A Formal Representation for Structured Data

János Demetrovics

MTA SZTAKI, Lágymányosi u. 11, H-1111 Budapest, Hungary
e-mail: demetrovics.janos@sztaki.mta.hu

Hua Nam Son, Ákos Gubán

Budapest Business School, Buzogány u. 11-13, H-1149 Budapest, Hungary
e-mail: Hua.NamSon@pszfb.bgf.hu, guban.akos@pszfb.bgf.hu

Abstract: Big Data is a technology developed for 3-V management of data by which large volumes and different varieties of data would be processed in optimal velocity. The data to be dealt with may be structured or unstructured. Relational databases (spreadsheets) are typical examples of structured data and the methods, as well as the techniques for researches of relational database management are well-known. In this paper, we describe a formalism, by which, structured data, can be considered as a directly generalized model of relational databases. A higher leveled structured data, in our generalization, are defined recursively as a set or a queue of lower leveled structured data. Consequently, our study proves that many concepts and results of relational database management can be transferred to structured data, accordingly to this generalization. The sub data, the components of structured data, the functional dependencies between structured data, as well as the keys data in structured data are defined and studied. Alternately, some concepts that are defined here for structured data can be applied for relational databases, as a special case. In this paper, some operations on structured data and the homomorphism between structured data are defined and studied that appear to be quite suitable for relational databases. In fact, the formalization introduced here, offers effective methods for further structural, algebraic researches of structured data.

Keywords: Data management; Big Data; Structured data; Relational database; Lattice; Partially ordered set

1 Introduction

Big Data storage and Big Data processing model design are essential problems of Big Data management (see, for example, [9]). Structured data in a common sense, are complex data, constructed by atomic data residing in fixed fields within a definite structure. In contrast to semi-structured data and unstructured data,

structured data is taking new and special roles in the world of Big Data. Spreadsheets and relational databases are evident examples of structured data. A data table in relational databases as defined by Codd [4] in fact can be considered as a set of its columns (or its rows) that in their turn are the queues of atomic data. By using the characteristics of the data structure, lots of properties of relational databases were discovered and vigorously studied in 1990s ([1], [10]). The studies were focused on keys, functional dependencies between attributes and normalization of databases (see [2], [7]). The lattice-type properties of functional dependencies in relational databases were studied thoroughly in [8]. We can note that many properties of the functional dependencies between attributes are induced by the lattice structure of the data.

This remark inspires an idea: not only for relational databases but in general, it is the structure of data that determines the dependencies between their components and other structural properties of them. Thus the first question to be considered is the definition of the concept of structure. Structure is an indefinite concept and one can hardly give a sufficient definition that concerns all possible structures. This paper deals only with those structured data that are built up recursively as *sets*, or *queues* of other less complex data. Thus, the data can be defined in different orders of complexity: atomic data are structured data of lowest order, a set or a queue of atomic data is structured data of first order, while the relations in relational databases being sets of data columns (or data rows) are structured data of second order, etc. This definition does not cover all structured data, but it deals with rather wide range of data. The relational databases as sets of interconnected relations are in fact 4-order data.

The formulation of structured data proposes an efficient approach for structural studies. On one hand, the approach reveals the lattice characteristics of the well-known properties of relational databases. On the other hand, the approach enables the generalization of the concepts and properties, well-known for relational databases, into those of structured data.

In Section 2 we generalize the concept of relations in structured data as defined later. It is pointed out in this section, that there is a natural order between the relations and all relations can be represented in a linear form or by tree graphs. In Section 3, structured data are defined. In fact, structured data are generalized relations with all their participant relations. Structured data may be considered as algebraic objects in which the various operations and homomorphism should be studied. The concept of sub data and components of data, as well as the queries on data, are also defined here. In this generalized model we study the dependency between components of data. The key components of structured data are defined as those components, that all other components, depend. We show in this section that relational databases are really special cases of structured data, where the dependencies between attributes, are in fact, dependencies of partial order types. Some aspects for further research, as well as open problems are discussed in the Conclusions.

2 Relations

In this section we propose a generalization of the concept of a relation. Relations are defined recursively accordingly to their order. The first-order relation on a set is a collection of subsets or queues of elements of the given set, while the higher order relations are collection of subsets or queues of lower order relations. Thus relations are defined based on subsets or queues of elements.

2.1 Sets and Queues of Atomic Data

By traditional algebraic definition the relation of elements is a set of n-tuples of elements. In a more generalized sense, a relation of elements can be understood as a set of finite tuples of elements.

Definition 1: For a set V, let $V^{\infty} = \bigcup_{i=1}^{\infty} V^i$. V^{∞} denotes the set of all finite queues of elements in V.

Remark:

1. Below, we should distinguish the sets and the queues of elements: the sets and the queues of elements are parenthesized by {} and by < >, respectively.
2. By Definition 1, in general, $\{u, v\} = \{v, u\}$ and $\{v, v\} = \{v\}$, while $\langle u, v\rangle \neq \langle v, u\rangle$ and $\langle v, v\rangle \neq \langle v\rangle$
3. We accept $\{v\} = \langle v\rangle$

Definition 2:

Let U, V be two sets of atomic data, $U \subseteq V$. For $s \subseteq V^{\infty}$ the *projection* of s denoted by $Pr_U(s)$ is defined as follows:

i. If $s = \langle v_1, v_2, \dots, v_k\rangle \in V^{\infty}$, then $Pr_U(s) = \langle v_i | v_i \in U\rangle$.

ii. If $s = \{s_1, s_2, \dots, s_q\} \subseteq V^{\infty}$ then $Pr_U(s) = \{Pr_U(s_1), Pr_U(s_2), \dots, Pr_U(s_q)\}$.

iii. If $s = \langle v_1, v_2, \dots, v_k\rangle \in V^{\infty}$ or $s = \{s_1, s_2, \dots, s_q\} \subseteq V^{\infty}$ then $A(s)$ denotes the set of all atomic data in V that appears in s.

Remark:

1. If $r = Pr_U(s)$ then r is a sub-queue of s in the case s is a queue, and r is a set of sub-queues of s in the case s is a set of sub-queues.
2. If $r = Pr_U(s)$ and $s = Pr_W(t)$ then $r = Pr_{U\cap W}(t)$.
3. If r, s are finite subsets of V^{∞} and $r = Pr_U(s)$, $s = Pr_W(r)$ then $s = r$ and $A(s) = A(r) \subseteq U \cap W$.

   This is evident if s, r are queues in V^{∞}.

If $r, s \subseteq V^{\infty}$, $r = \{r_1, r_2, \dots, r_p\}, s = \{s_1, s_2, \dots, s_q\}$, then by definitions for all k there is i such that $r_k = Pr_U(s_i)$ and vice versa, for all i there is k such that $s_i = Pr_W(r_k)$. Thus, for all k there are $r_{k_1} = r_k, r_{k_2}, \dots$ and $s_{k_1}, s_{k_2}, \dots$ such that $r_{k_j} = Pr_U\left(s_{k_j}\right)$ and $s_{k_j} = Pr_W\left(r_{k_{j+1}}\right)$, $j = 1,2, \dots$. By previous remarks one can see that $r_{k_j} = Pr_{U\cap W}\left(r_{k_{j+1}}\right)$. Since $r_{k_{j+1}} = Pr_{U\cap W}\left(r_{k_{j+2}}\right)$, we have $r_{k_j} = r_{k_{j+1}} = Pr_{U\cap W}\left(r_{k_{j+2}}\right)$. Now it is easy to see that $r_k = r_{k_1} = r_{k_2} = \cdots = s_{k_1} = s_{k_2} = \cdots$, i.e. r_k is a queue in s. The similar explain proves that arbitrary queue s_i in s is queue in r. We have $r = s$. By the proof we see also that $A(s) = A(r) \subseteq U \cap W$.

