Partial dependencies in relational databases and their realization*

J. Demetrovics

Computation and Automation Institute, Hungarian Academy of Sciences, Victor Hugo u. 18–22, H-1132 Budapest, Hungary

G.O.H. Katona and D. Miklós

Mathematical Institute, Hungarian Academy of Sciences, P.O.B. 127, H-1364 Budapest, Hungary

Received 12 January 1991

Abstract

Demetrovics, J., G.O.H. Katona and D. Miklós, Partial dependencies in relational databases and their realization, Discrete Applied Mathematics 40 (1992) 127–138.

Weakening the functional dependencies introduced by Amstrong we get the notion of the partial dependencies defined on the relational databases. We show that the partial dependencies can be characterized by the closure operations of the poset formed by the partial functions on the attributes of the databases. On the other hand, we give necessary and sufficient conditions so that for such a closure operation one can find on the given set of attributes a database whose partial dependencies generate the given closure operation. We also investigate some questions about how to realize certain structures related to databases by a database of minimal number of rows, columns or elements.

1. Introduction

In this paper we investigate a new kind of dependencies, called *partial dependency*, defined on databases first introduced in [4] and discussed in details in [5]. After repeating some results from [5] which show some kind of "negative" result on whether one can simplify a database using partial dependencies we will deal with the realization of these dependencies.

The model of a database is a matrix in this paper, columns of whose, called *at-*

Correspondence to: Professor G.O.H. Katona, Mathematics Institute, Hungarian Academy of Sciences, Realtanoda u. 13–15, Budapest 1053, Hungary.

* This research was partially supported by Hungarian National Foundation of Scientific Research Grant no. 2575.

tributes, are indexed by the names of the data (a column contains the same sort of data) while the rows are indexed by the names of the individuals and called *records* (a row contains the data of a single individual).

The set of the attributes (and the columns of the matrix) will be denoted by Ω. If $A \subset \Omega$, $b \in \Omega$, we say, according to the original definition of dependency [1,2], that *b functionally depends on A* iff the data in A uniquely determine the datum in b. In many real examples it may happen that this condition is only "almost" fulfilled, i.e., there are sets $A \subset \Omega$ and attributes $b \in \Omega$ such that for almost all rows if two of them coincide in A, then they coincide in b as well. In other words, certain data structures in the columns A uniquely determine the datum in b, but certain others do not. We say that b is partially dependent on a data structure in A if all records of the database system containing these data in A contain the same datum in b.

There are theoretical and practical reasons to investigate partial dependency. The theoretical one is, for example, that every functional dependency contains a partial dependency and the later one is a finer structure. A practical reason is the following: consider a very simple database consisting of four columns $\Omega = \{x_1, x_2, x_3, x_4\}$ such that x_2 functionally depends on x_1 and x_4 depends on x_3. Now instead of storing the whole matrix on four columns practically it is enough to store only the two columns x_1 and x_3 and another small matrix showing the functional dependencies. However, if x_2 only "almost" depends on x_1, and x_4 almost depends on x_3, i.e., the dependencies are violated only in a few number of rows, then the functional dependency model is not enough to simplify the storage of the database (and save memory), while it could be done using partial dependency.

In the second section of the paper we will give the necessary definitions and notations while the third section contains the investigation carried out to see if we can weaken the partial dependency structure keeping all the necessary information about the database but simplifying the storage and search. In the last section first we investigate if the structures introduced in Section 3 can be represented by databases then suggest a possible way to avoid redundancy (but we will loose information as well) and investigate the representability of this structure as well. We also raise and partially solve some questions about how to realize this structure with databases on a minimum number of rows and/or columns and with a limited number of elements.

2. Definitions, notations

Here a database will be considered as an $m \times n$ matrix, the columns be called attributes and their sets denoted by Ω. An element of the database is an element of the matrix. The subsets of Ω will be denoted by capital letters while the columns, i.e., the elements of Ω and the elements of the database will be denoted by lowercase letters.

The best-known dependency of databases is called functional dependency defined

in the following way: if we have two subsets A, B of Ω then B functionally depends on A (denoted by $A \to B$) iff there are no two rows (records) of the database which coincide in A but differ in at least one position of B. The singular databases are not uniquely determined by the set of the functional dependencies they satisfy but these dependencies contain much information about them.

