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

C. Faloutsos & A. Pavlo
Lecture#5: Relational 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 (ie., ‘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.
- Useful for proofs (see query optimization, later)

Rel. tuple calculus (RTC)

- first order logic

\{t | P(t)\}

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

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

Details

- symbols allowed:
  \( \land, \lor, \neg, \Rightarrow \)
  \( >, <, =, \neq, \leq, \geq, \)
  \( (, ), \in \)

- quantifiers \( \forall, \exists \)

Specifically

- Atom

\( t \in \text{TABLE} \)

\( t.\text{attr} \leq \text{const} \)

\( t.\text{attr} \leq s.\text{attr}' \)
Specifically

- **Formula:**
  - atom
  - if $P_1$, $P_2$ are formulas, so are $P_1 \land P_2; P_1 \lor P_2$
  - if $P(s)$ is a formula, so are $\exists s(P(s)); \forall s(P(s))$

Specifically

- **Reminders:**
  - De Morgan: $P_1 \land P_2 = \neg(\neg P_1 \lor \neg P_2)$
  - implication: $P_1 \Rightarrow P_2 = \neg P_1 \lor P_2$
  - double negation:
    \[ \forall s \in TABLE \ (P(s)) = \neg \exists s \in TABLE \ (\neg P(s)) \]
    
    ‘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>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>
<th>c-id</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>15-413</td>
<td>A</td>
</tr>
<tr>
<td>234</td>
<td>15-413</td>
<td>B</td>
</tr>
</tbody>
</table>

Examples

- **find all student records**

\[ \{ t | t \in STUDENT \} \]

 output tuple of type ‘STUDENT’
(Goal: evidence that RTC = RA)

- Full proof: complicated
- We’ll just show examples of the 5 RA fundamental operators, and how RTC can handle them
- (Quiz: which are the 5 fundamental op’s?)

**FUNDAMENTAL operators**

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

**Examples**

- (selection) find student record with ssn=123

\( \{ t \mid t \in \text{STUDENT} \wedge t.ssn = 123 \} \)
Examples

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

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

Examples

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

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

- ‘t’ has only one column

Examples cont’d

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

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

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

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

- ‘t’ has only one column
Examples cont’d

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

\{ t \mid t \in \text{FT\_STUDENT} \lor t \in \text{PT\_STUDENT} \}

Examples

• difference: find students that are not staff

\{ t \mid t \in \text{STUDENT} \land t \notin \text{STAFF} \}

(assuming that \text{STUDENT} and \text{STAFF} are union-compatible)

Cartesian product

• eg., dog-breeding: \text{MALE} \times \text{FEMALE}

• gives all possible couples

\[
\begin{array}{ccc}
\text{MALE} & \times & \text{FEMALE} \\
\hline
\text{name} & \times & \text{name} \\
\text{spike} & \times & \text{lassie} \\
\text{spot} & \times & \text{shiba} \\
\end{array}
\]
Cartesian product

- find all the pairs of (male, female)

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

‘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

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>c-id</td>
</tr>
<tr>
<td>Address</td>
<td>c-name</td>
</tr>
<tr>
<td>123</td>
<td>15-413</td>
</tr>
<tr>
<td>smith</td>
<td>s.e.</td>
</tr>
<tr>
<td>234</td>
<td>15-412</td>
</tr>
<tr>
<td>jones</td>
<td>o.s.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TAKES</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>c-id</td>
<td>grade</td>
</tr>
<tr>
<td>123</td>
<td>15-413</td>
<td>A</td>
</tr>
<tr>
<td>234</td>
<td>15-413</td>
<td>B</td>
</tr>
</tbody>
</table>

More examples

• join: find names of students taking 15-415

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

More examples

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

(Remember: ‘SPJ’)
Reminder: our Mini-U db

<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>
<tr>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>123</td>
<td>15-413</td>
</tr>
<tr>
<td>Smith</td>
<td>s.e.</td>
</tr>
<tr>
<td>Main</td>
<td>2</td>
</tr>
<tr>
<td>Str</td>
<td></td>
</tr>
<tr>
<td>234</td>
<td>15-412</td>
</tr>
<tr>
<td>Jones</td>
<td>o.s.</td>
</tr>
<tr>
<td>Forbes</td>
<td>2</td>
</tr>
<tr>
<td>Ave</td>
<td></td>
</tr>
</tbody>
</table>

More examples

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

\[
\{ t | \exists s \in \text{STUDENT} \land \exists e \in \text{TAKES} \\
\exists c \in \text{CLASS} ( s.\text{ssn} = e.\text{ssn} \land \\
e.c = c.c \land t.\text{name} = s.\text{name} \land \\
c.\text{units} = 2) \}
\]

