University of Wollongong

Research Online

Faculty of Informatics - Papers (Archive)

Faculty of Engineering and Information
Sciences

2010

Efficient processing of databases with inconsistent information

Janusz Getta

University of Wollongong, jrg@uow.edu.au

Follow this and additional works at: <https://ro.uow.edu.au/infopapers>

Part of the [Physical Sciences and Mathematics Commons](#)

Recommended Citation

Getta, Janusz: Efficient processing of databases with inconsistent information 2010.
<https://ro.uow.edu.au/infopapers/3563>

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

Efficient processing of databases with inconsistent information

Abstract

Abstract—Representation and efficient processing of inconsistent information is a common problem for centralized and distributed database systems. While detection of inconsistencies through offline verification of the consistency constraints is a relatively easy task, a complete elimination of inconsistent information from a database is almost impossible. Therefore, database systems should enable the features that allow for the representation and processing of inconsistent information by the user applications. Implementation of such features require the conceptual extensions to both data model and data manipulation language used for the implementation of a database system. This work presents a formal model for the representation and processing of inconsistent information in a database system. Whenever, the contents of a database fails the evaluation of a consistency constraints then the minimal number valid states of the database is represented by the model. Such approach allows for the flexible representation of various sorts of inconsistencies through the relational tables with variants. The model defines a set of operations on the representation of inconsistent information. This work also reviews the implementation aspects of a database system with inconsistent information.

Disciplines

Physical Sciences and Mathematics

Publication Details

Getta, J. R. (2010). Efficient processing of databases with inconsistent information. 2010 International Conference on Computational Intelligence and Software Engineering, CiSE 2010 (pp. 1-5). Piscataway, New Jersey, USA: IEEE.

Efficient Processing of Databases with Inconsistent Information

Janusz R. Getta

School of Computer Science and Software Engineering
University of Wollongong
Wollongong, NSW
Australia
Email: jrg@uow.edu.au

Abstract—Representation and efficient processing of inconsistent information is a common problem for centralized and distributed database systems. While detection of inconsistencies through offline verification of the consistency constraints is a relatively easy task, a complete elimination of inconsistent information from a database as almost impossible. Therefore, database systems should enable the features that allow for the representation and processing of inconsistent information by the user applications. Implementation of such features require the conceptual extensions to both data model and data manipulation language used for the implementation of a database system.

This work presents a formal model for the representation and processing of inconsistent information in a database system. Whenever, the contents of a database fails the evaluation of a consistency constraints then the minimal number valid states of the database is represented by the model. Such approach allows for the flexible representation of various sorts of inconsistencies through the relational tables with variants. The model defines a set of operations on the representation of inconsistent information. This work also reviews the implementation aspects of a database system with inconsistent information.

I. INTRODUCTION

Storing and processing inconsistent information is an important problem when information from a number of independent sources is integrated into a single multidatabase system. The inconsistencies between the contents of integrated database systems arise due to lack of global coordination of modifications performed on the local database systems. It may happen that modifications of a local database system are not automatically replicated in the other local databases because of no global distributed transaction processing mechanisms implemented over a multidatabase system. While detection of inconsistencies during offline verification of the consistency constraints is a relatively simple task, then a complete elimination of inconsistent information is a difficult problem. As a consequence, inconsistent information remains for a longer time in a global view of a multidatabase system and it has to be processed there in an efficient and clear way by the end-users.

Formally, the presence of inconsistent information in a database manifests itself as a number states a database can be positively validated against a given set of consistent constraints in any moment in time. If a database fails the evaluation of a consistency constraint then a number of different states of

a database may satisfy such constraint. For example, a failed evaluation of a primary key constraint in a relational table by two rows that have the identical values of the primary key creates three states in which either one of the rows is valid or none of them. The present database systems do not provide more sophisticated structures required for the representation of inconsistent information. The efficient processing of a database with inconsistent information is even harder because of a large number of possible states a database that can positively evaluate the consistency constraints. Then, processing of the queries over the data containers with inconsistent information requires repetition of the same query execution plans for many different combinations of consistent states of a database.

