Relational Algebra

Relational Algebra

- The relational algebra consists of a set operations which are either unary or binary, meaning that either one or two relations are operands to the set operations.
- Each of the set operations produces a relation as its output.
- There are five **fundamental operations** in the relational algebra and several additional operations which are defined in terms of the five fundamental operations.
- There is also a rename operation which is sometimes referred to as a fundamental operation, we'll save this one for a little while.

Relational Algebra

- The five fundamental operations are: select, project, union, set difference, and Cartesian product.
- There are several additional (redundant) operations that have been defined in the relational algebra. The most common of these include: intersection, natural join, outer join, semi-join, and division.
- We will examine each operation individually before combining operations into more powerful expressions.

Selection Operator

Type: unary
Symbol: Greek letter sigma, σ
General form: $\sigma_{(\textbf{predicate})}(\text{relation instance})$
Schema of result relation: same as operand relation
Size of result relation (tuples): $\leq$ |operand relation|
Examples: $\sigma_{(\textbf{major = "CS"})}(\text{students})$

$\sigma_{(\textbf{major = "CS"}) \wedge (\textbf{hair-color = "brown"})}(\text{students})$

$\sigma_{(\textbf{hours-attempted > hours-earned})}(\text{students})$

The select operation selects tuples from a relation instance which satisfy a specified predicate.

In general, a predicate may contain any of the logical comparative operators, which are =, $\neq$, <, $\leq$, >, $\geq$. Furthermore, several predicates may be combined using the connectives *and* ($\wedge$), *or* ($\vee$), and *not* ($\neg$).

The select operation may be thought of as providing a horizontal cross-section of the operand relation.

Selection Operator Examples

R

A	B	C	D
a	a	yes	1
b	d	no	7
c	f	yes	34
a	d	no	6
a	c	no	7
b	b	no	69
c	a	yes	24
d	d	yes	47
h	d	yes	34
e	c	no	26
a	a	yes	5

$r = \sigma_{(A = \text{'a'})}(R)$

A	B	C	D
a	a	yes	1
a	d	no	6
a	c	no	7
a	a	yes	5

$r = \sigma_{(A = \text{'a'} \wedge C = \text{"yes"})}(R)$

A	B	C	D
a	a	yes	1
a	a	yes	5

$r = \sigma_{(B = \text{'m'})}(R)$

A	B	C	D

an empty relation

Projection Operator

Type: unary

Symbol: Greek letter pi, π

General form: $\pi_{(\text{attribute-list})}(\text{relation instance})$

Schema of result relation: specified by <attribute-list>

Size of result relation (tuples): $\leq |$ operand relation $|$

Examples: $\pi_{(\text{student-id, name, major})}(\text{students})$

$\pi_{(\text{name, advisor})}(\text{students})$

$\pi_{(\text{name, gpa, hours-attempted})}(\text{students})$

The project operation can be viewed as producing a vertical cross-section of the operand relation.

If the operation produces duplicate tuples, they are typically removed from the result relation in keeping with its set-like characteristics.

Projection Operator Examples

R

A	B	C	D
a	a	yes	1
b	d	no	7
c	f	yes	34
a	d	no	6
a	c	no	7
b	b	no	69
c	a	yes	24
d	d	yes	47
h	d	yes	34
e	c	no	26
a	a	yes	5

$r = \pi_{(A, C)}(R)$

A	C
a	yes
b	no
c	yes
a	no
d	yes
h	yes
e	no

$r = \pi_{(A, D)}(R)$

A	D
a	1
b	7
c	34
a	6
a	7
b	69
c	24
d	47
h	34
e	26
a	5

$r = \pi_{(C)}(R)$

C
yes
no

Union Operator

Type: binary
Symbol: union symbol, $\cup$
General form: $r \cup s$, where *r* and *s* are union compatible
Schema of result relation: schema of operand relations
Size of result relation (tuples): $\leq \max\{|r| + |s|\}$
Examples:

$r \cup s$ $\qquad$ $\pi_{(a, b)}(r) \cup \pi_{(a, b)}(s)$

The union operation provides a means for extracting information which resides in two operand relations which must be union compatible. Union compatibility requires that two conditions hold:

1. Relations r(R) and s(S) in the expression $r \cup s$ must be of the same degree (arity). That is, they must have the same number of attributes.
2. The domains of the i^{th} attribute of r(R) and the i^{th} attributes of s(S) must be the same, for all i.

