

# Database Applications (15-415)

## Relational Calculus

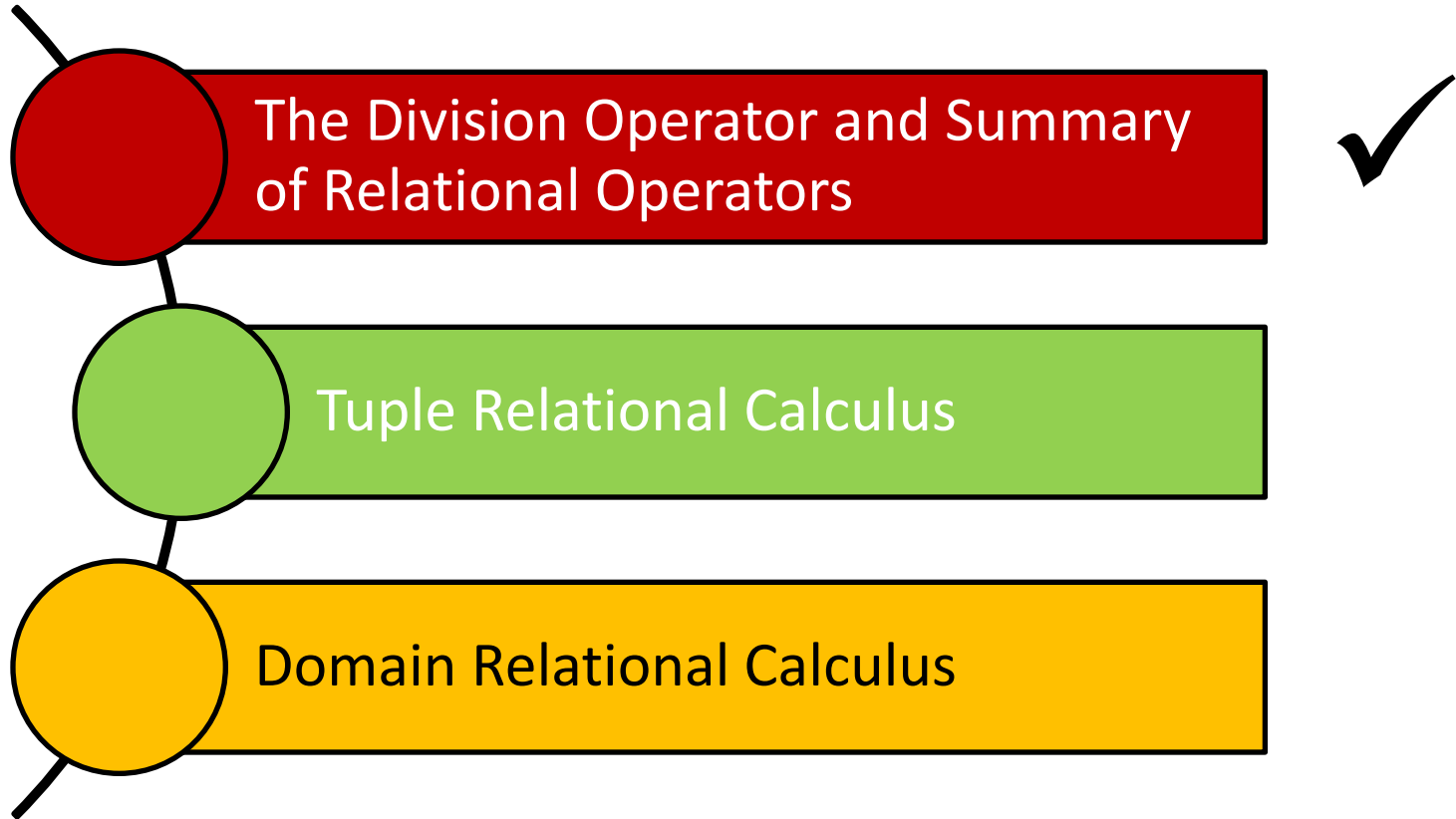
Lecture 5, January 27, 2014

Mohammad Hammoud

# Today...

- Last Session:
  - Relational Algebra
- Today's Session:
  - Relational algebra
    - The division operator and summary
  - Relational calculus
    - Tuple relational calculus
    - Domain relational calculus
- Announcement:
  - PS2 will be posted by tonight. It is due on Feb 06, 2014 by midnight

# Outline



# The Division Operation

- Division:  $R \div S$ 
  - Not supported as a primitive operator, but useful for expressing queries like:

*Find sailors who have reserved all boats*
  - Let  $A$  have 2 fields,  $x$  and  $y$ ;  $B$  has only field  $y$ :
    - $A/B$  contains all  $x$  tuples (sailors) such that for every  $y$  tuple (boat) in  $B$ , there is an  $xy$  tuple in  $A$
    - *Or*: If the set of  $y$  values (boats) associated with an  $x$  value (sailor) in  $A$  contains all  $y$  values in  $B$ , then  $x$  value is in  $A/B$
    - Formally:  $A/B = \{ \langle x \rangle \mid \exists \langle x, y \rangle \in A \quad \forall \langle y \rangle \in B \}$
- In general,  $x$  and  $y$  can be any lists of fields;  $y$  is the list of fields in  $B$ , and  $x$   $y$  is the list of fields in  $A$