4. By 2. and 3. if we define a relation on finite subsets of V^{∞} as follows:

   $r \lhd s \Leftrightarrow \exists U: r = Pr_U(s)$

   then $\lhd$ is a partial order on the finite subsets of V^{∞}.

2.2 m-Order Relations

The relations that we define below are a generalization of the relations (spreadsheets) in relational modeling.

Definition 3:

1. A *0-order relation* over V is V. The set of all 0-order relations over V is denoted by $\mathcal{R}^0(V)$. Thus $\mathcal{R}^0(V) = V$.
2. For $m \geq 0$ an *(m+1)-order relation* over V is a finite subset of $\mathcal{R}^{\mathrm{m}}(V)$ or a finite queue of elements of $\mathcal{R}^{\mathrm{m}}(V)$. The set of all ($m$+1)-order relations over V is denoted by $\mathcal{R}^{\mathrm{m}+1}(V)$.
3. $\mathcal{R}^{\infty}(V) = \bigcup_{\mathrm{m}=0}^{\infty} \mathcal{R}^{\mathrm{m}}(V)$.

In words, a relation of higher order over V is a finite subset or a finite queue of lower order relations. $\mathcal{R}^{\infty}(V)$ denotes the set of all relations over V.

Remark:

1. By Definition 1 we have $v = \{v\} = \langle v \rangle$, therefore $V \subseteq \mathcal{R}^1(V)$ and so on, $\mathcal{R}^m(V) \subseteq \mathcal{R}^{m+1}(V)$ for all $m \geq 0$.
2. Two relations of different order are different and are named differently, exceptionally, since $r = \{r\} = \langle r \rangle$ for all $r \in \mathcal{R}^m(V)$, we have also $r \in \mathcal{R}^{m+1}(V)$.

Definition 4:

Let $U \subseteq V$. For $s \in \mathcal{R}^{\infty}(V)$ the *projection* of s denoted by $Pr_U(s)$ is defined as follows:

i. If $s \in \mathcal{R}^1(V)$ then $Pr_U(s)$ is defined as in Definition 2.

ii. If $s = \langle v_1, v_2, \dots, v_k \rangle \in \mathcal{R}^{\mathrm{m}+1}(V)$, $v_i \in \mathcal{R}^{\mathrm{m}}(V)$, then $Pr_U(s) = \langle Pr_U(v_i) | \mathrm{i} = 1,2, \dots, \mathrm{k} \rangle$.

iii. If $s = \{s_1, s_2, \dots, s_q\} \in \mathcal{R}^{\mathrm{m}+1}(V)$ where $s_1, s_2, \dots, s_q$ are queues on $\mathcal{R}^{\mathrm{m}}(V)$, then $Pr_U(s) = \{Pr_U(s_1), Pr_U(s_2), \dots, Pr_U(s_q)\}$.

The projection of a relation on $U \subseteq V$ can be obtained from the given relation by deleting all atomic data that are not in U.

2.3 Partial Order on Relations

We show that on $\mathcal{R}^{\infty}(V)$ there exists a partial order between the relations.

Definition 5: For $r, s \in \mathcal{R}^{\infty}(V)$ we write $r \leq s$ if

i. $r = s$, or

ii. There exist $s_0, s_2, \dots, s_k \in \mathcal{R}^{\infty}(V)$, $s_0 = r$, $s_{\mathrm{k}} = s$, such that s_{i} is a finite subset or a finite queue of $s_{\mathrm{i}-1}$ and other elements of $\mathcal{R}^{\infty}(V)$, for all $i = 1,2, \dots, k$.

In words, $r \leq s$ if r appears in the presentation of s. We have a trivial theorem:

Theorem 1: $\leq$ is a partial order on $\mathcal{R}^{\infty}(V)$.

2.4 Representation of Relations

The relations can be represented by linear expressions and by tree graphs.

Definition 6 (linear representations of relations):

i. For $r \in \mathcal{R}^0(V)$, $r = v \in V$ the linear representation of r is the expression $l(r) = v$.

ii. For $r \in \mathcal{R}^{m+1}(V)$ the linear representation of r is the expression $l(r)$:

$$l(r) = \begin{cases} \langle l(r_1), l(r_2), \dots, l(r_k) \rangle \text{ if } r = \langle r_1, r_2, \dots, r_{\mathrm{k}} \rangle, r_{\mathrm{i}} \in \mathcal{R}^m(V) \\ \{l(r_1), l(r_2), \dots, l(r_{\mathrm{k}})\} \text{ if } r = \{r_1, r_2, \dots, r_{\mathrm{k}}\}, r_{\mathrm{i}} \in \mathcal{R}^m(V) \end{cases}$$

The set of all representations of on r is denoted by $\mathcal{P}(r)$.

In fact, the representations on V are the expressions that can be defined, formally, as follows:

Definition 7:

i. If $v \in V$ then the expression v is a formal representation. The set of all expressions of this form is denoted by $\mathcal{P}^0(V)$.

ii. If $r_1, r_2, \dots, r_k \in \mathcal{P}^i(V), i \leq m$, then the expressions of the form $\{r_1, r_2, \dots, r_k\}$ and $\langle r_1, r_2, \dots, r_k \rangle$ are formal representations on V. The set of all expressions of this form is denoted by $\mathcal{P}^{m+1}(V)$.

iii. All formal representations on V are defined as in i. and ii.

The set of all formal representations on V is denoted by $\mathcal{P}(V)$, i.e.

$\mathcal{P}(V) = \bigcup_{m=1}^{\infty} \mathcal{P}^m(V)$.

Remark:

1. The representation of relations is not unique: each relation has many representations.
2. The linear representations of relations over V are formal representations on V and vice versa, each formal representation on V represents some relation over V.

For $p, q \in \mathcal{P}$ we write $p \sim q$ if:

i. $p = \{q\}$ or $p = \langle q \rangle$, or $q = \{p\}$ or $q = \langle p \rangle$,

ii. $p = \{u_1, u_2, \dots, u_k\}$, $q = \{v_1, v_2, \dots, v_k\}$, $u_\mathrm{i}, v_\mathrm{i} \in V$ and $v_1, v_2, \dots, v_k$ is a permutation of $u_1, u_2, \dots, u_k$, or

iii. $p = \langle u_1, u_2, \dots, u_k \rangle$, $u_\mathrm{i} \in V$ and $q = p$, or

iv. $p = \{r_1, r_2, \dots, r_m\}$, $q = \{s_1, s_2, \dots, s_m\}$, $r_\mathrm{i}, s_\mathrm{i}$ are formal representations on V and there exists a permutation $u_1, u_2, \dots, u_m$ of $r_1, r_2, \dots, r_m$ such that $u_i \sim s_i$ for all $i = 1,2,..,m$.

v. $p = \langle r_1, r_2, \dots, r_m \rangle$, $q = \langle s_1, s_2, \dots, s_m \rangle$ and $r_i \sim s_i$ for all $i = 1,2,..,m$.

