Carnegie Mellon Univ.
School of Computer Science
15-415/615 - DB Applications

C. Faloutsos
Lecture #4: Relational Algebra

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 - reminder

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

Example

Database:

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>SSN</th>
<th>Name</th>
<th>Address</th>
<th>c-id</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>smith</td>
<td>main str</td>
<td>123-456</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>234</td>
<td>jones</td>
<td>forbes ave</td>
<td>234-567</td>
<td>B</td>
<td></td>
</tr>
</tbody>
</table>

Example: cont’d

Database:

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>k-th attribute</th>
<th>rel. schema (attr+domains)</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Name</td>
<td>tuple</td>
</tr>
<tr>
<td>234</td>
<td>Smith</td>
<td></td>
</tr>
</tbody>
</table>

(Dk domain)
Example: cont’d

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>rel. schema (attr+domains)</th>
<th>instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>San</td>
<td>Name</td>
<td>Address</td>
</tr>
<tr>
<td>123</td>
<td>smith</td>
<td>main str</td>
</tr>
<tr>
<td>234</td>
<td>jones</td>
<td>forbes ave</td>
</tr>
</tbody>
</table>

Example: cont’d

- $D_i$: the domain of the $i$-th attribute (e.g., char(10))

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

<table>
<thead>
<tr>
<th>FT-STUDENT</th>
<th>PT-STUDENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>San Name</td>
<td>Name</td>
</tr>
<tr>
<td>128 peters</td>
<td>smith</td>
</tr>
<tr>
<td>239 lee</td>
<td>jones</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>San Name Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>123 main str</td>
</tr>
<tr>
<td>234 forbes ave</td>
</tr>
</tbody>
</table>
Observations:

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

Observations:

• A: redundant:
• STUDENT intersection STAFF =

Observations:

• A: redundant:
• STUDENT intersection STAFF =

Observations:
Observations:

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

Double negation:
We'll see it again, later…

Relational operators

• .
• .
• .
• set union U
• set difference ‘-’
Other operators?

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

\[ \sigma_{\text{address} = \text{‘main str’}} \] (STUDENT)

<table>
<thead>
<tr>
<th>Ssn</th>
<th>Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>smith</td>
<td>main str</td>
</tr>
<tr>
<td>234</td>
<td>jones</td>
<td>forbes ave</td>
</tr>
</tbody>
</table>

Other operators?

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

\[ \sigma_{\text{condition}} \] (RELATION)

Selection - examples

- Find all ‘Smiths’ on ‘Forbes Ave’

\[ \sigma_{\text{name} = \text{‘Smith’} \land \text{address} = \text{‘Forbes ave’}} \] (STUDENT)

‘condition’ can be any boolean combination of ‘=’, ‘\(<\)’, ‘\(\leq\)’, ‘\(\geq\)’, ‘\(\neq\)’, ‘\(\>\)’...
Relational operators

- selection $\sigma_{\text{condition}} (R)$
- set union $R \cup S$
- set difference $R - S$

Relational operators

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

finds all the ‘ssn’ - removing duplicates

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>Ssn</th>
<th>Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>123</td>
<td>smith</td>
<td>main str</td>
</tr>
<tr>
<td></td>
<td>234</td>
<td>jones</td>
<td>forbes ave</td>
</tr>
</tbody>
</table>

Relational operators

Cascading: ‘find ssn of students on ‘forbes ave’

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

- selection \( \sigma_{\text{condition}} (R) \)
- projection \( \pi_{\text{attr-lst}} (R) \)
- set union \( R \cup S \)
- set difference \( R - S \)

Are we done yet?
Q: Give a query we can not answer yet!

A: any query across two or more tables, eg., "find names of students in 15-415"
Q: what extra operator do we need??
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!

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>TAKES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ssn</td>
<td>Name</td>
</tr>
<tr>
<td>123</td>
<td>smith</td>
</tr>
<tr>
<td>234</td>
<td>jones</td>
</tr>
</tbody>
</table>

Cartesian product

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

\[
\text{MALE} \times \text{FEMALE} = \text{M name} \times \text{F name} \\
\text{spike} \times \text{lassie} = \text{spike} \times \text{shiba} \\
\text{spot} \times \text{shiba} = \text{spot} \times \text{shiba}
\]

so what?

