Relational Algebra and SQL
2.4 and 6.1.

Recall:

Relational Algebra (RA)
- Operations on Relations.

Projection

$\Pi_{\langle List\ Expr \rangle} R$

list of expressions on the attributes of a relation.

Ex. $R(a,b)$

a	b
1	9
3	3

① $\Pi_a R$

a
1
3

② $\Pi_{a+5,\ -b} R$

a+5	-b
6	-9
8	-3

name of attributes

③ $\Pi_{b,a} R$

b	a
9	1
3	3

④ $\Pi_{-1,a} R$

"-1",	a
-1	1
-1	3

SQL:

select <list expr> from R

① SELECT a FROM R

② SELECT a+5, -b FROM R

③ SELECT b, a FROM R

④ SELECT -1, a FROM R

Name of Relation optional!!

SELECT 3;

3
3

Creates table of one tuple!!

SELECT 'abc', 5.2

=>

abc	5.2
'abc'	5.2

name of attributes.

Tuples.

The result of SELECT is always a relation

Renaming Relations and their attributes.

Sometimes we need to rename tables or their attributes.

$$\rho_{<\text{new schema}>} R$$

Ex: $R(a,b)$ $\quad \rho_{S(c,d)} R$ renames $R(a,b)$ to $S(c,d)$

dmg notation: you can rename during the projection.

If we want to rename the projected expression we can do it:

$$\Pi_{a \to c,\ b \to d}^{R \to S}$$

Result schema $S(c,d)$

Ex:

① $\Pi_{a+5 \to x,\ -b \to y}\ R$.

x	y
6	-9
8	-3

SQL.

Given $R(a,b)$ $\rho_{S(c,d)}\ R$

SELECT a, b FROM R as S(c, d)

or

SELECT a as c, b as d FROM R

①

SELECT a + 5 AS X, -b AS Y FROM R

SELECTION

$$\sigma_p R$$

p is a predicate on attributes of R

Expressions:

$<, >, <>, =, >=, <=$

<> : different, = : equal

AND, NOT and many others.

Ex:

R(a,b)

a	b
3	2
1	∅

p evaluated at each tuple.

(1) $\sigma_{a>1 \text{ OR } b>1} R$

a	b
3	2

SQL.

SELECT * FROM R WHERE p

* : original attributes of R

Ex:

① SELECT * FROM R
WHERE a > 1 OR b > 1

We can combine Π and σ:

Ex: $\Pi_a \sigma_{a>1 \text{ OR } b>1} R$.

SELECT a FROM R
WHERE a > 1 OR b > 1

NOT equivalent to.

$\sigma_{a>1 \text{ OR } b>1} \Pi_a R$.

b is not part of $\Pi_a R$.

Questions

What does this return?

1) $\sigma_{\text{FALSE}} R$

2) $\sigma_{\text{TRUE}} R$

Other expressions in predicates.

IN

att IN (List)

Ex.

a IN (3, 2, 5)

⇒ equivalent to (a = 3 or a = 2 or a = 5)

But we can also use a query:

a in (Π_c S)

SQL:

a IN (SELECT c FROM S)

EXISTS

EXISTS (R) true if R not empty

Ex:

EXISTS ($\sigma_{a>5}$ R)

Operations on 2 Relations.

Union	$\cup$
Intersection	$\cap$
Difference (Except)	$-$

Union Compatible

R and S are "union compatible" iff

$$|attrs(R)| = |attrs(S)|$$

and the type of the i-th attribute of S. is type compatible with the type of the i-th attribute of R.

One type t_1 is type compatible with type t_2 if t_1 can be converted to type t_2.

$A \cup B$
$A \cap B$
$A - B$

} Defined only iff A & B are union compatible.

UNION

$t \in R \cup S \iff t \in R$ or $t \in S$

$t \in R \cap S \iff t \in R$ and $t \in S$

$t \in R - S \iff t \in R$ and $t \notin S$

Schema of result is schema of first relation.

Ex:

$R(a,b)$

a	b
1	a
3	x

$S(c,d)$

c	d
1	e
3	x
4	f

$R \cup S$

a	b
1	a
3	x
1	e
4	f

$R \cap S$

a	b
3	x

$R - S$

a	b
1	a

$S - R$

c	d
1	e
4	f

SQL

TABLE R { UNION / INTERSECT / EXCEPT } TABLE S

or

<QUERY1> { UNION / INTERSECT / EXCEPT } <QUERY2>

```
SELECT a, b FROM R
UNION
SELECT c, d FROM S;
```

NULLs (6.1)

SQL has a special value: NULL.

⇒ unknown.

Example:

- Next year champion of the Stanley Cup.
- Grades of students currently enrolled in this course.

- SQL has special considerations for expressions involving NULL
- SQL Logic 3 valued:
  - True
  - False
  - Unknown

- Any expression involving NULL results into UNKNOWN

IMPORTANT

X = NULL
X > NULL
} ⇒ UNKNOWN.

To test if attr is NULL use

X IS NULL

Ex:

NULL > 5 ⇒ UNKNOWN

x is NULL ⇒ True if x contains NULL

UNKNOW is NOT TRUE

Ex:

UNKNOWN OR TRUE ⇒ TRUE

UNKNOWN AND TRUE ⇒ FALSE

See exercise!!

