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 → 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
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 … 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

Selection Example

Students: (<u>name, address</u>, gender, age, grad yr)

Instance:

<u>name</u>	<u>address</u>	gender	age	grad yr
Joe	… NY	M	24	2
Sally	…	F	25	3
Joe	… NJ	M	23	2
Jan	…	F	27	4

$\sigma_{age < 25}$ (Students): (<u>name, address</u>, gender, age, grad yr)

<u>name</u>	<u>address</u>	gender	age	grad yr
Joe	… NY	M	24	2
Joe	… NJ	M	23	2

Projection $\pi_S(R)$

- relation R
- S a list of attributes from R - projected attributes
- resulting relation:
  - scheme is attributes in S
  - contains all tuples formed by taking a tuple from R and keeping only the attributes listed in S
  - relations are sets ⇨ duplicates are removed
    - In practice, usually not removed unless explicitly requested
  - if {candidate / foreign} key projected, constraint preserved
  - if no candidate key is projected, key?

Projection $\pi_S(R)$

- relation R
- S a list of attributes from R - projected attributes
- resulting relation:
  - scheme is attributes in S
  - contains all tuples formed by taking a tuple from R and keeping only the attributes listed in S
  - relations are sets ⇨ duplicates are removed
    - In practice, usually not removed unless explicitly requested
  - if {candidate / foreign} key projected, constraint preserved
  - if no candidate key is projected, only candidate key may be all attributes in S
    - (set model)

Projection Example

Students: (<u>name, address</u>, gender, age, grad yr)

Instance:

<u>name</u>	<u>addr</u>	gender	age	grad yr
Joe	… NY	M	24	2
Sally	…	F	25	3
Joe	… NJ	M	23	2
Jan	…	F	27	4

$\pi_{name,\ grad\ yr}$(Students): (<u>name, grad yr</u>)

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

Composing operators

- An algebra
  - composition works as in other algebras
  - are properties to use to re-order operations
- Example
- $\pi_{name,\ age}(\sigma_{age < 25}(\text{Students}))$:

  $\sigma_{age < 25}(\pi_{name,\ age}(\text{Students}))$?

<u>name</u>	<u>age</u>
Joe	24
Joe	23

Set operations

- for relations R, S ⊆ $D_1 \times D_2 \times \ldots \times D_k$
  - where D_i is the domain for the i^{th} attribute
  - i.e. R and S on same universe
- Union R∪S ⊆ $D_1 \times D_2 \times \ldots \times D_k$:
  - contains any tuple in either R or S
  - formal model removes duplicates
  - candidate keys ?
  - foreign keys?
- Set difference R-S ⊆ $D_1 \times D_2 \times \ldots \times D_k$:
  - includes all tuples in R that are not in S
  - *constraints left as an exercise*

Example for Union

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

mayors ~~U~~ legislators? not same universe

redefine:
mayors: (name, street address, <u>city</u>, term, party)
If "term", "district" both integers
⇒ same domain ⇒ 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)
- 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

Set operations revisited

- for relations R, S $\subseteq$ D_1 X D_2 X … X D_k
  – where D_i is the domain for the i^{th} attribute
  – i.e. R and S on same universe
- Union RUS $\subseteq$ D_1 X D_2 X … 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 … X D_k:
  – includes all tuples in R that are not in S
  – *constraints left as an exercise*

Cross product R X T

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

Cross product R X T: keys

- Resulting relation:
  – R X T $\subseteq$ D_1 X D_2 X … X D_k X S_1 X S_2 X … X S_m
  – tuple (d_1 , d_2 ,… , d_k , s_1 , s_2 , … , s_m) ε R X T
    if and only if
    (d_1 , d_2 ,… , d_k) ε R and (s_1, s_2, … , s_m) ε T
  – |R X T| = |R|*|T|

➢ candidate keys:
  (d_{i1}, d_{i2}, … $d_{i\alpha}$) candidate key for R
  (s_{j1}, s_{j2}, … $s_{j\beta}$) candidate key for T
  the union of the attributes form a candidate key for R X T
  – positions i1, i2, … iα, k+j1, k+j2 … k+jβ of R X T

➢ foreign keys: for each of R and T are preserved using corresponding attributes of RXT.

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 ρ(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.

Using cross-product and renaming

- Cross-product allows coordination
- Example

  S: (<u>stuID</u>, name) R: (<u>stuID</u>, room#)

  find relation giving (name, room#) pairs:

  combine: S X R

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

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

  find pairs of names of roommates ?

What does this expression find?

Given relation R containing attribute *value*

$\pi_{value}(R) - \pi_{R.value}(\sigma_{R.value < Q.value}(R \times \rho(Q,R)))$

[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 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 → new name) mappings of attributes of E_1
- 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 ∩ T

- direct from set theory

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

  students: (<u>SS#</u>, name, PUaddr, homeAddr, classYr)

  employees: (<u>SS#</u>, name, addr, startYr)

  find student employees:

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

  or

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

  or

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

  or …

Natural Join R ◊◊ T: motivation

- Relations R and T
- Captures paradigm:
  - combine: R X T
  - coordinate: σ_P(R X T)
  - get result: π_S (σ_P(R X 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: (<u>SS#</u>, name, PUaddr, homeAddr, classYr)

  employees: (<u>SS#</u>, name, addr, startYr)

Natural Join R ◊◊ T: definition

Let α(R) = the set of names of attributes in the schema for R
- Example: α(Students) = {SS#, name, PUaddr, homeAddr, classYr}

Let α(T) = the set of names of attributes in the schema for T
- Example: α(Employees) = {SS#, name, addr, startYr}

Let α(R) ∩ α(T) = $\{a_1, a_2, \ldots, a_k\}$
- Example: α(Students) ∩ α(Employees) = {SS#, name}

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

- 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

Natural Join R ◊◊ T: remarks

for α(R) ∩ α(T) = $\{a_1, a_2, \ldots, a_k\}$

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

set union

\# attributes in R ◊◊ T =

\# attrib. in R + # attrib. in T - # attrib. in α(R)∩α(T)
- duplicate attributes removed
- customary order in R ◊◊ 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

Division R÷Q – motivation

- Suggested by inverse of cross-product

  (R÷Q) X Q ⊆ 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 – definition

Given relations Q and R with attribute sets α(Q) and α(R),
Such that
- α(Q) is a proper subset of α(R)
- corresponding attributes in α(R)∩α(Q) are on the same domain

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

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: $\pi_{tournament}$(Winners)
2. divide into something

   winners ÷ $\pi_{tournament}$(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: $\pi_{tournament}$(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 expressed with basic relational operations as

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

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) X Q that isn't in R

    ⇒ a tuple of $\pi_{\alpha(R) - \alpha(Q)}((\pi_{\alpha(R) - \alpha(Q)}(R) X Q) - R)$

Board Examples

Database:

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

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

jobs: (position, division, SS#, managerSS#)

division *foreign key referencing* PUdivision

study: (SS#, academic_dept., adviser)

SS# *foreign key referencing* students

PUdivision: (division_name, address, director)

Board Example 1

saw find student employees:

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

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