Theorem 2:

1. $\sim$ is an equivalence on $\mathcal{P}$.
2. $p \sim q \Leftrightarrow \exists r \in \mathcal{R}^\infty(V)$: $p, q \in \mathcal{P}(r)$.
3. There exists an algorithm that constructs $r \in \mathcal{R}^\infty(V)$ such that $l(r) = p$ for $p \in \mathcal{P}$.
4. There exists an algorithm that decides if $p \sim q$ for $p, q \in \mathcal{P}$.

We define a partial order on $\mathcal{P}$: For $p, q \in \mathcal{P}$ we write $p \leq q$ if

i. $p = q$, or

ii. There exist $p_1, p_2, \dots, p_k \in \mathcal{P}(V)$, $p_1 = p$, $p_\mathrm{k} = q$, such that $p_\mathrm{i} \in \mathcal{P}^{m+i}(V)$ and $p_{\mathrm{i}+1} = p_\mathrm{i}$ or $p_{\mathrm{i}+1}$ is the expression of the form $\{\dots, p_\mathrm{i}, \dots\}$ or $\langle \dots, p_\mathrm{i}, \dots \rangle$ for all $i = 1, \dots, k-1$.

$\leq$ is a partial order on $\mathcal{P}$. We have:

Theorem 3: For all $r, s \in \mathcal{R}^\infty(V)$ we have:

$r \leq s \Leftrightarrow l(r) \leq \mathrm{l}(s)$

The relations can also be represented by tree graphs as follows:

Definition 8 (graphical representation of relations): To each relation r we associate a tree graph $t(r)$ as follows:

i. For $r = v \in \mathcal{R}^0(V)$ the graph $t(v)$ is the tree graph that contains single node labeled by v.

ii. For $r \in \mathcal{R}^{m+1}(V), r = \langle r_1, r_2, \dots, r_k \rangle, r_i \in \mathcal{R}^m(V)$ the graph $t(r)$ is the tree graph with the root r that is connected directly to the nodes to that we attach the trees $t(r_1), t(r_2), \dots, t(r_k)$ from left to right.

iii. For $r \in \mathcal{R}^{m+1}(V), r = \{r_1, r_2, \dots, r_k\}, r_i \in \mathcal{R}^m(V)$, the graph $t(r)$ is the tree graph with the root r that is connected directly to the nodes to that we attach the trees $t(r_1), t(r_2), \dots, t(r_k)$ for $i = 1,2 \dots k$.

The set of all graphs representing r is denoted by $\mathcal{T}(r)$. We have:

Theorem 4:

1. If $\mathcal{T} = \mathcal{T}(r)$ is a tree that represents r, then
   i. The leaves are labeled by elements of V.
   ii. Only the labels of the leaves may be repeated.
2. If $\mathcal{T}$ is a tree graph with labeled nodes that satisfies i, ii conditions, then there exists a relation r on V such that $\mathcal{T} = \mathcal{T}(r)$.
3. For two relations $r, s \in \mathcal{R}^{\infty}(V)$ $r \leq s$ if and only if $\mathcal{T}(r)$ is a sub-tree of $\mathcal{T}(s)$.

Example 1:

A data table, i.e. a relation in relational database, may be considered as a relation defined in Definition 3: If $r = \{A_1, A_2, \dots, A_m\}$ is a relation with $A_1, A_2, \dots, A_m$ columns, where $A_i = (a_{1i}, a_{2i}, \dots, a_{ni})$ then r is 2-order relation

$$r = \{\langle a_{11}, a_{21}, \dots, a_{n1} \rangle, \langle a_{12}, a_{22}, \dots, a_{n2} \rangle, \dots, \langle a_{1m}, a_{2m}, \dots, a_{nm} \rangle\}$$

The tree graph of r is

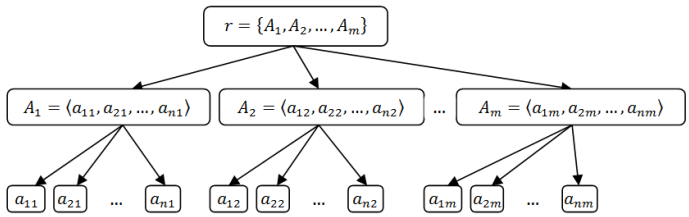

Figure 1
Tree graph of a relation in a relational database

3 Structured Data

Structured data are sets of relations with specified systems of participant relations. Formally, we have:

Definition 9: Let V be a set of atomic data.

1. A structured data is a finite sequence of relations

$$\alpha = [r_0, r_1, \dots, r_m]$$

where

i. r_0 is finite subset of V,

ii. For all $i = 1, \dots, m$, r_i is a relation constructed by atomic data in r_0 and preceding data, i.e. r_i is a finite set or a finite queue of elements from $r_0 \cup \{r_1, r_2, \dots, r_{i-1}\}$.

The set of all structured data over V is denoted by S_V.

2. A structured data $\alpha = [r_0, r_1, \dots, r_m]$ is *simple* if

iii. $r_j \neq r_i$ for $i \neq j$ and r_j, r_i are not elements from r_0

In a simple structured data the condition iii. guarantees that only elements from r_0 may be repeated.

Definition 10: Let $\alpha = [r_0, r_1, \dots, r_m]$ be a structured data.

1. By $l(\alpha) = [l(r_0), l(r_1), \dots, l(r_m)]$ we denote the linear representation of α where $l(r_i)$ is some linear representation of r_i for all $i = 1, \dots, m$.

2. By $t(\alpha) = \{t(r_0), t(r_1), \dots, t(r_m)\}$ we denote the multi tree of α which is constructed as follows:

i. $t(r_0)$ contains the nodes marked by elements in r_0,

ii. $t(r_i)$ is the tree of r_i for all $i = 1, \dots, m$.

Example 2: In Table 1 we can see a linear representation of a structured data:

Table 1
Linear representation of a structured data

$\alpha = [r_0, r_1, r_2, r_3, r_4, r_5]$	$l(\alpha) = [l(r_0), l(r_1), l(r_2), l(r_3), l(r_4), l(r_5)]$
$r_0 = \{v_1, v_2, v_3, v_4\}$	$l(r_0) = \{v_1, v_2, v_3, v_4\}$
$r_1 = \langle v_1, v_2 \rangle$	$l(r_1) = \langle v_1, v_2 \rangle$
$r_2 = \langle v_2, v_3 \rangle$	$l(r_2) = \langle v_2, v_3 \rangle$
$r_3 = \{v_1, r_1, r_2\}$	$l(r_3) = \{v_1, \langle v_1, v_2 \rangle, \langle v_2, v_3 \rangle\}$
$r_4 = \{v_2, \langle v_4, r_3 \rangle\}$	$l(r_4) = \{v_2, \langle v_4, \{v_1, \langle v_1, v_2 \rangle, \langle v_2, v_3 \rangle\}\}$
$r_5 = \{r_4, \langle r_1, r_3, r_4 \rangle\}$	$l(r_5) = \{\{v_2, \langle v_4, \{v_1, \langle v_1, v_2 \rangle, \langle v_2, v_3 \rangle\}\}, \langle \langle v_1, v_2 \rangle, \{v_1, \langle v_1, v_2 \rangle, \langle v_2, v_3 \rangle\}, \{v_2, \langle v_4, \{v_1, \langle v_1, v_2 \rangle, \langle v_2, v_3$