# Examples of Divisions

| sno | pno |
|-----|-----|
| s1  | p1  |
| s1  | p2  |
| s1  | p3  |
| s1  | p4  |
| s2  | p1  |
| s2  | p2  |
| s3  | p2  |
| s4  | p2  |
| s4  | p4  |

*A*

| pno |
|-----|
| p2  |

*B1*

| pno |
|-----|
| p2  |
| p4  |

*B2*

| pno |
|-----|
| p1  |
| p2  |
| p4  |

*B3*

| sno |
|-----|
| s1  |
| s2  |
| s3  |
| s4  |

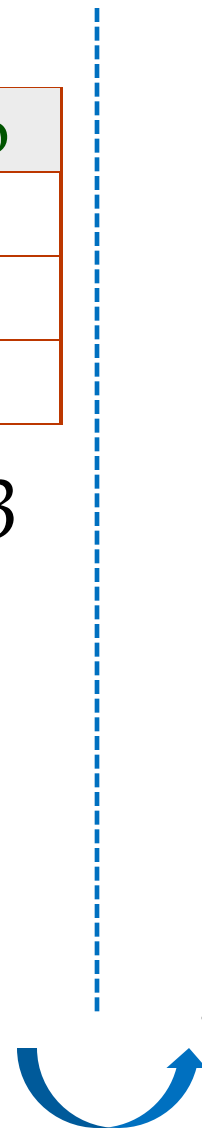
*A/B1*

| sno |
|-----|
| s1  |
| s4  |

*A/B2*

| sno |
|-----|
| s1  |

*A/B3*



# Expressing A/B Using Basic Operators

- Division can be derived from the fundamental operators
- **Idea:** For A/B, compute all x values that are not 'disqualified' by some y value in B
  - x value is disqualified if by attaching y value from B, we obtain an xy tuple that is "not" in A

Disqualified x values:  $\pi_x ((\pi_x(A) \times B) - A)$

$A/B$ :  $\pi_x(A) -$  all disqualified tuples

# A Query Example

- Find the names of sailors who've reserved all boats

$$\rho (Temp\ sids, (\pi_{sid, bid} Reserves) / (\pi_{bid} Boats))$$
$$\pi_{sname} (Temp\ sids \bowtie Sailors)$$

How can we find sailors who've reserved all 'Interlake' boats?

# Relational Algebra: Summary

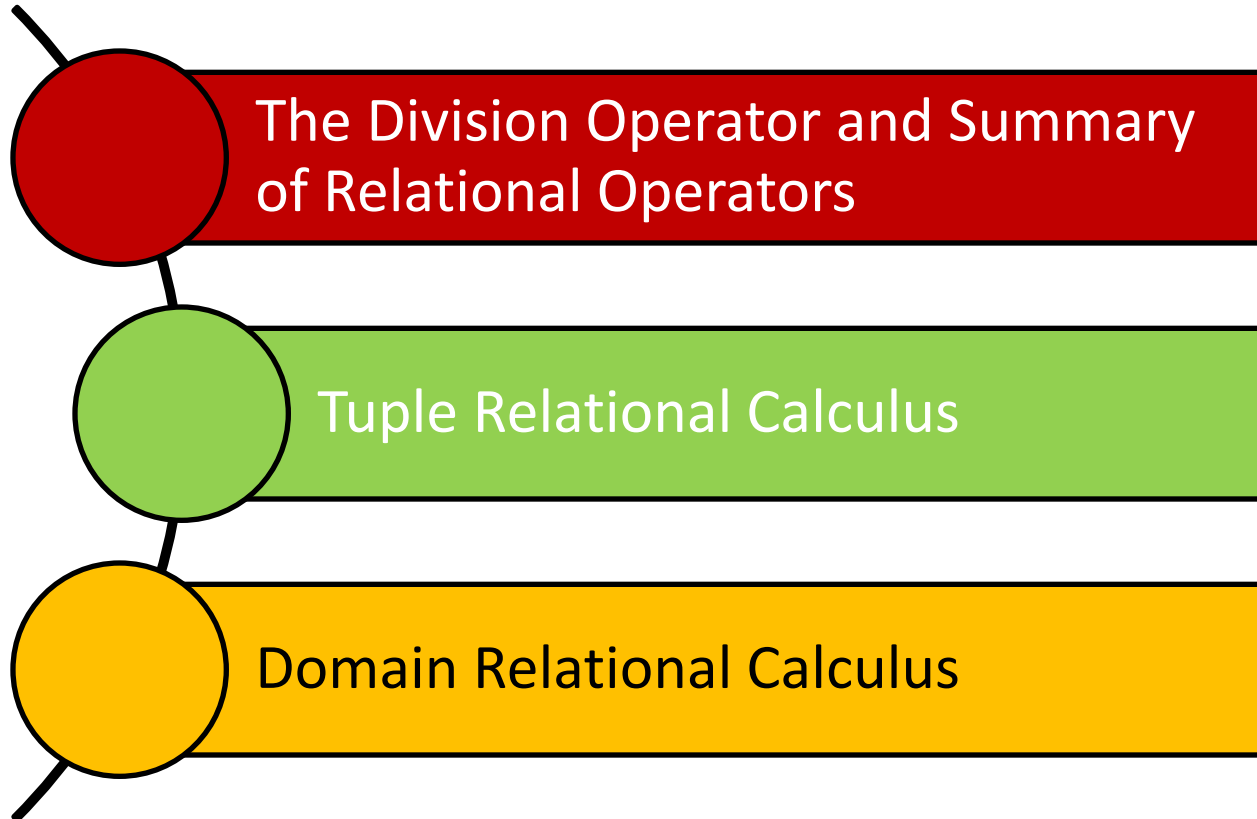
- Operators (with notations):
  1. **Selection** ( $\sigma$ ): selects a subset of rows from a relation
  2. **Projection** ( $\pi$ ): deletes unwanted columns from a relation
  3. **Cross-product** ( $\times$ ): allows combining two relations
  4. **Set-difference** ( $-$ ): retains tuples which are in relation 1, “but not” in relation 2
  5. **Union** ( $\cup$ ): retains tuples which are in “either” relation 1 “or” relation 2, “or in both”