Since the progresses of the computer networking technologies made integration of distributed and heterogeneous database systems the implementable reality, a number of research works has been conducted on both practical and theoretical aspects of handling the inconsistencies detected during data integration. These works included the extensions of relational, object-oriented, and semi-structured data models towards storing and processing uncertain and inconsistent data. The works [10], [11] introduced the models of *i-tables* and *m-tables* that partitioned information into three classes: *sure* or *definite information*, *indefinite information*, or *maybe information*. One of the first practical approaches [9] proposed the extensions of relational algebra operations on the relational tables with incompatible attributes. The model of flexible relations [1] considered the data inconsistencies obtained from integration the multiple autonomous relational databases. A paraconsistent relational model described in [6], [14] and later on extended on paraconsistent object-oriented and semi-structured database models [4], [12] defined a paraconsistent relation as a pair of two relational tables where a *positive table* represents all facts know to be true and *negative table* represents all facts known to be false. A nonempty intersection of the positive and negative tables represents the inconsistencies. Another group of solutions is based on the formal logic and logic programming. [8] investigated a problem of finding the consistent answers from an inconsistent database. The same problem addressed by [7], [2], and [3] used a concept of *residue* to derive the consistent answers.

In this work we concentrate on an environment of mul-

tidatabase system [13] that integrates a number of remote and independent database systems. A multidatabase system provides a transparent view of the integrated databases as a homogenous and centralized database system. As a motivating example we consider a domain of suppliers where John, Peter, and Paul supply bolts and nuts. The semantics are expressed through a functional dependency $supplier, part \rightarrow quantity$ and consistency constraint saying that each supplier to be recorded in a database must ship at last one part. Assume that the following sets of tuples have been extracted from the databases d_i and d_j .

d_i	supplier	part	quantity
	John	nut	100
	Peter	bolt	200
	Paul	nut	25
d_j	supplier	part	quantity
	John	nut	100
	Peter	bolt	20
	Paul	bolt	25

It is sure that John supplied 100 nuts. A functional dependency $supplier, part \rightarrow quantity$ indicates that Peter supplied either 200 bolts or Peter supplied 20 bolts. Additionally, when regarding a supplier Paul the following three options are possible: Paul supplied either 25 bolts or 25 nuts or 25 bolts and 25 nuts. The other cases when Peter or Paul supplied nothing are eliminated by the consistency constraint. Totally, six possible states of a database positively validate the functional dependence and the consistency constraint. It means that a relational table that contains information integrated from the databases d_i and d_j can be in six equally probable states. Therefore, when processing the queries each one of the states should be separately considered by a query processor. A problem is how to store information about the large number of database states and how to efficiently process the queries over the large number of database states.

The paper is organized in the following way. The next section introduces a model of database with variant information. A system of operations on the elementary and complex units of inconsistent information is defined in Section 3. Section 4 considers the problems of query processing in a database with variant information. Decomposition of tables with variants is discussed in Section 5. Finally, Section 6 concludes the paper.

II. VARIANTS

This section reviews the concepts of relational database model and introduces a new model of relational tables with variants. Let A be a set of attribute names, and let $dom(a)$ denotes a domain of attribute $a \in A$. A *relational schema* x as a non empty subset of A . A *relational table* $r(x)$ constructed over a schema x is a set of tuples such that each tuple t is a full mapping $t : x \rightarrow \cup_{a \in A} dom(a)$ and $\forall a \in x, t(a) \in dom(a)$.

Let $r(x)$ be a relational table. A variant of relational table $r(x)$ is a pair (v, μ) where

- (i) v is an identifier of the variant,

- (ii) and μ is a full mapping $r(x) \rightarrow \{0, 1\}$

A *relational table with variants*, later on called as *v-table*, is a pair $(r(x), V)$, where V is a non empty set of variants defined over $r(x)$ and such that $\forall t \in r(x), \exists (v, \mu) \in V, \mu(t) = 1$.

Example 2.1

A definition of *v-table* given above allows for its implementation as a pair of horizontally adjusted relational tables. The first element in the pair is a relational table $r(x)$ and the second element is a relational table that represents a set of characteristic functions V . The columns in the second table are the bitmaps the determine which rows belong to each variant. For instance two horizontally adjusted relational tables given below represent a *v-table* defined over a schema $x = \{part, supplier, quantity\}$.

supplier	part	quantity	v_1	v_2	v_3
Peter	bolt	20	0	1	0
Paul	bolt	10	1	0	0
Paul	nut	25	1	1	1

Different interpretations of variants provide different semantics of *v-tables*. In this paper we interpret a *v-table* as a set of relational tables.