If we weaken the functional dependency we get the much weaker but much more concrete partial dependency which depends on the single elements. Suppose that in a database the elements in the two columns a_1 and a_2 do not determine uniquely the element in column b in every case, but for example if in a row the elements in the columns a_1, a_2 are r_1, r_2, respectively, then the element in the attribute b is r_3, i.e., uniquely determined. We denote this fact by $(a_1, a_2; r_1, r_2) \to (b; r_3)$.

To give the general definition of partial dependency we first define the *partial functions*. If D_i denotes the set of the possible values of the attributes a_i in a database (the elements of the database) and $r_i \in D_i$, then $\alpha = (a_1, \ldots, a_k; r_1, \ldots, r_k)$, where $a_i \in \Omega$ is called a partial function whose *domain* is the set $D(\alpha) = \{a_1, \ldots, a_k\}$ and the value of the function taken at a_i is r_i. For example, the rows of the database are partial functions, and deleting some elements from the domain of the partial function together with the value taken there results another partial function. We say that a partial function $\beta = (b_1, \ldots, b_l; s_1, \ldots, s_l)$ *depends* (partially depends) on α (in a given database)—which we will denote by $\alpha \to \beta$—iff in every row of the database which contains the elements r_i in the rows a_i $(1 \leq i \leq k)$ the columns b_j will contain the elements s_j $(1 \leq j \leq l)$.

The above definition is given only for those partial functions, which are *coherent* in a given database, which means that they are realized by at least one row of the database, i.e., deleting some entries of this row we get the partial function. For a given database the coherent partial functions have a certain nice structure. If $\alpha = (a_1, \ldots, a_k; r_1, \ldots, r_k)$ and $\beta = (b_1, \ldots, b_l; s_1, \ldots, s_l)$ are two partial functions then we say that α is a part of β, denoted by $\alpha \subset \beta$ if $\{a_1, \ldots, a_k\} \subseteq \{b_1, \ldots, b_l\}$ and $a_i = b_j$ implies that $r_i = s_j$. For every database the set of coherent partial functions will have the *hereditary* structure, i.e., they will satisfy the following two properties:

(2.1) If $\alpha \in P$ and $\beta \subset \alpha$, then $\beta \in P$.

(2.2) For every $\alpha \in P$ there is a $\gamma \in P$, such that $\alpha \subseteq \gamma$, $\gamma = (c_1, \ldots, c_n; \ldots)$ and $\{c_1, \ldots, c_n\} = \Omega$.

We define the *intersection or meet* and *union* of partial functions, though the later one only for certain pairs. Let $\alpha = (a_1, \ldots, a_k; r_1, \ldots, r_k)$ and $\beta = (b_1, \ldots, b_l; s_1, \ldots, s_l)$ be two partial functions. The intersection of them is the partial function γ whose domain is the set of those attributes c which are in the domains of both α and β, say $c = a_i = b_j$ and $r_i = s_j$. The value of the partial function γ at c is of course $r_i = s_j$. For the partial functions $\alpha = (a_1, \ldots, a_k; r_1, \ldots, r_k)$ and $\beta = (b_1, \ldots, b_l; s_1, \ldots, s_l)$ we define the union of them only if the domain of their intersection equals the intersection of their domains (the earlier is always a subset of the later). In this case their union is the partial function γ whose domain is the union of the domains of α and β and which takes the following values at an attribute c in its domain:

r_i if $c = a_i$ and $c \notin \{b_1, \ldots, b_l\}$,

s_j if $c = b_j$ and $c \notin \{a_1, \ldots, a_k\}$,

$r_i = s_j$ if $c = a_i = b_j$.

In the last case $r_i = s_j$ is given by the fact that the domain of the intersection of α and β equals the intersection of their domains.

3. Different models and structures for the partial dependency

Using these definitions we give the *closure* of a partial function, similarly to the (well-known set-theoretical) closure of a subset of Ω (see [3]). We need the following straightforward lemma:

Lemma 3.1. *If α, β, γ, δ are partial functions in a database, then*

(3.1.1) $\alpha \to \alpha$;

(3.1.2) $\alpha \to \beta$ *and* $\beta \to \gamma$ *imply* $\alpha \to \gamma$;

(3.1.3) $\alpha \subseteq \gamma$, $\delta \subseteq \beta$ *and* $\alpha \to \beta$ *imply that* $\gamma \to \delta$;

(3.1.4) *if* $\alpha \to \beta$, $\gamma \to \delta$ *and* $\alpha \cup \gamma$ *exists and is coherent, then* $\beta \cup \delta$ *exists and is coherent as well, and* $\alpha \cup \gamma \to \beta \cup \delta$.

