Generalization of the classical result of Codd-Lacroix-Pirotte

IRYNA GLUSHKO
Applied Mathematics, Informatics and Educational Measurement Department
Nizhyn Mykola Gogol State University
16600, Nizhyn, Kropyvyanskoho str, 2
UKRAINE
iryna.glushko@ndu.edu.ua

Abstract: - The paper is focused on some theoretical questions of the Database Theory. The result which concerns equivalence of table algebra for infinite tables and corresponding relational calculi is presented. This result generalizes the classical result about the equivalence of Codd's relational algebra and tuple (domain) relation calculus. Concept of table (relation) is considered in terms of nominal sets. Under relation is understood any set of tuples (with common scheme), in particular infinite. Furthermore only one universal domain is considered. The classical relational calculi are filled up by functional and predicate signatures on the universal domain (while usually consider only binary predicates and functional signature is generally empty).

Key-Words: - Relation databases, tuple relation calculus, domain relation calculus, table algebra, nominal sets.

1 Introduction

Today, Database Management Systems or DBMS are used in almost all spheres of human activities that related to the preservation and processing of information. The development of database technology is largely based on the relational model of data proposed by E. F. Codd in 1970. The Relational Algebra was introduced by E. F. Codd as a set of operators on the relations [1]. The scientist also defined a tuple relation calculus and presented an algorithm for reducing an arbitrary relation-defining expression (based on the calculus) into a semantically equivalent expression of the relational algebra [2].

A little later M. Lacroix and A. Pirotte (1977) suggested an alternative version of tuple relational calculus – domain relational calculus [3]. In this version variables represent single domain values rather than entire tuples.

A set of relational algebra operations proposed by E.F. Codd, in course of time was expanded to meet the needs of query languages. A. Klug (1982) extended relational algebra and relational calculus to include aggregate functions and showed equivalence thus obtained two formal languages [4].

Three principal approaches to the design of query languages are discussed by J.D. Ullman (1982) in [5]: relational algebra, tuple relational calculus and domain relational calculus. Author restricted the relational calculi for use only finite relations, i.e. infinite relations are not considered. The restricted relational calculi expressions are called "safe". In this case, the equivalence of relational algebra and corresponding relational calculi in which consideration is restricted to only safe expressions is proved.

The issue of the equivalence of relational algebra and relational calculi is also considered by D. Maier (1983) [6]. Author discussed three query systems: tuple relational calculus, domain relational calculus and tableau queries. It was shown that both tuple relational calculus and domain relational calculus are equivalent in expressive power to relation algebra. D. Maier introduced two interpretations for formulas of both tuple relational calculus and domain relational calculus which called unlimited and limited interpretations. A class of safe expressions for which both interpretations always yield the same value is introduced too. However, D. Maier offers the readers to prove items of some theorems. It is not good, because the readers are unable to verify their proofs.

Specifying of table (relation) in terms of nominal sets is carried out by V. Redko, J. Brona, D. Buy, S. Poliakov [7]. Traditionally the finite set of tuple is understood under the table and the authors take it into account. However, as a rule, mathematical statements about standard properties of specification of relation operations remain true for infinite relations. Further under relation we will understand any set of tuples (with common scheme), in particular infinite. This raises the problem of the equivalence of table (relational) algebra and tuple (domain) calculus. In this paper the solution of this problem is proposed.

2 Table Algebras of Infinite Tables and Generalized Relational Calculi

Among the two sets that are considered, $\boldsymbol{A}$ is the set of attributes and $\boldsymbol{D}$ is the universal domain.

Definition 1. An arbitrary (finite) set of attributes $R \subseteq \boldsymbol{A}$ is called the scheme.

Definition 2. A tuple of the scheme R is a nominal set on pair R, $\boldsymbol{D}$. The projection of this nominal set for the first component is equal to R.

In other words, a tuple of the scheme R is a function $s: R \to \boldsymbol{D}$.

Definition 3. A table of scheme R ($R \subseteq \boldsymbol{A}$) is pair $\langle t, R \rangle$, where t is a set (in particular infinite) of tuples of fixed scheme R.

Thus, a certain scheme is ascribed to every table.

The set of all tuples (tables) on scheme R is designated as $S(R)$ ($\boldsymbol{T}(R)$ respectively) and the set of all tuples (tables) is designated as S ($\boldsymbol{T}$ respectively). Hence, $\boldsymbol{T}(R) = P(S(R))$, $S = \bigcup_{R \subseteq \boldsymbol{A}} S(R)$, $\boldsymbol{T} = \bigcup_{R \subseteq \boldsymbol{A}} \boldsymbol{T}(R)$, where $P(A)$ is a power set of the set A.

Definition 4. The table algebra of infinite tables is the algebra $\langle \boldsymbol{T}, \Omega_{P,\Xi} \rangle$, where $\boldsymbol{T}$ is the set of all tables, $\Omega_{P,\Xi} = \{\bigcup_R, \bigcap_R, \setminus_R, \sigma_{p,R}, \pi_{X,R}, \otimes_{R_1,R_2}, \div_{R_2}^{R_1}, Rt_{\xi,R}, \sim_R\}$ is the signature, $p \in P, \xi \in \Xi$, $X, R, R_1, R_2 \subseteq \boldsymbol{A}$, P, Ξ are the sets of parameters. The operations of signature $\Omega_{P,\Xi}$ are defined in [8].

Lemma 1. Any expression over table algebra of infinite tables can be replaced by equivalent to him expression which uses only operations of selection, join, projection, union, difference and renaming.

Proof. To prove the first statement we will show that operations of intersection, division, active complement can be expressed through the operations noted in formulation of lemma. Indeed, the following equalities hold:

1) $\langle t_1, R \rangle \bigcap_R \langle t_2, R \rangle = \langle t_1, R \rangle \setminus_R (\langle t_1, R \rangle \setminus_R \langle t_2, R \rangle)$;

2) $\langle t_1, R_1 \rangle \div_{R_2}^{R_1} \langle t_2, R_2 \rangle = \pi_{R',R_1}(\langle t_1, R_1 \rangle) \setminus_{R'} \setminus_{R'} \pi_{R',R_1}(\pi_{R',R_1}(\langle t_1, R_1 \rangle) \otimes_{R',R_2} \langle t_2, R_2 \rangle \setminus_{R_1} \langle t_1, R_1 \rangle)$, where $R_2 \subseteq R_1$, $R' = R_1 \setminus R_2$;

3) $\sim_R \langle t, R \rangle = C(\langle t, R \rangle) \setminus_R \langle t, R \rangle$, where $C(\langle t, R \rangle) = \pi_{A_1,R}(\langle t, R \rangle) \otimes_{\{A_1\},\{A_2\}} \ldots \otimes_{\{A_1,\ldots,A_{n-1}\},\{A_n\}} \pi_{A_n,R}(\langle t, R \rangle)$, and $R = \{A_1, \ldots, A_n\}$ is a scheme of the table $\langle t, R \rangle$ (see, for example, [6], [7]).

