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.

Nested Relations

Example 1 (Jaeschke & Schek, 1982)

domain *authors* **strg**;
domain *Authors*(*authors*);
domain *title* **strg**;
domain *price* **strg**;
domain *descriptors* **strg**;
domain *Descriptors*(*descriptors*);
relation *Books*(*Authors, title, price, Descriptors*) <− {
`({("A1"),("A2")},"T1","P1",{("D1"),("D2")}),`
`({("A2")},"T2","P2",{("D1"),("D2")}),`
`({("A1")},"T3","P1",{("D1"),("D2"),("D3")})` };

Books (*Authors* (*authors*)	*title*	*price*	*Descriptors* (*descriptors*))
A1 A2	T1	P1	D1 D2
A2	T2	P2	D1 D2
A1	T3	P1	D1 D2 D3

Nested Relations

No new syntax for processing: just subsume the relational algebra into the domain algebra.

Find all titles whose authors contain `A2` *and whose descriptors contain* `D1` *and* `D2`*.*

relation *A2*(*authors*) <− {("A2")};
relation *D1D2*(*descriptors*) <− {("D1"),("D2")};
A2D1D2 <− [*title*] **where** *Authors* **sup** *A2* **and**
    Descriptors **sup** *D1D2* **in** *Books*;

Or
let *Auth1* **be** *Authors* **sup** *A2*;
let *Des12* **be** *Descriptors* **sup** *D1D2*;
A2D1D2 <− [*title*] **where** *Auth1* **and** *Des12*
    in *Books*;

Books (*Authors* (*authors*)	*title*	*price*	*Descriptors* (*descriptors*))	*Auth1*	*Des12*
A1 A2	T1	P1	D1 D2	**true**	**true**
A2	T2	P2	D1 D2	**true**	**true**
A1	T3	P1	D1 D2 D3	**false**	**true**

Find books with only one author.

let *count* **be red** + **of** 1;

Books							
(*Authors*		*title*	*price*	*Descriptors*		)	*count*
(*authors*)	*count*			(*descriptors*)	*count*		
A1	2			D1	2		
A2	2	T1	P1	D2	2		3
				D1	2		
A2	1	T2	P2	D2	2		3
				D1	3		
A1	1	T3	P1	D2	3		3
				D3	3		

How do we select tuples of *Books* for *count*=1 in *Authors*?

where ([] **where** *count*=1 **in** *Authors*) **in** *Books*

Nested Relations

Example 2 (Deshpande & Larson, 1987)

Employee

(*ENO*	*name*	*Children* (*name*	*date*	*sex*)	*Training* (*CNO*	*date*))
105	John	Jane	800510	F	314	791010
		Eric	821005	M	606	810505
					714	820620
123	Anne	Maria	751112	F	315	810613
					423	820711
153	Bruce				314	791010
205	Ian	Bob	701016	M	314	791010
		Steve	750115	M		

Find employees without children

NoKids <− [*ENO*] **where not** ([ ] **in** *Children*) **in** *Employee*;

Find names of children of employees who took course 314

let C **be** [*name*] **in** *Children*;

CN314 <− [C] **where** ([ ] **where** *CNO*=`"314"` **in** *Training*) **in** *Employee*;

N.B. nested: *CN314*(C(*name*))

Employee								*CN314*
(*ENO*	*name*	*Children*			*Training*	)	C	(C)
		(*name*	*date*	*sex*)	(*CNO*	*date*)	(*name*)	(*name*)
105	John	Jane	800510	F	314	791010	Jane	Jane
		Eric	821005	M	606	810505	Eric	Eric
					714	820620		
123	Anne	Maria	751112	F	315	810613	Maria	
					423	820711		
153	Bruce				314	791010	{}	{}
205	Ian	Bob	701016	M	314	791010	Bob	Bob
		Steve	750115	M			Steve	Steve

If we want to make this a flat relation, we must remove the name C.

We can make C *anonymous* by not naming it (with the **let** statement).

We interpret this as raising the level. But to raise the level, there must be only one value: a *singleton*.

We can do both with a **red**uction operation:
red ujoin of C

N314 <− [**red ujoin of** C] **where**
([ ] **where** *CNO*="314" **in** *Training*)
in *Employee*;

Back to counting authors.

let *count* **be** [**red** + **of** 1] **in** *Authors*;

Books					
(*Authors* (*authors*)	*title*	*price*	*Descriptors* (*descriptors*)	)	*count*
A1 A2	T1	P1	D1 D2		2
A2	T2	P2	D1 D2		1
A1	T3	P1	D1 D2 D3		1

where *count*=1 **in** *Books*

Rules

1. Relational algebra subsumed by domain algebra.

2. The tuple containing a relation is the database for that relation.[1]

3. Lift levels by using anonymity,

4. but lifted relations must be singletons.