Union Operator Examples

R

A	B	D
a	a	1
b	d	7
c	f	34
a	d	6
a	c	7

T

A	B
a	a
b	d
c	f
a	d
a	c

r = R ∪ T

not valid – R and T are not union compatible

r = R ∪ S

E	F	G
a	a	1
b	d	7
c	f	34
a	d	6
a	c	7
a	m	4
b	c	22
a	d	16

S

X	Y	Z
a	m	4
b	c	22
a	d	16
a	c	7

X

A	B
a	a
b	d
a	c

r = T ∪ X

A	B
a	a
b	d
c	f
a	d
a	c

Set Difference Operator

Type: binary
Symbol: minus, –
General form: r – s, where *r* and *s* are union compatible
Schema of result relation: schema of operand relation
Size of result relation (tuples): $\leq |$ relation r $|$
Examples: r – s

The set difference operation allows for the extraction of information contained in one relation that is not contained in a second relation.

As with the union operation, the set difference operation requires that the two operand relations be union compatible.

Set Difference Operator Examples

R

A	B	D
a	a	1
b	d	7
c	f	34
a	d	6
a	c	7

T

A	B
a	a
b	d
c	f
a	d
a	c

S

X	Y	Z
a	m	4
b	c	22
a	d	16
a	c	7

X

A	B
a	a
b	d
a	c

r = R – T

not valid – R and T are not union compatible

r = T – X

A	B
c	f
a	d

r = X – T

A	B

empty relation

r = R – S

E	F	G
a	a	1
b	d	7
c	f	34
a	d	6

r = S – R

E	F	G
a	m	4
b	c	22
a	d	16

Cartesian Product Operator

Type: binary
Symbol: ×
General form: **r × s** (no restrictions on *r* and *s*)
Schema of result relation: schema *r* × schema *s* with renaming
Size of result relation (tuples): > | relation *r* | and > | relation *s* |
Examples:

r × s

The Cartesian product operation allows for the combining of any two relations into a single relation.

Recall that a relation is by definition a subset of a Cartesian product of a set of domains, so this gives you some idea of the behavior of the Cartesian product operation.

Cartesian Product Operator Examples

T

A	B
a	a
b	d

X

A	B
a	a
b	d
a	c
c	a

r = T × X

T.A	T.B	X.A	X.B
a	a	a	a
a	a	b	d
a	a	a	c
a	a	c	a
b	d	a	a
b	d	b	d
b	d	a	c
b	d	c	a

Cartesian Product Operator Examples

R

A	B	C	D
a	a	1	yes
b	d	7	yes
c	f	34	no

S

X	Y	Z
a	m	4
b	c	22
a	d	16
a	c	7

r = R × S

A	B	C	D	X	Y	Z
a	a	1	yes	a	m	4
a	a	1	yes	b	c	22
a	a	1	yes	a	d	16
a	a	1	yes	a	c	7
b	d	7	yes	a	m	4
b	d	7	yes	b	c	22
b	d	7	yes	a	d	16
b	d	7	yes	a	c	7
c	f	34	no	a	m	4
c	f	34	no	b	c	22
c	f	34	no	a	d	16
c	f	34	no	a	c	7

Rename Operator

The rename operation is represented by the lowercase Greek letter rho (ρ) and it can be used to rename both relations as well as attributes.

The first common form of the rename operation applies to relations.

General form: $\rho_{\text{new relation name}}(\text{relation})$

Thus, $\rho_x(r)$ renames the relation *r* to relation *x*.

Rename Operator

The second form of the rename operation applies to the renaming of both the relation as well as the attributes of that relation. Assuming that the operand relation is of degree *n*, then the form of this version of the rename operation is:

General form: $\rho_{\text{new relation name (A1, A2, …, AN)}}(\text{relation})$

Thus, $\rho_{\text{x(one, two, …, last)}}(\text{r})$ renames relation *r* to relation *x* and the *n* attributes of relation *x* are names *one, two, …, last*.

