

Extending the Relational Model

Complex Values and Nested Relations

Guillaume Raschia — Nantes Université

Last update: October 12, 2025

1

4NF

A Very First Example

Class Book

title
set of authors
publisher
set of keywords

- Easy to model in any programming language
- Tricky in relational database!

2

Basic proposal

- Either we ignore the normalization...

Title	Author	Publisher	Keyword
FoD	S. Abiteboul	Addison-Wesley	Database
FoD	R. Hull	Addison-Wesley	Database
FoD	V. Vianu	Addison-Wesley	Database
FoD	S. Abiteboul	Addison-Wesley	Logic
FoD	R. Hull	Addison-Wesley	Logic
FoD	V. Vianu	Addison-Wesley	Logic
TCB	J.D. Ullman	Pearson	Database
⋮	⋮	⋮	⋮

- Key: (Title, Author, Keyword)
- Not in 2NF, given Title \rightarrow Publisher

3

Intermediate State

- ...Or we go up to 3NF, BCNF

Title	Publisher
FoD	Addison-Wesley
TCB	Pearson
⋮	⋮

Title	Author	Keyword
FoD	S. Abiteboul	Database
FoD	R. Hull	Database
FoD	V. Vianu	Database
FoD	S. Abiteboul	Logic
FoD	R. Hull	Logic
FoD	V. Vianu	Logic
TCB	J.D. Ullman	Database
⋮	⋮	⋮

- But we still ignore the multivalued dependencies...

4

About MVD's and 4NF

- MVD: full constraint on relation¹

Definition (Multi-Valued Dependency)

Let R be a relation of schema $\{X, Y, Z\}$; $X \twoheadrightarrow Y$ holds whenever (x, y, z) and (x, t, u) both belong to R , it implies that (x, y, u) and (x, t, z) should also be in R

Example:

- Department {Building} {Employee {Telephone}}
- MVD's = {Department \twoheadrightarrow Building; Department, Employee \twoheadrightarrow Telephone}

¹All the attributes are necessarily involved.

5

About MVD's and 4NF (cont'd)

MVD Properties in R

- $X \twoheadrightarrow R - X$ and $XY \twoheadrightarrow X$ always hold (trivial MVD)
- Complementation: $X \twoheadrightarrow Y \Leftrightarrow X \twoheadrightarrow R - X - Y$
- Augmentation: $X \twoheadrightarrow Y \Leftrightarrow XZT \twoheadrightarrow YZ$
- Transitivity: $X \twoheadrightarrow Y$ and $Y \twoheadrightarrow Z \Leftrightarrow X \twoheadrightarrow Z - Y$
- Replication: $X \twoheadrightarrow Y \Leftrightarrow X \twoheadrightarrow Y$

Definition (4NF)

For every non trivial MVD $X \twoheadrightarrow Y$ in R , then X is a superkey

Lossless-join decomposition of $R(X, Y, Z)$

Decomposition (X, Y) and (X, Z) is **lossless-join** iff $X \twoheadrightarrow Y$ holds in R

6

About MVD's and 4NF (cont'd)

Follow-on from the Department Example:

Department {Building} {Employee {Telephone}}

Known MVD's are $\{D \twoheadrightarrow B; DE \twoheadrightarrow T\}$

Derived MVD's

- $(D \twoheadrightarrow B) \Leftrightarrow (D \twoheadrightarrow ET)$, denoted $D \twoheadrightarrow B \mid ET$
- $(DE \twoheadrightarrow T) \Leftrightarrow (DE \twoheadrightarrow B)$
- Every trivial MVD holds, like $DE \twoheadrightarrow D$ or $DE \twoheadrightarrow BT$

7

Back to the Class Book Introductory Example

Title	Publisher
FoD	Addison-Wesley
TCB	Pearson
⋮	⋮

Title	Author	Keyword
FoD	S. Abiteboul	Database
FoD	R. Hull	Database
FoD	V. Vianu	Database
FoD	S. Abiteboul	Logic
FoD	R. Hull	Logic
FoD	V. Vianu	Logic
TCB	J.D. Ullman	Database
⋮	⋮	⋮

Non trivial MVD's:

Title → Author | Keyword

8

The Ultimate Schema

- ...Or we go to 4NF

Title	Publisher
FoD	Addison-Wesley
TCB	Pearson
⋮	⋮

Title	Author
FoD	S. Abiteboul
FoD	R. Hull
FoD	V. Vianu
TCB	J.D. Ullman
⋮	⋮

Title	Keyword
FoD	Database
FoD	Logic
TCB	Database
⋮	⋮

9

Pros & Cons

- 4NF design
 - requires many joins in queries (performance pitfall)
 - and loses the big picture of class book entities (many tuples in different tables)
- 1NF relational view
 - eliminates the need for users/apps to perform deadly joins
 - but loses the one-to-one mapping between tuples and objects (many tuples in one table)
 - has a large amount of redundancy
 - and could yield to insertion, deletion, update anomalies

10

Contents

NF²

Nested Tables: Model and Languages

Nested Queries: an SQL Extension

PNF Relations: Design, Properties and Algebra

11

NF²

Preamble

Alice: Complex values?
Riccardo: We could have used a different title: nested relations, complex objects, structured objects...
Vittorio: ...N1NF, ¬1NF, NFNF, NF2, NF², V-relation...I have seen all these names and others as well.
Sergio: In a nutshell, relations are nested within relations; something like Matriochka relations.
Alice: Oh, yes. I love Matriochkas.

FoD: chap. 20, p. 508

12

Beyond the Relational Model

- Theoretical extensions of the Relational Model (RM)
 - NF²
 - Nested Relations
- New Requirements
 - Operations as extension to relational algebra
 - Normal form to provide consistency
- Today, part of SQL3 and commercial systems

13

The NF² Database Model

NF² = NFNF = Non First Normal Form

Principle

NF² relations permit **complex values** whenever we encounter atomic, i.e. indivisible, values

- Breaks first normal form
- Allows more intuitive—let say *conceptual*—modeling for applications with complex data
- Preserves mathematical foundations of the Relational Model

14

From Pointland...

Type—aka. sort—of a relation in 1NF

$$\tau := \langle A_1:\text{dom}, \dots, A_k:\text{dom} \rangle$$

- A schema $R:\tau$ is a relation name R with $\text{sort}(R) = \tau$
- A relation is a **set** of τ -tuples
- Sort constructors: **tuple** $\langle \cdot \rangle$ and **finite-set** $\{ \cdot \}$
- Construction pattern of a relation: **set(tuple(dom*))**

15

...to Lineland²

In N1NF: much more combinations

$$\tau := \text{dom} \mid \langle A_1:\tau, \dots, A_k:\tau \rangle \mid \{ \tau \}$$

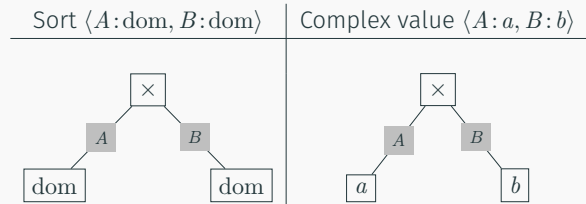
Examples

Sort τ	Complex value
dom	a
$\{\text{dom}\}$	$\{a, b, c\}$
$\{\{\text{dom}\}\}$	$\{\{a, b\}, \{a\}, \{\}\}$
$\langle A:\text{dom}, B:\text{dom} \rangle$	$\langle A:a, B:b \rangle$
$\{\langle A:\text{dom}, B:\text{dom} \rangle\}$	$\{\langle A:a, B:b \rangle, \langle A:a, B:b \rangle\}$
$\langle A:\{\langle B:\text{dom} \rangle\} \rangle$	$\langle A:\{\langle B:b \rangle, \langle B:c \rangle\} \rangle$

²Flatland, a Romance of Many Dimensions. Edwin A. Abbott (1884).

