Extension of Tuple Calculus to Multisets

Iryna Lysenko[1*], Olha Moroz[2]

[1]*Nizhyn Mykola Gogol State University, Grafska str., 2, Nizhyn, 16600, Ukraine*
[2]*International Research and Training Centre for Information Technologies and Systems of the NASU and MESU, Academician Hlushkov Ave., 40, Kyiv, 03680, Ukraine*

Abstract

This paper is a logical continuation of research devoted to the actual problem of developing the theoretical foundations of table (relational) databases. The issue of using multisets in table databases is important and relevant. Many database-oriented languages require a relational model with multiset semantics. There are many applied problems, the feature of which is multiplicity and repeatability of data. For example, these are sociological polls of different population groups, calculations on DNA, and others. In this context, the question of constructing a tuple calculus for a multiset table algebra is considered, in which the concept of a table is refined using the concept of a multiset. In the article, the formalization of tuple calculus for multiset table algebra is carried out. The alphabet, and the syntax of terms, atoms, and formulas are defined. A set of legal formulas is introduced through the concept of the free and bound variable. The concept of a scheme and set of attributes with which a tuple variable occurs in a formula are also introduced. The definition of tuple calculus expression for multiset table algebra is given, according to which it is a multiset of tuples that satisfy the condition defined by the legal formula. The article provides rules for determining the number of tuple duplicates in the resulting multiset. Another important result consists in proving that constructed tuple calculus is as expressive as multiset table algebra. This research opens up new possibilities for database theory development and may be useful for information technology and database professionals. It contributes to a deeper understanding of construction query principles, an important aspect of modern computer science and industry.

Keywords

Relation Databases, multiset, multiset table algebra, tuple calculus

1. Introduction

Relational calculus underlies most relational query languages, specifying only the expected result, while relational algebra involves constructing a relational expression and performing operations. There are two main approaches to relational calculus:

- Tuple calculus (E. Codd) which operates on table tuples [1];
- Domain calculus (M. Lacroix, A. Pirotte) which focuses on table domains [2].

The clarification of the concept of relation in terms of naminal sets was carried out by V.N. Redko, Yu.Y. Bron, D.B. Buy, and S.A. Polyakov [3]. The monograph [4] introduces the consideration of table algebra for infinite (finite) tables, which significantly generalizes and extends classical Codd's relational algebra. The generalization is that a relation is understood as an arbitrary set of single-scheme tuples, in particular, an infinite one, while each table is assigned a certain scheme. Tuple (domain) calculi have been also constructed for these algebras. Tuple calculus and domain calculus are supplemented with functional and predicate signatures on the universal domain. Two main results are presented. The first result demonstrates the equivalence between the

14th International Scientific and Practical Conference from Programming UkrPROG'2024, May 14-15, 2024, Kyiv, Ukraine
* Corresponding author.
iryna.glushko@ndu.edu.ua (I. Lysenko); mog_91@ukr.net (O. Moroz)
0000-0003-2549-5356 (I. Lysenko); 0000-0002-0356-8780 (O. Moroz)
© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR-WS.org/Vol-3806/S_16_Lysenko_Moroz.pdf

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

table algebra for infinite tables and the corresponding generalized relational calculi. The second result focuses on the equivalence between the subalgebra for finite tables of this algebra and the corresponding restricted generalized relational calculi. For the second result, the consideration is limited to only "safe" expressions. The methodological basis of these formalisms is the compositional approach to programming.

Additionally, in the work [4] multiset table algebra is constructed, which extends database capabilities through multisets. The signature of multiset table algebra is supplemented with new operations: inner and outer join operations, semijoin operations, and aggregate operations.

In [5, 6] the theorem-plural and logical-algebraic methods found that table algebra of infinite tables does not form subalgebra of multiset table algebra since it is not closed with respect to the union, projection and active complement. Thus, multiset table algebra is not a wider formalism then table algebra of infinite tables

Many database-oriented languages, such as SQL, require a relational model with multiset semantics. However, the traditional tuple (domain) calculi do not support queries that produce tables with duplicates. Therefore, there is a need to develop a calculus for multiset table algebra.

The research aims to construct a tuple calculus for multiset table algebra and to show that it is no less expressive than multiset table algebra.

2. Basics of Multisets

Many languages oriented towards working with databases require a relational data model with multiset semantics because, firstly, relations that allow duplicates are useful in many applied fields where duplicate objects may exist; secondly, in the relational data model, removing duplicates after performing projection and union operations implies merging identical elements or performing other labor-intensive actions. Various researchers have focused on the utilization of multisets in databases, including Paul Grefen and Rolf A. de By [7, 8], G. Lamperti, M. Melchiori, and M. Zanella [9], G. Garcia-Molina, J. Ullman, and J. Widom [10, 11], A. Silbeschatz, H. Korth, and S. Sudarshan [12], D.B. Buy, S.A. Polyakov, Yu.Y. Brona, and V.N. Redko [3]. However, this issue still requires clarification and expansion.

Let's start by considering the key terms of multisets based on sources [3, 13].

Let U be an arbitrary set. A multiset α with basis U is a function

$$\alpha: U \to N,$$

where $N = \{1, 2, \dots\}$ is the set of natural numbers.
Let $\boldsymbol{D}$ be the universe of elements of multiset bases. A characteristic function of multiset α is a function $\chi_\alpha: \boldsymbol{D} \to Z_+$, the values of which are specified by the following piecewise schema:

$$\chi_\alpha(d) = \begin{cases} \alpha(d) \text{ if } d \in dom\ \alpha \\ 0, \text{else} \end{cases}$$

for all $d \in D$. Here $dom\ \alpha$ is the range of definition of multiset as a function ($dom\ \alpha = U_\alpha$ – basis of multiset α).