3 Generalized Relational Calculi

Relational calculus is the basis of most relational query languages because unlike relational (table) algebra, calculus expresses only what must be the result, and does not determine how to get it. Relational calculus is based on first-order predicate calculus. There are two forms of relational calculus: tuple calculus and domain calculus. These forms have been proposed by E. Codd [2] and M. Lacroix and A. Pirotte [3] respectively.

3.1 Generalized tuple relational calculus

Consider generalized tuple relational calculus. In the classical tuple relational calculus and domain relational calculus only binary predicates usually consider and functional signature is generally empty. In this paper tuple relational calculus is extended by arbitrary predicate and functional signatures on the universal domain $\boldsymbol{D}$.

As known, tuple relational calculus builds its expressions from tuples. A tuple relational calculus expression looks like as $\{x(R) \mid P(x)\}$, where P is a predicate over tuple variable x, and R is a scheme. This expression indicates table $\langle t, R \rangle$, $t \in T(R)$ that contains tuples on which predicate P is true.

The set of legal tuple relational calculus formulas will be defined relative to:

- a set of attributes $\boldsymbol{A}$ and universal domain $\boldsymbol{D}$;
- a set of object variables (tuple variables) $x_1, x_2, \ldots$;
- a set of object constants $d_1, d_2, \ldots$;
- a set of function symbols $f_1^{n_1}, f_2^{n_2}, \ldots, n_i \geq 1$;
- a set of predicate symbols $p_1^{m_1}, p_2^{m_2}, \ldots, m_i \geq 1$;
- a set of constant tables symbols along with their schemes; constant tables are denoted as $\langle t, R \rangle$;
- a set of variable tables symbols along with their schemes; variable tables are denoted as $\langle X, R \rangle$.

The universal domain $\boldsymbol{D}$ is the domain of interpretation of object constants, and the set of all tuples over $\boldsymbol{D}$ is the domain of interpretation of object variables. We use $\boldsymbol{x}$ as syntactic variable, the domain of change of which is the set of variables; $\boldsymbol{f}$ as syntactic variable, the domain of change of which is the set of function symbols; $\boldsymbol{p}$ as syntactic variable, the domain of change of which is the set of predicate symbols; $\boldsymbol{d}$ as syntactic variable, the

domain of change of which is the set of constants; $\mathcal{A}$ as syntactic variable, the domain of change of which is the set of attributes.

Definition 5. The following expressions are terms (induction on length of terms):

a) $\boldsymbol{d}$ is a term;

b) $\boldsymbol{x}(\mathcal{A})$ is a term;

c) if $\boldsymbol{u}_1,...,\boldsymbol{u}_n$ are terms, $\boldsymbol{f}$ is a function symbol of arity n then $\boldsymbol{f}(\boldsymbol{u}_1,...,\boldsymbol{u}_n)$ $\boldsymbol{f}(\boldsymbol{u}_1,...,\boldsymbol{u}_n)$ is a term;

d) an expression is a term if and only if it can be shown to be a term on the basis of conditions a), b) and c).

We use $\boldsymbol{u}$ as syntactic variable, the domain of change of which is the set of terms. We will formulate the rules of formulas construction.

Definition 6. There are three kinds of atomic formulas (atoms):

a1. For any constant table $\langle t, R\rangle$ and for any tuple variable $\boldsymbol{x}$, $t_R(\boldsymbol{x})$ is an atom. $t_R(\boldsymbol{x})$ stands for $\boldsymbol{x} \in \langle t, R\rangle$.

a2. For any variable table $\langle X, R\rangle$ and for any tuple variable $\boldsymbol{x}$, $X_R(\boldsymbol{x})$ is an atom. $X_R(\boldsymbol{x})$ stands for $\boldsymbol{x} \in \langle X, R\rangle$.

a3. For any terms $\boldsymbol{u}_1,...,\boldsymbol{u}_m$, and for any predicate $\boldsymbol{p}$ of arity m on the universal domain $\boldsymbol{D}$, $\boldsymbol{p}(\boldsymbol{u}_1,...,\boldsymbol{u}_m)$ is an atom.

We use the connectives $\neg$, $\wedge$, $\vee$, quantifiers $\exists, \forall$ and brackets () to build formulas from atoms. We use $\boldsymbol{P}$, $\boldsymbol{Q}$ and $\boldsymbol{G}$ as syntactic variables, the domain of change of which is the set of formulas.

Definition 7. The following expressions are formulas (induction on length of formulas):

f1. Every atom is a formula.

f2. If $\boldsymbol{P}$ is a formula, then $\neg\, \boldsymbol{P}$ is a formula.

f3. If $\boldsymbol{P}$ i $\boldsymbol{Q}$ are formulas, then $(\boldsymbol{P} \wedge \boldsymbol{Q})$, $(\boldsymbol{P} \vee \boldsymbol{Q})$ are formulas.

f4. If $\boldsymbol{x}$ is a tuple variable, $\boldsymbol{P}$ is a formula, $R \subseteq \boldsymbol{A}$ is a scheme, then $\exists \boldsymbol{x}(R)\boldsymbol{P}$ is a formula.

f5. If $\boldsymbol{x}$ is a tuple variable, $\boldsymbol{P}$ is a formula, $R \subseteq \boldsymbol{A}$ is a scheme, then $\forall \boldsymbol{x}(R)\boldsymbol{P}$ is a formula.

f6. If $\boldsymbol{P}$ is a formula, then $(\boldsymbol{P})$ is a formula.

f7. There are no other formulas.

In general case tuple variables can be free or bound in a formula. Sense of these concepts is the same as well as in the predicate calculus: an occurrence of a variable x is said to be bound in a formula if either it is the occurrence of x in a quantifier ($\forall \boldsymbol{x}$ or $\exists \boldsymbol{x}$) in a formula or it lies within the scope of a quantifier ($\forall \boldsymbol{x}$ or $\exists \boldsymbol{x}$) in a formula. Otherwise, the occurrence is said to be free in a formula. A variable is said to be free (bound) in a formula if it has a free (bound) occurrence in a formula (see, for example, [9]).