Example 3: Let **Atomic data =** {N, BD, Na, NID, NV, NC, NM, T, C, NDL} where N, BD, Na, NID, NV, NC, NM, T, C, NDL is the abbreviation of Name, Birth date, Nationality, Number of ID card, Number of Vehicle registration card, Number of chassis, Number of motor, Type, Category of vehicle and Number of Driving license, respectively. Furthermore, let **ID card** = <NID, N, BD, Na>, **Vehicle registration card** = <NV, NC, NM, T, C>, **Driving license** = <NDL, N, BD, C>, **Personal Document** = {**ID card, Vehicle registration card, Driving license**}.

In fact, **Personal Document** may be considered as a structured data: **Personal Document** = [**Atomic data, ID card, Vehicle registration card, Driving license,** {**ID card, Vehicle registration card, Driving license**}].

The tree graph of **Personal Document** is t(**Personal Document**):

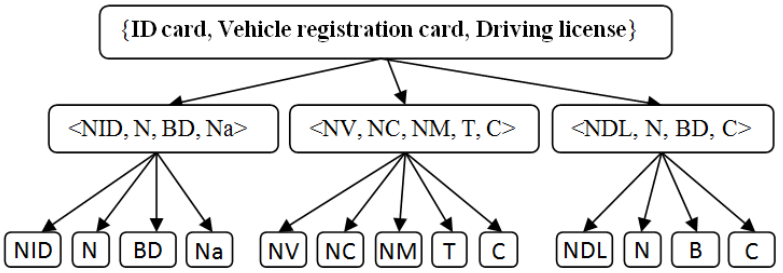

Figure 2
The tree graph of **Personal Document**

3.1 Operations of Structured Data

Let $\alpha = [r_0, r_1, \dots, r_m]$, $\beta = [s_0, s_1, \dots, s_n]$ be structured data over V. Then:

1. **Union:** The union of α, β is

   $\{\alpha, \beta\} = [r_0 \cup s_0, r_1, \dots, r_m, s_1, \dots, s_n, \{r_m, s_n\}]$

   The union of two structured data is also a structured data.

2. **Queuing :** The queuing of α, β is

   $\langle \alpha, \beta \rangle = [r_0 \cup s_0, r_1, \dots, r_m, s_1, \dots, s_n, \langle r_m, s_n \rangle]$

   The queuing of two structured data is also a structured data.

3. **Projection:** If $U \subseteq V$, then the projection of α on U is

   $Pr_U(\alpha) = [Pr_U(r_0), Pr_U(r_1), \dots, Pr_U(r_m)]$

   The projection of a structured data is also a structured data.

4. **Conjunction:** Let $\alpha_i = \left[r_0^i, r_1^i, \dots, r_{m_i}^i\right]$ be structured data over V for all $i = 1, \dots, k$, $r_0 = \cup_i r_0^i$. If r is a finite set of queues on $r_0 \cup \left\{r_j^i | j = 1, \dots, m_i, i = 1, \dots, k\right\}$, i.e. $r \in \mathcal{R}\left(r_0 \cup \left\{r_j^i | j = 1, \dots, m_i, i = 1, \dots, k\right\}\right)$, then the *conjunction* of α_i's by r is

$$r(\alpha_1, \alpha_2, \dots, \alpha_k) = \left[r_0, r_1^1, r_2^1, \dots, r_{m_1}^1, r_1^2, r_2^2, \dots, r_{m_2}^2, \dots, r_1^k, r_2^k, \dots, r_{m_k}^k, r\right]$$

One can see that the conjunction of structured data is also a structured data. The union and the queuing operations are special cases of the conjunction.

3.2 Homomorphism, Isomorphism between Structured Data

Let V, W be two sets of atomic data, $\varphi: V \to W$ and $r \in \mathcal{R}^{\mathrm{m}}(V)$. Then the homomorphic image of r is defined as follows:

Definition 11: Let V, W be two sets of atomic data, $\varphi: V \to W$ and $r \in \mathcal{R}^{\mathrm{m}}(V)$.

i. If $r \in \mathcal{R}^1(\mathcal{A}) = \mathcal{R}(V)$, i.e. r is finite set of queues from V, $r = \{r_1, \dots, r_k\}$, where $r_i = \langle r_1^i, r_2^i, \dots, r_{m_i}^i \rangle$, $r_{\mathrm{j}}^{\mathrm{i}} \in V$, then

$$\varphi(r) = \left\{\langle \varphi(r_1^1), \varphi(r_2^1), \dots, \varphi\left(r_{m_1}^1\right)\rangle, , \dots, \langle \varphi(r_1^k), \varphi(r_2^k), \dots, \varphi\left(r_{m_k}^k\right)\rangle\right\}$$

ii. Suppose that $\varphi(s)$ has been defined for all $s \in \mathcal{R}^{\mathrm{m}}(V)$. If $r \in \mathcal{R}^{\mathrm{m}+1}(V) = \mathcal{R}\left(\mathcal{R}^{\mathrm{m}}(V)\right), r = \{r_1, \dots, r_k\}$, where $r_i = \langle r_1^i, r_2^i, \dots, r_{m_i}^i \rangle$, $r_{\mathrm{j}}^{\mathrm{i}} \in \mathcal{R}^{\mathrm{m}}(V)$, then

$$\varphi(r) = \left\{\langle \varphi(r_1^1), \varphi(r_2^1), \dots, \varphi\left(r_{m_1}^1\right)\rangle, , \dots, \langle \varphi(r_1^k), \varphi(r_2^k), \dots, \varphi\left(r_{m_k}^k\right)\rangle\right\}$$

$\varphi(r)$ is the homomorphic image of r through φ.

Remark: Let $\varphi: V \to W$ and let $\alpha = [r_0, r_1, \dots, r_m] \in S_V$ be a structured data over V. It can be verified that $\varphi(\alpha) = [\varphi(r_0), \varphi(r_1), \dots, \varphi(r_m)]$ is also a structured data over W:

i. $\varphi(r_0) = \{\varphi(a) | a \in V\}$ is a finite set over W.

ii. For $i = 1,2, \dots, m$ we have $r_{\mathrm{i}} \in \mathcal{R}(r_0 \cup \{r_1, r_2, \dots, r_{i-1}\})$, i.e. $r_i = \{u_1, u_2, \dots, u_k\}$, where $u_j = \langle u_1^j, u_2^j, \dots, u_{m_j}^j \rangle$, $u_{\mathrm{t}}^{\mathrm{j}} \in r_0 \cup \{r_1, r_2, \dots, r_{i-1}\}$. By Definition 11 $\varphi(r_i) = \{\varphi(u_1), \varphi(u_2), \dots, \varphi(u_k)\}$, where $\varphi\left(u_j\right) = \langle \varphi(u_1^j), \varphi(u_2^j), \dots, \varphi(u_{m_j}^j)\rangle$.