The empty multiset $\emptyset_m$ is defined as a multiset which basis is an empty set.

The 1-multisets are multisets whose range of values is the empty set or a single-element set {1}. These multisets are the analogues of ordinary sets.

The operations over multisets are defined in terms of characteristic functions in monograph [3]. Authors define operations of multiset $\bigcup_1$, intersection $\bigcap_1$, difference $\setminus_1$, which build 1-multisets, and operations of multiset union $\bigcup_{All}^R$, intersection $\bigcap_{All}^R$, difference $\setminus_{All}^R$, which build multisets of

general view. The Cartesian product of multiset $\otimes$, the operation Dist α, which build 1-multiset, and analog of a full image for multisets are defined too.

3. Multiset Table Algebra

Let's examine the main concepts and statements of multiset table algebra, based on the monograph [4]. In this case, under relation, understand a multiset, in particular, infinite.

Let's consider two sets: $\boldsymbol{A}$ is the set of attributes, and $\boldsymbol{D}$ is the universal domain.

A scheme will be defined as any arbitrary finite set of attributes $R \subseteq \boldsymbol{A}$. 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. We will use the following notations: $S(R)$ is the set of all tuples of the schema R, and S is the set of all tuples.

A table of the scheme R ($R \subseteq \boldsymbol{A}$) is a pair

$$\langle \psi,\ R \rangle,$$

where the first component ψ is an arbitrary multiset, the basis of which $\Theta(\psi)$ is the set of tuples of the same scheme and the second component R is a scheme of the table.

The set of all table on scheme R is designated as $\Psi(R)$ and the set of all table is designated as $\Psi = \bigcup_R \Psi(\ )$.

The notation $Occ(s, \psi)$ represents the number of duplicate tuple s in the multiset ψ. A multiset ψ can also be written as $\{s_1^{n_1}, \dots, s_k^{n_k}\}$, where $n_i = Occ(s_i, \psi)$, $i = 1, \dots, k$, and $\Theta(\psi) = \{s_1, \dots, s_k\}$ is a of the multiset ψ.

Under multiset table algebra is understood an algebra

$$\langle \Psi,\ \Omega_{P,\Xi} \rangle,$$

where Ψ is the set of all tables, $\Omega_{P,\Xi} = \{\bigcup_{All}^{R}, \bigcap_{All}^{R}, \setminus_{All}^{R}, \sigma_{p,R}, \pi_{X,R}, \bigotimes_{R_1,R_2}, Rt_{\xi,R}, \sim_R\}_{X,R,R_1,R_2 \subseteq \boldsymbol{A}}^{p \in P, \xi \in \Xi}$ is the signature, P, Ξ are the sets of parameters. The signature $\Omega_{P,\Xi}$ contains operations of multiset (union $\bigcup_{All}^{R}$, intersection $\bigcap_{All}^{R}$, difference $\setminus_{All}^{R}$) and special operations (selection $\sigma_{p,R}$, projection $\pi_{X,R}$, *join* $\bigotimes_{R_1,R_2}$, *renaming* $Rt_{\xi,R}$, *and active complement* $\sim_R$).

Multiset table algebra is also filled up with additional operations such as inner join (Cartesian join, natural join, join using attributes and join on predicate), outer join (outer left join, outer right join, outer full join and union join), semijoin and aggregate operations (Sum, Avg, Max, Min, Count). The special element NULL is inserted in the universal domain for to define of outer join and outer set operations. The operations of signature $\Omega_{P,\Xi}$ and a formal mathematical semantics of additional operations are defined in [4].

The following statement holds.

Theorem 1. Any expression of the multiset table algebra can be replaced with an equivalent expression that uses only the operations of selection, join, projection, union, difference, and renaming.

Proof. To prove the first statement, we will show that the operations of intersection and active complement can be expressed through other operations of multiset table algebra. The operation of multiset intersection can be replaced by the difference [13]:

$$\langle \psi_1, R \rangle \bigcap_{All}^{R} \langle \psi_2, R \rangle = \langle \psi_1, R \rangle \setminus_{All}^{R} (\langle \psi_1, R \rangle \setminus_{All}^{R} \langle \psi_2, R \rangle).$$

The operation of active complement is expressed through the operations of projection, difference, and join (by definition):

$$\sim_R (\langle \psi, R \rangle) = C(\langle \psi, R \rangle) \setminus_{All}^{R} \langle \psi, R \rangle,$$

where $C(\langle\psi, R\rangle) = \pi_{\{A_1\},R}(\langle\psi, R\rangle) \underset{\{A_1\},\{A_2\}}{\otimes} \dots \underset{\{A_1,\dots,A_{n-1}\},\{A_n\}}{\otimes} \pi_{\{A_n\},R}(\langle\psi, R\rangle)$, and $R = \{A_1, \dots, A_n\}$ is a scheme of the table $\langle\psi, R\rangle$.

According to the proof, the operations of intersection and active complement are derived from other operations in the signature of multiset table algebra and can henceforth be excluded from consideration.

4. Tuple Calculus for Multiset Table Algebra

The basis of relational calculus is the calculus of first-order predicates. We will start the construction of tuple calculus for multiset table algebra with the definition of the alphabet.