A scheme (a finite set of attribute) $scheme(\boldsymbol{x}, \boldsymbol{P})$ and a set of attributes with which tuple variable $\boldsymbol{x}$ occurs in a formula $\boldsymbol{P}$, $attr(\boldsymbol{x}, \boldsymbol{P})$, are defined for every tuple variable $\boldsymbol{x}$. The expressions $scheme(\boldsymbol{x}, \boldsymbol{P})$ and $attr(\boldsymbol{x}, \boldsymbol{P})$ are defined only when $\boldsymbol{x}$ has a free occurrence in formula $\boldsymbol{P}$, and takes place including $attr(\boldsymbol{x}, \boldsymbol{P}) \subseteq scheme(\boldsymbol{x}, \boldsymbol{P})$ (that follow from subsequent determinations, on condition of determination of expressions).

We use the concepts of free and bound variables, the scheme, and the set of attributes with which tuple variable occurs in a formula to define the class of legal formulas.

Definition 8. We will define expression *attr* for terms first:

1. if $\boldsymbol{u} = \boldsymbol{d}$, then $attr(\boldsymbol{x}, \boldsymbol{u}) = \varnothing$;
2. if $\boldsymbol{u} = \boldsymbol{x}(\mathcal{A})$, then $attr(\boldsymbol{x}, \boldsymbol{u}) = \{\mathcal{A}\}$, and $attr(\boldsymbol{x}, \boldsymbol{y}(\mathcal{A})) = \varnothing$, where $\boldsymbol{x} \neq \boldsymbol{y}$;
3. if $\boldsymbol{u} = \boldsymbol{f}(\boldsymbol{u}_1,...,\boldsymbol{u}_n)$, where $\boldsymbol{u}_i$ are terms then $attr(\boldsymbol{x}, \boldsymbol{u}) = \bigcup_{i=1}^{n} attr(\boldsymbol{x}, \boldsymbol{u}_i)$.

Definition 9. Consider the cases where $\boldsymbol{P}$ is an atomic formula, then

a1. if $\boldsymbol{P} = t_R(\boldsymbol{x})$, then $\boldsymbol{x}$ is free in $\boldsymbol{P}$ and $scheme(\boldsymbol{x}, \boldsymbol{P}) = attr(\boldsymbol{x}, \boldsymbol{P}) = R$;

a2. if $\boldsymbol{P} = X_R(\boldsymbol{x})$, then $\boldsymbol{x}$ is free in $\boldsymbol{P}$ and $scheme(\boldsymbol{x}, \boldsymbol{P}) = attr(\boldsymbol{x}, \boldsymbol{P}) = R$;

a3. if $\boldsymbol{P} = \boldsymbol{p}(\boldsymbol{u}_1,...,\boldsymbol{u}_m)$, where $\boldsymbol{u}_i$ are terms, and $\boldsymbol{x}_1,...,\boldsymbol{x}_k$ are all variables of these terms, then this tuple variables are free in formula $\boldsymbol{P}$, $scheme(\boldsymbol{x}_i, \boldsymbol{P})$ is undefined, and $attr(\boldsymbol{x}_i, \boldsymbol{P}) = \bigcup_{j=1}^{m} attr(\boldsymbol{x}_i, \boldsymbol{u}_j)$, $i = 1,...,k$.

Atomic formulas are all legal.

Definition 10. The construction of all legal formulas proceeds by induction on the length of formulas. Assume $\boldsymbol{G}$ and $\boldsymbol{Q}$ are both legal formulas.

f2. If $\boldsymbol{P} = \neg \boldsymbol{G}$, then $\boldsymbol{P}$ is legal, and all occurrences of variables in $\boldsymbol{P}$ free or bound as they are in $\boldsymbol{G}$. For every variable $\boldsymbol{x}$ that occurs free in $\boldsymbol{G}$,

$scheme(\boldsymbol{x},\boldsymbol{P}) \simeq scheme(\boldsymbol{x},\boldsymbol{G})$ and $attr(\boldsymbol{x},\boldsymbol{P}) = attr(\boldsymbol{x},\boldsymbol{G})$.

f3. If $\boldsymbol{P} = \boldsymbol{G} \wedge \boldsymbol{Q}$ or $\boldsymbol{P} = \boldsymbol{G} \vee \boldsymbol{Q}$, then all occurrences of variables in $\boldsymbol{P}$ are free or bound as their corresponding occurrences are in $\boldsymbol{G}$ and $\boldsymbol{Q}$. Assume variable $\boldsymbol{x}$ occurs free in subformulas $\boldsymbol{G}$ and/or $\boldsymbol{Q}$. Define the scheme, and the set of attributes with which tuple variable $\boldsymbol{x}$ occurs in a formula for formula $\boldsymbol{P}$. Next cases take place.

a. The schemes of formulas $scheme(\boldsymbol{x},\boldsymbol{G})$ and $scheme(\boldsymbol{x},\boldsymbol{Q})$ are both defined. Formula $\boldsymbol{P}$ is legal if equality $scheme(\boldsymbol{x},\boldsymbol{G}) = scheme(\boldsymbol{x},\boldsymbol{Q})$ holds. After determination $scheme(\boldsymbol{x},\boldsymbol{P}) = scheme(\boldsymbol{x},\boldsymbol{G})$.

b. The scheme is defined for only one subformula. Assume $scheme(\boldsymbol{x},\boldsymbol{G})$ is defined, and $scheme(\boldsymbol{x},\boldsymbol{Q})$ is undefined. Then $attr(\boldsymbol{x},\boldsymbol{Q}) \subseteq scheme(\boldsymbol{x},\boldsymbol{G})$ must hold for $\boldsymbol{P}$ to be legal. After determination $scheme(\boldsymbol{x},\boldsymbol{P}) = scheme(\boldsymbol{x},\boldsymbol{G})$.

c. The scheme is undefined for both subformulas, then $scheme(\boldsymbol{x},\boldsymbol{P})$ is undefined.

