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

C. Faloutsos
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 (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

\{t \mid P(t)\}

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

\{t \mid t \in \text{STUDENT}\}
Details

- symbols allowed:
  \( \land, \lor, \neg, \Rightarrow, \geq, \leq, =, \neq, \in, (, \), \exists, \forall \)
- quantifiers \( \forall, \exists \)

Specifically

- Atom
  
  \[ t \in \text{TABLE} \]
  
  \[ t.\text{attr} = \text{const} \]
  
  \[ t.\text{attr} = \text{attr}' \]

Specifically

- Formula:
  - atom
  - if P1, P2 are formulas, so are \( P_1 \land P_2, P_1 \lor P_2 \ldots \)
  - if \( P(s) \) is a formula, so are \( \exists s(P(s)) \)
  - \( \forall s(P(s)) \)
Specifically

- Reminders:
  - DeMorgan: $P_1 \land P_2 = \neg(\neg P_1 \lor \neg P_2)$
  - implication: $P_1 \Rightarrow P_2 = \neg P_1 \lor P_2$
  - double negation:
    $\forall s \in \text{TABLE} (P(s)) = \neg \exists s \in \text{TABLE} (\neg P(s))$
    
    '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>Name</td>
</tr>
<tr>
<td>123</td>
<td>smith</td>
</tr>
<tr>
<td>234</td>
<td>jones</td>
</tr>
</tbody>
</table>

Examples

- find all student records

  \[
  \{ t \mid t \in \text{STUDENT} \}
  \]

  output tuple of type 'STUDENT'
Examples

• (selection) find student record with \( \text{ssn}=123 \)

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

Examples

• (projection) find \textbf{name} of student with \( \text{ssn}=123 \)

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

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

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

\`t` has only one column

'Tracing'

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

Examples cont’d

• (union) get records of both PT and FT students
Examples cont’d

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

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

Examples

- difference: find students that are not staff

(assuming that STUDENT and STAFF are union-compatible)

Examples

- difference: find students that are not staff

\[ \{ t | t \in STUDENT \land t \notin STAFF \} \]
Cartesian product

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

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

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

Cartesian product

- find all the pairs of (male, female)

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

‘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
More examples

• join: find names of students taking 15-415

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

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

| STUDENT | | CLASS |
|---------|------------------|
| ssn     | Name            | c-id   | c-name | units |
| 123     | smith           | 15-413 | s.e.   | 2      |
| 234     | jones           | 15-412 | o.s.   | 2      |

| TAKES | | grade |
|-------|------------------|
| GSN   | c-id     | grade |
| 123   | 15-413   | A     |
| 234   | 15-413   | B     |

More examples

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

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

More examples

- 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>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)

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

Hard examples: DIVISION

- find suppliers that shipped all the ABOMB parts

<table>
<thead>
<tr>
<th>SHIPMENT</th>
<th>ABOMB</th>
<th>BAD_S</th>
</tr>
</thead>
<tbody>
<tr>
<td>s#</td>
<td>p#</td>
<td>s#</td>
</tr>
<tr>
<td>s1</td>
<td>p1</td>
<td>s1</td>
</tr>
<tr>
<td>s2</td>
<td>p1</td>
<td>s1</td>
</tr>
<tr>
<td>s3</td>
<td>p1</td>
<td>s1</td>
</tr>
<tr>
<td>s5</td>
<td>p3</td>
<td>s1</td>
</tr>
</tbody>
</table>
Hard examples: DIVISION

• find suppliers that shipped all the ABOMB parts

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

General pattern

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

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

\[
\begin{array}{c|c|c|c|c|c|c|c}
 a & b & T & T & T & T & F & F \\
 T & T & F & F & T & T & T & T \\
 F & T & T & T & F & F & T & T \\
\end{array}
\]

• 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)
  \( \{ o \mid \forall t ((t \in TAKES \land t.ssn = 123) \implies \exists \bar{t} \in TAKES( \begin{array}{l} \bar{t}.c - id = t.c - id \land \\
                     \bar{t}.ssn = o.ssn \end{array} ) \} \) 

Safety of expressions

• FORBIDDEN: \( \{ t \mid t \not\in \text{STUDENT} \} \)
  It has infinite output!!
• Instead, always use
  \( \{ t \mid \ldots t \in \text{SOME - TABLE} \} \)
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:
  \(\land, \lor, \neg, \implies\)
  \(>, <, =, \neq, \leq, \geq,\)
  \((, ), \in\)

• quantifiers \(\forall, \exists\)

Details

• but: domain (= column) variables, as opposed to tuple variables, eg:

\(< s, n, a >\in STUDENT\)

\(\text{ssn, name, address}\)
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></td>
<td>units</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>grade</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>15-413</td>
</tr>
</tbody>
</table>

Examples

• find all student records

\{<s,n,a>|<s,n,a> \in \text{STUDENT}\}

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

Examples

• (selection) find student record with ssn=123
Examples

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

Examples

- (projection) find name of student with ssn=123
  \[ \langle n \rangle | \langle 123, n, a \rangle \in \text{STUDENT} \]

Examples

- (projection) find name of student with ssn=123
  \[ \langle n \rangle | \exists a (\langle 123, n, a \rangle \in \text{STUDENT}) \]
  need to ‘restrict’ “a”
  RTC:
  \[ \{ t | \exists x \in \text{STUDENT} (x.ssn = 123 \land t.name = x.name) \} \]
Examples cont’d

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

\[ \text{RTC: } \{ t \mid t \in FT\_\text{STUDENT} \lor t \in PT\_\text{STUDENT} \} \]

