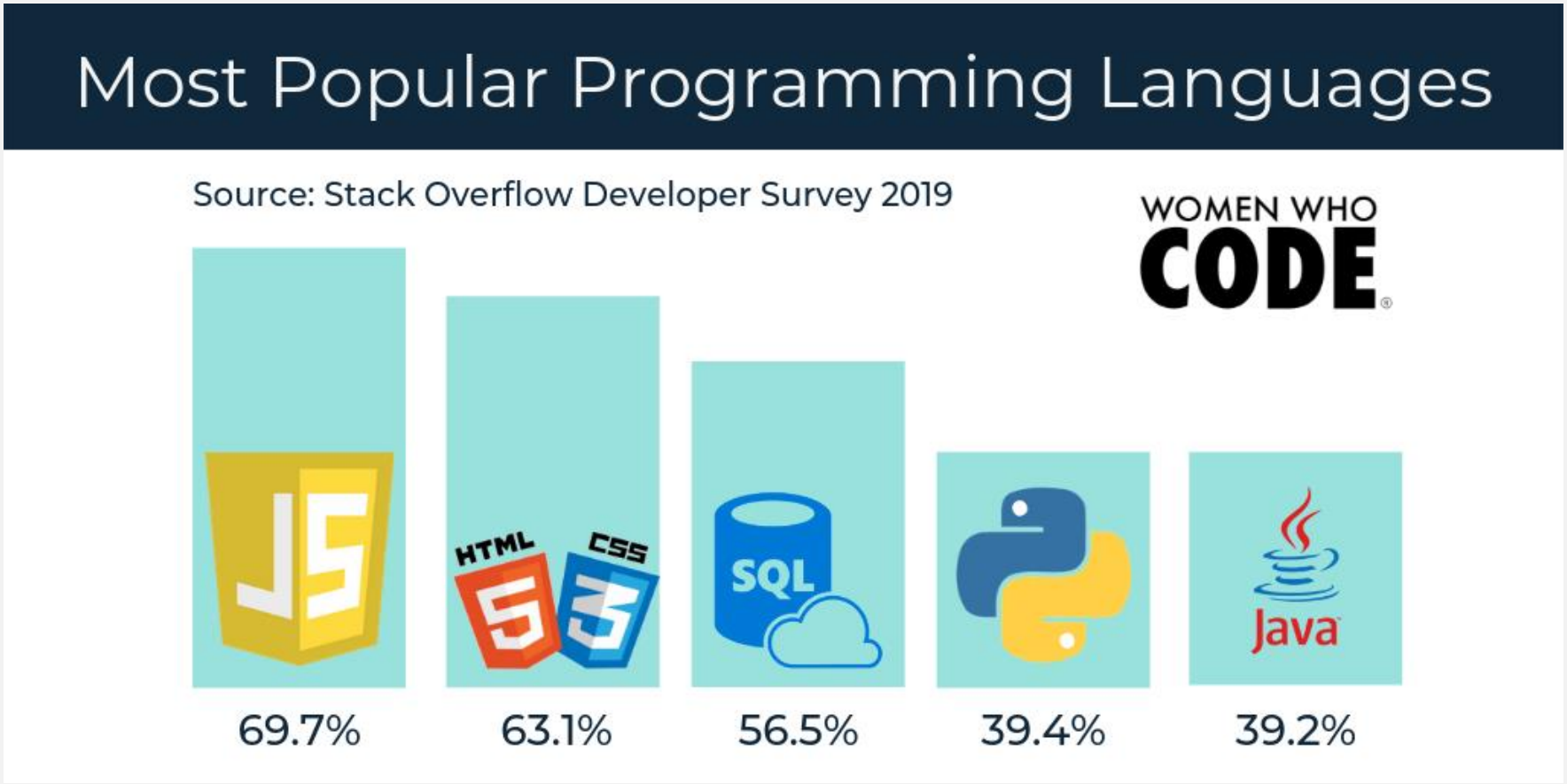
Relational Algebra

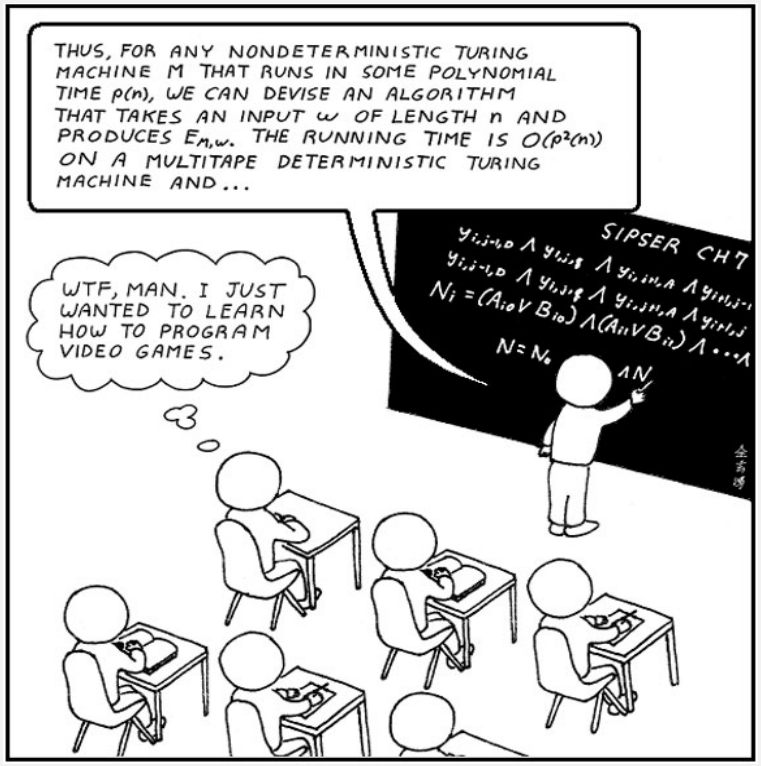
CS 355 Database Management System Design

S. Taneja
Oct 1, 2019

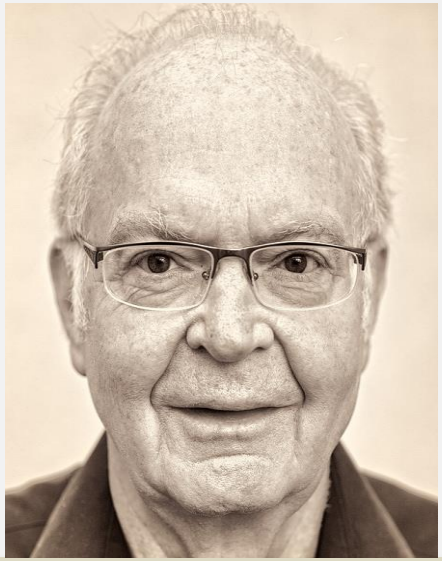
Saw this somewhere on the Internet yesterday….