In all cases $attr(\boldsymbol{x},\boldsymbol{P}) = attr(\boldsymbol{x},\boldsymbol{G}) \cup attr(\boldsymbol{x},\boldsymbol{Q})$.

f4. If $\boldsymbol{P} = \exists \boldsymbol{x}(R)\boldsymbol{G}$ then $\boldsymbol{x}$ must occur free in $\boldsymbol{G}$ for $\boldsymbol{P}$ to be legal. Furthermore, if $scheme(\boldsymbol{x},\boldsymbol{G})$ is defined, then equality $scheme(\boldsymbol{x},\boldsymbol{G}) = R$ must hold if including $attr(\boldsymbol{x},\boldsymbol{G}) \subseteq R$ is held. $scheme(\boldsymbol{x},\boldsymbol{P})$ and $attr(\boldsymbol{x},\boldsymbol{P})$ are not defined, since $\boldsymbol{x}$ does not occur free in $\boldsymbol{P}$. Any occurrence of a variable $\boldsymbol{y} \neq \boldsymbol{x}$ is free or bound in $\boldsymbol{P}$ as it was in $\boldsymbol{G}$. If $\boldsymbol{y}$ occur free in $\boldsymbol{P}$, then $scheme(\boldsymbol{y},\boldsymbol{P}) \simeq scheme(\boldsymbol{y},G)$ and $attr(\boldsymbol{y},\boldsymbol{P}) = attr(\boldsymbol{y},\boldsymbol{G})$.

f5. If $\boldsymbol{P} = \forall \boldsymbol{x}(R)\boldsymbol{G}$, then all restrictions and definitions are the same as in f4.

f6. If $\boldsymbol{P} = (\boldsymbol{G})$, then $\boldsymbol{P}$ is legal, and freedom, boundness, *scheme* and *attr* are the same as for $\boldsymbol{G}$.

After introducing the set of legal formulas we can give a finally determination of expression of tuple calculus.

Definition 11. Generalized tuple relation calculus expression has the form $\{\boldsymbol{x}(R) \mid \boldsymbol{P}(\boldsymbol{x})\}$, where

1) formula $\boldsymbol{P}$ is legal;
2) $\boldsymbol{x}$ is the only tuple variable that occurs free in $\boldsymbol{P}$;
3) if $scheme(\boldsymbol{x},\boldsymbol{P})$ is defined, then $scheme(\boldsymbol{x},\boldsymbol{P}) = R$, otherwise, $attr(\boldsymbol{x},\boldsymbol{P}) \subseteq R$, where $scheme(\boldsymbol{x},\boldsymbol{P})$ is the scheme of tuple $\boldsymbol{x}$ in the formula $\boldsymbol{P}$, $attr(\boldsymbol{x},\boldsymbol{P})$ is the set of attributes with which tuple variable $\boldsymbol{x}$ occurs in the formula $\boldsymbol{P}$.

3.1 Generalized domain relational calculus

Domain relational calculus is quite similar to tuple relational calculus. But there are essential differences. There are no tuple variables in the domain relational calculus, but there are variables to represent components of tuples, instead. Domain relational calculus are also supported by the membership condition: $\langle t(\langle A_1, d_1\rangle, \langle A_2, d_2\rangle, ...), R\rangle$, where R is scheme, A_i is attribute of table t and d_i is variable of domain or literal (object constant). This condition is true iff in the set t there is a tuple having specified values over universal domain $\boldsymbol{D}$ for specified attributes.

The set of legal domain relational calculus formulas will be defined relative to:

- a set of attributes $\boldsymbol{A}$ and universal domain $\boldsymbol{D}$;
- a set of object variables (tuple variables) $x_1, x_2, ...$;
- a set of object constants $d_1, d_2, ...$;
- a set of function symbols $f_1^{n_1}, f_2^{n_2}, ..., n_i \geq 1$;
- a set of predicate symbols $p_1^{m_1}, p_2^{m_2}, ..., m_i \geq 1$;
- a set of constant tables symbols along with their schemes; constant tables are denoted as $\langle t, R\rangle$;
- a set of variable tables symbols along with their schemes; variable tables are denoted as $\langle X, R\rangle$.

The universal domain $\boldsymbol{D}$ is the domain of interpretation of object constants and object variables. We use $\boldsymbol{x}$ as syntactic variable, the domain of change of which is the set of variables; $\boldsymbol{f}$ ($\boldsymbol{p}$) as syntactic variable, the domain of change of which is the set of function (predicate) symbols; $\boldsymbol{d}$ as syntactic variable, the domain of change of which is the set of constants; $\mathcal{A}$ as syntactic variable, the domain of change of which is the set of attributes.

Definition 12. The following expressions are terms (induction on length of terms):

a. any object constant $\boldsymbol{d}^A$ is a term, where A is an attribute associated with this constant;
b. any object variable $\boldsymbol{x}^A$ is a term, where A is an attribute associated with this variable;
c. if $\boldsymbol{u}_1...,\boldsymbol{u}_n$ are terms, $\boldsymbol{f}$ is a function symbol of arity n then $\boldsymbol{f}(\boldsymbol{u}_1...,\boldsymbol{u}_n)$ $\boldsymbol{f}(\boldsymbol{u}_1...,\boldsymbol{u}_n)$ is a term;
d. an expression is a term if and only if it can be shown to be a term on the basis of conditions a), b) and c).

We use $\boldsymbol{u}$ as syntactic variable, the domain of change of which is the set of terms.

Definition 13. There are three kinds of atomic formulas (atoms):

a1. For any constant table $\langle t, R\rangle$, where $R=\{A_1,...,A_n\}\subseteq \boldsymbol{A}$, $t_R(\boldsymbol{a}_1^{A_1},...,\boldsymbol{a}_n^{A_n})$ is an atom, $\boldsymbol{a}_i^{A_i}$ is a constant or variable which associated with attribute A_i.

a2. For any variable table $\langle X, R\rangle$, where $R=\{A_1,...,A_n\}\subseteq \boldsymbol{A}$, $X_R(\boldsymbol{a}_1^{A_1},...,\boldsymbol{a}_n^{A_n})$ is an atom, $\boldsymbol{a}_i^{A_i}$ is a constant or variable which associated with attribute A_i.

a3. For any terms $\boldsymbol{u}_1...,\boldsymbol{u}_m$, and for any predicate $\boldsymbol{p}$ of arity m on the universal domain $\boldsymbol{D}$, $\boldsymbol{p}(\boldsymbol{u}_1...,\boldsymbol{u}_m)$ is an atom.

