Relational Model 4: Relational Calculus

Gary KL Tam

Department of Computer Science
Swansea University

Q & A

- Is Relational Calculus mathematics?

Q & A

- Is Relational Calculus mathematics?
- Yes, based on a branch of Predicate Calculus (Logic).

Q & A

- Is Relational Calculus mathematics?
- Yes, based on a branch of Predicate Calculus (Logic).
- Why should we learn it?

Q & A

- Is Relational Calculus mathematics?
- Yes, based on a branch of Predicate Calculus (Logic).
- Why should we learn it?
- Because university teaches it !???

Q & A

- Is Relational Calculus mathematics?
- Yes, based on a branch of Predicate Calculus (Logic).
- Why should we learn it?
- Because university teaches it !???

Example Use

Microsoft Access: Query by example!

Relational Algebra & Calculus

- Relational Algebra is procedural.
- Relational calculus is declarative

Relational Algebra & Calculus

- Relational Algebra is procedural.
- Relational calculus is declarative
- It has been proven that Relational Calculus is equivalent to Relational Algebra in term of expressive power.

Relational Algebra & Calculus

- Relational Algebra is procedural.
- Relational calculus is declarative
- It has been proven that Relational Calculus is equivalent to Relational Algebra in term of expressive power.
- The upshot of this is that no matter in what form a query (e.g. using SQL) is made, the DBMS should be able to ignore the steps implied by the formulation of the query and make decisions about how to fulfil that query in the most efficient manner.

Relational Algebra & Calculus

- Relational Algebra is procedural.
- Relational calculus is declarative
- It has been proven that Relational Calculus is equivalent to Relational Algebra in term of expressive power.
- The upshot of this is that no matter in what form a query (e.g. using SQL) is made, the DBMS should be able to ignore the steps implied by the formulation of the query and make decisions about how to fulfil that query in the most efficient manner.
- This is the process known as Query Optimisation.

Tuple Relational Calculus

- A non-procedural query language, where each query is of the form: $\{t|P(t)\}$
- It is the set of all tuples t such that predicate P is true for t
- t is a tuple variable, $t[A]$ denotes the value of tuple t on attribute A
- $t \in r$ denotes that tuple t is in relation r
- P is a formula

Predicate Calculus Formula

- Set of attributes and constants
- Set of comparison operators: (e.g., $<, \leq, =, \neq, \geq, >$)
- Set of connectives: and ($\wedge$), or ($\vee$), not ($\neg$)
- Implication ($\Rightarrow$): $x \Rightarrow y$, if x is true, then y is true
  $x \Rightarrow y \equiv \neg x \vee y$

Predicate Calculus Formula

- Set of attributes and constants
- Set of comparison operators: (e.g., $<, \leq, =, \neq, \geq, >$)
- Set of connectives: and ($\wedge$), or ($\vee$), not ($\neg$)
- Implication ($\Rightarrow$): $x \Rightarrow y$, if x is true, then y is true
  $x \Rightarrow y \equiv \neg x \vee y$
- Set of quantifiers:
  $\exists t \in r(Q(t))$: "there exists ($\exists$)" a tuple t in relation r such that predicate $Q(t)$ is true.
  $\forall t \in r(Q(t))$: Q is true "for all ($\forall$)" tuples t in relation r.

loan

loan-number	branch-name	amount
L110	swansea	1530
L223	cardiff	2140
L331	neath	1000

Query 1:

Find the loan-number, branch-name and amount for loans of over £1200

Example Query

loan

loan-number	branch-name	amount
L110	swansea	1530
L223	cardiff	2140
L331	neath	1000

Query 1:
Find the loan-number, branch-name and amount for loans of over £1200

Relational Algebra: $\sigma_{\text{amount}>1200}(\text{loan})$

Example Query

loan

loan-number	branch-name	amount	
L110	swansea	1530	✓
L223	cardiff	2140	✓
L331	neath	1000	