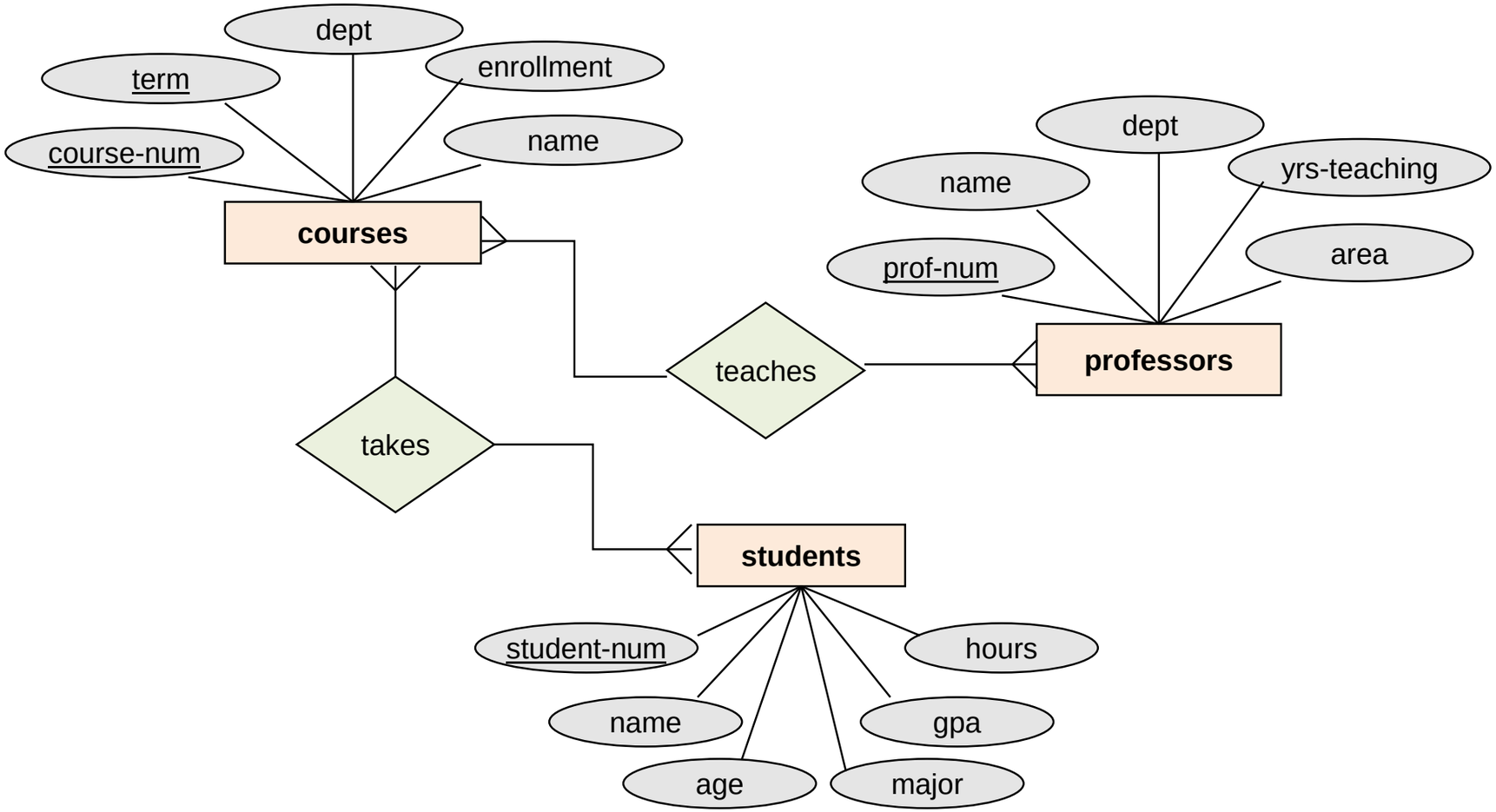
Relational Algebra Expressions

While each of the five fundamental relational algebra operators can be used individually to form a query, their expressive power is tremendously enhanced when they are combined together to form query expressions.

Before we introduce the redundant operations in relational algebra we'll look at forming more complicated combinations of the five fundamental operations. [This will also make you appreciate the redundant operations all the more.]

To form meaningful queries we need to be able to pose them against a database.

Relational Algebra Expressions



Relational Algebra Expressions

After converting an ERD into a set of relational schemas we have the following resulting schemas:

S = STUDENTS (s#, name, age, major, gpa, hours_completed)

C = COURSES (c#, term, name, dept, enrollment)

P = PROFESSORS (p#, name, dept, yrs_teaching, area)

TA = TAKES (s#, c#, term, grade)

TE = TEACH (p#, c#, term)

When you first begin to write queries in a new query language, it is sometimes helpful to actually visualize the data that might be in one of the operand (argument) relations upon which you are operating.

Relational Algebra Expressions

Example Query 1:

Find the names of all the students who are Computer Science majors.

Approach:

First select all of the students who are CS majors.

$$r = \sigma_{(major = \text{“Computer Science”})}(S)$$

Next project only the name attribute from the previous result.

$$result = \pi_{(name)}(r)$$

Complete Query Expression:

$$result = \pi_{(name)}(\sigma_{(major = \text{“Computer Science”})}(S))$$

Relational Algebra Expressions

Example Query 2:

Find the student-num (s#) and name of all the students who have completed more than 90 hours.

