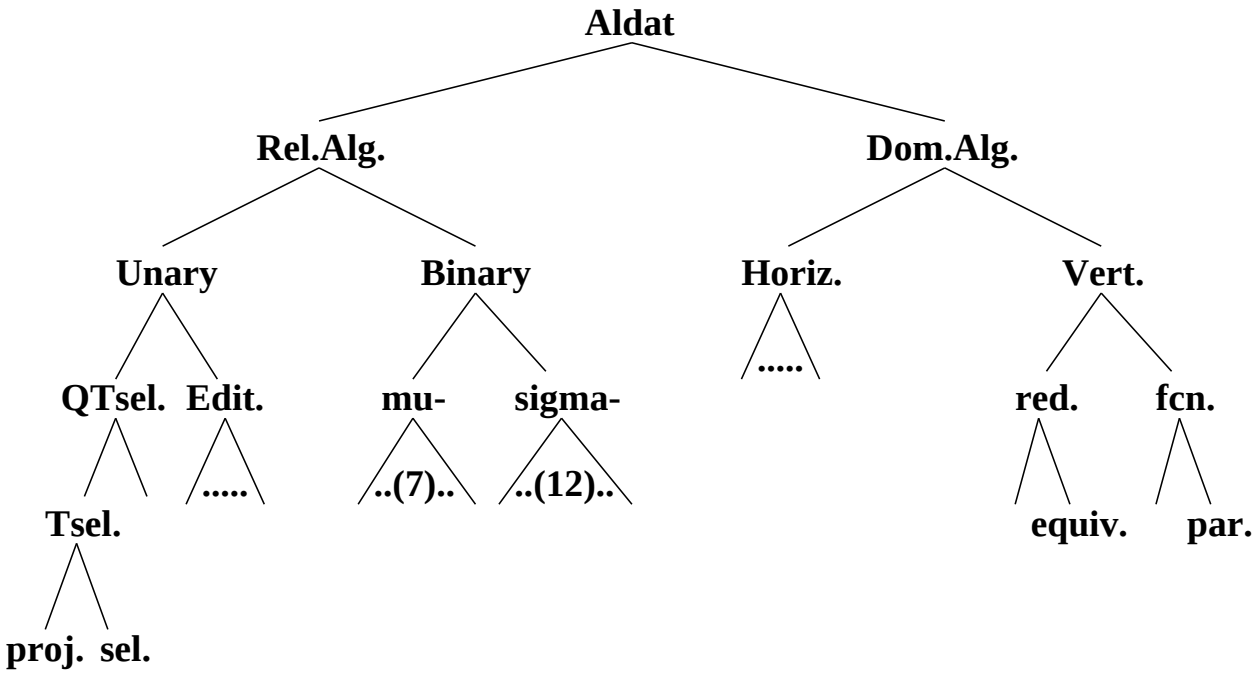
The author gratefully acknowledges support from the taxpayers of Québec and of Canada who have paid his salary and research grants while this work was developed at McGill University, and from his students (who built the implementations and investigated the data structures and algorithms) and their funding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

T. H. Merrett ©98/9

A Taxonomy of Aldat



plus Updates, Computations, Recursion, Nesting

The Relational Algebra

Binary Operators

- μ-joins (*"set"-valued*)

ijoin $\cap$ **natjoin**	
ujoin $\cup$	**ljoin**
sjoin $+$	**rjoin**
djoin $-$	**drjoin**

- σ-joins (*"truth"-valued*)

$\supset$ **gtjoin**	$\supseteq$ **div**	$=$ **eqjoin**	$\subseteq$ **lejoin**	$\subset$ **ltjoin**	$\circledcirc$ **sep**
$\not\supset$ **!gtjoin**	$\not\supseteq$ **!gejoin**	$\neq$ **!eqjoin**	$\not\subseteq$ **!lejoin**	$\not\subset$ **!ltjoin**	$\not\circledcirc$ **icomp**

Unary Operators

- QT-expressions
  - T-selectors: *project*, *select*
  - Quantifiers
- Editors

The μ-join Family

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

ujoin

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 **ujoin** *Location*

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

sjoin

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 **sjoin** *Location*

(*Agent*	*Item*	*Floor*)
Smith	V.C.R.	—
—	Videodisk	2

ljoin

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

Responsibility **ljoin** *Location*

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

rjoin

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 **rjoin** *Location* (*Agent* *Item* *Floor*)

Raman	Micro	1
Hung	Micro	1
Raman	Terminal	1
Raman	Terminal	2
—	Videodisk	2

djoin

Responsibility (*Agent* *Item*)

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

Location (*Item* *Floor*)

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

Responsibility **djoin** *Location*

(*Agent* *Item* *Floor*)

Agent	*Item*	*Floor*
Smith	V.C.R.	—

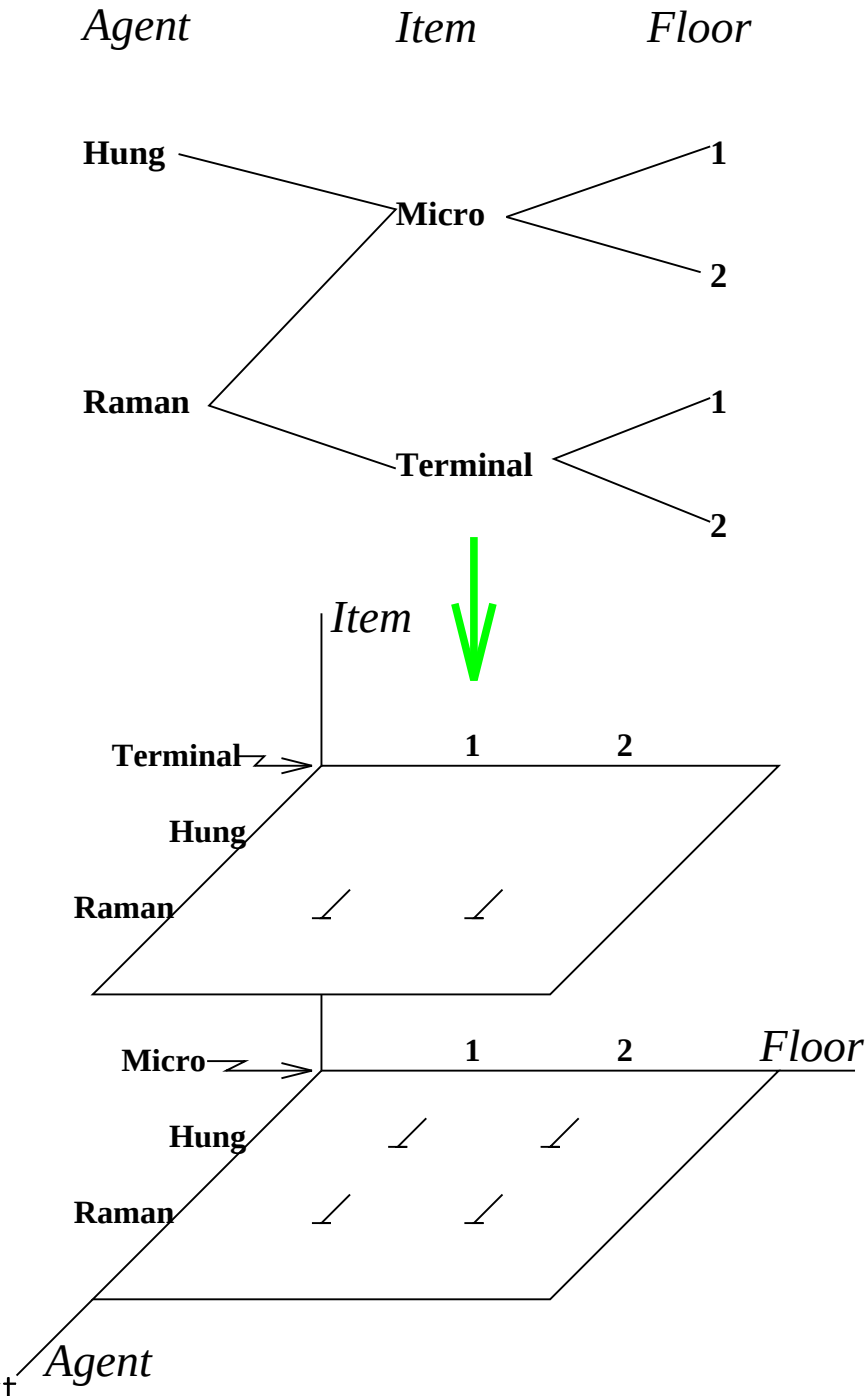
That is (a column of all nulls $\equiv$ a column that is not present):

