III) COMPLEX VALUE DATABASES

Introduction

- Relax the 1 Normal Form of the Relational Model
  - Set-value attributes (e.g., set of tuples \(\Rightarrow\) Relations)
- Alternative Names: N1NF, NF2, and Nested Relations Models
- Three simultaneous innovating approaches:
  - University of Texas 1983: N1NF (P. Fisher & S. Thomas)
- No programming languages
- Only experimental prototypes
A Model for Complex Values

- Set of Atomic values: \( D = \bigcup D_i \), where \( D_i \) are atomic domains such as strings, integers, reals, etc.
- Set of Attribute names \( A \)
- Set of Complex values:
  - each element \( v \) of \( D \) is a complex value
  - if \( v_1, v_2, ..., v_n \) are structured values and \( a_1, a_2, ..., a_n \) are attribute names then \( [a_1:v_1, a_2:v_2, ..., a_n:v_n] \) is a complex value, called tuple
  - if \( v_1, v_2, ..., v_n \) are structured values then \( \{v_1, v_2, ..., v_n\} \) is a complex value, called set

\( \Rightarrow \) **NOTE**: In N1NF Models the set and tuple value constructors can be applied only alternatively
  - is not possible to represent set of sets or tuples of tuples

Complex Values: Examples

- In the FineArt database we want to represent artists with their work
  - From a relational perspective we have the multi-valued dependency: \( \text{Name} \rightarrow \rightarrow \text{Artifacts} \)

<table>
<thead>
<tr>
<th>Name</th>
<th>Style</th>
<th>Artifacts Title</th>
<th>Live-Time</th>
<th>Nationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pablo Picasso</td>
<td>Cubism</td>
<td>Guernica Seated Woman</td>
<td>1881-1973</td>
<td>Spain</td>
</tr>
<tr>
<td>Claude Monet</td>
<td>Impressionism</td>
<td>Haystacks Wheatstacks</td>
<td>1840-1926</td>
<td>France</td>
</tr>
<tr>
<td>Kasimir Malevich</td>
<td>Geom. Abstract</td>
<td>Soldier of the first Division The Woodcutter</td>
<td>1878-1935</td>
<td>Russia</td>
</tr>
<tr>
<td>Alberto Giacometti</td>
<td>Surrealism</td>
<td></td>
<td>1901-1966</td>
<td>Swiss</td>
</tr>
<tr>
<td>Paul (Eugène-Henri) Gauguin</td>
<td>Impressionism</td>
<td>The YellowChrist Tahitian Women</td>
<td>1848-1903</td>
<td>France</td>
</tr>
<tr>
<td>Edouard Manet</td>
<td>Impressionism</td>
<td>The Waitress Em. Zola Portrait</td>
<td>1832-1883</td>
<td>France</td>
</tr>
</tbody>
</table>
Complex Value Types

- Set of Atomic types: STRING, INT, REAL
- Set of Attribute names \( A \)
- Set of types for Complex values:
  - each atomic type is a type
  - if \( t_1, t_2, \ldots, t_n \) are types and \( a_1, a_2, \ldots, a_n \) are attribute names then \([a_1:t_1, a_2:t_2, \ldots, a_n:t_n]\) is a tuple type
  - if \( t \) is a type then \( \{t\} \) is a set type
- EXAMPLE: The type (sort) of the FINE-ART N1NF Relation is:
  \[
  \{[\text{Name}: \text{STRING}, \text{Style}: \text{STRING}, \text{Artifacts}: \{[\text{Title}: \text{STRING}]}, \text{Live-Time}: \text{STRING}, \text{Nationality}: \text{STRING}]\}
  \]

Typing: DB vs. PL view

- Types in PL:
  - Part of the program
  - Focus on variables and functions
  - Used to avoid inconsistent computations
  - Type checking is mainly a static activity (compiler)
- Types in DB:
  - Part of the schema
  - Focus on data
  - Used to check semantic integrity
  - Type checking could be also a dynamic activity (run time)
- NOTE: The conception of good constructs in DBPL is twofold:
  - Support a type system for data modeling
  - Support a data model for program typing
A schema $S$ is a triple $(N, T, \sigma)$:

- $N$ is a set of names (e.g., relation names)
- $T$ is a set of types
- $\sigma$ is a function from $N$ to $T$

The extension (interpretation) $I$ of a name $n$ is the set of values (instances) of sort $\sigma(n)$.