We use the connectives $\neg$, $\wedge$, $\vee$, quantifiers $\exists,\forall$ and brackets () to build formulas from atoms. We use $\boldsymbol{P}$, $\boldsymbol{Q}$ and $\boldsymbol{G}$ as syntactic variables, the domain of change of which is the set of formulas.

Definition 14. The following expressions are formulas (induction on length of formulas):

f1. Every atom is a formula.
f2. If $\boldsymbol{P}$ is a formula, then $\neg\,\boldsymbol{P}$ is a formula.
f3. If $\boldsymbol{P}$ i $\boldsymbol{Q}$ are formulas, then $(\boldsymbol{P}\wedge\boldsymbol{Q})$, $(\boldsymbol{P}\vee\boldsymbol{Q})$ are formulas.
f4. If $\boldsymbol{x}^A$ is a tuple variable, $\boldsymbol{P}$ is a formula, $A\in\boldsymbol{A}$ is a scheme, then $\exists\boldsymbol{x}^A(A)\,\boldsymbol{P}$ is a formula.
f5. If $\boldsymbol{x}^A$ is a tuple variable, $\boldsymbol{P}$ is a formula, $A\in\boldsymbol{A}$ is a scheme, then $\forall\boldsymbol{x}^A(A)\,\boldsymbol{P}$ is a formula.
f6. If $\boldsymbol{P}$ is a formula, then $(\boldsymbol{P})$ is a formula.
f7. There are no other formulas.

Definition 15. Generalized domain relational calculus expression has the form $\{\boldsymbol{x}_1^{A_1},...,\boldsymbol{x}_n^{A_n}\mid \boldsymbol{P}(\boldsymbol{x}_1^{A_1},...,\boldsymbol{x}_n^{A_n})\}$, where

1) formula $\boldsymbol{P}$ is a legal domain calculus formula with exactly the free variables $\boldsymbol{x}_1^{A_1},...,\boldsymbol{x}_n^{A_n}$; every variable $\boldsymbol{x}_i$ is associated with the attribute $\boldsymbol{x}_i^{A_i}$, $i=1,\ldots,n$, where $A_i\neq A_j$ for $i\neq j$;
2) $R=\{A_1,...,A_n\}$ is a scheme of expression.

3 Equivalence of Table Algebra of Infinite Tables and Corresponding Relational Calculi

Theorem 1. If F is the expression of table algebra of infinite tables, then it is possible effectively to build equivalent to it expression E of generalized tuple relational calculus.

Proof. For proof of the theorem we consider expressions of table algebra which contain the operations of union, intersection, difference, selection, projection, join and rename only because the operations of division and active complement it is possible to express through these operations (see Lemma 1).

The proof proceeds by induction on the number of operators in F.

Basis (No operators). There are two cases. Firstly, $F=\langle t,R\rangle$ is the constant table, where t is a infinite set of the scheme R, then $E=\{\boldsymbol{x}(R)\mid t_R(\boldsymbol{x})\}$. Secondly, $F=\langle X,R\rangle$ is the variable table, then $E=\{\boldsymbol{x}(R)\mid X_R(\boldsymbol{x})\}$.

Induction. Assume the theorem holds for any table algebra expression with fewer than k operators. Let F have k operators.

Case 1 (union). $F=F_1\bigcup_R F_2$. F_1 and F_2 each have less than k operators, and by the inductive hypothesis we can find tuple relational calculus expressions $\{\boldsymbol{x}(R)\mid P(\boldsymbol{x})\}$ and $\{\boldsymbol{x}(R)\mid Q(\boldsymbol{x})\}$ equivalent to F_1 and F_2 respectively. Then E is $\{\boldsymbol{x}(R)\mid P(\boldsymbol{x})\vee Q(\boldsymbol{x})\}$.

Case 2 (difference). $F=F_1\setminus_R F_2$. Then tuple relational calculus expressions $\{\boldsymbol{x}(R)\mid P(\boldsymbol{x})\}$ and $\{\boldsymbol{x}(R)\mid Q(\boldsymbol{x})\}$ equivalent to F_1 and F_2 respectively exist as in Case 1. Then E is $\{\boldsymbol{x}(R)\mid P(\boldsymbol{x})\wedge\neg Q(\boldsymbol{x})\}$.

Case 3 (selection). $F=\sigma_{\tilde{p},R}(F_1)$. Let $\{\boldsymbol{x}(R)\mid P(\boldsymbol{x})\}$ be tuple relational calculus expression equivalent to F_1. Then E is $\{\boldsymbol{x}(R)\mid P(\boldsymbol{x})\wedge\boldsymbol{p}(\boldsymbol{x}(A_1),...,\boldsymbol{x}(A_m))\}$, where $R=\{A_1,...,A_m\}$ is scheme of table that is the value of expression F_1. Assume that predicate-parameter of select is defined as

$\tilde{p}(s) = true \Leftrightarrow \boldsymbol{p}(s(A_1),...,s(A_m)) = true$, $s \in S(R)$, where $\boldsymbol{p}$ is a signature predicate symbol of arity m.

Case 4 (projection). $F = \pi_{X,R}(F_1)$. Let $\{\boldsymbol{x}(R) \mid P(\boldsymbol{x})\}$ be tuple relational calculus expression equivalent to F_1. Then E is $\{\boldsymbol{y}(X \cap R) \mid \exists \boldsymbol{x}(R)(P(\boldsymbol{x}) \wedge \underset{A \in X \cap R}{\wedge} \boldsymbol{y}(A) = \boldsymbol{x}(A))\}$.

Special case: if $X \cap R = \varnothing$ then $E = \{\boldsymbol{y}(\varnothing) \mid \exists \boldsymbol{x}(R) P(\boldsymbol{x})\}$.

Case 5 (join). $F = F_1 \underset{R_1,R_2}{\otimes} F_2$. Let $\{\boldsymbol{x}(R_1) \mid P(\boldsymbol{x})\}$ and $\{\boldsymbol{y}(R_2) \mid Q(\boldsymbol{y})\}$ be tuple relational calculus expressions equivalent to F_1 and F_2 respectively. Then E is $\{\boldsymbol{z}(R_1 \bigcup R_2) \mid \exists \boldsymbol{x}(R_1) \exists \boldsymbol{y}(R_2)(P(\boldsymbol{x}) \wedge \wedge Q(\boldsymbol{y}) \wedge \underset{A \in R_1}{\wedge} \boldsymbol{z}(A) = \boldsymbol{x}(A) \wedge \underset{A \in R_2}{\wedge} \boldsymbol{z}(A) = \boldsymbol{y}(A))\}$.

