The author gratefully acknowledges support from the taxpayers of Québec and of Canada who have paid his salary and research grants while this work was developed at McGill University, and from his students (who built the implementations and investigated the data structures and algorithms) and their funding agencies.

T. H. Merrett ©98/9

- Relations
- Relational Algebra
  - Unary Operators
  - Binary Operators
- Domain Algebra
  - Scalar Operators
  - Aggregate Operators

Relations

Responsibility (*Agent*	*Item*)	*Location* (*Item*	*Floor*)
Raman	Micro	Micro	1
Raman	Terminal	Terminal	1
Smith	V.C.R.	Terminal	2
Hung	Micro	Videodisk	2

Properties of Relations

- All rows are distinct.
- The ordering of rows is immaterial.
- Each column is labelled, making the ordering of columns insignificant.
- The value in each row under a given column is "simple".

First two imply relations are sets.

Fourth adds "first normal form".

Relational Algebra

Unary Operators: Project

Responsibility
(*Agent* *Item*)

Raman	Micro
Raman	Terminal
Smith	V.C.R.
Hung	Micro

[*Item*] **in** *Responsibility*
(*Item*)

Micro
Terminal
V.C.R.

Location
(*Item* *Floor*)

Micro	1
Terminal	1
Terminal	2
Videodisk	2

[*Item*] **in** *Location*
(*Item*)

Micro
Terminal
Videodisk

Unary Operators: Select

Responsibility

(*Agent*	*Item*)
Raman	Micro
Raman	Terminal
Smith	V.C.R.
Hung	Micro

where *Item*=Micro **in** *Responsibility*

(*Agent*	*Item*)
Raman	Micro
Hung	Micro

Location

(*Item*	*Floor*)
Micro	1
Terminal	1
Terminal	2
Videodisk	2

where *Item*=Micro **in** *Location*

(*Item*	*Floor*)
Micro	1

Binary Operators: The Natural Join (ijoin)

Responsibility

(*Agent*	*Item*)
Raman	Micro
Raman	Terminal
Smith	V.C.R.
Hung	Micro

Location

(*Item*	*Floor*)
Micro	1
Terminal	1
Terminal	2
Videodisk	2

Responsibility **ijoin** *Location*

(*Agent*	*Item*	*Floor*)
Raman	Micro	1
Hung	Micro	1
Raman	Terminal	1
Raman	Terminal	2

Reinforcement:

Complexities

Project: $\mathcal{O}(N \log N)$

Select: $\mathcal{O}(N)$

Join: $\mathcal{O}(MN)$

Join Size

$R(A,B)$		$S(B,C)$
R	**ijoin**	S
3	b1	2
2	b2	4
5	b3	7

49 tuples

Syntax

$R(A, B),\ S(C, D),$
$R[B$ **ijoin** $C]S\colon\ (A, {}^{B}_{C}, D)$

$R(A, B, C),\ S(D, E, F),$
$R[B, C$ **ijoin** $D, E]S\colon\ (A, {}^{B}_{D}, {}^{C}_{E}, F)$

$R(A, B, C),\ S(D, E, F),$
$R[C$ **ijoin** $D]S\colon\ (A, B, {}^{C}_{D}, E, F)$

Algebraic Principles

- Things

- Operations on things

The *Principle of Abstraction*

the structure and the context of a thing should be of no concern to the operation

The *Principle of Closure*

operations on a thing should produce things of the same type

Domain Algebra

I Horizontal operations

e.g., **let** *TotMk* **be** *Asst* + *Exam*;

NewMk (*Student*	*Asst*	*Exam*)	*TotMk*
Smith	25.	60.	85.
Jones	28.	66.	94.
Brown	20.	50.	70.
Hung	24.	58.	82.
Raman	24.	66.	90.

N.B. *Virtual* domains (principle of abstraction)

Actualization via Project

> **let** *TotMk* **be** *Asst* + *Exam*;
> *Result* <– [*Student*, *TotMk*] **in** *NewMk*;

Any expression allowed that can be actualized on each tuple separately.

Domain Algebra

Frequently Asked Questions

- What if *Asst* and *Exam* are also in some other relation?
  Nothing is affected: TotMk could be actualized there, too.

- What if *Asst* and *Exam* each come from different relations?
  Use relational algebra to put the relations together before actualizing.

- What if we update *NewMk* after actualizing *TotMk*?
  TotMk is an actual attribute of Result and unaffected by changes to other relations.

Domain Algebra

II Vertical operations e.g. of *Reduction*

let *Total* **be red** + **of** *TotMk*;

let *Count* **be red** + **of** *One*;

let *AvgMk* **be** *Total*/*Count*;

(**let** *AvgMk* **be** (**red** + **of** *TotMk*)/(**red** + **of** 1);)

Course (*Student*	*Asst*	*Final*	*TotMk*	*One*)	*Total*	*Count*	*AvgMk*
Smith	25.	60.	85.	1	421	5	84.2
Jones	28.	66.	94.	1	421	5	84.2
Brown	20.	50.	70.	1	421	5	84.2
Hung	24.	58.	82.	1	421	5	84.2
Raman	24.	66.	90.	1	421	5	84.2

Vertical operations

Operators: $+, \times$, **max**, **min**, **and**, **or**

e.g., **let** *MaxMk* **be red max of** *TotMk*;

Course (*Student*	*Asst*	*Final*	*TotMk*	*One*)	*MaxMk*
Smith	25.	60.	85.	1	94
Jones	28.	66.	94.	1	94
Brown	20.	50.	70.	1	94
Hung	24.	58.	82.	1	94
Raman	24.	66.	90.	1	94

Of course, the attributes outside the (..) are *virtual*, so there is no "waste": the programmer actualizes and controls the waste.

Class <– [*AvgMk*, *MaxMK*] **in** *Course*;

Course (*AvgMk*	*MaxMk*)
84.2	94

Domain Algebra

II Vertical operations

e.g. of *Equivalence Reduction*

let *STot* **be equiv** + **of** *Mark* **by** *S#*;
let *CTot* **be equiv** + **of** *Mark* **by** *C#*;

StuCour

(*S#*	*C#*	*Mark*)	*STot*	*CTot*
1	1	73.	233.	213.
1	2	82.	233.	147.
1	3	78.	233.	78.
2	1	64.	64.	213.
2	1	76.	141.	213.
2	2	65.	141.	147.