COS 597A: Principles of Database and Information Systems

> Relational model: Relational algebra

Modeling access

- Have looked at modeling information as data + structure
- Now: how model access to data in relational model?
- · Formal specification of access provides:
 - Unambiguous queries
 - Correctness of results
 - Expressiveness of query languages

Queries

A query is a mapping from a set of relations to a relation

Query: relations \rightarrow relation

- Can derive schema of result from schemas of input relations
- Can deduce constraints on resulting relation that
 must hold for any input relations
- · Can identify properties of result relation

Relational query languages

- Two formal relational languages to describe mapping
 Relational algebra
 - Procedural lists operations to form query result
 - Relational calculus
 - Declarative describes results of query
- · Equivalent expressiveness
- Each has strong points for usefulness
- DB system query languages (e.g. SQL) take best of both

begin with Relational Algebra

Basic operations of relational algebra:

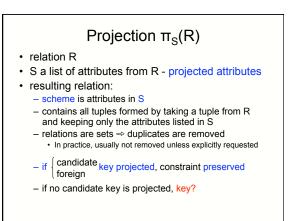
- 1. Selection σ :select a subset of tuples from a relation according to a condition
- 2. Projection π :delete unwanted attributes (columns) from tuples of a relation
- cross product X : combine all pairs of tuples of two relations by making tuples with all attributes of both
- 4. Set difference :* tuples in first relation and not in second
- union U:* tuples in first relation or second relation
- 6. Renaming ρ : to deal with name conflicts
- e. Renaming p. to doar with hame connicto

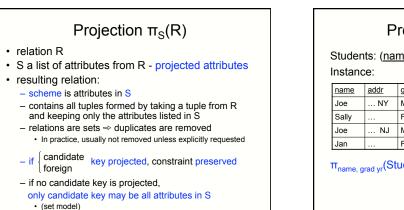
* Set operations: $D_1 X D_2 \dots X D_k$ of two relations must agree

Selection $\sigma_P(R)$

- relation R
- predicate P on attributes of R
- resulting relation
 - schema same as R
 - contains those tuples of R that satisfy P
 - candidate keys and foreign keys in R are preserved
 - eliminating tuples doesn't cause violations

| | Se | ele | ctic | on | Ex | ar | npl | е | | |
|---|--------------|-------|----------|---------|----------|--------|--------|---------|------|-------|
| Students: | (<u>nam</u> | e, a | addre | ess | , gei | nde | er, ag | ge, | grad | l yr) |
| Instance: | name | name | | address | | gender | | age | | yr |
| | Joe | Joe | | NY | | М | | 24 | | |
| | Sally | Sally | | | | F | | 25 | | |
| | Joe | Joe | | NJ M | | 23 | | | 2 | |
| | Jan | Jan | | F | | | 27 | | 4 | |
| σ _{age < 25} (Students): (<u>name, address</u> , gender, age, grad yr) | | | | | | | | | | |
| <u>n</u> a | ame | ado | dress ge | | nder age | | 9 | grad yr | | |
| Joe | | | NY | М | | 24 | | 2 | | |
| J | be | | NJ | М | | 23 | | 2 | | |
| | | | | | | | | | | |





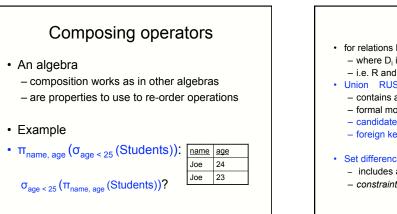
Projection Example

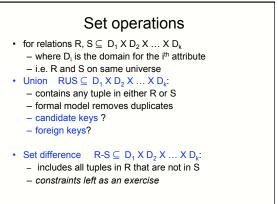
Students: (name, address, gender, age, grad yr)

| | | _ | | _ |
|-------------|------|--------|-----|---------|
| <u>name</u> | addr | gender | age | grad yr |
| Joe | NY | М | 24 | 2 |
| Sally | | F | 25 | 3 |
| Joe | NJ | м | 23 | 2 |
| Jan | | F | 27 | 4 |

π_{name, grad yr}(Students): (<u>name, grad yr</u>)

| <u>name</u> | grad yr | |
|-------------|---------|--|
| Joe | 2 | |
| Sally | 3 | |
| Jan | 4 | |
| | | |





Example for Union

 relations: mayors: (name, street address, <u>city</u>, party) legislators: (name, street address, city, <u>district</u>, party)

mayors klegislators? not same universe redefine:

mayors: (name, street address, <u>city</u>, term, party) If "term", "district" both integers \Rightarrow same domain \Rightarrow can union

candidate key of mayors U legislators?