Definition. The *closure* of the (coherent) partial function α (denoted by $C(\alpha)$) is the partial function β which has the largest domain among those partial functions γ for which $\alpha \to \gamma$.

There should be a partial function β satisfying the above definition since if β_1 and β_2 are two partial functions with $\alpha \to \beta_1$ and $\alpha \to \beta_2$ then by (3.1.4) their union exists and is coherent and $\alpha \to \beta_1 \cup \beta_2$. Thus we have

$$C(\alpha) = \bigcup_{\alpha \to \beta} \beta \tag{1}$$

or in other words

$$C(\alpha) = \bigcup_{\alpha \to (b;s)} (b;s). \tag{2}$$

We have the following lemma listing the properties of the closure (the proof is straightforward and omitted):

Lemma 3.2. *If α and β are two partial functions on the same database, then*

(3.2.1) $\alpha \subseteq C(\alpha)$;

(3.2.2) $\alpha \subseteq \beta$ *implies that* $C(\alpha) \subseteq C(\beta)$;

(3.2.3) $C(C(\alpha)) = C(\alpha)$.

In general we will call a function a *C function closure* if it is defined on a hereditary

set of partial functions and satisfies the properties (3.2.1)–(3.2.3). Thus the function $\alpha \to C(\alpha)$ where α is a given partial function in a database and $C(\alpha)$ is its closure according to the Definition is a function closure. In the remaining of this section we will repeat the analogue of the investigation carried out by Demetrovics and Katona in [3,4] to find the connection between the usual closure operation and the functional dependency. The details and the easier proofs of these results are given in [5] but the argument here is self-contained.

First note that in a given database the closure operation given in the Definition by the partial dependency uniquely determines the partial dependency.

Lemma 3.3. *$\alpha \to \beta$ holds if and only if $\beta \subseteq C(\alpha)$.*

Let T be a family of pairs of partial functions in a database. T is called a *dependency family* if the $\to_T$ relation defined by T as

$$\alpha \to_T \beta \quad \text{if and only if} \quad (\alpha, \beta) \in T$$

satisfies properties (3.1.1)–(3.1.4). Lemma 3.1 shows that if we define the family T so that $(\alpha, \beta) \in T$ iff β depends on α then we get a dependency family. In the proof of the statements of Lemma 3.2 one only needs to have the facts about $\to$ given in Lemma 3.1, so we can define a function closure C for every dependency family.

Theorem 3.4. *There is a one-to-one correspondence between the T dependency families and C function closures given on the same ground set as it is given below:*

$$T \to_1 C(\alpha) = \bigcup_{(\alpha,(b;s)) \in T} (b;s). \tag{3}$$

The inverse of this is given by:

$$C \to_2 T = \{(\alpha, \beta) \colon \beta \subseteq C(\alpha)\}. \tag{4}$$

Proof. Lemma 3.2 proves that the operation given by $\to_1$ in (3) will be a function closure. On the other hand, we have to prove that the families T given by (4) will be dependency families, i.e., the corresponding relation $\to_T$ satisfies properties (3.1.1)–(3.1.4). Here C satisfies (3.2.1) which implies that $\to_T$ satisfies (3.1.1). If $\beta \subseteq C(\alpha)$ and $\gamma \subseteq C(\beta)$, then (3.2.2) and (3.2.3) give $\gamma \subseteq C(\beta) \subseteq C(C(\alpha)) = C(\alpha)$, that is (3.1.2) also holds for T. (3.2.3) follows from the definitions and property (3.2.2): If $\alpha \subseteq \gamma$, $\delta \subseteq \beta$ and $\beta \subseteq C(\alpha)$, then $\delta \subseteq \beta \subseteq C(\alpha) \subseteq C(\gamma)$, that is $(\gamma, \delta) \in T$.

Now we prove that $\to_T$ satisfies (3.1.4). We know that $\alpha \cup \gamma$ exists and that it is coherent, thus $C(\alpha \cup \gamma)$ exists and is coherent. $\beta \subseteq C(\alpha) \subseteq C(\alpha \cup \gamma)$ and $\delta \subseteq C(\gamma) \subseteq C(\alpha \cup \gamma)$, that is β and δ has a common coherent superset which easily implies that their union exists and that it is coherent and of course a subset of $C(\alpha \cup \gamma)$, which is exactly what we had to prove.

