Partial dependencies in relational databases and their realization

J. Demetrovics
Computer and Automation Institute, Hungarian Academy of Sciences, P.O. Box 12, H-1525 Budapest, Hungary

G.

D. Miklos
Central Statistical Office, Hungarian Academy of Sciences, P.O. Box 12, H-1525 Budapest, Hungary

Abstract

In the paper, the problem of partial dependencies in relational databases is considered. A new concept of "partial dependency" is introduced and its significance is discussed. The concept is illustrated by an example. The problem of detecting and eliminating partial dependencies is discussed. The paper concludes with a discussion of the implications of the results for the design of relational databases.

Introduction

In this paper, we investigate a new kind of dependency, called partial dependency, defined on databases. This concept was introduced in [1] and discussed in detail in [2]. After reviewing some results from [3] which show that a kind of "partial" weak on another relation can be expressed as a partial dependency, we deal with the implications of these dependencies.

The results of the paper are a tool to the investigation of the problem of partial dependencies, a concept introduced by the authors in a previous paper, and it is well studied in the literature.

The paper is divided into four sections. In the first section, the concept of partial dependency is introduced and its significance is discussed. In the second section, some results from the literature are reviewed. In the third section, the problem of detecting and eliminating partial dependencies is discussed. Finally, the conclusions of the paper are presented.
utensils are ordered by the names of the data in columns consider the same set of
items while the ones are ordered by the names of the individuals and order number
in cases of the data of each individual.

The size of the database sets the collection of the matrix will be described by $D$.

The set of columns in the database is described by $A$. The relation of a column in $A$
that is particularly depends on $X$ in the data is a weak dependency the column in $A$
may contain $X$ to $Y$, which is a weak dependency if the column in $A$ contains only $X$.

The set of all columns in $D$ that the column in $A$ contains in its $X$ and the weak
dependency $A$, then $A$ is called $A$-weak.

We say that the partial column in a data structure $X$ if $X$ all sets of
relations of the database cross-containing these data. $X$ contains the same element.

There are theoretical and practical reasons to investigate partial dependencies. The
strongest one is, for example, that every functional dependency contains a partial
dependency and hence one is a strong one. A practical reason is the following:
consider a very simple database consisting of two columns $(X, Y, Z, W)$ such
that $X$ contains $Y$ and $Z$ depends on $X$, $W$ does not depend on $X$, and $W$
contains $Z$. Then $W$ must be considered as a weak dependency. However,
if $X$ contains $Y$ and $Z$ depends on $X$, $W$ depends on $X$, i.e., the dependent
columns are included only in a total set of columns, then the functional
dependency $X$ contains $Y$ and the partial dependency is the same as the
functionality and hence no weak dependency could be drawn using partial dependency.

In the second section of the paper we will put the necessary definitions and lemmas
while the third section concerns the investigation of the cases that will be used
for investigating partial dependencies. The second section contains all the necessary
information the reader needs to understand the results presented in the third section.


dependence are:

1. Definition: relation:

A database will be considered as an $X$ a relation, the columns are called $X$
if there is an $X$ such that the column is a relation of $X$. The relation of the database is an element of
the columns. The columns of $D$ do not depend on each other, while the columns,
the order of columns and the order of database are defined as a relation.

2. Definition: dependencies:

A database of columns is called a functional dependency defined

In the following we consider two relations \( R \) and \( S \) in a database, and we are interested in the functional dependencies that hold among the attributes of the relations. The database is a collection of information, and the functional dependencies are constraints that ensure the consistency of the data.

If we consider the functional dependencies, we can see that the database must maintain certain properties to ensure the correctness of the information. For example, if we have a relation \( R \) with attributes \( A \) and \( B \), and if \( \text{func}(R, A \rightarrow B) \) holds, then for any pair of tuples \((a, b_1)\) and \((a, b_2)\) in \( R \), we have \( b_1 = b_2 \). This means that the attribute \( B \) is functionally dependent on the attribute \( A \), and any value of \( A \) determines a unique value of \( B \).

To define the functional dependencies, we first consider the partial functions. A partial function is a function that is not defined for all possible inputs. In a database, we can consider partial functions as functions that are defined only for certain inputs. For example, if we have a relation \( R \) with attributes \( A \) and \( B \), and if \( \text{partial}(R, A \rightarrow B) \) holds, then for any pair of tuples \((a, b_1)\) and \((a, b_2)\) in \( R \), we have \( b_1 \neq b_2 \) if \( a \) is not in the domain of the function. This means that any value of \( A \) can determine more than one value of \( B \).
In the last case, it is given by the fact that the domain of the intervention of \( u \) and \( v \) is the intersection of their domains.

