CMU SCS

Carnegie Mellon Univ.
School of Computer Science
15-415 - Database Applications

15-515 (Fall 2010)

Lecture #6: *Relational Algebra*

(Slides from Christos Faloutsos)

CMU SCS

Overview

- history
- concepts
- Formal query languages
  - relational algebra
  - rel. tuple calculus
  - rel. domain calculus

CMU SCS

History

- before: records, pointers, sets etc
- introduced by E.F. Codd in 1970
- revolutionary!
- first systems: 1977-8 (System R; Ingres)
- Turing award in 1981

CMU SCS

Concepts - reminder

- Database: a set of relations (= tables)
- rows: tuples
- columns: attributes (or keys)
- superkey, candidate key, primary key

CMU SCS

Example

Database:

STUDENT		
Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

SSN	c-id	grade
123	15-413	A
234	15-413	B

CMU SCS

Example: cont'd

Database:

k-th attribute
(Dk domain)

STUDENT		
Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

rel. schema (attr+domains)

tuple

CMU SCS

Example: cont'd

STUDENT		
Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

rel. schema (attr+domains)

instance

CMU SCS

Example: cont'd

- Di: the domain of the i-th attribute (eg., char(10)

STUDENT		
Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

rel. schema (attr+domains)

instance

CMU SCS

Overview

- history
- concepts
- **Formal query languages**
  - relational algebra
  - rel. tuple calculus
  - rel. domain calculus

CMU SCS

Formal query languages

- How do we collect information?
- Eg., find ssn's of people in 415
- (recall: everything is a set!)
- One solution: Rel. algebra, ie., set operators
- Q1: Which ones??
- Q2: what is a minimal set of operators?

CMU SCS

Relational operators

- .
- .
- .
- set union **U**
- set difference '-'

CMU SCS

Example:

- Q: find all students (part or full time)
- A: PT-STUDENT union FT-STUDENT

FT-STUDENT		
Ssn	Name	
129	peters	main str
239	lee	5th ave

PT-STUDENT		
Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

CMU SCS

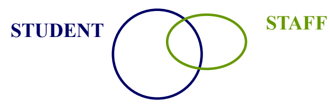
Observations:

- two tables are 'union compatible' if they have the same attributes ('domains')
- Q: how about intersection ∩

CMU SCS

Observations:

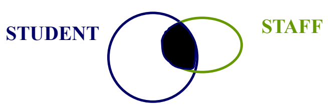
- A: redundant:
- STUDENT intersection STAFF =



CMU SCS

Observations:

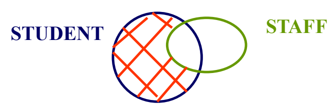
- A: redundant:
- STUDENT intersection STAFF =



CMU SCS

Observations:

- A: redundant:
- STUDENT intersection STAFF =
  STUDENT - (STUDENT - STAFF)



CMU SCS

Observations:

- A: redundant:
- STUDENT intersection STAFF =
  STUDENT - (STUDENT - STAFF)

Double negation:

We'll see it again, later…

CMU SCS

Relational operators

- .
- .
- .
- set union U
- set difference '-'

Other operators?

- eg, find all students on 'Main street'
- A: 'selection'

$$\sigma_{address='mainstr'}(STUDENT)$$

STUDENT		
Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

Other operators?

- Notice: selection (and rest of operators) expect tables, and produce tables (-> can be cascaded!!)
- For selection, in general:

$$\sigma_{condition}(RELATION)$$

Selection - examples

- Find all 'Smiths' on 'Forbes Ave'

$$\sigma_{name='Smith' \wedge address='Forbes ave'}(STUDENT)$$

'condition' can be any boolean combination of '=', '>', '>=', ...