Since $\varphi(u_t^j) \in \varphi(r_0) \cup \{\varphi(r_1), \varphi(r_2), \dots, \varphi(r_{i-1})\}$, we have:

$\varphi(r_i) \in \mathcal{R}(\varphi(r_0) \cup \{\varphi(r_1), \varphi(r_2), ..., \varphi(r_{i-1})\})$, i.e. $[\varphi(r_0), \varphi(r_1), \dots, \varphi(r_m)]$ is a structured data over W

Definition 12:

Let $\varphi: V \to W$ and $\alpha = [r_0, r_1, \dots, r_m] \in S_V$.

1. The *homomorphic image* of α by φ is $\varphi(\alpha) = [\varphi(r_0), \varphi(r_1), \dots, \varphi(r_m)]$.
2. If φ is bijective then $\varphi(\alpha)$ is the *isomorphic image* of α by φ.

By the previous remark we can see that the homomorphic image of a structured data is also a structured data. In other words, $\varphi: V \to W$ can be extended into $\varphi: S_V \to S_W$. We have:

Theorem 5: Let $\varphi: V \to W$. Then

1. For all $r \in \mathcal{R}^{\mathrm{m}}(V)$ we have $\varphi\big(l(r)\big) = l(\varphi(r))$
2. For all $\alpha \in S_V$ we have $\varphi\big(l(\alpha)\big) = l(\varphi(\alpha))$

Example 4:

Let $\varphi: V \to W$, where $V = \{x_1, x_2, x_3, x_4\}$, $W = \{a, b, c, d\}$ and $\varphi(x_1) = a$, $\varphi(x_2) = b$, $\varphi(x_3) = \varphi(x_4) = c$. Then

Table 2
Homomorphic image of a structured data

$\alpha = [r_0, r_1, r_2, r_3, r_4, r_5]$	$\varphi(\alpha) = [s_0, s_1, s_2, s_3, s_4, s_5]$
$r_0 = \{x_1, x_2, x_3, x_4\}$	$s_0 = \varphi(r_0) = \{a, b, c\}$
$r_1 = \langle x_1, x_2 \rangle$	$s_1 = \varphi(r_1) = \langle a, b \rangle$
$r_2 = \langle x_2, x_3 \rangle$	$s_2 = \varphi(r_2) = \langle b, c \rangle$
$r_3 = \{x_1, \quad r_1, \quad r_2\}$	$s_3 = \varphi(r_3) = \{a, s_1, s_2\}$
$r_4 = \{x_2, \langle x_4, r_3 \rangle\}$	$s_4 = \varphi(r_4) = \{b, \langle c, s_3 \rangle\} = \{b, \langle c, \{a, \langle a, b \rangle, \langle b, c \rangle\}\rangle\}$
$r_5 = \{r_4, \langle r_1, r_3, r_4 \rangle\}$	$s_5 = \varphi(r_5) = \{s_4, \langle \langle a, b \rangle, s_3, s_4 \rangle\}$

Remark: There exists $\varphi: V \to W, r, s \in \mathcal{R}^{\mathrm{m}}(V)$ such that:

1. $\varphi\big(Pr_U(\mathrm{s})\big) \neq Pr_{\varphi(U)}\big(\varphi(s)\big)$
2. $r \lhd s$, but $\varphi(r) \not\lhd \varphi(s)$

Let $V = \{x, y, u\}$, $U = \{y, u\}$, $\varphi(x) = \varphi(y) = \mathrm{a}$, $\varphi(u) = b$. Then $\varphi(U) = \{a, b\}$. For $r = \{u, \langle y, u \rangle\}$, $s = \{x, \langle x, u \rangle, \langle y, u \rangle\}$, we see $r = Pr_U(s)$, therefore $r \lhd s$. Moreover, $\varphi(r) = \varphi\big(Pr_U(\mathrm{s})\big) = \{b, \langle a, b \rangle\}$ and $\varphi(s) = Pr_{\varphi(U)}\big(\varphi(s)\big) = \{a, \langle a, b \rangle\}$. One can see $\varphi\big(Pr_U(\mathrm{s})\big) \neq Pr_{\varphi(U)}\big(\varphi(s)\big)$ and $\varphi(r) \not\lhd \varphi(s)$.

3.3 Sub-Data

Below we define sub-data of the given structured data. In a sense, sub-data are not simply parts of structured data, but inherit the given structure.

Definition 13:

For two structured data $\alpha = [r_0, r_1, \dots, r_m]$, $\beta = [s_0, s_1, \dots, s_n]$ over V we say that α is a sub-data of β if:

i. $r_0 \subseteq s_0$, and

ii. $\forall i \geq 1 \ \exists j \geq 1 : r_i = s_j$

Then we write $\alpha \sqsubseteq \beta$. The set of all sub-data of β is denoted by $SUBD(\beta)$.

Example 5:

Let $\alpha = [r_0, r_1, r_2, r_3], \beta = [s_0, s_1, s_2, s_3, s_4]$ where $s_0 = \{x_1, x_2, x_3, x_4\}$, $s_1 = \langle x_2, x_4 \rangle$, $s_2 = \{x_1, s_1\} = \{x_1, \langle x_2, x_4 \rangle\}$, $s_3 = \langle x_3, x_4 \rangle$, $s_4 = \{s_2, s_3\} = \{\{x_1, \langle x_2, x_4 \rangle\}, \langle x_3, x_4 \rangle\}$ and $r_0 = \{x_1, x_2, x_4\}$, $r_1 = s_1 = \langle x_2, x_4 \rangle$, $r_2 = s_2 = \{x_1, \langle x_2, x_4 \rangle\}$. One can see that $\alpha \sqsubseteq \beta$.

It is evident that the relation $\sqsubseteq$ defined in Definition 13 is a partial order on the set of structured data.

3.4 Components of Structured Data

Not all sub-data of structured data are its components. The components of structured data are all those sub-data that are maximal in a sense.

Definition 14:

1. Let $\alpha, \beta \in S_V$ be structured data. We say that α is a *component* of β if
   i. $\alpha \sqsubseteq \beta$,
   ii. There is no structured data $\gamma \in S_V$ such that $\alpha \sqsubseteq \gamma, \gamma \sqsubseteq \beta$, and $\gamma \neq \alpha, \gamma \neq \beta$.

The set of all components of a structured data β is denoted by $COMP(\beta)$.

2. Let $\alpha_1, \alpha_2, \dots, \alpha_n \in COMP(\beta)$. We say that $\{\alpha_1, \alpha_2, \dots, \alpha_n\}$ is an *adequate set of components* of β if $\beta = \alpha_1 \cup \alpha_1 \cup \dots \cup \alpha_n$.
3. We say that $\{\alpha_1, \alpha_2, \dots, \alpha_n\}$ is a *minimal adequate set of components* (MASC) of β if:
   i. $\{\alpha_1, \alpha_2, \dots, \alpha_n\}$ is an adequate set of components of β,
   ii. There is no real subset of $\{\alpha_1, \alpha_1, \dots, \alpha_n\}$ that is also adequate set of components of β.

In other words, a component of a structured data is some it's sub-data that is maximal in the partial order defined by $\sqsubseteq$.