Still we have to prove that the two operations $\to_1$ and $\to_2$ are inverse of each other. Let $T_1 \to_1 C \to_2 T_2$ and (α, β) be an element of T_1. Then—according to prop-

erty (3.1.3) of T_1—every attribute of β takes the same value in the union which gives $C(\alpha)$ in (3) as it takes by β, that is $\beta \subseteq C(\alpha)$ which means that $(\alpha, \beta) \in T_2$. On the other hand, if $(\alpha, \beta) \in T_2$, then $\beta \subseteq C(\alpha)$, that is for every attribute b of β and value s taken there by β we have $\alpha \to (b;s)$, which implies with property (3.1.4) of T_1 that $\alpha \to \beta$, $(\alpha, \beta) \in T_1$.

Finally let $C_1 \to_2 T \to_1 C_2$ and α a coherent partial function. For every attribute b of $C_1(\alpha)$ and value s taken there by $C_1(\alpha)$ we have $(b;s) \subseteq C_1(\alpha)$, and thus $(\alpha, (b;s)) \in T$. This fact and definition (3) together imply $C_2(\alpha) = C_1(\alpha)$. □

What we have so far is that partial dependencies can be uniquely described by function closures (Lemma 3.2) or dependency families (Theorem 3.4). The dependency family corresponding to a partial dependency is essentially the same as the partial dependency and the function closure is definitely a simpler structure, so in the remaining of this section we will focus on the function closure operation. On the other hand, we have to know that not all function closure operations defined on partial functions are closure operations given by the partial dependency structure of a certain database. This question will be more thoroughly investigated in Section 4.

We will call a partial function α (in a database or according to a closure operation C defined by the database) *closed* if $\alpha = C(\alpha)$. We prove that the set of closed partial functions uniquely determines the closure and/or the dependency.

Theorem 3.5. *Let $\mathcal{G}$ be a family of partial functions defined on the same ground set Ω. $\mathcal{G}$ will be the set of closed partial functions according to a closure operation C iff*

(3.5.1) *for every $\alpha \in \mathcal{G}$ there is a $\gamma \in \mathcal{G}$, such that $\alpha \subseteq \gamma$, $\gamma = \{c_1, \ldots, c_n; \ldots\}$ and $\{c_1, \ldots, c_n\} = \Omega$;*

(3.5.2) *$\alpha, \beta \in \mathcal{G}$ implies that $\alpha \cap \beta \in \mathcal{G}$.*

Proof. The set of closed functions corresponding to a function closure satisfies properties (3.5.1) and (3.5.2). (2.2) assures that for every (closed) α there is a γ satisfying (3.5.1) and this γ will trivially be closed. If α and β are two closed partial functions, then $\alpha = C(\alpha)$ and $\beta = C(\beta)$, $\alpha \cap \beta \subseteq \alpha$ and so by (3.2.2), $C(\alpha \cap \beta) \subseteq C(\alpha) = \alpha$. Similarly, $C(\alpha \cap \beta) \subseteq C(\beta) = \beta$ and so $C(\alpha \cap \beta) \subseteq \alpha \cap \beta$. (3.2.1) gives the opposite of this, altogether $C(\alpha \cap \beta) = \alpha \cap \beta$, that is $\alpha \cap \beta$ is a closed partial function in the given dependency.

If an arbitrary family $\mathcal{G}$ of partial functions satisfies properties (3.5.1) and (3.5.2), we give the closure operation whose closed sets will be exactly the elements of $\mathcal{G}$. The function closure will be defined on those partial functions which are subsets of some maximal elements (those, whose domain is Ω) of $\mathcal{G}$ by

$$C(\alpha) = \bigcap_{\alpha \subseteq \beta \in \mathcal{G}} \beta.$$

It is easy to see that this closure will have as closed sets exactly the elements of $\mathcal{G}$. □

Here property (3.5.2) means that $\mathcal{G}$ is closed. The families of partial functions satisfying properties (3.5.1) and (3.5.2) will be called *partial meet-semilattices*. The previous theorem means that the families of closed partial functions form a partial meet-semilattice and so the function closure uniquely gives the meet-semilattice. Next we prove the opposite of this.