3. Shrinkage models and estimators for the partial dependency

Using these definitions we give the closure of a partial function, 
...
of partial functions and satisfies the properties (2.3-2.5). Thus, the function $\pi_\delta$ where $\delta$ is a given partial function takes a directed graph $\G$ to its closure according to the definition in a function closure. In the remaining of this section we will explore the analysis of the morphism carried out by the two operations, and the properties of $\pi_\delta$ that are connected to the closed closure operation and the closed graph operation. We will conclude this section with some more details about the properties of these results.

First note that if we obtain the closure operation given in the definition by the partial dependence then the results obtained in the second section are obtained in the natural way.

**Lemma 5.2.** $\pi_\delta$ is a function if and only if $\pi_\delta(\G)$.

Let $\G$ be the family of pairs of partial functions in a directed graph. $\pi_\delta$ is called a dependence family if the $\pi_\delta$ operation defined by $\pi_\delta$

\[ \forall \delta \in \Delta \quad \forall \G \in \mathcal{G} \quad \pi_\delta(\G) \]

satisfies properties (2.3-2.5). Lemma 3.1 shows that if we define the family

\[ \forall \delta \in \Delta \quad \forall \G \in \mathcal{G} \quad \pi_\delta(\G) \]

as the proof of Lemma 3.1 we obtain a dependence family. The proof of Lemma 3.2 is left as an exercise for the reader.

**Theorem 6.4.** There is a one-to-one correspondence between the $\pi_\delta$ dependence family and the closed graph operation for the given order set as it is given below.

\[ \forall \delta \in \Delta \quad \forall \G \in \mathcal{G} \]

The inverse of this is given by

\[ \forall \delta \in \Delta \quad \forall \G \in \mathcal{G} \]

**Proof.** Lemma 3.2 proves that the operation given by $\pi_\delta$ on $\G$ will be a dependence family. On the other hand, we have to prove that the family $\pi_\delta$ given by the definition satisfies properties (2.3-2.5) for the given directed graph $\G$. Thus, by Lemma 3.1 we obtain a dependence family.

Note that $\pi_\delta(\G) \supseteq \pi_\delta(\G), \forall \delta \in \Delta, \forall \G \in \mathcal{G}$. Hence, the definition of $\pi_\delta$ for $\G$ follows from the closure of the directed graph $\G$.

Now we prove that $\pi_\delta$ satisfies (2.3). We know that (2.3) is a candidate for the closure of $\G$. Consider $\mathcal{G} \in \mathcal{G}$, where $\mathcal{G} = \{\G \in \mathcal{G} : \pi_\delta(\G) \subseteq \mathcal{G} \}$. Consider $\mathcal{G}_0 = \{\G \in \mathcal{G} : \pi_\delta(\G) \subseteq \mathcal{G} \}$.

If we have to ensure that the two operations $\pi_\delta$ and $\pi_\delta$ are inverses of each other, let $\pi_\delta(\G) = \mathcal{G}$, then according to prop.
Theorem 1. Let $h$ be a family of partial functions defined on the same ground set $d$. It will be set of closed partial functions according to a closed operation $\alpha$ if, for any set $h$ and any $x \in h$, such that $x, \dot{x} \subseteq d$, and $x \cup \dot{x}$ is the domain of $h$.

Proof. The set of closed partial functions according to a closed operation $\alpha$ satisfies properties (C1) and (C2) for both $x \subseteq \dot{x}$ and for $x \subseteq \dot{x}$.

Theorem 2. Let $h$ be a family of partial functions defined on the same ground set $d$. If $h$ is a set of closed partial functions according to a closed operation $\alpha$, then $h$ is a family of partial functions defined on the same ground set $d$, such that $x \subseteq \dot{x}$.

Theorem 3. Let $h$ be a family of partial functions defined on the same ground set $d$. If $h$ is a set of closed partial functions according to a closed operation $\alpha$, then $h$ is a family of partial functions defined on the same ground set $d$, such that $x \subseteq \dot{x}$.
Here (property 1.3.2) means that \( C \) is closed. The family of partial functions 

\[
\mathcal{F} = \{ f \mid f \text{ is a partial function} \}
\]

satisfies properties 1.3.1 and 1.3.2, why satisfied partial two conditions, the 

property deems that the family of closed partial functions have a partial 

two-valuation and so the function closure operation gives the same results.

Now we prove the opposite of this.