Query 1:
Find the loan-number, branch-name and amount for loans of over £1200

Relational Algebra: $\sigma_{\text{amount}>1200}(\text{loan})$

[Selection]
Step1: go to loan table
Step2: go through each tuple in loan
Step2-1: select the tuple with amount > 1200

Example Query

loan

loan-number	branch-name	amount	
L110	swansea	1530	✓
L223	cardiff	2140	✓
L331	neath	1000	

Query 1:
Find the loan-number, branch-name and amount for loans of over £1200

Relational Calculus: $\{t \mid t \in loan(t[amount] > 1200)\}$

loan	loan-number	branch-name	amount	
t:	L110	swansea	1530	✓
t:	L223	cardiff	2140	✓
	L331	neath	1000	

Query 1:

Find the loan-number, branch-name and amount for loans of over £1200

Relational Calculus:

$$\{t \mid t \in loan(t[amount] > 1200)\}$$

t is the result

what is *t*?

| means such that

t is a tuple in loan

t[amount] > 1200

Note: RC defines the data (not steps)!

loan

loan-number	branch-name	amount
L110	swansea	1530
L223	cardiff	2140
L331	neath	1000

Query 2:

Find the loan number for each loan of an amount > £1200

Example Query

loan

loan-number	branch-name	amount
L110	swansea	1530
L223	cardiff	2140
L331	neath	1000

Query 2:
Find the loan number for each loan of an amount > £1200

Relational Algebra: $\Pi_{loan\text{-}number}(\sigma_{amount>1200}(loan))$

Example Query

loan

loan-number	branch-name	amount
L110	swansea	1530
L223	cardiff	2140
L331	neath	1000

Query 2:
Find the loan number for each loan of an amount > £1200

Relational Algebra: $\boxed{\Pi_{\text{loan-number}} \left(\boxed{\sigma_{\text{amount}>1200} (\text{loan})} \right)}$

[Selection]
Step1: go to loan table
Step2: go through each tuple in loan
Step3: select the tuple with amount > 1200
[Projection]
Step1: go through each tuple in the above result
Step1-1: take loan-number

Example Query

loan

loan-number	branch-name	amount
L110	swansea	1530
L223	cardiff	2140
L331	neath	1000

Query 2:
Find the loan number for each loan of an amount > £1200

Relational Calculus:

$$\{t \mid \exists s \in loan(t[\text{loan-number}] = s[\text{loan-number}] \land s[\text{amount}] > 1200)\}$$

loan	loan-number	branch-name	amount
s:	L110	swansea	1530
s:	L223	cardiff	2140
	L331	neath	1000

t

Query 2:
Find the loan number for each loan of an amount > £1200

Relational Calculus:

t is the result

what is *t*?

there exists a tuple *s* in loan

is equal to, aka. == (in java)
i.e., not assignment!

$$\{t \mid \exists s \in loan(t[\text{loan-number}] = s[\text{loan-number}] \wedge s[\text{amount}] > 1200)\}$$

and

s[amount] > 1200

t is a tuple, but contains one attribute [loan-number]

borrower

loan-number	cust-name
L110	Gary
L223	Maple
L331	Syrup

depositor

cust-name	acct-num
Gary	123
April	345
Dave	567

Query 3:

Find the names of all customers having a loan, an account, or both at the bank

Example Query

borrower

loan-number	cust-name
L110	Gary
L223	Maple
L331	Syrup

depositor

cust-name	acct-num
Gary	123
April	345
Dave	567

Query 3:
Find the names of all customers having a loan, an account, or both at the bank

Relational Algebra:

$$\Pi_{\text{cust-name}}\ (\text{borrower})\ \cup\ \Pi_{\text{cust-name}}\ (\text{depositor})$$

Steps:
Step1: a projection on borrower
Step2: a projection on depositor
Step3: union of the above two results

cust-name
Gary
April
Dave
Maple
Syrup

borrower

loan-number	cust-name
L110	Gary
L223	Maple
L331	Syrup

depositor

cust-name	acct-num
Gary	123
April	345
Dave	567

Query 3:
Find the names of all customers having a loan, an account, or both at the bank