Interpretation of a variant (v, μ) of a relational table $r(x)$ is denoted by $int(v, \mu)$ and it is defined as a pair (v, r') where $r' = \{t \in r(x) : \mu(t) = 1\}$.

Interpretation of *v-table* $t = (r(x), V)$ where $V = \{(v_1, \mu_1), \dots, (v_k, \mu_k)\}$ is denoted by $INT(t)$ and it is defined as a set of all pairs (v_i, r_i) where $r_i = int(v_i, \mu_i)$ for $i = 1, \dots, k$.

Such interpretation assumes that only one of the variants represents the correct contents of a database, however at the moment it is not precisely known which one. For example, a (*v-table*) given above represents three possible states of a database where information that Paul supplies 25 nuts is sure because it is valid in all states and the database contains information that either Peter supplies 20 bolts or Paul supplies 10 bolts or no bolts are supplied.

In the other possible interpretations a *v-table* may represent the contents of three integrated databases or a sequence of states of a database.

III. OPERATIONS ON V-TABLES

Query processing and data manipulations on *v-tables* require a system of operations similar to the system of relational algebra operations. We start from the definitions of elementary operations on variants and later we define syntax and evaluation of *expressions on v-tables*, or simply *v-expressions*. Then, we prove that the system of *v-expressions* is a correct representation system and that it satisfies the same properties as relational algebra. Let $\pi, \sigma, \times, \cup, \cap, -$ denote the relational algebra operations of projection, selection, Cartesian product, union, intersection and difference.

Let (v, μ) be a variant such that $\mu : r(x) \rightarrow \{0, 1\}$. Let $y \subseteq x$. Projection of a variant (v, μ) onto scheme y is denoted by $project_y(v, \mu)$ and it is equal to a variant (v, μ') where

$\mu' : \pi_y(r(x)) \rightarrow \{0,1\}$ such that $\forall t \in \pi_y(r(x)), \mu'(t) = \max(\mu(t_1), \dots, \mu(t_k))$ where $\{t_1, \dots, t_k\}$ is a set of all tuples in $r(x)$ such that $t = \pi_y(t_1) = \dots = \pi_y(t_k)$.

Let ϕ be a well-formed formula of the prepositional calculus. Selection from a variant (v, μ) is denoted by $\text{select}_\phi(v, \mu)$ and it is equal to a variant (v, μ') where $\mu' : r(x) \rightarrow \{0,1\}$ such that $\forall t \in \sigma_\phi(r(x)), \mu'(t) = v(t)$, otherwise $v'(t) = 0$.

Consider the relational variants $(v, \mu_v), \mu_v : r(x) \rightarrow \{0,1\}$ and $(w, \mu_w), \mu_w : s(x) \rightarrow \{0,1\}$. Union of the variants (v, μ_v) and (w, μ_w) is denoted by $\text{union}((v, \mu_v), (w, \mu_w))$ and it is equal to a variant $(u, \mu_u), \mu_u : r(x) \cup s(x) \rightarrow \{0,1\}$ such that $\forall t \in r(x) - s(x), \mu'_u(t) = \mu_v(t), \forall t \in s(x) - r(x), \mu'_u(t) = \mu_w(t), \forall t \in r(x) \cap s(x), \mu'_u(t) = \max(\mu_v(t), \mu_w(t))$.

Difference of the variants (v, μ_v) and (w, μ_w) is denoted by $\text{difference}((v, \mu_v), (w, \mu_w))$ and it is equal to a variant (u, μ_u) , where $\mu_u : r(x) \rightarrow \{0,1\}$ such that $\forall t \in r(x) - s(x), \mu_u(t) = \mu_v(t)$, otherwise $\mu_u(t) = 0$.