Examples cont’d

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

\[ \{ (s, n, a) \mid (s, n, a) \in FT\_\text{STUDENT} \lor (s, n, a) \in PT\_\text{STUDENT} \} \]

Examples

• difference: find students that are not staff

\[ \text{RTC: } \{ t \mid t \in \text{STUDENT} \land t \notin \text{STAFF} \} \]
Examples

• difference: find students that are not staff

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

Cartesian product

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

<table>
<thead>
<tr>
<th>MALE</th>
<th>FEMALE</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>name</td>
</tr>
<tr>
<td>spike</td>
<td>lassie</td>
</tr>
<tr>
<td>spot</td>
<td>shiba</td>
</tr>
</tbody>
</table>

\( \times \)

Cartesian product

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

\{ t | \exists m \in \text{MALE} \land
\exists f \in \text{FEMALE}
\land
m \text{- name} = m.name \land
f \text{- name} = f.name \}
Cartesian product

• find all the pairs of \((\text{male, female})\) - RDC:

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

‘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>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>

More examples

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

\{(t | \exists s \in \text{STUDENT} \\
\quad \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 >| \exists a \exists g(< s, n, a > \in \text{STUDENT} \\
\land < s, 15 - 415, g > \in \text{TAKES})\}

Sneak preview of QBE:

\{< n >| \exists a \exists g(< s, n, a > \in \text{STUDENT} \\
\land < s, 15 - 415, g > \in \text{TAKES})\}

Sneak preview of QBE:

- very user friendly
- heavily based on RDC
- very similar to MS Access interface
More examples

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

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

Reminder: our Mini-U db

<table>
<thead>
<tr>
<th>x</th>
<th>p</th>
<th>y</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>STUDENT</td>
<td>CLASS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ssn</td>
<td>Name</td>
<td>Address</td>
<td>c-id</td>
</tr>
<tr>
<td>123</td>
<td>smith</td>
<td>main str</td>
<td>16-413</td>
</tr>
<tr>
<td>234</td>
<td>jones</td>
<td>forbes ave</td>
<td>16-412</td>
</tr>
</tbody>
</table>

More examples

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

\[
\{< n > \mid \\
< s, n, a > \in \text{STUDENT} \land \\
< s, c, g > \in \text{TAKES} \land \\
< c, cn, 2 > \in \text{CLASS}\}
\]
More examples

• 3-way join: find names of students taking a 2-unit course
  \(\{n \mid \exists s, a, c, g, cn(\)
  < 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>c-id</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)

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

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

\[ \exists p \in PC \land \exists q \in PC \quad \{ g > \exists p(\langle g, p \rangle \in PC \land p, p = id \land p, q = id \land q, c = id = "Tom") \} \]

Even more examples:

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

\[ \{ g > \exists p(\langle g, p \rangle \in PC \land p, p = id \land p, q = id \land q, c = id = "Tom") \} \]

Hard examples: DIVISION

- find suppliers that shipped all the ABOMB parts

<table>
<thead>
<tr>
<th>SHIPMENT</th>
<th>ABOMB</th>
<th>BAD_S</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>p1</td>
<td>s1</td>
</tr>
<tr>
<td>s2</td>
<td>p1</td>
<td></td>
</tr>
<tr>
<td>s3</td>
<td>p1</td>
<td></td>
</tr>
<tr>
<td>s4</td>
<td>p2</td>
<td></td>
</tr>
<tr>
<td>s5</td>
<td>p3</td>
<td></td>
</tr>
</tbody>
</table>

\[ s1 + s2 = s1 \]
Hard examples: DIVISION

- find suppliers that shipped all the ABOMB parts

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

More on division

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

\[
\{ o \mid \forall t ((t \in \text{TAKES} \land t.ssн = 123) \Rightarrow (\exists \mathbf{l} \in \text{TAKES} (t1.c.id = t.c.id \land t1.ssн = o.ssн)) \}
\]
More on division

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

\{< s >| \forall c (\exists g (\langle 123, c, g \rangle \in \text{TAKES}) \Rightarrow \\
\exists g' (\langle s, c, g' \rangle \in \text{TAKES})}\}

Safety of expressions

• similar to RTC
• FORBIDDEN:

\{< s, n, a >|< 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 | M ∈ Movie ∧ M.studioName = ‘Paramount’ \}
Answers …

• Movies starring Kevin Bacon

\{M \mid M \in \text{Movie} \land \exists A \in \text{ActsIn}(A.csvTitle = M.title \land A.starName = 'Bacon')\}\}

Answers …

• Stars who have been in a film w/Kevin Bacon

\{S \mid S \in \text{Star} \land \exists A \in \text{ActsIn}(A.csvName = S.name \land \exists A2 \in \text{ActsIn}(A2.csvTitle = A.csvTitle \land A2.starName = 'Bacon')\}\}

Answers …

• Stars within six degrees of Kevin Bacon

\{S \mid S \in \text{Star} \land \exists A \in \text{ActsIn}(A.csvName = S.name \land \exists A2 \in \text{ActsIn}(A2.csvTitle = A.csvTitle \land \exists A3 \in \text{ActsIn}(A3.csvName = A2.csvName \land \exists A4 \in \text{ActsIn}(A4.csvTitle = A3.csvTitle \land A4.csvName = 'Bacon'))\}\}
Two degrees:

S: name …

A3: movie star

A4: movie star

Two degrees:

S: name …

A1: movie star

A2: movie star

A3: movie star

A4: movie star

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 DRC / TRC;  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 should choose the most efficient version.
- Algebra and safe calculus have same expressive power
  - leads to the notion of relational completeness.