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

\( \{ t \mid P(t) \} \)

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

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

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

- quantifiers $\forall$, $\exists$

Specifically

- Atom
  - $t \in \text{TABLE}$
  - $t.\text{attr} \approx \text{const}$
  - $t.\text{attr} \approx s.\text{attr}$

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>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
  \[ \{ t \mid t \in STUDENT \} \]
  
  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
Relational operators

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

Examples

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

- (selection) find student record with ssn=123

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

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

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

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

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

'\( t \)' has only one column
‘Tracing’

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

\[
\begin{array}{ccc}
\text{Name} & \text{STUDENT} & \text{Address} \\
\text{Jones} & 123 \text{Smith} & \text{Main Str} \\
\end{array}
\]

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

(assuming that STUDENT and STAFF are union-compatible)

\{ t \mid t \in \text{STUDENT} \land t \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>

<table>
<thead>
<tr>
<th></th>
<th>name</th>
<th>F.name</th>
</tr>
</thead>
<tbody>
<tr>
<td>spike</td>
<td>lassie</td>
<td></td>
</tr>
<tr>
<td>spot</td>
<td>shiba</td>
<td></td>
</tr>
</tbody>
</table>
Cartesian product

- find all the pairs of (male, female)

\{ t | \exists m \in MALE \land \exists f \in FEMALE \\
\quad t.m - name = m.name \land \\
\quad t.f - name = f.name \}
More examples

• join: find names of students taking 15-415

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

• join: find names of students taking 15-415

\{t | \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)\}

(Remember: ‘SPJ’)

More examples

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

Reminder: our Mini-U db
More examples

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

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

projection

selection

More examples

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

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

```
{t | ∃p ∈ PC ∧ ∃q ∈ PC
  ( p.c - id = q.p - id ∧
    p.p - id = t.p - id ∧
    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 p1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s2 p1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1 p2</td>
<td></td>
<td>s1</td>
</tr>
<tr>
<td>s3 p1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s5 p3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Hard examples: DIVISION

• find suppliers that shipped all the ABOMB parts

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

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
  \[ \{ t \, \forall p (p \notin ABOMB \lor (P(t))) \}\]
  – 3) there is no bad shipment that he missed
  \[ \{ t \, \neg \exists p (p \in ABOMB \land \neg (P(t))) \} \]
General pattern

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

\[ \forall x (P(x)) \equiv \neg \exists x (\neg P(x)) \]

\[ a \implies b \text{ is the same as } \neg a \lor b \]

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

- 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 \( \implies \) so does ‘s’
More on division

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

\[ \{ o | \forall t ((t \in \text{TAKEs} \land t.ssn = 123) \Rightarrow \exists t \in \text{TAKEs}\}
\]

\[ t.l.c - id = t.c - id \land t.l.ssn = o.ssn \}
\]

Safety of expressions

• FORBIDDEN:

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

It has infinite output!!

• Instead, always use

\[ \{ t | t \in \text{SOME-TABLE} \} \]

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> ∈ STUDENT ∧ s = 123 \}
Details

• Like R.T.C - symbols allowed:
  ∧, ∨, ¬, ⇒
  ≥, ≤, ≠, ∈
  (, ), ∈

• quantifiers ∀, ∃
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

(‘Proof’ of RDC = RA)

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

• (selection) find student record with ssn=123

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

or

\[ \{ s, n, a \mid s, n, a \in \text{STUDENT} \land s = 123 \} \]

Examples

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

\[ \{ n \mid <123, n, a > \in \text{STUDENT} \} \]
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 s \in \text{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 \text{FT\_STUDENT} \lor t \in \text{PT\_STUDENT} \}

Examples cont’d

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

\{ \langle s, n, a \rangle | \langle s, n, a \rangle \in \text{FT\_STUDENT} \lor \langle s, n, a \rangle \in \text{PT\_STUDENT} \}
Examples

- difference: find students that are not staff

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

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

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

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

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

Cartesian product

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

\[
\{ < m, f > \mid < m > \in MALE \land \\
\quad < f > \in FEMALE \}
\]
‘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>c-id</th>
<th>c-name</th>
<th>units</th>
</tr>
</thead>
<tbody>
<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>
</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>
<tr>
<td>grade</td>
<td>A</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}\langle s, \text{ssn} = e, \text{ssn} \land \\
\quad t.name = s.name \land \\
\quad e.c - c.id = 15 - 415\}}

More examples

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

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

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

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>TAKES</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>Name</td>
</tr>
<tr>
<td>x</td>
<td>P.</td>
</tr>
</tbody>
</table>

- 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:

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

join

projection

selection
Reminder: our Mini-U db

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>sан</td>
<td>c-id</td>
</tr>
<tr>
<td>smith</td>
<td>15-413</td>
</tr>
<tr>
<td>jones</td>
<td>15-413</td>
</tr>
</tbody>
</table>

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

More examples

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

\{< n >\}…………

\{< s, n, a >∈ STUDENT ∧
\{< s, c, g >∈ TAKES ∧
\{< c, cn, 2 >∈ CLASS\}

More examples

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

\{< n >} 3 s, a, c, g, cn(

\{< s, n, a >∈ STUDENT ∧
\{< s, c, g >∈ TAKES ∧
\{< c, cn, 2 >∈ CLASS

\})
Even more examples:

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

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