Approach:

First select all of the students who have completed more than 90 hours.

$$r = \sigma_{(hours\_completed > 90)}(S)$$

Next project the student-num and name attributes from the previous result.

$$result = \pi_{(s\#, name)}(r)$$

Complete Query Expression:

$$result = \pi_{(s\#, name)}(\sigma_{(hours\_completed > 90)}(S))$$

Relational Algebra Expressions

Example Query 3:

Find the names of all those students who are less than 20 years old who have completed more than 80 hours.

Approach:

First select all of the students who have completed more than 80 hours and are less than 20 years old.

$$r = \sigma_{((hours\_completed > 80) \wedge (age < 20))}(S)$$

Next project the name attribute from the previous result.

$$result = \pi_{(name)}(r)$$

Complete Query Expression:

$$result = \pi_{(name)}(\sigma_{((hours\_completed > 80) \wedge (age < 20))}(S))$$

Relational Algebra Expressions

Example Query 4:

Find the names of all the courses that are offered by either Computer Science or Physics.

Approach:

First select all of the courses that are offered by either CS or Physics.

$$r = \sigma_{((dept = Computer\ Science)\ v\ (dept = Physics))}(C))$$

Next project the name attribute from the previous result.

$$result = \pi_{(name)}(r)$$

Complete Query Expression:

$$result = \pi_{(name)}(\sigma_{((dept = Computer\ Science)\ v\ (dept = Physics))}(C))$$

Relational Algebra Expressions

Example Query 5:

Find the name of every professor who taught a course in the Fall 2015 term.

Approach:

First put the professor information together with the course information.

Next, select only related professors and courses from previous result.

Finally, select only the students name from the previous result.

Complete Query Expression:

$$result = \pi_{(P.name)}(\sigma_{((TE.term = \text{“Fall 2016”}) \text{ AND } (P.p\# = TE.p\#))}(P \times TE))$$

Relational Algebra Expressions

Example Query 6:

Find the names of all the students who took a course in the Spring 2015 term that was taught by a professor who had more than 20 years of teaching experience.

Approach:

First, put the professor information together with the course information together with the teaches information together with the takes information.

Next, select only related students, professors and courses from previous result.

Finally, select only the students name from the previous result.

Complete Query Expression:

$$result = \pi_{(S.name)}(\sigma_{((TA.term = \text{“Spring 2015”}) \wedge (P.yrs\_teaching > 20) \wedge (S.s\# = TA.s\#) \wedge (P.p\# = TE.p\#) \wedge (TA.c\# = TE.c\#) \wedge (TA.term = TE.term))}(S \times P \times TA \times TE))$$

Relational Algebra Expressions

Example Query 7:

Find the names of all the professors who are either in the Computer Science department or have more than 20 years of teaching experience.

Complete Query Expression:

$$result = [\pi_{(name)}(\sigma_{(dept = Computer\ Science)}(P))] \cup [\pi_{(name)}(\sigma_{(yrs\_teaching > 20)}(P))]$$

or:

$$result = \pi_{(name)}(\sigma_{((dept = Computer\ Science)\ v\ (yrs\_teaching > 20))}(P))$$

Relational Algebra Expressions

Example Query 8:

Find the student numbers for those students who were enrolled only in the Spring 2015 term.

Note: The following query expression is not correct for this query!!! Why?

$$result = \pi_{(TA.s\#)}(\sigma_{(TA.term = \text{“Spring 2015”})}(TA))$$

Complete Query Expression:

$$result = [\pi_{(TA.s\#)}(\sigma_{(TA.term = \text{“Spring 2015”})}(TA))] - [\pi_{(TA.s\#)}(\sigma_{(TA.term \neq \text{“Spring 2015”})}(TA))]$$

Redundant Operators in Relational Algebra

- It can be proven that the five fundamental relational operations are sufficient to express any relational-algebra query.
- However, some complex queries will require extremely lengthy and difficult query expressions.
- There have been several extensions of the set of operations available in the relational algebra that provide no additional expressive power, but do provide a simplification in the expression required for more complex queries.
- We'll look at the most important and common of these redundant operations and also show their definition in terms of the five fundamental operations

Intersection Operator

Type: binary
Symbol: $\cap$
General form: **r $\cap$ s** where *r* and *s* are union compatible relations
Schema of result relation: schema of operation relation
Size of result relation (tuples): $\leq |r|$
Definition: **r $\cap$ s $\equiv$ r – (r – s)**
Example:

$$(\pi_{(p\#)}(SPJ)) \cap (\pi_{(p\#)}(P))$$

The intersection operation produces the set of tuples that appear in both operand relations.