The alphabet of tuple calculus for multiset table algebra consists of:

- a set of attributes $\boldsymbol{A}$ and universal domain $\boldsymbol{D}$;
- a set of object variables (tuple variables) $x_1, x_2, \dots$;
- a set of object constants $d_1, d_2, \dots$;
- a set of function symbols $f_1^{n_1}, f_2^{n_2}, \dots, n_i \geq 1$;
- a set of predicate symbols $p_1^{m_1}, p_2^{m_2}, \dots, m_i \geq 1$;
- a set of symbols of constant tables $\langle\alpha, R\rangle$;
- a set of symbols of variable tables $\langle X, R\rangle$;
- the signs of logical operations $\neg$, $\wedge, \vee$, and quantifiers $\forall, \exists$;
- punctuation marks – parentheses () and commas.

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

Terms and formulas are distinguished among the words written using alphabet symbols. Let us define these syntactic objects by induction on their length.

The following expressions are terms:

a) $\boldsymbol{d}$ is a term;
b) $\boldsymbol{x}(\boldsymbol{\mathcal{A}})$ is a term;
c) if $\boldsymbol{u}_1, \dots, \boldsymbol{u}_n$ are terms and $\boldsymbol{f}$ is a function symbol of arity n then $\boldsymbol{f}(\boldsymbol{u}_1, \dots, \boldsymbol{u}_n)$ is a term;
d) there are no terms other than those specified in points a)-c).

Further in the text, $\boldsymbol{u}$ is a syntactic variable, the domain of change of which is the set of terms.

Let's define the formulas, starting with atomic formulas (atoms), which come in three types:

a1. For any constant table $\langle\alpha, R\rangle$ and for any tuple variable $\boldsymbol{x}$, $\alpha_R(\boldsymbol{x})$ is an atom. $\alpha_R(\boldsymbol{x})$ stands for $\boldsymbol{x} \in \langle\alpha, R\rangle$.
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 $\boldsymbol{x} \in \langle\alpha, R\rangle$.
a3. For any terms $\boldsymbol{u}_1, \dots, \boldsymbol{u}_m$, and for any predicate $\boldsymbol{p}$ of arity m on the universal domain $\boldsymbol{D}$, $\boldsymbol{p}(\boldsymbol{u}_1, \dots, \boldsymbol{u}_m)$ is an atom.
f1. Let's construct formulas from atoms using the logical connectives $\neg$, $\wedge, \vee$, quantifiers $\forall, \exists$, and parentheses.
f2. Every atom is a formula.
f3. If $\boldsymbol{P}$ and $\boldsymbol{Q}$ are formulas, then $\neg\boldsymbol{P}$, $\boldsymbol{P}\wedge\boldsymbol{Q}, \boldsymbol{P}\vee\boldsymbol{Q}$ are formulas.

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

f5. If $\boldsymbol{P}$ is a formula, then $(\boldsymbol{P})$ is a formula.

f6. There are no other formulas besides those specified in points f1-f4.

We will use $\boldsymbol{P}$, $\boldsymbol{Q}$ and $\boldsymbol{G}$ as syntactic variables, the domain of change of which is the set of formulas.

Let's define the class of legal formulas, using the concepts of free and bound variables, the scheme $scheme(\boldsymbol{x}, \boldsymbol{P})$ for tuple variable $\boldsymbol{x}$, and the set of attributes with which tuple variable $\boldsymbol{x}$ occurs in a formula $\boldsymbol{P}$, $attr(\boldsymbol{x}, \boldsymbol{P})$. The expressions $cheme(\boldsymbol{x}, \boldsymbol{P})$ and $attr(\boldsymbol{x}, \boldsymbol{P})$ are defined if the tuple $\boldsymbol{x}$ has at least one free occurrence in the formula $\boldsymbol{P}$. Additionally, the including $attr(\boldsymbol{x}, \boldsymbol{P}) \subseteq scheme(\boldsymbol{x}, \boldsymbol{P})$ holds if these expressions are defined.

We will define expression $attr$ for terms first:

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

In other words, $attr(\boldsymbol{x}, \boldsymbol{u})$ is the set of attributes that the scheme of the tuple $\boldsymbol{x}$ must have.

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

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

a2. similarly 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 $\boldsymbol{P} = \boldsymbol{p}(\boldsymbol{u}_1, \dots, \boldsymbol{u}_m)$, where $\boldsymbol{u}_j$ are terms, and $\boldsymbol{x}_1, \dots, \boldsymbol{x}_k$ 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 $attr(\boldsymbol{x}_i, \boldsymbol{P}) = \bigcup_{j=1}^{m} attr(\boldsymbol{x}_i, \boldsymbol{u}_j)$, $i = 1, \dots, k$.

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

f1. 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 tuple $\boldsymbol{x}$ that occurs free in $\boldsymbol{G}$, $scheme(\boldsymbol{x}, \boldsymbol{P}) \simeq scheme(\boldsymbol{x}, \boldsymbol{G})$ and $attr(\boldsymbol{x}, \boldsymbol{P}) = attr(\boldsymbol{x}, \boldsymbol{G})$, where $\simeq$ is a generalized equality, meaning that both sides of the equality are either undefined or defined and have equal values [14].

f2. If $\boldsymbol{P} = \boldsymbol{G} \wedge$ 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. According to the definition $scheme(\boldsymbol{x}, \boldsymbol{P}) = scheme(\boldsymbol{x}, \boldsymbol{G})$.

