Extending the Relational Model

Complex Values and Nested Relations

Guillaume Raschia — Nantes Université

Last update: October 17, 2023

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!

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
$\vdots$	$\vdots$	$\vdots$	$\vdots$

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

Intermediate State

- …Or we go to 3NF, BCNF

Title	Publisher
FoD	Addison-Wesley

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

- But we still ignore the multivalued dependencies…

About MVD's and 4NF

- MVD: full constraint on relation[1]

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}

[1] All the attributes are necessarily involved.

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

MVD Properties in $R(X, Y, Z)$

- $X \twoheadrightarrow Y \Rightarrow X \twoheadrightarrow Z$
- $X \rightarrow Y \Rightarrow X \twoheadrightarrow Y$
- $X \twoheadrightarrow R - X$ always holds (trivial MVD)

Definition (4NF)

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

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

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

Follow-on from the Department Example:

Department {Building} {Employee {Telephone}}

- $(D \twoheadrightarrow B) \quad \Rightarrow \quad (D \twoheadrightarrow E, T)$
- $(D, E \twoheadrightarrow T) \quad \Rightarrow \quad (D, E \twoheadrightarrow B)$
- Every trivial MVD holds, like $D, E \twoheadrightarrow B, T$

Back to the Class Book Introductory Example

Title	Publisher
FoD	Addison-Wesley

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

List of—non trivial—MVD's:

- Title $\twoheadrightarrow$ Author
- Title $\twoheadrightarrow$ Keyword

The Ultimate Schema

- ...Or we go to 4NF

Title	Publisher
FoD	Addison-Wesley

Title	Author
FoD	S. Abiteboul
FoD	R. Hull
FoD	V. Vianu

Title	Keyword
FoD	Database
FoD	Logic

Pros & Cons

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

Contents

NF^2

Preamble

Alice:	Complex values?
Riccardo:	We could have used a different title: nested relations, complex objects, structured objects...
Vittorio:	...N1NF, $\neg$1NF, NFNF, NF2, NF^2, 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

Beyond the Relational Model

- Theoretical extensions of the Relational Model (RM)
  - NF^2
  - Nested Relations
- New Requirements
  - Operations as extension to relational algebra
  - Normal form to provide consistency
- Today, part of SQL3 and commercial systems

The NF^2 Database Model

NF^2 = NFNF = Non First Normal Form

Principle

NF^2 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

From Pointland...

Type—aka. sort—of a relation in 1NF

$$\tau := \langle A_1 : \mathrm{dom}, \ldots, A_k : \mathrm{dom} \rangle$$

- A schema $R : \tau$ is a relation name R with $\mathrm{sort}(R) = \tau$
- 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*))

In N1NF: much more combinations

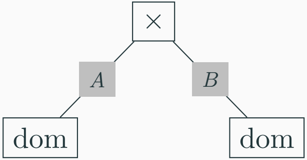
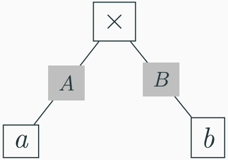
$$\tau := \mathrm{dom} \mid \langle A_1 : \tau, \ldots, A_k : \tau \rangle \mid \{\tau\}$$

Examples

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

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

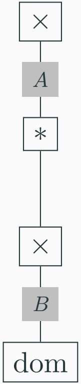
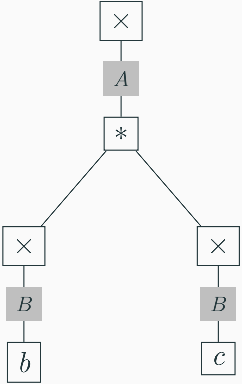
Sorts and Complex Values as Finite Trees

Sort $\langle A\!:\!\mathrm{dom}, B\!:\!\mathrm{dom}\rangle$	Complex value $\langle A\!:\!a, B\!:\!b\rangle$
	

Gentle Reminder

A relation is a—finite—**set** of complex values

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

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

Nested Tables

A Popular Restriction

Definition (Nested relation)

A nested relation is a NF^2 relation where **set** and **tuple** constructors are required to alternate

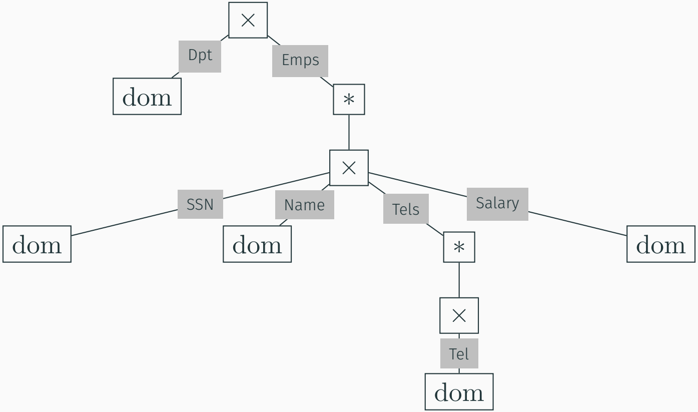
The outermost constructor must be a tuple, as for the 1NF sort