Course Content



http://abstrusegoose.com/strips/computer_science_major.PNG



"If you find that you're spending almost all your time on theory, start turning some attention to practical things; it will improve your theories. If you find that you're spending almost all your time on practice, start turning some attention to theoretical things; it will improve your practice." **Don Knuth**

Formal Relational Languages

Relational algebra and **relational calculus** are two formal languages defined for the relational model. In contrast, **SQL** was the practical language for the relational model.

Both the algebra and the calculus were defined before SQL and in many ways form the basis of SQL and its implementation.

Relational algebra defines the basic **operators** for the relational model and specifies the rules for their use and composition.

IMP: Relational algebra *operators* are applied to *relations* and produce *relations* as their result.

Queries, or retrieval requests, can be specified as **relational algebra expressions**.

Some more facts about Relational Algebra

Relational algebra is inherently **procedural** and forms the basis of how SQL statements are implemented and executed. *(tell us HOW to do)*

Relational calculus is a higher-level **declarative** language for specifying relational queries. *(tells us WHAT to do) – we will not cover this in CS355*

Both relational algebra and relational calculus were defined by E.F. Codd, with elements of the algebra being defined in the landmark original paper on the relational model. **Codd's Theorem** states that the algebra and the calculus are logically equivalent.

Relational algebra

Basic Operators Required to provide full expressive power

Projection	$p_{attributes}(R)$	Project attribute values from all tuples of R
Selection	$s_{condition}(R)$	Select tuples from R that satisfy the condition
Cross-Product	$R \times S$	Concatenate each tuple in R with each tuple in S
Union	$R \cup S$	All tuples in either R or S
Difference	$R - S$	The tuples in R that are not in S
Rename	$Name \leftarrow Expression$	Names the result of an expression or renames an attribute

Relational algebra

Additional Operators Convenient but not strictly necessary

Intersection	$R \cap S$	Tuples in both R and S
Theta Join	$R \bowtie_C S$	Combined tuples from R and S that satisfy the condition C
Equijoin	$R \bowtie_C S$	Special case of theta join where C uses only equality
Natural Join	$R \bowtie S$	Join based on equality of shared attributes
Left Outer Join	$R \mathbin{⟕}_\theta S$	Join with non-matching tuples from R
Right Outer Join	$R \mathbin{⟖}_\theta S$	Join with non-matching tuples from S
Full Outer Join	$R \mathbin{⟗}_\theta S$	Join with non-matching tuples from R and S
Division	$R(XY),\ S(Y)$	Tuples in R that match with all relevant tuples in S

Relational algebra

Extended Operators Add missing functionality

Aggregate Functions Specify mathematical functions (e.g., counting, summing, etc.) on values or groups of values.

$$<grouping>\mathcal{F}_{<function>}(R)$$

Relational algebra interpreter

RA http://www.cs.duke.edu/~junyang/ra/

RA supports the following relational algebra operators:

\select_{cond} is the relational selection operator. For example, to select Drinker tuples with name Amy or Ben, we can write \select_{name = 'Amy' or name = 'Ben'} Drinker;. Syntax for cond follows SQL. Note that string literals should be enclosed in single quotes, and you may use boolean operators and, or, and not. Comparison operators <=, <, =, >, >=, and <> work on both string and numeric types. For string match you can use the SQL LIKE operator; e.g., \select_{name like 'A%'} drinker; finds all drinkers whose name start with A, as % is a wildcard character that matches any number of characters.
\project_{attr_list} is the relational projection operator, where attr_list is a comma-separated list of attribute names. For example, to find out what beers are served by Talk of the Town (but without the price information), we can write \project_{bar, beer} (\select_{bar = 'Talk of the Town'} Serves);.
\join_{cond} is the relational theta-join operator. For example, to join Drinker(name, address) and Frequents(drinker, bar, times_a_week) relations together using drinker name, we can write Drinker \join_{name = drinker} Frequents;. Syntax for cond again follows SQL; see notes on \select for more details.
\join is the relational natural join operator. For example, to join Drinker(name, address) and Frequents(drinker, bar, times_a_week) relations together using drinker name, we can write Drinker \join \rename_{name, bar, times_a_week} Frequents;. Natural join will automatically equate all pairs of identically named attributes from its inputs (in this case, name), and output only one attribute per pair. Here we use \rename to create two name attributes for the natural join; see notes on \rename below for more details.
\cross is the relational cross product operator. For example, to compute the cross product of Drinker and Frequents, we can write Drinker \cross Frequents;.
\union, \diff, and \intersect are the relational union, difference, and intersect operators. For a trivial example, to compute the union, difference, and intersection between Drinker and itself, we can write Drinker \union Drinker;, Drinker \diff Drinker;, and Drinker \intersect Drinker;, which would return Drinker itself, an empty relation, and Drinker itself, respectively.
\rename_{new_attr_name_list} is the relational rename operator, where new_attr_name_list is a comma-separated list of new names, one for each attribute of the input relation. For example, to rename the attributes of relation Drinker and compute the cross product of Drinker and itself, we can write \rename_{name1, address1} Drinker \cross \rename_{name2, address2} Drinker;.