Case 7 (rename). $F = Rt_{\xi,R}(F_1)$, where $\xi : \boldsymbol{A} \xrightarrow{\sim} \boldsymbol{A}$ is an injective partial function that renames attributes. Tuple relational calculus expression $\{\boldsymbol{x}(R) \mid P(\boldsymbol{x})\}$ equivalent to F_1 exists. Then $E = \{\boldsymbol{y}(R_2) \mid \exists \boldsymbol{x}(R)(P(\boldsymbol{x}) \wedge \underset{C \in R \setminus dom\xi}{\wedge} \boldsymbol{y}(C) = = \boldsymbol{x}(C) \wedge \underset{A \in R \cap dom\xi}{\wedge} \boldsymbol{x}(A) = \boldsymbol{y}(\xi(A)))\}$, where $R_2 = R \setminus dom\xi \bigcup \xi[R]$.

Theorem 1 proves that generalized tuple relational calculus is as expressive as table algebra of infinite tables (in terms of [6]).

Now consider how for generalized tuple relational calculus expression to build generalized domain relational calculus expression. Consider the mapping of the form $E \overset{\varphi}{\mapsto} H$ such that to every expression of generalized tuple relational calculus E puts in correspondence equivalent expression of generalized domain relational calculus H. Let $E = \{\boldsymbol{y}(R) \mid \boldsymbol{P}(\boldsymbol{y})\}$, where tuple $\boldsymbol{y}$ is the only tuple variable that occurs free in $\boldsymbol{P}$ and $R = \{A_1,...,A_n\}$.

Make the following replacements and obtain the generalized domain calculus expression $H = \{\boldsymbol{y}_1,...,\boldsymbol{y}_n \mid \boldsymbol{P}(\boldsymbol{y}_1,...,\boldsymbol{y}_n)\}$. First apply the mapping φ to the terms of generalized tuple relational calculus:

1) every object constant $\boldsymbol{d}$ of generalized tuple relational calculus becomes object constant $\boldsymbol{d}^A$ of generalized domain relational calculus, where $A \in R$ is the attribute associated with this constant;
2) every term $\boldsymbol{x}(A_i)$ of generalized tuple relational calculus becomes variable $\boldsymbol{x}_i^{A_i}$ of generalized domain relational calculus;
3) every term $\boldsymbol{f}(\boldsymbol{v}_1,...,\boldsymbol{v}_n)$ of generalized tuple relational calculus, where $\boldsymbol{v}_i$ is the terms of generalized tuple relational calculus, $i = 1,...,n$, becomes term $\boldsymbol{f}(\boldsymbol{u}_1,...,\boldsymbol{u}_n)$ of generalized domain relational calculus, where $\boldsymbol{u}_j$ is the terms of generalized domain relational calculus, $j = 1,...,n$ which derived from the previous replacements;

Apply the mapping φ to the atoms of generalized tuple relational calculus:

a1) any atom $t_{R'}(\boldsymbol{z})$ of generalized tuple relational calculus becomes $t_{R'}(\boldsymbol{z}_1^{A_1},\ldots,\boldsymbol{z}_m^{A_m})$, where $\boldsymbol{z}_1^{A_1},\ldots,\boldsymbol{z}_m^{A_m}$ are the variable of generalized domain relational calculus, $R' = \{A_1,...,A_m\}$ is the scheme of constant table $\langle t, R' \rangle$;

a2) any atom $X_{R'}(\boldsymbol{z})$ of generalized tuple relational calculus becomes $X_{R'}(\boldsymbol{z}_1^{A_1},\ldots,\boldsymbol{z}_m^{A_m})$, where $\boldsymbol{z}_1^{A_1},\ldots,\boldsymbol{z}_m^{A_m}$ are the variable of generalized domain relational calculus, $R' = \{A_1,...,A_m\}$ is the scheme of variable table $\langle X, R' \rangle$;

a3) any atom $\boldsymbol{p}(\boldsymbol{v}_1,...,\boldsymbol{v}_m)$ of generalized tuple relational calculus, where $\boldsymbol{v}_i$ are the terms, $i = 1,...,m$, becomes $\boldsymbol{p}(\boldsymbol{u}_1,...,\boldsymbol{u}_m)$, where $\boldsymbol{u}_j$ are the terms of generalized domain relational calculus which derived from the previous replacements, $j = 1,...,m$.

Any formula $\boldsymbol{P}$ of generalized tuple relational calculus replaced by $\boldsymbol{P}'$, where for each atoms made substitute a1-a3 and each free occurrence of $\boldsymbol{z}$ replaced by $\boldsymbol{z}_1^{A_1},...,\boldsymbol{z}_n^{A_n}$, $R = \{A_1,...,A_n\}$ is the scheme of $\boldsymbol{z}$. A quantified subformula $\exists \boldsymbol{z}(R_2)\boldsymbol{G}$ of generalized tuple relational calculus, $R_2 = \{A_1,...,A_m\}$, becomes $\exists \boldsymbol{z}_1^{A_1}(A_1)...\exists \boldsymbol{z}_m^{A_m}(A_m)\boldsymbol{G}'$, while $\forall \boldsymbol{z}(R_2)\boldsymbol{G}$ becomes $\forall \boldsymbol{z}_1^{A_1}(A_1)...\forall \boldsymbol{z}_m^{A_m}(A_m)\boldsymbol{G}'$.

After completing these changes, we will get expression of generalized domain relational calculus $H = \{\boldsymbol{y}_1^{A_1},...,\boldsymbol{y}_n^{A_n} \mid \boldsymbol{P}(\boldsymbol{y}_1^{A_1},...,\boldsymbol{y}_n^{A_n})\}$. It should also be clear that the values that may be assumed by every variables $\boldsymbol{z}_i$ of generalized domain calculus are exactly those that could be assumed by $\boldsymbol{z}(A_i)$ in the original expression. Thus, the expressions E and H are equivalent. Consequently, the following theorem takes place.

Theorem 2. If E is an expression of generalized tuple relational calculus, then it is possible

effectively to build equivalent to it expression H of generalized domain relational calculus. □

So, theorem 2 proves that generalized domain relational calculus is as expressive as generalized tuple relational calculus (in terms of [6]).

Let's prove that table algebra of infinite tables is as expressive as generalized domain relational calculus.

Theorem 3. If H is an expression of generalized domain relational calculus, then it is possible effectively to build equivalent to it expression F of table algebra of infinite tables.