Relational operators

- selection $\sigma_{condition}(R)$
- .
- .
- set union R ∪ S
- set difference R - S

Relational operators

- selection picks rows - how about columns?
- A: 'projection' - eg.: $\pi_{ssn}(STUDENT)$

finds all the 'ssn' - removing duplicates

STUDENT		
Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

Relational operators

Cascading: 'find ssn of students on 'forbes ave'

$$\pi_{ssn}(\sigma_{address='forbes ave'}(STUDENT))$$

STUDENT		
Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

CMU SCS

Relational operators

- selection $\sigma_{condition}\ (R)$
- projection $\pi_{att-list}(R)$
- .
- set union R ∪ S
- set difference R - S

CMU SCS

Relational operators

Are we done yet?

Q: Give a query we can **not** answer yet!

CMU SCS

Relational operators

A: any query across **two** or more tables,

eg., 'find names of students in 15-415'

Q: what extra operator do we need??

STUDENT

Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

TAKES

SSN	c-id	grade
123	15-413	A
234	15-413	B

CMU SCS

Relational operators

A: any query across **two** or more tables,

eg., 'find names of students in 15-415'

Q: what extra operator do we need??

A: surprisingly, cartesian product is enough!

STUDENT

Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

TAKES

SSN	c-id	grade
123	15-413	A
234	15-413	B

CMU SCS

Cartesian product

- eg., dog-breeding: MALE x FEMALE
- gives all possible couples

MALE

name
spike
spot

x

FEMALE

name
lassie
shiba

=

M.name	F.name
spike	lassie
spike	shiba
spot	lassie
spot	shiba

CMU SCS

so what?

- Eg., how do we find names of students taking 415?

STUDENT

Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

SSN	c-id	grade
123	15-415	A
234	15-413	B

CMU SCS

Cartesian product

- A: ....$\sigma_{STUDENT.ssn=TAKES.ssn}(STUDENT\ x\ TAKES)$

Ssn	Name	Address	ssn	cid	grade
123	smith	main str	123	15-415	A
~~234~~	~~jones~~	~~forbes ave~~	~~123~~	~~15-415~~	~~A~~
~~123~~	~~smith~~	~~main str~~	~~234~~	~~15-413~~	~~B~~
234	jones	forbes ave	234	15-413	B

CMU SCS

Cartesian product

..$\sigma_{cid=15-415}(\sigma_{STUDENT.ssn=TAKES.ssn}(STUDENT\ x\ TAKES))$

Ssn	Name	Address	ssn	cid	grade
123	smith	main str	123	15-415	A
~~234~~	~~jones~~	~~forbes ave~~	~~123~~	~~15-415~~	~~A~~
~~123~~	~~smith~~	~~main str~~	~~234~~	~~15-413~~	~~B~~
~~234~~	~~jones~~	~~forbes ave~~	~~234~~	~~15-413~~	~~B~~

CMU SCS

$\pi_{name}($

$\sigma_{cid=15-415}(\sigma_{STUDENT.ssn=TAKES.ssn}(STUDENT\ x\ TAKES))$

$)$

Ssn	Name	Address	ssn	cid	grade
123	smith	main str	123	15-415	A
~~234~~	~~jones~~	~~forbes ave~~	~~123~~	~~15-415~~	~~A~~
~~123~~	~~smith~~	~~main str~~	~~234~~	~~15-413~~	~~B~~
~~234~~	~~jones~~	~~forbes ave~~	~~234~~	~~15-413~~	~~B~~

CMU SCS

FUNDAMENTAL Relational operators

- selection $\sigma_{condition}\ (R)$
- projection $\pi_{att-list}(R)$
- cartesian product MALE x FEMALE
- set union R ∪ S
- set difference R - S

CMU SCS

Relational ops

- Surprisingly, they are enough, to help us answer almost any query we want!!
- derived/convenience operators:
  - set intersection
  - **join** (theta join, equi-join, natural join) $\bowtie$
  - 'rename' operator $\rho_{R'}(R)$
  - division $R \div S$