Corollary 3.2. There is a case to cover since between the partial two-valuation

4, 5, 6 however closure operation is defined on the same set as it is above shown

\[
\begin{align*}
4 & \rightarrow \mathcal{F}(4) = 1 \text{ (for } 4 \text{ in } \mathcal{F}) \\
5 & \rightarrow \mathcal{F}(5) = 1 \text{ (for } 5 \text{ in } \mathcal{F}) \\
6 & \rightarrow \mathcal{F}(6) = 1 \text{ (for } 6 \text{ in } \mathcal{F})
\end{align*}
\]

The closure of \( \mathcal{F} \) is given by

\[
C = \{ f \mid f \text{ is a partial function} \}
\]

Proof. It has been already shown that definition 3.2 gives a function closure 

operation and definition 3.3 gives a partial two-valuation and also that the 

family of the partial two-valuation. We will show that \( C \) is closed and 

that \( C \) is a partial function.

\[
\begin{align*}
7 & \rightarrow \mathcal{F}(7) = 1 \text{ (for } 7 \text{ in } \mathcal{F}) \\
8 & \rightarrow \mathcal{F}(8) = 1 \text{ (for } 8 \text{ in } \mathcal{F}) \\
9 & \rightarrow \mathcal{F}(9) = 1 \text{ (for } 9 \text{ in } \mathcal{F})
\end{align*}
\]

The closest of \( \mathcal{F} \) is given by

\[
\begin{align*}
C' = & \{ f \mid f \text{ is a partial function} \}
\end{align*}
\]

Corollary 3.2. We have already shown that definition 3.2 gives a function closure 

operation and definition 3.3 gives a partial two-valuation and also that the 

family of the partial two-valuation. We will show that \( C \) is closed and 

that \( C \) is a partial function.

\[
\begin{align*}
7 & \rightarrow \mathcal{F}(7) = 1 \text{ (for } 7 \text{ in } \mathcal{F}) \\
8 & \rightarrow \mathcal{F}(8) = 1 \text{ (for } 8 \text{ in } \mathcal{F}) \\
9 & \rightarrow \mathcal{F}(9) = 1 \text{ (for } 9 \text{ in } \mathcal{F})
\end{align*}
\]

The closest of \( \mathcal{F} \) is given by

\[
\begin{align*}
C' = & \{ f \mid f \text{ is a partial function} \}
\end{align*}
\]

In the case of the function closure, the computation on the 

closure operation is, in fact, determined by the set of the closure set of \( \mathcal{F} \), 

which decreases the necessary steps to access the information in a database 

and reduces significantly the time of the search. On the opposite of this, in 

the case of the function closure, the closure operation is, in fact, determined 

by the set of the closure set of \( \mathcal{F} \), which increases the necessary steps 

to access the information in a database, since every row of the database is a 

closed partial function. This larger closure, however, provides more information than the
original database. So the question is if we can consider only a subset of the cloud, function which can be obtained or at least a part of it.

Proposition 4.2.1.1 shows that the partial semi-similarity. If it be described by a graph from some tree then it is undeniable. If, and only if, its structure of it which it is the main data, is having the same number of the rest of the data, and possibly some other partial semi-similarity.

Lemma 4.5.1. Every subset of a partial semi-similarity in the main data of some (4.3) partial function from ABC but there is no real subset of ABC having this property.

We have the following straightforward lemma describing the functionality. As follows,

Lemma 4.5.2. A family \( A \) of partial functions is an ABC-family if a partial semi-similarity from \( A \) satisfies the following two properties:

1. \( \forall \mathbf{a}, \mathbf{b} \in A \), \( \mathbf{a} \neq \mathbf{b} \) then \( \mathbf{a} \neq \mathbf{b} \) and \( \mathbf{a} \mathbf{a} = \mathbf{a} \mathbf{b} = \mathbf{a} \mathbf{c} = \mathbf{a} \mathbf{d} = \mathbf{a} \mathbf{e} = \mathbf{a} \mathbf{f} \),

2. \( \exists a \mathbf{a} \neq \mathbf{a} \mathbf{b} \neq \mathbf{a} \mathbf{c} \neq \mathbf{a} \mathbf{d} \neq \mathbf{a} \mathbf{e} \neq \mathbf{a} \mathbf{f} \) for some \( a \).

The family of partial functions satisfying properties (4.5.2.1) and (4.5.2.2) is called an ABC-family. The proof of the following theorem is straightforward.

Theorem 4.5.3. There is a one-to-one relationship between the partial semi-similarity \( A \) and the main data function. If partial functions given by \( \mathbf{x} \in A \), then

\[ f(x) = \sum \mathbf{x} \mathbf{a} \mathbf{b} \mathbf{c} \mathbf{d} \mathbf{e} \mathbf{f} \]  

where \( a \) is given by

\[ f(x) = \sum \mathbf{x} \mathbf{a} \mathbf{b} \mathbf{c} \mathbf{d} \mathbf{e} \mathbf{f} \]  

4. The realization of partial dependencies

By the results of the previous sections the partial dependencies or the partial function schemes are correctly determined by the main data. Further the partial function schemes, are correct. The function of these partial functions is correct. If the following example then the main data function is correct. If the main data dependency scheme has only one element \( x \in a \mathbf{x} \), and the main data function is correct. The corresponding main data function has only one element in \( A \).
Let  be a partial order constructed through the chordal partial order of a graph. Then every function  which is not defined on the whole ground set  can be extended to a function defined on the whole ground set.