Example for Union relations: mayors: (name, street address, <u>city</u>, term, party) legislators: (name, street address, city, <u>district</u>, party) gislators: (name, street address, city, <u>district</u>, party) candidate key of mayors U legislators? not (city, district) (Joe Smith, 9 Main St., Kingston, 1, democrat) Joe is mayor of Kingston in his first term (Sally Jones, 11 River Rd., Kingston, 1, republican) Sally is the legislator from the first district and lives in Kingston foreign key of mayors U legislators? corresponding components need not be the same attribute "term" versus "district"

CORRECTION

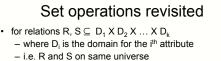
Candidate keys for union

I suggested if both R and S have same candidate key then will be candidate key for R U S. NO!

Generally, one key value determines two tuples – one from S and one from R.

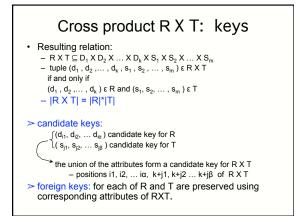
Example: gs_alum: (<u>ss#</u>, dept) ugrad_alum: (<u>ss#</u>, dept) ss# of alum who was both ugrad and grad but in different departments will appear in two tuples of

gs_alum U ugrad_alum



- Union $RUS \subseteq D_1 X D_2 X \dots X D_k$:
- contains any tuple in either R or S
 formal model removes duplicates
- candidate keys are not generally preserved
- a foreign key is preserved if it is a foreign key for both R and S using corresponding attributes and referencing the same relation
- Set difference $R-S \subseteq D_1 X D_2 X \dots X D_k$:
 - includes all tuples in R that are not in S
 - constraints left as an exercise

$\begin{array}{l} \textbf{Cross product R X T} \\ \bullet \text{ Relations} \\ & -R \subseteq D_1 \times D_2 \times \ldots \times D_k \\ & -T \subseteq S_1 \times S_2 \times \ldots \times S_m \\ \bullet \text{ Resulting relation:} \\ & -R \times T \subseteq D_1 \times D_2 \times \ldots \times D_k \times S_1 \times S_2 \times \ldots \times S_m \\ & - \text{ tuple } (d_1, d_2, \ldots, d_k, s_1, s_2, \ldots, s_m) \in R \times T \\ & \text{ if and only if} \\ & (d_1, d_2, \ldots, d_k) \in R \text{ and } (s_1, s_2, \ldots, s_m) \in T \\ & - |R \times T| ? \quad |R| \text{ denotes the number of tuples in R} \\ & - \text{ candidate keys?} \\ & - \text{ foreign keys?} \end{array}$



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, and T.s, - 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 \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.

Using cross-product and renaming

- · Cross-product allows coordination
- · Example S: (stulD, name) R: (stuID, 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 ?

What does this expression find?

Given relation R containing attribute value

 $\pi_{\textit{value}}\left(\mathsf{R}\right) - \pi_{\mathsf{R}.\textit{value}}\left(\sigma_{\mathsf{R}.\textit{value}} < \mathsf{Q}.\textit{value}\left(\mathsf{R} \mid \mathsf{X} \mid \rho(\mathsf{Q},\mathsf{R})\right)\right)$

[From Silberchatz et. al. Section 6.1.1.7]

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
 - $\sigma_P(E_1)$ for predicate P on attributes of E_1