Theorem 3.6. *There is a one-to-one relation between the partial meet-semilattices* $\mathcal{G}$ *and function closure operations C defined on the same ground sets as it is shown below*:

$$\mathcal{G} \to_1 C(\alpha) = \bigcap_{\alpha \subseteq \beta \in \mathcal{G}} \beta. \tag{5}$$

The inverse of this is given by:

$$C \to_2 \mathcal{G} = \{\alpha\colon C(\alpha) = \alpha\}. \tag{6}$$

Proof. It has been already shown that definition (5) gives a function closure operation and definition (6) gives a partial meet-semilattice and also that (6) gives all the partial meet-semilattices. We will show that $\to_2$ is injective and that $\to_1$ is an inverse of $\to_2$. Let C_1 and C_2 be two different function closures, α a partial function such that $C_1(\alpha) \neq C_2(\alpha)$ and $\mathcal{G}_1$ and $\mathcal{G}_2$ the two corresponding meet-semilattices according to $\to_2$ (here $C_1(\alpha) \neq C_2(\alpha)$ may mean that one of $C_1(\alpha)$ and $C_2(\alpha)$ does not exist). If we have $C_1(\alpha)$ but not $C_2(\alpha)$ then the earlier (which is in $\mathcal{G}_1$) may not be in $\mathcal{G}_2$ since the opposite would mean that there is a partial function in $\mathcal{G}_2$ which contains α, and so $C_2(\alpha)$ would be defined. If both $C_1(\alpha)$ and $C_2(\alpha)$ are defined but $C_1(\alpha) \neq C_2(\alpha)$ then there is a partial function $(b;s)$ defined only on one attribute such that exactly one of $C_1(\alpha)$ and $C_2(\alpha)$ contains it, say $(b;s) \subseteq C_1(\alpha)$ and $(b;s) \not\subseteq C_2(\alpha)$. Then $\alpha \subseteq C_2(\alpha)$ and so $C_1(C_2(\alpha)) \supseteq C_1(\alpha) \ni (b;s)$. At the same time we have $C_2(\alpha)$ and so $C_2(C_2(\alpha)) = C_2(\alpha) \not\ni (b;s)$. This implies that the closure of the partial function $C_2(\alpha)$ in the closure C_1 is a superset of $C_2(\alpha)$, that is $\mathcal{G}_1$ does not contain $C_2(\alpha)$, which is obviously an element of $\mathcal{G}_1$.

Let C_1 be a closure, $C_1 \to_2 \mathcal{G}$ and $\mathcal{G} \to_2 C_2$. If for an arbitrary partial function α we have $C_1(\alpha)$ then $C_1(\alpha) \in \mathcal{G}$, so $C_2(\alpha)$ is defined and—as it was proved in Theorem 3.5—it equals $C_1(\alpha)$. On the other hand, if we have $C_2(\alpha)$ then there is a partial function in $\mathcal{G}$ containing α, and so the closure of α is defined in C_1. The existence of $C_1(\alpha)$ again implies $C_1(\alpha) = C_2(\alpha)$. □

In the case of the functional dependency the fact that the closure operation and so the dependency itself is uniquely determined by the set of the closed sets of attributes decreases the necessary space to store the information in a database and may significantly simplify the structure of the storage. On the opposite of this, in the case of the partial dependency the set of closed partial functions is a bigger set than even the whole database, since every row of the database is a closed partial function. This bigger structure, however, carries no more information than the

original database. So the question is if we can consider only a subset of the closed functions which carries all information, or at least a big part of it.

Property (3.5.2) shows that the partial meet-semilattice $\mathcal{G}$ can be described by much fewer elements than its cardinality. Let $\mathcal{M}(\mathcal{G})$ denote the set of those elements of $\mathcal{G}$ which are not the meet of two other different elements of $\mathcal{G}$. (Even this weaker structure will contain all the rows of the database, and possible some other partial functions as well.)

Lemma 3.7. *Every element of a partial meet-semilattice $\mathcal{G}$ is the meet of some (≥ 1) partial functions from $\mathcal{M}(\mathcal{G})$ but there is no real subset of $\mathcal{M}(\mathcal{G})$ having this property.*

We have the following straightforward lemma describing the families $\mathcal{M}(\mathcal{G})$.