Consider the relational variants $(v, \mu_v), \mu_v : r(x) \rightarrow \{0,1\}$ and $(w, \mu_w), \mu_w : s(y) \rightarrow \{0,1\}$. Cartesian product of the variants (v, μ_v) and (w, μ_w) is denoted by $\text{product}((v, \mu_v), (w, \mu_w))$ and it is equal to a variant (u, μ_u) where $\mu_u : r(x) \times s(y) \rightarrow \{0,1\}$ such that $\forall t \in r(x) \times s(y), \exists t_r \in r(x), \exists t_s \in s(y), t = t_r \bullet t_s$ and $\mu_u(t) = \min(\mu_v(t_r), \mu_w(t_s))$ where $t_r \bullet t_s$ denotes concatenation of the tuples t_r, t_s .

The operations of join and antijoin of variants can be defined as the expressions over the operations of product, selection, and projection. The operations of intersection and difference of variants are the special cases of the operations of join and antijoin.

Join of variants is defined as an expression: $\text{join}((v, \mu_v), (w, \mu_w)) = \text{select}_\phi(\text{product}((v, \mu_v), (w, \mu_w)))$ where $\phi = (x.a_i = y.a_i)$ and...and $(x.a_j = y.a_j)$ and the attributes $a_i, \dots, a_j$ are common to the schemas x and y .

Antijoin of variants is defined as an expression: $\text{antijoin}((v, \mu_v), (w, \mu_w)) = \text{project}_x(\text{select}_\phi(\text{product}((v, \mu_v), (w, \mu_w))))$ where $\phi = (x.a_i \neq y.a_i)$ or ...or $(x.a_j \neq y.a_j)$.

Example 3.1

Consider the variants (v, μ_v) and (w, μ_w) where $\mu_v = \{(a, 0), (b, 1)\}, \mu_w = \{(c, 1), (d, 1)\}$. Then, $\text{product}((v, \mu_v), (w, \mu_w)) = (u, \mu_u)$ where $\mu_u = \{(ac, 0), (ad, 0), (bc, 1), (bd, 1)\}$.

Let ϕ be a well-formed formula of prepositional calculus with the names of all its variables included in a set of attribute names x . Let c, d be v -tables. The language of v -expressions is defined as follows.

- (i) $\Pi_x(c), \Sigma_\phi(c), c \cup d, c \bowtie d, c \sim d$ are v -expressions,
- (ii) if e, f are v -expressions then $(e), \Pi_x(e), \Sigma_\phi(e), e \cup f, e \bowtie f, e \sim f$ are v -expressions,
- (iii) no other expression is a valid v -expression.

Let $F(c_1, \dots, c_k)$ be a v -expression constructed over v -tables $c_1 = (r_1, V_1), \dots, c_k = (r_k, V_k)$.

Computation of $F(c_1, \dots, c_k)$ is denoted by $|F(c_1, \dots, c_k)|$ and it is defined as $|F(c_1, \dots, c_k)| = (r, V)$ such that:

- (i) $V = \{(v, \mu) : \exists (v_1, \mu_1) \in V_1, \dots, \exists (v_k, \mu_k) \in V_k, (v, \mu) = f((v_1, \mu_1), \dots, (v_k, \mu_k))\}$ where f is an expression over relational variants obtained from $F(c_1, \dots, c_k)$ by systematic replacement of the operations $\Pi, \Sigma, \cup, \bowtie, \sim$ with the respective operations on variants: **project**, **select**, **union**, **join**, **antijoin** and v -tables $c_1, \dots, c_k$ with the variants $(v_1, \mu_1), \dots, (v_k, \mu_k)$.
- (ii) $r = \cup_{(v, \mu) \in V} \text{dom}(\mu)$ where $\text{dom}(\mu)$ is a domain of mapping μ , i.e. a relational table on which a variant (v, μ) is constructed. At the end we remove from r all tuples t such that $\forall (v, \mu) \in V, \mu(t) = 0$.

It is important to note that evaluation of v -expressions is performed in a different way from evaluation of relational algebra expressions. Evaluation of relational algebra expression is performed through the successive evaluations of the innermost operations, and by systematic replacement of the innermost operations and their arguments with the results. In our case, entire v -expression is considered as one multi-argument operation. Then, v -expression is independently evaluated for all combinations of the database variants. Whenever an argument of v -expression is used in more than one place all its instances are replaced with the same variant. Such method of evaluation preserves all properties of the standard relational algebra and allows for syntax-based query optimization.