Responsibility **djoin** *Location*

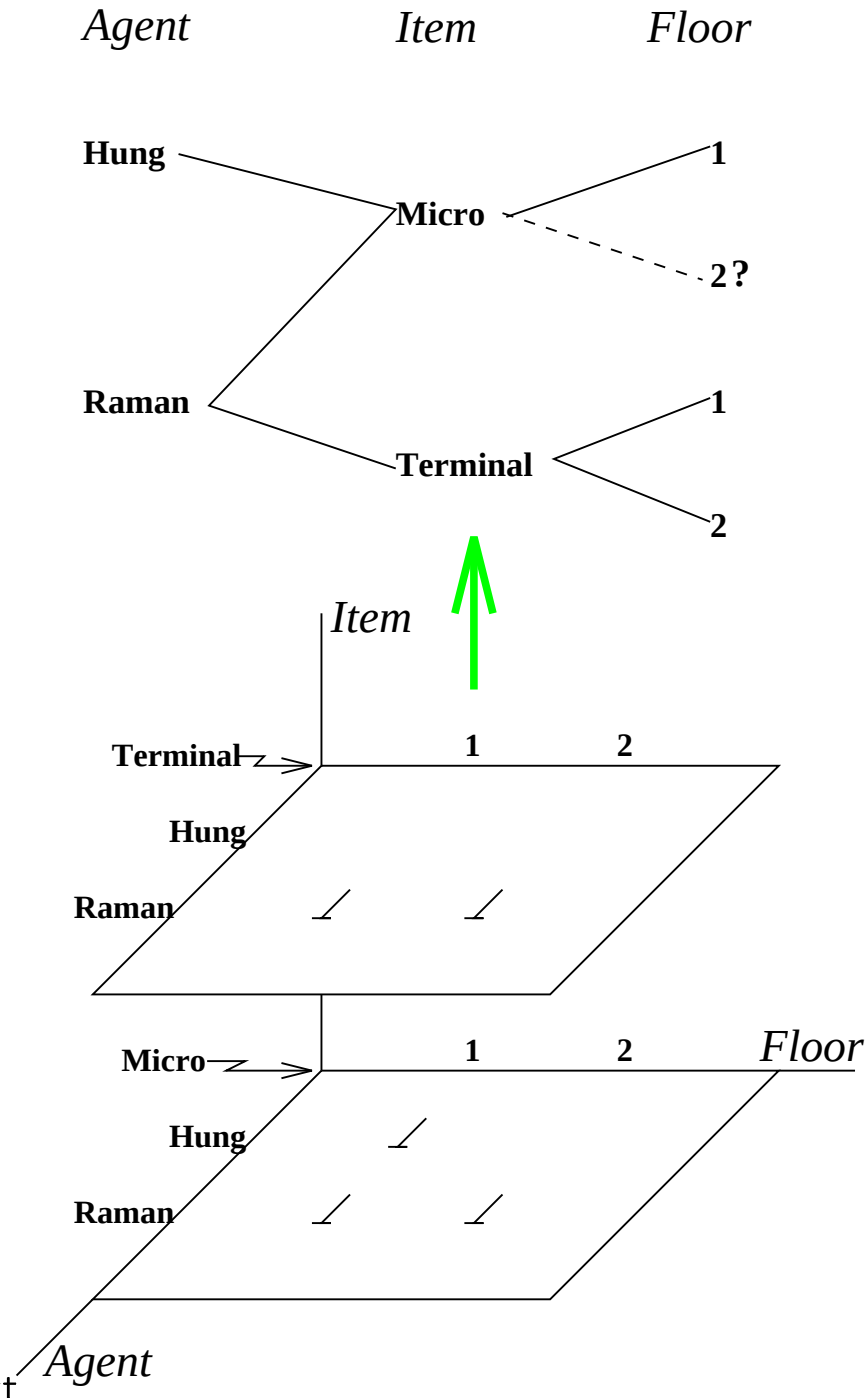
(*Agent* *Item*)

Agent	*Item*
Smith	V.C.R.

Natural Join = **ijoin** and Decomposition



Natural Join = **ijoin** and Non-decomposability



Natural Join and Decomposition

(Back to *Orderbook*)

Orderbook = *Orders* **ijoin** *Ordline*

Aldat Notation (join = **ijoin**, **ujoin**, **sjoin**, **ljoin**, **rjoin**)

1. Assigning the result
   Warehouse <− *Responsibility* **ijoin** *Location*;
2. Join on common attributes

$$R(X,Y),\ S(Y,Z) \quad T <- R \text{ join } S; \quad T(X,Y,Z)$$

$$T <- R \textbf{ djoin } S; \quad T(X,Y)$$

3. Join on different attributes

$$Q(W,X),\ S(Y,Z) \quad T <- Q[X \text{ join } Y]S; \quad T(W,X,Y,Z)$$

(X and Y are aliases in T.)
N.B. Q **ijoin** S gives Cartesian product: X, Y *not* aliases.

4. Join on several attributes

$$U(A,B,C,X,Y,Z),\ V(X,Y,Z,D,E)$$

$$T <- U \text{ join } V;$$

$$T(A,B,C,X,Y,Z,D,E)$$

Union Join — Example

CourseWork (*Student* *Asst*)

Smith	25
Hung	24
Raman	24
Tremblay	29

Final (*Student* *Exam*)

Smith	60
Hung	58
Raman	66
Trombly	68

CourseWork **ujoin** *Final*

(*Student* *Asst* *Exam*)

