

Relational model: Relational algebra continued

1

Basic operations of relational algebra:

- ✓ 1. Selection σ : select a subset of tuples from a relation according to a condition
- ✓ 2. Projection π : delete unwanted attributes (columns) from tuples of a relation
- 3. cross product \times : combine all pairs of tuples of two relations by making tuples with all attributes of both
- ✓ 4. Set difference $-$: * tuples in first relation and not in second
- ✓ 5. union \cup : * tuples in first relation or second relation
- 6. Renaming ρ : to deal with name conflicts

* Set operations: $D_1 \times D_2 \dots \times D_k$ of two relations must agree

2

Cross product $R \times T$

- Relations
 - $R \subseteq D_1 \times D_2 \times \dots \times D_k$
 - $T \subseteq S_1 \times S_2 \times \dots \times S_m$
- Resulting relation:
 - $R \times T \subseteq D_1 \times D_2 \times \dots \times D_k \times S_1 \times S_2 \times \dots \times S_m$
 - tuple $(d_1, d_2, \dots, d_k, s_1, s_2, \dots, s_m) \in R \times T$ if and only if $(d_1, d_2, \dots, d_k) \in R$ and $(s_1, s_2, \dots, s_m) \in T$
 - $|R \times T| = |R| \times |T|$? $|R|$ denotes the number of tuples in R
 - candidate keys?
 - foreign keys?

3

Cross product $R \times T$: keys

- Resulting relation:
 - $R \times T \subseteq D_1 \times D_2 \times \dots \times D_k \times S_1 \times S_2 \times \dots \times S_m$
 - tuple $(d_1, d_2, \dots, d_k, s_1, s_2, \dots, s_m) \in R \times T$ if and only if $(d_1, d_2, \dots, d_k) \in R$ and $(s_1, s_2, \dots, s_m) \in T$
 - $|R \times T| = |R| \times |T|$
- candidate keys:
 - $\{(d_{i1}, d_{i2}, \dots, d_{ia}) \text{ candidate key for } R$
 - $\{(s_{j1}, s_{j2}, \dots, s_{j\beta}) \text{ candidate key for } T$
 - the union of the attributes form a candidate key for $R \times T$
 - positions $i1, i2, \dots, ia, k+j1, k+j2, \dots, k+j\beta$ of $R \times T$
- foreign keys: for each of R and T are preserved using corresponding attributes of RXT.

4

Naming attributes

- Usually give attributes names
 - SS#, city, age, ...
- For cross-product, candidate key used positions in tuples to identify attributes
- Alternative naming: $R.d_i$ and $T.s_j$
 - Mayors.city, Legislators.city
- What if $R \times R$?
 - use positions of resulting tuples
 - rename one of the copies of R

5

Renaming $\rho_{Q(L)}(E)$

- E a relational algebra expression
- Q a new relation name
- L is a list of mappings of attributes of E:
 - mapping (old name \rightarrow new name)
 - mapping (attribute position \rightarrow new name)
- resulting relation named Q
 - is relation expressed by E
 - attributes renamed according to mappings in list L
 - Q can be omitted; L can be empty
- All constraints on relation expressed by E are preserved with appropriate renaming of attributes.
- Facilitates expressing queries; not indispensable

6

Using cross-product and renaming

- Cross-product allows coordination
 - see calculation of *max* in text §2.2.7
- Example

S: (studID, name) R: (studID, room#)

find relation giving (name, room#) pairs:

combine: S X R

coordinate: $\sigma_{S.stuID = R.stuID}(S X R)$

get result: $\pi_{S.name, R.room\#}(\sigma_{S.stuID = R.stuID}(S X R))$

find pairs of names of roommates ?

7

Example: find pairs of names of roommates:

S: (studID, name) R: (studID, room#)

relation:(name, room#) = $\pi_{S.name, R.room\#}(\sigma_{S.stuID = R.stuID}(S X R))$

combine: $(\pi_{S.name, R.room\#}(\sigma_{S.stuID = R.stuID}(S X R))) X$
 $\rho_{M(1 \rightarrow name, 2 \rightarrow room\#)}(\pi_{S.name, R.room\#}(\sigma_{S.stuID = R.stuID}(S X R)))$

now have (S.name, R.room#, M.name, M.room#)

coordinate: $\sigma_{R.room\# = M.room\#}(\pi_{S.name, R.room\#}(\sigma_{S.stuID = R.stuID}(S X R))) X$
 $\rho_{M(1 \rightarrow name, 2 \rightarrow room\#)}(\pi_{S.name, R.room\#}(\sigma_{S.stuID = R.stuID}(S X R)))$

get result: $\pi_{S.name, M.name}(\sigma_{R.room\# = M.room\#}(\pi_{S.name, R.room\#}(\sigma_{S.stuID = R.stuID}(S X R))) X$
 $\rho_{M(1 \rightarrow name, 2 \rightarrow room\#)}(\pi_{S.name, R.room\#}(\sigma_{S.stuID = R.stuID}(S X R))))$

8

Example: find pairs of names of roommates:

S: (studID, name) R: (studID, room#)

proposed solution:

$\pi_{S.name, M.name}(\sigma_{R.room\# = M.room\#}(\pi_{S.name, R.room\#}(\sigma_{S.stuID = R.stuID}(S X R)))) X$
 $\rho_{M(1 \rightarrow name, 2 \rightarrow room\#)}(\pi_{S.name, R.room\#}(\sigma_{S.stuID = R.stuID}(S X R)))$

keeps pairs representing "person roommate of his/her self"

can't recognize these after eliminate SS#

could be 2 people with same name in same room

fix: do RXR first and check SS#'s agree:

$\sigma_{R.room\# = Q.room\# \text{ AND } R.stuID \neq Q.stuID}(R X \rho_Q(R))$

9

Formal definition

- A relational expression is
 - A relation R in the database
 - A constant relation
 - For any relational expressions E_1 and E_2
 - $E_1 \cup E_2$
 - $E_1 - E_2$
 - $E_1 \times E_2$
 - $\sigma_P(E_1)$ for predicate P on attributes of E_1
 - $\pi_S(E_1)$ where S is a subset of attributes of E_1
 - $\rho_{Q(L)}(E_1)$ where Q is a new relation name and L is a list of (old name \rightarrow new name) mappings of attributes of E_1
- A query in the relational algebra is a relational expression

10

Relational algebra: derived operations

- operations can be expressed as **compositions** of fundamental operations
- operations represent **common patterns**
- operations are **very** useful for clarity

11

Intersection $R \cap T$

- direct from set theory

$$R \cap T = R - (R - T)$$
- example

students: (SS#, name, PUaddr, homeAddr, classYr)

employees: (SS#, name, addr, startYr)

find student employees:

$\pi_{SS\#, name, PUaddr}(\text{students}) \cap \pi_{SS\#, name, addr}(\text{employees})$

or

$\pi_{SS\#, name}(\text{students}) \cap \pi_{SS\#, name}(\text{employees})$

or

$\pi_{SS\#}(\text{students}) \cap \pi_{SS\#}(\text{employees})$ ← safest

or ...

12

Natural Join $R \bowtie T$: motivation

- Relations R and T
- Captures paradigm:
 - combine: $R \times T$
 - coordinate: $\sigma_P(R \times T)$
 - get result: $\pi_S(\sigma_P(R \times T))$
- For relations that have one or more attributes that **share name and domain**
- Need to refer to attributes shared by **identical name**
- Example:
 - students: (SS#, name, PUaddr, homeAddr, classYr)
 - employees: (SS#, name) addr, startYr

13

Natural Join $R \bowtie T$: definition

Let $\alpha(R)$ = the set of **names of attributes** in the schema for R

- Example: $\alpha(\text{Students}) = \{\text{SS\#, name, PUaddr, homeAddr, classYr}\}$

Let $\alpha(T)$ = the set of **names of attributes** in the schema for T

- Example: $\alpha(\text{Employees}) = \{\text{SS\#, name, addr, startYr}\}$

Let $\alpha(R) \cap \alpha(T) = \{a_1, a_2, \dots, a_k\}$

- Example: $\alpha(\text{Students}) \cap \alpha(\text{Employees}) = \{\text{SS\#, name}\}$

$$R \bowtie T = \pi_{\alpha(R) \cup \alpha(T)} (\sigma_{R.a_1=T.a_1, R.a_2=T.a_2, \dots, R.a_k=T.a_k} (R \times T))$$

- Students \bowtie Employees
- scheme: (SS#, name, PUaddr, homeAddr, classYr, addr, startYr)
- Student tuple and Employee tuple agree on values of SS#, name
- => tuple in join
- fill in values of the other attributes of the pair