# Relational Algebra: Summary

- Operators (with notations):
  6. **Intersection** ( $\cap$ ): retains tuples which are in relation 1 “and” in relation 2
  7. **Join** ( $\bowtie$ ): allows combining two relations according to a specific condition (e.g., *theta*, *equi* and *natural* joins)
  8. **Division** ( $\div$ ): generates the largest instance Q such that  $Q \times B \subseteq A$  when computing  $A/B$
  9. **Renaming** ( $\rho$ ): returns an instance of a new relation with some fields being potentially “renamed”

# Outline



# Overview - Detailed

- Tuple Relational Calculus (TRC)
  - Why?
  - Details
  - Examples
  - Equivalence with relational algebra
  - ‘Safety’ of expressions

# Motivation

- **Question:** What is the weakness of relational algebra?
- **Answer:** Procedural
  - It describes the steps for computing the desired answer (i.e., 'how')
  - Still useful, especially for query optimization

# Relational Calculus (in General)

- It describes 'what' we want (*not how*)
- It has two equivalent flavors, 'tuple' and 'domain' calculus
- It is the basis for SQL and Query By Example (QBE)
- It is useful for proofs (see query optimization, later)

# Tuple Relational Calculus (TRC)

- RTC is a subset of ‘first order logic’:

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

A “formula” that describes  $t$

Give me tuples ‘t’, satisfying predicate ‘P’

- **Examples:**

- Find all students:  $\{t \mid t \in STUDENT\}$
- Find all sailors with a rating above 7:

$$\{t \mid t \in Sailors \wedge t.rating > 7\}$$

# Syntax of TRC Queries

- The allowed symbols:

$\wedge, \vee, \neg, \Rightarrow$

$>, <, =, \neq, \leq, \geq,$

$(, ), \in$

- Quantifiers:

$\forall, \exists$

# Syntax of TRC Queries

- 'Atomic formulas':

$t \in TABLE$

$t.attr \text{ op } const$

$t.attr \text{ op } s.attr$

Where **op** is an operator in the set  $\{<, >, =, \leq, \geq, \neq\}$



# Syntax of TRC Queries

- A 'formula' is:

- Any atomic formula

- If  $P1$  and  $P2$  are formulas, so are

$$\neg P1; \neg P2; P1 \wedge P2; P1 \vee P2; P1 \Rightarrow P2$$

- If  $P(s)$  is a formula, so are

$$\exists s(P(s))$$

$$\forall s(P(s))$$

# Basic Rules

- Reminders:

- De Morgan:  $P1 \wedge P2 \equiv \neg(\neg P1 \vee \neg P2)$

- Implication:  $P1 \Rightarrow P2 \equiv \neg P1 \vee P2$

- Double Negation:

$$\forall s \in TABLE (P(s)) \equiv \neg \exists s \in TABLE (\neg P(s))$$

**‘every human is mortal : no human is immortal’**

# A Mini University Database

| STUDENT    |       |          |
|------------|-------|----------|
| <u>Ssn</u> | Name  | Address  |
| 123        | smith | main str |
| 234        | jones | QF ave   |

| CLASS       |        |       |
|-------------|--------|-------|
| <u>c-id</u> | c-name | units |
| 15-413      | s.e.   | 2     |
| 15-412      | o.s.   | 2     |

| TAKES      |             |       |
|------------|-------------|-------|
| <u>SSN</u> | <u>c-id</u> | grade |
| 123        | 15-413      | A     |
| 234        | 15-413      | B     |

# Examples

- Find all student records

$$\{t \mid t \in \textit{STUDENT}\}$$


**output  
tuple**

**of type 'STUDENT'**

# Examples

- Find the student record with `ssn=123`

# Examples

- Find the student record with  $ssn=123$

$$\{t \mid t \in STUDENT \wedge t.ssn = 123\}$$

This is equivalent to the 'Selection' operator in Relational Algebra!

