Relational Model 4: Relational Calculus

Gary KL Tam

Department of Computer Science
Swansea University

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

  $x \Rightarrow y \equiv \neg x \vee y$
- 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.

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

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

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

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

Example Query

Projection on both tables

borrower

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

depositor

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

Union of both tables

Query 3:

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

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

Relational calculus:

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

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

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

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

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