Intersection Operator Examples

R

A	B	C	D
a	a	yes	1
b	d	no	7
c	f	yes	34
a	d	no	6

S

E	F	G	H
a	a	yes	1
b	r	yes	3
c	f	yes	34
m	n	no	56

r = R ∩ S

A	B	C	D
a	a	yes	1
c	f	yes	34

r = R ∩ T

A	B	C	D

T

E	F	G	H
a	r	no	31
b	f	yes	30

Join Operator

In query expressions which involved the Cartesian product operator, we had to provide additional selection operations to remove those combinations of tuples that resulted from the Cartesian product which weren't related.

This occurs so commonly that an operation which is a combination of the Cartesian product and selection operations was developed called a join operation.

There are several different join operations which are called, theta-join, equijoin, natural join, outer join, and semi-join.

Theta-Join and Equijoin Operators

Type: binary

Symbol/general form: $r \rhd\lhd_{(predicate)} s$

Schema of result relation: concatenation of operand relations

Definition: $r \rhd\lhd_{(predicate)} s \equiv \sigma_{(predicate)}(r \times s)$

Examples:

$r \rhd\lhd_{(color='blue' \ AND \ size=3)} s$ $\quad$ $r \rhd\lhd_{(color='blue' \ AND \ size>3)} s$

an equijoin

a theta-join

The theta-join operation is a shorthand for a Cartesian product followed by a selection operation.

The equijoin operation is a special case of the theta-join operation that occurs when all of the conditions in the predicate are equality conditions.

Neither a theta-join nor an equijoin operation eliminates extraneous tuples by default. Therefore, the elimination of extraneous tuples must be handled explicitly via the predicate.

Theta-Join Operator Examples

R

A	B	C	D
a	a	yes	1
b	d	no	7
c	f	yes	34
a	d	no	6

S

E	F	G	H
a	a	yes	1
b	r	yes	3
c	f	yes	34
m	n	no	56

$r = R \triangleright\!\triangleleft_{(R.B < S.F)} S$

A	B	C	D	E	F	G	H
a	a	yes	1	b	r	yes	3
a	a	yes	1	c	f	yes	34
a	a	yes	1	m	n	no	56
b	d	no	7	b	r	yes	3
b	d	no	7	c	f	yes	34
b	d	no	7	m	n	no	56
c	f	yes	34	b	r	yes	3
c	f	yes	34	m	n	no	56
a	d	no	6	b	r	yes	3
a	d	no	6	c	f	yes	34
a	d	no	6	m	n	no	56

Natural Join Operator

Type: binary

Symbol/general form: $r * s$

Schema of result relation: concatenation of operand relations with only one occurrence of commonly named attributes

Definition: $r * s \equiv r \rhd\lhd_{(r.commonattributes = s.commonattributes)} s$

Examples:

$$s * spj * p$$

The natural-join operation performs an equijoin over all attributes in the two operand relations which have the same attribute name.

The degree of the result relation of a natural-join is sum of the degrees of the two operand relations subtracted by the number of attributes which are common to both operand relations.

The natural join is extremely useful in the removal of extraneous tuples. Those attributes which are commonly named between the two operand relations are commonly referred to as the join attributes.

Natural Join Operator Examples

R

A	B	C	D
a	a	yes	1
b	r	no	7
c	f	yes	34
a	m	no	6

S

B	M	G	H
a	a	yes	1
b	r	yes	3
a	f	yes	34
m	n	no	56

T

A	B	G	H
a	f	no	31
b	r	yes	30

r = R * S

A	B	C	D	M	G	H
a	a	yes	1	a	yes	1
a	a	yes	1	f	yes	34
a	m	no	6	n	no	56

r = R * T

A	B	C	D	G	H
b	r	no	7	yes	30

Outer Join Operator

Type: binary

Symbol/general form: left-outer-join: $r \supset\!\lhd s$ right-outer-join: $r \rhd\!\subset s$

full outer join: $r \supset\!\lhd\!\rhd\!\subset s$

Schema of result relation: concatenation of operand relations

Definition:

$r \supset\!\lhd s \equiv$ natural join of r and s with tuples from r which do not have a match in s included in the result. Any missing values from s are set to null.

$r \rhd\!\subset s \equiv$ natural join of r and s with tuples from s which do not have a match in r included in the result. Any missing values from r are set to null.

