Relational Algebra

Thomas Heinis

t.heinis@imperial.ac.uk
Scale Lab - scale.doc.ic.ac.uk

Imperial College
London

The Relational Model

ATTRIBUTES (the columns)
name:type

HEADING

BODY

title:string	year:int	length:int	genre:string
Gone with the Wind	1939	231	Drama
Star Wars	1977	124	SF
Wayne's World	1992	95	Comedy

TUPLES
(the rows)

Movies RELATION

Relations

- Relation = Heading plus Body
- Heading = (**Unordered**) **Set** of Attributes
- Attribute = Attribute Name plus Type (normally simple indivisible types).
- Body = (**Unordered**) **Set** of Tuples
- Tuple = **Set** of attribute values, one for each attribute and of the attribute's type.
- **Schema for Relation** = Name of relation plus heading, for example,

  movies(title:string, year:int, length:int, genre:string)

- Database = One or more Relations
- Schema for Database = Schemas for all relations.

Relations II

In mathematics:

Given sets T_1, T_2, ... T_n

A relation R is a set of tuples (V_1, V_2, ... V_n) where $V_k \in T_k$

R is a subset of T_1 x T_2 ... x T_n.

Tuples are said to be R-related or 'in R'. e.g. Movies-related or in Movies

In the relational model we have *attributed* rather than *ordered* tuples:

R is the set of tuples (A_1:T_1=V_1, ... A_n:T_n=V_n) where $V_k \in T_k$

n is the **degree** of the relation, while the number of tuples is the **cardinality**

Drawing a Relation

Important. In the relational model, the ordering of attributes and tuples is unimportant. We can present relations using whatever permutation of attributes and tuples we like, for example:

Movies

year:int	genre:string	title:string	length:int
1977	SF	Star Wars	124
1992	Comedy	Wayne's World	95
1939	Drama	Gone with the Wind	231

Drawing a relation II

Relations are not 2-dimensional tables. Although this a convenient way of presenting them on paper.

A better way to think of them is as a *set of n-dimensional values*.

For example:

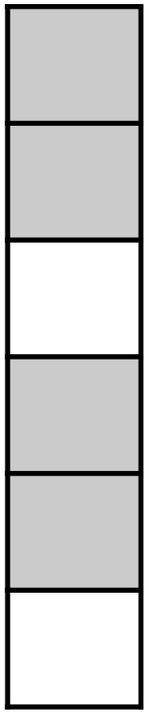
(title:string=StarWars, year:int=1977, length:int=127, genre:string=SF)

is a 4-dimensional movie value.

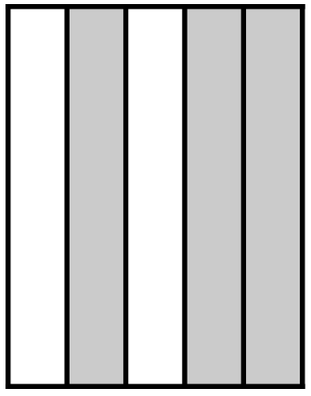
Relational Algebra

Relational expressions are used to construct new relations from other relations. Operators *include*:

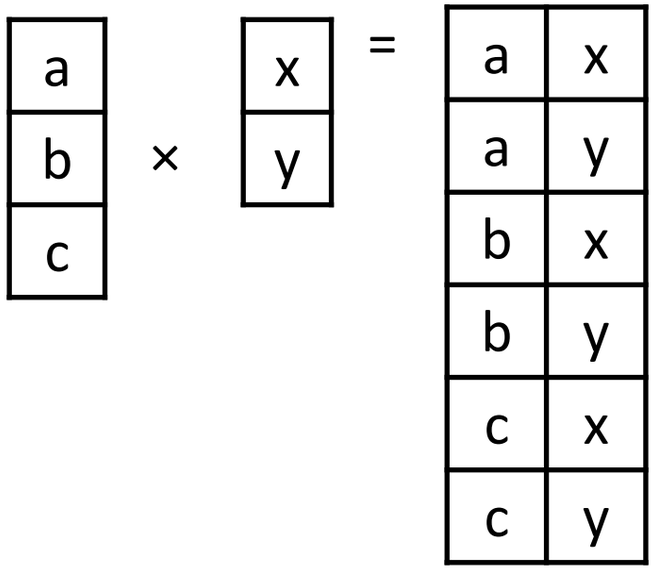
selection



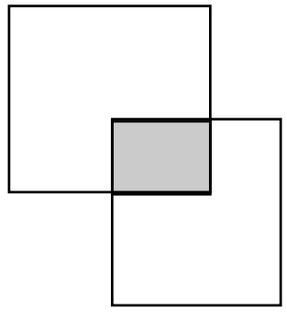
projection



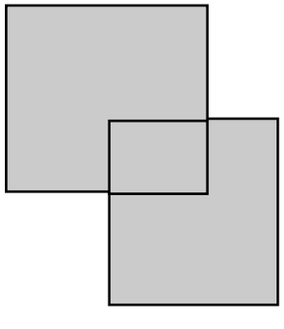
product



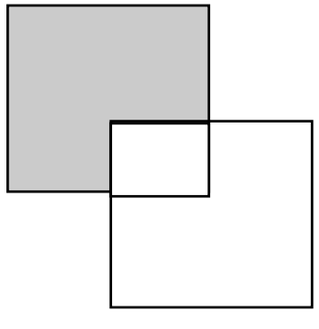
intersection



union



difference



Applying Set Operations on Relations

- Before applying union, intersection, or difference we need to ensure that the attributes used for operands have the same names and types.
- We can use the renaming operator (see later) to change the attribute names of a relation. This is useful where relations are the same except for one or more attribute names. For example, an attribute might be called *Surname* in one relation and *Lastname* in another relation.
- For conciseness in the following slides, we shall restrict ourselves to using integer attributes named *a*, *b*, *c* etc. We'll omit the type in headings.

Union R ∪ S

R

a	b	c
1	2	3
4	5	6
7	8	9

S

a	b	c
7	8	9
2	2	2
3	4	5
1	2	3

R ∪ S

a	b	c

The set of tuples in that are in R **or in** S.

Solution: Union R ∪ S

R

a	b	c
1	2	3
4	5	6
7	8	9

S

a	b	c
7	8	9
2	2	2
3	4	5
1	2	3

R ∪ S

a	b	c
1	2	3
4	5	6
7	8	9
2	2	2
3	4	5

The set of tuples in that are in R **or in** S.

Difference R – S

R

a	b	c
1	2	3
4	5	6
7	8	9
2	2	2
3	4	8

S

a	b	c
7	8	9
2	2	2
3	4	5
1	2	3

R – S

a	b	c

The set of tuples that are in R **but not in** S

Solution: Difference R – S

R

a	b	c
1	2	3
4	5	6
7	8	9
2	2	2
3	4	8

S

a	b	c
7	8	9
2	2	2
3	4	5
1	2	3

R – S

a	b	c
4	5	6
3	4	8

The set of tuples in that are in R **but not in** S

R – S ≢ S – R

R

a	b	c
1	2	3
4	5	6
7	8	9
2	2	2
3	4	8

S

a	b	c
7	8	9
2	2	2
3	4	5
1	2	3

R – S

a	b	c
4	5	6
3	4	8

S – R

a	b	c

R – S is not equivalent to S – R

Solution: R – S ≢ S – R

R

a	b	c
1	2	3
4	5	6
7	8	9
2	2	2
3	4	8

S

a	b	c
7	8	9
2	2	2
3	4	5
1	2	3

R – S

a	b	c
4	5	6
3	4	8

S – R

a	b	c
3	4	5

R – S is not equivalent to S – R

Intersection R ∩ S

R

a	b	c
1	2	3
4	5	6
7	8	9

S

a	b	c
7	8	9
2	2	2
3	4	5
1	2	3

R ∩ S

a	b	c

The set of tuples in that are in R **and in** S

Solution: Intersection R ∩ S

Intersection can be formulated using difference: R ∩ S = R – (R – S)

R

a	b	c
1	2	3
4	5	6
7	8	9

S

a	b	c
7	8	9
2	2	2
3	4	5
1	2	3

R ∩ S

a	b	c
1	2	3
7	8	9

The set of tuples in that are in R **and in** S

Projection $\pi_{attributes}(R)$

Note: there are no duplicate tuples in the result.

R

a	b	c	d	e	f
1	2	3	4	5	6
1	1	1	1	1	1
2	2	2	2	2	2
1	2	3	4	5	8

$\pi_{a,c,e}(R)$

a	c	e

Projection returns relation with listed attributes only.