Lemma 3.8. *A family $\mathcal{F}$ of partial functions is an $\mathcal{M}(\mathcal{G})$ family of a partial meet-semilattice $\mathcal{G}$ iff it satisfies the following two properties:*

(3.8.1) *for all $\alpha \in \mathcal{F}$ there is a $\gamma \in \mathcal{F}$ such that $\alpha \subseteq \gamma$, $\gamma = \{c_1, \ldots, c_n; \ldots\}$ and $\{c_1, \ldots, c_n\} = \Omega$;*

(3.8.2) *if $\alpha = \bigcap_{i=1}^{r} \alpha_i$, $r \geq 1$, $\alpha, \alpha_1, \ldots, \alpha_r \in \mathcal{F}$ then $\alpha = \alpha_i$ for some i.*

The families of partial functions satisfying properties (3.8.1) and (3.8.2) are called *meet-free*. The proof of the following theorem is straightforward.

Theorem 3.9. *There is a one-to-one realization between the partial meet-semilattices $\mathcal{G}$ and the meet-free families $\mathcal{F}$ of partial functions given by*

$$\mathcal{G} \to_1 \mathcal{M}(\mathcal{G}), \tag{7}$$

whose inverse is given by

$$\mathcal{F} \to_2 \mathcal{G} = \{\alpha_1 \cap \cdots \cap \alpha_r \colon r \geq 1,\ \alpha_1, \ldots, \alpha_r \in \mathcal{F}\}. \tag{8}$$

4. The realization of partial dependencies

By the results of the previous section the partial dependencies or the partial function closures are uniquely determined by the meet-free families of the partial functions. However, a similar statement about the databases and meet-free families of partial functions is not true as the following example shows (we have seen that the database uniquely determines the corresponding meet-free family but it could be that for a meet-free family there is no database corresponding to it). Let the meet-free family of partial functions be $\{(a, b, c; p, q, r), (a, b; p, q)\}$. The only database whose partial functions closure's meet-free family could be this is the one having three attributes (a, b, c) and the only one row (p, q, r), but the corresponding meet-free family has only one element $(a, b, c; p, q, r)$.

The problem in the above example was that the meet-free family there had a non-maximal element (i.e., an element not being defined on the whole ground set Ω). We show that this can never happen.

Theorem 4.1. *Let $\mathcal{G}$ be a partial meet-semilattice formed by the closed partial functions of a database. Then every element of $\mathcal{G}$ which is not defined on the whole ground set Ω is the meet of two partial functions different from itself.*

Proof. Let $\{a_1, \ldots, a_k\}$ be a real subset of Ω and $\alpha = (a_1, \ldots, a_k; r_1, \ldots, r_k)$ be a closed partial function. Let $c \in \Omega$, but $c \notin \{a_1, \ldots, a_k\}$. Since $\alpha = (a_1, \ldots, a_k; r_1, \ldots, r_k)$ is a closed partial function we have two different elements of the database s, t such that there are two rows of the database containing $(a_1, \ldots, \alpha_k, c; r_1, \ldots, r_k, s)$ and $(a_1, \ldots, \alpha_k, c; r_1, \ldots, r_k, t)$. Let the closure of $(a_1, \ldots, \alpha_k, c; r_1, \ldots, r_k, s)$ be β and the closure of $(a_1, \ldots, \alpha_k, c; r_1, \ldots, r_k, t)$ be γ. Then β and γ both contain α and so $\beta \cap \gamma \supseteq \alpha$. Let now β and γ be two closed partial functions different from α such that $\beta \cap \gamma \supseteq \alpha$ and $D(\beta \cap \gamma) \setminus D(\alpha)$ is a minimal set. Suppose that $D(\beta \cap \gamma) \setminus D(\alpha)$ contains at least one element, say c and $\beta \cap \gamma$ takes the value s there. $\alpha = (a_1, \ldots, a_k; r_1, \ldots, r_k)$ is closed, and so there should be a row of the database which contains α but takes a different value, say t, at c. Let δ be the closure of $(a_1, \ldots, \alpha_k, c; r_1, \ldots, r_k, t)$. Then $\delta \supseteq \alpha$ and so $\delta \cap (\beta \cap \gamma) \supseteq \alpha$ but δ and $\beta \cap \gamma$ take different values at c and so the domain of $\delta \cap (\beta \cap \gamma)$ is strictly smaller than the domain of $\beta \cap \gamma$, a contradiction. □

This means that in the case of a given database $\mathcal{M}(\mathcal{G})$ satisfies the following property:

(4.1) for all $\alpha \in \mathcal{M}(\mathcal{G})$-re, $\alpha = \{c_1, \ldots, c_n; \ldots\}$ and $\{c_1, \ldots, c_n\} = \Omega$.

