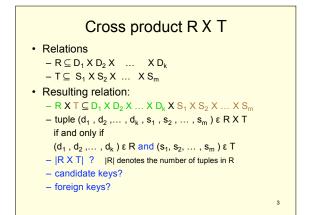


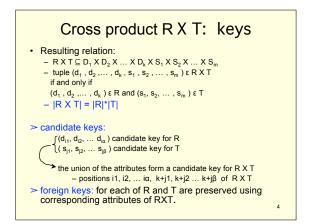
Basic operations of relational algebra:

- ✓ 1. Selection σ :select a subset of tuples from a relation according to a condition
- Projection π :delete unwanted attributes (columns) from tuples of a relation
- 3. cross product X : 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 U:* tuples in first relation or second relation
- > 6. Renaming ρ : to deal with name conflicts

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

2





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 X R?
 use positions of resulting tuples
 rename one of the copies of R

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 → new name)
- mapping (attribute position → 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: (stulD, name) R: (stulD, room#) find relation giving (name, room#) pairs: combine: SXR coordinate: $\sigma_{S.stulD = R.stulD}(S X R)$

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

find pairs of names of roommates ?

Example: find pairs of names of roommates: S: (stulD, name) R: (stulD, room#) relation:(name, room#) = $\pi_{S.name, R.room#}$ ($\sigma_{S.stulD = R.stulD}(S X R)$) combine: $(\pi_{S.name, R.room\#} (\sigma_{S.stulD = R.stulD}(S X R))) X$ $\rho_{M(1 \rightarrow name, 2 \rightarrow room\#)}(\pi_{S.name, R.room\#}(\sigma_{S.stulD = R.stulD}(S X R)))$ now have (S.name, R.room#, M.name, M.room#) coordinate: $\sigma_{R,room\#=M,room\#}$ ($(\pi_{S.name, R.room\#} (\sigma_{S.stulD = R.stulD}(S X R))) X$ $\rho_{M(1 \rightarrow name, 2 \rightarrow room\#)}(\pi_{S.name, R.room\#}(\sigma_{S.stulD = R.stulD}(S X R))))$ get result: $\pi_{S.name, M.name}$ ($\sigma_{\text{R.room#=M.room#}}$ ($(\pi_{\text{S.name, R.room#}} (\sigma_{\text{S.stuD} = \text{R.stuID}}(\text{S X R})))$ X $\rho_{M(1 \rightarrow name, 2 \rightarrow room\#)}(\pi_{S.name, R.room\#}(\sigma_{S.stulD = R.stulD}(S X R)))))$

Example: find pairs of names of roommates:

S: (stuID, name) R: (stulD, room#)

proposed solution:

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

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: $\sigma_{\text{R.room#=Q.room# AND R.stulD \neq Q.stulD}} (\text{ R X } \rho_{\text{Q}}(\text{ R }))$

Formal definition

- A relational expression is A relation R in the database
 - A constant relation
 - For any relational expressions E₁ and E₂
 - E1 U E2
 - E₁ E₂ E₁ X E₂

 - $\begin{array}{l} E_1 \land E_2 \\ \sigma_P(E_1) \text{ for predicate P on attributes of } E_1 \\ \pi_S(E_1) \text{ where S is a subset of attributes of } E_1 \\ \rho_{Q(L)}(E_1) \text{ where Q is a new relation name and L is a list of } \\ (\text{old name} \rightarrow \text{new name}) \text{ mappings of attributes of } E_1 \end{array}$

10

· A query in the relational algebra is a relational expression

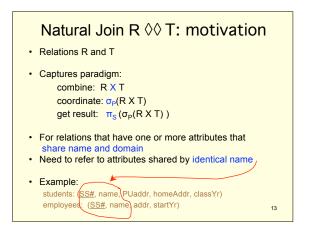
Relational algebra: derived operations

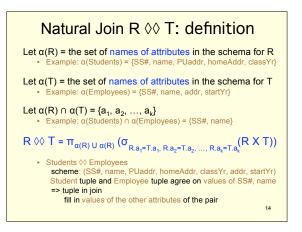
- operations can be expressed as compositions of fundamental operations
- operations represent common patterns

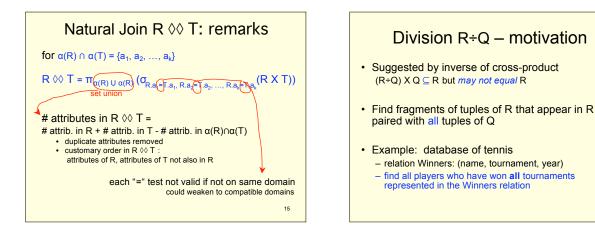
11

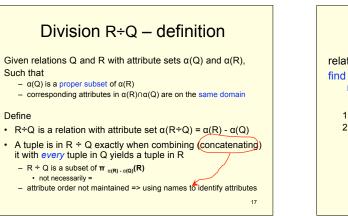
• operations are very useful for clarity

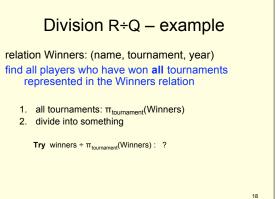
Intersection $R \cap T$ · direct from set theory $R \cap T = R - (R - T)$ example students: (SS#, name, PUaddr, homeAddr, classYr) employees: (SS#, name, addr, startYr) find student employees: π_{SS#, name, PUaddr}(students) **∩** π_{SS#, name, addr}(employees) or $\pi_{SS\#, name}(students) \cap \pi_{SS\#, name}(employees)$ or π_{SS#}(students) **∩** π_{SS#}(employees) ← safest or ... 12



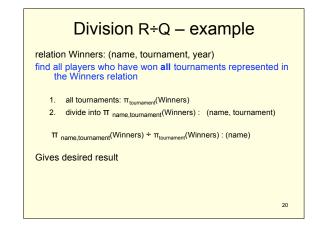


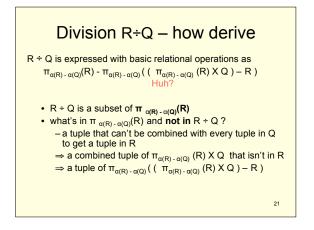


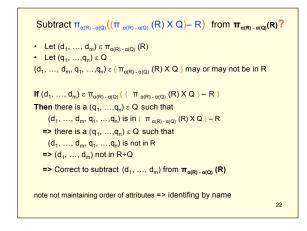


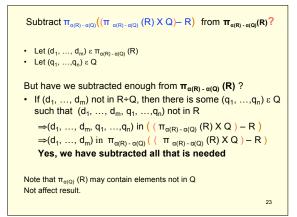


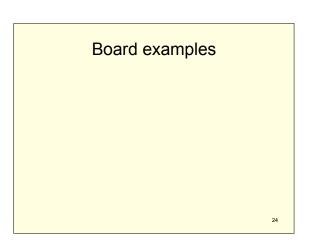












Board Example 1

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

saw find student employees: $\pi_{SS\#}(students) \cap \pi_{SS\#}(employees) \leftarrow safest$

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

25

27

Board Example 2

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

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

26

28

30

Board Example 3 students: (<u>SS#</u>, name, PUaddr, homeAddr, classYr) employees: (<u>SS#</u>, name, addr, startYr) jobs: (<u>position</u>, division, SS#, managerSS#) study: (<u>SS#</u>, academic_dept., adviser) find *names* of all CS students working for the library (library a division)

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

Board Example 4 students: (SS#, name, PUaddr, homeAddr, classYr)

Find academic departments that have students working in all divisions

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

5