Smith	25	60
Hung	24	58
Raman	24	66
Tremblay	29	
Trombly		68

CourseWork **sjoin** *Final*

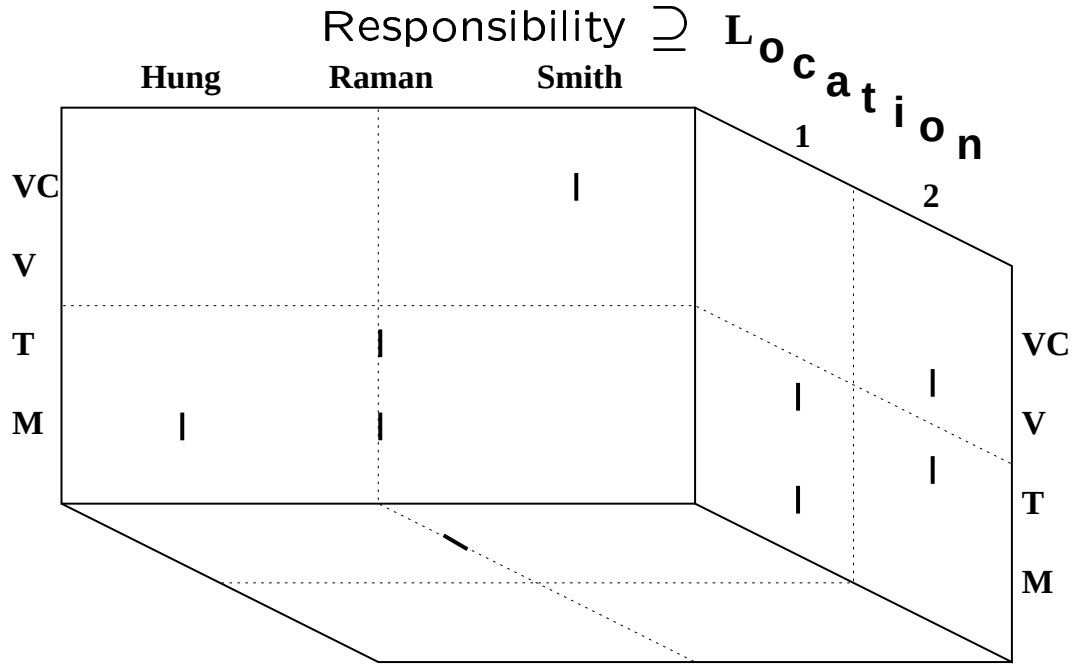
(*Student* *Asst* *Exam*)

Tremblay	29	
Trombly		68

The σ-join Family

division **sup**

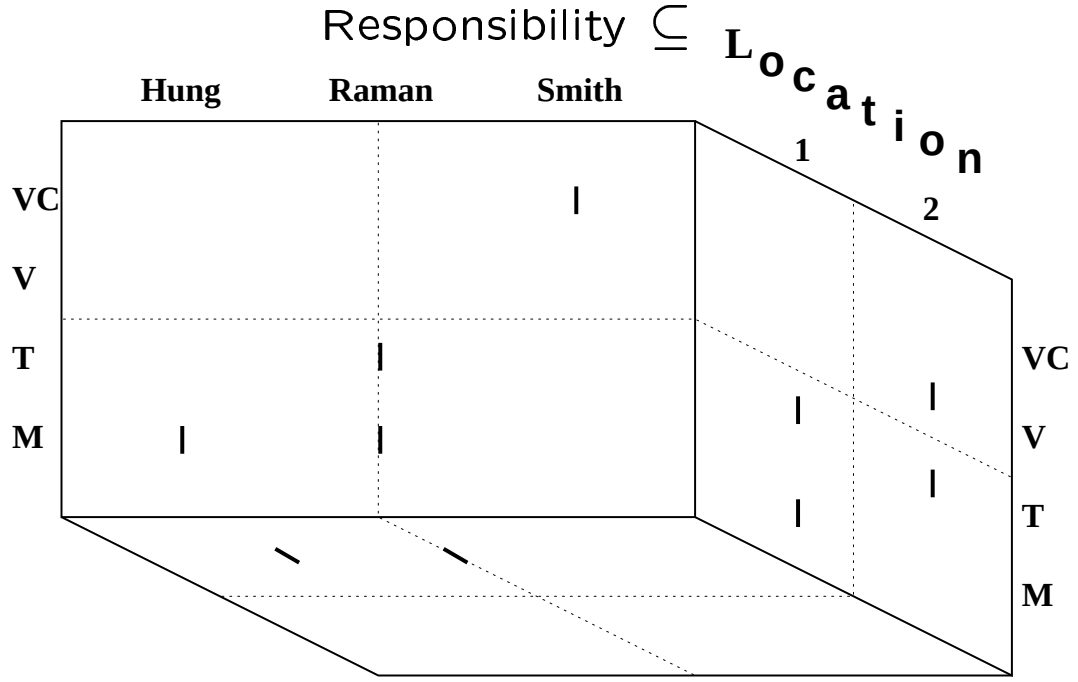
Responsibility (*Agent*	*Item*)	*Location* (*Item*	*Floor*)
Raman	Micro	Micro	1
Raman	Terminal	Terminal	1
Smith	V.C.R.	Terminal	2
Hung	Micro	Videodisk	2



*Agent*s for all *Item*s on a *Floor*

subset join **sub**

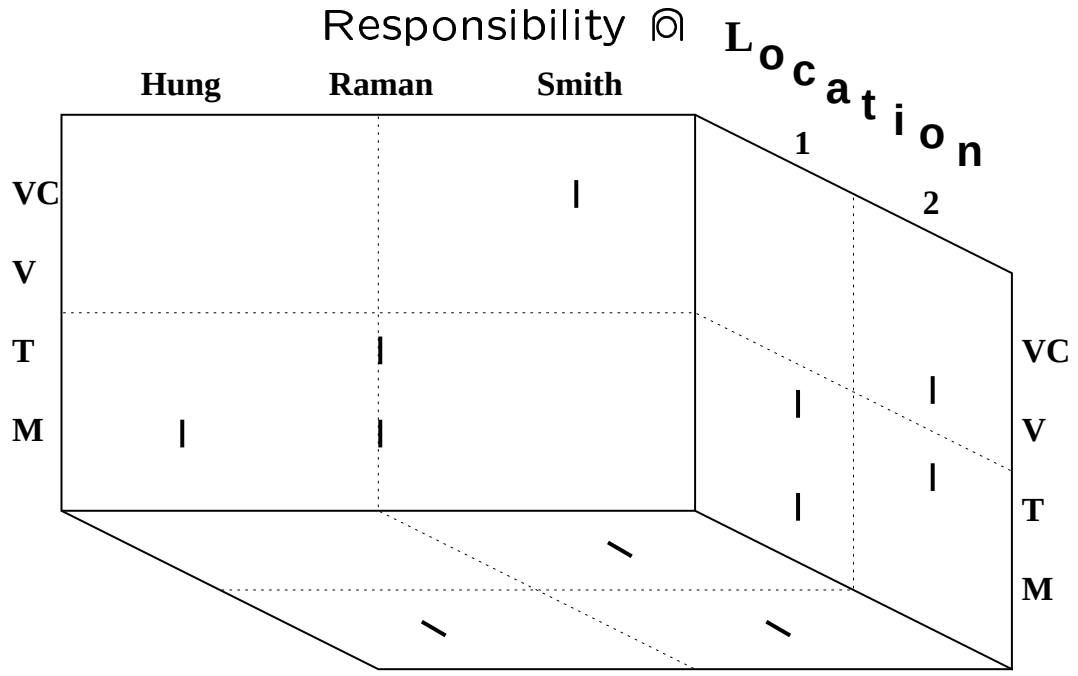
Responsibility (*Agent*	*Item*)	*Location* (*Item*	*Floor*)
Raman	Micro	Micro	1
Raman	Terminal	Terminal	1
Smith	V.C.R.	Terminal	2
Hung	Micro	Videodisk	2