- 3-way join: find names of students taking a 2-unit course - in rel. algebra??

\[
\pi_{\text{name}} (\sigma_{\text{units}=2} (\text{STUDENT} \bowtie \text{TAKES} \bowtie \text{CLASS}))
\]
Even more examples:

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

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

Hard examples: DIVISION

- find suppliers that shipped all the ABOMB parts

\[
\{ t | \forall p \in ABOMB \Rightarrow ( \exists s \in SHIPMENT ( t.s# = s.s# \land s.p# = p.p# ) ) \}
\]
General pattern

• three equivalent versions:
  – 1) if it’s bad, he shipped it
    \( \{ t \mid \forall p (p \in ABOMB \Rightarrow (P(t))) \} \)
  – 2) either it was good, or he shipped it
    \( \{ t \mid \forall p (p \not\in ABOMB \vee (P(t))) \} \)
  – 3) there is no bad shipment that he missed
    \( \{ t \mid \exists p (p \in ABOMB \land (\neg P(t))) \} \)

General pattern

• three equivalent versions:
  – 1) if it’s bad, he shipped it
    \( \{ t \mid \forall p (p \in ABOMB \Rightarrow (P(t))) \} \)
  – 2) either it was good, or he shipped it
    \( \{ t \mid \forall p (p \not\in ABOMB \vee (P(t))) \} \)
  – 3) there is no bad shipment that he missed
    \( \{ t \mid \exists p (p \in ABOMB \land (\neg P(t))) \} \)

a \Rightarrow b is the same as \neg a \lor b

• If a is true, b must be true for the implication to be true. If a is true and b is false, the implication evaluates to false.
• If a is not true, we don’t care about b, the expression is always true.
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 => so does ‘s’

More on division

• find students that take all the courses that ssn=123 does (and maybe even more)
  \[ \{ o \mid \forall t((t \in TAKES \land t.ssn = 123) \implies \exists t1 \in TAKES( t1.c - id = t.c - id \land t1.ssn = o.ssn)) \}\]

Safety of expressions

• FORBIDDEN: \[ \{ t \mid t \notin STUDENT \} \]
  It has infinite output!!
• Instead, always use
  \[ \{ t \mid \ldots t \in SOME - TABLE \} \]

Overview - conclusions

• rel. tuple calculus: DECLARATIVE
  – dfn
  – details
  – equivalence to rel. algebra
• rel. domain calculus + QBE
General Overview

- relational model
- Formal query languages
  - relational algebra
  - rel. tuple calculus
  - rel. domain calculus

Rel. domain calculus (RDC)

- Q: why?
- A: slightly easier than RTC, although equivalent - basis for QBE.
- idea: domain variables (w/ F.O.L.) - eg:
  - ‘find STUDENT record with ssn=123’

Rel. Dom. Calculus

- find STUDENT record with ssn=123’

\{ <s,n,a >| <s,n,a> \in STUDENT \land s =123 \}

Details

- Like R.T.C - symbols allowed:
  - \&, \lor, \neg, \Rightarrow
  - >, <, =, \neq, \leq, \geq
  - (), \in

- quantifiers \forall, \exists
Details

- but: domain (= column) variables, as opposed to tuple variables, e.g:

\[ \langle s, n, a \rangle \in \text{STUDENT} \]

\[
\begin{array}{ccc}
\text{ssn} & \text{name} & \text{address} \\
\end{array}
\]

Reminder: our Mini-U db

<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>
<tr>
<td>123</td>
<td>15-413</td>
</tr>
<tr>
<td>smith</td>
<td>main str</td>
</tr>
<tr>
<td>234</td>
<td>15-412</td>
</tr>
<tr>
<td>jones</td>
<td>forbes ave</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 all student records

\[ \{ \langle s, n, a \rangle | \langle s, n, a \rangle \in \text{STUDENT} \} \]

RTC: \[ \{ t | t \in \text{STUDENT} \} \]

Examples

- (selection) find student record with ssn=123
(‘Proof’ of RDC = RA)

• Again, we show examples of the 5 fundamental operators, in RDC

Examples

• (selection) find student record with ssn=123
  \[ \{ \langle 123, n, a \rangle | \langle 123, n, a \rangle \in \text{STUDENT} \} \]
  \[ \text{RTC:} \quad \{ t | t \in \text{STUDENT} \land t.\text{ssn} = 123 \} \]

or

• (selection) find name of student with ssn=123
  \[ \{ \langle s, n, a \rangle | \langle s, n, a \rangle \in \text{STUDENT} \land s = 123 \} \]
  \[ \text{RTC:} \quad \{ t | t \in \text{STUDENT} \land t.\text{ssn} = 123 \} \]
Examples

