Data Structure Estimation Tutorial

Martin Řimnáč

Ústav informatiky AV ČR

Seminář SEMWEB, Stachy
05. - 07. října 2006

Introduction

Motto: The web pages contain a lot of "human oriented" information.
Is there any possibility to search any part of the web not for the list of the pages containing the query relevant information, but directly the relevant information?

1. `Semantic Web Ideas` (RDF, OWL, reasoners,...)
2. `Data Extraction` from web pages and querying...
   - Product catalogs
   - Can be this process automatic in practice?
     - If so, the `Data Structure` has to be known

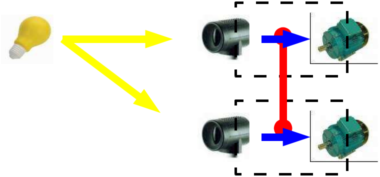
Importance of Relationships?

- Mechanism 1:

Importance of Relationships?

- Mechanism 1:

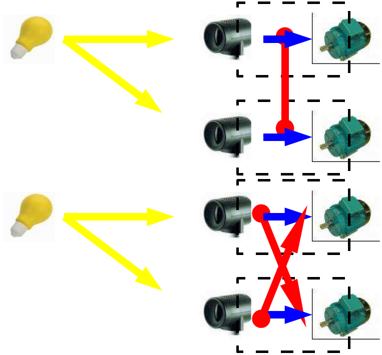
- Mechanism 2:



Importance of Relationships?

- Mechanism 1:

- Mechanism 2:



Naïve Algorithm

- Input: a set of tuples
- Output: a set of functional dependencies driven by the input (on the extensional level related to the input tuple set)

$$M'' = \emptyset$$

$$\text{for } \forall A \in \mathscr{A}(R) \quad (1)$$

$$\text{for } \forall X \in \mathscr{P}(\mathscr{A}(R) - A) \quad (2)$$

$$\text{if } \exists \mathscr{I} : \mathscr{D}_\alpha(X) \rightarrow \mathscr{D}_\alpha(A) \text{ then} \quad (3)$$

$$M'' := M'' \cup \{X \rightarrow A\}$$

Drawbacks:

- NP complexity
- The model M is not minimal

Model Skeleton

- Def: Minimal subset implying all valid functional dependencies
- No trivial functional dependencies
- Due to transitivity - searching for the closure of the set.
  - Ambiguous solutions
  - The model expressiveness can be maximalised:

$$M = \arg \min_{M' \sim M''} \{ \sum_{\forall f \in M'} \sigma(f) \} \tag{4}$$

  - where:

$$\sigma(A_i \rightarrow A_j) = |\mathscr{D}_\alpha(A_i)| - |\mathscr{D}_\alpha(A_j)| \tag{5}$$

- `Incremental model skeleton building`
  - Incremental step: Polynomial complex issue

Representation Proposal

- As simple as possible (only simple functional dependencies)
- To handle the fact f_i implies f_j:
  - The value of the attribute A_i implies the value of the attribute A_j ($A_i \rightarrow A_j \in M$)
  - The element e_i (attribute - value pair) implies the element e_j ($e_i \rightsquigarrow e_j$)
- The proposal is:
  - to use binary matrix H defined as: $h_{ij} = \begin{cases} 1 & \text{if } f_i \rightsquigarrow f_j, \\ 0 & \text{otherwise} \end{cases}$
  - Generalisation
    - Which facts are implied by ones represented by the vector x: $y = H \cdot x$
  - Specialisation
    - From which facts can be x implied $z = H^{-1} \cdot x = H^T \cdot x$

Transitivity - Minimalisation

1. Due to transitivity:
   - One generalisation step: $y_1 = H \cdot x$
   - The second step: $y_2 = H \cdot y_1 = H \cdot H \cdot x$
   - General: $y_n = H^n \cdot x$
     - $n \leq size(H)$ (complexity estimation)
2. Matrix Form:
   - Full (H^n): redundant items / reachable in one step
   - Minimal (H): no redundant items / reachable in n steps
3. The `Minimalisation Issue`:
   - to find G and $n : H = G^n$
   - G is as minimal as possible if $\nexists n' > n : G^n \neq G^{n'}$
   - ambiguous solution
     - a minimalising/maximalising criteria can be used

Notion

Let be:

- Indeces:
  - I_A the attribute index ($\mathscr{A} \longrightarrow N$)
  - I_T the term (value) index ($\bigcup_{\forall A \in \mathscr{A}} \{\mathscr{D}_\alpha(A)\} \longrightarrow N$)
  - I_E the element index ($I_A \times I_T \longrightarrow N$)
- Matrices:
  - Ω the valid single functional dependency matrix
  - $\mho$ the corrupted functional dependency matrix
  - Δ the attribute active domain matrix

$$\delta_{I_E(e), I_A(A)} = \begin{cases} 1 & \text{if } e = (A, v), v \in \mathscr{D}_\alpha(A) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

  - Φ the repository matrix

Example

Let the set $\mathscr{T}$ represent tuples:

$$\begin{array}{cccc} K & A & B & C \\ k_1 & 0 & 0 & 0 \\ k_2 & 0 & 1 & 1 \\ k_3 & 1 & 0 & 1 \end{array} \tag{7}$$

$$\Delta = \begin{array}{c|ccccccccc} & \frac{K}{k_1} & \frac{A}{0} & \frac{B}{0} & \frac{C}{0} & \frac{K}{k_2} & \frac{A}{1} & \frac{C}{1} & \frac{K}{k_3} & \frac{B}{1} \\ K & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ A & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ B & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ C & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{array} \tag{8}$$

Initialisation

For the tuple $\{K = k_1, A = 0, B = 0, C = 0\}$, the matrices will be :

- $\Omega_1 =$

	K	A	B	C
K	1	1	1	1
A	1	1	1	1
B	1	1	1	1
C	1	1	1	1

- $\Phi_1 =$

	$\frac{K}{k_1}$	$\frac{A}{0}$	$\frac{B}{0}$	$\frac{C}{0}$	$\frac{K}{k_2}$	$\frac{A}{1}$	$\frac{C}{1}$	$\frac{K}{k_3}$	$\frac{B}{1}$
$K\|k_1$	1	1	1	1	0	0	0	0	0
$A\|0$	1	1	1	1	0	0	0	0	0
$B\|0$	1	1	1	1	0	0	0	0	0
$C\|0$	1	1	1	1	0	0	0	0	0
$K\|k_2$	0	0	0	0	0	0	0	0	0
$A\|1$	0	0	0	0	0	0	0	0	0
$C\|1$	0	0	0	0	0	0	0	0	0
$K\|k_3$	0	0	0	0	0	0	0	0	0
$B\|1$	0	0	0	0	0	0	0	0	0

Initialisation

For the tuple $\{K = k_2, A = 1, B = 0, C = 1\}$, the matrices will be :

- $$\Omega_2 = \begin{array}{c|cccc} & K & A & B & C \\ \hline K & 1 & 1 & 1 & 1 \\ A & 1 & 1 & 1 & 1 \\ B & 1 & 1 & 1 & 1 \\ C & 1 & 1 & 1 & 1 \end{array}$$

- $$\Phi_2 = \begin{array}{c|ccccccccc} & \frac{K}{k_1} & \frac{A}{0} & \frac{B}{0} & \frac{C}{0} & \frac{K}{k_2} & \frac{A}{1} & \frac{C}{1} & \frac{K}{k_3} & \frac{B}{1} \\ \hline K|k_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A|0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ B|0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ C|0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ K|k_2 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ A|1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ C|1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ K|k_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ B|1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Repository Merging

- The repository can be merged as: $\Phi_{12} = \Phi_1 + \Phi_2$
- The repository merge result will be:

$$
\Phi_{12} = \begin{array}{c|ccccccccc}
 & \frac{K}{k_1} & \frac{A}{0} & \frac{B}{0} & \frac{C}{0} & \frac{K}{k_2} & \frac{A}{1} & \frac{C}{1} & \frac{K}{k_3} & \frac{B}{1} \\
K|k_1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
A|0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
B|0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
C|0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
K|k_2 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
A|1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
C|1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
K|k_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
B|1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

Testing to the Functional Dependency Corruption

- The repository is not consistent: From the element $B|0$ can be derived more than 1 element of 1 attribute ($A|0$ and $A|1$).
- To detect corrupted functional dependencies:

$$\mho^{\Delta} = \Delta((\Phi^{T} \cdot \Delta^{T}) \gtrdot 1) \tag{9}$$

- $$\Phi_{12} = \begin{array}{c|ccccccccc} & \frac{K}{k_1} & \frac{A}{0} & \frac{B}{0} & \frac{C}{0} & \frac{K}{k_2} & \frac{A}{1} & \frac{C}{1} & \frac{K}{k_3} & \frac{B}{1} \\ K|k_1 & 1 & 1 & \underline{1} & 1 & 0 & 0 & 0 & 0 & 0 \\ A|0 & 1 & 1 & \underline{1} & 1 & 0 & 0 & 0 & 0 & 0 \\ B|0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ C|0 & 1 & 1 & \underline{1} & 1 & 0 & 0 & 0 & 0 & 0 \\ K|k_2 & 0 & 0 & \underline{1} & 0 & 1 & 1 & 1 & 0 & 0 \\ A|1 & 0 & 0 & \underline{1} & 0 & 1 & 1 & 1 & 0 & 0 \\ C|1 & 0 & 0 & \underline{1} & 0 & 1 & 1 & 1 & 0 & 0 \\ K|k_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ B|1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Testing to the Functional Dependency Corruption

- The functional dependency matrices and repository are updated according to $\mho^{\Delta}$:
  - $\Omega := \Omega - \mho^{\Delta}$
  - $\mho := \mho + \mho^{\Delta}$
- $$\Phi_{12} = \begin{array}{c|ccccccccc} & \frac{K}{k_1} & \frac{A}{0} & \frac{B}{0} & \frac{C}{0} & \frac{K}{k_2} & \frac{A}{1} & \frac{C}{1} & \frac{K}{k_3} & \frac{B}{1} \\ K|k_1 & 1 & 1 & \underline{0} & 1 & 0 & 0 & 0 & 0 & 0 \\ A|0 & 1 & 1 & \underline{0} & 1 & 0 & 0 & 0 & 0 & 0 \\ B|0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ C|0 & 1 & 1 & \underline{0} & 1 & 0 & 0 & 0 & 0 & 0 \\ K|k_2 & 0 & 0 & \underline{0} & 0 & 1 & 1 & 1 & 0 & 0 \\ A|1 & 0 & 0 & \underline{0} & 0 & 1 & 1 & 1 & 0 & 0 \\ C|1 & 0 & 0 & \underline{0} & 0 & 1 & 1 & 1 & 0 & 0 \\ K|k_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ B|1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Repository merging Φ_1, Φ_2, Φ_3 result

- $$\Omega_{123} = \begin{array}{c|cccc} & K & A & B & C \\ \hline K & 1 & 0 & 0 & 0 \\ A & 1 & 1 & 0 & 0 \\ B & 1 & 0 & 1 & 0 \\ C & 1 & 0 & 0 & 1 \end{array} \qquad \mho_{123} = \begin{array}{c|cccc} & K & A & B & C \\ \hline K & 0 & 1 & 1 & 1 \\ A & 0 & 0 & 1 & 1 \\ B & 0 & 1 & 0 & 1 \\ C & 0 & 1 & 1 & 0 \end{array}$$

- $$\Phi_{123} = \begin{array}{c|ccccccccc} & \frac{K}{k_1} & \frac{A}{0} & \frac{B}{0} & \frac{C}{0} & \frac{K}{k_2} & \frac{A}{1} & \frac{C}{1} & \frac{K}{k_3} & \frac{B}{1} \\ \hline K|k_1 & 1 & \underline{0} & \underline{0} & \underline{0} & 0 & 0 & 0 & 0 & 0 \\ A|0 & 1 & 1 & \underline{0} & \underline{0} & 0 & 0 & 0 & 1 & \underline{0} \\ B|0 & 1 & \underline{0} & 1 & \underline{0} & 1 & \underline{0} & \underline{0} & 0 & 0 \\ C|0 & 1 & \underline{0} & \underline{0} & 1 & 0 & 0 & 0 & 0 & 0 \\ K|k_2 & 0 & 0 & \underline{0} & \underline{0} & 1 & \underline{0} & \underline{0} & 0 & 0 \\ A|1 & 0 & 0 & \underline{0} & \underline{0} & 1 & 1 & \underline{0} & 0 & 0 \\ C|1 & 0 & 0 & \underline{0} & \underline{0} & 1 & \underline{0} & 1 & 1 & \underline{0} \\ K|k_3 & 0 & \underline{0} & 0 & \underline{0} & 0 & 0 & 0 & 1 & \underline{0} \\ B|1 & 0 & \underline{0} & 0 & \underline{0} & 0 & 0 & 0 & 1 & 1 \end{array}$$

Repository × Functional Dependency System - Observation

- $$\Omega_{123} = \begin{array}{c|cccc} & K & A & B & C \\ K & 1 & 0 & 0 & 0 \\ A & 1 & 1 & 0 & 0 \\ B & 1 & 0 & 1 & 0 \\ C & 1 & 0 & 0 & 1 \end{array}$$