A database instance is a mapping of the schema $S$ names $n$ such that for each $n$ in $S$, $I(n)$ is a set of values of sort $\sigma(n)$.

Complex value Query Languages: Algebra

- Uses relation names + constants + new operations
  - Basic Set Operations: Union, Intersection and Difference
  - Constructive Operations: PowerSet, Tuple_create, Set_create
  - Destructive Operations: Flatten, Tuple_destroy
  - Restructuring Operations: Nest and Unnest
  - Filtering Operations: Selection, Projection, Renaming
  - Additional Operations: Cross Product, Join
Complex Value Algebraic Operators

- PowerSet
  - $B \subseteq b_1, b_2$ (B:T)
  - $B \subseteq b_1, b_2$ (B:T)
  - $B \subseteq b_1, b_2$ (B:T)

- Flatten
  - $B \subseteq b_1, b_2$ (B:T)

- Join
  - $A \times B$ (A:T1, B:{C:T2, D:T3})
  - $E \times F$ (E:T1, F:T2)
  - $A \times E$ (A:T1, E:T4)

- Product
  - $E \times F$ (E:T1, F:T2)
  - $A \times B$ (A:T3, B:{C:T4, D:T5})

- Tuple_create $B$
  - $a_2, e_1$ (E:T1, F:T2, A:T3, B:{C:T4, D:T5})

- Tuple_destroy $B$
  - $a_1, e_1$ (E:T1, F:T2, A:T3, B:{C:T4, D:T5})
Complex Value Query Examples

- **Find the artifacts of Impressionist artists**
  \[(\text{Flatten}(\text{Tuple
destroy}((\pi\text{Artifacts}\quad(\sigma_{\text{style}='\text{Impressionist}'}(\text{FINEART}))))))\]

<table>
<thead>
<tr>
<th>Name</th>
<th>Style</th>
<th>Artifacts Title</th>
<th>Live-Time</th>
<th>Nationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Claude Monet</td>
<td>Impressionism</td>
<td>Haystacks</td>
<td>1840-1926</td>
<td>France</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wheatstacks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paul (Eugéne-Henri)</td>
<td>Impressionism</td>
<td>The Yellow Christ</td>
<td>1848-1903</td>
<td>France</td>
</tr>
<tr>
<td>Gauguin</td>
<td></td>
<td>Tahitian Women</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edouard Manet</td>
<td>Impressionism</td>
<td>The Waitress</td>
<td>1832-1883</td>
<td>France</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Em. Zola Portrait</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Complex Value Unnest & Nest

\[R1 = \text{Unnest}_{\text{Artifacts}}(\text{FINEART})\]

<table>
<thead>
<tr>
<th>Name</th>
<th>Style</th>
<th>Title</th>
<th>Live-Time</th>
<th>Nationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pablo Picasso</td>
<td>Cubism</td>
<td>Guernica</td>
<td>1881-1973</td>
<td>Spain</td>
</tr>
<tr>
<td>Pablo Picasso</td>
<td>Cubism</td>
<td>Seated Woman</td>
<td>1881-1973</td>
<td>Spain</td>
</tr>
<tr>
<td>Claude Monet</td>
<td>Impressionism</td>
<td>Haystacks</td>
<td>1840-1926</td>
<td>France</td>
</tr>
<tr>
<td>Claude Monet</td>
<td>Impressionism</td>
<td>Wheatstacks</td>
<td>1840-1926</td>
<td>France</td>
</tr>
<tr>
<td>Kasimir Malevich</td>
<td>Geom. Abstract</td>
<td>Solder of the 1 Div.</td>
<td>1878-1935</td>
<td>Russia</td>
</tr>
<tr>
<td>Kasimir Malevich</td>
<td>Geom. Abstract</td>
<td>The Woodcutter</td>
<td>1878-1935</td>
<td>Russia</td>
</tr>
<tr>
<td>Alberto Giacometti</td>
<td>Surrealism</td>
<td></td>
<td>1901-1966</td>
<td>Swiss</td>
</tr>
<tr>
<td>Paul Gauguin</td>
<td>Impressionism</td>
<td>The Yellow Christ</td>
<td>1848-1903</td>
<td>France</td>
</tr>
<tr>
<td>Paul Gauguin</td>
<td>Impressionism</td>
<td>Tahitian Women</td>
<td>1848-1903</td>
<td>France</td>
</tr>
<tr>
<td>Edouard Manet</td>
<td>Impressionism</td>
<td>The Waitress</td>
<td>1832-1883</td>
<td>France</td>
</tr>
<tr>
<td>Edouard Manet</td>
<td>Impressionism</td>
<td>Em. Zola Portrait</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Complex Value Unnest & Nest

\[ R_2 = \text{Nest} \text{ArtWork} = \text{Name, Title, Live-Time, Nationality} (R_1) \]