16

Sorts and Complex Values as Finite Trees

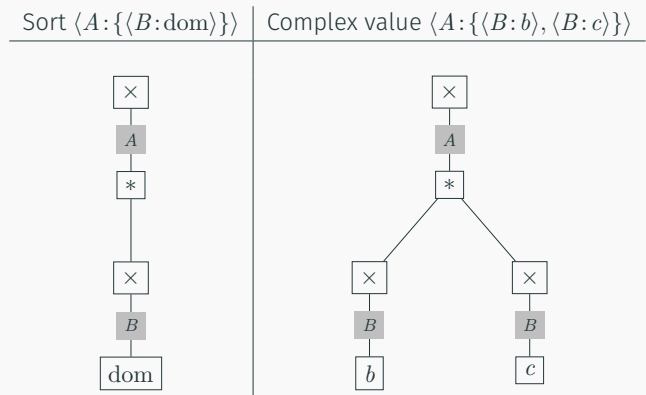


Gentle Reminder

A relation is a—finite—set of complex values

17

Sorts and Complex Values τ as Finite Trees (cont'd)



18

Nested Tables

A Popular Restriction

Definition (Nested relation)

A nested relation is a NF² relation where **set** and **tuple** constructors are required to **alternate**

The outermost constructor must be a tuple, as for the 1NF sort

Examples

$$\tau_1 = \langle A, B, C : \{ \langle D, E : \{ \langle F, G \rangle \} \} \rangle \quad \text{OK}$$

$$\tau_2 = \langle A, B, C : \{ \langle E : \{ \langle F, G \rangle \} \} \rangle \quad \text{OK}$$

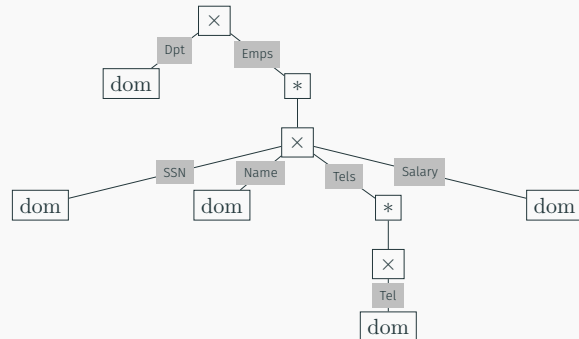
$$\tau_3 = \langle A, B, C : \langle D, E : \{ \langle F, G \rangle \} \rangle \quad \text{No!}$$

$$\tau_4 = \langle A, B, C : \{ \{ \langle F, G \rangle \} \} \rangle \quad \text{No!}$$

19

One Real-Life Example to Take Away

Departments: $\langle \text{Dpt}, \text{Emps} : \{ \langle \text{SSN}, \text{Name}, \text{Tels} : \{ \langle \text{Tel} \rangle \}, \text{Salary} \} \rangle \rangle$



Type constructors alternate on every path from the root to the leaves

20

Instance of a Nested-Departments Table

Department	Employees			
	SSN	Name	Telephones	Salary
Computer Science	4711	Todd	Tel	6,000
			038203-12230 0381-498-3401	
	5588	Whitman	Tel	6,000
			0391-334677 0391-5592-3452	
7754	Miller	Tel	550	
8832	Kowalski	Tel	2,800	
Mathematics	6834	Wheat	Tel	750
			0345-56923	

21

About Nested Relations

Nested relations vs. N1NF-relations

Cosmetic restriction only!

Size of nested relations

$\mathcal{O}(2^{2^{\dots 2^n}})$ with n being the size of the active domain of R and “the tower of 2” equals the depth of R (#nested levels)

Reminder: the size of a flat relation is polynomial

22

Languages for Nested Relations

Logic

Mainly extend the Relational Calculus to **variables denoting sets**

$$\{t.\text{Dpt} \mid t \in \text{Dpts} \wedge \forall X, u : (t.\text{Emps} = X \wedge u \in X \rightarrow u.\text{Salary} \leq 5,000)\}$$

Flavor with queries as terms:

$$\{t.\text{Dpt} \mid t \in \text{Dpts} \wedge t.\text{Emps} \subseteq \{u \mid u \in \text{Dpts}.\text{Emps} \wedge u.\text{Salary} \leq 5,000\}\}$$

23

Operations on Nested Relations

$R(A, B(C, D))$ and $S(A(C, D), B(C, D), E)$ and $T(A, B(C, D))$

The usual way

$$\sigma_{A=a}(R) \quad \text{and} \quad \pi_A(R)$$
$$R \bowtie_{R.A=S.E} S$$
$$R - T \quad \text{and} \quad R \cup T \quad (\text{on union-compliant relations})$$