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

\[\{< n >| \exists a(<123, n, a> \in STUDENT)\}\]

need to ‘restrict’ “a”

**RTC:** \[\{t | \exists s \in STUDENT(s.ssn = 123 \land t.name = s.name)\}\]

Examples cont’d

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

**RTC:** \[\{t | t \in FT\_STUDENT \lor t \in PT\_STUDENT\}\]

Examples cont’d

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

\[\{< s, n, a >| < s, n, a > \in FT\_STUDENT \lor < s, n, a > \in PT\_STUDENT\}\]

Examples

- difference: find students that are not staff

**RTC:** \[\{t | t \in STUDENT \land t \notin STAFF\}\]
Examples

• difference: find students that are not staff

\{< s, n, a >|< s, n, a > \in \text{STUDENT} \land
\quad < s, n, a > \notin \text{STAFF}\}

Cartesian product

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

\[
\begin{array}{c|c}
\text{MALE} & \text{FEMALE} \\
\hline
\text{name} & \text{name} \\
\text{spike} & \text{lassie} \\
\text{spot} & \text{shiba} \\
\end{array}
\times
\begin{array}{c|c}
\text{M.name} & \text{F.name} \\
\text{spike} & \text{lassie} \\
\text{spike} & \text{shiba} \\
\text{spot} & \text{lassie} \\
\text{spot} & \text{shiba} \\
\end{array}
= 
\begin{array}{c|c}
\text{M.name} & \text{F.name} \\
\text{spike} & \text{lassie} \\
\text{spike} & \text{shiba} \\
\text{spot} & \text{lassie} \\
\text{spot} & \text{shiba} \\
\end{array}
\]

Cartesian product

• find all the pairs of (male, female) - RTC:

\[
\{t | \exists m \in \text{MALE} \land
\quad \exists f \in \text{FEMALE}
\quad \quad t.m \text{name} = m.\text{name} \land
\quad t.f \text{name} = f.\text{name}\}
\]

Cartesian product

• find all the pairs of (male, female) - RDC:
Cartesian product

- find all the pairs of (male, female) - RDC:

\[ \{ < m, f > | < m > \in MALE \land < f > \in FEMALE \} \]

‘Proof’ of equivalence

- rel. algebra <-> rel. domain calculus
  <- rel. tuple calculus

Overview - detailed

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

More examples

- join: find names of students taking 15-415
Reminder: our Mini-U db

<table>
<thead>
<tr>
<th>STUDENT</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ssn</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>

<table>
<thead>
<tr>
<th>CLASS</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>c-id</td>
<td>c-name</td>
<td>units</td>
<td></td>
</tr>
<tr>
<td>15-413</td>
<td>s.e.</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>15-412</td>
<td>o.s.</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>takes</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>c-id</td>
<td>grade</td>
</tr>
<tr>
<td>123</td>
<td>15-413</td>
<td>A</td>
</tr>
<tr>
<td>234</td>
<td>15-413</td>
<td>B</td>
</tr>
</tbody>
</table>

More examples

- join: find names of students taking 15-415 - in RTC

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

More examples

- join: find names of students taking 15-415 - in RDC

\[
\{ n \mid \exists s \exists a \exists g (n, s, a) \in \text{STUDENT} \\
\land n, 15-415, g \in \text{TAKES} \}
\]

Sneak preview of QBE:

\[
\{ n \mid \exists s \exists a \exists g (n, s, a) \in \text{STUDENT} \\
\land n, 15-415, g \in \text{TAKES} \}
\]
Sneak preview of QBE:

- very user friendly
- heavily based on RDC
- very similar to MS Access interface and pgAdminIII

More examples

- 3-way join: find names of students taking a 2-unit course - in RTC:
  \[ \{t \mid \exists s \in \text{STUDENT} \land \exists e \in \text{TAKES} \exists c \in \text{CLASS}(s.ssn = e.ssn \land e.c-id = c.c-id \land t.name = s.name \land c.units = 2)\} \]

Reminder: our Mini-U db

More examples

- 3-way join: find names of students taking a 2-unit course
  \{< n >|............
  < s,n,a >\in \text{STUDENT} \land 
  < s,c,g >\in \text{TAKES} \land 
  < c,cn,2 >\in \text{CLASS}\}
More examples

• 3-way join: find names of students taking a 2-unit course
  \[<n>\exists s,a,c,g,cn(\]
  \[<s,n,a>\in\text{STUDENT} \land \]
  \[<s,c,g>\in\text{TAKES} \land \]
  \[<c,cn,2>\in\text{CLASS} \]
  \})

Even more examples:

