$\mathcal{L}$-Fuzzy Databases in Arrow Categories

Evans Adjei
Abdul Wazed Chowdhury
Michael Winter

Department of Computer Science
Brock University
St. Catharines, Ontario, Canada

{*ea12gq*|*wc12ss*|*mwinter*}*@brocku.ca*

September 30, 2015

Overview

Structured Query Language and Relational Databases.

School Table

ID	Name
S001	University of Technology
S002	University of Applied Science

Student Table

School ID	ID	Name	DOB
S001	UT-1000	Tommy	05/06/1995
S001	UT-1000	Better	16/04/1995
S002	UAS-1000	Linda	02/09/1995
S002	UAS-1000	Jonathan	22/06/1995

Fig. 1: An example of a database table.

- A database consists of relations (tables) with attributes (columns) and unique instances with values for each attribute (rows).
- SQL is the standard language used to interact with a relational database.
- Relational databases are based on the classical set theory.

FSQL and Fuzzy Relational Databases

Crisp relational databases and SQL can not handle imprecise or vague information.

A fuzzy set maps elements to the fixed unit interval [0,1].

Fuzzy databases and FSQL are based on the properties and operations of fuzzy theory.

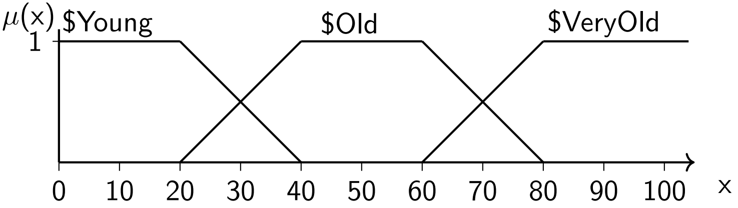

Figure: 3 Age distribution in terms of linguistic labels

Elements of FSQL

Select Statement

SELECT Name, Age, Ability, **CDEG(*)**
FROM Student
WHERE Age **NFEQ $Young THOLD 0.6**
AND Ability **FGEQ $Skilled THOLD 0.6**

- **Linguistic Labels:** Names of fuzzy sets already stored in the Meta-knowledge base. E.g. **$Young**, **$Long**.
- **Fulfillment Threshold:** It represents an α - cut i.e., the condition is true if its degree is greater than the threshold.
- **Logical Connectors:** They combine two or more conditions. E.g. **AND**, **OR**.

The Elements of FSQL.

Fuzzy Comparators: They are used to compare two attributes of the same type or an attribute and a constant.

- **Possibility Comparators:** They select tuples which generally comply to a given condition.
- In other words, the condition is true for all or some of the possible instances.
  E.g. Possibly Fuzzy Greater Than (**FGT** or **F**$>$).

- **Necessity Comparators:** They select tuples which strictly comply to a given condition.
- In other words, the condition is true for all possible instances.
  Eg. Necessarily Fuzzy Less Or Equal To (**NFLEQ** or **NF**$\leqslant$).

An Example of a Fuzzy SQL Statement.

Std_Id	Name	Age (yrs)	Height (ft)	Courses
104	Tom Benson	32	5.5	4
108	Peter Yankey	38	6.8	5
102	Eric Boadu	26	5.9	3
106	John Smith	41	6.9	4

Table 2: Students' Records.

SELECT Std_Id, Name, Course, **CDEG(*) FROM** Students **WHERE** Age **FEQ $Young 0.8 AND** Height **FGEQ $Tall 0.6**

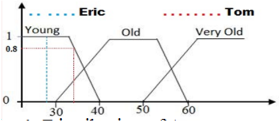

Fig. 4: Distribution of Age.

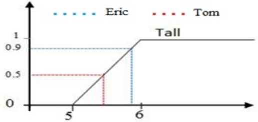

Fig. 5: Distribution of Height.

CDEG	St_ID	Name	Course
0.9	102	Eric Boadu	3

Table 3: Resulting table.

The Concept of $\mathcal{L}$-Fuzziness.

- It is a generalization of the fuzziness.
- An $\mathcal{L}$-fuzzy set maps elements to arbitrary lattice $\mathcal{L}$ of membership degrees.
- $\mathcal{L}$ can be $\{0, 1\}$ modeling crisp sets or $[0, 1]$ modeling the ordinary fuzzy set developed by Zadeh.
- It allows a more flexible selection of membership degrees than the unit interval [0,1].
- It gives variety of degrees of membership.
- It also comes with essential properties and operations.