# Examples

- Find the **name** of the student with  $ssn=123$

~~$\{t \mid t \in STUDENT \wedge t.ssn = 123\}$~~

Will this work?

# Examples

- Find the **name** of the student with  $ssn=123$

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



**'t' has only one column**

This is equivalent to the 'Projection' operator in Relational Algebra!



# Examples

- Get records of part time or full time students\*

$$\{t \mid t \in FT\_STUDENT \vee t \in PT\_STUDENT\}$$

This is equivalent to the 'Union' operator in Relational Algebra!

\* Assume we maintain tables for *PT\_STUDENT* and *FT\_STUDENT* in our Mini University DB

# Examples

- Find students that are not staff\*

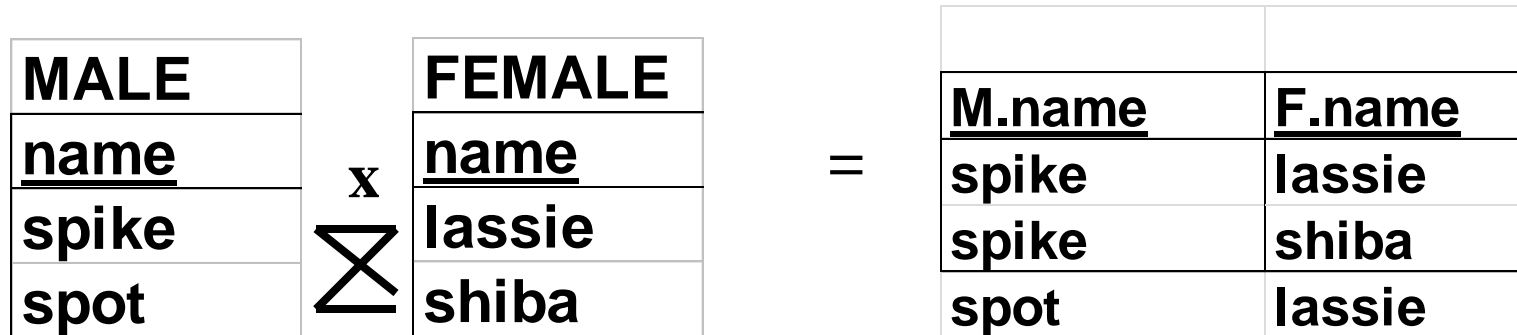
$$\{t \mid t \in \mathit{STUDENT} \wedge t \notin \mathit{STAFF}\}$$

This is equivalent to the 'Difference' operator in Relational Algebra!

- \* Assume we maintain a table for STAFF in our Mini University DB and that STUDENT and STAFF are union-compatible

# Cartesian Product: A Reminder

- Assume MALE and FEMALE dog tables as follows:



This gives *all* possible couples!

# Examples (Cont'd)

- Find all the pairs of (male, female) dogs

$$\{t \mid \exists m \in \text{MALE} \wedge \\ \exists f \in \text{FEMALE} \\ (t.m - \text{name} = m.\text{name} \wedge \\ t.f - \text{name} = f.\text{name})\}$$

This is equivalent to the 'Cartesian Product' operator in Relational Algebra!

# More Examples

- Find the names of students taking 15-415

| STUDENT    |       |          |
|------------|-------|----------|
| <u>Ssn</u> | Name  | Address  |
| 123        | smith | main str |
| 234        | jones | QF ave   |

| CLASS       |        |       |
|-------------|--------|-------|
| <u>c-id</u> | c-name | units |
| 15-413      | s.e.   | 2     |
| 15-412      | o.s.   | 2     |

| TAKES      |             |       |
|------------|-------------|-------|
| <u>SSN</u> | <u>c-id</u> | grade |
| 123        | 15-413      | A     |
| 234        | 15-413      | B     |

2-way Join!

# More Examples

- Find the names of students taking 15-415

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

# More Examples

- Find the names of students taking 15-415

$\{t \mid \exists s \in STUDENT$

$\wedge \exists e \in TAKES ( s.ssn = e.ssn \wedge$

**join**

$t.name = s.name \wedge$

**projection**

$e.c - id = 15 - 415) \}$

**selection**

# More Examples

- Find the names of students taking a 2-unit course

| STUDENT    |       |          |
|------------|-------|----------|
| <u>Ssn</u> | Name  | Address  |
| 123        | smith | main str |
| 234        | jones | QF ave   |

| CLASS       |        |       |
|-------------|--------|-------|
| <u>c-id</u> | c-name | units |
| 15-413      | s.e.   | 2     |
| 15-412      | o.s.   | 2     |

| TAKES      |             |       |
|------------|-------------|-------|
| <u>SSN</u> | <u>c-id</u> | grade |
| 123        | 15-413      | A     |
| 234        | 15-413      | B     |

3-way Join!