A meet-free family of partial functions having this additional property will be called *big*. A big family is uniquely realizable by the partial dependency relation *of a database*, as the following clear lemma shows.

Lemma 4.2. *For a given big family $\mathcal{F}$ of partial functions let M be a database (matrix) whose attributes are the elements of Ω and rows the elements of $\mathcal{F}$, and so defined on the whole set Ω. Then the meet-free family of partial functions corresponding to M as described in Section 3 is $\mathcal{F}$, or equivalently, the system of partial dependencies given by M is the same as given by $\mathcal{F}$.*

So far we have been investigating structures equivalent to the original database (having no less information) and it turned out that in view of the partial dependency we cannot simplify the structure of a database. The case may be different if we combine functional and partial dependencies to solve the problem mentioned in the introduction, which may be the subject of a further investigation. Here we focus on the possibility of loosing some information in exchange of simplifying the structure. We could avoid the redundandy mentioned at the end of Section 3 by deleting all the maximal elements of $\mathcal{M}(\mathcal{G})$ but then we would be left by an empty set, as proved

here. A better approach could be to delete first the maximal partial functions from the set of closed partial functions $\mathcal{G}$, denote the remainder by $\mathcal{G}'$ and then take those elements from $\mathcal{G}'$ which are not the meet of two other elements. Let us denote the result by $\mathcal{M}_1(\mathcal{G})$. One can easily see that from $\mathcal{M}_1(\mathcal{G})$ we can get all the partial functions of $\mathcal{G}$ which are not defined on the whole ground set Ω and for the relation of $\mathcal{G}'$ and $\mathcal{M}_1(\mathcal{G})$ we can repeat Lemma 3.8 and Theorem 3.9, except that from Lemma 3.8 property (3.8.1) should be deleted.

We have a new structure which may or may not be used to describe the database. For a given $\mathcal{G}$ the corresponding $\mathcal{M}_1(\mathcal{G})$ will satisfy (3.8.2) but it will not be necessarily true that every nonmaximal element of $\mathcal{M}_1(\mathcal{G})$ will be the meet of two elements different from it. For example in Table 1 the closed partial functions in the rows are $(a, b; p, q)$ and $(a; p)$, where the latter is nonmaximal in $\mathcal{M}_1(\mathcal{G})$ but is not the meet of other closed partial functions from the set $\mathcal{M}_1(\mathcal{G})$.

Of course we have a property $\mathcal{M}_1(\mathcal{G})$ trivially satisfies:

(4.2) for every $(a_1, \ldots, a_k; \ldots) \in \mathcal{M}_1(\mathcal{G})$ $\{a_1, \ldots, a_k\}$ is a real subset of Ω.

Theorem 4.3. *For every set $\mathcal{F}$ of partial functions satisfying properties* (3.8.2) *and* (4.2) *there is a database such that for the corresponding partial meet-semilattice $\mathcal{G}$ we have $\mathcal{M}_1(\mathcal{G}) = \mathcal{F}$.*

Proof. We prove that there is meet-semilattice $\mathcal{G}$ of partial functions for which $\mathcal{M}(\mathcal{G})$ is not only meet-free, but big as well. Then Lemma 4.2 assures that there is a corresponding database.

For every element α of $\mathcal{F}$ take two new elements of the future database, say a and b. Then for α define two partial functions defined on the whole ground set Ω in the following way: both contain α and on the attributes not in the domain of α one takes everywhere a while the other one takes everywhere b. Consider now the big set $\mathcal{N}$ consisting of the pairs taken for every element α of $\mathcal{F}$ and only those. The meet of the elements of the pair taken from α will trivially be α and one can easily see that every other meet of at least two elements from $\mathcal{N}$ will be the meet of some elements of $\mathcal{F}$. So if we take the set $\mathcal{G}$ corresponding to $\mathcal{N}$ then $\mathcal{M}_1(\mathcal{G})$ will really be $\mathcal{F}$. □