Sample relations for example queries

Based on Codd's "suppliers" database

Suppliers

snum	sname	status	city
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens
S6	Brown	15	Berlin

Parts

pnum	pname	color	weight	city
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London
P7	Gear	Yellow	18	Berlin

Projects

jnum	jname	city
J1	Sorter	Paris
J2	Punch	Rome
J3	Reader	Athens
J4	Console	Athens
J5	Filler	London
J6	Layer	Oslo
J7	Tape	London

Shipments

snum	pnum	jnum	qty
S1	P1	J1	200
S1	P1	J4	700
S2	P3	J1	400
S2	P3	J2	200
S2	P3	J3	200
S2	P3	J4	500
S2	P3	J5	600
S2	P3	J6	400
S2	P3	J7	800
S2	P5	J2	100
S3	P3	J1	200
S3	P4	J2	500
S4	P6	J3	300
S4	P6	J7	300
S5	P2	J2	200
S5	P2	J4	100
S5	P5	J5	500
S5	P5	J7	100
S1	P4	J1	100
S1	P6	J2	200

Projection

$$\rho_{attributes}(R)$$

Projects the values of the specified attributes from all tuples in R.

$$\rho_{snum}(Suppliers)$$

```
ra> \project_{snum}(Suppliers);

Output schema: (snum text)
-----
S1
S2
S3
S4
S5
S6
-----
Total number of rows: 6
```

Suppliers

snum	sname	status	city
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens
S6	Brown	15	Berlin

The result of the PROJECT operation can be visualized as a **vertical partition** of the relation into two relations:

- One with the desired columns (attributes)
- The other contains the discarded columns

Projection

$$\rho_{attributes}(R)$$

Projects the values of the specified attributes from all tuples in R.

$$\rho_{pnum}(Shipments)$$

```
ra> \project_{pnum}(Shipments);

Output schema: (pnum text)
-----
P1
P2
P3
P4
P5
P6
-----
Total number of rows: 6
```

Shipments

snum	pnum	jnum	qty
S1	P1	J1	200
S1	P1	J4	700
S2	P3	J1	400
S2	P3	J2	200
S2	P3	J3	200
S2	P3	J4	500
S2	P3	J5	600
S2	P3	J6	400
S2	P3	J7	800
S2	P5	J2	100
S3	P3	J1	200
S3	P4	J2	500
S4	P6	J3	300
S4	P6	J7	300
S5	P2	J2	200
S5	P2	J4	100
S5	P5	J5	500
S5	P5	J7	100
S1	P4	J1	100
S1	P6	J2	200

Projection

$\pi_{city}(Suppliers)$

```
ra> \project_{city}(Suppliers);

Output schema: (city text)
-----
Athens
Berlin
London
Paris
-----
Total number of rows: 4
```

$\pi_{jnum,jname}(Projects)$

```
ra> \project_{jnum, jname}(Projects);

Output schema: (jnum text, jname text)
-----
J1|Sorter
J2|Punch
J3|Reader
J4|Console
J5|Filler
J6|Layer
J7|Tape
-----
Total number of rows: 7
```

Observations:

1. The result of a PROJECT operation has only the attributes specified in <attribute list> in the same order as they appear in the list. Thus, its **degree** is equal to the number of attributes in <attribute list>
2. *If the attribute list included only nonkey attributes of R, duplicate tuples are likely to occur (see e.g. on the left above). The PROJECT operation *removes any duplicate tuples.*

Projection

$$\rho_{pnum,jnum}(Shipments)$$

```
ra> \project_{pnum, jnum} (Shipments);

Output schema: (pnum text, jnum text)
-----
P1|J1
P1|J4
P2|J2
P2|J4
P3|J1
P3|J2
P3|J3
P3|J4
P3|J5
P3|J6
P3|J7
P4|J1
P4|J2
P5|J2
P5|J5
P5|J7
P6|J2
P6|J3
P6|J7
-----
Total number of rows: 19
```

Shipments

snum	pnum	jnum	qty
S1	P1	J1	200
S1	P1	J4	700
S2	P3	J1	400
S2	P3	J2	200
S2	P3	J3	200
S2	P3	J4	500
S2	P3	J5	600
S2	P3	J6	400
S2	P3	J7	800
S2	P5	J2	100
S3	P3	J1	200
S3	P4	J2	500
S4	P6	J3	300
S4	P6	J7	300
S5	P2	J2	200
S5	P2	J4	100
S5	P5	J5	500
S5	P5	J7	100
S1	P4	J1	100
S1	P6	J2	200

Selection

$$\sigma_{condition}(R)$$

Selects tuples from R that satisfy the given condition.

Think of it as a **horizontal partition of a relation** into two set of tuples
– one that satisfies the condition and are selected,
- Others that do not satisfy and are filtered out

$$\sigma_{status>15}(Suppliers)$$

```
ra> \select_{status > 15}(Suppliers);

-----
S1|Smith|20|London
S3|Blake|30|Paris
S4|Clark|20|London
S5|Adams|30|Athens
-----
Total number of rows: 4
```

Suppliers

snum	sname	status	city
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens
S6	Brown	15	Berlin

Selection

Note that the SELECT operator is unary; that is, it is applied to a single relation.

Moreover, SELECT is applied to each tuple individually.

The **degree** of the relation resulting from a SELECT operation – its number of attributes – is same as degree of R.

SELECT operation is commutative, that is

$$\sigma_{<cond1>}(\sigma_{<cond2>}(R)) = \sigma_{<cond2>}(\sigma_{<cond1>}(R))$$

Selection

$$\sigma_{qty<200}(Shipments)$$

```
ra> \select_{qty < 200}(Shipments);

-----
S1|P4|J1|100
S2|P5|J2|100
S5|P2|J4|100
S5|P5|J7|100
-----
Total number of rows: 4
```

Shipments

snum	pnum	jnum	qty
S1	P1	J1	200
S1	P1	J4	700
S2	P3	J1	400
S2	P3	J2	200
S2	P3	J3	200
S2	P3	J4	500
S2	P3	J5	600
S2	P3	J6	400
S2	P3	J7	800
S2	**P5**	**J2**	**100**
S3	P3	J1	200
S3	P4	J2	500
S4	P6	J3	300
S4	P6	J7	300
S5	P2	J2	200
S5	**P2**	**J4**	**100**
S5	P5	J5	500
S5	**P5**	**J7**	**100**
S1	**P4**	**J1**	**100**
S1	P6	J2	200

Selection

$$\sigma_{color='Blue' OR city='London'}(Parts)$$

```
ra> \select_{color = 'Blue' OR city = 'London'}(Parts);

-----
P1|Nut|Red|12.0|London
P3|Screw|Blue|17.0|Rome
P4|Screw|Red|14.0|London
P5|Cam|Blue|12.0|Paris
P6|Cog|Red|19.0|London
-----
Total number of rows: 5
```