Text Matching.

Regular expressions (Postgres)

expr ~ RegExp

Ex

a ~ '^ab'

attribute a starts with string ab

a ~ '\.txt$'

attribute a end with string .txt

Cross product: $\times$

Given relations R and S:

- $(r, t) \in R \times S$ iff $r \in R$ and $s \in S$

R(a, b)

a	b
1	x
2	y

S(c, d)

c	d
5	8
2	12

$T = R \times S$

a	b	c	d
1	x	5	8
1	x	2	12
2	y	5	8
2	y	2	12

What is schema of T?

Natural Join $\bowtie$

Given relations R and S

C is set of attributes of both S and R with the same name

- if C is empty.

$$R \bowtie S = R \times S$$

- otherwise

$$\Pi_{Att(R), attr(S)-C} \left(\sigma_{\bigwedge_{a_i \in C} R.a_i = S.a_i} (R \times S) \right)$$

$\Pi_{Att(R), attr(S)-C}$: Do not project both common attributes (only the first).

$\sigma_{\bigwedge_{a_i \in C} R.a_i = S.a_i}$: match tuples with same value in common attributes; conjunction over all common attributes

Ex:

R(a, b)

a	b
1	x
2	y

S(a, c)

a	c
5	8
2	12

Common attributes = {a}

$$T = R \bowtie S = \Pi_{a,b,c}\, \sigma_{R.a = S.a}(R \times S)$$

R × S

R.a	R.b	S.a	S.c
1	x	5	8
1	x	2	12
2	y	5	8
2	y	2	12

} R.a = S.a

R ⋈ S

a	b	c
2	y	12

Theta Join

$$R \underset{p}{\bowtie} S = \sigma_p (R \times S)$$

```
SELECT * FROM
    R JOIN S ON (p);
```

Cross Product $\times$

$R \times S$

SQL

```
SELECT * FROM R, S;
```

NATURAL JOIN

$R \bowtie S$

SQL.

```
SELECT * FROM R NATURAL JOIN S
```

Theta Join

$$R \underset{p}{\bowtie} S = \sigma_p (R \times S)$$

SQL:

```
SELECT * FROM
    R JOIN S ON (p);
```

FULL | NATURAL JOIN $R \overset{\circ}{\bowtie} S$

THETA JOIN $R \overset{\circ}{\underset{p}{\bowtie}} S$

- Compute non-full join
- Add tuples in R not in join padded with NULL
- Add tuples in S not in join padded with NULL

$\therefore$

$R(a,b)$

a	b
3	x
1	y

$S(a,c)$

a	c
2	3.1
1	2.5

$R \overset{\circ}{\bowtie} S$

a	b	c
1	y	2.5
3	x	$\perp$
2	$\perp$	3.1

← Represents NULL in RA

SELECT * FROM R NATURAL FULL JOIN S

$R \overset{\circ}{\underset{R.a<S.a}{\bowtie}} S$

R.a	b	S.a	c
1	y	2	3.1
3	x	$\perp$	$\perp$
$\perp$	$\perp$	1	2.5

SELECT * FROM R FULL JOIN S ON (R.a > S.a)

· { LEFT / RIGHT } JOIN $R \overset{\circ}{\bowtie}_L S$ $R \overset{\circ}{\bowtie}_R S$

- Compute non-full join
- Add tuples in LEFT or RIGHT relation, padding other attributes with NULL

Example:

∴

$R(a,b)$

a	b
3	x
1	y

$S(a,c)$

a	c
2	3.1
1	2.5

$R \overset{\circ}{\bowtie}_L S$

a	b	c
1	y	2.5
3	x	⊥

SELECT * FROM

R NATURAL LEFT JOIN S;

$R \overset{\circ}{\bowtie}_{R} S$

a	b	c
1	y	2.5
2	⊥	3.1

```
SELECT * FROM
  R NATURAL RIGHT JOIN S;
```

$R \overset{\circ}{\bowtie}_{L\ R.a<S.a} S$

R.a	R.b	S.a	S.b
1	y	2	3.1
3	x	⊥	⊥

```
SELECT * FROM
  R LEFT JOIN S ON
    (R.a < S.a);
```

JOIN USING

SQL provides a special variant of NATURAL JOIN in which we can specify the attribute to join by. Ex.

$R(a, b, c)$ and $S(a, b, d)$

The schema of $R \bowtie S$ is (a, b, c, d)

We can specify a join only on a as follow:

```
SELECT * FROM
R JOIN S USING a.
```

The schema is $(a, R.b, S.b, c, d)$

We write it as:

$$R \bowtie_a S.$$

In practice it is a good idea to write NATURAL JOINS as JOIN USING.

Eg. $R \bowtie S$

```
SELECT * FROM
R NATURAL JOIN S
```

The common attributes are a, b.

It is better to write it as

```
SELECT * FROM
R JOIN S USING (a,b)
```

$R \bowtie_{a,b} S$

↑ my notation

For these relations

$$R \bowtie S = R \bowtie_{a,b} S$$