Example 3.2

Consider the v -tables sp and mp given below. Figure 3 contains the result of $|\Pi_{\text{manufacturer, supplier}}(sp \bowtie mp)|$ where the new variants are the following combinations of the argument variants: $v_6 = [v_1 v_4], v_7 = [v_2 v_4], v_8 = [v_3 v_4], v_9 = [v_1 v_5], v_{10} = [v_2 v_5], v_{11} = [v_3 v_5]$.

sp

supplier	part	v_1	v_2	v_3
Peter	bolt	1	0	0
Paul	bolt	0	1	0
Paul	screw	0	1	0

mp

manufacturer	part	v_4	v_5
John	bolt	1	0
James	bolt	0	1

$|\Pi_{\text{manufacturer, supplier}}(sp \bowtie mp)|$

manufacturer	supplier	v_6	v_7	v_8	v_9	v_{10}	v_{11}
John	Peter	1	0	0	0	0	0
John	Paul	0	1	0	0	0	0
James	Peter	0	0	0	1	0	0
James	Paul	0	0	0	0	1	0

Figure 3. Evaluation of v -expression

IV. EFFICIENT EVALUATION OF v -expressions

Query processing in database systems with variant information is similar to query processing in traditional relational

database systems. A query expressed in a high-level declarative query language is initially translated into an expression of relational algebra. If all of its arguments contain variants then the query is translated into *v-expression* through the replacement of the relational arguments with *v-tables* and the relational algebra operations with the respective operations on *v-tables* or combination of such operations. A case when only some of the arguments contain variant data is considered in the next section. Then syntax-based optimization techniques applicable to the relational algebra expressions, like for instance pushing selection down a syntax tree are also applied to *v-expressions*. Evaluation of *v-expressions* through the successive computations of variant subexpressions for all combinations of variants is not realistic because computation of variant expression requires multiple accesses to a data part of each *v-table*. This section presents more efficient evaluation techniques of *v-expressions*.

Evaluation of *v-expression* $F(c_1, \dots, c_k)$ must be performed for all combination of variants included in the *v-tables* $c_1, \dots, c_k$. Unfortunately, re-evaluation of entire *v-expression* for each combination of variants is very ineffective because the replacement of only one variant in a set of parameters of *v-expression* requires reprocessing of almost all operations on the same variants. If all arguments in the expression $F(c_1, \dots, c_k)$ are different then it can be computed in the same way as relational algebra expressions, i.e. in operation-by-operation mode. Such approach allows to store the intermediate results of the computations and re-use these result when *v-expression* is re-evaluated. For any *v-expression* with unique arguments it is true that :

- (i) $|\Sigma_\phi(e(c_1, \dots, c_k))| = |\Sigma_\phi(|e(c_1, \dots, c_k)|)|$
- (ii) $|\Pi_x(e(c_1, \dots, c_k))| = |\Pi_x(|e(c_1, \dots, c_k)|)|$

Additionally if ω is a binary operation on *v-tables*, and $e(c_1, \dots, c_m) \omega f(d_1, \dots, d_n)$ is an *v-expression with unique arguments*. Then it is true, that: $|e(c_1, \dots, c_m) \omega f(d_1, \dots, d_n)| = ||e(c_1, \dots, c_m)| \omega |f(d_1, \dots, d_n)||$

These properties of *v-expressions* with unique arguments allow for its evaluation in the same way as evaluation of a respective expressions of relational algebra i.e. in an operation by operation mode assuming that evaluation of the single operations in *v-expression* is performed in the way described above.

The efficient computations of *v-expressions non-unique arguments* require few simple modifications to the model. A *v-expression with non-unique arguments*, like for example $c - (c - d)$ cannot be evaluated in the same way as *v-expression* with unique arguments because of an assumption that only one variant in a *v-table* is true. Then, it may happen that independent computations of the difference operations use different variants taken from the first and the second copy of *v-table* c . To avoid this, the signatures of variants must be considered during the computations of *v-expressions* with non-unique arguments. A signature of a variant is a set of identifiers of variants included in the original *v-tables*. The signatures s_i and s_j are contradictory if $r.v_m \in s_i$ and $r.v_n \in s_j$, i.e. if the signatures include different identifiers from the same

v-table. In order to efficiently evaluate a *v-expression* with non-unique arguments the unary and binary operations on *v-tables* must take under the consideration the non-contradictory signatures of the variants included in the intermediate results created during the evaluation.