   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. Formula $\boldsymbol{P}$ is legal if including $attr(\boldsymbol{x}, \boldsymbol{Q}) \subseteq scheme(\boldsymbol{x}, \boldsymbol{G})$ holds. According to the definition $scheme(\boldsymbol{x}, \boldsymbol{P}) = scheme(\boldsymbol{x}, \boldsymbol{G})$.

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

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

f3. 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 $scheme(\boldsymbol{y}, \boldsymbol{P}) \simeq scheme(\boldsymbol{y}, \boldsymbol{G})$ and $attr(\boldsymbol{y}, \boldsymbol{P}) = attr(\boldsymbol{y}, \boldsymbol{G})$.

f4. If $\boldsymbol{P} = \forall \boldsymbol{x}(R)\boldsymbol{G}$, then all restrictions and definitions are the same as in f3.

f5. If $\boldsymbol{P} = (\boldsymbol{G})$, then $\boldsymbol{P}$ is legal, and free and bound variables, $scheme$ and $attr$ are the same as for $\boldsymbol{G}$.

In other words, the equality $attr(\boldsymbol{x}, \boldsymbol{P}) = R$ means that for a specific interpretation of the formula $\boldsymbol{P}$, when the variable $\boldsymbol{x}$ takes on a value in the form of a tuple s of the scheme R', the inclusion $R \subseteq R'$ must hold.

Expressions of tuple calculus for multiset table algebra have the form

$$\{\boldsymbol{x}^n(R)|\boldsymbol{P}(\boldsymbol{x})\},$$

where:

1. The formula $\boldsymbol{P}$ is legal.
2. The variable $\boldsymbol{x}$ is the only variable that occurs free in the formula $\boldsymbol{P}$.
3. If $scheme(\boldsymbol{x},\boldsymbol{P})$ is defined, then $scheme(\boldsymbol{x},\boldsymbol{P}) = R$, otherwise $attr(\boldsymbol{x},\boldsymbol{P}) \subseteq R$.
4. n is the number of duplicates of the tuple $\boldsymbol{x}$.

It is worth emphasizing that the result of executing the query defined by the expression $\{\boldsymbol{x}^n(R)|\boldsymbol{P}(\boldsymbol{x})\}$ is a multiset of tuple described by the expression $\boldsymbol{P}(\boldsymbol{x})$.

Let $\boldsymbol{P}(\boldsymbol{x})$ be a legal formula, $R \subseteq \boldsymbol{A}$. Then s substituted for $\boldsymbol{x}$ in formula $\boldsymbol{P}$ is the formula $\boldsymbol{P}(s/\boldsymbol{x})$. The value of the formula $\boldsymbol{P}(s/\boldsymbol{x})$ is determined by modifying each atom from $\boldsymbol{P}$ according to the following rules:

a1. Let the tuple variable $\boldsymbol{x}$ in $\alpha_{R'}(\boldsymbol{x})$ be free in formula $\boldsymbol{P}$. By the definition of an legal formula, we have the inclusion $R' \subseteq R$. Atom $\alpha_{R'}(\boldsymbol{x})$ acquires the value of true at s substituted for $\boldsymbol{x}$, if $s|R' \in \theta(\alpha)$, otherwise, atom $\alpha_{R'}(\boldsymbol{x})$ acquires the value of false.

a2. Let the tuple variable $\boldsymbol{x}$ in $X_{R'}(\boldsymbol{x})$ be free in formula $\boldsymbol{P}$. By the definition of an legal formula, we have the inclusion $R' \subseteq R$. Atom $X_{R'}(\boldsymbol{x})$ acquires the value of true at s substituted for $\boldsymbol{x}$, if $s|R' \in \theta(X)$, otherwise, atom $X_{R'}(\boldsymbol{x})$ acquires the value of false.

a3. Let the tuple variable $\boldsymbol{x}$ in $\boldsymbol{P} = \boldsymbol{p}(\boldsymbol{u}_1, \dots, \boldsymbol{u}_m)$ be free in formula $\boldsymbol{P}$, then at s substituted for $\boldsymbol{x}$, replace $\boldsymbol{x}(A_i)$ by $d_i \in \boldsymbol{D}$, where $\langle A_i, d_i\rangle \in s$ (d_i is value of attribute A_i in tuple s). Atom $\boldsymbol{p}(\boldsymbol{u}_1, \dots, \boldsymbol{u}_m)$ acquires the value of true, if predicate $\boldsymbol{p}$ is true on the proper values, otherwise atom acquires the value of false.

The set of values of truth of all atoms of formula is called interpretation. Let $\boldsymbol{P}$ be a legal formula with no free tuple variables. The interpretation of formula $\boldsymbol{P}$ is defined as follows.

f1. If $\boldsymbol{P} = \neg\boldsymbol{G}$, then $\boldsymbol{G}$ must have no free variables. The formula $\boldsymbol{P}$ is true, if $\boldsymbol{G}$ is false, and it is false, if formula $\boldsymbol{G}$ is true.

f2. If $\boldsymbol{P} = \boldsymbol{G}\wedge\boldsymbol{Q}$ or $\boldsymbol{P} = \boldsymbol{G}\vee\boldsymbol{Q}$, then neither $\boldsymbol{G}$ or Q have free variables. If $\boldsymbol{P} = \boldsymbol{G}\wedge\boldsymbol{Q}$, then $\boldsymbol{P}$ is true exactly when $\boldsymbol{G}$ and $\boldsymbol{Q}$ are both true, otherwise, $\boldsymbol{P}$ is false. If $\boldsymbol{P} = \boldsymbol{G}\vee\boldsymbol{Q}$, then $\boldsymbol{P}$ is false exactly when $\boldsymbol{G}$ and $\boldsymbol{Q}$ are both false, otherwise, $\boldsymbol{P}$ is true.