Proof. Let $H = \{\boldsymbol{x}_1^{A_1},...,\boldsymbol{x}_n^{A_n} \mid \boldsymbol{P}(\boldsymbol{x}_1^{A_1},...,\boldsymbol{x}_n^{A_n})\}$ be expression of generalized domain relational calculus, where $R = \{A_1,...,A_n\}$ is a scheme of the table, that is the value of expression H. For each subexpression $\boldsymbol{G}$ of $\boldsymbol{P}$ we will find an algebraic expression F_G.

We will prove by induction on the number of operators in a subformula $\boldsymbol{G}$ of $\boldsymbol{P}$ that if $\boldsymbol{G}$ has free domain variables $\boldsymbol{y}_1^{A_1},...,\boldsymbol{y}_m^{A_m}$ then $\{\boldsymbol{y}_1^{A_1},...,\boldsymbol{y}_m^{A_m} \mid \boldsymbol{G}(\boldsymbol{y}_1^{A_1},...,\boldsymbol{y}_m^{A_m})\}$ has an equivalent expression in table algebra of infinite tables F_G.

Then, as a special case, when $\boldsymbol{G}$ is $\boldsymbol{P}$ itself, we have an algebraic expression for $\{\boldsymbol{x}_1^{A_1},...,\boldsymbol{x}_n^{A_n} \mid \boldsymbol{P}(\boldsymbol{x}_1^{A_1},...,\boldsymbol{x}_n^{A_n})\}$. It is assumed that $\boldsymbol{y}_1^{A_1},...,\boldsymbol{y}_m^{A_m}$ are the only tuple variables that occurs free in $\boldsymbol{G}$ and the table of scheme $R_G = \{A_1,...,A_m\}$ is a value of expression $\{\boldsymbol{y}_1^{A_1},...,\boldsymbol{y}_m^{A_m} \mid \boldsymbol{G}(\boldsymbol{y}_1^{A_1},...,\boldsymbol{y}_m^{A_m})\}$.

Replace domain variables in $\boldsymbol{P}$ so that no variable is bound in two places or occurs both free and bound in $\boldsymbol{P}$. Note that every variable is associated with an attribute, either by a quantifier $\forall \boldsymbol{x}(A)$ or $\exists \boldsymbol{x}(A)$ if a variable is bound in $\boldsymbol{P}$ or by appearing to the left of the bar in the expression H, if a variable is free in $\boldsymbol{P}$.

For any attribute A we will map algebraic expression. A single-attribute table $\langle t, \{A\}\rangle$ is a value of this algebraic expression. This table contains all tuples of kind $s = \{\langle A, d\rangle\}$, $d \in \boldsymbol{D}$. We will designate this algebraic expression through $[\boldsymbol{D}]_A$. Consider all possible cases.

Basis (no operators). Subformula $\boldsymbol{G}$ is an atom of the form $\boldsymbol{p}(\boldsymbol{u}_1,...,\boldsymbol{u}_m)$ or $t_R(\boldsymbol{a}_1^{A_1},...,\boldsymbol{a}_m^{A_m})$ or $X_R(\boldsymbol{z}_1^{A_1},\ldots,\boldsymbol{z}_m^{A_m})$.

1. Let $\boldsymbol{G}$ be $\boldsymbol{p}(\boldsymbol{u}_1,...,\boldsymbol{u}_m)$, where $\boldsymbol{u}_i$ are the terms of generalized domain relational calculus, $\boldsymbol{y}_1^{A_1},...,\boldsymbol{y}_k^{A_k}$ are the all variables of this terms and $A_1,\ldots,A_k$ are the attributes associated with these variables. Then F_G is $\sigma_{\tilde{p},\{A_1,\ldots,A_k\}}([D]_{A_1} \underset{R_1,R_2}{\otimes} ... \underset{R_1\cup...\cup R_{k-1},R_k}{\otimes} [D]_{A_k})$, where $R_i = \{A_i\}$, $i = 1,\ldots,k$. Assume that predicate-parameter of select is defined as $\tilde{p}(s) = true \Leftrightarrow \boldsymbol{p}(s(A_{k_1}),...,s(A_{k_m})) = true$, $s \in S(R)$, where $\boldsymbol{p}$ is signature predicate symbol of arity m.
2. Let $\boldsymbol{G}$ be $t(\boldsymbol{a}_1^{A_1},...,\boldsymbol{a}_m^{A_m})$, where $\boldsymbol{a}_i^{A_i}$ is either a constant or a variable over the universal domain $\boldsymbol{D}$. Let $R = \{A_1,...,A_m\}$ be a scheme of table $\langle t, R\rangle$. The algebraic expression F_G is $\pi_{Y,R}(\sigma_{\tilde{p},R}(\langle t, R\rangle))$, where $\tilde{p}$ is predicate-parameter of select which is a conjunction of comparisons $A_i = \boldsymbol{a}_i^{A_i}$ for each $\boldsymbol{a}_i^{A_i}$ that is a constant; $Y = \{A_j \mid \boldsymbol{a}_j^{A_j}$ is a variable$\}$.
3. Let $\boldsymbol{G}$ be $X(\boldsymbol{a}_1^{A_1},...,\boldsymbol{a}_m^{A_m})$, where $\boldsymbol{a}_i^{A_i}$ is either a constant or a variable over the universal domain $\boldsymbol{D}$. Let $R = \{A_1,...,A_m\}$ be a scheme of table $\langle X, R\rangle$. The algebraic expression F_G is $\pi_{Y,R}(\sigma_{\tilde{p},R}(\langle X, R\rangle))$, where $\tilde{p}$ is predicate-parameter of select which is a conjunction of comparisons $A_i = \boldsymbol{a}_i^{A_i}$ for each $\boldsymbol{a}_i^{A_i}$ that is a constant; $Y = \{A_j \mid \boldsymbol{a}_j^{A_j}$ is a variable$\}$.

Induction. Assume $\boldsymbol{G}$ has at least one operator and that the inductive hypothesis is true for all subformulas of $\boldsymbol{P}$ having fewer operators than $\boldsymbol{G}$.

Case 1. $\boldsymbol{G} = \neg \boldsymbol{Q}$. Let F_Q be the algebraic expression for $\boldsymbol{Q}$ and the table of scheme R_Q is a value of expression F_Q. Then $F_G = \sim_{R_Q} F_Q$.