V. DECOMPOSITION OF *v-tables*

An assumption that inconsistent information related to a particular relational table can be represented by a single *v-table* is rather inconvenient from the implementation point of view. Usually it is such that only a very limited number of rows does not belong to all variants and the majority of the rows represent sure and consistent information. The large *v-tables* are inconvenient both from conceptual and implementation points of view. It is difficult to explain the semantics of *v-tables* that contain variants where only some rows do not the members of a variant, i.e. the columns that represent characteristic function contain a lot of 1s. Additionally, the operations on the large *v-tables* require more space and time than operations on smaller *v-tables*. This section shows that decompositions of large *v-tables* may improve the efficiency of query processing.

A horizontal decomposition δ of *v-table* $c = (r(x), V)$ is defined as a set of formulas $\{\phi_1, \dots, \phi_k\}$ such that $\sigma_{\phi_1}(r(x)) \cup \dots \cup \sigma_{\phi_k}(r(x)) = r(x)$.

We say that decomposition δ is a nonloss when $|\Sigma_{\phi_1}(c) \cup \dots \cup \Sigma_{\phi_k}(c)| = c$. For example, a decomposition $\delta(\phi_{const}, \phi_{var})$ where a formula ϕ_{var} selects all rows for which the values of all variants are equal to 1 while $\phi_{var} = \text{not}(\phi_{const})$ is a nonloss decomposition. Any *v-table* c can be horizontally decomposed in a nonloss way into a pair of *v-tables* that separately represents its *constant* (c_c), and *variable* (c_v) components, i.e. $c = c_c \cup c_v$. Such decomposition eliminates the redundancies from the representations of variants through characteristic functions and allows for more efficient evaluation of *v-expressions*. Below, we show that the majority of operations on *v-tables* can be computed independently on their *constant* and *variable* components.

Consider *v-tables* c, d and their decomposition into the constant and variable components components $c = c_c \cup c_v$ and $d = d_c \cup d_v$. Then

- (i) $|\Sigma_\phi(c)| = |\Sigma_\phi(c_c)| \cup |\Sigma_\phi(c_v)|$
- (ii) $|\Pi_x(c)| = |\Pi_x(c_c)| \cup |\Pi_x(c_v)|$
- (iii) $|c \cup d| = |c_c \cup d_c| \cup |c_v \cup d_v|$
- (iv) $|c \cap d| = |c_c \cap d_c| \cup |(c_c \cap d_v) \cup (c_v \cap d_c) \cup (c_v \cap d_v)|$
- (v) $|c \times d| = |c_c \times d_c| \cup |(c_c \times d_v) \cup (c_v \times d_c) \cup (c_v \times d_v)|$

Moreover, note that the operations on the constant parts of *v-tables* i.e. $\Sigma_\phi(c_c)$, $\Pi_x(c_c)$, $c_c \cup d_c$, $c_c \cap d_c$ and $c_c \times d_c$ are equivalent to respective operations of relational algebra. For instance, *v-expressions* $\Sigma_\phi(c \cap d)$ can be evaluated by independent computation of selection on the *constant* and *variable* components of c and d and then by application of equality (iv). For difference operation the respective equality is as follows.

- (vi) $|c - d| = |c_c - d_c| - d_v| \cup |c_v - d_c| - d_v|$

In this case a constant component cannot be computed completely independently on a respective variable part. However, it is possible to compute first constant component of the result through evaluation of expression $c_c - d_c$ and later on find how it is affected by a variable component in d_v .

Decomposition of *v-tables* into the constant and variable components eliminates the redundancies and speeds up query processing. because the majority of data belongs to the constant components and for these components there is no need to represent and to process the variants. Processing of the constant components is implemented through the ordinary evaluation of the relational algebra expressions. Processing of the operations on the mixtures of constant and variable components can be done faster because of the expected small size of imprecise components which allow to permanently keep them in the fast transient memory. Decomposition of *v-tables* also allows for the unified evaluation of expressions where only some of the arguments contain variant data and the rest are the plain relational tables.