# More Examples

- Find the names of students taking a 2-unit course

$$\{t \mid \exists s \in STUDENT \wedge \exists e \in TAKES$$
$$\exists c \in CLASS( s.ssn = e.ssn \wedge$$
$$e.c - id = c.c - id \wedge$$
$$t.name = s.name \wedge$$
$$c.units = 2)\}$$

join

projection


selection

What is the equivalence of this in Relational Algebra?

# More on Joins

- Assume a Parent-Children (PC) table instance as follows:

| PC          |      |
|-------------|------|
| <u>p-id</u> | c-id |
| Mary        | Tom  |
| Peter       | Mary |
| John        | Tom  |



| PC          |      |
|-------------|------|
| <u>p-id</u> | c-id |
| Mary        | Tom  |
| Peter       | Mary |
| John        | Tom  |

- Who are Tom's grandparent(s)? (*this is a self-join*)

# More Join Examples

- Find Tom's grandparent(s)

$$\{t \mid \exists p \in PC \wedge \exists q \in PC \\ ( p.c - id = q.p - id \wedge \\ p.p - id = t.p - id \wedge \\ q.c - id = "Tom" ) \}$$

What is the equivalence of this in Relational Algebra?

# Harder Examples: DIVISION

- Find suppliers that shipped all the bad parts

| SHIPMENT  |           |
|-----------|-----------|
| <u>s#</u> | <u>p#</u> |
| s1        | p1        |
| s2        | p1        |
| s1        | p2        |
| s3        | p1        |
| s5        | p3        |

÷

| BAD_P     |
|-----------|
| <u>p#</u> |
| p1        |
| p2        |
|           |

=

| BAD_S     |
|-----------|
| <u>s#</u> |
| s1        |

# Harder Examples: DIVISION

- Find suppliers that shipped all the bad parts

$$\{t \mid \forall p(p \in \text{BAD\_P} \Rightarrow (\exists s \in \text{SHIPMENT}(t.s\# = s.s\# \wedge s.p\# = p.p\#))))\}$$

What is the equivalence of this in Relational Algebra?

# General Patterns

- There are three equivalent versions:

1) If it is bad, he shipped it

$$\{t \mid \forall p(p \in \text{BAD} \_ P \Rightarrow (P(t)))\}$$

2) Either it was good, or he shipped it

$$\{t \mid \forall p(p \notin \text{BAD} \_ P \vee (P(t)))\}$$

3) There is no bad shipment that he missed

$$\{t \mid \neg \exists p(p \in \text{BAD} \_ P \wedge (\neg P(t)))\}$$

# More on Division

- Find (SSNs of) students that take all the courses that  $ssn=123$  does (and maybe even more)

One way to think about this:  
Find students 's' so that if 123 takes a course => so does 's'

# More on Division

- Find (SSNs of) students that take all the courses that  $ssn=123$  does (and maybe even more)

$$\{o \mid \forall t((t \in TAKES \wedge t.ssn = 123) \Rightarrow \\ \exists t1 \in TAKES ( \\ t1.c - id = t.c - id \wedge \\ t1.ssn = o.ssn) \\ )\}$$



# 'Proof' of Equivalence

- Relational Algebra  $\leftrightarrow$  TRC

But...

# Safety of Expressions

- What about?