*Floor*s with all *Item*s by an *Agent*

disjoint join **sep**

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

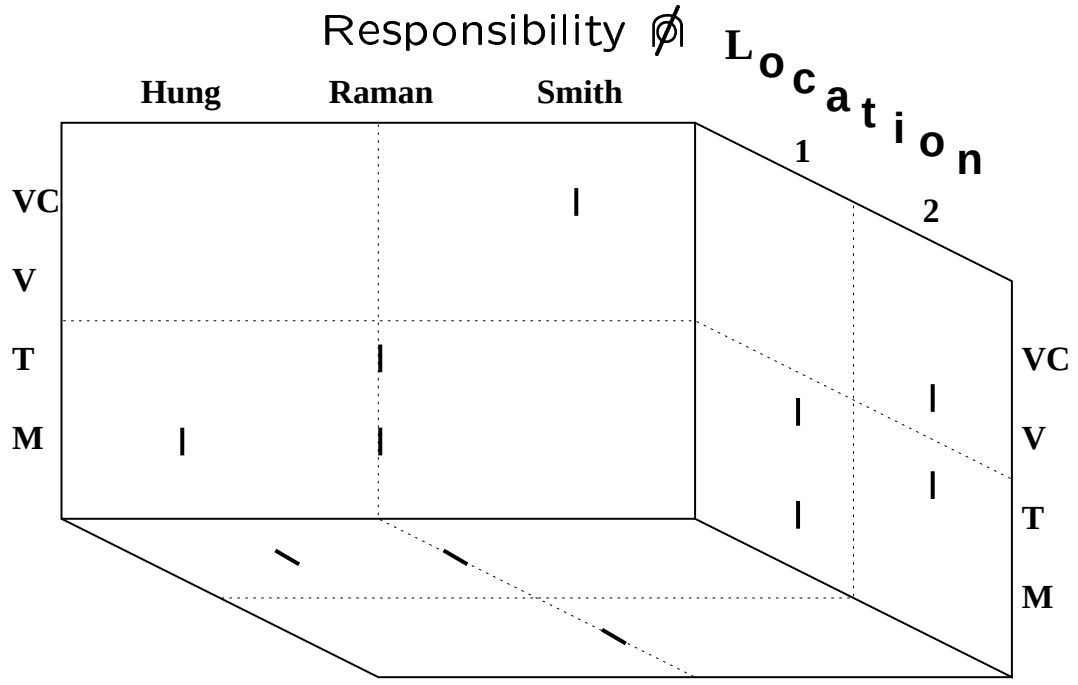


*Agent*s for no *Item*s on a *Floor*

*Floor*s with no *Item*s by an *Agent*

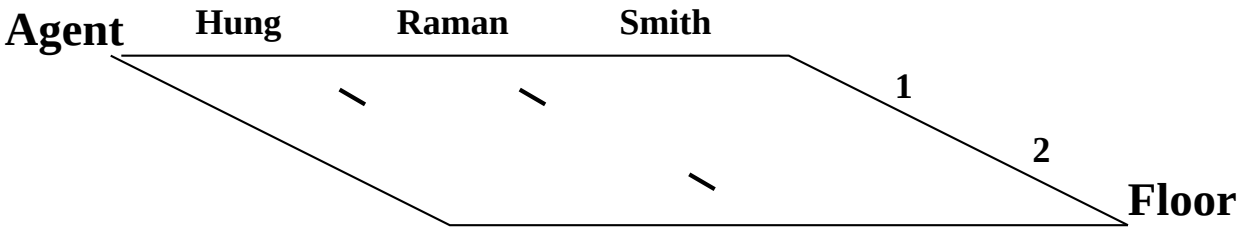
natural composition **icomp**

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

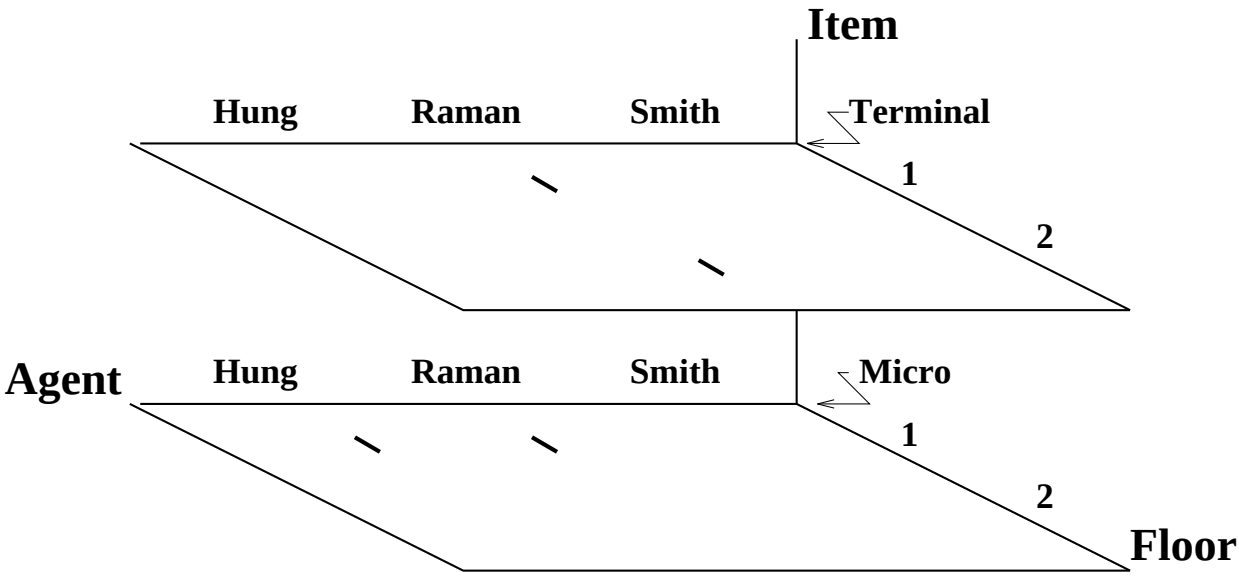


*Agent*s for some *Item*s on a *Floor*
*Floor*s with some *Item*s by an *Agent*

Natural Composition = $\not\bowtie$



and **Natural Join**



Natural Composition — Example

Parent(*Sr* *Jr*)

Sam	Pete
Pete	Sue
Pete	Joe

Grandparent(*Sr* *Jr*)

Sam	Sue
Sam	Joe

Grandparent <− *Parent*[*Jr* **icomp** *Sr*]*Parent*;

σ-joins — Example

Given:

PC(*Part*	*Colour*)
P1	red
P1	blue
P1	yellow
P2	red
P2	green
P3	blue
P3	purple

Find:

1. parts that come in all primary colours
2. all colours except those used in parts P1 *or* P2
3. all colours except those used in parts P1 *and* P2

1. $PC \supseteq$ {red, yellow, blue} (*Part*)
   P1
2. $PC \circledcirc$ {P1, P2} (*Colour*)
   purple
3. $PC \not\supseteq$ {P1, P2} (*Colour*)
   blue
   yellow
   green
   purple

Aldat Notation (join = [!]**gtjoin**, **div**, **!gejoin**, [!]**eq-join**, [!]**lejoin**, [!]**ltjoin**, **sep**, **icomp**), ...

1. Join on common attributes

$$R(X,Y),\ S(Y,Z) \quad T <\!\!- R \text{ join } S; \quad T(X,Z)$$

2. Join on different attributes

$$Q(W,X),\ S(Y,Z) \quad T <\!\!- Q[X \text{ join } Y]S; \quad T(W,Z)$$

3. Join on several attributes

$$U(A,B,C,X,Y,Z),\ V(X,Y,Z,D,E)$$

$$T <\!\!- U \text{ join } V;$$

$$T(A,B,C,D,E)$$

4. What about *set* comparisons?

$$R(Y),\ S(Y) \quad T <\!\!- R \text{ join } S; \quad T()$$

A *nullary* relation can have only two values, empty or not empty: it is a *boolean*;

5. Declaring and initializing relations

domain *Colour* **strg**;

relation C(*Colour*)<− {`"red", "yellow", "blue"`};

Relational Recursion

1. Transitive Closure

An *ancestor* **is** a *parent* **or** the *parent* **of** an *ancestor*.

