ДОПОВІДІ
НАЦІОНАЛЬНОЇ
АКАДЕМІЇ НАУК
УКРАЇНИ

11 • 2016

ІНФОРМАТИКА

doi: https://doi.org/10.15407/dopovidi2016.11.017

УДК 004.655

И.С. Канарская

Киевский национальный университет им. Тараса Шевченко
E-mail: iren_kiss@mail.ru

Оценки сложности алгоритмов реализации теоретико-множественных операций в табличных алгебрах

(Представлено академиком НАН Украины В.Н. Редько)

Исследованы алгоритмы реализации пересечения, объединения и разности таблиц в табличных алгебрах: сначала рассматриваются наиболее естественные алгоритмы, а затем предлагаются их модификации, позволяющие уменьшить количество вычислений. Для всех предложенных алгоритмов найдены точные оценки сложности в худшем случае и в среднем, на основе которых были найдены наиболее быстрые алгоритмы для каждой операции. Проведены численные эксперименты, которые подтверждают теоретические оценки.

Ключевые слова: *сложность алгоритма, табличные алгебры, базы данных.*

В настоящее время системы управления базами данных широко используются во многих областях деятельности человека. Наиболее распространённой остается реляционная модель данных, впервые предложенная Коддом в 1970 г. [1]. Табличные алгебры, введённые В.Н. Редько и Д.Б. Буем [2], построены на основе реляционных алгебр Кодда и существенно их развивают. Они составляют теоретический фундамент языков запросов современных табличных баз данных. Элементы носителя табличной алгебры уточняют реляционные структуры данных, а сигнатурные операции построены на базе основных табличных манипуляций в реляционных алгебрах и SQL-подобных языках.

Одной из наиболее важных задач в современных базах данных является задача оптимизации запросов [3]. В ряде работ рассмотрены различные алгоритмы эффективного выполнения операции соединения, что в некоторых случаях позволяет оптимизировать запросы [4–7]. В настоящей работе рассмотрены алгоритмы выполнения операций пересечения, объединения, разности и найдена сложность этих алгоритмов. Эти алгоритмы используют как теоретико-множественную природу таблиц, так и их специфику — структуру строк. Найденные оценки сложности алгоритмов могут быть применены для вычисления “стоимости” запросов и эквивалентных преобразований выражений, а также выбора оптимального плана их выполнения.

Основные определения. Зафиксируем некоторое непустое множество атрибутов $\mathfrak{R} = \{A_1,..., A_n\}$. Произвольное конечное подмножество множества $\mathfrak{R}$ назовем схемой, причем схема может быть пустым множеством. Строкой $s=\{(A'_1,d_1),\ldots,(A'_k,d_k)\}$ схемы R называется множество пар, проекция которого по первой компоненте равна R, причем атрибуты $A'_1,\ldots,A'_k$ попарно различны, т.е. строка является функциональным бинарным отношением. Таблицей схемы R называется конечное множество строк схемы R, количество строк в таблице T обозначаем $|T|$. Далее, если не оговорено противное, в работе рассматриваются таблицы схемы R с количеством атрибутов n. На множестве всех таких таблиц введены операции пересечения $\bigcap\limits_R$, объединения $\bigcup\limits_R$ и разности $\underset{R}{-}$ таблиц как ограничения одноименных теоретико-множественных операций.

Табличной алгеброй называется частичная алгебра с носителем — множеством всех таблиц произвольной схемы — и девятью, вообще говоря, параметрическими операциями: пересечением, объединением, разностью, проекцией, селекцией, активным дополнением, соединением, делением и переименованием; определения шести из этих операций, не рассматриваемых в данной работе, приведены в [2, 8, 9].

Вычислительной сложностью алгоритма называется функция зависимости объёма работы, которая выполняется им, от размера входных данных. Рассматривают два вида вычислительной сложности: временную, которая оценивает время выполнения алгоритма (количество выполняемых вычислений), и емкостную, которая оценивает количество ячеек памяти, необходимое для работы алгоритма. Известны два подхода к определению временной сложности — использование равномерного (все элементарные операции имеют одинаковый вес) либо логарифмического (элементарные операции могут иметь различные веса) весов [10]; в настоящей работе используется равномерный вес. Выделяют три типа временной сложности алгоритма: сложность в худшем случае, сложность для почти всех входов и сложность в среднем. Исследование временной сложности алгоритма в худшем случае существенно проще, чем исследование других двух типов его сложности, хотя при анализе временной сложности обработки запросов наиболее приемлемым является использование сложности в среднем, поскольку именно она дает возможность оценить эффективность обработки запросов.

В работе [8] выделены основные преобразования таблиц: добавление *A D D (T, s)*, удаление *D E L (T, s)* строки s и замена *C H (T, s, s_1)* строки s строкой s_1 в таблице T; в алгоритмах, рассматриваемых в данной работе, будут использованы первые два преобразования, их сложность в худшем случае и в среднем принимаем равной n. Далее будем рассматривать алгоритмы реализации пересечения, объединения и разности в табличных алгебрах: для каждой операции сначала рассматривается наиболее естественный, на наш взгляд, алгоритм, после этого предлагаются его модификации, позволяющие уменьшить количество вычислений. Затем будем находить временную сложность в худшем случае и в среднем для каждого из этих алгоритмов.

