Relational Model 4: Relational Calculus

Gary KL Tam

Department of Computer Science Swansea University

• Is Relational Calculus mathematics?

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

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

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

- 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 is procedural.
- Relational calculus is declarative

- 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 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 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 (\land) , or (\lor) , not (\neg)
- Implication (\Rightarrow): $x \Rightarrow y$, if x is true, then y is true $x \Rightarrow y \equiv \neg x \lor y$

Predicate Calculus Formula

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

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

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

[Selection]

Step1: go to loan table

Step2: go through each tuple in loan

Step2-1: select the tuple with amount > 1200

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

Query 1:

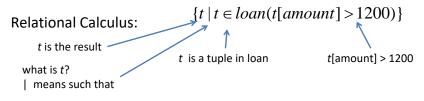
Find the loan-number, branch-name and amount for loans of over £1200

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



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

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

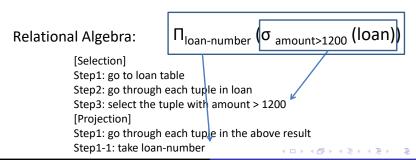
Query 2:

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

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

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

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



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

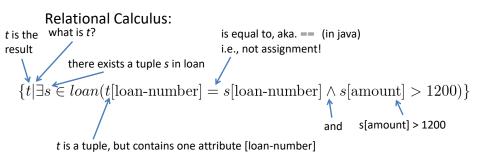
Relational Calculus:

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

loan	loan-num	ber	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



borrower	loan-number	cust-name	depositor	cust-name	acct-num
	L110	Gary		Gary	123
	L223	Maple		April	345
	L331	Syrup		Dave	567

Query 3:

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

borrower	loan-number	cust-nam
	L110	Gary
	L223	Maple

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

Query 3:

1331

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

Relational Algebra:

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

Steps:

Syrup

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

Syrup

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

Query 3:

1331

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

Relational Calculus:

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



b

orrower	loan-number	cust-name	depositor	cust-name	acct-num
s:	L110	Gary	u:	Gary	123
s:	L223	Maple	u:	April	345
s:	L331	Syrup	u:	Dave	567
		t		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

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

there exists a tuple u in depositor

t is the cust-name of s or u



cust-name

Gary April

Dave Maple

Svrup

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

Query 4:

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

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

Ioan-numberbranch-nameamountL110swansea1530L223cardiff2140L331neath1000

Query 4:

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

Relational Algebra:

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

orrower	loan-number	cust-name	loan
	L110	Gary	
	L223	Maple	
	1331	Syrun	

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

amount

1530

2140

1000

branch-

Query 4:

Find the names of all customers having a loan at the Neath

loan-

branch

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

name
Syrup

borrower	loan-number	cust-name	loan	le
	L110	Gary		L
	L223	Maple		L
	L331	Syrup		L

loan-numberbranch-nameamountL110swansea1530L223cardiff2140L331neath1000

Query 4:

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

Relational Calculus:

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

$$\exists u \in loan(u[\text{loan-number}] = s[\text{loan-number}] \land u[\text{branch-name}] = \text{`neath'})]$$

borrower	loan-number	cust-name	loan	loan-number	branch-name	amount
	L110	Gary		L110	swansea	1530
	L223	Maple		L223	cardiff	2140
s:	L331 (Syrup	u:	L331	neath	1000
	A.	\rightarrow_t		JA		

Query 4:

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

Relational Calculus:

$$\{t | \exists s \in borrower(t[cust-name] = s[cust-name]) \land \}$$

 $\exists u \in loan(u[loan-number] = s[loan-number] \land u[branch-name] = `neath')$

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

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

loan

	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

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 Algebra:

 $\Pi_{cust-name} \left(\sigma_{branch-name='neath'} \left(loan \bowtie borrower \right) \right) - \Pi_{cust-name} \left(depositor \right)$ $\left(\begin{array}{c} cust-loan-in-neath \\ Maple \\ Syrup \end{array} \right) - \left(\begin{array}{c} Gary \\ Maple \\ Dave \end{array} \right)$ Results: Syrup

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

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

Relational Calculus:

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

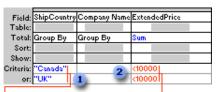
$$\{t | \exists s \in borrower(t[\text{cust-name}] = s[\text{cust-name}]) \land \\ \exists u \in loan(u[\text{loan-number}] = s[\text{loan-number}] \land u[\text{branch-name}] = \text{`neath'}) \\ \land \bigcap \exists v \in depositor(t[\text{cust-name}] = v[\text{cust-name}])\}$$

loan

borrower	loan-number	cust-name	ioan	loan-number	branci, name	amount
	L110	Gary		L110	swansea	1530
s:	L223 (Maple *	u:	L223	neath	2140
s:	L331	Syrup	u:	L331	neath	1000
Query 5: Find the names of all customers who have a loan at the 'neath' branch, but no account at Gary depositor cust- acct-num Gary 23						
	branch of th		, but ii	o account a	v: Maple	345
Rela	ational Calcu	ılus:			Dave	567

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

Microsoft Access: Query-By-Example



Ship Country	Company Name	SumOf	ExtendedPrice
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	SumOf	Ext	endedPrice
Canada	Laughing Bacchus		П	\$522.50
UK	B's Beverages		П	\$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 (**Declarative**)