5. **relation**() is the only new syntax needed: domain algebra operator.

[1] A tuple is a mapping from names to values.
A scope "
A file directory "
A database "

Nested Relations — Deeper Nesting

Faculty								
(*dept*	*Profs*							)
	(*name*	*sal*	*office*	*Students*				)
				(*name*	*office*	*Courses*		)
						(*numb*	*grade*)	
CS	Pat	55	312	Kim	300	612	A	
						617	A-	
				James	300	612	B+	
						617	A-	
	Sue	42	322	Chris	328	506	A	
	Luc	48	319	Kim	300	612	A	
						617	A-	
EE	Pat	55	312	Kim	300	612	A	
						617	A-	
				James	300	612	B+	
						617	A-	
	Denis	52	412	Joe	425	530	B+	
				Peter	426	531	A-	

Find depts with two `A` *grades.*

Deeper Nesting—Queries

Find depts with (at least) two A *grades.*

let *CA* **be** [**red** + **of if** *grade*=A **then** 1 **else** 0] **in** *Courses*;

let *SA* **be** [**red** + **of** *CA*] **in** *Students*;

let *PA* **be** [**red** + **of** *SA*] **in** *Profs*;

[*dept*] **where** *PA*≥2 **in** *Faculty*

Faculty (*dept*	*Profs* (*name*	*Students* (*name*	*Courses* (*numb*	*grade*)	) *CA*	) *SA*	) *PA*
CS	Pat	Kim	612	A	1	1	3
			617	A-			
		James	612	B+	0		
			617	A-			
	Sue	Chris	506	A	1	1	
	Luc	Kim	612	A	1	1	
			617	A-			
EE	Pat	Kim	612	A	1	1	1
			617	A-			
		James	612	B+	0		
			617	A-			
	Denis	Joe	530	B+	0	0	
		Peter	531	A-			

Another way gets a different (better?) answer:

let *SA* **be** [**red ujoin of** (**relation**(*name*) **ijoin** *Course*)] **in** *Students*;

let *PA* **be** [**red ujoin of** *SA*] **in** *Profs*;

let *count* **be** [**red** + **of if** *grade*="A" **then** 1 **else** 0] **in** *PA*;

[*dept*] **where** *count*=2 **in** *Faculty*

Faculty (*dept*)	*PA* (*name*	*numb*	*grade*)	*count*
CS	Kim	612	A	2
	Kim	617	A-	
	James	612	B+	
	James	617	A-	
	Chris	506	A	
EE	Kim	612	A	1
	Kim	617	A-	
	James	612	B+	
	James	617	A-	
	Joe	530	B+	
	Peter	531	A-	

NB **relation**() is (the only) new syntax: it creates a singleton relation out of the attributes in the (). Also useful for nesting a flat relation.

Recursive nesting

domain *Id* **strg**;
domain *Cost* **intg**;
relation *Assembly*(*Id*, *Assembly*, *Cost*) <− {

```
    ("A", {("A1", {("A11", dc, 6),
                    ("A12", dc, 5)}, 3),
           ("A2", dc,              2)}, 4)
```

};

let *CalCost* **be** *Cost* + [**red** + **of** *CalCost*] **in** *Assembly*;
TotalCost <− [**red** + **of** *CalCost*] **in** *Assembly*;
Or, more explicitly,
let *CalCost* **be** **if** *Assembly*=$\mathcal{DC}$ **then** *Cost*
else *Cost* + *Assembly.SumCost*;
let *SumCost* **be** **red** + **of** *CalCost*;
TotalCost <− *Assembly.SumCost*;
where *Assembly.SumCost* ≡
[**red** + **of** *CalCost*] **in** *Assembly*

Assembly												
(*Id*	*Assembly*									*C*)	*CC*	*SC*
	(*Id*	*Assembly*					*C*)	*CC*	*SC*			
		(*Id*	*Assy*	*C*)	*CC*	*SC*						
A	A1	A11	dc	6	6	11	3	14	16	4	20	20
		A12	dc	5	5	11						
	A2			dc			2	2	16			