<table>
<thead>
<tr>
<th>Style</th>
<th>Name</th>
<th>Title</th>
<th>Live-Time</th>
<th>Nationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cubism</td>
<td>Pablo Picasso</td>
<td>Guernica</td>
<td>1881-1973</td>
<td>Spain</td>
</tr>
<tr>
<td>Impressionism</td>
<td>Pablo Picasso</td>
<td>Seated Woman</td>
<td>1881-1973</td>
<td>Spain</td>
</tr>
<tr>
<td></td>
<td>Claude Monet</td>
<td>Haystacks</td>
<td>1840-1926</td>
<td>France</td>
</tr>
<tr>
<td></td>
<td>Claude Monet</td>
<td>Wheatstacks</td>
<td>1840-1926</td>
<td>France</td>
</tr>
<tr>
<td></td>
<td>Paul Gauguin</td>
<td>The Yellow Christ</td>
<td>1832-1883</td>
<td>France</td>
</tr>
<tr>
<td></td>
<td>Paul Gauguin</td>
<td>Tahitian Women</td>
<td>1832-1883</td>
<td>France</td>
</tr>
<tr>
<td></td>
<td>Edouard Manet</td>
<td>The Waitress</td>
<td>1832-1883</td>
<td>France</td>
</tr>
<tr>
<td></td>
<td>Edouard Manet</td>
<td>Em. Zola Portrait</td>
<td>1832-1883</td>
<td>France</td>
</tr>
<tr>
<td>Geom. Abstract</td>
<td>Kasimir Malevich</td>
<td>Solder of the 1 Div.</td>
<td>1878-1935</td>
<td>Russia</td>
</tr>
<tr>
<td></td>
<td>Kasimir Malevich</td>
<td>The Woodcutter</td>
<td>1878-1935</td>
<td>Russia</td>
</tr>
<tr>
<td>Surrealism</td>
<td>Alberto Giacometti</td>
<td></td>
<td>1901-1966</td>
<td>Swiss</td>
</tr>
</tbody>
</table>

- Unnest is the right inverse of nest, whereas unnest is in general not information preserving

\[
\text{Nest} \text{Artifacts} = \text{Title} (\text{Unnest} \text{Artifacts} (\text{FINEART})) \neq \text{FINEART}
\]

\[
\text{Unnest} \text{Artifacts} (\text{Nest} \text{Artifacts} = \text{Title} (R_1))) = R_1
\]

- Why?
  - Grouped attributes may be empty
  - Remaining attributes may have the same values
Complex Value Query Languages: Calculus

- Relies on (first-order) relational calculus + quantification over set (second-order) + many sorted (complex value types)
  - **Alphabet** of constants $D$ (complex values) + a set $V$ of variable names + a set $A$ of attribute names
  - **Terms**:
    - constants of $D$ (e.g. 1, [name: "Vassilis"], etc.)
    - variables of $V$ (e.g., X, Y, Z, etc.)
    - expressions of the form $t.a$ where $a$ an attribute of $A$ and $t$ a tuple term
    - Positive Literals: $R(t)$, $t = t'$, $t < t'$
    - Well Formed Formulas: and, not, exists, forall
    - Query: $\{X \mid \phi\}$ where $X$ is the only free variable in $\phi$
- Range Restrictions to formulae variables guarantee safety of calculus
- Set-type Variables + PowerSet result in more expressive power than the relational calculus (equivalent to Datalog-Transitive Closure)
- **THEOREME**: The above complex value algebra and the safe calculus have the same expressive power [Kupper, Abiteboul, Grumbach, Beeri].

A Critic of Complex Value Databases

- Compared to the Relational Model
  - **More** Complex Attribute Values
  - **Typed** Data Model
  - **Expressive Power** of N1NF Languages
- But
  - **No** Object Identity (Data Sharing)
  - **No** Programming Language (Extensibility, Flexibility)