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

<table>
<thead>
<tr>
<th>STUDENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ssn</td>
</tr>
<tr>
<td>123</td>
</tr>
<tr>
<td>234</td>
</tr>
</tbody>
</table>
Cartesian product

\[ A = \sigma_{\text{student.stm=takes.stm}}(\text{STUDENT} \times \text{TAKES}) \]

<table>
<thead>
<tr>
<th>Ssn</th>
<th>Name</th>
<th>Address</th>
<th>ssn</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>smith</td>
<td>main str</td>
<td>123</td>
<td>16-415 A</td>
<td>A</td>
</tr>
<tr>
<td>234</td>
<td>jones</td>
<td>forbes ave</td>
<td>234</td>
<td>16-413 A</td>
<td>B</td>
</tr>
<tr>
<td>123</td>
<td>smith</td>
<td>main str</td>
<td>123</td>
<td>16-415 B</td>
<td>B</td>
</tr>
<tr>
<td>234</td>
<td>jones</td>
<td>forbes ave</td>
<td>234</td>
<td>16-413 B</td>
<td>B</td>
</tr>
</tbody>
</table>

Cartesian product

\[ A = \sigma_{\text{cid=15-413}}(\sigma_{\text{student.stm=takes.stm}}(\text{STUDENT} \times \text{TAKES})) \]

<table>
<thead>
<tr>
<th>Ssn</th>
<th>Name</th>
<th>Address</th>
<th>ssn</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>smith</td>
<td>main str</td>
<td>123</td>
<td>16-415 A</td>
<td>A</td>
</tr>
<tr>
<td>234</td>
<td>jones</td>
<td>forbes ave</td>
<td>234</td>
<td>16-413 A</td>
<td>B</td>
</tr>
<tr>
<td>123</td>
<td>smith</td>
<td>main str</td>
<td>123</td>
<td>16-415 B</td>
<td>B</td>
</tr>
<tr>
<td>234</td>
<td>jones</td>
<td>forbes ave</td>
<td>234</td>
<td>16-413 B</td>
<td>B</td>
</tr>
</tbody>
</table>

\[ A = \pi_{\text{name}}(\sigma_{\text{cid=15-413}}(\sigma_{\text{student.stm=takes.stm}}(\text{STUDENT} \times \text{TAKES}))) \]

<table>
<thead>
<tr>
<th>Ssn</th>
<th>Name</th>
<th>Address</th>
<th>ssn</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>smith</td>
<td>main str</td>
<td>123</td>
<td>16-415 A</td>
<td>A</td>
</tr>
<tr>
<td>234</td>
<td>jones</td>
<td>forbes ave</td>
<td>234</td>
<td>16-413 A</td>
<td>B</td>
</tr>
<tr>
<td>123</td>
<td>smith</td>
<td>main str</td>
<td>123</td>
<td>16-415 B</td>
<td>B</td>
</tr>
<tr>
<td>234</td>
<td>jones</td>
<td>forbes ave</td>
<td>234</td>
<td>16-413 B</td>
<td>B</td>
</tr>
</tbody>
</table>
FUNDAMENTAL
Relational operators

• selection \( \sigma_{\text{condition}} (R) \)
• projection \( \pi_{\text{attr-list}} (R) \)
• cartesian product \( \text{MALE} \times \text{FEMALE} \)
• set union \( R \cup S \)
• set difference \( R - S \)

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 \( \rho_{\pi} (R) \)
  – division \( R \div S \)

Joins

• Equijoin: \( R \bowtie_{a=b} S = \sigma_{a=b} (R \times S) \)
Cartesian product

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

<table>
<thead>
<tr>
<th>Ssn</th>
<th>Name</th>
<th>Address</th>
<th>ssn</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>smith</td>
<td>main str</td>
<td>123</td>
<td>16</td>
<td>415</td>
</tr>
<tr>
<td>234</td>
<td>jones</td>
<td>forbes ave</td>
<td>234</td>
<td>16</td>
<td>418</td>
</tr>
<tr>
<td>123</td>
<td>smith</td>
<td>main str</td>
<td>234</td>
<td>15</td>
<td>413</td>
</tr>
<tr>
<td>234</td>
<td>jones</td>
<td>forbes ave</td>
<td>234</td>
<td>15</td>
<td>413</td>
</tr>
</tbody>
</table>