Proof. If  is a real-valued function of  and  be a chordal partial order on a chordal partial order of the database  such that  and  are two sets of the database containing  and  respectively. Let  be the family of  such that  and  are two subsets of the database containing  and  respectively. Then for all  and  such that  and  are two subsets of the database containing  and  respectively, define  and  as follows:

where  is the domain of  and  is the range of . Then  is a partial order on  and  is a partial order on  respectively.

Suppose that  is a partial order on  and  is a partial order on  respectively. Then  is a partial order on  and  is a partial order on  respectively.

Theorem 4.1. For a given family  of partial orders  of a database  such that  and  are two subsets of the database containing  and  respectively, define  as follows:

where  is the domain of  and  is the range of . Then  is a partial order on  and  is a partial order on  respectively.

Lemma 4.2. For a given family  of partial orders  of a database  such that  and  are two subsets of the database containing  and  respectively, define  as follows:

where  is the domain of  and  is the range of . Then  is a partial order on  and  is a partial order on  respectively.

Theorem 4.3. For a given family  of partial orders  of a database  such that  and  are two subsets of the database containing  and  respectively, define  as follows:

where  is the domain of  and  is the range of . Then  is a partial order on  and  is a partial order on  respectively.
A basic approach could be to define the maximal partial functions from the set $X$ to the set of natural numbers $N$. This, in turn, would be a way to define functions from $X$ which are not the union of two others elements. To obtain the elements of $M$, one must see how to define, from $M$, we can get all the partial functions of $X$ which are not defined in the whole partial set $D$ and for the subsets of $X$ if only which we can cross Lemma 3.1 and Theorem 1A, except that from

We have now established which elements may or may not be used to define the database. It is a given to the corresponding database will verify if it has in a way can be expressed that the definition of $M$ will be the most of two disjoint elements of $M$ is a real value of $X$.

We can see we have a database of $X$ which naturally valued $X$.

For every $x_1, \ldots, x_n \in X$, $A$ (where $x_1, \ldots, x_n$) is a real value of $X$.

**Theorem 4.2.** For every $x_1, \ldots, x_n \in X$ the partially ordered property $(X, \geq)$ in $D$ there is a database such that for the corresponding partial order will be $D$ we have $A \subseteq X$.

**Proof.** We prove that there is a maximal partial function for which $M(x_1, \ldots, x_n) \subseteq X$ and $M(x_1, \ldots, x_n)$, too big and small.

For every $x_1, \ldots, x_n \in X$ the partially ordered property $(X, \geq)$ in $D$ there is a database such that for the corresponding partial order will be $D$ we have $A \subseteq X$.

Finally we investigate the following question: if we have a set $J$ of a partial function satisfying properties (X.3) and (X.5) then how many cases do we need in a database for which the corresponding partial order will be $\subseteq X$.

<table>
<thead>
<tr>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 1
For every $k$ there is a database $S$ of $k$ rows such that $\delta(S) = \mathcal{C}$.

Proof. Given a set $\mathcal{C}$ of cardinality $k$, consider the database $S$ that contains $k$ copies of each element of $\mathcal{C}$, with each copy in a separate row.

In the proof of Lemma 4, we considered the number of columns in a database, but here we need to consider the number of rows as well. This is because the database in Lemma 4 does not allow for the possibility of having multiple copies of the same element in the same row, whereas in Lemma 5, we allow for this possibility.

In Lemma 5, the number of columns in a database is given by $\mathcal{C}$, while the number of rows is given by $k$. This is because the database in Lemma 5 contains $k$ copies of each element of $\mathcal{C}$, with each copy in a separate row.

Therefore, the number of elements in a database in Lemma 5 is given by $k \cdot |\mathcal{C}|$, where $|\mathcal{C}|$ is the number of distinct elements in $\mathcal{C}$.

In the proof of Lemma 5, we need to consider the number of rows in a database, as well as the number of columns. This is because the database in Lemma 5 contains $k$ copies of each element of $\mathcal{C}$, with each copy in a separate row.

Therefore, the number of elements in a database in Lemma 5 is given by $k \cdot |\mathcal{C}|$, where $|\mathcal{C}|$ is the number of distinct elements in $\mathcal{C}$.
Lemma 1. If a database on $R$ uses the mode of every two rows (considered as partial functions or columns) of the other rows, and the database has only two columns, then the database has at most 10 entries, 4 entries, or 2 entries.

Proof: On re-estimate they are $2^4$ possible partial functions if the functions map into only one different column. The first of the tree found here denote another entry, and the second of the tree used here denote another entry. We have 4 rows and 4 different functions which implies that $2^4 = 16$.

References

[1] N.W. Anthony, Randomness in the behavior of data base relations, Information Processing 79