Relational Calculus:

$$\{t | \exists s \in borrower(t[\text{cust-name}] = s[\text{cust-name}]) \lor \exists u \in depositor(t[\text{cust-name}] = u[\text{cust-name}])\}$$

cust-name
Gary
April
Dave
Maple
Syrup

borrower	loan-number	cust-name
s:	L110	Gary
s:	L223	Maple
s:	L331	Syrup

t

depositor	cust-name	acct-num
u:	Gary	123
u:	April	345
u:	Dave	567

t

Query 3:
Find the names of all customers having a loan, an account, or both at the bank

Relational Calculus:

there exists a tuple *s* in borrower

$$\{t \mid \exists s \in borrower(t[\text{cust-name}] = s[\text{cust-name}]) \lor$$
$$\exists u \in depositor(t[\text{cust-name}] = u[\text{cust-name}])\}$$

there exists a tuple *u* in depositor

t is the cust-name of s or u

cust-name
Gary
April
Dave
Maple
Syrup

Example Query

borrower

loan-number	cust-name
L110	Gary
L223	Maple
L331	Syrup

loan

loan-number	branch-name	amount
L110	swansea	1530
L223	cardiff	2140
L331	neath	1000

Query 4:
Find the names of all customers having a loan at the Neath branch

Example Query

borrower

loan-number	cust-name
L110	Gary
L223	Maple
L331	Syrup

loan

loan-number	branch-name	amount
L110	swansea	1530
L223	cardiff	2140
L331	neath	1000

Query 4:
Find the names of all customers having a loan at the Neath branch

Relational Algebra:

$$\Pi_{\text{cust-name}} (\sigma_{\text{branch-name='neath'}} (\text{loan} \bowtie \text{borrower}))$$

Steps:
Step1: natural join loan and borrower on ALL common attributes
Step2: for all tuple from step1, select tuple with branch-name = ‘neath’
Step3: pick the cust-name column

Example Query

borrower

loan-number	cust-name
L110	Gary
L223	Maple
L331	Syrup

loan

loan-number	branch-name	amount
L110	swansea	1530
L223	cardiff	2140
L331	neath	1000

Query 4:
Find the names of all customers having a loan at the Neath branch

$\Pi_{\text{cust-name}} (\sigma_{\text{branch-name='neath'}} ($

loan-number	cust-name	branch-name	amount
L110	Gary	swansea	1530
L223	Maple	cardiff	2140
L331	Syrup	neath	1000

$))$

$\Pi_{\text{cust-name}} ($

loan-number	cust-name	branch-name	amount
L331	Syrup	neath	1000

$)$

cust-name
Syrup

Example Query

borrower

loan-number	cust-name
L110	Gary
L223	Maple
L331	Syrup

loan

loan-number	branch-name	amount
L110	swansea	1530
L223	cardiff	2140
L331	neath	1000

Query 4:
Find the names of all customers having a loan at the Neath branch

Relational Calculus:

$$\{t \mid \exists s \in borrower(t[\text{cust-name}] = s[\text{cust-name}]) \wedge$$
$$\exists u \in loan(u[\text{loan-number}] = s[\text{loan-number}] \wedge u[\text{branch-name}] = \text{`neath'})\}$$

Example Query

borrower

loan-number	cust-name
L110	Gary
L223	Maple
L331	Syrup

s: (row L331)

loan

loan-number	branch-name	amount
L110	swansea	1530
L223	cardiff	2140
L331	neath	1000

u: (row L331)

t

Query 4:
Find the names of all customers having a loan at the Neath branch

Relational Calculus:

$$\{t \mid \exists s \in borrower(t[\text{cust-name}] = s[\text{cust-name}]) \land$$
$$\exists u \in loan(u[\text{loan-number}] = s[\text{loan-number}] \land u[\text{branch-name}] = \text{`neath'})\}$$

