>>

RA Join Operations

- RA Operations to Combine Relations
- Product
- Natural Join
- Theta Join
- Outer Join
- Division

/\ >>

❖ RA Operations to Combine Relations

Relational algebra has several ways to combine info from relations

- product ... $R \times S$ ... select * from R join S on (true)
- natural join ... $R \bowtie S$ ... select * from R natural join S
- (inner) join ... $R \bowtie_C S$ ... select * from R join S on (C)
- outer join ... $R \overset{\bowtie}{}_C S$ ... select * from R left outer join S on (C)
- division ... R / S ... see SQL slides for examples

Join conditions involve related attributes from the two relations

Frequently, primary-key joined with foreign-key

<< ^ >>

❖ Product

Product (Cartesian product) combines information from two relations pairwise on tuples.

$r \times s = \{ (t_1 : t_2) \mid t_1 \in r \wedge t_2 \in s \}$, where $r(R), s(S)$

Each tuple in the result contains all attributes from r and s, possibly with some fields renamed to avoid ambiguity.

If $t_1 = (A_1 ... A_n)$ and $t_2 = (B_1 ... B_n)$ then $(t_1 : t_2) = (A_1 ... A_n, B_1 ... B_n)$

Result size is large: $|r \times s| = |r| . |s|$ Schema: $R \cup S$

Algorithmic view:

```
result = {}
for each tuple t1 in relation r
    for each tuple t2 in relation s
        result = result ∪ {(t1:t2)} }
```

COMP3311 21T1 ♢ RA Join Operations ♢ [2/13]

❖ Product (cont)

Example of product:

R

A	B	C
a	m	x
e	k	y
f	p	z

S

X	Y
1	1
2	2

R x S

A	B	C	X	Y
a	m	x	1	1
a	m	x	2	2
e	k	y	1	1
e	k	y	2	2
f	p	z	1	1
f	p	z	2	2

<< ^ >>

❖ Natural Join

Natural join is a specialised product:

- containing only pairs that match on common attributes
- with one of each pair of common attributes eliminated

Consider relation schemas *R(ABC..JKLM)*, *S(KLMN..XYZ)*.

The natural join of relations *r(R)* and *s(S)* is defined as:

$$r \bowtie s \;=\; r \; Join \; s \;=\; \{\, (t_1[ABC..J] : t_2[K..XYZ]) \mid t_1 \in r \wedge t_2 \in s \wedge match \,\}$$

where $match \;=\; t_1[K] = t_2[K] \wedge t_1[L] = t_2[L] \wedge t_1[M] = t_2[M]$

Algorithmic view:

```
result = {}
for each tuple t1 in relation r
```

```
for each tuple t2 in relation s
    if (matches(t1,t2))
        result = result ∪ {combine(t1,t2)}
```

<< ^ >>

❖ Natural Join (cont)

Example of natural join:

R

A	B	C
a	m	x
e	k	y
f	p	z

S

C	D
x	1
z	2

R **Join** S

A	B	C	D
a	m	x	1
f	p	z	2

<< ∧ >>

❖ Theta Join

The theta join is a specialised product containing only pairs that match on a supplied condition *C*.

$r \bowtie_C s = \{ (t_1 : t_2) \mid t_1 \in r \wedge t_2 \in s \wedge C(t_1 : t_2) \}$,
where *r(R),s(S)*

Examples: *(r1 Join[B>E] r2)* ... *(r1 Join[E<D∧C=G] r2)*

All attribute names are required to be distinct (cf natural join)

Can be defined in terms of other RA operations:

$r \bowtie_C s$ = *r Join[C] s* = *Sel[C] (r × s)*

Note that $r \bowtie_{true} s = r \times s$.

<< ^ >>

❖ Theta Join (cont)

Example theta join:

R

A	B	C
a	m	x
e	k	y
f	p	z

S

D	E
x	1
x	2
y	3

R **Join**[C>D] S

A	B	C	D	E
e	k	y	x	1
e	k	y	x	2
f	p	z	x	1
f	p	z	x	2
f	p	z	y	3

(Theta join is the most frequently-used join in SQL queries)

<< ^ >>

❖ Theta Join (cont)

Querying with relational algebra (join) ...

- Who drinks in Newtown bars?

```
NewtownBars(nbar) = Sel[addr=Newtown](Bars)
Tmp = Frequents Join[bar=nbar] NewtownBars
Result(drinker) = Proj[drinker](Tmp)
```

- Who drinks beers made by Carlton?

```
CarltonBeers = Sel[manf=Carlton](Beers)
Tmp = Likes Join[beer=name] CarltonBeers
Result(drinker) = Proj[drinker)Tmp
```

Reminder: projection removes duplicates

❖ Outer Join

R $\bowtie_C$ S does not include in its result

- values from any R tuples that do not match some S tuple under C
- values from any S tuples that do not match some R tuple under C

R $⟕_C$ S (left outer join) includes

- all tuples that would result from a theta join
- values from all R tuples, even with no matching S tuple

For tuples with no match, assign NULL to "unmatched" attributes

Variants are right outer join and full outer join

<< ^ >>

❖ Outer Join (cont)

Example left outer join:

R

A	B	C
a	m	x
e	k	y
f	p	z

S

D	E
x	1
x	2
y	3

R **Left Outer Join**[C=D] S

A	B	C	D	E
a	m	x	x	1
a	m	x	x	2
e	k	y	y	3
f	p	z	NULL	NULL

<< /\ >>

❖ Division

Consider two relation schemas R and S where $S \subset R$.

The division operation is defined on instances $r(R)$, $s(S)$ as:

$$r / s = r \; Div \; s = \{ t \mid t \in r[R\text{-}S] \wedge satisfy \}$$

where $satisfy = \forall t_s \in S (\exists t_r \in R (t_r[S] = t_s \wedge t_r[R\text{-}S] = t))$

Operationally:

- consider each subset of tuples in R that match on $t[R\text{-}S]$
- for this subset of tuples, take the $t[S]$ values from each
- if this covers all tuples in S, then include $t[R\text{-}S]$ in the result

❖ Division (cont)

Example of division:

R

A	B
4	x
4	y
4	z
5	x
5	y
5	z

S

A	B
4	x
4	y
4	z
5	x
5	z

T

B
x
y

R / T

A
4
5

S / T

A
4

<< ^

❖ Division (cont)

Querying with relational algebra (division) ...

Division handles queries that include the notion "for all".

E.g. Which beers are sold in all bars?

We can answer this as follows:

- generate a relation of beers and bars where they are sold
  - *r1 = Proj[beer,bar](Sold)*
- generate a relation of all bars
  - *r2 = Rename[r2(bar)](Proj[name](Bars))*
- find which beers appear in tuples with every bar
  - *res = r1 Div r2*

Produced: 7 Apr 2021