Straightforward—recursive—extensions

$$\begin{array}{lll} \sigma_{A(C,D)=B(C,D)}(S) & \sigma_{A(C,D) \subset B(C,D)}(S) & \sigma_{A \in B}(R) \\ \pi_B(R) & S \bowtie_{S.B \subseteq T.B} T & \end{array}$$

24

Nested Relational Algebra

Selection-Projection-Join-Union-Negation

- $\cup - \pi$ nearly as in relational algebra
- σ and \bowtie : condition extended to support
 - **Relations as operands** (instead of constants in dom)
 - **Set operations** like $\theta \in \{\in, \subseteq, \subset, \supseteq, \supset\}$
- **Recursively structured** operation parameters, e.g.
 - π : nested projection attribute lists
 - σ and \bowtie : predicates on nested relations

25

Nested Relational Algebra (cont'd)

First real-world implementation: DREMEL (2010) by Google

Sergey Melnik et al. 2020. *Dremel: a decade of interactive SQL analysis at web scale*. Proc. VLDB Endow. 13, 12 (August 2020), 3461–3472.

A language of the NoSQL era, built upon the *Protocol Buffer* – Protobuf – format

“Hierarchical schemas were a big departure from typical SQL schema design. Textbook normal forms would use many tables, and query-time joins. Avoiding joins was a key enabler for Dremel’s scalable and fast execution. (Dremel initially had no join support, and was successful for years with only limited join support.) Denormalizing related data into one nested record was common in Google’s datasets; hierarchical schemas made it unnecessary to flatten or duplicate any data, which would have increased storage and processing cost.”

26

Nested Relational Algebra (cont'd)

Additional operations: Nest (ν) and Unnest (μ)

- $\nu_{A=(A_1, A_2, \dots, A_n)}(R)$: create column A as a nesting from A_1, A_2, \dots, A_n of R
- $\mu_{A(A_1, A_2, \dots, A_n)}(R)$: remove 1 level of nesting from the A column of R and then, promote nested columns (A_1, A_2, \dots, A_n) as regular outermost columns

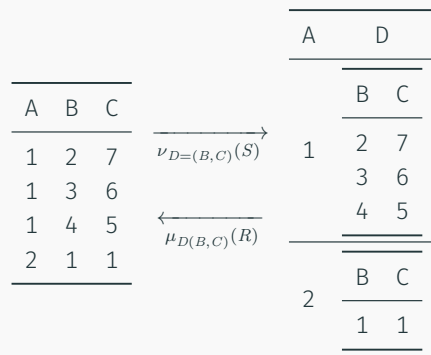
A curiosity: The Powerset operator

$$\Omega(\mathbf{I}(R)) = \{\vartheta \mid \vartheta \subseteq \mathbf{I}(R)\}$$

Powerset Ω extends algebra up to **reachability** (eq. Datalog)

27

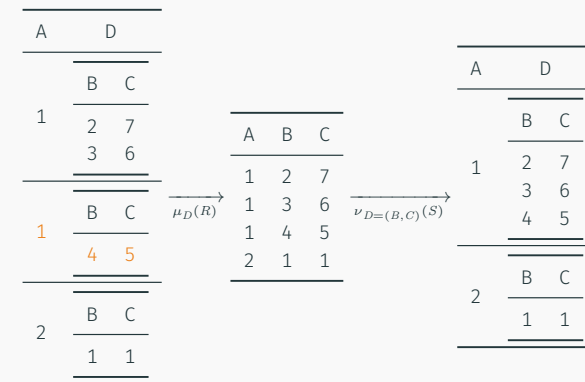
Nest & Unnest



28

About the Duality of Nest & Unnest

Unnesting is not generally reversible!



29

To Sum Up

- Unnest is the **right inverse** of nest: $\mu_{A(\alpha)} \circ \nu_{A=\alpha} \equiv \text{Id}$
- Unnest is **not information preserving** (one-to-one) and so has no right inverse

30

Nested Queries

Nesting in Queries

Flat-Flat Theorem

Let Q be a nested relational algebra expression;

- Q takes a non-nested relation as input
- Q produces a non-nested relation as output

Then, Q can be rewritten as a **regular relational algebra expression** (i.e., w/o nesting)

31

Nesting in Queries (cont'd)

Result is actually stronger for query Q

Nested Query Theorem

Assume a d_1 -nested relation as input and a d_2 -nested relation as output; there is no need for intermediate results having depth greater than $\max(d_1, d_2)$

What for?