Operator composition

$$\rho_{snum}(\sigma_{status>15}(Suppliers))$$

```
ra> \project_{snum}(\select_{status > 15}(Suppliers));

-----
S1
S3
S4
S5
-----
Total number of rows: 4
```

Suppliers

snum	sname	status	city
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens
S6	Brown	15	Berlin

Operator composition

$$\rho_{pnum,pname}(\sigma_{color='Blue' OR city='London'}(Parts))$$

```
ra> \project_{pnum, pname}(\select_{color = 'Blue' OR city = 'London'}(Parts));

-----
P1|Nut
P3|Screw
P4|Screw
P5|Cam
P6|Cog
-----
Total number of rows: 5
```

Cross product

$R \times S$ Concatenates all tuples in R with all tuples in S

R1

A	B	C
1	2	3
4	2	1
5	6	7

R2

D	E	A
8	2	4
1	5	5
9	1	4

$R1 \times R2$

```
ra> R1 \cross R2;
Output schema: (A integer, B integer, C integer, D integer, E integer, A:1
integer)
-----
1|2|3|1|5|5
1|2|3|8|2|4
1|2|3|9|1|4
4|2|1|1|5|5
4|2|1|8|2|4
4|2|1|9|1|4
5|6|7|1|5|5
5|6|7|8|2|4
5|6|7|9|1|4
-----
Total number of rows: 9
```

Cross product

Suppliers´ Parts

```
ra> Suppliers \cross Parts;

Output schema: (snum text, sname text, status integer, city text, pnum
text, pname text, color text, weight float, city:1 text)
-----
S1|Smith|20|London|P1|Nut|Red|12.0|London
S1|Smith|20|London|P2|Bolt|Green|17.0|Paris
S1|Smith|20|London|P3|Screw|Blue|17.0|Rome
S1|Smith|20|London|P4|Screw|Red|14.0|London
S1|Smith|20|London|P5|Cam|Blue|12.0|Paris
S1|Smith|20|London|P6|Cog|Red|19.0|London
S1|Smith|20|London|P7|Gear|Yellow|18.0|Berlin
S2|Jones|10|Paris|P1|Nut|Red|12.0|London
S2|Jones|10|Paris|P2|Bolt|Green|17.0|Paris
S2|Jones|10|Paris|P3|Screw|Blue|17.0|Rome
S2|Jones|10|Paris|P4|Screw|Red|14.0|London
S2|Jones|10|Paris|P5|Cam|Blue|12.0|Paris

. . .

S6|Brown|15|Berlin|P5|Cam|Blue|12.0|Paris
S6|Brown|15|Berlin|P6|Cog|Red|19.0|London
S6|Brown|15|Berlin|P7|Gear|Yellow|18.0|Berlin
-----
Total number of rows: 42
```

Cross product

Suppliers´ Shipments

```
ra> Suppliers \cross Shipments;

Output schema: (snum text, sname text, status integer, city text, snum:1
text, pnum text, jnum text, qty integer)
-----
S1|Smith|20|London|S1|P1|J1|200
S1|Smith|20|London|S1|P1|J4|700
S1|Smith|20|London|S1|P4|J1|100

. . .

S4|Clark|20|London|S2|P3|J7|800
S4|Clark|20|London|S2|P5|J2|100
S4|Clark|20|London|S3|P3|J1|200
S6|Brown|15|Berlin|S5|P2|J2|200
S6|Brown|15|Berlin|S5|P2|J4|100
S6|Brown|15|Berlin|S5|P5|J5|500
S6|Brown|15|Berlin|S5|P5|J7|100
-----
Total number of rows: 120
```

Cross product

$$\sigma_{snum=snum}(Suppliers \times Shipments)$$

```
\select_{snum = snum}(Suppliers \cross Shipments); // Not in RA
// must rename duplicate attributes; messy but …
-----
S1|Smith|20|London|S1|P1|J1|200
S1|Smith|20|London|S1|P1|J4|700
S1|Smith|20|London|S1|P4|J1|100
S1|Smith|20|London|S1|P6|J2|200
S2|Jones|10|Paris|S2|P3|J1|400
S2|Jones|10|Paris|S2|P3|J2|200
S2|Jones|10|Paris|S2|P3|J3|200
S2|Jones|10|Paris|S2|P3|J4|500
S2|Jones|10|Paris|S2|P3|J5|600
S2|Jones|10|Paris|S2|P3|J6|400
S2|Jones|10|Paris|S2|P3|J7|800
S2|Jones|10|Paris|S2|P5|J2|100
S3|Blake|30|Paris|S3|P3|J1|200
S3|Blake|30|Paris|S3|P4|J2|500
S4|Clark|20|London|S4|P6|J3|300
S4|Clark|20|London|S4|P6|J7|300
S5|Adams|30|Athens|S5|P2|J2|200
S5|Adams|30|Athens|S5|P2|J4|100
S5|Adams|30|Athens|S5|P5|J5|500
S5|Adams|30|Athens|S5|P5|J7|100
-----
Total number of rows: 20
```

Union

$R \cup S$ The tuples in either R or S. R and S must be "union compatible."

Undergrad

sid	sname	gpa
9021	Pat	4.00
9022	Kelly	3.85
9023	Logan	3.25

Grad

sid	sname	gpa
9024	Taylor	3.89
9025	Jessie	3.37

$Undergrad \cup Grad$

```
ra> Undergrad \union Grad;
Output schema: (sid text, sname text, GPA float)
-----
9021|Pat|4.0
9022|Kelly|3.85
9023|Logan|3.25
9024|Taylor|3.89
9025|Jessie|3.37
-----
Total number of rows: 5
```

Union

$$\pi_{pnum}(\sigma_{city='Paris'}(Parts)) \cup \pi_{pnum}(\sigma_{jnum='J2'}(Shipments))$$

This returns the

```
ra>
\project_{pnum}(\select_{city = 'Paris'}(Parts))
     \union
\project_{pnum}(\select_{jnum = 'J2'}(Shipments));

Output schema: (pnum text)
-----
P2
P3
P4
P5
P6
-----
Total number of rows: 5
```

Difference

$R - S$ The tuples in R that are not in S. R and S must be "union compatible."