VI. SUMMARY AND CONCLUSION

This work considers a problem of efficient processing of databases with inconsistent information. The existence of inconsistent information is inevitable in the multidatabase systems that provide integrated access to the various independent sources of information and that allow for storing and processing historical information and variants of informations. The paper introduces a concept of *v-table* as an elementary unit of inconsistent information. A language *v-expressions* is proposed for querying and for data manipulations on *v-tables*. The language is syntactically compatible with a system of operation of relational algebra, even though evaluation rules of *v-expressions* is different. The proposed system of representation and manipulation on variants of information satisfies the requirements of correct representation system and it also satisfies the equational properties of relational algebra. These properties of *v-expressions* allow for the same syntax based optimization of *v-expressions* as for the optimizations of relational algebra expressions.

Identification of a class of *v-expressions* computable in the same way as the relational algebra expressions is another important contribution of this work. It has an important impact on efficiency of query processing in the systems with variants of information. We show that in a general case the computations of any *v-expression* require the variants of information to preserve unique identification system during the computations.

Decomposition of variants of information is proposed as another method for efficiency improvements in query processing. In particular we considered decomposition of *v-tables* into "constant" and "variable" components. It has been shown that for a subclass of *v-expressions* with no set difference operation it is possible to independently compute "constant" and "variable" components of an application.

An area that needs further work concerns the impact of the other decomposition methods on efficiency of query processing. For example, decomposition of *v-tables* into "constant"

"maybe" "exclusive" and the other more specific sorts of variants information may provide the chances for more efficient query processing. A vertical decomposition of the variant parts of *v-tables* is another idea that also needs to be considered.

REFERENCES

- [1] S. S. Agarwal and A. Keller and G. Wiederhold and S. Saraswat, *Flexible relation: An approach for integrating data from multiple, possibly inconsistent databases*, Proceedings of IEEE Data Engineering Conference, 1995.
- [2] M. Arenas and L. E. Bertosi and J. Chomicki, *Consistent query answers in inconsistent database*, 1999, 68-79.
- [3] M. Arenas and L. E. Bertosi and M. Kifer, *Applications of annotated predicate calculus to querying inconsistent databases*, Proceedings of the First International Conference on Computational Logic, 2000.
- [4] R. Bagai, S. J. Kelley, *Paraconsistency in object-oriented databases*, Soft-Ware 2002: Computing in an Imperfect World, First International Conference, 2002, 141-150.
- [5] R. Bagai and M. A. Orgun, *A temporal paraconsistent relational algebra for incomplete and inconsistent information*, Proceedings of the 33rd Annual ACM Southeast Conference, 1995, 240-248.
- [6] R. Bagai and R. Sunderraman, *A paraconsistent relational data model*, International Journal of Computer Mathematics, 1995, vol. 55, 39-55.
- [7] P. Barcel and L. Bertossi, *Logic programs for querying inconsistent databases*, Proceedings of 5th Int. Symp. on Practical Aspects of Declarative Languages, 2003, 208-222.
- [8] F. Bry, *Query answering in information systems with integrity constraints*, IFIP WG 11.5 Working Conf. on Integrity and Control in Information System, 1997.
- [9] L. G. DeMichiel, *Resolving database incompatibility: An approach to performing relational operations over mismatched domains*, IEEE Transactions on Knowledge and Data Engineering, vol. 1, no. 4, 1989, 485-493.
- [10] K-C. Liu and R. Sunderraman, *Indefinite and maybe information in relational databases*, ACM Transactions on Database Systems, vol. 15, no.1, 1990, 1-39.
- [11] K-C. Liu and R. Sunderraman, *A generalized relational model for indefinite and maybe information*, IEEE Transactions on Knowledge and Data Engineering, vol. 3, no. 1, 1991, 65-77.
- [12] L. Mengchi and W. L. Tok, *A data model for semistructured data with partial and inconsistent information* International Conference on Extending Database Technology, 2000, 317-331.
- [13] *Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases*, ACM Computing Surveys, 1990, vol. 22, no. 1, 183-236.
- [14] Nicholas Q. Trân and Rajiv Bagai, *Efficient representation and algebraic manipulations of infinite relations in paraconsistent database*, Information Systems, 2000, vol. 25, no. 8, 491-502.
- [15] P. S. M. Tsai and A. L. P. Chen, *Querying Uncertain Data in Heterogeneous Databases*, Proceedings. of the IEEE International Workshop on Research Issues in Data Engineering, 1993.