- Can be used by query optimizers
- No need to introduce intermediate nesting
- Standard techniques for query evaluation

32

NF² Concepts in SQL3

- SQL-99 introduced **tuple type constructor** ROW
- Only few changes to type system in SQL:2003
 - **Bag type constructor** MULTISET
 - XML data types
- Implementations in commercial DBMS most often do NOT comply with standard!

33

ROW Type Constructor

- ROW implements **tuple** type constructor

Example

```
CREATE ROW TYPE AddressType (  
    Street VARCHAR(30),  
    City   VARCHAR(30),  
    Zip    VARCHAR(10) );  
  
CREATE ROW TYPE CustomerType (  
    Name    VARCHAR(40),  
    Address AddressType );  
  
CREATE TABLE Customer OF TYPE CustomerType  
    ( PRIMARY KEY Name );
```

34

ROW Type Constructor (cont'd)

- Insertion of records requires call to **ROW constructor**

```
INSERT INTO Customer  
VALUES('Doe', ROW('50 Otages', 'Nantes', '44000'));
```

- Component access by usual dot "." notation with field parenthesis (≠ table prefix)

```
SELECT C.Name, (C.Address).City FROM Customer C;
```

35

MULTISET Type Constructor

- SQL:2003 **MULTISET** implements set/bag type constructor
- Can be combined with the **ROW** constructor
- Allows creation of nested tables (NF²)

```
CREATE TABLE Department (  
    Name VARCHAR(40),  
    Buildings INTEGER MULTISET,  
    Employees ROW( Firstname VARCHAR(30),  
                    Lastname  VARCHAR(30),  
                    Office    INTEGER ) MULTISET );
```

36

MULTISET Type Constructor (cont'd)

Operations

- MULTISET constructor
- UNNEST implements μ
- COLLECT: special aggregate function to implement ν
- FUSION: special aggregate function to build union of aggregated multisets
- MULTISET UNION|INTERSECT|EXCEPT
- CARDINALITY for size
- SET eliminates duplicates
- ELEMENT converts singleton to a tuple (row) expression

37

MULTISET Type Constructor (cont'd)

Predicates

- MEMBER: $x \in E$
- SUBMULTISET multiset containment: $S \subseteq E$
- IS [NOT] A SET test whether there are duplicates or not

```
SELECT D.Name FROM Department D
WHERE CARDINALITY(D.Buildings) >= 2 AND
      D.Employees IS A SET;
```

38

MULTISET Type Constructor (cont'd)

Insert and Update statements

```
INSERT INTO Department
VALUES( 'Computer Science',
       MULTISET[29,30],
       MULTISET( ROW( ... ) ) );
```

```
INSERT INTO Department
VALUES( 'Physics',
       MULTISET[28],
       MULTISET( SELECT ... FROM ... );
```

```
UPDATE Department
SET Buildings=Buildings MULTISET UNION MULTISET[17]
WHERE Name='Computer Science';
```

39

MULTISET Type Constructor (cont'd)

- Unnesting of a multiset

```
SELECT D.Name, Emp.LastName
FROM Department D,
      UNNEST( D.Employees ) Emp;
```

- Nesting using the COLLECT aggregation function

```
SELECT C.Title,
       COLLECT( C.Keyword ) AS Keywords,
       COLLECT( C.Author ) AS Authors
FROM Classbook C GROUP BY C.Title;
```

40

PNF Relations

Design of Flat Tables

Normal Forms that Matter

- 1NF
- 3NF
- BCNF (Boyce-Codd NF)
- 4NF

Other Normal Forms

- 2NF
- 5NF or PJNF (Project-Join NF)
- DKNF (Domain-Key NF)
- 6NF, for Temporal Data

41

PNF Nested Relations

An important subclass of nested relations

Principle

The Partitioned Normal Form (PNF) requires a **flat key** on every nesting level

PNF relation:

A	D
1	B C
	2 7
	3 6
	4 5
2	B C
	1 1

Non-PNF relation:

A	D
1	B C
	2 7
	3 6
1	B C
	4 5
2	B C
	1 1

42

Partitioned Normal Form

Definition (PNF)