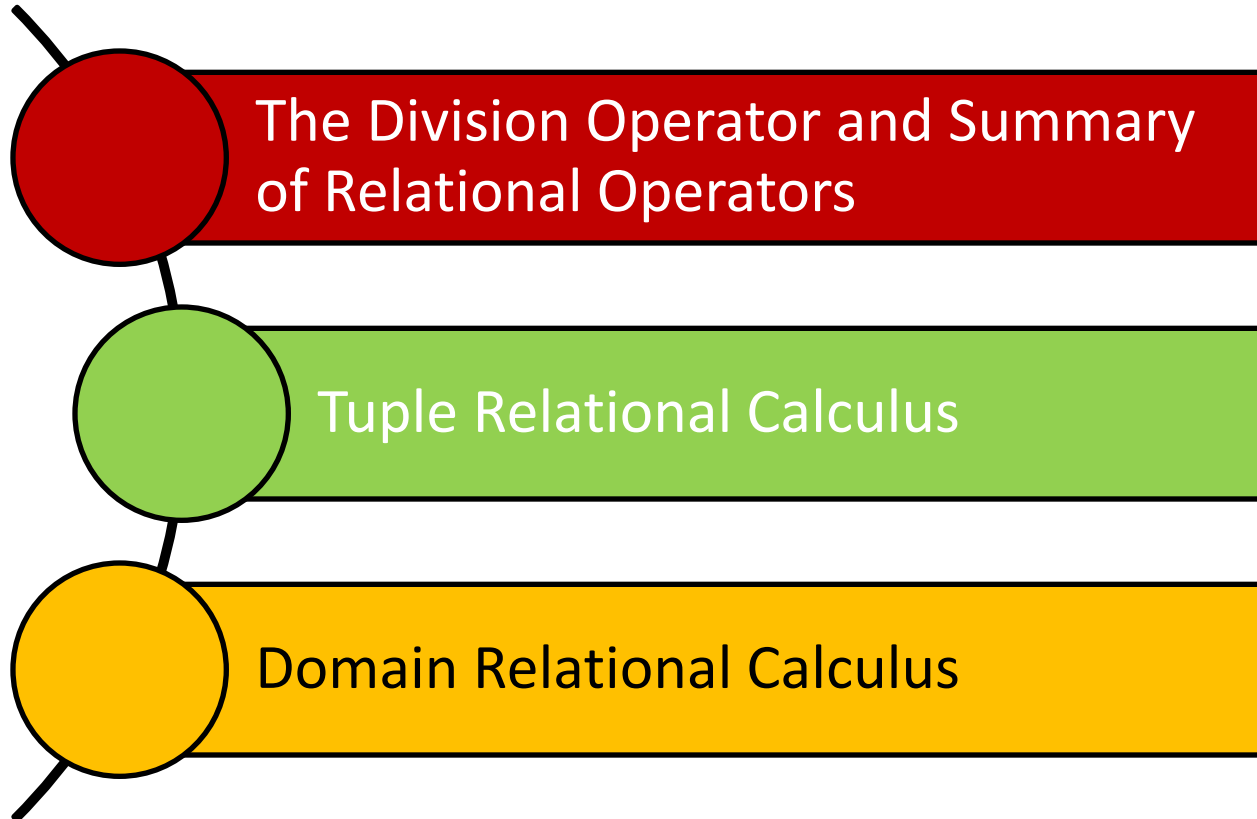
~~$\{t \mid t \notin \text{STUDENT}\}$~~

It has infinite output!!

- Instead, always use:

$\{t \mid \dots t \in \text{SOME} - \text{TABLE}\}$

# Outline



# Overview - Detailed

- Domain Relational Calculus (DRC)
  - Why?
  - Details
  - Examples
  - Equivalence with TRC and relational algebra
  - 'Safety' of expressions

# Domain Relational Calculus (DRC)

- **Question:** why?
- **Answer:** slightly easier than TRC, although equivalent- basis for QBE
- **Idea:** “domain” variables instead of “tuple” variables
- **Example:** ‘find STUDENT record with ssn=123’

$$\{ \langle s, n, a \rangle \mid \langle s, n, a \rangle \in STUDENT \wedge s = 123 \}$$

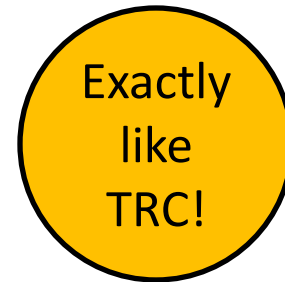
# Syntax of DRC Queries

- The allowed symbols are:

$\wedge, \vee, \neg, \Rightarrow$

$>, <, =, \neq, \leq, \geq,$

$(, ), \in$

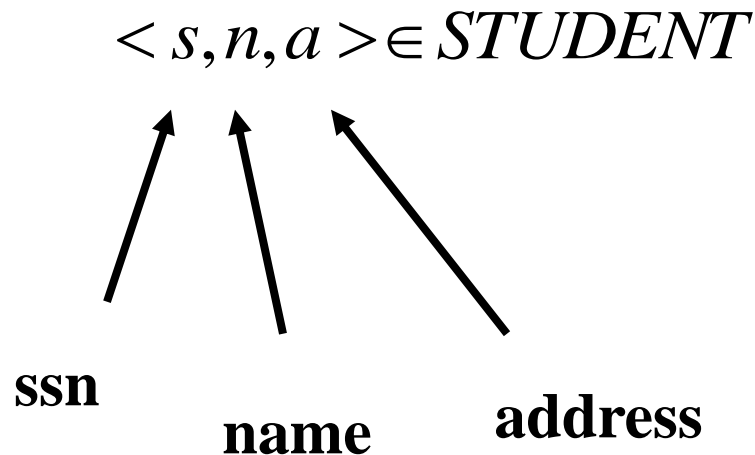


- Quantifiers:

$\forall, \exists$

# Syntax of DRC Queries

- But: domain (= column) variables, as opposed to tuple variables:



# Reminder: Our Mini University DB

| STUDENT    |       |          |
|------------|-------|----------|
| <u>Ssn</u> | Name  | Address  |
| 123        | smith | main str |
| 234        | jones | QF ave   |

| CLASS       |        |       |
|-------------|--------|-------|
| <u>c-id</u> | c-name | units |
| 15-413      | s.e.   | 2     |
| 15-412      | o.s.   | 2     |

| TAKES      |             |       |
|------------|-------------|-------|
| <u>SSN</u> | <u>c-id</u> | grade |
| 123        | 15-413      | A     |
| 234        | 15-413      | B     |



# Examples

- Find all student records

$$\{ \langle s, n, a \rangle \mid \langle s, n, a \rangle \in STUDENT \}$$

- What is the equivalence of this in TRC?

$$\{ t \mid t \in STUDENT \}$$

# Examples

- Find the student record with  $ssn=123$

$$\{ \langle s, n, a \rangle \mid \langle s, n, a \rangle \in STUDENT \wedge s = 123 \}$$

**OR:**

$$\{ \langle 123, n, a \rangle \mid \langle 123, n, a \rangle \in STUDENT \}$$

In TRC:  $\{ t \mid t \in STUDENT \wedge t.ssn = 123 \}$

This is equivalent to the 'Selection' operator in Relational Algebra!

# Examples

- Find the name of student with  $ssn=123$

$$\{ \langle n \rangle \mid \langle 123, n, a \rangle \in STUDENT \}$$

In TRC:  $\{ t \mid \exists s \in STUDENT (s.ssn = 123 \wedge t.name = s.name) \}$

# Examples

- Find the name of student with ssn=123

$$\{ \langle n \rangle \mid \exists a (\langle 123, n, a \rangle \in STUDENT) \}$$

↑ need to 'bind' "a"

In TRC:  $\{ t \mid \exists s \in STUDENT (s.ssn = 123 \wedge t.name = s.name) \}$

This is equivalent to the 'Projection' operator in Relational Algebra!

# Examples

- Get records of both PT and FT students

$$\{ \langle s, n, a \rangle \mid \langle s, n, a \rangle \in FT\_STUDENT \vee \langle s, n, a \rangle \in PT\_STUDENT \}$$

In TRC:  $\{ t \mid t \in FT\_STUDENT \vee t \in PT\_STUDENT \}$

This is equivalent to the 'Union' operator in Relational Algebra!

# Examples

- Find the students that are not staff

$$\{ \langle s, n, a \rangle \mid \langle s, n, a \rangle \in STUDENT \wedge \langle s, n, a \rangle \notin STAFF \}$$

In TRC:  $\{ t \mid t \in STUDENT \wedge t \notin STAFF \}$

This is equivalent to the 'Difference' operator in Relational Algebra!

# Examples

- Find all the pairs of (male, female)

$$\{ \langle m, f \rangle \mid \langle m \rangle \in \text{MALE} \wedge \langle f \rangle \in \text{FEMALE} \}$$

**In TRC:**  $\{ t \mid \exists m \in \text{MALE} \wedge \exists f \in \text{FEMALE} (t.m - name = m.name \wedge t.f - name = f.name) \}$

This is equivalent to the 'Cartesian Product' operator in Relational Algebra!

# Examples

- Find the names of students taking 15-415

| STUDENT    |       |          |
|------------|-------|----------|
| <u>Ssn</u> | Name  | Address  |
| 123        | smith | main str |
| 234        | jones | QF ave   |

| CLASS       |        |       |
|-------------|--------|-------|
| <u>c-id</u> | c-name | units |
| 15-413      | s.e.   | 2     |
| 15-412      | o.s.   | 2     |

| TAKES      |             |       |
|------------|-------------|-------|
| <u>SSN</u> | <u>c-id</u> | grade |
| 123        | 15-413      | A     |
| 234        | 15-413      | B     |

2-way Join!



# Examples

- Find the names of students taking 15-415

$$\{ \langle n \rangle \mid \exists s \exists a \exists g (\langle s, n, a \rangle \in \text{STUDENT} \\ \wedge \langle s, 15-415, g \rangle \in \text{TAKES}) \}$$

**In TRC:**  $\{ t \mid \exists s \in \text{STUDENT} \\ \wedge \exists e \in \text{TAKES} ( s.ssn = e.ssn \wedge \\ t.name = s.name \wedge \\ e.c - id = 15-415) \}$

This is equivalent to the 'Join' operator in Relational Algebra!