Undergrad

sid	sname	gpa
9021	Pat	4.00
9022	Kelly	3.85
9023	Logan	3.25

Rotc

sid	sname	gpa
9021	Pat	4.00

$Undergrad - Rotc$

```
ra> Undergrad \diff Rotc;

Output schema: (sid text, sname text, GPA
float)
-----
9022|Kelly|3.85
9023|Logan|3.25
-----
Total number of rows: 2
```

Difference

$$\rho_{pnum}(Parts) \; - \; \rho_{pnum}(\sigma_{jnum='J2'}(Shipments))$$

This returns

```
ra> \project_{pnum}(Parts) \diff \project_{pnum}(\select_{jnum = 'J2'}(Shipments));

Output schema: (pnum text)
-----
P1
P7
-----
Total number of rows: 2
```

RENAME Operation

- In general, for most queries, we need to apply **several** relational algebra operations one after the other.

$$\pi_{\text{Fname, Lname, Salary}}(\sigma_{\text{Dno=5}}(\text{EMPLOYEE}))$$

- Either we can write the operations as a single **relational algebra expression** by nesting the operations, or we can apply one operation at a time and create intermediate result relations.
  - In the latter case, we must give names to the relations that hold the intermediate results.

$$\text{DEP5\_EMPS} \leftarrow \sigma_{\text{Dno=5}}(\text{EMPLOYEE})$$
$$\text{RESULT} \leftarrow \pi_{\text{Fname, Lname, Salary}}(\text{DEP5\_EMPS})$$

- The general **RENAME** operation when applied to a relation *R* of degree *n* is denoted by any of the following three forms:

$$\rho_{S(B1,\, B2,\, \ldots,\, Bn)}(R) \quad \text{or} \quad \rho_{S}(R) \quad \text{or} \quad \rho_{(B1,\, B2,\, \ldots,\, Bn)}(R)$$

symbol ρ (rho) = RENAME operator
S = new relation name
*B*1, *B*2, … , *Bn* are the new attribute names.

Queries in relational algebra

Suppliers database

Query: Retrieve the name and id number of all suppliers in Paris having status greater than 20.

$$p_{sname,snum}(s_{(city='Paris')and(status>20)}(Suppliers))$$

```
ra> \project_{sname, snum}(\select_{city = 'Paris' AND status > 20}(Suppliers));

Output schema: (sname text, snum text)
-----
Blake|S3
-----
Total number of rows: 1
```

Queries in relational algebra

Suppliers database

Query: Retrieve the name of suppliers who supply part P2.

$$\pi_{sname}(\sigma_{pnum='P2'}(\sigma_{snum=snum}(Suppliers \times Shipments)))$$

```
ra>
\project_{sname}(
     \select_{pnum = 'P2'}(
          \select_{snum = sid}(
               Suppliers
               \cross
               \rename_{sid, pnum, jnum, qty} Shipments
          )
     )
);

Output schema: (sname text)
-----
Adams
-----
Total number of rows: 1
```

Queries in relational algebra

Suppliers database

Query: Retrieve the name of suppliers who supply part P2.

$$\pi_{sname}(\sigma_{pnum='P2'}(\sigma_{snum=snum}(Suppliers \times Shipments)))$$

join

$\bowtie$

Since the cross product almost always combines too much, we will typically follow it with a selection. The **join** operator is simply the composition of a product and a selection.

Relational algebra

Additional Operators Convenient but not strictly necessary

Intersection	$R \cap S$	Tuples in both R and S
Theta Join	$R \bowtie_C S$	Combined tuples from R and S that satisfy the condition C
Equijoin	$R \bowtie_C S$	Special case of theta join where C uses only equality
Natural Join	$R \bowtie S$	Join based on equality of shared attributes
Left Outer Join	$R \mathbin{⟕}_\theta S$	Join with non-matching tuples from R
Right Outer Join	$R \mathbin{⟖}_\theta S$	Join with non-matching tuples from S
Full Outer Join	$R \mathbin{⟗}_\theta S$	Join with non-matching tuples from R and S
Division	$R(XY),\ S(Y)$	Tuples in R that match with all relevant tuples in S

Queries in relational algebra

Suppliers database

Query: Retrieve the name of suppliers who supply part P2.

$$\pi_{sname}(\sigma_{pnum='P2'}(Suppliers \bowtie Shipments))$$

```
ra>
\project_{sname}(\select_{pnum = 'P2'}(Suppliers \join Shipments));

Output schema: (sname text)
-----
Adams
-----
Total number of rows: 1
```

Three “inner” joins

Theta Join	$R \bowtie_C S$	Combined tuples from R and S that satisfy the condition C
Equijoin	$R \bowtie_C S$	Special case of theta join where condition uses only equality
Natural Join	$R \bowtie S$	Join based on equality of shared attributes

Natural join

Natural Join $R \bowtie S$ Join based on equality of **all** shared attributes

Shared attributes appear only once in resulting schema

R

A	B	C
1	2	3
4	2	1
5	6	7

S

D	E	A	C
8	2	4	3
1	5	5	2
9	1	4	1

$R \bowtie S$

A	B	C	D	E
4	2	1	9	1

```
ra> R \join S;

Output schema: (A integer, B integer, C integer, D integer, E
integer)
-----
4|2|1|9|1
-----
Total number of rows: 1
```

Natural join

Suppliers database

Query: Retrieve the name of suppliers who supply part P2.

$$\pi_{sname}(\sigma_{pnum='P2'}(Suppliers \bowtie Shipments))$$

The natural join is appropriate since snum is the only shared attribute between Suppliers and Shipments.

```
ra> \project_{sname}(Suppliers \join (\select_{pnum = 'P2'}(Shipments)));

Output schema: (sname text)
-----
Adams
-----
Total number of rows: 1
```

Natural join

Suppliers database

Query: Retrieve the name of suppliers who supply a blue part.

$$\pi_{sname}(\sigma_{color='Blue'}((Parts \bowtie Shipments) \bowtie Suppliers))$$

```
\project_{sname}(\select_{color = 'Blue'}(Parts \join Shipments \join Suppliers));

Output schema: (sname text)
-----
Jones
-----
Total number of rows: 1
```

Incorrect! Adams and Blake are missing.

