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

C. Faloutsos
Relational tuple calculus

General Overview - rel. model

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

Overview - detailed

- rel. tuple calculus
  - why?
  - details
  - examples
  - equivalence with rel. algebra
  - more examples; ‘safety’ of expressions
- rel. domain calculus + QBE

Motivation

- Q: weakness of rel. algebra?
- A: procedural
  - describes the steps (i.e., ‘how’)
  - (still useful for query optimization)

Solution: rel. calculus

- describes what we want
- two equivalent flavors: ‘tuple’ and ‘domain’ calculus
- basis for SQL and QBE, resp.

Rel. tuple calculus (RTC)

- first order logic

(let P

‘Give me tuples ‘t’, satisfying predicate P - eg:

(let (A))
Details

• symbols allowed:
  \( \land, \lor, \neg \Rightarrow \)
  \( \geq, \leq \Rightarrow \leq, \geq \)
  \( (, ), \in \)

• quantifiers \( \forall, \exists \)

Specifically

• Atom
  \( r \in \text{TABLE} \)
  \( \{ \text{attr} \} \subseteq \text{const} \)
  \( \{ \text{attr} \} \subseteq \{ \text{attr}' \} \)

Specifically

• Formula:
  – atom
  – if \( P_1, P_2 \) are formulas, so are \( P_1 \land P_2, P_1 \lor P_2, \neg P \)
  – if \( P(s) \) is a formula, so are:
    \( \exists s(P(s)) \)
    \( \forall s(P(s)) \)

Specifically

• Reminders:
  – DeMorgan: \( P_1 \land P_2 \equiv \neg (\neg P_1 \lor \neg P_2) \)
  – implication: \( P_1 \Rightarrow P_2 \equiv \neg P_1 \lor P_2 \)
  – double negation:
    \( \forall s \in \text{TABLE}(P(s)) \equiv \neg \exists s \in \text{TABLE}(\neg P(s)) \)
    “every human is mortal : no human is immortal”

Reminder: our Mini-U db

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>Code</td>
</tr>
<tr>
<td>123</td>
<td>15-113</td>
</tr>
<tr>
<td>234</td>
<td>15-112</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TAKES</th>
<th>CODE</th>
<th>GRADE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>Code</td>
<td>Grade</td>
</tr>
<tr>
<td>123</td>
<td>15-113</td>
<td>A</td>
</tr>
<tr>
<td>234</td>
<td>15-113</td>
<td>B</td>
</tr>
</tbody>
</table>

Examples

• find all student records

\( \{ t | t \in \text{STUDENT} \} \)

output topic of type “STUDENT”
Examples

- (selection) find student record with ssn=123

\{t \in \text{STUDENT} \land t[\text{ssn}] = 123\}

- (projection) find name of student with ssn=123

\{t \in \text{STUDENT} \land t[\text{ssn}] = 123\}

- (projection) find name of student with ssn=123

\{t \in \text{STUDENT} \land t[\text{name}] = s[\text{name}]\}

"t" has only one column

"Tracing"

\{t \exists s \in \text{STUDENT} (s[\text{ssn}] = 123 \land t[\text{name}] = s[\text{name}]\}

Examples cont'd

- (union) get records of both PT and FT students

CMU - 15-415
Examples cont’d

- (union) get records of both PT and FT students

\[ \{ t \mid t \in FT\_STUDENT \lor t \in PT\_STUDENT \} \]

Examples

- difference: find students that are not staff

\[ \{ t \mid t \in STUDENT \land t \not\in STAFF \} \]

Cartesian product

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

\[
\begin{array}{c}
\text{MALE} \\
\text{name} \\
\text{spike} \\
\text{spot}
\end{array}
\times
\begin{array}{c}
\text{FEMALE} \\
\text{name} \\
\text{leslie} \\
\text{shiba}
\end{array}
= \begin{array}{c}
\text{Name} \\
\text{spike leslie} \\
\text{spot leslie} \\
\text{spot shiba}
\end{array}
\]