Relation *Parent*(*Sr, Jr*) <− ... ;
Relation *Ancestor*(*Sr, Jr*);

Ancestor **is** *Parent* **ujoin**
Parent[*Jr* **icomp** *Sr*]*Ancestor*;

is indicates a *view*, which is not calculated until the result is used, as in **pr!!***Ancestor*

Ancestor is a *recursive view*, and has the meaning:

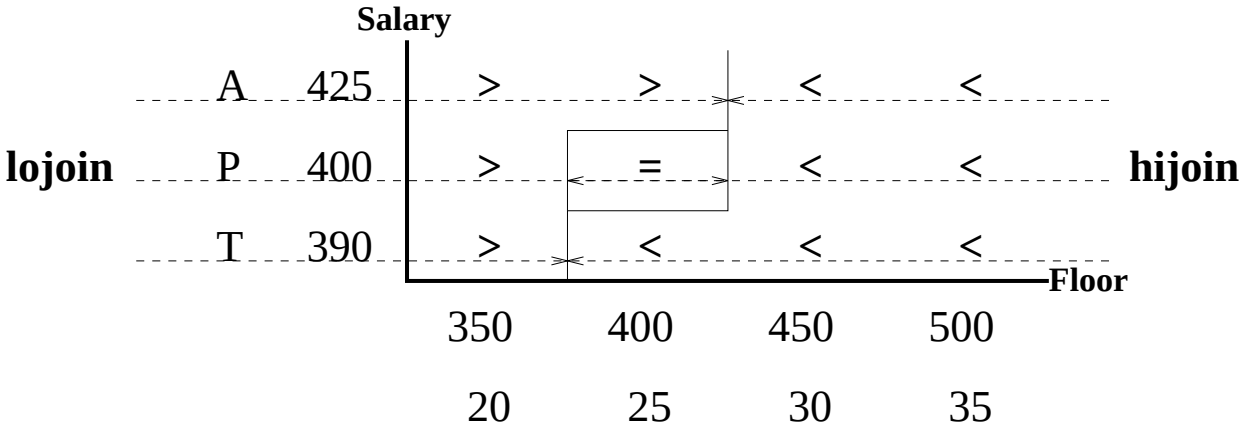
Ancestor <− ϕ // empty
repeat *Test* <− *Ancestor*
Ancestor <− *Parent* **ujoin**
Parent[*Jr* **icomp** *Sr*]*Ancestor*
until *Ancestor*=*Test*

Range Joins

Sals (*Emp*	*Salary*)	*Taxes* (*Floor*	*Tax%*)
T	390	350	20
P	400	400	25
A	425	450	30
		500	35

Sals[*Salary* **lojoin** *Floor*]*Taxes*

(*Emp*	*Salary*	*Floor*	*Tax%*)
T	390	350	20
P	400	400	25
A	425	400	25



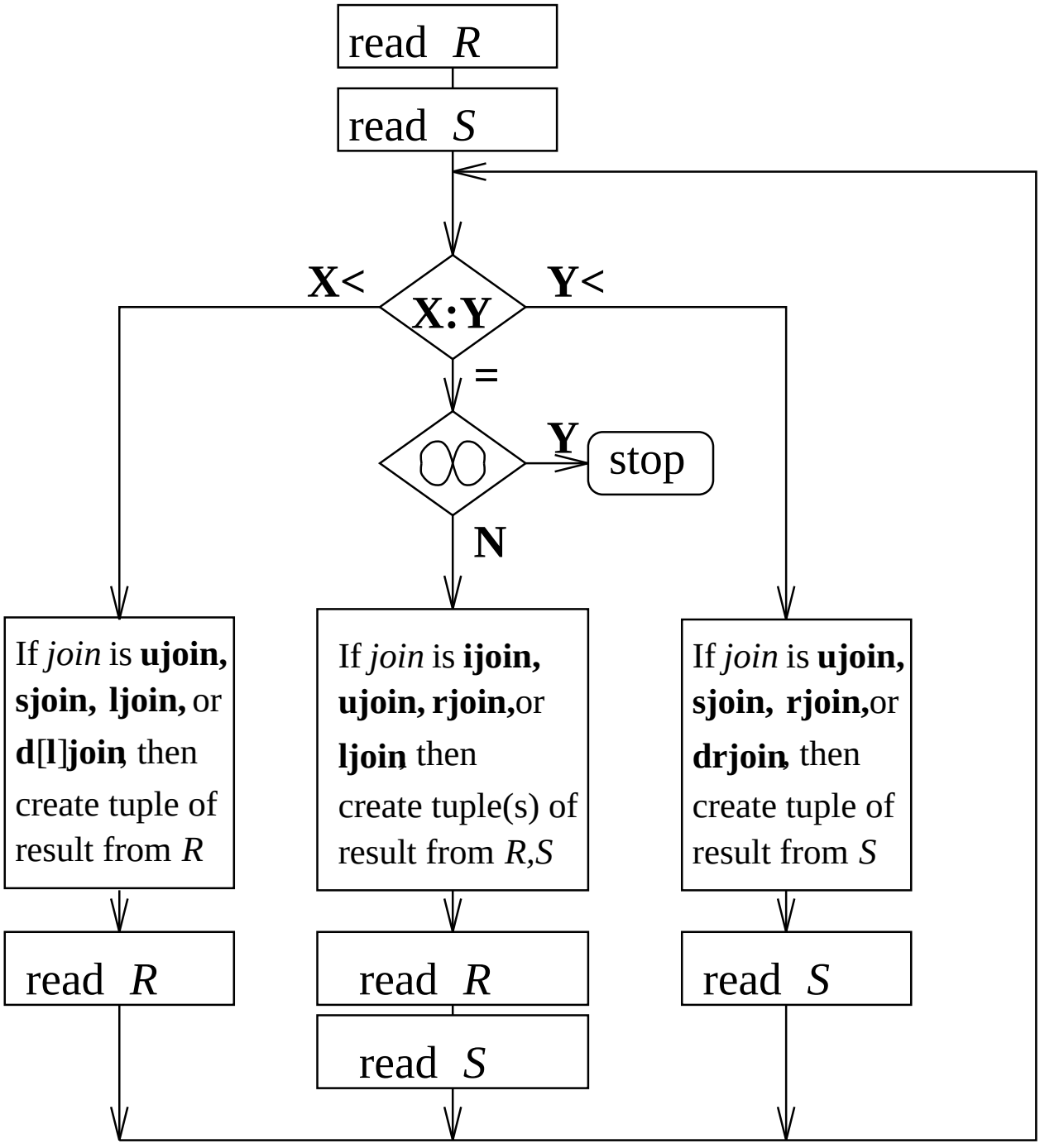
Implementing Joins

(Briefly: to reinforce the ideas, not to dwell on the machinery underneath)

Sort-Merge Techniques
(Sequential Files) for:

- μ-join
- σ-join
- Range-join

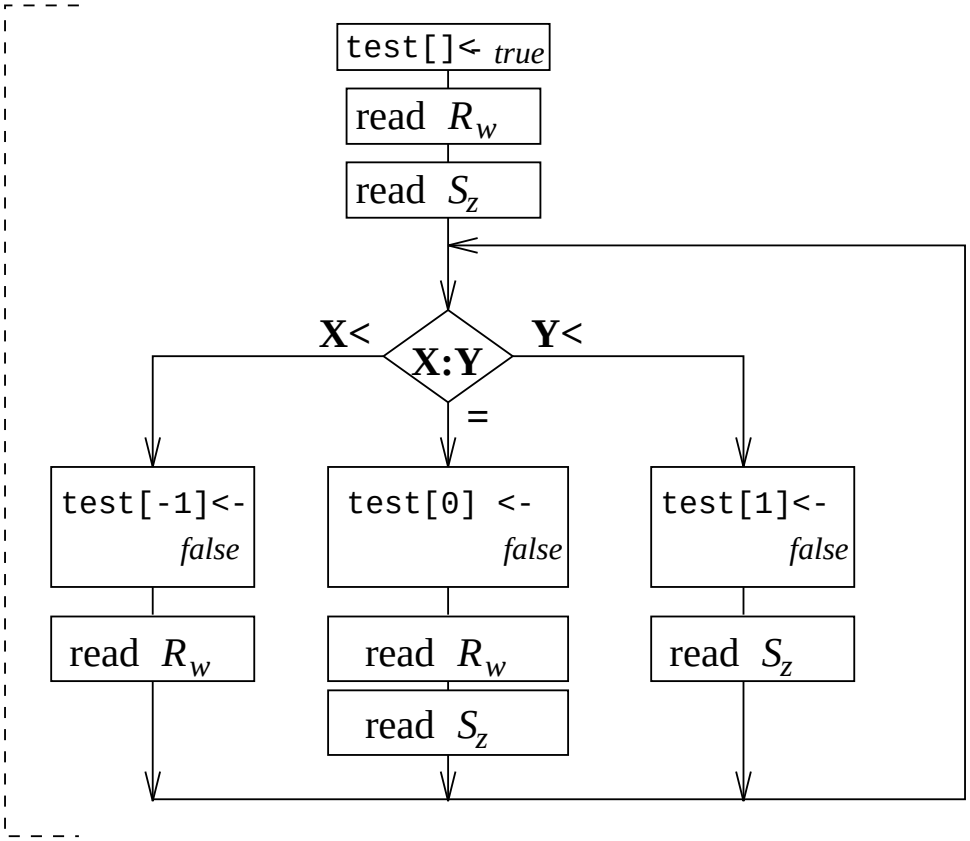
Implementing μ-join $R[X\ join\ Y]S$



Implementing σ-join $R[X \ join \ Y]S$

$R(W, X)$ $S(Y, Z)$

Set `test[-1]` ($\subseteq$), `test[0]` ($\circledcirc$), `test[1]` ($\supseteq$) to *true*.



for each $w \in [W]$ **in** R, each $z \in [Z]$ **in** S

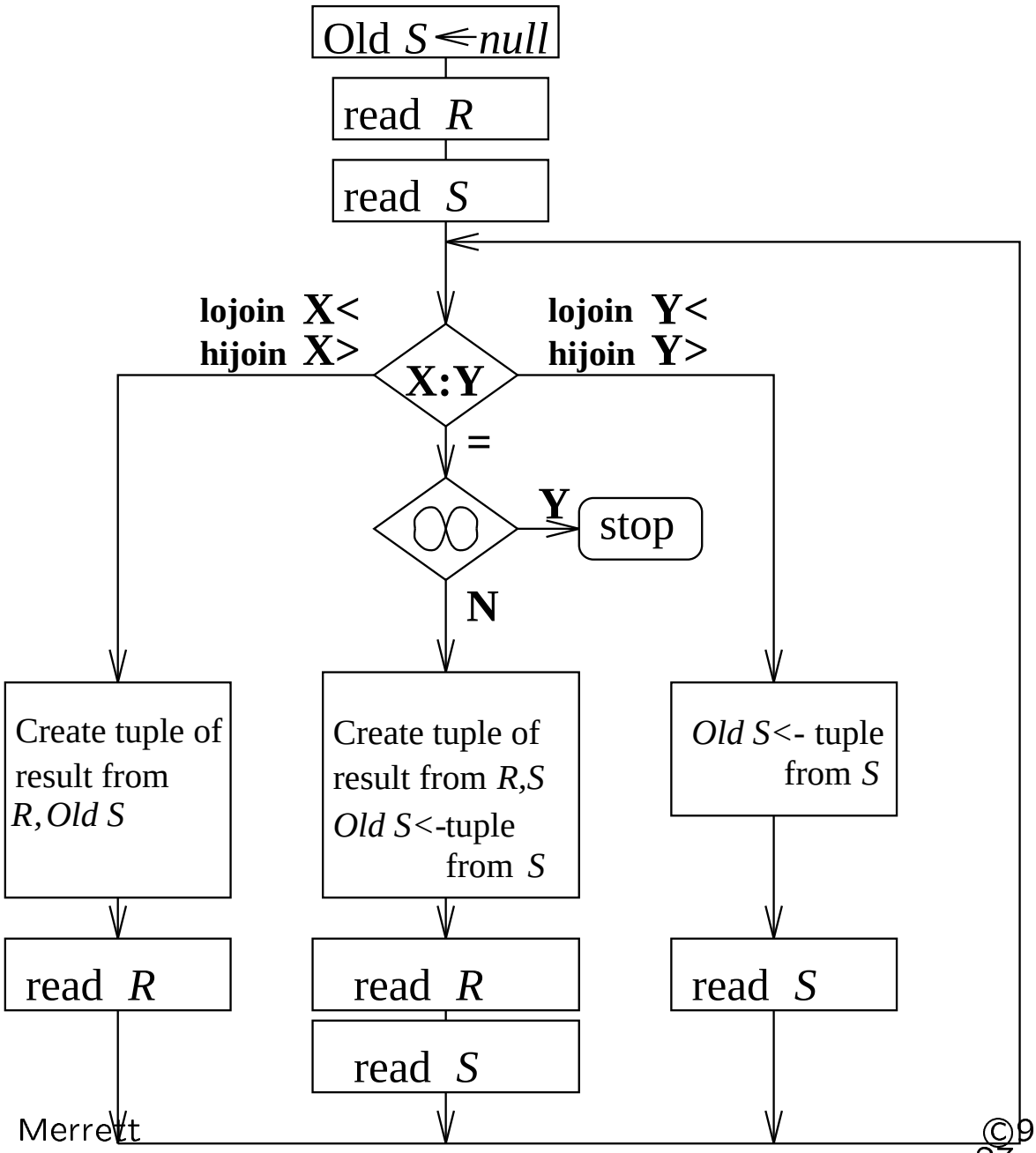
Create tuple in result from w, z each time around outer loop if the conditions shown hold:

join=	$\supset$	$\supseteq$	$=$	$\subseteq$	$\subset$	$\circledcirc$
`test`[$\subseteq$]	F		T	T	T	
`test`[$\circledcirc$]						T
`test`[$\supseteq$]	T	T	T		F	

(Complement the decision for the complementary joins.)