The natural join is incorrect since city becomes one of the shared attributes.

Equijoin

Equijoin $R \bowtie_C S$

Join based on equality comparisons only

Shared attributes appear twice in resulting schema

R

A	B	C
1	2	3
4	2	1
5	6	7

S

D	E	A	C
8	2	4	3
1	5	5	2
9	1	4	1

$R \bowtie_{R.A=S.A} S$

A	B	C	D	E	A	C
4	2	1	8	2	4	3
4	2	1	9	1	4	1
5	6	7	1	5	5	2

```
ra> R \join_{A = SA} \rename_{D,E,SA,SC} S;
Output schema: (A integer, B integer, C integer, D integer, E
integer, SA integer, SC integer)
-----
4|2|1|8|2|4|3
4|2|1|9|1|4|1
5|6|7|1|5|5|2
-----
Total number of rows: 3
```

Equijoin

Suppliers database

Query: Retrieve the name of suppliers who supply a blue part.

$$\pi_{sname}(\sigma_{color='Blue'}((Parts \bowtie Shipments) \bowtie_{snum=snum} Suppliers))$$

```
ra> \project_{sname}(\select_{color = 'Blue'}(Parts \join Shipments \join_{snum =
sid} \rename_{sid,sname,status,city} Suppliers));

Output schema: (sname text)
-----
Adams
Blake
Jones
-----
Total number of rows: 3
```

The equijoin is correct since it allows us to specify which shared attribute to use.

Theta join

Theta Join $R \bowtie_C S$

Join condition based on any boolean condition.

All attributes of both R and S are in in the resulting schema.

R

A	B	C
1	2	3
4	2	1
5	6	7

S

D	E	A	C
8	2	4	3
1	5	5	2
9	1	4	1

A	B	C	D	E	A	C
5	6	7	8	2	4	3
5	6	7	9	1	4	1

$R \bowtie_{R.A>S.A} S$

```
ra> R \join_{A > SA} \rename_{D,E,SA,SC} S;
Output schema: (A integer, B integer, C integer, D integer, E
integer, SA integer, SC integer)
-----
5|6|7|8|2|4|3
5|6|7|9|1|4|1
-----
Total number of rows: 2
```

Theta join

Suppliers database

Query: Retrieve the supplier numbers of all suppliers with a status greater than the status of S1.

$$\pi_{snum}(\sigma_{snum='S1'}(Suppliers) \bowtie_{status_{right} > status_{left}} (Suppliers)$$

```
\project_{snum} ( \rename_{stat1}(\project_{status}(\select_{snum =
'S1'}(Suppliers))) \join_{status > stat1} Suppliers );

Output schema: (snum text)
-----
S3
S5
-----
Total number of rows: 2
```

Algebraic properties of joins

Inner joins are **associative** and **commutative** operators.

R

A	B	C
1	2	3
4	2	1
5	6	7

S

D	E	A	C
8	2	4	3
1	5	5	2
9	1	4	1

T

F	A
2	3
3	4
4	5

Associative $$R \bowtie S \bowtie T = (R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$$

```
ra> (R \join S) \join T;

Output schema: (A integer, B
integer, C integer, D integer, E
integer, F integer)
-----
4|2|1|9|1|3
-----
Total number of rows: 1
```

```
ra> R \join (S \join T);

Output schema: (A integer, B
integer, C integer, D integer, E
integer, F integer)
-----
4|2|1|9|1|3
-----
Total number of rows: 1
```

Algebraic properties of joins

Inner joins are **associative** and **commutative** operators.

R

A	B	C
1	2	3
4	2	1
5	6	7

S

D	E	A	C
8	2	4	3
1	5	5	2
9	1	4	1

Commutative $R \bowtie S = S \bowtie R$

```
ra> R \join S;

Output schema: (A integer, B
integer, C integer, D integer, E
integer)
-----
4|2|1|9|1
-----
Total number of rows: 1
```

```
ra> S \join R;

Output schema: (D integer, E
integer, A integer, C integer, B
integer)
-----
9|1|4|1|2
-----
Total number of rows: 1
```

Queries in relational algebra

Suppliers database

Query: Retrieve the number, name, and city of all projects. If any project is located in the same city as a supplier, list the supplier number as well.

$$\pi_{jnum,jname,city,snum}(Projects \bowtie_{Projects.city=Suppliers.city} Suppliers)$$

```
ra> \project_{jnum,jname,city1,snum}( (\rename_{jnum,jname,city1}(Projects))
\join_{city1 = city2} (\rename_{snum,sname,status,city2}(Suppliers)) );

Output schema: (jnum text, jname text, city1 text, snum text)
-----
J1|Sorter|Paris|S2
J1|Sorter|Paris|S3
J3|Reader|Athens|S5
J4|Console|Athens|S5
J5|Filler|London|S1
J5|Filler|London|S4
J7|Tape|London|S1
J7|Tape|London|S4
-----
Total number of rows: 8
```

Incorrect – J2 and J6 are missing.

Three “outer” joins

Left Outer Join $R \mathbin{⟕}_\theta S$ Join with non-matching tuples from R

Right Outer Join $R \mathbin{⟖}_\theta S$ Join with non-matching tuples from S

Full Outer Join $R \mathbin{⟗}_\theta S$ Join with non-matching tuples from R and S

R

A	B	C
1	2	3
4	2	1
5	6	7

S

F	A
2	3
3	4
4	5

$R \mathbin{⟕} S$

A	B	C	F	A
1	2	3	NULL	NULL
4	2	1	3	4
5	6	7	4	5

$R \mathbin{⟖} S$

A	B	C	F	A
NULL	NULL	NULL	2	3
4	2	1	3	4
5	6	7	4	5

Queries in relational algebra

Query: Retrieve the number, name, and city of all projects along with the supplier numbers, if any, that are located in the same city as a project.

$$\pi_{jnum,jname,city,snum}(Projects \rtimes_{Projects.city=Suppliers.city} Suppliers)$$