Case 2. $\boldsymbol{G} = \boldsymbol{Q} \vee \boldsymbol{Q}'$. Let $\boldsymbol{Q}$ has free variables $z_1^{B_1},...,z_k^{B_k}, v_1^{C_1},...,v_p^{C_p}$ and let $\boldsymbol{Q}'$ has free variables $z_1^{B_1},...,z_k^{B_k}, w_1^{K_1},...,w_q^{K_q}$, where $v_1^{C_1},...,v_p^{C_p}$ and $w_1^{K_1},...,w_q^{K_q}$ are distinct. F_Q and $F_{Q'}$ are the algebraic expressions for $\boldsymbol{Q}$ and $\boldsymbol{Q}'$ respectively. Let C_i, $i = 1,...,p$ be the attributes associated with variables $v_1^{C_1},...,v_p^{C_p}$ and let K_j, $j = 1,...,q$ be the attributes associated with variables $w_1^{K_1},...,w_q^{K_q}$. Let $F_1 = F_Q \underset{R_Q,\{K_1\}}{\otimes} [D]_{K_1} \underset{R_Q\cup\{K_1\},\{K_2\}}{\otimes} ... \underset{R_Q\cup\{K_1,...,K_{q-1}\},\{K_q\}}{\otimes} [D]_{K_q}$ and let

$F_2 = F_{Q'} \underset{R_{Q'},\{C_1\}}{\otimes} [D]_{C_1} \underset{R_{Q'}\cup\{C_1\},\{C_2\}}{\otimes} ... \underset{R_{Q'}\cup\{C_1,...,C_{p-1}\},\{C_p\}}{\otimes} [D]_{C_p}$,

where R_Q and $R_{Q'}$ are the schemes of the tables that are the values of those algebraic expressions respectively.

Let R_{F_1} and R_{F_2} be the schemes of the tables that are the values of algebraic expressions F_1 and F_2 respectively. Note that $R_{F_1} = R_{F_2}$. Then $F_G = F_1 \bigcup_{R_{F_1}} F_2$.

Case 3. $\boldsymbol{G} = \boldsymbol{Q} \wedge \boldsymbol{Q}'$. This generalized domain calculus expression can be replaced by $\boldsymbol{G} = \neg(\neg\boldsymbol{Q} \vee \neg\boldsymbol{Q}')$ (De Morgan's law).

Case 4. $\boldsymbol{G} = \exists \boldsymbol{x}^A(A)\boldsymbol{Q}$. Let F_Q be the algebraic expression for $\boldsymbol{Q}$. Then F_G is $\pi_{X\setminus\{A\},X}(F_Q)$, where X is the scheme of the table that is the value of algebraic expression F_Q.

Case 5. $\boldsymbol{G} = \forall \boldsymbol{x}^A(A)\boldsymbol{Q}$. This generalized domain calculus expression can be replaced by $\forall \boldsymbol{x}^A(A)\boldsymbol{Q} = \neg(\exists \boldsymbol{x}^A(A))(\neg\boldsymbol{Q})$. □

Theorem 3 proves that table algebra of infinite tables is as expressive as generalized domain relational calculus (in terms of [6]).

Taking into account the theorems 1, 2, 3 we are getting a basic result. This situation is shown in Fig. 1.

Theorem 4. Table algebra of infinite tables, generalized tuple relational calculus and generalized domain relational calculus are equivalent. □

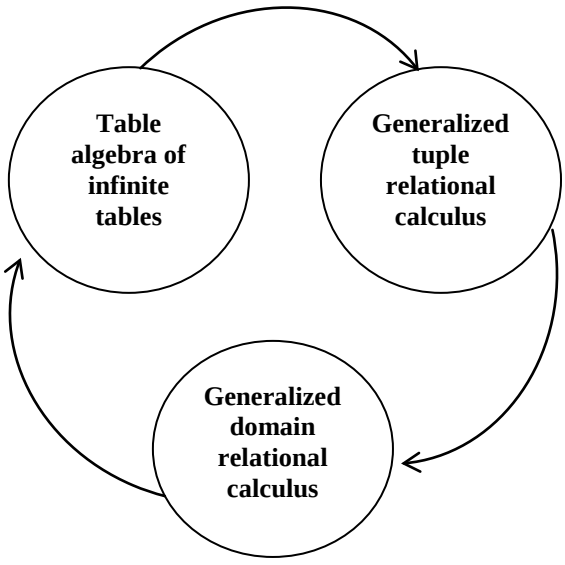

Fig.1 Equivalence of Table Algebra of Infinite Tables and Corresponding Relational Calculi

4 Conclusion

In article table algebras of infinite tables are considered. The classical relational al calculi are extended by functional and predicate signatures on the universal domain (while usually consider only binary predicates and functional signature is generally empty). It is proved the equivalence of table algebra of infinite tables and corresponding relational calculi. These results generalize the classical result about the equivalence of Codd's relational algebra and tuple (domain) relational calculus.

References:

[1] E.F. Codd, A Relational Model of Data for Large Shared Data Banks, *Comm. of ACM*, 13(6), 1970, pp. 377-387.

[2] E.F. Codd, Relational Completeness of Data Base Sublanguages, *Data Base Systems, Proceedings of 6th Courant Computer Science Symposium*, 1972, pp. 65-93.

[3] M. Lacroix, A. Pirotte, Domain-oriented Relational Languages, *Proceedings of 3rd Int. Conf. on Very Large Data Bases*, 1977, pp. 370-378.

[4] A. Klug, Equivalence of Relational Algebra and Relational Calculus Query Languages Having Aggregate Functions, *J. ACM*, 29(3), 1982, pp. 699-717.

[5] J.D. Ullman, *Principles of database systems*, Rockville, Maryland: Computer Science Press, 1982.

[6] D. Maier, *The theory of relational databases*, Rockville, Maryland: Computer Science Press, 1983.

[7] V.N. Redko, Yu.J. Borona, D.B. Buy, S.A. Poliakov, *Reliatsiini bazy danykh: tablychni alhebry ta SQL-podibni movy*, Kyiv, Vydavnychyi dim «Akademperiodyka», 2001. (in Ukrainian)

[8] D. Buy, I. Glushko, Generalized Table Algebra, Generalized Tuple Calculus and Theirs Equivalence, *Proceedings of CSE 2010 International Scientific Conference on Computer Science and Engineering*, 2010, pp. 231-238.

[9] E. Mendelson, *Introduction to mathematical logic*, Chapman & Hall, London, 1997.