The Matrix Representation of $\mathcal{L}$-Fuzzy Relation

$$\begin{tikzcd}[column sep=small]
& 1 \arrow[dl, dash] \arrow[dr, dash] & \\
NI^+ \arrow[dr, dash] & & TI^+ \arrow[dl, dash] \\
& I \arrow[dl, dash] \arrow[dr, dash] & \\
NI^- \arrow[dr, dash] & & TI^- \arrow[dl, dash] \\
& 0 &
\end{tikzcd}$$

Fig. 8: Lattice structure of students' responses.

$$\begin{array}{cc}
 & \begin{matrix} P.Sci & Comp & Geog & Maths \end{matrix} \\
\begin{matrix} S1 \\ S4 \\ S3 \\ S2 \\ S5 \end{matrix} &
\begin{pmatrix}
1 & TI^+ & NI^+ & TI^- \\
0 & 1 & I & TI^+ \\
NI^+ & TI^- & 1 & 0 \\
0 & NI^+ & NT^- & 1 \\
TI^+ & 1 & 0 & I
\end{pmatrix}
\end{array}$$

Fig. 9: Representation of the students' records in $\mathcal{L}$-fuzzy relation.

Every entry into the $\mathcal{L}$-fuzzy database is an $\mathcal{L}$- fuzzy set.

- We can explicitly list an $\mathcal{L}$-fuzzy set in the database. E.g. $\{\alpha/5.5\text{ft},.., \alpha/5.9\text{ft}\}$, $\{1/25\text{yrs},\gamma/29\text{yrs}\}$.
- We can obtain $\mathcal{L}$-fuzzy set through a pre-implemented characteristic function. E.g.

$$\textbf{\$Young}(x) = \begin{cases} 1 & \text{iff } x \leqslant 25yrs, \\ \gamma & \text{iff } 25yrs < x \leqslant 29yrs, \\ \alpha & \text{iff } 29yrs < x \leqslant 39yrs, \\ 0 & \text{otherwise} \end{cases}$$

$\mathcal{L}$-Fuzzy Databases and $\mathcal{L}$-Fuzzy Data.

- We can obtain new fuzzy set through computation of upper bounds or lower bounds if the domain is ordered. E.g.

  $lbd(\textbf{\$Old}) = \textbf{\$Young}$.

  $ubd(\textbf{\$Old}) = \textbf{\$Very\_Old}$.

- We can also achieve new set through approximate equality.

  $extremely(\equiv, s)$ and $very(\equiv, s)$ are strengthening modifiers.

  $more\_or\_less(\equiv, s)$ and $roughly(\equiv, s)$ are loosening modifiers.

The Elements of $\mathcal{L}$FSQL.

$\mathcal{L}$-Fuzzy Comparators: They compare an attribute and an $\mathcal{L}$-Fuzzy value or attributes of the same kind.

- **Possibility Comparators** are based on general criteria.

  \$Heavy $= \{..,0/64,\beta/65,..,\ \beta/70,\ \delta/71,\ ..,\delta/80,\ 1/81,..\}$.

  Tina's Weight could be 69kg, 70kg or 71.

  Is Tina's Weight **F= \$Heavy** with degree δ? Yes.

- **Necessity Comparator** are based on all the specified criteria.

  Is Tina's weight **NF= \$Heavy** with degree δ? No.

The Elements of $\mathcal{L}$FSQL.

$\mathcal{L}$-**Fuzzy Threshold** puts additional restriction on the condition in the **WHERE** clause.

- It has the word **THOLD** and comes with a degree.
- The $\mathcal{L}$-fuzzy threshold is optional.

  Weight **F**$\leqslant$ **$Heavy THOLD** α

$\mathcal{L}$-**Fuzzy Logical Connectors** combine two or more conditions.

- The **OR** Logical Connector is basically the join operator of the given lattice $\mathcal{L}$.

  Weight **F**$\leqslant$ **$Heavy THOLD** α **OR** Height **NF**$\geqslant$ **$Tall**

The Elements of $\mathcal{L}$FSQL

- The **AND** Logical Connector is basically the meet operator of the given lattice $\mathcal{L}$.

  Weight **F**$\leqslant$ **$Heavy THOLD** α **AND** Height **NF**$\geqslant$ **$Tall**

- We can obtain additional logical connectors through lattice-ordered semigroups.

- Such connectors could be **AND($*$)** or **OR($*$)**.