RA doesn't support outer joins, so this output is from PostgreSQL:

```
 jnum |  jname  |  city  | snum
------+---------+--------+------
 J1   | Sorter  | Paris  | S3
 J1   | Sorter  | Paris  | S2
 J2   | Punch   | Rome   |
 J3   | Reader  | Athens | S5
 J4   | Console | Athens | S5
 J5   | Filler  | London | S4
 J5   | Filler  | London | S1
 J6   | Layer   | Oslo   |
 J7   | Tape    | London | S4
 J7   | Tape    | London | S1
(10 rows)
```

Queries in relational algebra

Suppliers database

Query: Retrieve the name and number of all parts that are carried in any of the following colors: red, yellow, green.

$$\pi_{pname,plum}(\sigma_{color='Red'\ OR\ color='Yellow'\ OR\ color='Green'}(Parts))$$

```
ra> \project_{pname, pnum}(\select_{color = 'Red' OR color = 'Yellow' OR color =
'Green'}(Parts));

Output schema: (pname text, pnum text)
-----
Bolt|P2
Cog|P6
Gear|P7
Nut|P1
Screw|P4
-----
Total number of rows: 5
```

Queries in relational algebra

Suppliers database

Query: Retrieve the name and number of all parts that are carried in any of the following colors: red, yellow, green.

$$\pi_{pname,plum}(\sigma_{color='Red'}(Parts) \cup \sigma_{color='Yellow'}(Parts) \cup \sigma_{color='Green'}(Parts))$$

```
ra>
\project_{pname, pnum}(
     \select_{color = 'Red'}(Parts)
          \union
     \select_{color = 'Yellow'}(Parts)
          \union
     \select_{color = 'Green'}(Parts)
);

Output schema: (pname text, pnum text)
-----
Bolt|P2
Cog|P6
Gear|P7
Nut|P1
Screw|P4
-----
Total number of rows: 5
```

Queries in relational algebra

Suppliers database

Query: Retrieve the name and number of all suppliers who supply at least one red part.

$$\pi_{sname,snum}(Suppliers \bowtie Shipments \bowtie (\pi_{pnum}(\sigma_{color='Red'}(Parts))))$$

```
ra> \project_{sname, snum}(Suppliers \join Shipments \join
(\project_{pnum}(\select_{color = 'Red'}(Parts))));

Output schema: (sname text, snum text)
-----
Blake|S3
Clark|S4
Smith|S1
-----
Total number of rows: 3
```

Queries in relational algebra

Suppliers database

Query: Retrieve the name and number of all suppliers who supply either a red part or a blue part.

$$\pi_{sname,snum}(Suppliers \bowtie Shipments \bowtie (\pi_{pnum}(\sigma_{color='Red' \ OR \ color='Blue'}(Parts))));$$

```
ra> \project_{sname, snum}(Suppliers \join Shipments \join
(\project_{pnum}(\select_{color = 'Red' OR color = 'Blue'}(Parts))));

Output schema: (sname text, snum text)
-----
Adams|S5
Blake|S3
Clark|S4
Jones|S2
Smith|S1
-----
Total number of rows: 5
```

Queries in relational algebra

Suppliers database

Query: Retrieve the name and number of all suppliers who supply both a red part and a blue part.

$$\pi_{sname,snum}(Suppliers \bowtie Shipments \bowtie (\pi_{pnum}(\sigma_{color='Red' \ AND \ color='Blue'}(Parts))));$$

A part can only have one color.

```
ra> \project_{sname, snum}(Suppliers \join Shipments \join
(\project_{pnum}(\select_{color = 'Red' AND color = 'Blue'}(Parts))));

Output schema: (sname null, snum null)
-----
-----
Total number of rows: 0
```

Queries in relational algebra

Suppliers database

Query: Retrieve the name and number of all suppliers who supply both a red part and a blue part.

$$\pi_{sname,snum}(Suppliers \bowtie Shipments \bowtie (\pi_{pnum}(\sigma_{color='Red'}(Parts))))$$
$$\cap\ \pi_{sname,snum}(Suppliers \bowtie Shipments \bowtie (\pi_{pnum}(\sigma_{color='Blue'}(Parts))));$$

```
ra> \project_{sname, snum}(
     Suppliers \join Shipments
     \join (\project_{pnum}(\select_{color = 'Red'}(Parts))))

     \intersect

     \project_{sname, snum}(
     Suppliers \join Shipments
     \join (\project_{pnum}(\select_{color = 'Blue'}(Parts))));

Output schema: (sname text, snum text)
-----
Blake|S3
-----
Total number of rows: 1
```

Queries in relational algebra

Suppliers database

Query: Retrieve the name, number, and color of all parts that are supplied by supplier S3.

$$\pi_{pname,pnum,color}(Parts \bowtie (\pi_{pnum}(\sigma_{snum='S3'}(Shipments))));$$

```
ra> \project_{pname, pnum, color}(Parts \join
(\project_{pnum}(\select_{snum='S3'}(Shipments))));

Output schema: (pname text, pnum text, color text)
-----
Screw|P3|Blue
Screw|P4|Red
-----
Total number of rows: 2
```

What if we need the details of the parts which are not supplied by S3? Should we change = to != ?

Queries in relational algebra

Suppliers database

Query: Retrieve the number of all parts that are not supplied by supplier S3.

$$\pi_{pnum}(Parts) - \pi_{pnum}(\sigma_{snum='S3'}(Shipments))$$

```
ra> \project_{pnum}(Parts) \diff \project_{pnum}(\select_{snum = 'S3'}(Shipments));

Output schema: (pnum text)
-----
P1
P2
P5
P6
P7
-----
Total number of rows: 5
```

Queries in relational algebra

Suppliers database

Query: Retrieve the name and number of suppliers who are currently not shipping any parts.

$$\pi_{sname,snum}(Suppliers) - \pi_{sname,snum}(Suppliers \bowtie Shipments)$$

```
ra> \project_{sname, snum}(Suppliers) \diff \project_{sname, snum}(Suppliers
\join Shipments);

Output schema: (sname text, snum text)
-----
Brown|S6
-----
Total number of rows: 1
```

When using difference, be sure that the schemas on both sides of the '-' are similar in structure.

Queries in relational algebra

Suppliers database

This is a "for all" query, and the division operator is specifically intended to be used for these type queries.

Query: Retrieve the name and number of suppliers who are currently shipping all the parts that are carried.

$A(XY) \div B(Y)$ Returns the X values in A that are paired with ALL the Y values in B.