# A Sneak Preview of QBE

- Very user friendly
- Heavily based on RDC
- Very similar to MS Access interface

$$\{ \langle n \rangle \mid \exists s \exists a \exists g (\langle s, n, a \rangle \in STUDENT \wedge \langle s, 15-415, g \rangle \in TAKES) \}$$

| STUDENT    |      |         |
|------------|------|---------|
| <u>Ssn</u> | Name | Address |
| _x         | P.   |         |
|            |      |         |

| TAKES      |             |       |
|------------|-------------|-------|
| <u>SSN</u> | <u>c-id</u> | grade |
| _x         | 15-415      |       |

# More Examples

- Find the names of students taking a 2-unit course

| STUDENT    |       |          |
|------------|-------|----------|
| <u>Ssn</u> | Name  | Address  |
| 123        | smith | main str |
| 234        | jones | QF ave   |

| CLASS       |        |       |
|-------------|--------|-------|
| <u>c-id</u> | c-name | units |
| 15-413      | s.e.   | 2     |
| 15-412      | o.s.   | 2     |

| TAKES      |             |       |
|------------|-------------|-------|
| <u>SSN</u> | <u>c-id</u> | grade |
| 123        | 15-413      | A     |
| 234        | 15-413      | B     |

3-way Join!

# More Examples

- Find the names of students taking a 2-unit course

In TRC:

$$\{t \mid \exists s \in STUDENT \wedge \exists e \in TAKES$$
$$\exists c \in CLASS( s.ssn = e.ssn \wedge$$
$$e.c - id = c.c - id \wedge$$
$$t.name = s.name \wedge$$
$$c.units = 2)\}$$

join

projection

selection

# More Examples

- Find the names of students taking a 2-unit course

In DRC:

$\{ \langle n \rangle \mid \dots \dots \dots \}$

$\langle s, n, a \rangle \in STUDENT \wedge$

$\langle s, c, g \rangle \in TAKES \wedge$

$\langle c, cn, 2 \rangle \in CLASS \}$

# More Examples

- Find the names of students taking a 2-unit course

In DRC:

$$\{ \langle n \rangle \mid \exists s, a, c, g, cn ($$
$$\langle s, n, a \rangle \in STUDENT \wedge$$
$$\langle s, c, g \rangle \in TAKES \wedge$$
$$\langle c, cn, 2 \rangle \in CLASS$$
$$) \}$$


Easier  
than  
TRC!

# Even More Examples

- Find Tom's grandparent(s)

| PC          |      |
|-------------|------|
| <u>p-id</u> | c-id |
| Mary        | Tom  |
| Peter       | Mary |
| John        | Tom  |

| PC          |      |
|-------------|------|
| <u>p-id</u> | c-id |
| Mary        | Tom  |
| Peter       | Mary |
| John        | Tom  |

In TRC:

$$\{t \mid \exists p \in PC \wedge \exists q \in PC$$

$$(p.c-id = q.p-id \wedge$$

$$p.p-id = t.p-id \wedge$$

$$q.c-id = "Tom")\}$$

In DRC:

$$\{ \langle g \rangle \mid \exists p (\langle g, p \rangle \in PC \wedge$$

$$\langle p, "Tom" \rangle \in PC) \}$$

# Harder Examples: DIVISION

- Find suppliers that shipped all the bad parts

| SHIPMENT  |           |
|-----------|-----------|
| <u>s#</u> | <u>p#</u> |
| s1        | p1        |
| s2        | p1        |
| s1        | p2        |
| s3        | p1        |
| s5        | p3        |

÷

| BAD_P     |
|-----------|
| <u>p#</u> |
| p1        |
| p2        |
|           |

=

| BAD_S     |
|-----------|
| <u>s#</u> |
| s1        |



# Harder Examples: DIVISION

- Find suppliers that shipped all the bad parts

In TRC:

$$\{t \mid \forall p(p \in BAD\_P \Rightarrow (\exists s \in SHIPMENT($$
$$t.s\# = s.s\# \wedge$$
$$s.p\# = p.p\#))))\}$$

In DRC:

$$\{ \langle s \rangle \mid \forall p(\langle p \rangle \in BAD\_P \Rightarrow$$
$$\langle s, p \rangle \in SHIPMENT) \}$$


# More on Division

- Find (SSNs of) students that take all the courses that  $ssn=123$  does (and maybe even more)

In TRC:

$$\{o \mid \forall t((t \in TAKES \wedge t.ssn = 123) \Rightarrow \\ \exists t1 \in TAKES ( \\ t1.c - id = t.c - id \wedge \\ t1.ssn = o.ssn) \\ )\}$$

# More on Division

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

In DRC:

$$\{ \langle s \rangle \mid \forall c (\exists g (\langle 123, c, g \rangle \in TAKES) \Rightarrow \exists g' (\langle s, c, g' \rangle \in TAKES)) \}$$

# 'Proof' of Equivalence

- Relational Algebra  $\leftrightarrow$  Domain Relational Calculus  $\leftrightarrow$  Tuple Relational Calculus

But...

# Safety of Expressions

- Similar to TRC
- FORBIDDEN:

$$\{ \langle s, n, a \rangle \mid \langle s, n, a \rangle \notin STUDENT \}$$

# Summary

- The relational model has rigorously defined query languages — simple and powerful
- Relational algebra is more operational/procedural
  - Useful for internal representation of query evaluation plans
- Relational calculus is **declarative**
  - Users define queries in terms of what they want, not in terms of how to compute them

# Summary

- 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*

# Next Class

## SQL- Part I