f3. If $\boldsymbol{P} = \exists\boldsymbol{x}(R)\boldsymbol{G}$, then $\boldsymbol{x}$ is the only variable that occurs free in $\boldsymbol{G}$. The formula $\boldsymbol{P}$ is true, if there is at least one tuple $s \in S(R)$ such that formula $\boldsymbol{G}(s/\boldsymbol{x})$ is true, otherwise, formula $\boldsymbol{P}$ is false.

f4. If $\boldsymbol{P} = \forall\boldsymbol{x}(R)\boldsymbol{G}$, then $\boldsymbol{x}$ is the only variable that occurs free in $\boldsymbol{G}$. ***The formula P is true***, if for every tuple $s \in S(R)$ formula $\boldsymbol{G}(s/\boldsymbol{x})$ is true, otherwise, formula $\boldsymbol{P}$ is false.

If $\boldsymbol{P} = (\boldsymbol{G})$, then $\boldsymbol{P}$ is true, if formula $\boldsymbol{G}$ is true and $\boldsymbol{P}$ is false, if $\boldsymbol{G}$ is false.

Let $E = \{\boldsymbol{x}^n(R)|\boldsymbol{P}(\boldsymbol{x})\}$ be a tuple calculus expression. The value of expression E is the table $\langle\varphi, R\rangle$ of multiset table algebra containing every tuple $s \in S(R)$ such that formula $\boldsymbol{P}(s/\boldsymbol{x})$ is true. The number of duplicates of the tuple s in the table $\langle\varphi, R\rangle$ is determined as follows:

1) if $\boldsymbol{P} = \alpha_R(\boldsymbol{x})$, then $n = Occ(s,\varphi) = Occ(s,\alpha)$;
2) if $\boldsymbol{P} = X_R(\boldsymbol{x})$, then $n = Occ(s,\varphi) = Occ(s,X)$;
3) if $\boldsymbol{P} = \boldsymbol{p}(\boldsymbol{u}_1, \dots, \boldsymbol{u}_m)$, where $\boldsymbol{u}_j$ are terms, $j = \overline{1,m}$, and $\boldsymbol{x}_1, \dots, \boldsymbol{x}_k$ are all variables of these terms, then

$$n = Occ(s,\varphi) = \sum_{\substack{s'\in\Theta(\psi), s'|R'=s \\ \boldsymbol{p}(\boldsymbol{u}_1,\dots,\boldsymbol{u}_m)=true}} Occ(s',\psi),$$

where $\langle\psi, R\rangle$ is the table to which the query is constructed, $R' = \bigcup_{j=1}^{m} attr(\boldsymbol{x}_i, \boldsymbol{u}_j), i = \overline{1,k}$;

4) if $\boldsymbol{P} = \neg\boldsymbol{G}$ and the formula $\boldsymbol{G}$ generates m duplicates of the tuple s, then
5)

$$n = Occ(s,\varphi) = Occ(s, C(\psi)) \dot{-} m,$$

where $C(\psi)$ is a multiset of the table $C(\langle\psi, R\rangle)$ which is the saturation of the table $\langle\psi, R\rangle$, which is the value of the expression $\{\boldsymbol{y}^n(R)|\boldsymbol{P}(\boldsymbol{y})\}$, and $x \dot{-} y = \begin{cases} x - y \; if \; x \geq y \\ 0 \; if \; x < y \end{cases}$;

6) if $\boldsymbol{P} = \boldsymbol{G} \wedge \boldsymbol{Q}$, the formula $\boldsymbol{G}$ generates k duplicates of the tuple s, and the formula $\boldsymbol{Q}$ generates m duplicates of the tuple s, then $n = Occ(s, \varphi) = \min(k, m)$;
7) if $\boldsymbol{P} = \boldsymbol{G} \vee \boldsymbol{Q}$, the formula $\boldsymbol{G}$ generates k duplicates of the tuple s, and the formula $\boldsymbol{Q}$ generates m duplicates of the tuple s, then $n = Occ(s, \varphi) = k + m$;
8) if $\boldsymbol{P} = \exists \boldsymbol{x}(R)\boldsymbol{G}$ and the formula $\boldsymbol{G}$ generates k duplicates of the tuple s, then $n = Occ(s, \varphi) = k$.
9) if $\boldsymbol{P} = \forall \boldsymbol{x}(R)\boldsymbol{G}$ and the formula $\boldsymbol{G}$ generates k duplicates of the tuple s, then $n = Occ(s, \varphi) = k$.
10) if $\boldsymbol{P} = (\boldsymbol{G})$ and the formula $\boldsymbol{G}$ generates k duplicates of the tuple s, then $n = Occ(s, \varphi) = k$.

Theorem 2. If F is a multiset table algebra expression, then it is possible effectively to build equivalent to it expression E of tuple calculation.

Proof. According to Theorem 1, in the proof, it is sufficient to consider expressions of multiset table algebra that contain only the operations of union, difference, selection, projection, join, and renaming. We will prove this theorem by mathematical induction on the number of operations in the expression F.