Finally we investigate the following question: if we have a set $\mathcal{F}$ of n partial functions satisfying properties (3.8.2) and (4.2) then how many rows do we need in a database for which for the corresponding partial meet-semilattice $\mathcal{G}$ we have

Table 1

Attributes	a	b	c
First row	p	r	s
Second row	p	q	t
Third row	p	q	u

$\mathcal{M}_1(\mathcal{G}) = \mathcal{F}$? The theorem above implies that $2n$ rows are enough. On the other hand, it is easy to see that the partial functions in $\mathcal{F}$ should be contained in at least two different rows such that it is the meet of those two rows. This means that if we have k rows then $\binom{k}{2} \geq n$ or roughly $k \geq \sqrt{2n}$. Theoretically one can reach this lower bound: if in a database the meets of every two rows are different from each other we will have many elements of $\mathcal{M}_1(\mathcal{G})$ on relatively few rows.

Lemma 4.4. *For every k there is a database $\mathcal{G}$ on k rows such that $\mathcal{M}_1(\mathcal{G}) = \binom{k}{2}$.*

Proof. Suppose we have an indefinite supply of different elements for the database and let the database be defined on $\binom{k}{2}$ columns, each being assigned to a different pair of rows. If column a corresponds to a pair of rows, then these rows at the position given by the columns contain the same element (the corresponding database takes the same value) and with these exceptions the database contains all different elements. One can easily see that in this database for every pair of rows we have one element in $\mathcal{M}_1(\mathcal{G})$ which has exactly one element in its domain. □

In the proof of Lemma 4.4 we considered as many columns and as many elements as it could be reasonable. The question becomes much more difficult if we limit the number of columns and elements of the database and still want to reach a bound in Lemma 4.4. For example, we may ask: what is the minimum number of the columns of a database if it has k rows and $|\mathcal{M}_1(\mathcal{G})| = \binom{k}{2}$, i.e., the meet of every two rows of the database is different from the other ones. Frankl and Füredi investigated a similar question for sets [6]. Although they considered union instead of meet, the questions are trivially equivalent in case of subsets of a set. However, in our case their results do not apply: here the meet and union are not dual of each other (the latter is not even defined sometimes) and the definition of the meet of two partial functions is rather different from the definition of the intersection of two sets.

Thus we have only the following two simple results on the minimum number of columns:

Lemma 4.5. *If in a database on k rows the meet of every two rows (considered as partial functions) are different of the other ones the database has at least $\log_2 k$ columns.*

Proof. On m columns the domain of every row has $2^m - 1$ nonempty subsets and so the row as partial function may have $2^m - 1$ different meets of other functions. If there are $k-1$ other rows and all of them have to have different meet with the given row we have $2^m - 1 \geq k - 1$. □

In this lemma we had no information on the number of the elements of the database. It is clear that a limit on the number of elements may increase the number of necessary columns but we have results only if we are restricted to two elements.

Lemma 4.6. *If in a database on k rows the meet of every two rows (considered as partial functions) are different of the other ones and the database has only two elements the database has at least* $(2/\log 3)\log_2 k$ *columns.*

Proof. On m columns there are 3^k possible partial functions if the functions may take only two different values. The meet of the rows should have domain smaller than the whole ground set Ω, and so a row cannot be the meet of any (other) two rows. We have k rows and so $\binom{k}{2}$ different intersections which implies that $3^m - k \geq \binom{k}{2}$. □

References

[1] W.W. Armstrong, Dependency structures of data base relationships, Information Processing '74 (North-Holland, Amsterdam, 1974) 580–583.

[2] E.F. Codd, A relational model of data for large shared data banks, Comm. ACM 13 (1970) 377–387.

[3] J. Demetrovics and G.O.H. Katona, Combinatorial problems of database models, in: Algebra, Combinatorics and Logic in Computer Science, Colloquia Mathematica Societatis János Bolyai 42 (North-Holland, Amsterdam, 1983) 331–353.

[4] J. Demetrovics and G.O.H. Katona, Extremal Combinatorial problems of database models, Manuscript.

[5] J. Demetrovics, G.O.H. Katona and D. Miklós, Partial dependencies in relational databases, Alkalmaz. Mat. Lapok 15 (1990/91) 163–179 (in Hungarian).

[6] P. Frankl and Z. Füredi, Union-free hypergraphs and probability theory, European J. Combin. 5 (1984) 127–131.