Implementing Range-join $R[X \ join \ Y]S$

Sort in ascending order for **lojoin**, descending order for **hijoin**



The Relational Algebra

Unary Operators

- **QT-Expression**
  - T-Selector
    - Project
    - Select
- Relational Editor

QT-Expression: 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

QT-Expression: 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

QT-Expression: T-Selector

Responsibility
(*Agent* *Item*)

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

[*Agent*] **where** *Item*=Micro **in** *Responsibility*
(*Agent*)

Raman

Hung

Location
(*Item* *Floor*)

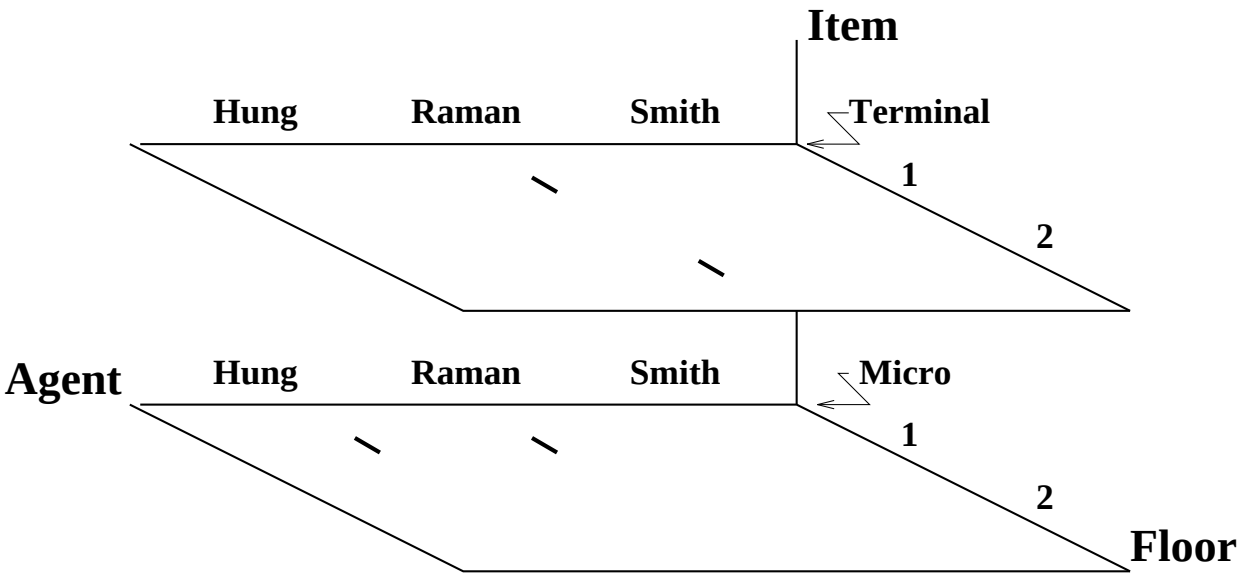
Micro	1
Terminal	1
Terminal	2
Videodisk	2

[*Floor*] **where** *Item*=Micro **in** *Location*
(*Floor*)

1

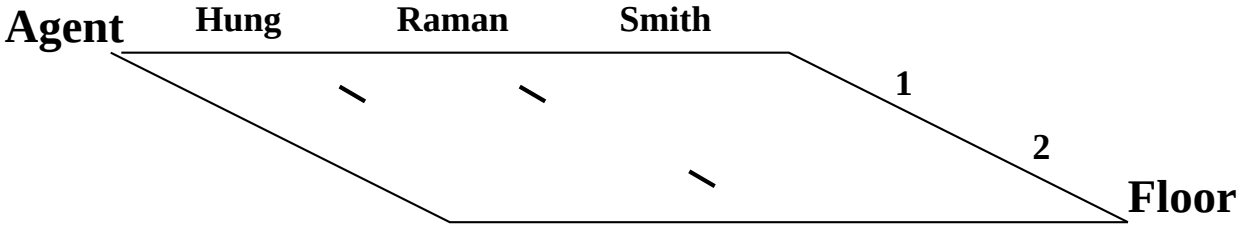
Natural Join

Responsibility **ijoin** *Location*



and **Projection** and **Natural Composition**

[*Agent, Floor*] **in** (*Responsibility* **ijoin** *Location*)



More Relational Recursion

2. Inference Engine

[*New*]*Facts* (*Concl*)	*Horn* (*Rule#*	*Ante*	*Concl*)
lays eggs	1	lays eggs	is bird
has feathers	1	has feathers	is bird
swims	2	flies	is bird
is brown	2	is not mammal	is bird
———	3	is bird	is duck
is bird	3	swims	is duck
———	3	is brown	is duck
is duck	4	is bird	is duck
	4	swims	is duck
	4	is green	is duck
	4	is red	is duck
	5	is duck	migrates
	5	is not tame	migrates

NewFacts **is** *Facts* **ujoin**
[*Concl*] **in** (*NewFacts*[*Concl* $\supseteq$ *Ante*]*Horn*)

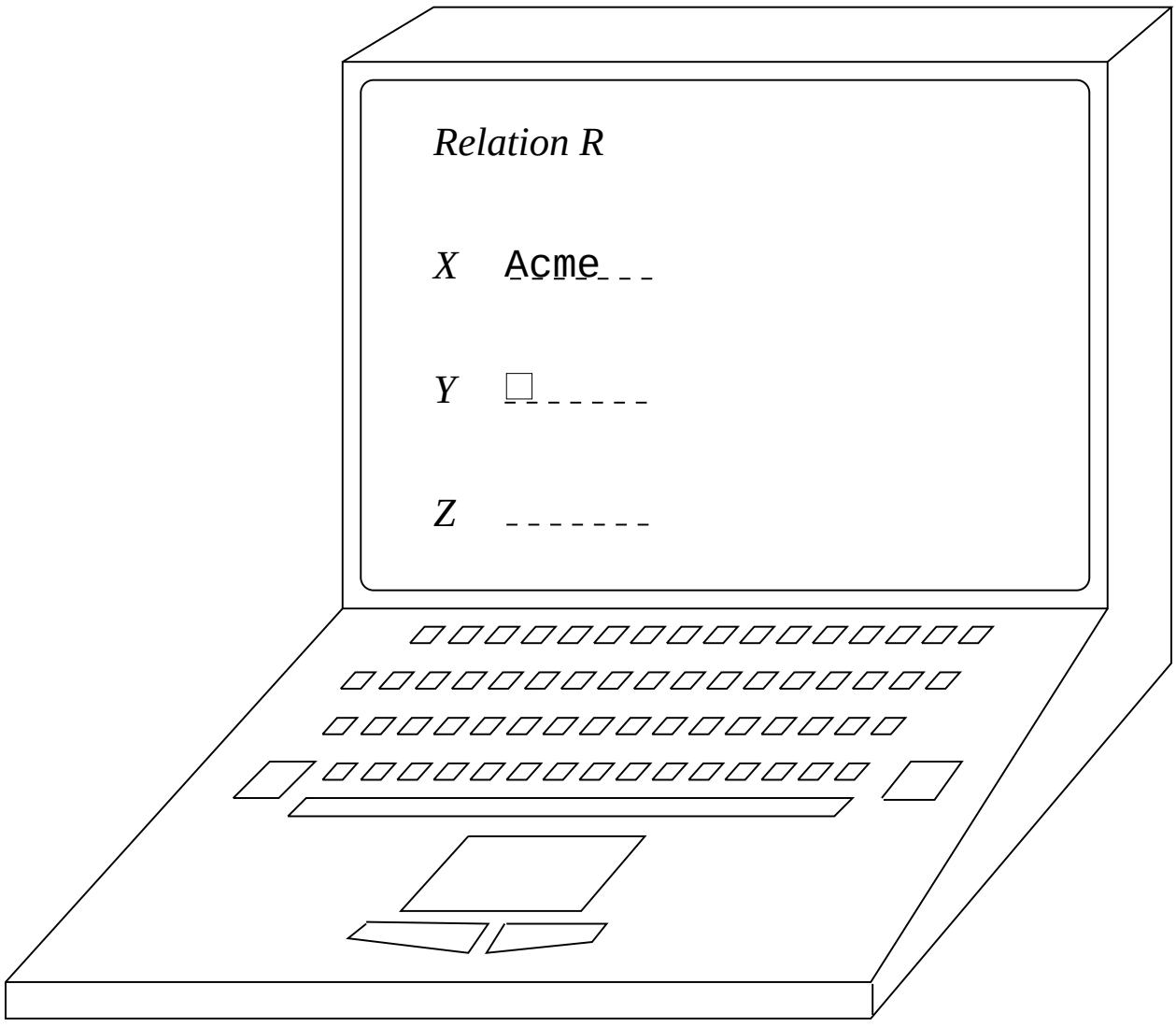
Editors

- graphics editor **gedit**
- spreadsheet editor **spredit**
- prolog editor
- apl editor
- ...
- relational editor **edit**

Relational Editor

1. Programmer-user $R <- [X]$ **edit** R;

2. End-user tuple-at-a-time (*TATI*):



Null Values An approximate treatment

Don't Care, $\mathcal{DC}$

1. $\mathcal{DC}$ is not a value among others.