CMU SCS

Joins

- Equijoin: $R \bowtie_{R.a=S.b} S \;\; = \sigma_{R.a=S.b}(R \times S)$

CMU SCS

Cartesian product

- A: ....$\sigma_{STUDENT.ssn=TAKES.ssn}(STUDENT\ x\ TAKES)$

Ssn	Name	Address	ssn	cid	grade
123	smith	main str	123	15-415	A
~~234~~	~~jones~~	~~forbes ave~~	~~123~~	~~15-415~~	~~A~~
~~123~~	~~smith~~	~~main str~~	~~234~~	~~15-413~~	~~B~~
234	jones	forbes ave	234	15-413	B

CMU SCS

Joins

- Equijoin: $R \bowtie_{R.a=S.b} S \ = \sigma_{R.a=S.b}(R \times S)$
- theta-joins: $R \bowtie_{\theta} S$
  generalization of equi-join - any condition θ

CMU SCS

Joins

- **very** popular: natural join: R $\bowtie$ S
- like equi-join, but it drops duplicate columns:
  STUDENT (ssn, name, address)
  TAKES (ssn, cid, grade)

CMU SCS

Joins

- nat. join has 5 attributes $STUDENT \bowtie TAKES$

Ssn	Name	Address	ssn	cid	grade
123	smith	main str	123	15-415	A
234	jones	forbes ave	123	15-415	A
123	smith	main str	234	15-413	B
234	jones	forbes ave	234	15-413	B

equi-join: 6 $STUDENT \bowtie_{STUDENT.ssn=TAKES.ssn} TAKES$

CMU SCS

Natural Joins - nit-picking

- if no attributes in common between R, S:
  nat. join -> cartesian product

CMU SCS

Overview - rel. algebra

- fundamental operators
- derived operators
  - joins etc
  - **rename**
  - division
- examples

Rename op.

- Q: why? $\rho_{AFTER}(BEFORE)$
- A: shorthand; self-joins; …
- for example, find the grand-parents of ‘Tom’, given PC (parent-id, child-id)

Rename op.

- PC (parent-id, child-id) $PC \bowtie PC$

PC	
p-id	c-id
Mary	Tom
Peter	Mary
John	Tom

PC	
p-id	c-id
Mary	Tom
Peter	Mary
John	Tom

Rename op.

- first, WRONG attempt:

$$PC \bowtie PC$$

- (why? how many columns?)
- Second WRONG attempt:

$$PC \bowtie_{PC.c-id=PC.p-id} PC$$

Rename op.

- we clearly need two different names for the same table - hence, the ‘rename’ op.

$$\rho_{PC1}(PC) \bowtie_{PC1.c-id=PC.p-id} PC$$

Overview - rel. algebra

- fundamental operators
- derived operators
  - joins etc
  - rename
  - **division**
- examples

Division

- Rarely used, but powerful.
- Example: find suspicious suppliers, ie., suppliers that supplied all the parts in A_BOMB

CMU SCS

Division

SHIPMENT	
s#	p#
s1	p1
s2	p1
s1	p2
s3	p1
s5	p3

÷

ABOMB
p#
p1
p2

=

BAD_S
s#
s1

CMU SCS

Division

- Observations: ~reverse of cartesian product
- It can be derived from the 5 fundamental operators (!!)
- How?

CMU SCS

Division

- Answer:

$$r \div s = \pi_{(R-S)}(r) - \pi_{(R-S)}[(\pi_{(R-S)}(r) \times s) - r]$$

- Observation: find ‘good’ suppliers, and subtract! (**double negation**)

CMU SCS

Division

- Answer:

SHIPMENT	
s#	p#
s1	p1
s2	p1
s1	p2
s3	p1
s5	p3

÷

ABOMB
p#
p1
p2

=

BAD_S
s#
s1

$$r \div s = \pi_{(R-S)}(r) - \pi_{(R-S)}[(\pi_{(R-S)}(r) \times s) - r]$$

- Observation: find ‘good’ suppliers, and subtract! (**double negation**)

CMU SCS

Division

- Answer:

SHIPMENT	
s#	p#
s1	p1
s2	p1
s1	p2
s3	p1
s5	p3

÷

ABOMB
p#
p1
p2

=

BAD_S
s#
s1

$$r \div s = \pi_{(R-S)}(r) - \pi_{(R-S)}[(\pi_{(R-S)}(r) \times s) - r]$$

All suppliers

All bad parts

CMU SCS

Division

- Answer:

SHIPMENT	
s#	p#
s1	p1
s2	p1
s1	p2
s3	p1
s5	p3

÷

ABOMB
p#
p1
p2

=

BAD_S
s#
s1

$$r \div s = \pi_{(R-S)}(r) - \pi_{(R-S)}[(\pi_{(R-S)}(r) \times s) - r]$$

all possible

suspicious shipments

CMU SCS

Division

- Answer:

SHIPMENT

s#	p#
s1	p1
s2	p1
s1	p2
s3	p1
s5	p3

÷

ABOMB

p#
p1
p2

=

BAD_S

s#
s1

$$r \div s = \pi_{(R-S)}(r) - \pi_{(R-S)}[(\pi_{(R-S)}(r) \times s) - r]$$

all possible

suspicious shipments

that didn't happen

CMU SCS

Division

- Answer:

SHIPMENT

s#	p#
s1	p1
s2	p1
s1	p2
s3	p1
s5	p3

÷

ABOMB

p#
p1
p2

=

BAD_S

s#
s1

$$r \div s = \pi_{(R-S)}(r) - \pi_{(R-S)}[(\pi_{(R-S)}(r) \times s) - r]$$

all suppliers who missed

at least one suspicious shipment,

i.e.: 'good' suppliers

CMU SCS

Overview - rel. algebra

- fundamental operators
- derived operators
  - joins etc
  - rename
  - division
- **examples**

CMU SCS

Sample schema

find names of students that take 15-415

STUDENT

Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

CLASS

c-id	c-name	units
15-413	s.e.	2
15-412	o.s.	2

TAKES

SSN	c-id	grade
123	15-413	A
234	15-413	B

CMU SCS

Examples

- find names of students that take 15-415

CMU SCS

Examples

- find names of students that take 15-415

$$\pi_{name}[\sigma_{c-id=15-415}(STUDENT \bowtie TAKES)]$$

Sample schema

find course names of 'smith'

STUDENT		
Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

CLASS		
c-id	c-name	units
15-413	s.e.	2
15-412	o.s.	2

TAKES		
SSN	c-id	grade
123	15-413	A
234	15-413	B

Examples

- find course names of 'smith'

$$\pi_{c-name}[\,\sigma_{name='smith'}(\;STUDENT \bowtie TAKES \bowtie CLASS\;)]$$

Examples

- find ssn of 'overworked' students, ie., that take 412, 413, 415

Examples

- find ssn of 'overworked' students, ie., that take 412, 413, 415: almost correct answer:

$$\sigma_{c-name=412}(TAKES) \bigcap$$
$$\sigma_{c-name=413}(TAKES) \bigcap$$
$$\sigma_{c-name=415}(TAKES)$$

Examples

- find ssn of 'overworked' students, ie., that take 412, 413, 415 - Correct answer:

$$\pi_{ssn}[\sigma_{c-name=412}(TAKES)] \bigcap$$
$$\pi_{ssn}[\sigma_{c\text{-}name=413}\;(TAKES)] \bigcap$$
$$\pi_{ssn}[\sigma_{c\text{-}name=415}\;(TAKES)]$$

Examples

- find ssn of students that work at least as hard as ssn=123, ie., they take all the courses of ssn=123, and maybe more

CMU SCS

Sample schema

STUDENT		
Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

CLASS		
c-id	c-name	units
15-413	s.e.	2
15-412	o.s.	2

TAKES		
SSN	c-id	grade
123	15-413	A
234	15-413	B

CMU SCS

Examples

- find ssn of students that work at least as hard as ssn=123 (ie., they take all the courses of ssn=123, and maybe more

$$[\pi_{ssn,c-id}(TAKES)] \div \pi_{c-id}[\sigma_{ssn=123}(TAKES)]$$

CMU SCS

Conclusions

- Relational model: only tables ('relations')
- relational algebra: powerful, minimal: 5 operators can handle almost any query!