• self-joins: find Tom’s grandparent(s)

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

Even more examples:

• self-joins: find Tom’s grandparent(s)

\[\{t\mid\exists p \in PC \land \exists q \in PC \]
\[\quad (p.c - id = q.p - id \land \]
\[\quad p.p - id = t.p - id \land \]
\[\quad q.c - id = "Tom")\}
Even more examples:

• self-joins: find Tom’s grandparent(s)

\[ \{< g >| \exists p(< g, p >\in PC \land < p," Tom" >\in PC)\} \]

Hard examples: DIVISION

• find suppliers that shipped all the ABOMB parts

\[ \{t \mid \forall p(p \in ABOMB \Rightarrow (\exists s \in SHIPMENT (t.s# = s.s# \land s.p# = p.p#)))\} \]

Hard examples: DIVISION

• find suppliers that shipped all the ABOMB parts

\[ \{< s >| \forall p(< p >\in ABOMB \Rightarrow (\exists s \in SHIPMENT (t.s# = s.s# \land s.p# = p.p#)))\} \]
More on division

• find students that take all the courses that ssn=123 does (and maybe even more)

\{ o \mid \forall t((t \in \text{TAKE}\land t.\text{ssn} = 123) \Rightarrow \\
\exists t1 \in \text{TAKE}\{
\ t1.c - id = t.c - id \land \\
\ t1.\text{ssn} = o.\text{ssn}
\})\}

Safety of expressions

• similar to RTC
• FORBIDDEN:

\{< s,n,a >\mid < s,n,a >\notin \text{STUDENT} \}

Overview - detailed

• rel. domain calculus + QBE
  – dfn
  – details
  – equivalence to rel. algebra
Fun Drill: Your turn …

• Schema:
  Movie(title, year, studioName)
  ActsIn(movieTitle, starName)
  Star(name, gender, birthdate, salary)

Your turn …

• Queries to write in TRC:
  – Find all movies by Paramount studio
  – … movies starring Kevin Bacon
  – Find stars who have been in a film w/Kevin Bacon
  – Stars within six degrees of Kevin Bacon*
  – Stars connected to K. Bacon via any number of films**

  * Try two degrees for starters          ** Good luck with this one!

Answers …

• Find all movies by Paramount studio

\{M \mid M \in \text{Movie} \land
\exists A \in \text{ActsIn}(A.\text{movieTitle} = M.\text{title} \land
A.\text{starName} = \text{‘Bacon’})\}

Answers …

• Movies starring Kevin Bacon

\{\text{M} \mid \text{M} \in \text{Movie} \land
\exists A \in \text{ActsIn}(A.\text{movieTitle} = \text{M.title} \land
A.\text{starName} = \text{‘Bacon’})\}
Answers ...

- Stars who have been in a film w/Kevin Bacon
  \( \{ S | S \in \text{Star} \land \exists A \in \text{ActsIn}(A.\text{starName} = S.\text{name} \land \exists A2 \in \text{ActsIn}(A2.\text{movieTitle} = A.\text{movieTitle} \land A2.\text{starName} = \text{‘Bacon’}) ) \} \)

Two degrees:

- Stars within six degrees of Kevin Bacon
  \( \{ S | S \in \text{Star} \land \exists A \in \text{ActsIn}(A.\text{starName} = S.\text{name} \land \exists A2 \in \text{ActsIn}(A2.\text{movieTitle} = A.\text{movieTitle} \land \exists A3 \in \text{ActsIn}(A3.\text{starName} = A2.\text{starName} \land \exists A4 \in \text{ActsIn}(A4.\text{movieTitle} = A3.\text{movieTitle} \land A4.\text{starName} = \text{‘Bacon’}) ) ) \} \)

Two degrees:
Answers …

• Stars connected to K. Bacon via any number of films

• Sorry … that was a trick question
  – Not expressible in relational calculus!!

• What about in relational algebra?
  – No – RA, RTC, RDC are equivalent

Expressive Power

• Expressive Power (Theorem due to Codd):
  – Every query that can be expressed in relational algebra can be expressed as a safe query in RDC / RTC; the converse is also true.

• Relational Completeness:
  Query language (e.g., SQL) can express every query that is expressible in relational algebra/calculus.
  (actually, SQL is more powerful, as we will see…)

Summary

• The relational model has rigorously defined query languages — simple and powerful.
• Relational algebra is more operational/procedural
  – useful as internal representation for query evaluation plans
• Relational calculus is declarative
  – users define queries in terms of what they want, not in terms of how to compute it.

Summary - cnt’d

• Several ways of expressing a given query
  – a query optimizer chooses best plan.
• Algebra and safe calculus: same expressive power
  => relational completeness.