Basis. In this case, the expression F does not contain any operations. There are two cases.
First case. If $F = \langle\alpha, R\rangle$ is constant table, where α is a multiset of the scheme R, then $E = \{\boldsymbol{x}^n(R)|\alpha_R(\boldsymbol{x})\}$.
Second case. If $F = \langle X, R\rangle$ is variable table, then $E = \{\boldsymbol{x}^n(R)|X_R(\boldsymbol{x})\}$.

Induction. Assume the theorem holds for any multiset table algebra expression with fewer than i operators. Let F have i operators. There are six cases.

Case 1 (union). $F = F_1 \bigcup_{All}^{R} F_2$,

where expressions F_1 and F_2 each have less than i operators. By the inductive hypothesis we can find tuple calculus expressions $\{\boldsymbol{x}^n(R)|\boldsymbol{P}(\boldsymbol{x})\}$ and $\{\boldsymbol{x}^m(R)|\boldsymbol{Q}(\boldsymbol{x})\}$ equivalent to F_1 and F_2 respectively. The values of these expressions are tables in which the tuple $\boldsymbol{x}$ appears n and m times, respectively. Then E is $\{\boldsymbol{x}^k(R)|\boldsymbol{P}(\boldsymbol{x}) \vee \boldsymbol{Q}(\boldsymbol{x})\}$, where $k = n + m$.

Case 2 (difference). $F = F_1 \setminus_{All}^{R} F_2$,

where expressions F_1 and F_2 each have less than i operators. Tuple calculus expressions $\{\boldsymbol{x}^n(R)|\boldsymbol{P}(\boldsymbol{x})\}$ and $\{\boldsymbol{x}^m(R)|\boldsymbol{Q}(\boldsymbol{x})\}$ are equivalent to F_1 and F_2 respectively as in Case 1. Then $E = \{\boldsymbol{x}^k(R)|\boldsymbol{P}(\boldsymbol{x}) \wedge \neg \boldsymbol{Q}(\boldsymbol{x})\}$, where $k = n \dot{-} m$.

Case 3 (selection). $F = \sigma_{\tilde{p},R}(F_1)$,

where expression F_1 has less than i operators. Let $\{\boldsymbol{x}^n(R)|\boldsymbol{P}(\boldsymbol{x})\}$ be tuple calculus expression equivalent to F_1. Then $E = \{\boldsymbol{x}^k(R)|\boldsymbol{P}(\boldsymbol{x}) \wedge \boldsymbol{p}(\boldsymbol{x}(A_1), \dots, \boldsymbol{x}(A_m))\}$, where $R = \{A_1, \dots, A_m\}$ is the scheme of table that is the value of expression F_1. The number of duplicates of the tuple $\boldsymbol{x}$ in the output table does not change, therefore $k = n$. Assume that predicate-parameter of select is defined as $\tilde{p}(s) = true \Leftrightarrow \boldsymbol{p}\big(s(A_1), \dots, s(A_m)\big) = true$, $s \in S(R)$, where $\boldsymbol{p}$ is a predicate symbol of arity m. □

Case 4 (projection). $F = \pi_{X,R}(F_1)$.

Let $\{\boldsymbol{x}^n(R)|\boldsymbol{P}(\boldsymbol{x})\}$ be tuple calculus expression equivalent to F_1. Then $E = \{\boldsymbol{y}^k(X \cap R)|\exists \boldsymbol{x}(R)(\boldsymbol{P}(\boldsymbol{x}) \wedge \bigwedge_{A \in X \cap R} \boldsymbol{y}(A) = \boldsymbol{x}(A))\}$, where $k = \sum_{\bigwedge_{A \in X \cap R} \boldsymbol{y}(A) = \boldsymbol{x}(A)} n$.

Case 5 (join). $F = F_1 \underset{R_1, R_2}{\otimes} F_2$.

Tuple calculus expressions $\{\boldsymbol{x}^n(R)|\boldsymbol{P}(\boldsymbol{x})\}$ and $\{\boldsymbol{x}^m(R)|\boldsymbol{Q}(\boldsymbol{x})\}$ are equivalent to F_1 and F_2 respectively Then E is

$\{\boldsymbol{z}^k(R_1 \cup R_2)|\exists \boldsymbol{x}(R_1) \exists \boldsymbol{y}(R_2)(\boldsymbol{P}(\boldsymbol{x}) \wedge \boldsymbol{Q}(\boldsymbol{x}) \wedge \bigwedge_{A \in R_1} \boldsymbol{z}(A) = \boldsymbol{x}(A) \wedge \bigwedge_{A \in R_2} \boldsymbol{z}(A) = \boldsymbol{y}(A))\}$, where $k = n \times m$. □

Case 6 (renaming). $F = Rt_{\xi,R}(F_1)$,

where $\xi: \boldsymbol{A} \xrightarrow{\sim} \boldsymbol{A}$ is injective function that renames attributes. Let $\{\boldsymbol{x}^n(R)|\boldsymbol{P}(\boldsymbol{x})\}$ be tuple calculus expression equivalent to F_1. Then

$E = \{\boldsymbol{y}^k(R_2)|\exists \boldsymbol{x}(R)(\boldsymbol{P}(\boldsymbol{x}) \wedge \wedge_{C \in R \setminus dom\xi}\, \boldsymbol{y}(C) = \boldsymbol{x}(C) \wedge \wedge_{A \in R \cap dom\xi}\, \boldsymbol{x}(A) = \boldsymbol{y}(\xi(A)))\}$, *where* $R_2 = R \setminus dom\xi \cup \xi[R]$, *and* $k = n$.