Solution: Projection $\pi_{attributes}(R)$

Note: there are no duplicate tuples in the result.

R

a	b	c	d	e	f
1	2	3	4	5	6
1	1	1	1	1	1
2	2	2	2	2	2
1	2	3	4	5	8

$\pi_{a,c,e}(R)$

a	c	e
1	3	5
1	1	1
2	2	2

Projection returns relation with listed attributes only.

Selection $\sigma_{condition}(R)$

R

a	b	c
1	2	3
4	5	6
7	8	9
1	5	3
2	9	1
5	8	9
2	6	2
3	4	5

$\sigma_{a>3 \mid b<5}(R)$

a	b	c

Selection returns relation with all tuples that satisfy *condition*.

σ is lowercase sigma (greek alphabet)

Solution: Selection $\sigma_{condition}(R)$

R

a	b	c
1	2	3
4	5	6
7	8	9
1	5	3
2	9	1
5	8	9
2	6	2
3	4	5

$\sigma_{a>3 \mid b<5}(R)$

a	b	c
1	2	3
4	5	6
7	8	9
5	8	9
3	4	5

Selection returns relation with all tuples that satisfy *condition*.

σ is lowercase sigma (greek alphabet)

Cartesian Product R × S

Schema for the result includes the schemas for both relations. Also known as cross-product.

R

a	b
1	2
3	4

S

c	d	e
1	2	3
4	5	6
7	8	9

R × S

a	b	c	d	e

Resulting relation is the set of **all** tuples that can be paired by including a tuple from R and a tuple from S.

Solution: Cartesian Product R × S

Schema for the result includes the schemas for both relations. Also known as cross-product.

R

a	b
1	2
3	4

S

c	d	e
1	2	3
4	5	6
7	8	9

R × S

a	b	c	d	e
1	2	1	2	3
1	2	4	5	6
1	2	7	8	9
3	4	1	2	3
3	4	4	5	6
3	4	7	8	9

Resulting relation is the set of **all** tuples that can be paired by including a tuple from R and a tuple from S.

Cartesian Product R × S

If a R and S have attribute names in common, then we distinguish them by prefixing the attribute name with the name of the relation.

R

a	b
1	2
3	4

S

b	c	d
1	2	3
4	5	6
7	8	9

R × S

a	R.b	S.b	c	d

Solution: Cartesian Product R × S

If a R and S have attribute names in common, then we distinguish them by prefixing the attribute name with the name of the relation.

R

a	b
1	2
3	4

S

b	c	d
1	2	3
4	5	6
7	8	9

R × S

a	R.b	S.b	c	d
1	2	1	2	3
1	2	4	5	6
1	2	7	8	9
3	4	1	2	3
3	4	4	5	6
3	4	7	8	9

Cartesian Product $R \times S = S \times R$

R × S

a	b	c	d	e
1	2	1	2	3
1	2	4	5	6
1	2	7	8	9
3	4	1	2	3
3	4	4	5	6
3	4	7	8	9

S × R

c	d	e	a	b
1	2	3	1	2
1	2	3	3	4
4	5	6	1	2
4	5	6	3	4
7	8	9	1	2
7	8	9	3	4

Solution: Cartesian Product $R \times S = S \times R$

R × S

a	b	c	d	e
1	2	1	2	3
1	2	4	5	6
1	2	7	8	9
3	4	1	2	3
3	4	4	5	6
3	4	7	8	9

S × R

c	d	e	a	b
1	2	3	1	2
1	2	3	3	4
4	5	6	1	2
4	5	6	3	4
7	8	9	1	2
7	8	9	3	4

Natural Join R ⋈ S

Resulting schema is the set of both schemas.

R

a	b
1	2
1	3
3	4

S

b	c	d
1	2	3
2	5	6
4	8	9
4	9	9

R ⋈ S

a	b	c	d

Resulting relation has all tuples that can be “joined” using all matching attributes of R and S.

Solution: Natural Join $R \bowtie S$

Resulting schema is the set of both schemas.

R

a	b
1	2
1	3
3	4

S

b	c	d
1	2	3
2	5	6
4	8	9
4	9	9

$R \bowtie S$

a	b	c	d
1	2	5	6
3	4	8	9
3	4	9	9

Resulting relation has all tuples that can be "joined" using all matching attributes of R and S.