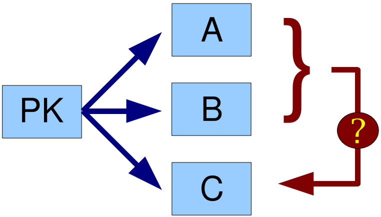
- $$\Phi_{123} = \begin{array}{l|ccccccccc} & \frac{K}{k_1} & \frac{A}{0} & \frac{B}{0} & \frac{C}{0} & \frac{K}{k_2} & \frac{A}{1} & \frac{C}{1} & \frac{K}{k_3} & \frac{B}{1} \\ K|k_1 & 1 & \underline{0} & \underline{0} & \underline{0} & 0 & 0 & 0 & 0 & 0 \\ A|0 & 1 & 1 & \underline{0} & \underline{0} & 0 & 0 & 0 & 1 & \underline{0} \\ B|0 & 1 & \underline{0} & 1 & \underline{0} & 1 & \underline{0} & \underline{0} & 0 & 0 \\ C|0 & 1 & \underline{0} & \underline{0} & 1 & 0 & 0 & 0 & 0 & 0 \\ K|k_2 & 0 & 0 & \underline{0} & \underline{0} & 1 & \underline{0} & \underline{0} & 0 & 0 \\ A|1 & 0 & 0 & \underline{0} & \underline{0} & 1 & 1 & \underline{0} & 0 & 0 \\ C|1 & 0 & 0 & \underline{0} & \underline{0} & 1 & \underline{0} & 1 & 1 & \underline{0} \\ K|k_3 & 0 & \underline{0} & 0 & \underline{0} & 0 & 0 & 0 & 1 & \underline{0} \\ B|1 & 0 & \underline{0} & 0 & \underline{0} & 0 & 0 & 0 & 1 & 1 \end{array}$$

Repository × Functional Dependency System - Formulation

- Instances in the repository Φ are instances of the functional dependencies Ω. So:
$$\Omega = \Delta \Phi \Delta^T \tag{10}$$
- The functional dependencies Ω implies the positions i, j at Φ, which can be $\phi_{ij} = 1$: $\Phi' = \Phi \odot \Delta^T \Omega \Delta$
- Precisely, these positions are implied by non-corrupted functional dependencies:
$$\Phi' = \Phi \odot \Delta^T (1 - \mho) \Delta \tag{11}$$
  - The reason is $\Omega \neq 1 - \mho$ for nonhomogenic tuples (NULL values)
    (having always all attribute covered in any tuple)

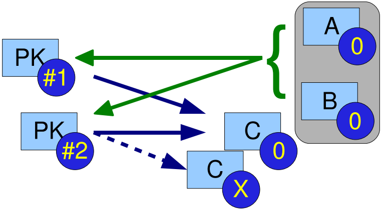
Complex Attribute Support - Condition

- If all attributes (on left and right side) of the complex dependency depend on the one key attribute A_k, all required information for handling this dependency is stored in the binary repository matrix Φ complained with metainformation, which attributes are on the left side γ_L and which is on right side γ_R.



Complex Attribute Support - Demonstration

- The query x is specialised (all from γ_L satisfied):
  - one key: Generalising of element corresponding to the attribute in γ_R. (Φ is consistent, one element to activate) (OK)
  - several keys. Generalisation activates
    - the same element (OK)
    - several different elements (NOT a functional dependency)



Complex Attribute Support - Formulation

- The existence of the key attribute can be quaranted by the requirement to the tuple uniquely idenfying attribute to be present in each tuple.
- In such a case, the generalisation mechanism can be extended by:

$$y = \Phi \cdot x + \Phi((\Phi^T \cdot ((\Delta^T \cdot \gamma_L) \odot x)) == 1) \odot (\Delta^T \cdot \gamma_R) \quad (12)$$

Conclusion

- The expressive internal representation (directly as an RDF document)
- Easy to see the complexity issue (at most given by the matrix multiplication complexity)
- The incremental algorithm with polynomial complexity.
- Algorithm implementations:
  - in PostGres database management system (driven by triggers)
  - in Octave (using PostGres for indeces and matrix storage)
  - in CLIPS (rule based system)
- The script interpreting and visualising the repository in RDF format (available trought web server)

Future Work

1. Usage of sparce matrices (graph algorithms)
2. Distributing the repository / Integration Issue
3. Storing XHTML web pages to the repository
    - the tree structure handling in the repository
    - structure modification (from formating to structure)
4. ...