5. Relation to the SQL query language

The fundamental data object in the SQL language is not the classical relation of E. Codd but rather a table. Moreover, SQL tables do not contain sets of tuples but multisets of tuples, meaning duplicates are allowed. The basic SQL operators are not relational operators in the strict sense but are analogs of relational operators designed to work with multisets. When creating a new table using a query, an SQL system typically does not remove duplicate tuples but returns a result in which the same tuple can appear multiple times. To exclude duplicates, the keyword DISTINCT must be placed after the SELECT operator.

Let's demonstrate the appropriateness of constructing tuple calculus for multiset table algebra with the following example.

Example 1. Consider the table $\langle \text{Scores}, R \rangle$, where the scheme $R = \{\text{№},\ \text{Name},\ \text{Topic 1},\ \text{Topic 2},\ \text{Topic 3},\ \text{Quiz}\}$ (see Table 1).

Table 1
Table $\langle \text{Scores}, R \rangle$

№	Name	Topic 1	Topic 2	Topic 3	Quiz
1.	Student 1	5	15	14	16
2.	Student 2	6	14	15	16
3.	Student 3	7	17	20	20
4.	Student 4	9	20	19	20
5.	Student 5	5	18	15	18
6.	Student 6	6	19	13	20
7.	Student 7	4	8	9	16

The answer to the question "What scores did the first five students get for the quiz?" in SQL would look like this:

```
SELECT Quiz
FROM Scores
LIMIT 5;
```

The result is a table $\langle \text{Quiz}, R_1 \rangle$, where the schema $R_1 = \{\text{Quiz}\}$, which contains duplicate values (see Table 2).

Table 2
Table $\langle \text{Quiz}, R_1 \rangle$

Quiz
16
16
20
20
18

It is impossible to implement this query in terms of classical tuple calculus, since the result will be a set of tuples, not a multiset (as expected), meaning duplicates will not be considered in the result.

In the tuple calculus for multiset table algebra, the expression equivalent to this query will have the form:

$$\{x^n(\text{Quiz}) \mid \square y(R)(\text{Scores}(y) \wedge (\text{№}) = 1 \vee (\text{№}) = 2 \vee (\text{№}) = 3 \vee (\text{№}) = 4 \vee (\text{№}) = 5) \wedge \\ \wedge x(\text{Quiz}) = y(\text{Quiz}))\}.$$

The result described by this tuple calculus expression is similar to the result obtained when executing the corresponding query in SQL.

Example 2. Consider the table $\langle \text{Results}, R' \rangle$, where the scheme $R' = \{ \text{№}, \text{Name}, \text{Total}, \text{ECTS}\}$ (see Table 3).

Let's consider the query "How many points for the quiz have students who scored more than 83 points in total?".

We will write the corresponding expression of multiset table algebra, assuming that we only need to know the quiz scores:

$$F = \pi_{\text{Quiz}} \left(\langle \text{Scores}, R \rangle \underset{R,R'}{\otimes} \sigma_{Total>83} \langle \text{Results}, R' \rangle \right).$$

The result is a table $\langle \text{Query}, \{\text{Quiz}\} \rangle$, which contains duplicate values (see Table 4).

Table 3
Table $\langle \text{Results}, R' \rangle$

№	Name	Total	ECTS
1.	Student 1	77	C
2.	Student 2	80	C
3.	Student 3	93	A
4.	Student 4	90	A
5.	Student 5	81	C
6.	Student 6	79	C
7.	Student 7	51	Fx

From Table 4, it is clear that only two students in total have more than 83 points. Let's write the tuple calculus expression equivalent to this algebraic expression.

Table 4
Table$\langle \text{Query}, \{\text{Quiz}\} \rangle$

Quiz
20
20

Tuple calculus expressions $\{\boldsymbol{x}^n(R) | \text{Scores}(\boldsymbol{x})\}$ and $\{\boldsymbol{y}^m(R') | \boldsymbol{Results}(\boldsymbol{y})\}$ are equivalent to $\langle \text{Scores}, R \rangle$ and $\langle \text{Results}, R' \rangle$ respectively, where $R = \{\text{№}, \text{Name}, \text{Topic 1}, \text{Topic 2}, \text{Topic 3}, Quiz\}$ *and* $R' = \{ \text{№}, Name, Total, ECTS\}$.

We have tuple calculus expression

$$\{\boldsymbol{y}^m(R') | \text{Results}(\boldsymbol{y}) \wedge \boldsymbol{y}(\text{Total}) > 83\}$$

for algebraic expression

$$\sigma_{Total>83} \langle \text{Results}, R' \rangle.$$

Its value is a table without duplicates because table $\langle \text{Results}, R' \rangle$ does not have duplicates.
Tuple calculus expressions which equivalent to algebraic expression

$$\langle \text{Scores}, R\rangle \underset{R,R'}{\otimes} \sigma_{Total>83}\langle \text{Results}, R'\rangle$$

is

$$\{\boldsymbol{z}^{k=n\times m}(R\cup R')|\exists \boldsymbol{x}(R)\exists \boldsymbol{y}(R')(\text{Scores}(\boldsymbol{x})\wedge\text{Results}(\boldsymbol{y})\wedge \boldsymbol{y}(\text{Total}) > 83\wedge \boldsymbol{z}(№) = \boldsymbol{x}(№)\wedge \boldsymbol{z}(\text{Name}) = \boldsymbol{x}(\text{Name})\wedge \boldsymbol{z}(№) = \boldsymbol{y}(№)\wedge \boldsymbol{z}(\text{Name}) = \boldsymbol{y}(\text{Name}))\}.$$

Since there are no duplicate tuples in the tables $\langle \text{Scores}, R\rangle$ and $\langle \text{Results}, R'\rangle$, there will also be no duplicate tuples in the resulting table after the join.