- The user can specify the operation he/she wants to use.

  Weight **F**$\leqslant$ **$Heavy THOLD** α **AND($*$)** Height **NF**$\geqslant$ **$Tall**.

The $\mathcal{L}$FSQL Create Statement.

- The **CREATE** statement creates an empty table.
- It contains the table's name, attributes' names and the domains of the attributes.
- The table's name and the attributes' names must be unique in the database and the table respectively.

CREATE TABLE Players (Player_Id String, Name String, Height Float, Weight Float, Contract Int);

The $\mathcal{L}$FSQL Insert Statement

- The **INSERT** statement adds new record to an existing table.
- The values to be inserted should be $\mathcal{L}$-fuzzy sets.
- Each value must belong to the domain of their corresponding attribute.

INSERT INTO Players (Player_Id, Name, Height, Weight, Contract) **VALUES** (P10, Kelly, **$Tall**, 75, 5);

- The values P10 and Kelly are crisp value i.e., $\mathcal{L}$-fuzzy set with one element.

The $\mathcal{L}$FSQL Select Statement

- The **SELECT** statement retrieves records which satisfy a given condition.
- It has the **SELECT** clause indicating which attribute(s) to retrieve the record(s) from.
- The **FROM** clause contains the table(s) from which we can compare and retrieve values from.
- The **WHERE** clause contains the condition(s) to be satisfied.

SELECT Player_Id, Name, Height, Weight **FROM** Players **WHERE** Height F= **$Tall THOLD** γ **AND** Weight NF$\leqslant$ **$Heavy THOLD** δ;

The $\mathcal{L}$FSQL Delete Statement

- The **DELETE** statement acts like the **SELECT** statement.
- It drops the selected records from the table.
- It states the name of table from which the records are to be deleted and the condition(s) to be satisfied.
- The **WHERE** clause in the **DELETE** statement is the same as the one in the **SELECT** statement.

  DELETE FROM Player **WHERE** Weight F$\geqslant$ **$Heavy THOLD** δ;

Example of an $\mathcal{L}$FSQL Statement

$\$Tall = \{0/5.0,.., 0/5.4, \alpha/5.5,.., \alpha/5.9, \gamma/6.0,.., \gamma/6.4, 1/6.5,..\}$
$\$Heavy = \{0/45,..,0/64, \beta/65,.., \beta/70, \delta/71, .., \delta/80, 1/81,..\}$

Player_Id	Name	Height (ft.)	Weight
P10	Kelly	$Tall	75
P08	Thomas	5.2	$Heavy
P02	John	{5.4,5.5,6}	79

Table 4: Players' records.

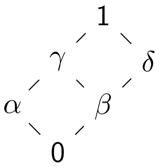

Figure: Lattice

SELECT Player_Id, Name, Height, Weight **FROM** Players **WHERE** Height **F**$\geqslant$ **$Tall** γ **AND** Weight **NF**$\leqslant$ **$Heavy** δ;

Player_Id	Name	Height (ft.)	Weight
P10	Kelly	$Tall	75
P02	John	{5.4,5.5,6}	79

Table 5: Resulting table.

A Suitable Categorical Framework of $\mathcal{L}$-Fuzzy Relations

- $\mathcal{L}$-**Rel** defines the basic operations on $\mathcal{L}$-fuzzy relations
- Freyd and Scedrov introduced and then extended **allegories** as a categorical relational calculus
- Olivier and Sarrato introduced **Dedekind categories** which are equivalent to locally complete distributive allegories
- It can be shown that the class of $\mathcal{L}$-fuzzy relations form a Dedekind category, but it is too weak to express 0-1 crispness
- **Arrow categories** extend Dedekind categories by the addition of two arrow operations: the up-arrow ($\uparrow$) and the down-arrow ($\downarrow$)

$$\begin{pmatrix} 0.5 & 0 & 0.9 \\ 0.4 & 1 & 0 \\ 1 & 0.2 & 0 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

The Arrow Category $\mathcal{A}$ and the Interpretations

- We require that the **relational products**, **relational sums**, splittings, unit, zero object, injections, projections in $\mathcal{A}$ to be crisp.
- In $\mathcal{A}$ the complete **Heyting algebra** of **scalar elements** is isomorphic to $\mathcal{L}$. For every $l \in \mathcal{L}$ we have a scalar $I(l)$ in $\mathcal{A}$.