Example Query

borrower

loan-number	cust-name
L110	Gary
L223	Maple
L331	Syrup

loan

loan-number	branch-name	amount
L110	swansea	1530
L223	neath	2140
L331	neath	1000

Query 5:
Find the names of all customers who have a loan at the 'neath' branch, but no account at any branch of the bank

depositor

cust-name	acct-num
Gary	123
Maple	345
Dave	567

borrower

loan-number	cust-name
L110	Gary
L223	Maple
L331	Syrup

loan

loan-number	branch-name	amount
L110	swansea	1530
L223	neath	2140
L331	neath	1000

depositor

cust-name	acct-num
Gary	123
Maple	345
Dave	567

Query 5:

Find the names of all customers who have a loan at the 'neath' branch, but no account at any branch of the bank

Relational Algebra:

$$\Pi_{\text{cust-name}} (\sigma_{\text{branch-name='neath'}} (\text{loan} \bowtie \text{borrower})) - \Pi_{\text{cust-name}} (\text{depositor})$$

(

cust-loan-in-neath
Maple
Syrup

) - (

cust-acct
Gary
Maple
Dave

)

Results:

Syrup

Example Query

borrower

loan-number	cust-name
L110	Gary
L223	Maple
L331	Syrup

loan

loan-number	branch-name	amount
L110	swansea	1530
L223	neath	2140
L331	neath	1000

Query 5:
Find the names of all customers who have a loan at the ‘neath’ branch, but no account at any branch of the bank

depositor

cust-name	acct-num
Gary	123
Maple	345
Dave	567

Relational Calculus:

$$\{t \mid \exists s \in borrower(t[\text{cust-name}] = s[\text{cust-name}]) \land$$
$$\exists u \in loan(u[\text{loan-number}] = s[\text{loan-number}] \land u[\text{branch-name}] = \text{‘neath’})$$
$$\land \neg \exists v \in depositor(t[\text{cust-name}] = v[\text{cust-name}])\}$$

borrower

	loan-number	cust-name
	L110	Gary
s:	L223	Maple
s:	L331	Syrup

t

loan

	loan-number	branch-name	amount
	L110	swansea	1530
u:	L223	neath	2140
u:	L331	neath	1000

depositor

	cust-name	acct-num
	Gary	123
v:	Maple	345
	Dave	567

Query 5:

Find the names of all customers who have a loan at the 'neath' branch, but no account at any branch of the bank

Relational Calculus:

$$
\{t \mid \exists s \in borrower(t[\text{cust-name}] = s[\text{cust-name}]) \wedge
\exists u \in loan(u[\text{loan-number}] = s[\text{loan-number}] \wedge u[\text{branch-name}] = \text{'neath'})
\wedge \neg \exists v \in depositor(t[\text{cust-name}] = v[\text{cust-name}])\}
$$

Microsoft Access: Query-By-Example

Field:	ShipCountry	Company Name	ExtendedPrice
Table:			
Total:	Group By	Group By	Sum
Sort:			
Show:			
Criteria:	"Canada"		<10000
or:	"UK"		<10000

1 2

Ship Country	Company Name	SumOfExtendedPrice
Canada	Bottom-Dollar Markets	$28,025.51
Canada	Laughing Bacchus	$522.50
Canada	Mére Paillarde	$37,123.65
UK	Around the Horn	$14,602.15
UK	B's Beverages	$7,383.90

Ship Country	Company Name	SumOfExtendedPrice
Canada	Laughing Bacchus	$522.50
UK	B's Beverages	$7,383.90

Base on:

Relational calculus theory

Usage:

Define what you want

Not how you get it

DBMS generate the query code for you.

For quick exploration

No knowledge of SQL

Declarative Programming

- CS-205

- A different way of thinking.

SQL is built on both:

Relational Algebra (Procedure, step by step)

Relational Calculus (<u>**Declarative**</u>)

You declare what you want.