Finally, the tuple calculus expression equivalent to the algebraic expression F has the form:

$$\{\boldsymbol{w}^{\boldsymbol{q}}(\{\boldsymbol{Quiz}\}\cap(\boldsymbol{R}\cup\boldsymbol{R}'))|\exists \boldsymbol{z}(\boldsymbol{R}\cup\boldsymbol{R}')(\exists \boldsymbol{x}(\boldsymbol{R})\exists \boldsymbol{y}(\boldsymbol{R}')\big(\boldsymbol{Scores}(\boldsymbol{x})\wedge\boldsymbol{Results}(\boldsymbol{y})\wedge \boldsymbol{y}(\boldsymbol{Total}) > \boldsymbol{83}\wedge \boldsymbol{z}(№) = \boldsymbol{x}(№)\wedge \boldsymbol{z}(\boldsymbol{Name}) = \boldsymbol{x}(\boldsymbol{Name})\wedge \boldsymbol{z}(№) = \boldsymbol{y}(№)\wedge \boldsymbol{z}(\boldsymbol{Name}) = \boldsymbol{y}(\boldsymbol{Name})\big))\}.$$

In the table obtained after the join, the tuple $\boldsymbol{z}$ appears once. Therefore, in the output table, the number of duplicates of the tuple w is

$$\boldsymbol{q} = \sum_{\boldsymbol{A}\in\{\boldsymbol{Quiz}\}\cap(\boldsymbol{R}\cup\boldsymbol{R}')\boldsymbol{w}(\boldsymbol{A})=\boldsymbol{z}(\boldsymbol{A})} \boldsymbol{k}.$$

The result is a table $\langle \text{Query}, \{\text{Quiz}\}\rangle$, where $Occ(s, \text{Query}) = 1 + 1 = 2$, $s = \{\langle\{Quiz\}, 20\rangle\}$.
The corresponding SQL query has the form:

```
SELECT Quiz
FROM scores
INNER JOIN result ON scores.`№`=result.`№` AND scores.Name=result.Name
WHERE result.Total>83
```

Thus, as can be seen from the examples, the constructed tuple calculus for multiset table algebra allows for the adequate formalization of query languages, particularly SQL, considering the multiset semantics embedded in them.

Conclusions

The article proposes a tuple calculus for multiset table algebra. The alphabet, syntax of terms, atoms, and formulas of the tuple calculus are defined. Using the concept of free and bound tuple, the notion of scheme, and the set of attributes with which a tuple appears in formulas, a class of legal formulas is introduced. It is shown that the proposed tuple calculus is no less expressive than multiset table algebra. An example demonstrates the feasibility of constructing tuple calculus for multiset table algebra, considering that query languages oriented towards database work imply the repeatability of elements in a table.

The next research challenge is to establish the corresponding dual result.

References

[1] E.F. Codd, Relational Completeness of Data Base Sublanguages, in: Data Base Systems (1972) 65-98.

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

[3] V.N. Redko, et al., Relational Databases: Table Algebras and SQL-like Language. Kyiv: Publishing house Academperiodica, 2001. [in Ukrainian]

[4] D.B Buy, I.M. Glushko, Calculi and extensions of table algebras signature. Nizhyn: NDU im. M. Gogol, 2016. [in Ukrainian]
[5] I.Glushko, About relationship between table algebra of infinite tables and multiset table algebra, in CEUR Workshop Proceedings, 2139, 2018, pp. 159–163.
[6] I. Lysenko, Extended table algebra, extended multiset table algebra, and their relationship, in: Proceedings of International Conference on Software Engineering "SoftEngine 2020", 2020, pp 28-32.
[7] Paul W.P.J. Grefen, Rolf A. de By, A Multi-Set Extended Relational Algebra. A Formal Approach to a Practical Issue, in: Proceedings of 10th International Conference on Data Engineering, ICDE, 1994, pp. 80-88.
[8] Paul W.P.J. Grefen, J. Flokstra, Extending a Multi-Set Relational Algebra to a Parallel Environment, in: Distributed and Parallel Databases, Vol 4. (1996) 81-99.
[9] G. Lamperti, M. Melchiori, M. Zanella, On Multisets in Database Systems, in Multiset Processing: Mathematical, Computer Science, and Molecular Computing Points of View, number 2235 in Lecture Notes in Computing Since (2001)147-215.
[10] H. Garcia-Molina, J.D. Ullman, J. Widom, Database Systems: The Complete Book, 2nd. ed., Prentice Hall, 2008.
[11] J.D. Ullman, J. Widom.Ullman, A First Course in Database Systems, 3rd ed., Prentice Hall, 2007.
[12] A. Silbeschatz, H. Korth, S. Sudarshan, Database System Concepts, 6th ed. McGraw-Hill, 2011.
[13] J.A. Bogatyreva, Multisets theory and its applications. Ph.D. thesis, Kyiv National Taras Shevchenko University, 2011. [in Ukrainian]
[14] N.Cutland, Computability. An introduction to recursive function theory, Cambridge University Press, 1980.