$$\begin{tikzcd}[arrows=dash]
& & 1 \arrow[dl] \arrow[dr] & \\
& c \arrow[dl] \arrow[dr] & & d \arrow[dl] \\
a \arrow[dr] & & b \arrow[dl] & \\
& 0 & &
\end{tikzcd}
\qquad
\begin{array}{c c}
 & \begin{array}{cccc} a_1 & a_2 & \dots & a_n \end{array} \\
\begin{array}{c} a_1 \\ a_2 \\ \dots \\ a_n \end{array} & \begin{pmatrix} c & 0 & 0 & 0 \\ 0 & c & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & c \end{pmatrix}
\end{array}
\qquad
\begin{tikzcd}[arrows=dash]
1 \arrow[d] \\
m \arrow[d] \\
0
\end{tikzcd}$$

- A domain D is interpreted by an object $I(D)$ in $\mathcal{A}$ and an element $d \in D$, by a crisp point $I(d) : \mathbf{1} \to I(D)$

$$I(Richy) = \begin{array}{c c} & \begin{array}{ccccc} John & Kevin & Linda & Richy & Tijo \end{array} \\ * & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

- For a natural number n, $I(n)$ denotes the object $\underbrace{1 + \cdots + 1}_{n-\text{times}}$

Semantics of $\mathcal{L}$-Fuzzy Sets

- $\sigma_{\mathbf{s}}$ is the metadatabase that stores $\mathcal{L}$ sets, i.e., $\sigma_s(\$m) : 1 \to I(D)$
- For explicitly given sets: $\llbracket\{l_1/d_1, \dots, l_n/d_n\}\rrbracket(\sigma_s) = \bigsqcup_{i=1}^{n} I(l_i); I(d_i)$

$$\llbracket\{c/4, 1/5, d/6\}\rrbracket(\sigma_s) = \begin{array}{cc} & * \\ * & \begin{pmatrix} c \end{pmatrix} \end{array}; \begin{array}{cc} & \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & \dots \end{array} \\ * & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \sqcup$$

$$\begin{array}{cc} & * \\ * & \begin{pmatrix} 1 \end{pmatrix} \end{array}; \begin{array}{cc} & \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & \dots \end{array} \\ * & \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array} \sqcup$$

$$\begin{array}{cc} & * \\ * & \begin{pmatrix} d \end{pmatrix} \end{array}; \begin{array}{cc} & \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & \dots \end{array} \\ * & \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

$$= \begin{array}{cc} & \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & \dots \end{array} \\ * & \begin{pmatrix} 0 & 0 & 0 & c & 1 & d & 0 & 0 \end{pmatrix} \end{array}$$

- For linguistic labels: $\llbracket\$m\rrbracket(\sigma_s) = \sigma_s(\$m)$

$$\llbracket\$Close\rrbracket(\sigma_s) = \sigma_s(\$Close) = \begin{array}{cc} & \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & \dots \end{array} \\ * & \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & c & a & 0 \end{pmatrix} \end{array}$$

Semantics of Tables

In the non-fuzzy case a table R can be seen as a finite subset of $D_1 \times \cdots \times D_n$. Relation algebraically this can be modelled by

- a point $[\![R]\!] : 1 \to \mathcal{P}(I(D_1) \times \cdots \times I(D_n))$, where $\mathcal{P}(X)$ is an abstract version of a power set construction
- a vector $[\![R]\!] : 1 \to I(D_1) \times \cdots \times I(D_n)$
- a function $[\![R]\!] : I(r) \to I(D_1) \times \cdots \times I(D_n)$ since we deal with finite sets.

In fuzzy database, the target object within last option becomes $\mathcal{P}(I(D_1)) \times \cdots \times \mathcal{P}(I(D_n))$ which isomorphic to $\mathcal{P}(I(D_1) + \cdots + I(D_n))$

Having a function of the form $[\![R]\!] : I(r) \to \mathcal{P}(I(D_1) + \cdots + I(D_n))$ is equivalent to having a relation of the form $[\![\mathbf{R}]\!] : \mathbf{I}(\mathbf{r}) \to \mathbf{I}(\mathbf{D_1}) + \cdots + \mathbf{I}(\mathbf{D_n})$.

Semantics of Tables (cont.)