14

Natural Join $R \bowtie T$: remarks

for $\alpha(R) \cap \alpha(T) = \{a_1, a_2, \dots, a_k\}$

$$R \bowtie T = \pi_{\alpha(R) \cup \alpha(T)} (\sigma_{R.a_1=T.a_1, R.a_2=T.a_2, \dots, R.a_k=T.a_k} (R \times T))$$

set union

- # attributes in $R \bowtie T$ =
- # attrib. in R + # attrib. in T - # attrib. in $\alpha(R) \cap \alpha(T)$
 - duplicate attributes removed
 - customary order in $R \bowtie T$:
attributes of R, attributes of T not also in R

each "=" test not valid if not on same domain
could weaken to compatible domains

15

Division $R \div Q$ – motivation

- Suggested by inverse of cross-product
 $(R \div Q) \times Q \subseteq R$ but *may not equal* R
- Find fragments of tuples of R that appear in R paired with **all** tuples of Q
- Example: database of tennis
 - relation Winners: (name, tournament, year)
 - find all players who have won **all** tournaments represented in the Winners relation

16

Division $R \div Q$ – definition

Given relations Q and R with attribute sets $\alpha(Q)$ and $\alpha(R)$,
Such that

- $\alpha(Q)$ is a **proper subset** of $\alpha(R)$
- corresponding attributes in $\alpha(R) \cap \alpha(Q)$ are on the **same domain**

Define

- $R \div Q$ is a relation with attribute set $\alpha(R \div Q) = \alpha(R) - \alpha(Q)$
- A tuple is in $R \div Q$ exactly when combining (concatenating) it with **every** tuple in Q yields a tuple in R
 - $R \div Q$ is a subset of $\pi_{\alpha(R) - \alpha(Q)}(R)$
 - not necessarily =
 - attribute order not maintained => using names to identify attributes

17

Division $R \div Q$ – example

relation Winners: (name, tournament, year)

find all players who have won all tournaments represented in the Winners relation

1. all tournaments: $\pi_{\text{tournament}}(\text{Winners})$
2. divide into something

Try winners \div $\pi_{\text{tournament}}(\text{Winners})$: ?

18

Division R÷Q – example

relation Winners: (name, tournament, year)

find all players who have won **all** tournaments represented in the Winners relation

1. all tournaments: $\pi_{\text{tournament}}(\text{Winners})$
2. divide into something
 $\text{winners} \div \pi_{\text{tournament}}(\text{Winners}) : (\text{name, year})$
 if tournaments are {US, French, Australian} need
 (S.Williams, US, 2008)
 (S.Williams, French, 2008)
 (S.Williams, Australian, 2008)
 to get S.Williams as a result
 and result tuple is (S.Williams, 2008)

⇒ get win all tournaments in **same year**

next try?

19

Division R÷Q – example

relation Winners: (name, tournament, year)

find all players who have won **all** tournaments represented in the Winners relation

1. all tournaments: $\pi_{\text{tournament}}(\text{Winners})$
2. divide into $\pi_{\text{name,tournament}}(\text{Winners}) : (\text{name, tournament})$

$\pi_{\text{name,tournament}}(\text{Winners}) \div \pi_{\text{tournament}}(\text{Winners}) : (\text{name})$

Gives desired result

20

Division R÷Q – how derive

R ÷ Q is expressed with basic relational operations as

$$\pi_{\alpha(R) - \alpha(Q)}(R) - \pi_{\alpha(R) - \alpha(Q)}((\pi_{\alpha(R) - \alpha(Q)}(R) \times Q) - R)$$

Huh?

- R ÷ Q is a subset of $\pi_{\alpha(R) - \alpha(Q)}(R)$
- what's in $\pi_{\alpha(R) - \alpha(Q)}(R)$ and **not** in R ÷ Q ?
 – a tuple that can't be combined with every tuple in Q to get a tuple in R
 ⇒ a combined tuple of $\pi_{\alpha(R) - \alpha(Q)}(R) \times Q$ that isn't in R
 ⇒ a tuple of $\pi_{\alpha(R) - \alpha(Q)}((\pi_{\alpha(R) - \alpha(Q)}(R) \times Q) - R)$

21

Subtract $\pi_{\alpha(R) - \alpha(Q)}((\pi_{\alpha(R) - \alpha(Q)}(R) \times Q) - R)$ from $\pi_{\alpha(R) - \alpha(Q)}(R)$?