$r \supset\!\lhd\!\rhd\!\subset s \equiv$ natural join of r and s with tuples from both r and s which do not have a match are included in the result. Any missing values are set to null.

Examples: Let r(A,B) = {(a, b), (c, d), (b,c)} , and let

s(A,C) = {(a, d), (s, t), (b, d)}

then $r \supset\!\lhd s$ = (A,B,C) = {(a,b,d), (b,c,d), (c,d,null)},

$r \rhd\!\subset s$ = (A,B,C) = {(a,b,d), (b,c,d), (s,null,t)}, and

$r \supset\!\lhd\!\rhd\!\subset s$ = (A,B,C) = {(a,b,d), (b,c,d), (s,null,t), (c,d,null)},

Outer Join Operator Examples

R

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

S

B	C	D
2	3	10
2	3	11
6	7	12

$r = R \supset\!\triangleleft\triangleright\!\subset S$

A	B	C	D
1	2	3	10
1	2	3	11
4	5	6	null
7	8	9	null
null	6	7	12

$r = R \supset\!\triangleleft S$

A	B	C	D
1	2	3	10
1	2	3	11
4	5	6	null
7	8	9	null

$r = R \triangleright\!\subset S$

B	C	D	A
2	3	10	1
2	3	11	1
6	7	12	null

Semi Join Operator

Type: binary

Symbol/general form: $r \triangleright_{(\text{predicate})} s$

Schema of result relation: schema of r

Definition: $r \triangleright_{(\text{predicate})} s \equiv \pi_{(\text{attributes of r})} \left(r \triangleright\!\triangleleft_{(\text{predicate})} s \right)$

Examples: see next page

The semi-join operation performs a join of the two operand relations and then projects over the attributes of the left-hand operand relation.

The primary advantage of the semi-join operation is that it decreases the number of tuples that need to be handled to form the join. This is particularly useful in a distributed environment.

In its general form, the semi-join is a semi-theta-join. The expected variants of a semi-equijoin and a semi-natural-join are defined in a similar fashion.

Semi Join Operator Examples

R

A	B	C	D
a	a	yes	1
b	r	no	7
c	f	yes	34
a	m	no	6

S

B	M	C
a	e	yes
b	r	yes
a	f	no
r	n	no

$r = R \triangleright_{(R.B > S.M)} S$

A	B	C	D
b	r	no	7
c	f	yes	34
a	m	no	6

Division Operator

Type: binary

Symbol/general form: $r \div s$ where r({A}) and s({B})

Schema of result relation: C where C = A – B

Definition: $r \div s \equiv \pi_{(A-B)}(r) - (\pi_{(A-B)}((\pi_{(A-B)}(r) \times s) - r))$

Examples:

Let r(A,B,C) = {(a,b,c), (a,d,d), (a,b,d), (a,c,c), (a,d,d)}

and s(C) = {(c), (d)} then: $r \div s = t(A,B) = \{(a,b)\}$

<u>Requirements for the division operation:</u>

1. Relation *r* is defined over the attribute set *A* and relation *s* is defined over the attribute set *B* such that $B \subseteq A$.

2. Let *C* be the set of attributes in $A - B$.

Given these constraints the division operation is defined as: a tuple *t* is in $r \div s$ if for every tuple t_s in *s* there is a tuple t_r in *r* which satisfies both:

$t_r[C] = t_s[C]$ and $t_r[A-B] = t[A-B]$

Division Operator Examples

R

A	B	C	D
a	f	yes	1
b	r	no	1
a	f	yes	34
e	g	yes	34
a	m	no	6
b	r	no	34

$r = R \div S$

A	B	C
a	f	yes
b	r	no

$r = R \div T$

A	B
a	f

$r = R \div V$

A
a

$r = R \div W$

A

$r = R \div U$

A	B
b	r

S

D
1
34

T

C	D
yes	1
yes	34

U

C	D
no	1
no	34

V

B	C	D
f	yes	1
f	yes	34
m	no	6

W

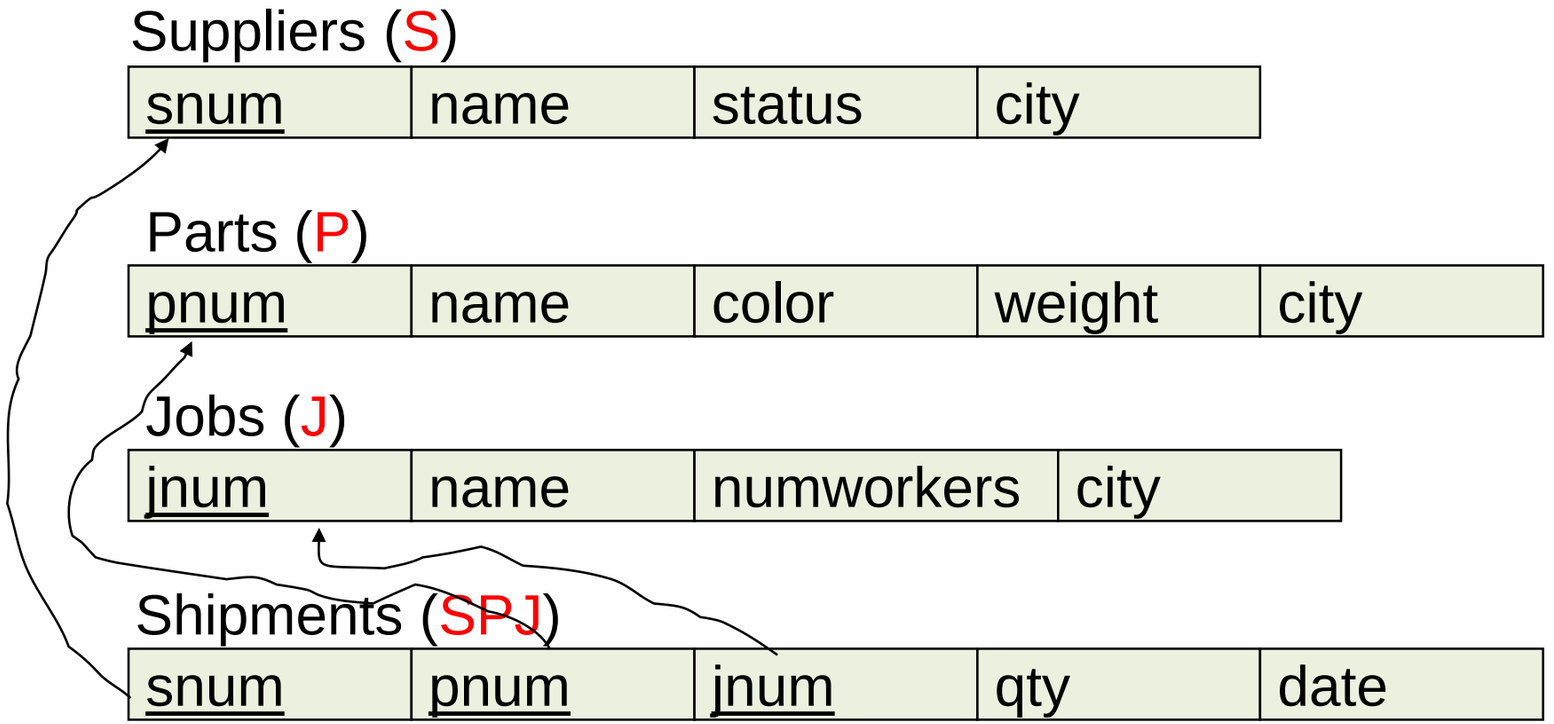
B	C	D
f	yes	1
g	yes	69

Usefulness of the Redundant Operators

The redundant relational algebra operators are redundant because they are all defined in terms of the five fundamental operators.

Their usefulness however, is best illustrated by the division operator.

Sample Database Scheme



Usefulness of the Redundant Operators (cont.)

Query: Find the supplier numbers for those suppliers who ship every part.

Using only the five fundamental operators

T1 = $\pi_{(s\#,\ p\#)}(spj)$ //all (s#,p#) pairs for actual shipments

T2 = $\pi_{(p\#)}(p)$ //all (p#) {all parts that exist, whether shipped or not}

T3 = $\pi_{(s\#)}(T1) \times T2$ //all s# in T1 paired with every tuple in T2 {spj.s#, p.p#}

T4 = T3 – T1 //all tuples in T3 which are not also in T1

T5 = T1 – T4 //all tuples in T1 which are not also in T4.