Natural Join R ⋈ S

Must only pair tuples if values of each common attribute are equal.

R

a	b	c
1	2	3
6	7	8
9	7	8
4	2	7
1	7	8

S

b	c	d
2	3	3
2	3	6
4	8	9
7	8	3

R ⋈ S

a	b	c	d

Solution: Natural Join R ⋈ S

Must only pair tuples if values of each common attribute are equal.

R

a	b	c
1	2	3
6	7	8
9	7	8
4	2	7
1	7	8

S

b	c	d
2	3	3
2	3	6
4	8	9
7	8	3

R ⋈ S

a	b	c	d
1	2	3	3
1	2	3	6
6	7	8	3
9	7	8	3
1	7	8	3

Natural Join

Natural join can be expressed in terms of product, selection and projection:

$$R \bowtie S = \pi_L (\sigma_C(R \times S))$$

where L is union of all attributes of R and S, i.e., attr(R) $\cup$ attr(S)

C is $R.a_1=S.a_1$ & $R.a_2 = S.a_2$ & ... & $R.a_n = S.a_n$

where a_k are the attributes common to R and S,

$a_k \in$ attr(R) $\cap$ attr(S)

Natural join is associative: $(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$

If there is no common attribute then: $R \bowtie S = R \times S$, i.e., we don't apply selection and projection.

More Commonly Used Join $R \bowtie_{R.c=S.b} S$

Must only pair tuples if values of each common attribute are equal.

R

a	b	c
1	2	2
6	7	8
9	2	2
4	2	7
1	3	8

S

b	c	d
2	2	3
2	3	6
4	8	9
7	8	3

$R \bowtie_{R.c=S.b} S$

a	R.b	R.c	S.b	S.c	d
1	2	2	2	2	3
1	2	2	2	3	6
9	2	2	2	2	3
9	2	2	2	3	6
4	2	7	7	8	3

Left Outer Join R ⟕ R.c=S.b S

R

a	b	c
1	2	2
6	7	8
9	2	2
4	2	7
1	3	8

S

b	c	d
2	2	3
2	3	6
4	8	9
7	8	3

R ⟕ R.c=S.b S

a	b	R.c	S.b	c	d
1	2	2	2	2	3
1	2	2	2	3	6
9	2	2	2	2	3
9	2	2	2	3	6
4	2	7	7	8	3
6	7	8			
1	3	8			

Right Outer Join R $\bowtie_{R.c=S.b}$ S

R

a	b	c
1	2	2
6	7	8
9	2	2
4	2	7
1	3	8

S

b	c	d
2	2	3
2	3	6
4	8	9
7	8	3

R $\bowtie_{R.c=S.b}$ S

a	b	R.c	S.b	c	d
1	2	2	2	2	3
1	2	2	2	3	6
9	2	2	2	2	3
9	2	2	2	3	6
4	2	7	7	8	3
			4	8	9

Full Outer Join R ⟗ R.c=S.b S

R

a	b	c
1	2	2
6	7	8
9	2	2
4	2	7
1	3	8

S

b	c	d
2	2	3
2	3	6
4	8	9
7	8	3

R ⟗R.c=S.b S

a	b	R.c	S.b	c	d
1	2	2	2	2	3
1	2	2	2	3	6
9	2	2	2	2	3
9	2	2	2	3	6
4	2	7	7	8	3
6	7	8			
1	3	8			
			4	8	9

Renaming $\rho_{new_1/old_1, new_2/old_2, \ldots, new_n/old_n}(R)$

R

a	b	c	d
1	2	5	6
3	4	8	9
3	4	9	9

$\rho_{x/a, y/d}(R)$

1	2	5	6
3	4	8	9
3	4	9	9

Changes the names of one or more attributes.

ρ is lowercase rho

Note: the values and types of the relation are not affected.

Solution: Renaming $\rho_{new_1/old_1, new_2/old_2, \ldots, new_n/old_n}(R)$

R

a	b	c	d
1	2	5	6
3	4	8	9
3	4	9	9

$\rho_{x/a, y/d}(R)$

x	b	c	y
1	2	5	6
3	4	8	9
3	4	9	9

Changes the names of one or more attributes.

Note: the values and types of the relation are not affected.