A	B
a	d
b	c

(A) A classical database

A	B
{m/a,1/b}	{0/c,m/d}
{0/a,m/b}	{1/c,0/d}

(B) An $\mathcal{L}$-fuzzy database

$$\begin{array}{ccc} \mathcal{L}^A & \times & \mathcal{L}^B \end{array}$$
$$\left(\begin{array}{ccc} \{m/a, & & \{0/c, \\ 1/b\} & , & m/d\} \end{array} \right)$$

(C) $A \times B$ $\mathcal{L}$-subset

$$\begin{array}{c} \mathcal{L}^{A+B} \\ \left\{ \begin{array}{c} m/a \\ 0/c \\ 1/b \\ m/d \end{array} \right\} \end{array}$$

(E) $A + B$ $\mathcal{L}$-subset

$$\begin{array}{cc} & \begin{array}{ccccccccc} S_1 & S_2 & S_3 & S_4 & S_5 & S_6 & ... & S_{80} & S_{81} \end{array} \\ \begin{array}{c} 1 \\ 2 \end{array} & \left(\begin{array}{ccccccccc} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array} \right) \end{array}$$

(D) The table as a crisp relation (to $\mathcal{L}^A \times \mathcal{L}^B$)

$$\begin{array}{cc} & \begin{array}{ccccccccc} T_1 & T_2 & T_3 & T_4 & T_5 & T_6 & ... & T_{80} & T_{81} \end{array} \\ \begin{array}{c} 1 \\ 2 \end{array} & \left(\begin{array}{ccccccccc} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right) \end{array}$$

(F) The table as a crisp relation (to $\mathcal{L}^{A+B}$)

$$\begin{array}{cc} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} 1 \\ 2 \end{array} & \left(\begin{array}{cccc} m & 1 & 0 & m \\ 0 & m & 1 & 0 \end{array} \right) \end{array}$$

(G) The table as an $\mathcal{L}$-relation

Figure: Semantics of $\mathcal{L}$-fuzzy tables

Semantics of Tables (cont.)

- $\sigma_{\mathbf{t}}$ maps a table name to its semantics, i.e., a relation of the form $[\![R]\!] : I(r) \to I(D_1) + \cdots + I(D_n)$
- Also, $\sigma_t[Q/R]$ denotes the update of the database at table R by the relation Q. Mathematically,

$$\sigma_t[Q/R](X) = \begin{cases} Q, & \text{if } X = R, \\ \sigma_t(X), & \text{otherwise.} \end{cases}$$

- Selecting an attribute: $[\![R.A_i]\!](\sigma_s, \sigma_t) = [\![R.A_i]\!](\sigma_t) = \sigma_t(R); \iota_i^{\smile}$

$$= \begin{array}{c c} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} 1 \\ 2 \end{array} & \begin{pmatrix} m & 1 & 0 & m \\ 0 & m & 1 & 0 \end{pmatrix} \end{array} ; \begin{array}{c c} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} a \\ b \end{array} & \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}^{\smile} \end{array}$$

$$= \begin{array}{c c} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} 1 \\ 2 \end{array} & \begin{pmatrix} m & 1 & 0 & m \\ 0 & m & 1 & 0 \end{pmatrix} \end{array} ; \begin{array}{c c} & \begin{array}{cc} a & b \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \end{array} & \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \end{array} = \begin{array}{c c} & \begin{array}{cc} a & b \end{array} \\ \begin{array}{c} 1 \\ 2 \end{array} & \begin{pmatrix} m & 1 \\ 0 & m \end{pmatrix} \end{array} .$$

Semantics of INSERT Statement

Let us say $\{m_1, \ldots, m_n\}$ are $\mathcal{L}$-fuzzy subsets of the domains $(D_1, \ldots, D_n)$. We define the semantics for a tuple of that table by

$$[\![(m_1, \ldots, m_n)]\!](\sigma_s) = \bigsqcup_{i=1}^{n} [\![m_i]\!](\sigma_s); \iota_i.$$

Example:

$$[\![(\{1/a, m/b\}, \{0/c, 1/d\})]\!](\sigma_s) = [\![\{1/a, m/b\}]\!](\sigma_s); \iota_1 \sqcup [\![\{0/c, 1/d\}]\!](\sigma_s); \iota_2$$

$$= \begin{array}{c} \\ * \end{array} \begin{array}{cc} a & b \\ \multicolumn{2}{c}{\begin{pmatrix} 1 & m \end{pmatrix}} \end{array}; \begin{array}{c|c} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} a \\ b \end{array} & \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \end{array} \sqcup \begin{array}{c} \\ * \end{array} \begin{array}{cc} c & d \\ \multicolumn{2}{c}{\begin{pmatrix} 0 & 1 \end{pmatrix}} \end{array}; \begin{array}{c|c} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} c \\ d \end{array} & \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

$$= \begin{array}{c} \\ * \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{pmatrix} 1 & m & 0 & 0 \end{pmatrix} \end{array} \sqcup \begin{array}{c} \\ * \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

$$= \begin{array}{c} \\ * \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{pmatrix} 1 & m & 0 & 1 \end{pmatrix} \end{array}$$

Semantics of INSERT Statement (cont.)