Example 6:

Let $\beta = [\{a, b, c\}, \{b, c\}, \langle a, \{b, c\} \rangle, \{a, c\}, \langle b, \{c, a\} \rangle, \{\langle a, \{b, c\} \rangle, \langle b, \{c, a\} \rangle\}]$ and $\alpha = [\{a, b, c\}, \{b, c\}, \langle a, \{b, c\} \rangle]$. We can see that α is a component of β.

The following theorems are evident:

Theorem 6:

1. Let $\alpha, \beta \in S_V$ be structured data and $\alpha \sqsubseteq \beta$. Then there is $\gamma \in S_V$ such that $\alpha \sqsubseteq \gamma$ and γ is a component of β.
2. Every structured data has at least one component.
3. Every structured data has at least one MASC.

Theorem 7:

Let $\alpha \in S_X$ be structured data and $\varphi: \mathcal{A} \to \mathcal{B}$. If a set of structured data $\{\beta_1, \beta_1, \dots, \beta_n\}$ is a MASC of α, then $\{\varphi(\beta_1), \varphi(\beta_1), \dots, \varphi(\beta_n)\}$ is a MASC of $\varphi(\alpha)$.

3.5 Queries on Structured Data

Queries are operations that for a given set of data produce a set of data. In general, a simple query retrieves from a structured data some its sub-data. In this sense selections and projections in relational databases are such simple queries. Joins are queries, but are not simple queries.

Definition 15: Let $\mathcal{D} \subseteq S_V$ be a set of structured data over V.

1. A *query* over $\mathcal{D}$ is a mapping $q: \mathcal{D} \to \mathcal{D}$.
2. A query $q: \mathcal{D} \to \mathcal{D}$ is *proper* for $\alpha \in S_V$ if $q(\alpha) \sqsubseteq \alpha$.

   A query q is proper for $S \subseteq S_V$ if it is proper for all $\alpha \in S$.
3. Let $\mathcal{Q}$ be a set of queries over $\mathcal{D}$ and $\alpha \in S_V$ be a structured data over V. Then α is *minimal applicable data* for $\mathcal{Q}$ if:
   i. α is applicable for all query $q \in \mathcal{Q}$.
   ii. There is no $\beta \in S_V$ such that $\beta \sqsubseteq \alpha$ and β is applicable for all queries $q \in \mathcal{Q}$.

3.6 Dependency Types and Keys

In this section we propose a concept of dependency types and the dependencies between the sub-data and components defined accordingly to the given dependency types are studied. The idea is simple: structured data and their sub-data, as well as their components are associated to the elements of a "sample set" where the "sample dependencies" have been well defined. Thus, the "sample dependencies" in the "sample set" induce, on the set of structured data, sample-like dependencies. Our study is focused on the dependency types defined by the lattices with partial order. We show here that functional dependencies in relational databases are, in fact, partial order type (or lattice-type) dependencies. This approach reveals that most of properties of functional dependencies are inherited from the properties of the partial order on the "sample" lattice.

Let $\mathcal{L}$ be a set with the partial order $\preccurlyeq$, then, as usual, for $L \subseteq \mathcal{L}$ we denote $Sup(L) = \{l \in \mathcal{L} | \forall l' \in L: l' \preccurlyeq l\}$ and $Sup^*(L) = \{l \in Sup(L) | \forall l' \in \mathcal{L}: l' \preccurlyeq l \Rightarrow l' \notin Sup(L)\}$. Similarly, we denote $Inf(L) = \{l \in \mathcal{L} | \forall l' \in L: l \preccurlyeq l'\}$ and $Inf^*(L) = \{l \in Inf(L) | \forall l' \in \mathcal{L}: l \preccurlyeq l' \Rightarrow l' \notin Inf(L)\}$.

Definition 16:

1. Let $\mathcal{R} \subseteq \mathcal{R}^{\infty}(V)$ be a set of relations over V. By *dependency type* on $\mathcal{R}$ we understand a couple $(\mathcal{L}, \varphi)$ where $\mathcal{L}$ is a lattice with the partial order $\preccurlyeq$, $\varphi: \mathcal{R} \to \mathcal{L}$ is a mapping that satisfies the following conditions:
   i. For $r, s \in \mathcal{R}$ if $\varphi(r)$ are determined and $s \leq r$ then $\varphi(s)$ is determined and $\varphi(s) \preccurlyeq \varphi(r)$
   ii. For $r, s \in \mathcal{R}$, if $s = \langle s_1, s_2, \dots, s_n \rangle$, $r = \langle r_1, r_2, \dots, r_n \rangle$; $\varphi(s_i), \varphi(r_i)$ are determined and $\varphi(s_i) \preccurlyeq \varphi(r_i)$ for all $i = 1,2, \dots, n$, then $\varphi(s)$, $\varphi(r)$ are determined and $\varphi(s) \preccurlyeq \varphi(r)$.
   iii. For $r, s \in \mathcal{R}$, if $s = \{s_1, s_2, \dots, s_n\}$, $r = \{r_1, r_2, \dots, r_m\}$; $\varphi(s_i)$, $\varphi(r_j)$ are determined and for all $i = 1,2, \dots, n$ there exists $j = 1,2, \dots, m$ such that $\varphi(s_i) \preccurlyeq \varphi(r_j)$, then $\varphi(s)$, $\varphi(r)$ are determined and $\varphi(s) \preccurlyeq \varphi(r)$
2. For two relations $r, s \in \mathcal{R}$ we say that s *depends on* r in dependency type $(\mathcal{L}, \varphi)$ if $\varphi(s) \preccurlyeq \varphi(r)$. Then we write $r \underset{\mathcal{L},\varphi}{\longrightarrow} s$ or for simplicity $r \to s$ if $\mathcal{L}, \varphi$ are well known.
3. For two structured data $\alpha, \beta \in S_V$, $\alpha = [r_1, r_2, \dots, r_m]$, $\beta = [s_1, s_2, \dots, s_n]$, we say that β *depends on* α in dependency type $(\mathcal{L}, \varphi)$ if $\{r_1, r_2, \dots, r_m\} \to s_i$ for all $i = 1,2, \dots, n$. Then we write $\alpha \underset{\mathcal{L},\varphi}{\longrightarrow} \beta$ or for simplicity $\alpha \to \beta$ if $\mathcal{L}, \varphi$ are well known.

The dependencies defined in the Definition 16 are called partial order dependencies or lattice-like dependencies.

Theorem 8: Let $(\mathcal{L}, \varphi)$ be a dependency type, where $\mathcal{L}$ is a set with the partial order $\preccurlyeq$, $\varphi: \mathcal{R} \to \mathcal{L}$.