Let $R(X, Y)$ be a n -ary relation where X is the set of atomic attributes and Y is the set of relation-valued attributes; R is in partitioned normal form (PNF) iff

1. $X \rightarrow X, Y$ (X is a super-key)
2. Recursively, $\forall r \in Y$ and $\forall \mathbf{I}(r) \in \pi_r(R)$, $\mathbf{I}(r)$ is in PNF

- If $X = \emptyset$, then $\emptyset \rightarrow Y$ must hold
- If $Y = \emptyset$, then $X \rightarrow X$ holds trivially

Thus a **1NF relation is in PNF**

43

Properties of PNF

1. A flat (1NF) relation is always in PNF
2. PNF relations are **closed** under unnesting
3. Nesting and unnesting operations **commute** for PNF relations
4. Size of PNF relations remains **polynomial!**

Strong theoretical results and many practical applications

44

PNF as an Alternative to 4NF

PNF relation R and the “equivalent” unnested relation S

A	E		F
	B	C	D
1	2	3	1
	4	2	1
<hr/>			
2	1	1	2
	4	1	3
<hr/>			
3	1	1	2

$$\xrightarrow{\mu_{E(BC)} \circ \mu_{F(D)}}$$

A	B	C	D
1	2	3	1
1	4	2	1
2	1	1	2
2	4	1	2
2	1	1	3
2	4	1	3
3	1	1	2

- $A \rightarrow BC \mid D$ holds in S : S should be split to reach 4NF
- PNF compactly mimics 4NF (A is a superkey in R)

45

PNF and MVD's and Scheme Tree

Preliminary statement

A **scheme tree** captures the logical structure of a nested relation schema and explicitly represents the **set of MVD's**

One more property of PNF relations

A nested relation R is in PNF iff the scheme of R follows a scheme tree with respect to the given set of MVD's

MVD's by example

- Book db: $\{ \text{Title} \rightarrow \text{Author} \mid \text{Keyword} \}$
- Class db: Student, Major, Class, Exam, Project
 $\{ S \rightarrow M, SC \rightarrow E, SC \rightarrow P \}$

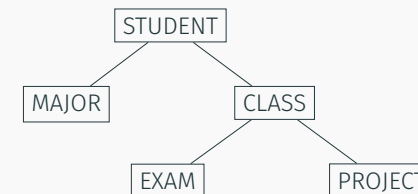
46

Scheme—or Schema—Tree

A tool for nested relation design

Definition (Scheme Tree)

A scheme tree is a tree containing at least one node and whose nodes are labelled with nonempty sets of attributes that form a disjoint partition of a set U of atomic attributes



47

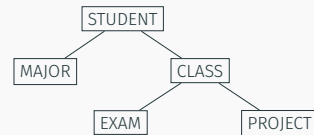
Design by MVD's

Pattern

Ancestors-and-self \rightarrow Child-and-descendants

Example (cont'd)

- STUDENT \rightarrow MAJOR
- STUDENT \rightarrow CLASS EXAM PROJECT
- STUDENT CLASS \rightarrow EXAM
- STUDENT CLASS \rightarrow PROJECT



48

Nested Relation Schema

Definition (NRS)

A nested relation scheme (NRS) for a scheme tree T , denoted by \mathcal{T} , is a sort defined recursively by:

1. If T is empty, i.e. T is defined over an empty set of attributes, then $\mathcal{T} = \emptyset$;
2. If T is a leaf node X , then $\mathcal{T} = \langle X \rangle$;
3. If A is the root of T and T_1, \dots, T_n , $n \geq 1$, are the principal subtrees of T then $\mathcal{T} = \langle A, B_1: \{ \mathcal{T}_1 \}, \dots, B_n: \{ \mathcal{T}_n \} \rangle$

Example (cont'd)

$\langle \text{STUDENT, Majors:} \{ \langle \text{MAJOR} \rangle \}, \text{Classes:} \{ \langle \text{CLASS, Exams:} \{ \langle \text{EXAM} \rangle \}, \text{Projects:} \{ \langle \text{PROJECT} \rangle \} \} \rangle \rangle$