In order to deduce the semantics for the final table where this tuple has been added, we refer to the following figure.

$$\begin{tikzcd}
r \arrow[d, "\iota"'] \arrow[dr, "\sigma_t(R)"] & \\
r+1 \arrow[r, "{[\![R(m_1,\ldots,m_n)]\!](\sigma_s,\sigma_t)}"] & I(D_1)+\cdots+I(D_n) \\
1 \arrow[u, "\kappa"] \arrow[ur, "{[\![(m_1,\ldots,m_n)]\!](\sigma_s)}"'] &
\end{tikzcd}$$

- The table has r rows: $I(r) \to I(D_1) + \cdots + I(D_n)$
- A vector $1 \to I(D_1) + \cdots + I(D_n)$ represents the tuple to be inserted
- The resultant table has $r+1$ tuple and maps from the object $r+1$ to $I(D_1) + \cdots + I(D_n)$ in $\mathcal{A}$.

$$[\![R(m_1,\ldots,m_n)]\!](\sigma_s,\sigma_t) = \iota^{\smile};\sigma_t(R) \sqcup \kappa^{\smile};[\![(m_1,\ldots,m_n)]\!](\sigma_s)$$

Semantics of INSERT Statement: An Example

$$[\![R(m_1,\ldots,m_n)]\!](\sigma_s,\sigma_t) = \iota^{\smile};\sigma_t(R) \sqcup \kappa^{\smile};[\![(m_1,\ldots,m_n)]\!](\sigma_s)$$

$$[\![R(\{1/a, m/b\},\{0/c,1/d\})]\!](\sigma_s,\sigma_t)$$

$$= \begin{array}{c|c} & \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{array}{c} 1 \\ 2 \end{array} & \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \end{array}^{\smile} ; \begin{array}{c|c} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} 1 \\ 2 \end{array} & \begin{pmatrix} m & 1 & 0 & m \\ 0 & m & 1 & 0 \end{pmatrix} \end{array} \sqcup \begin{array}{c|c} & * \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \end{array} ; \begin{array}{c|c} & \begin{array}{cccc} a & b & c & d \end{array} \\ * & \begin{pmatrix} 1 & m & 0 & 1 \end{pmatrix} \end{array}$$

$$= \begin{array}{c|c} & \begin{array}{cc} 1 & 2 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \end{array} ; \begin{array}{c|c} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} 1 \\ 2 \end{array} & \begin{pmatrix} m & 1 & 0 & m \\ 0 & m & 1 & 0 \end{pmatrix} \end{array} \sqcup \begin{array}{c|c} & * \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \end{array} ; \begin{array}{c|c} & \begin{array}{cccc} a & b & c & d \end{array} \\ * & \begin{pmatrix} 1 & m & 0 & 1 \end{pmatrix} \end{array}$$

$$= \begin{array}{c|c} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} m & 1 & 0 & m \\ 0 & m & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \sqcup \begin{array}{c|c} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & m & 0 & 1 \end{pmatrix} \end{array} = \begin{array}{c|c} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} m & 1 & 0 & m \\ 0 & m & 0 & 1 \\ 1 & m & 0 & 1 \end{pmatrix} \end{array}$$

Semantics of INSERT Statement (cont.)

Note that the first injection ι maps from the r-ary relational sum of the unit object to the $(r+1)$-ary relational sum. In our example, as the table already contains 2 tuples and so the final table would have 3 tuples, ι injects to the object $I(3)$.

Finally, we define the semantics of an INSERT statement as follows:

$$\llbracket \text{INSERT INTO } R \text{ VALUES } (m_1, \ldots, m_n); \rrbracket(\sigma_s, \sigma_t)$$
$$= \sigma_t[\llbracket R(m_1, \ldots, m_n) \rrbracket(\sigma_s, \sigma_t)/R].$$

From the definition of update function it is clear that the new relation replaces the existing one for the particular table.

Thank You