T6 = $\pi_{(s\#)}(T5)$ //solution

Final solution is: $\pi_{(s\#)}(spj) - (\pi_{(s\#)}((\pi_{(s\#)}(spj) \times p) - spj))$

Using the redundant operators

Alternative solution is: $(\pi_{(s\#,\ p\#)}(spj)) \div (\pi_{(p\#)}(p))$

Some Practice Queries Using 5 Fundamental Operators

1. Find all the supplier numbers for suppliers located in Milan or who ship to any job in a quantity greater than 40.

$$[\pi_{(s\#)}(\sigma_{(city = Milan)}(S))] \cup [\pi_{(s\#)}(\sigma_{(qty > 40)}(SPJ))]$$

2. Find all the supplier numbers for suppliers who ship only red parts.

$$[\pi_{(S.name)}(\sigma_{((SPJ.s\#=S.S\#) \text{ AND } (SPJ.p\#=P.p\#) \text{ AND } (color=red))}(SPJ \times S \times P))]$$

$$- [\pi_{(S.name)}(\sigma_{((SPJ.s\#=S.S\#) \text{ AND } (SPJ.p\#=P.p\#) \text{ AND } (color \neq red))}(SPJ \times S \times P))]$$

Some Practice Queries Using 5 Fundamental Operators

3. Find the supplier names for those suppliers who are located in the same city as a job to which they ship parts.

T1 = (S $\times$ SPJ $\times$ J)

T2 = $\sigma_{(S.s\# = SPJ.s\#)}$(T1) //select tuples which match on s#

T3 = $\sigma_{(J.j\# = SPJ.j\#)}$(T2) //select tuples which match on j#

T4 = $\sigma_{(J.city = S.city)}$(T3) //select tuples from the same city

T5 = $\pi_{(S.name)}$(T4) //project final attribute set

Some Practice Queries Using 5 Fundamental Operators

4. Find all the part numbers which are shipped by both supplier “S1” and supplier “S2”.

NOTE: The following expression in **not** correct! Why not?

$$\pi_{(p\#)}(\sigma_{((s\# = S1) \text{ AND } (s\# = S2))}(SPJ))$$

The following is the correct way of expressing this query in RA.

$$[\pi_{(p\#)}(\sigma_{(s\#=S1)}(SPJ)] - ([\pi_{(p\#)}(\sigma_{(s\#=S1)}(SPJ)] - [\pi_{(p\#)}(\sigma_{(s\#=S2)}(SPJ)])$$

Some Practice Queries Using 5 Fundamental Operators

5. Find the supplier numbers for those suppliers who supply both a red part and a blue part.

   NOTE: The following expression in **not** correct! Why not?

$$\pi_{(s\#)}(\sigma_{((color = blue) \text{ AND } (SPJ.p\# = P.p\#) \text{ AND } (color=red))} (P \times SPJ))$$

The following is the correct way of expressing this query in RA.

$$T1 = \pi_{(s\#)}(\sigma_{((color = blue) \text{ AND } (SPJ.p\# = P.p\#))} (P \times SPJ))$$

$$T2 = \pi_{(s\#)}(\sigma_{((color = red) \text{ AND } (SPJ.p\# = P.p\#))} (P \times SPJ))$$

$$T3 = T2 - T1$$

$$T4 = T2 - T3$$

Some Practice Queries Using 5 Fundamental Operators

6. Find all pairs (s#, j#) such that the supplier and the job are located in the same city, yet that supplier does not have a shipment to that job.

$T1 = \pi_{(s\#,\ j\#)}(\sigma_{(S.city = J.city)}(S \times J))$ //all (s#,j#) pairs in same city

$T2 = \pi_{(s\#,\ j\#)}(\sigma_{((S.city = J.city)\ AND\ (SPJ.j\# = J.j\#)\ AND\ (SPJ.s\# = S.s\#))}(S \times SPJ \times J))$

//T2 contains all (s#,j#) pairs representing shipments by suppliers to jobs in the same city.

T3 = T1 – T2

Practice Queries Using All Relational Algebra Operators (Answers on Next Page)

1. List all pairs of supplier numbers for suppliers who are located in the same city.
2. List every shipment involving a green part.
3. List all the supplier numbers for suppliers who ship a part that is manufactured in the same city in which the supplier is located.
4. List the names of those suppliers who ship all the blue parts.
5. List the supplier numbers for those suppliers who ship only blue parts.

Answers

1. $\pi_{(s.s\#,x.s\#)}(s \rhd\lhd (\rho_x(s)))$
2. $spj \rhd\lhd (\sigma_{(color=green)}(p))$
3. $\pi_{(s.s\#)}(s \rhd\lhd p \rhd\lhd spj)$
4. $\pi_{(s.name)}(s \rhd\lhd (\pi_{(s\#,p\#)}(spj) \div \pi_{(p\#)}(\sigma_{(color=blue)}(p))))$
5. $(\pi_{(s\#)}(spj \rhd\lhd (\sigma_{(color=blue)}(p)))) - (\pi_{(s\#)}(spj \rhd\lhd (\sigma_{(color\neq blue)}(p))))$