Examples

$$
\begin{aligned}
\tau_1 &= \langle A, B, C : \{\langle D, E : \{\langle F, G\rangle\}\rangle\}\rangle && \text{Ok} \\
\tau_2 &= \langle A, B, C : \{\langle E : \{\langle F, G\rangle\}\rangle\}\rangle && \text{Ok} \\
\tau_3 &= \langle A, B, C : \langle D, E : \{\langle F, G\rangle\}\rangle\rangle && \text{No!} \\
\tau_4 &= \langle A, B, C : \{\{\langle F, G\rangle\}\}\rangle && \text{No!}
\end{aligned}
$$

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

Instance of a – Nested – Departments Table

<table>
<tr><th>Department</th><th colspan="4">Employees</th></tr>
<tr><td rowspan="5">Computer Science</td><td>SSN</td><td>Name</td><td>Telephones</td><td>Salary</td></tr>
<tr><td>4711</td><td>Todd</td><td>Tel
038203-12230
0381-498-3401</td><td>6,000</td></tr>
<tr><td>5588</td><td>Whitman</td><td>Tel
0391-334677
0391-5592-3452</td><td>6,000</td></tr>
<tr><td>7754</td><td>Miller</td><td>Tel</td><td>550</td></tr>
<tr><td>8832</td><td>Kowalski</td><td>Tel</td><td>2,800</td></tr>
<tr><td rowspan="2">Mathematics</td><td>SSN</td><td>Name</td><td>Telephones</td><td>Salary</td></tr>
<tr><td>6834</td><td>Wheat</td><td>Tel
0345-56923</td><td>750</td></tr>
</table>

About Nested Relations

Nested relations vs. N1NF-relations

Cosmetic restriction only!

Size of nested relations

$\mathcal{O}(2^{2^{\cdots^{2^n}}})$ 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

Logic

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

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

Flavor with queries as terms:

$$\{t.\mathsf{Dpt} \mid \mathsf{Dpts}(t) \ \wedge \ t.\mathsf{Emps} \subseteq \{u \mid u.\mathsf{Salary} \leq 5,000\}\}$$

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

$$\sigma_{A(C,D)=B(C,D)}(S) \quad \sigma_{A(C,D)\subset B(C,D)}(S) \quad \sigma_{A\in B.C}(R)$$

$$\pi_{A,B.C}(R) \qquad R \bowtie_{R.B\subseteq S.A} S$$

Nested Relational Algebra

Selection-Projection-Join-Union-Negation

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

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

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

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

Nested Relational Algebra (cont'd)

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

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

A curiosity: The Powerset operator

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

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

Nest & Unnest

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

$\xrightarrow[\nu_{D=(B,C)}(S)]{}$

$\xleftarrow[\mu_{D(B,C)}(R)]{}$

<table>
<tr><th>A</th><th>D</th></tr>
<tr><td>1</td><td><table><tr><th>B</th><th>C</th></tr><tr><td>2</td><td>7</td></tr><tr><td>3</td><td>6</td></tr><tr><td>4</td><td>5</td></tr></table></td></tr>
<tr><td>2</td><td><table><tr><th>B</th><th>C</th></tr><tr><td>1</td><td>1</td></tr></table></td></tr>
</table>

Unnesting is not generally reversible!

<table>
<tr><th>A</th><th>D</th></tr>
<tr><td>1</td><td><table><tr><th>B</th><th>C</th></tr><tr><td>2</td><td>7</td></tr><tr><td>3</td><td>6</td></tr></table></td></tr>
<tr><td>1</td><td><table><tr><th>B</th><th>C</th></tr><tr><td>4</td><td>5</td></tr></table></td></tr>
<tr><td>2</td><td><table><tr><th>B</th><th>C</th></tr><tr><td>1</td><td>1</td></tr></table></td></tr>
</table>

$\xrightarrow{}$ $\mu_D(R)$

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

$\xrightarrow{}$ $\nu_{D=(B,C)}(S)$

<table>
<tr><th>A</th><th>D</th></tr>
<tr><td>1</td><td><table><tr><th>B</th><th>C</th></tr><tr><td>2</td><td>7</td></tr><tr><td>3</td><td>6</td></tr><tr><td>4</td><td>5</td></tr></table></td></tr>
<tr><td>2</td><td><table><tr><th>B</th><th>C</th></tr><tr><td>1</td><td>1</td></tr></table></td></tr>
</table>

To Sum Up

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

Nested 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)

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

NF^2 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!