1. If $s = \langle s_1, s_2, \dots, s_n \rangle$, $\varphi(s_i)$ is determined for all $i = 1,2, \dots, n$, then $\varphi(s)$ is determined and
   $$\varphi(s) \in Sup^*\big(\varphi(s_1), \varphi(s_2), \dots, \varphi(s_n)\big)$$
2. If $s = \{s_1, s_2, \dots, s_n\}$, $\varphi(s_i)$ is determined for all $i = 1,2, \dots, n$, then $\varphi(s)$ is determined and
   $$\varphi(s) \in Sup^*\big(\varphi(s_1), \varphi(s_2), \dots, \varphi(s_n)\big)$$

Proof:

If $s = \langle s_1, s_2, \dots, s_n \rangle$ or $s = \{s_1, s_2, \dots, s_n\}$, $\varphi(s_i)$ is determined for all $i = 1,2, \dots, n$, then $\varphi(s)$ is determined. Moreover, $s_i \leq s$, therefore $\varphi(s_i) \preccurlyeq \varphi(s)$ for all $i = 1,2, \dots, n$, i.e. $\varphi(s) \in Sup\big(\varphi(s_1), \varphi(s_2), \dots, \varphi(s_n)\big)$.

Moreover, if $\varphi(s') \in Sup\big(\varphi(s_1), \varphi(s_2), \dots, \varphi(s_n)\big)$, then $\varphi(s_i) \preccurlyeq \varphi(s')$, for all $i = 1,2, \dots, n$.

1. By ii. in Definition 16, if $s = \langle s_1, s_2, \dots, s_n \rangle$ then by $s' = \langle s', s', \dots, s' \rangle$, we have $\varphi(s) \preccurlyeq \varphi(s')$. This proves that $\varphi(s) \in Sup^*\big(\varphi(s_1), \varphi(s_2), \dots, \varphi(s_n)\big)$.
2. By ii. in Definition 16, if $s = \{s_1, s_2, \dots, s_n\}$ then by $s' = \{s'\}$, we have $\varphi(s) \preccurlyeq \varphi(s')$. This proves that $\varphi(s) \in Sup^*\big(\varphi(s_1), \varphi(s_2), \dots, \varphi(s_n)\big)$

Theorem 9: The functional dependencies in relations of relational database are partial order dependencies.

Proof: In the Example 1 we have shown that every relation $r = \{A_1, A_2, \dots, A_m\}$ in relational database with the columns $A_i = \big(a_{ij}\big)$, $i = 1,2, \dots, m$, $j = 1,2, \dots, n$ can be considered as a 2-order relation:

$r = \{\langle a_{11}, a_{12}, \dots, a_{1n} \rangle, \langle a_{21}, a_{22}, \dots, a_{2n} \rangle, \dots, \langle a_{m1}, a_{m2}, \dots, a_{mn} \rangle\}$

In fact, r can be considered as a structured data

$r = [r_0, r_1, r_2, \dots, r_m, r_{m+1}]$

where $r_0 = \big\{a_{ij} | i = 1,2, \dots, m, j = 1,2, \dots, n\big\}$, $r_i = \langle a_{i1}, a_{i2}, \dots, a_{in} \rangle$ for $i = 1,2, \dots, m$ and $r_{m+1} = \{r_1, r_2, \dots, r_m\}$.

The functional dependency between the columns of r can be defined as follows: Let $\mathcal{L}$ the set of all partitions on the set $\{l_1, l_2, \dots, l_n\}$, where $l_1, l_2, \dots, l_n$ are the rows of r. We denote by $\preccurlyeq$ the following partial order on $\mathcal{L}$: for two partitions p, q on $l_1, l_2, \dots, l_n$ we write $p \preccurlyeq q$ if

$l_j \sim_q l_k \Rightarrow l_j \sim_p l_k$

To each a_{ij} we associate the partition $\varphi\big(a_{ij}\big)$ over $\{l_1, l_2, \dots, l_n\}$ defined by the following equivalence: $l_k \sim_{a_{ij}} l_t$ if and only if both l_k, l_t contain a_{ij} or both l_k, l_t do not contain a_{ij}.

To the column A_i we associate the partition $\varphi(A_i)$ over $\{l_1, l_2, \dots, l_n\}$ defined by the following equivalence: $l_k \sim_{A_i} l_t$ if and only if $a_{ik} = a_{it}$.

To a set of columns $\mathcal{A}$ we associate the partition $\varphi(\mathcal{A})$ over $\{l_1, l_2, \dots, l_n\}$ defined by the following equivalence: $l_j \sim_{\mathcal{A}} l_k$ if and only if $l_j \sim_{A_i} l_k$ for all $A_i \in \mathcal{A}$.

One can verify that φ satisfies the conditions in Definition 16, thus $(\mathcal{L}, \preccurlyeq)$ is partial order dependency type. It is easy to see that for two set of columns $\mathcal{A}, \mathcal{B}$ in r, $\mathcal{B}$ functionally depends on $\mathcal{A}$, i.e. $\mathcal{A} \to \mathcal{B}$, if and only if $\varphi(\mathcal{B}) \preccurlyeq \varphi(\mathcal{A})$.

One can verify that most of rules that hold for functional dependency in fact hold also for generalized model, i.e. for partial order dependency. We have:

Theorem 10: Let $(\mathcal{L}, \varphi)$ be a dependency type on $\mathcal{R} \subseteq \mathcal{R}^{\infty}(V)$. Let α, β, γ be relations (or structured data) over V. We have:

1. (Reflexivity) If $\beta \leq \alpha$, then $\alpha \to \beta$.
2. (Augmentation) If $\alpha \to \beta$, then $\alpha \cup \gamma \to \beta \cup \gamma$ for any γ.

   In the case α, β, γ are relations by $\alpha \cup \gamma$, $\beta \cup \gamma$ we understand $\{\alpha, \gamma\}$ and $\{\beta, \gamma\}$, respectively.
3. (Transitivity) If $\alpha \to \beta$, $\beta \to \gamma$, then $\alpha \to \gamma$.

Definition 17: For a structured data $\alpha \in S_V$, $\alpha = [r_1, r_2, \dots, r_m]$, we say that a set $\beta = \{r_{i_1}, r_{i_2}, \dots, r_{i_k}\}$ is a *key set* of α in dependency type $(\mathcal{L}, \varphi)$ if $\{r_{i_1}, r_{i_2}, \dots, r_{i_k}\} \underset{\mathcal{L},\varphi}{\longrightarrow} r_i$ for all $i = 1,2, \dots, m$.

The following example shows how a relation in relational database can be considered as structured data and how the functional dependencies in it can be studied as partial order dependencies.

Example 7: Let $\boldsymbol{r}$ be the relation in the Figure x, where $\boldsymbol{l_i}$s and $\boldsymbol{c_2}$ are the rows and the columns of $\boldsymbol{r}$, respectively.

Table 3

Relation $\boldsymbol{r}$ in relational database

	$\boldsymbol{c_1}$	$\boldsymbol{c_2}$	$\boldsymbol{c_3}$	$\boldsymbol{c_4}$
$\boldsymbol{l_1}$	x_1	y_1	z_1	w_1
$\boldsymbol{l_2}$	x_1	y_2	z_2	w_1
$\boldsymbol{l_3}$	x_2	y_2	z_3	w_1

$\boldsymbol{r}$ can be considered as a structured data $[\boldsymbol{r_0}, \boldsymbol{c_1}, \boldsymbol{c_2}, \boldsymbol{c_3}, \boldsymbol{c_4}, \boldsymbol{r}]$ where $\boldsymbol{r_0} = \{x_1, x_2, y_1, y_2, z_1, z_2, z_3, w_1\}$ is the set of all atomic data, $\boldsymbol{c_1} = \langle x_1, x_1, x_2 \rangle$, $\boldsymbol{c_2} = \langle y_1, y_2, y_2 \rangle$, $\boldsymbol{c_3} = \langle z_1, z_2, z_3 \rangle$, $\boldsymbol{c_4} = \langle w_1, w_1, w_1 \rangle$ are the columns of the relation $\boldsymbol{r} = \{\boldsymbol{c_1}, \boldsymbol{c_2}, \boldsymbol{c_3}, \boldsymbol{c_4}\}$.

