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

C. Faloutsos

Relational model

Overview

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

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

Concepts

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

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

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**

Example: cont'd

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

rel. schema (attr+domains)

instance

Example: cont'd

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

•Formally: an instance is a subset of (D1 x D2 x …x Dn)

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

rel. schema (attr+domains)

instance

Example: cont'd

- superkey (eg., 'ssn , name'): determines record
- cand. key (eg., 'ssn', or 'st#'): minimal superkey
- primary key: one of the cand. keys

Overview

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

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?

Relational operators

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

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

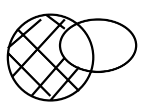
Observations:

- two tables are ‘union compatible’ if they have the same attributes (‘domains’)
- Q: how about intersection $\cap$

Observations:

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

STUDENT



STAFF

Relational operators

- .
- .
- .
- set union $\cup$
- set difference ‘-’

Other operators?

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

$$\sigma_{address='main\ str'}(STUDENT)$$

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 '=', '>', '>=', ...

Carnegie Mellon 15-415 - C. Faloutsos 19

Relational operators

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

Carnegie Mellon 15-415 - C. Faloutsos 20

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

Carnegie Mellon 15-415 - C. Faloutsos 21

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

Carnegie Mellon 15-415 - C. Faloutsos 22

Relational operators

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

Carnegie Mellon 15-415 - C. Faloutsos 23

Relational operators

Are we done yet?

Q: Give a query we can not answer yet!

Carnegie Mellon 15-415 - C. Faloutsos 24

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

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

so what?

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

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

Cartesian product

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

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

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~~	~~15-413~~	~~A~~
~~234~~	~~jones~~	~~forbes ave~~	~~123~~	~~15-413~~	~~A~~
~~123~~	~~smith~~	~~main str~~	~~234~~	~~15-413~~	~~B~~
~~234~~	~~jones~~	~~forbes ave~~	~~234~~	~~15-413~~	~~B~~

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

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

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

Relational ops

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

Joins

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

Cartesian product

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

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

Joins

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

Joins

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

Joins

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

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

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

Natural Joins - nit-picking

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

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

Division

SHIPMENT

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

÷

ABOMB

p#
p1
p2

=

BAD_S

s#
s1

Division

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

Division

- Answer:

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

Overview - rel. algebra

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

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

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:

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

Examples

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

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

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

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

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

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

Conclusions

- Relational model: only tables ('relations')
- relational algebra: powerful, minimal: 5 operators can handle almost any query!
- most non-trivial op.: join