Cartesian product

- find all the pairs of (male, female)

\[ \{ t \mid \exists m \in MALE \land \exists f \in FEMALE \land t[m - name] = m[name] \land t[f - name] = f[name] \} \]

‘Proof’ of equivalence

- rel. algebra <-> rel. tuple calculus
Overview - detailed

- rel. tuple calculus
  - why?
  - details
  - examples
  - equivalence with rel. algebra
  - more examples: "safety" of expressions
- re. domain calculus + QBE

More examples

- join: find names of students taking 15-415

Reminder: our Mini-U db

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>CLASS</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>

<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>

More examples

- join: find names of students taking 15-415

\[ \{ t \exists s \in STUDENT \land \exists e \in TAKES (s[ssn] = e[ssn] \land t[name] = s[name] \land e[c-id] = 15 - 415) \} \]

More examples

- join: find names of students taking 15-415

\[ \{ t \exists s \in STUDENT \land \exists e \in TAKES (s[ssn] = e[ssn] \land t[name] = s[name] \land e[c-id] = 15 - 415) \} \]

More examples

- 3-way join: find names of students taking a 2-unit course
Reminder: our Mini-U db

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>c-name</td>
</tr>
<tr>
<td>Smith</td>
<td>A</td>
</tr>
<tr>
<td>Jones</td>
<td>B</td>
</tr>
</tbody>
</table>

More examples

- 3-way join: find names of students taking a 2-unit course

\[
\pi_{\text{name}}(\sigma_{\text{units}=2}\langle \text{STUDENT} > \text{TAKES} > \text{CLASS} ))
\]

Even more examples:

- self-joins: find Tom's grandparent(s)

\[
\{t \mid \exists p \in \text{PC} \land \exists q \in \text{PC} \\
(p \cdot \text{id} = q \cdot \text{id}) \land \\
(p \cdot \text{id} = t \cdot \text{id}) \land \\
(q \cdot \text{id} = \text{'Tom'})\}
\]

Hard examples: DIVISION

- find suppliers that shipped all the ABOMB parts

\[
\begin{array}{c|c|c}
\text{SHIPMENT} & \text{ABOMB} & \text{BAD-S} \\
\hline
s1 & p1 & s1 \\
\hline
s2 & p1 & s1 \\
\hline
s3 & p2 & s1 \\
\hline
s4 & p3 & s1 \\
\end{array}
\]
Hard examples: DIVISION

- find suppliers that shipped all the ABOMB parts

\[ t \forall p (p \in ABOMB \Rightarrow (\exists s \in SHIPMENT (t[s\#] = s[p\#] \wedge s[p\#] = p[p\#]))) \]

General pattern

- three equivalent versions:
  - 1) if it's bad, he shipped it
    \[ t \forall p (p \in ABOMB \Rightarrow P(t)) \]
  - 2) either it was good, or he shipped it
    \[ t \forall p (p \in ABOMB \vee P(t)) \]
  - 3) there is no bad shipment that he missed
    \[ t \neg \exists p (p \in ABOMB \wedge \neg P(t)) \]

More on division

- find (SSNs of) students that take all the courses that ssn=123 does (and maybe even more)
  - find students 's' so that
  - if 123 takes a course \(\Rightarrow\) so does 's'

Safety of expressions

- FORBIDDEN: \( \{ t \forall \in \text{STUDENT} \} \)
  - It has infinite output!!
- Instead, always use
  \( \{ t \in \text{SOME-TABLE} \} \)

More on division

- find students that take all the courses that ssn=123 does (and maybe even more)
  \[ \forall t (t \in \text{TAKES} \wedge t[\text{ssn}] = 123) \Rightarrow \exists t \in \text{TAKES} (t[c-id] = t[c-id] \wedge t[\text{ssn}] = c[\text{ssn}] \}) \]

Overview - conclusions

- relational tuple calculus: DECLARATIVE
  - dfn
  - details
  - equivalence to relational algebra
- relational domain calculus + QBE