Denote the rows of $\boldsymbol{r}$ by $\boldsymbol{l_1}, \boldsymbol{l_2}, \boldsymbol{l_3}$. The set of all partitions on $\{\boldsymbol{l_1}, \boldsymbol{l_2}, \boldsymbol{l_3}\}$ is denoted by $\mathcal{L}$. Every partition $p \in \mathcal{L}$ can be determined by the equivalence $\sim_p$: $\boldsymbol{l_i}, \boldsymbol{l_j}$ belong to a same class in p if and only if $\boldsymbol{l_i} \sim_p \boldsymbol{l_j}$. $\mathcal{L}$ is a lattice where the partial order on $\mathcal{L}$ is defined as usual: $p \preccurlyeq q$ if and only if $\boldsymbol{l_i} \sim_q \boldsymbol{l_j} \Rightarrow \boldsymbol{l_i} \sim_p \boldsymbol{l_j}$. The partial order on $\mathcal{L}$ is described in the following diagram:

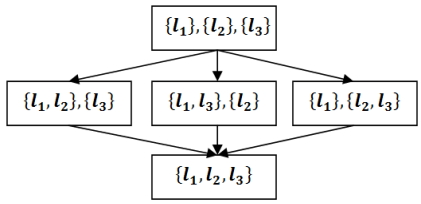

Figure 3
The partial order between the partitions

Each relation $\boldsymbol{x}$ in the structured data $\boldsymbol{r}$, $\boldsymbol{x} \in \{\boldsymbol{r_0}, \boldsymbol{c_1}, \boldsymbol{c_2}, \boldsymbol{c_3}, \boldsymbol{c_4}, \boldsymbol{r}\}$, accordingly to Theorem 9, can be associated to a partition $\varphi(\boldsymbol{x}) \in \mathcal{L}$ as follows:

Table 4
Relations in a structured data and their image in a partially ordered set

$\boldsymbol{x}$	$\varphi(\boldsymbol{x})$	$\boldsymbol{x}$	$\varphi(\boldsymbol{x})$	$\boldsymbol{x}$	$\varphi(\boldsymbol{x})$	$\boldsymbol{x}$	$\varphi(\boldsymbol{x})$
x_1	$\{l_1,l_2\},\{l_3\}$	y_2	$\{l_1\},\{l_2,l_3\}$	z_3	$\{l_1,l_2\},\{l_3\}$	$\boldsymbol{c_2}$	$\{l_1\},\{l_2,l_3\}$
x_2	$\{l_1,l_2\},\{l_3\}$	z_1	$\{l_1\},\{l_2,l_3\}$	w_1	$\{l_1,l_2,l_3\}$	$\boldsymbol{c_3}$	$\{l_1\},\{l_2\},\{l_3\}$
y_1	$\{l_1\},\{l_2,l_3\}$	z_2	$\{l_1,l_3\},\{l_2\}$	$\boldsymbol{c_1}$	$\{l_1,l_2\},\{l_3\}$	$\boldsymbol{c_4}$	$\{l_1,l_2,l_3\}$
						$\boldsymbol{r}$	$\{l_1\},\{l_2\},\{l_3\}$

Thus one can see the dependencies between the relations in the structured data $\boldsymbol{r}$:

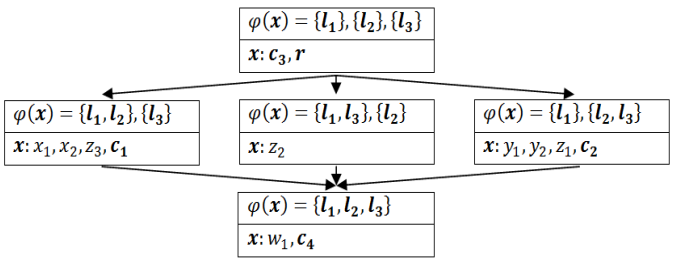

Figure 4
The dependencies between the relations in a structured data

Conclusions

In this paper we have proposed a formalization for structured data, in which data are constructed recursively by two basic structures, namely, by sets and queues, based upon atomic data. Although the approach may not deal with all structured data, it does touch on a large portion. The relations, relational databases can be handled in this formalization. We show that many well-known concepts and results in relational databases, such as keys and functional dependencies, can be studied in this generalized model of data. The generalization has certain advantages: the concepts and results in relational databases are quite clear in this formalization, the properties of keys and functional dependencies are inherited from the sample hierarchy in a lattice, etc. Moreover, the proposed approach also offers a unique method for managing different operations on structured data.

As one can see, the approach poses several problems that may be interesting topics for further studies. These problems are:

- The operations on structured data should be studied more thoroughly, including the composition and decomposition of structured data.
- The relational algebra should be developed for structured data.
- An optimization and normalization of structured data should be studied that guarantees the optimality and consistency of structured data management systems.

References

[1] Békéssy, J. Demetrovics: Contribution to the Theory of Data Base Relations, Discrete Math., 27, 1979, pp. 1-10

[2] C. Beeri, M. Dowd, R. Fagin, R. Statman: On the Structure of Armstrong Relations for Functional Dependencies, J. ACM, 31, 1984, pp. 30-46

[3] Brigitte Le Roux, Henry Rouanet: Geometric Data Analysis: From Correspondence Analysis to Structured Data Analysis, Kluwer Academic Publishers, ISBN 1402022352, 2004

[4] Codd, E. F. (1970). "A Relational Model of Data for Large Shared Data Banks". *Communications of the ACM* **13** (6): 377. doi:10.1145/362384.362685

[5] Dana Scott: Data Types as Lattices, SIAM Journal on Computing 1976, Vol. 5, No. 3, pp. 522-587, ISSN: 0097-5397

[6] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber Bigtable: A Distributed Storage System for Structured Data, J. ACM Transactions on Computer Systems (TOCS), Volume 26, Issue 2, June 2008, Article No. 4, ACM New York, NY, USA

[7] J. Demetrovics: Candidate Keys and Antichains, SIAM J. Algebraic Discrete Math., 1 (1980), p. 92

[8] János Demetrovics, Leonid Libkin, Ilya B. Muchnik: Functional Dependencies in Relational Databases: A Lattice Point of View, Discrete Applied Mathematics,Volume 40, Issue 2, 10 December 1992, pp. 155-185

[9] Karthik Kambatla, Giorgios Kollias, Vipin Kumar, Ananth Grama: Trends in Big Data analytics. J. Parallel Distrib. Comput. 74(7), 2014, pp. 2561-2573

[10] Paredaens, J., De Bra, P., Gyssens, M., Van Gucht, D.: The Structure of the Relational Database Model, EATCS Monographs on Computer Science, nr. 17, Springer Verlag, ISBN 3540137149, 1989