- Let $(d_1, \dots, d_m) \in \pi_{\alpha(R) - \alpha(Q)}(R)$
 - Let $(q_1, \dots, q_n) \in Q$
- $(d_1, \dots, d_m, q_1, \dots, q_n) \in (\pi_{\alpha(R) - \alpha(Q)}(R) \times Q)$ may or may not be in R

If $(d_1, \dots, d_m) \in \pi_{\alpha(R) - \alpha(Q)}((\pi_{\alpha(R) - \alpha(Q)}(R) \times Q) - R)$

Then there is a $(q_1, \dots, q_n) \in Q$ such that

$(d_1, \dots, d_m, q_1, \dots, q_n)$ is in $(\pi_{\alpha(R) - \alpha(Q)}(R) \times Q) - R$

⇒ there is a $(q_1, \dots, q_n) \in Q$ such that

$(d_1, \dots, d_m, q_1, \dots, q_n)$ is not in R

⇒ (d_1, \dots, d_m) not in R ÷ Q

⇒ Correct to subtract (d_1, \dots, d_m) from $\pi_{\alpha(R) - \alpha(Q)}(R)$

note not maintaining order of attributes ⇒ identifying by name

22

Subtract $\pi_{\alpha(R) - \alpha(Q)}((\pi_{\alpha(R) - \alpha(Q)}(R) \times Q) - R)$ from $\pi_{\alpha(R) - \alpha(Q)}(R)$?

- Let $(d_1, \dots, d_m) \in \pi_{\alpha(R) - \alpha(Q)}(R)$
- Let $(q_1, \dots, q_n) \in Q$

But have we subtracted enough from $\pi_{\alpha(R) - \alpha(Q)}(R)$?

- If (d_1, \dots, d_m) not in R ÷ Q, then there is some $(q_1, \dots, q_n) \in Q$ such that $(d_1, \dots, d_m, q_1, \dots, q_n)$ not in R

⇒ $(d_1, \dots, d_m, q_1, \dots, q_n)$ in $((\pi_{\alpha(R) - \alpha(Q)}(R) \times Q) - R)$

⇒ (d_1, \dots, d_m) in $\pi_{\alpha(R) - \alpha(Q)}((\pi_{\alpha(R) - \alpha(Q)}(R) \times Q) - R)$

Yes, we have subtracted all that is needed

Note that $\pi_{\alpha(Q)}(R)$ may contain elements not in Q
 Not affect result.

23

Board examples

24

Board Example 1

students: (SS#, name, PUaddr, homeAddr, classYr)
employees: (SS#, name, addr, startYr)
assignment: (position, division, SS#, managerSS#)
study: (SS#, academic_dept., adviser)

saw find student employees:

$\pi_{SS\#}(students) \cap \pi_{SS\#}(employees)$ ← safest

now: find SS#, name, and classYr of all student employees

25

Board Example 2

students: (SS#, name, PUaddr, homeAddr, classYr)
employees: (SS#, name, addr, startYr)
assignment: (position, division, SS#, managerSS#)
study: (SS#, academic_dept., adviser)

find (student, manager) pairs where both are students - report SS#s

26

Board Example 3

students: (SS#, name, PUaddr, homeAddr, classYr)
employees: (SS#, name, addr, startYr)
jobs: (position, division, SS#, managerSS#)
study: (SS#, academic_dept., adviser)

find names of all CS students working for the library (library a division)

27

Board Example 4

students: (SS#, name, PUaddr, homeAddr, classYr)
employees: (SS#, name, addr, startYr)
assignment: (position, division, SS#, managerSS\$)
division foreign key referencing PUdivision
study: (SS#, academic_dept., adviser)
SS# foreign key referencing students
PUdivision: (division_name, address, director)

Find academic departments that have students working in all divisions

28

Relational algebra: extended operations

- operations **cannot** be expressed as compositions of fundamental operations
- operations allow **arithmetic, counting, grouping, and extending relations**
- part of database system language
 - postpone to SQL discussion

29

Summary

- Relational algebra operations provide foundation of query languages for database systems
- Derived operations, especially joins, simplify expressing queries
- Formal algebraic definition allow for provably correct simplifications, optimizations for query evaluation

30