```
\{ 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") \}
```

Even more examples:

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

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

Even more examples:

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

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

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

\[
\{ \langle g \rangle \exists p(\langle g, p \rangle \in PC \land \langle p, "Tom" \rangle \in PC) \}
\]

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></td>
</tr>
<tr>
<td>s2</td>
<td>p1</td>
<td></td>
</tr>
<tr>
<td>s3</td>
<td>p1</td>
<td></td>
</tr>
<tr>
<td>s5</td>
<td>p3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>p2</td>
<td>s1</td>
</tr>
</tbody>
</table>
```

Hard examples: DIVISION

- find suppliers that shipped all the ABOMB parts

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

• find suppliers that shipped all the ABOMB parts

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

More on division

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

\[
\{ o \} \forall t(t \in TAKES \land t.ssn = 123) \Rightarrow \exists t \in TAKES(
\begin{align*}
& tl.c - id = t.c - id \land \\
& tl.ssn = o.ssn
\end{align*}
)
\]

More on division

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

\[
\{ < s > \} \forall c(\exists g(< 123, c, g > \in TAKES) \Rightarrow \exists g'( < s, c, g' > ) \in TAKES))
\]
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

\[ \{ M \mid M \in \text{Movie} \land M.\text{studioName} = \text{'Paramount'} \} \]

Answers …

• Movies starring Kevin Bacon

\[ \{ M \mid M \in \text{Movie} \land \exists A \in \text{ActsIn}(A.\text{movieTitle} = M.\text{title} \land A.\text{starName} = \text{'Bacon'}) \} \]
Answers …

• Stars who have been in a film with Kevin Bacon

\[
\{ S | S \in \text{Star} \land \exists A \in \text{ActsIn}(A.\text{starName} = S.\text{name} \land \\
\exists A_2 \in \text{ActsIn}(A_2.\text{movieTitle} = A.\text{movieTitle} \land \\
A_2.\text{starName} = 'Bacon')) \}
\]

Answers …

• Stars within six degrees of Kevin Bacon

\[
\{ S | S \in \text{Star} \land \exists A \in \text{ActsIn}(A.\text{starName} = S.\text{name} \land \\
\exists A_2 \in \text{ActsIn}(A_2.\text{movieTitle} = A.\text{movieTitle} \land \\
\exists A_3 \in \text{ActsIn}(A_3.\text{starName} = A_2.\text{starName} \land \\
\exists A_4 \in \text{ActsIn}(A_4.\text{movieTitle} = A_3.\text{movieTitle} \land \\
A_4.\text{starName} = 'Bacon')) \}
\]

Two degrees:

\[
\{ S | S \in \text{Star} \land \exists A \in \text{ActsIn}(A.\text{starName} = S.\text{name} \land \\
\exists A_3 \in \text{ActsIn}(A_3.\text{movieTitle} = A.\text{movieTitle} \land \\
\exists A_4 \in \text{ActsIn}(A_4.\text{movieTitle} = A_3.\text{movieTitle} \land \\
A_4.\text{starName} = 'Bacon')) \}
\]
Two degrees:

S: movie star
A: movie star
A2: movie star
A3: movie star
A4: movie star

‘Bacon’

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 should choose the most efficient version.
• Algebra and safe calculus have same expressive power — leads to the notion of relational completeness.