 - $\begin{array}{l} \underset{T_{S}(E_{1})}{\underset{T_{S}(E_{1})}{\underset{Where S is a subset of attributes of E_{1}} \\ \underset{P(Q(L),E_{1})}{\underset{P(Q(L),E_{1})}{\underset{Where Q is a new relation name and L is a list of (old name \rightarrow new name) mappings of attributes of E_{1} \end{array}$
- 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
- operations are very useful for clarity

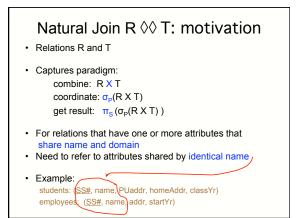
Intersection R n T

· direct from set theory

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

example

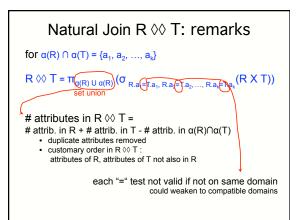
students: (SS#, name, PUaddr, homeAddr, classYr) employees: (SS#, name, addr, startYr) find student employees π_{SS#, name, PUaddr}(students) ∩ π_{SS#, name, addr}(employees) or $\pi_{\text{SS\#, name}}(\text{students}) \cap \pi_{\text{SS\#, name}}(\text{employees})$ or $\pi_{SS\#}(students) \cap \pi_{SS\#}(employees) \leftarrow safest$ or ...



Natural Join R 00 T: definition Let $\alpha(R)$ = the set of names of attributes in the schema for R Example: α(Students) = {SS#, name, PUaddr, homeAddr, classYr} Let $\alpha(T)$ = the set of names of attributes in the schema for T Example: α(Employees) = {SS#, name, addr, startYr} Let $\alpha(\mathbf{R}) \cap \alpha(\mathbf{T}) = \{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_k\}$ • Example: $\alpha(\text{Students}) \cap \alpha(\text{Employees}) = \{\text{SS#, name}\}$ $\mathsf{R} \diamond \diamond \mathsf{T} = \pi_{\alpha(\mathsf{R}) \cup \alpha(\mathsf{R})} \left(\sigma_{\mathsf{R}:a_1 = \mathsf{T}:a_1, \mathsf{R}:a_2 = \mathsf{T}:a_2, \dots, \mathsf{R}:a_k = \mathsf{T}:a_k} (\mathsf{R} \mathsf{X} \mathsf{T}) \right)$

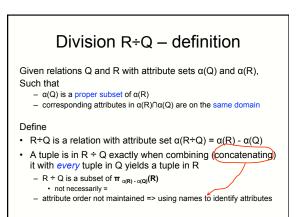
 Students ◊◊ Employees scheme: (SS#, name, PUaddr, homeAddr, classYr, addr, startYr) Student tuple and Employee tuple agree on values of SS#, name => tuple in join

fill in values of the other attributes of the pair



Division R+Q – motivation

- · Suggested by inverse of cross-product $(R+Q) X Q \subseteq R$ 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



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: π_{tournament}(Winners) 2. divide into something winners + π_{tournament}(Winners) : (name, year) if tournaments are {US, French, Australian} need (S.Williams, US, 2008) (S.Williams, French, 2008) (S.Williams, Australian, 2008) to get S.Willaims as a result and result tuple is (S.Willaims, 2008)

⇒ get win all tournaments in *same year*

next try?

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: π_{tournament}(Winners)
- 2. divide into $\pi_{name,tournament}$ (Winners) : (name, tournament)
- $\pi_{name,tournament}$ (Winners) ÷ $\pi_{tournament}$ (Winners) : (name)

Gives desired result

Division R÷Q – how derive

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

 $\label{eq:acombined tuple of $\pi_{\alpha(R) \, - \, \alpha(Q)}$ (R) X Q that isn't in R } \\ \end{tabular} \end{tabular} a tuple of $\pi_{\alpha(R) \, - \, \alpha(Q)}$ ($\pi_{\alpha(R) \, - \, \alpha(Q)}$ (R) X Q) - R) }$

Board Examples

Database:

students: (<u>SS#</u>, name, PUaddr, homeAddr, classYr)
employees: (<u>SS#</u>, name, addr, startYr)
jobs: (<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 1

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

Board Example 2

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

Board Example 3

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

Board Example 4

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

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