- $x\ \kappa\ \mathcal{DC}$ is $\mathcal{DC}$ for any comparison,
  $\kappa \in \{<, \leq, =, \geq, >, \neq\}$

- (If we made $x\ \kappa\ \mathcal{DC}$ **false**:
  true $= \neg(x < \mathcal{DC}) = (x \geq \mathcal{DC}) =$ **false**)

- So the θ-joins ($\theta \in \{<, \leq, =, \geq, >, \neq\}$) are not well defined, including the equijoin (natural join), if a join attribute is $\mathcal{DC}$: we get Codd's "maybe" join.

- (If we made $x\ \kappa\ \mathcal{DC}\ \mathcal{DC}$ for $\kappa \in \{<, \leq, \geq, >\}$,
  $x = \mathcal{DC}$ **false**, and $x \neq \mathcal{DC}$ **true**, (ordinary x):
  $$\begin{aligned}\mathcal{DC} &= (x < \mathcal{DC}) = (x \leq \mathcal{DC}) \wedge \neg(x = \mathcal{DC}) \\ &= ((x < \mathcal{DC}) \vee (x = \mathcal{DC})) \wedge \neg(x = \mathcal{DC}) \\ &= (\mathcal{DC} \vee \textbf{false}) \wedge \textbf{true} \\ &= \textbf{false})\end{aligned}$$

Don't Care, $\mathcal{DC}$

2. $\mathcal{DC}$ should have no effect on operations.

- $\mathcal{DC}$ is the right and left identity for any binary operator that preserves type:
- $x + \mathcal{DC} = x$, $\mathcal{DC} + x = x$, $x \times \mathcal{DC} = x$, ...
- $\mathcal{DC} - x = \mathcal{DC} + (-x) = -x$ (by caveat: practice could change this)
- $\mathcal{DC} \div x = \mathcal{DC} \times (\div x) = \div x$
- Unary operations on $\mathcal{DC}$ are ignored: $-\mathcal{DC} = \mathcal{DC}, \neg\mathcal{DC} = \mathcal{DC}$
- (Note that $a\mathcal{DC} + by = a + by$, not by; and $ay = a(\mathcal{DC} + y) = a\mathcal{DC} + ay = a + ay$)

Don't Know, $\mathcal{DK}$

1. $\mathcal{DK}$ ranges over all possible ordinary values.
   - $x\ \kappa\ \mathcal{DK} = \mathcal{DK}$, $\kappa \in \{<, \leq, =, \geq, >, \neq\}$
     Again, we get the "maybe" join.
   - Operators with $\mathcal{DK}$ almost always give $\mathcal{DK}$:
     $\neg\mathcal{DK} = \mathcal{DK}$, $\mathcal{DK} - x = \mathcal{DK}$, $x \times \mathcal{DK} = \mathcal{DK}$,
     x **max** $\mathcal{DK} = \mathcal{DK}$, etc.
   - **true or** $\mathcal{DK}$ **= true**,
     false and $\mathcal{DK}$ **= false**

2. Each $\mathcal{DK}$ ranges independently.
   - $\mathcal{DK} \vee \neg\mathcal{DK} = \mathcal{DK}$, and is not a tautology.
   - So I can't know that two people have the same age, but not know the age.
   - A full treatment might use a *set of variables* for $\mathcal{DK}$.

Null Values

This can be summarized by pretending $\mathcal{DC}$ and $\mathcal{DK}$ *are* extra values which augment each language type.

κ	x	$\mathcal{DC}$	$\mathcal{DK}$	$\kappa \in \{<, \leq, =, \geq, >, \neq\}$
x		$\mathcal{DC}$	$\mathcal{DK}$	
$\mathcal{DC}$	$\mathcal{DC}$	$\mathcal{DC}$	$\mathcal{DC}$	
$\mathcal{DK}$	$\mathcal{DK}$	$\mathcal{DC}$	$\mathcal{DK}$	

and	F	T	$\mathcal{DC}$	$\mathcal{DK}$	**or**	F	T	$\mathcal{DC}$	$\mathcal{DK}$	**not**	
F	F	F	F	F	F	F	T	F	$\mathcal{DK}$	F	T
T	F	T	T	$\mathcal{DK}$	T	T	T	T	T	T	F
$\mathcal{DC}$	F	T	$\mathcal{DC}$	$\mathcal{DK}$	$\mathcal{DC}$	F	T	$\mathcal{DC}$	$\mathcal{DK}$	$\mathcal{DC}$	$\mathcal{DC}$
$\mathcal{DK}$	F	$\mathcal{DK}$	$\mathcal{DK}$	$\mathcal{DK}$	$\mathcal{DK}$	$\mathcal{DK}$	T	$\mathcal{DK}$	$\mathcal{DK}$	$\mathcal{DK}$	$\mathcal{DK}$

$+$	x	$\mathcal{DC}$	$\mathcal{DK}$	$-$	x	$\mathcal{DC}$	$\mathcal{DK}$	$-$	
x		x	$\mathcal{DK}$	x		x	$\mathcal{DK}$	x	$-x$
$\mathcal{DC}$	x	$\mathcal{DC}$	$\mathcal{DK}$	$\mathcal{DC}$	$-x$	$\mathcal{DC}$	$\mathcal{DK}$	$\mathcal{DC}$	$\mathcal{DC}$
$\mathcal{DK}$	$\mathcal{DK}$	$\mathcal{DK}$	$\mathcal{DK}$	$\mathcal{DK}$	$\mathcal{DK}$	$\mathcal{DK}$	$\mathcal{DK}$	$\mathcal{DK}$	$\mathcal{DK}$

$\subset$	$\{x\}$	$\{x, \mathcal{DC}\}$	$\{x, \mathcal{DK}\}$	$\{x, \mathcal{DC}, \mathcal{DK}\}$
$\{x\}$	F	T	$\mathcal{DK}$	T
$\{x, \mathcal{DC}\}$	F	F	F	$\mathcal{DK}$
$\{x, \mathcal{DK}\}$	F	$\mathcal{DK}$	$\mathcal{DK}$	$\mathcal{DK}$
$\{x, \mathcal{DC}, \mathcal{DK}\}$	F	F	F	$\mathcal{DK}$

$\subseteq$	$\{x\}$	$\{x, \mathcal{DC}\}$	$\{x, \mathcal{DK}\}$	$\{x, \mathcal{DC}, \mathcal{DK}\}$
$\{x\}$	T	T	T	T
$\{x, \mathcal{DC}\}$	F	$\mathcal{DC}$	F	$\mathcal{DC}$
$\{x, \mathcal{DK}\}$	$\mathcal{DK}$	$\mathcal{DK}$	$\mathcal{DK}$	$\mathcal{DK}$
$\{x, \mathcal{DC}, \mathcal{DK}\}$	F	$\mathcal{DK}$	F	$\mathcal{DK}$