Joins

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

Joins

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

- nat. join has 5 attributes \textit{STUDENT} \textbf{\&} \textit{TAKES}

<table>
<thead>
<tr>
<th>Ssn</th>
<th>Name</th>
<th>Address</th>
<th>Ssn</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>smith</td>
<td>main str</td>
<td>123</td>
<td>15-416</td>
<td>A</td>
</tr>
<tr>
<td>234</td>
<td>jones</td>
<td>forbes ave</td>
<td>123</td>
<td>15-415</td>
<td>A</td>
</tr>
<tr>
<td>123</td>
<td>smith</td>
<td>main str</td>
<td>234</td>
<td>15-413</td>
<td>B</td>
</tr>
<tr>
<td>234</td>
<td>jones</td>
<td>forbes ave</td>
<td>234</td>
<td>15-413</td>
<td>B</td>
</tr>
</tbody>
</table>

equi-join: 6 \textit{STUDENT} \textbf{\&} \textit{STUDENT} \textit{..\&} \textit{TAKES} \textit{..\&} \textit{TAKES}

Natural Joins - nit-picking

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

Overview - rel. algebra

- fundamental operators
- derived operators
  - joins etc
  - rename
  - division
- examples
Rename op.

- Q: why? \( \rho_{\text{after}}(\text{BEFORE}) \)
- 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) \( PC \bowtie PC \)

<table>
<thead>
<tr>
<th>p-id</th>
<th>c-id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>Tom</td>
</tr>
<tr>
<td>Peter</td>
<td>Mary</td>
</tr>
<tr>
<td>John</td>
<td>Tom</td>
</tr>
<tr>
<td>Tom</td>
<td></td>
</tr>
</tbody>
</table>

Rename op.

- first, WRONG attempt: \( PC \bowtie PC \)
  - (why? how many columns?)
- Second WRONG attempt: \( PC \bowtie_{\text{p-id}=PC\_p-id} PC \)
Rename op.

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

\[ \rho_{PC}(PC) \bowtie_{PC_{new} = PC_{old}} PC \]

Overview - rel. algebra

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

Division

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

**Division**

<table>
<thead>
<tr>
<th>SHIPMENT</th>
<th>ABOM</th>
<th>BAD S</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_1 p_1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s_2 p_1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s_3 p_2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s_5 p_3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Observations: ~reverse of cartesian product

It can be derived from the 5 fundamental operators (!!!)

How?

**Division**

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

\[ r + s = \pi_{(R \times S)}(r) \ast \pi_{(R \times S)}[(\pi_{(R \times S)}(r) \times s) - r] \]
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)

Division

- Answer:

\[ r \div s = \pi_{(r,s)}(r) - \pi_{(r,s)}[(\pi_{(r,s)}(r) \times s) - r] \]

All suppliers

All bad parts

Division

- Answer:

\[ r \div s = \pi_{(r,s)}(r) - \pi_{(r,s)}[(\pi_{(r,s)}(r) \times s) - r] \]

- All possible suspicious shipments
Division

- Answer:

\[ r / s = \Pi_{(R \setminus S)}(r) - \Pi_{(R \setminus S)}((\Pi_{(R \setminus S)}(r) \times s) - r) \]

all possible suspicious shipments that didn’t happen

Division

- Answer:

\[ r / s = \Pi_{(R \setminus S)}(r) - \Pi_{(R \setminus S)}((\Pi_{(R \setminus S)}(r) \times s) - r) \]

all suppliers who missed at least one suspicious shipment, i.e.: ‘good’ suppliers

Overview - rel. algebra

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

find names of students that take 15-415

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssn</td>
<td>c-id</td>
</tr>
<tr>
<td>name</td>
<td>c-name</td>
</tr>
<tr>
<td>address</td>
<td>units</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TAKES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssn</td>
</tr>
<tr>
<td>grade</td>
</tr>
</tbody>
</table>

Examples

• find names of students that take 15-415
Sample schema

find course names of ‘smith’

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>c-id</td>
</tr>
<tr>
<td>123</td>
<td>15-413</td>
</tr>
<tr>
<td>234</td>
<td>15-412</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TAKES</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
</tr>
<tr>
<td>123</td>
</tr>
<tr>
<td>234</td>
</tr>
</tbody>
</table>

Examples

• find course names of ‘smith’

\[
\pi_{\text{name}} \left( \sigma_{\text{name} = \text{smith}} \left( \text{STUDENT} \bowtie \text{TAKES} \bowtie \text{CLASS} \right) \right)
\]

• 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:

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

Examples

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

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

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
Sample schema

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>c-id</td>
</tr>
<tr>
<td>123</td>
<td>15-413</td>
</tr>
<tr>
<td>234</td>
<td>16-413</td>
</tr>
</tbody>
</table>

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)

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

Conclusions

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