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 AdressType );

CREATE TABLE Customer OF TYPE CustomerType
                             ( PRIMARY KEY Name );
```

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 ($\neq$ table prefix)

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

MULTISET Type Constructor

- SQL:2003 `MULTISET` implements set/bag type constructor
- Can be combined with the `ROW` constructor
- Allows creation of nested tables (NF^2)

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

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

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

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

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';
```

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;
```

Design

On Flat Tables

Normal Forms that Matter

- 1NF
- 3NF
- BCNF
- 4NF

Other Normal Forms

- 2NF
- 5NF
- DKNF
- 6NF
- ...

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:

<table>
<tr><th>A</th><th>D</th></tr>
<tr><td>1</td><td><table><tr><th>B</th><th>C</th></tr><tr><td>2</td><td>7</td></tr><tr><td>3</td><td>6</td></tr><tr><td>4</td><td>5</td></tr></table></td></tr>
<tr><td>2</td><td><table><tr><th>B</th><th>C</th></tr><tr><td>1</td><td>1</td></tr></table></td></tr>
</table>

Non-PNF relation:

<table>
<tr><th>A</th><th>D</th></tr>
<tr><td>1</td><td><table><tr><th>B</th><th>C</th></tr><tr><td>2</td><td>7</td></tr><tr><td>3</td><td>6</td></tr></table></td></tr>
<tr><td>1</td><td><table><tr><th>B</th><th>C</th></tr><tr><td>4</td><td>5</td></tr></table></td></tr>
<tr><td>2</td><td><table><tr><th>B</th><th>C</th></tr><tr><td>1</td><td>1</td></tr></table></td></tr>
</table>

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

- If $X = \emptyset$, then $\emptyset \longrightarrow Y$ must hold
- If $Y = \emptyset$, then $X \longrightarrow X$ holds trivially
  Thus a **1NF relation is in PNF**

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

PNF as an Alternative to 4NF

PNF relation R and the "equivalent" unnested relation S

<table>
<tr><th>A</th><th>E</th><th>F</th></tr>
<tr><td>1</td><td><table><tr><th>B</th><th>C</th></tr><tr><td>2</td><td>3</td></tr><tr><td>4</td><td>2</td></tr></table></td><td><table><tr><th>D</th></tr><tr><td>1</td></tr></table></td></tr>
<tr><td>2</td><td><table><tr><td>1</td><td>1</td></tr><tr><td>4</td><td>1</td></tr></table></td><td><table><tr><td>2</td></tr><tr><td>3</td></tr></table></td></tr>
<tr><td>3</td><td><table><tr><td>1</td><td>1</td></tr></table></td><td><table><tr><td>2</td></tr></table></td></tr>
</table>

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

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

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

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

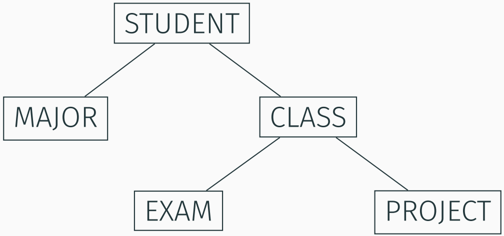
- Book db: $\{$Title $\twoheadrightarrow$ Author$\}$
- Class db: Student, Major, Class, Exam, Project
  $\{S \twoheadrightarrow M, SC \twoheadrightarrow E, SC \twoheadrightarrow P\}$

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



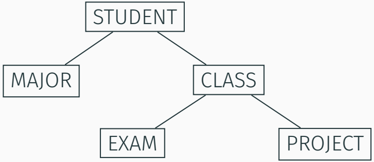
Design by MVD's

Pattern

Ancestors-and-self $\twoheadrightarrow$ Child-and-descendants

Example (cont'd)

- STUDENT $\twoheadrightarrow$ MAJOR
- STUDENT $\twoheadrightarrow$ CLASS EXAM PROJECT
- STUDENT CLASS $\twoheadrightarrow$ EXAM
- STUDENT CLASS $\twoheadrightarrow$ PROJECT



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 $T_1, \ldots, T_n$, $n \geq 1$, are the principal subtrees of T then $\mathcal{T} = \langle A, B_1 : \{\mathcal{T}_1\}, \ldots, B_n : \{\mathcal{T}_n\} \rangle$

Example (cont'd)

⟨STUDENT, Majors:{⟨MAJOR⟩}, Classes:{⟨CLASS, Exams:{⟨EXAM⟩}, Projects:{⟨PROJECT⟩} ⟩} ⟩

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)))

PNF from NRS From Schema Tree from MVD’s!

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