49

The Initial Flat Class Table

STUDENT	MAJOR	CLASS	EXAM	PROJECT
Anna	Math	CS100	mid-year	Proj A
Anna	Math	CS100	mid-year	Proj B
Anna	Math	CS100	mid-year	Proj C
Anna	Math	CS100	final	Proj A
Anna	Math	CS100	final	Proj B
Anna	Math	CS100	final	Proj C
Anna	Computing	CS100	mid-year	Proj A
Anna	Computing	CS100	mid-year	Proj B
Anna	Computing	CS100	mid-year	Proj C
Anna	Computing	CS100	final	Proj A
Anna	Computing	CS100	final	Proj B
Anna	Computing	CS100	final	Proj C
Bill

Hint

NRS follows **serialization** of the schema tree:

(Student (Major) (Class (Exam) (Project)))

50

PNF from NRS From Schema Tree from MVD's!

STUDENT	Majors	Classes		
Anna		CLASS	Exams	Projects
	MAJOR		EXAM	PROJECT
	Math Computing	CS100	mid-year final	Proj A Proj B Proj C
Bill		CLASS	Exams	Projects
	MAJOR	P100	EXAM final	PROJECT Pract Test 1 Pract Test 2
	Physics Chemistry	CH200	EXAM test A test B test C	PROJECT Exp 1 Exp 2 Exp 3

51

Extended RA on PNF Relations (Roth et al., TODS 1988)

A	X		
	B	Y	
a ₁	b ₁	C	D
		c ₁	d ₁
a ₂	b ₂	c ₁	d ₁
		c ₂	d ₂
a ₃	b ₃	c ₂	d ₂
		c ₃	d ₃
a ₄	b ₄	c ₃	d ₃
		c ₁	d ₂

A	X		
	B	Y	
a ₁	b ₁	C	D
		c ₁	d ₁
a ₂	b ₃	c ₄	d ₄
		c ₂	d ₂
a ₄	b ₂	c ₁	d ₂
		c ₁	d ₃

A	X		
	B	Y	
a ₁	b ₁	c ₁	d ₁
		c ₁	d ₂
a ₂	b ₂	c ₁	d ₁
		c ₂	d ₂
a ₃	b ₃	c ₄	d ₄
		c ₂	d ₂
a ₄	b ₄	c ₁	d ₂
		c ₁	d ₃

A	X		
	B	Y	
a ₁	b ₁	C	D
		c ₁	d ₁
a ₂	b ₂	c ₁	d ₁
		c ₂	d ₂
a ₃	b ₃	c ₂	d ₂
		c ₃	d ₃
a ₄	b ₄	c ₁	d ₂
		c ₁	d ₃

A	X		
	B	Y	
a ₁	b ₁	C	D
		c ₁	d ₂
a ₂	b ₂	c ₁	d ₁
		c ₂	d ₂
a ₃	b ₃	c ₂	d ₂
		c ₃	d ₃
a ₄	b ₄	c ₁	d ₂
		c ₁	d ₃

52

Extended RA on PNF Relations (cont'd)

A	B	X	
		C	D
a ₁	b ₁	c ₁	d ₁
		c ₂	d ₂
a ₂	b ₁	c ₁	d ₃
		c ₂	d ₁
a ₂	b ₂	c ₃	d ₂
		c ₁	d ₁

A	E	B	X	
			C	D
a ₁	e ₁	b ₁	c ₁	d ₁
			c ₁	d ₃
a ₂	e ₁	b ₁	c ₁	d ₁
			c ₂	d ₁
a ₂	e ₄	b ₁	c ₃	d ₁
			c ₃	d ₂
a ₂	e ₃	b ₂	c ₃	d ₂
			c ₁	d ₁

A	X	
	C	D
a ₁	c ₁	d ₁
	c ₁	d ₃
a ₂	c ₁	d ₁
	c ₃	d ₁
a ₂	c ₃	d ₂
	c ₁	d ₁

E	B	X	
		C	D
e ₁	b ₁	c ₁	d ₁
		c ₁	d ₃
e ₄	b ₁	c ₃	d ₁
		c ₃	d ₂
e ₃	b ₂	c ₃	d ₂
		c ₁	d ₁

Mark A. Roth, Herry F. Korth, and Abraham Silberschatz. 1988. Extended algebra and calculus for nested relational databases. ACM Trans. Database Syst. 13, 4 (Dec. 1988), 389-417. <https://doi.org/10.1145/49346.49347>

53