A(snum, pnum) = suppliers and the parts they ship
B(pnum) = all parts carried

A

snum	pnum
S1	P1
S1	P2
S1	P3
S1	P4
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4

B1

pnum
P2

B2

pnum
P2
P4

B3

pnum
P1
P2
P4

A ÷ B1

snum
S1
S2
S3
S4

A ÷ B2

snum
S1
S4

A ÷ B3

snum
S1

Queries in relational algebra

Suppliers database

Query: Retrieve the name and number of suppliers who are currently shipping all the parts that are carried.

This is a "for all" query, and the division operator is specifically intended to be used for these type queries.

Division $A(XY) \div B(Y)$ The X values that are paired in A with all the Y values in B.

Suppliers and the parts they ship

snum	pnum

All parts carried

pnum

Suppliers shipping all parts

snum

$$p_{snum,pnum}(Shipments) \div p_{pnum}(Parts) =$$

The relation returned by division operator will return those tuples from relation A which are associated with every B's tuple.

Queries in relational algebra

Suppliers database

Query: Retrieve the id number of suppliers who are currently shipping all the parts that are carried.

This is a "for all" query, and the division operator is specifically intended to be used for these type queries.

Division $A(XY) \div B(Y)$ The X values that are paired in A with all the Y values in B.

Suppliers and the parts they ship

snum	pnum
S1	P1
S2	P3
S2	P5
S3	P3
S3	P4
S4	P6
S5	P2
S5	P5

÷

All parts carried

pnum
P1
P2
P3
P4
P5
P6
P7

=

Suppliers shipping all parts

snum

None for the example data.

Queries in relational algebra

Suppliers database

Query: Retrieve the id number of suppliers who are currently shipping to all the projects.

$$\pi_{snum,jnum}(Shipments) \div \pi_{jnum}(Projects)$$

Suppliers and the projects they ship to

snum	jnum
S1	J1
S1	J2
S1	J4
S2	J1
S2	J2
S2	J3
S2	J4
S2	J5
S2	J6
S2	J7
S3	J1
S3	J2
S4	J3
S4	J7
S5	J2
S5	J4
S5	J5
S5	J7

÷

All projects

jnum
J1
J2
J3
J4
J5
J6
J7

=

Suppliers shipping to all projects

snum
S2

Division in terms of core operations

Just like multiplication and division are inverse operators in arithmetic, cross product and division are inverse operators in relational algebra.

$$C(XY) \leftarrow A(X) \times B(Y)$$

$$A(X) = C(XY) \div B(Y)$$

$$B(Y) = C(XY) \div A(X)$$

Division in terms of core operations

Given A(XY) and B(Y): $A \div B = \pi_X(A) - \pi_X((\pi_X(A) \times B) - A)$

$$A \leftarrow \pi_{snum,jnum}(Shipments)$$

$$B \leftarrow \pi_{jnum}(Projects)$$

$$A \div B = \pi_{snum}(A) - \pi_{snum}((\pi_{snum}(A) \times B) - A)$$

Division in terms of core operations

Given A(XY) and B(Y): $A \div B = \pi_X(A) - \pi_X((\pi_X(A) \times B) - A)$

$$A \leftarrow \pi_{snum,jnum}(Shipments)$$

$$B \leftarrow \pi_{jnum}(Projects)$$

$$A \div B = \pi_{snum}(A) - \pi_{snum}((\underline{\pi_{snum}(A) \times B}) - A)$$

All possible pairings of snum and jnum

Division in terms of core operations

Given A(XY) and B(Y): $A \div B = \pi_X(A) - \pi_X((\pi_X(A) \times B) - A)$

$$A \leftarrow \pi_{snum,jnum}(Shipments)$$

$$B \leftarrow \pi_{jnum}(Projects)$$

$$A \div B = \pi_{snum}(A) - \pi_{snum}((\pi_{snum}(A) \times B) - A)$$

The pairings of snum and jnum that do not appear in our data

Division in terms of core operations

Given A(XY) and B(Y): $A \div B = \pi_X(A) - \pi_X((\pi_X(A) \times B) - A)$

$$A \leftarrow \pi_{snum,jnum}(Shipments)$$

$$B \leftarrow \pi_{jnum}(Projects)$$

$$A \div B = \pi_{snum}(A) - \pi_{snum}((\pi_{snum}(A) \times B) - A)$$

The suppliers that are not the ones we're looking for

Division in terms of core operations

Given A(XY) and B(Y): $A \div B = \pi_X(A) - \pi_X((\pi_X(A) \times B) - A)$

$$A \leftarrow \pi_{snum,jnum}(Shipments)$$

$$B \leftarrow \pi_{jnum}(Projects)$$

$$A \div B = \pi_{snum}(A) - \pi_{snum}((\pi_{snum}(A) \times B) - A)$$

The suppliers that are left in our data set after we remove the ones we're not looking for

Division in terms of core operations

Given A(XY) and B(Y): $A \div B = \pi_X(A) - \pi_X((\pi_X(A) \times B) - A)$

$$A \leftarrow \pi_{snum,jnum}(Shipments)$$

$$B \leftarrow \pi_{jnum}(Projects)$$

$$A \div B = \pi_{snum}(A) - \pi_{snum}((\pi_{snum}(A) \times B) - A)$$

```
ra> \project_{snum}(\project_{snum,jnum}(Shipments)) \diff \project_{snum}( (
(\project_{snum}(\project_{snum,jnum}(Shipments))) \cross
\project_{jnum}(Projects) ) \diff \project_{snum,jnum}(Shipments) );

Output schema: (snum text)
-----
S2
-----
Total number of rows: 1
```

Queries in relational algebra

Suppliers database

Query: Retrieve the name and number of suppliers who are currently shipping all the red parts that are being shipped.

$$SuppliersShippingParts \leftarrow \pi_{snum,pnum}(Shipments)$$

$$RedPartsBeingShipped \leftarrow \pi_{pnum}((\sigma_{color='Red'}((Parts)) \bowtie Shipments)$$

$$ShippingAllRed \leftarrow SuppliersShippingParts \div RedPartsBeingShipped$$

$$\pi_{sname,snum}(Suppliers \bowtie ShippingAllRed)$$

Now try this on radb