Relational Expressions

Relational expressions are used to construct new relations from other relations for:

- **Retrieval** - to define data to be retrieved.
- **Updates** - to define data to be inserted, changed, or deleted.
- **Constraints** - to define constraints that the database must satisfy.
- **Derived relations (views)** - to define one relation in terms of others, without constructing it.
- **Concurrency** - to define the data to use for concurrent actions.
- **Access control** - to define the data over which a permission should be granted.
- A relational database language is said to be **relationally complete** if it is at least as expressive as Relational Algebra.

Exercise 1

Consider the database schema (types omitted):

employee(personname, street, city)
worksfor(personname, companyname, salary)
company(companyname, city)

Give relational expressions for each of the following:

1. Names of all employees who work for Barclays.
2. Names of Barclays employees and the city where they live.
3. Name, street and city of all Barclays employees who earn more than £1M.
4. What will $\pi_{personname}(employee \bowtie worksfor \bowtie company)$ return?

Solutions 1

Solution

employee(personname, street, city)
worksfor(personname, companyname, salary)
company(companyname, city)

1. Names of all employees who work for Barclays.

$\pi_{personname}(\sigma_{companyname=\text{“Barclays”}}(worksfor))$

Solution

employee(personname, street, city)
worksfor(personname, companyname, salary)
company(companyname, city)

2. Names of Barclays employees and the city where they live.

$\pi_{\text{personname, city}}(\sigma_{\text{companyname}=\text{“Barclays”}}(\text{workfor} \bowtie \text{employee}))$

Solution

employee(personname, street, city)
worksfor(personname, companyname, salary)
company(companyname, city)

3. Name, street and city of all Barclays employees who earn more than £1M.

$\pi_{\text{personname, street, city}}$ (

$\sigma_{\text{companyname=“Barclays” \& salary>1000000}}$

(worksfor ⋈ employee))

Solution

employee(personname, street, city)
worksfor(personname, companyname, salary)
company(companyname, city)

What will

$\pi_{personname}$ (employee ⋈ worksfor ⋈ company)

return?

Names of all employees who live in the same city as the company they work for.

Exercise 2

Consider the database schema (types omitted):

supplier(sname, address)
part(pname, colour)
catalog(sname, pname, cost)

Give relational expressions for each of the following:

1. Names of all suppliers who supply a red part.
2. Names of all suppliers who supply a red part or are at 18 Queensgate
3. Names of parts supplied by at least two different suppliers.
   Assume: C1 and C2 are temporary copies of the catalog relation.
4. Pairs of suppliers such that the first suppliers charges more for some part than the second supplier. Assume: C1 and C2 are temporary copies of the catalog relation.

Solution

supplier(sname, address)
part(pname, colour)
catalog(sname, pname, cost)

1. Names of all suppliers who supply a red part

$\pi_{sname}(\sigma_{colour= \text{“red”}}(part \bowtie catalog))$

Solution

supplier(sname, address)
part(pname, colour)
catalog(sname, pname, cost)

2. Names of all suppliers “who supply a red part or are at 180 Queensgate

$$\pi_{sname}(\sigma_{colour=\text{“red”}}(part \bowtie catalog)) \cup$$

$$\pi_{sname}(\sigma_{address=\text{ “180 Queensgate”}}(supplier))$$

$$\pi_{sname}(\sigma_{colour=\text{“red”} \mid address=\text{ “180 Queensgate”}}(part \bowtie catalog \bowtie supplier))$$

Note: will not return suppliers at 180 Queensgate that have no parts in the catalog.

Solution

supplier(sname, address)
part(pname, colour)
catalog(sname, pname, cost)

3. Names of parts supplied by at least two different suppliers.
Assume: C1 and C2 are copies of the catalog relation.

$\pi_{C1.pname}(\sigma_{C1.pname=C2.pname \,\&\, C1.sname \neq C2.sname}(C1 \times C2))$

Solution

supplier(sname, address)
part(pname, colour)
catalog(sname, pname, cost)

4. Pairs of suppliers such that the first supplier charges more for some part than the second supplier. Assume: C1 and C2 are temporary copies of the catalog relation.

$$\pi_{C1.sname,\ C2.sname}(\sigma_{C1.pname=C2.pname\ \&\ C1.sname\neq C2.sname\ \&\ C1.cost>C2.cost}(C1 \times C2))$$