Описание предложенных алгоритмов. Рассмотрим четыре алгоритма реализации операции пересечения таблиц. Сначала рассмотрим алгоритм, являющийся, на наш взгляд, наиболее естественным; этот алгоритм выдает результат пересечения таблиц T_1 и T_2 в новую таблицу T.

Алгоритм А 1.1. С каждой строкой $s_1 \in T_1$ сравниваем все строки таблицы T_2: если существует такая строка $s_2 \in T_2$, что $s_1 = s_2$, то строку s_1 добавляем в таблицу T и переходим к следующей строке таблицы T_1.

Далее рассмотрим модификацию алгоритма А 1.1, которая заключается в том, что, ввиду уникальности строк в таблицах, при нахождении в таблице T_2 строки, совпадающей с рассматриваемой строкой таблицы T_1, строку из таблицы T_2 можно удалить.

Алгоритм А 1.2. С каждой строкой $s_1 \in T_1$ сравниваем все строки таблицы T_2, и если существует такая строка $s_2 \in T_2$, что $s_1 = s_2$, то строку s_1 добавляем в таблицу T, строку s_2 удаляем из таблицы T_2 и переходим к следующей строке таблицы T_1.

Далее рассмотрим модификацию алгоритма А 1.1, которая выдает результат пересечения таблиц T_1 и T_2 в ту из исходных таблиц, у которой количество строк не больше, чем у другой таблицы.

Алгоритм А 1.3. Если $|T_1| \leqslant |T_2|$ то с каждой строкой $s_1 \in T_1$ сравниваем все строки таблицы T_2 и если не существует такой строки $s_2 \in T_2$, что $s_1 = s_2$, то строку s_1 удаляем из таблицы T_1, а если такая строка существует, то при ее обнаружении переходим к следующей строке таблицы T_1; результат содержится в таблице T_1. Если же $|T_1| > |T_2|$ то таблицы T_1 и T_2 меняются ролями.

Теперь рассмотрим компиляцию алгоритмов А 1.2 и А 1.3.

Алгоритм А 1.4. Если $|T_1| \leqslant |T_2|$ то с каждой строкой $s_1 \in T_1$ сравниваем все строки таблицы T_2, и если не существует такой строки $s_2 \in T_2$, что $s_1 = s_2$, то строку s_1 удаляем из таблицы T_1, а если такая строка существует, то строку s_2 удаляем из таблицы T_2 и переходим к следующей строке таблицы T_1; результат содержится в таблице T_1. Если же $|T_1| > |T_2|$, то таблицы T_1 и T_2 меняются ролями.

Рассмотрим три алгоритма реализации операции объединения таблиц. Сначала рассмотрим наиболее естественный, на наш взгляд, алгоритм, который выдает результат объединения таблиц T_1, и T_2, в новую таблицу T.

Алгоритм А 2.1. Добавляем все строки $s_1 \in T_1$ в новую таблицу T, а затем все строки s_2 таблицы T_2 сравниваем со строками таблицы T_1, и если не существует такой строки s_1, что $s_2 = s_1$, то строку s_2 добавляем в таблицу T, а если такая строка существует, то при ее обнаружении переходим к следующей строке таблицы T_1.

Далее рассмотрим модификацию (алгоритм А 2.2) алгоритма А 2.1, которая заключается в том, что после добавления всех строк таблицы T_1 в таблицу T мы сравниваем все строки таблицы T со строками таблицы T_2 (а не строки таблицы T_2 со строками таблицы T_1, как в алгоритме А 2.1).

Алгоритм А 2.2. Добавляем все строки $s_1 \in T_1$ в новую таблицу T, а затем все строки s таблицы T сравниваем со строками таблицы T_2, и если существует такая $s_2 \in T_2$, что $s = s_2$, то удаляем строку s_2 из таблицы T_2 и переходим к следующей строке таблицы T. После этого добавляем все оставшиеся строки таблицы T_2 в таблицу T.

Теперь рассмотрим алгоритм, который выдает результат объединения таблиц T_1 и T_2 в одну из исходных таблиц с большим количеством строк.

Алгоритм А 2.3. Если $|T_1| \leqslant |T_2|$, то с каждой строкой $s_2 \in T_2$ сравниваем все строки таблицы T_1, и если существует такая строка $s_1 \in T_1$, что $s_2 = s_1$, то удаляем строку s_1 из таблицы T_1 и переходим к следующей строке таблицы T_2; после этого добавляем все оставшиеся строки из таблицы T_1 в таблицу T_2, в которой содержится результат. Если же $|T_1| > |T_2|$, то таблицы T_1 и T_2 меняются ролями.

Рассмотрим пять алгоритмов реализации операции разности таблиц. Сначала рассмотрим наиболее естественный, на наш взгляд, алгоритм, который выдает результат разности таблиц T_1 и T_2 в новую таблицу T.

Алгоритм А 3.1. С каждой строкой $s_1 \in T_1$ сравниваем все строки таблицы T_2: если не существует такой строки $s_2 \in T_2$, что $s_1 = s_2$, то строку s_1 добавляем в новую таблицу T, а если такая строка существует, то при ее обнаружении переходим к следующей строке таблицы T_1.

Далее рассмотрим модификацию алгоритма А 3.1, которая заключается в том, что, ввиду уникальности строк в таблицах, при нахождении в таблице T_2 строки, совпадающей с рассматриваемой строкой таблицы T_1, строку из таблицы T_2 можно удалить.

Алгоритм А 3.2. С каждой строкой $s_1 \in T_1$ сравниваем все строки таблицы T_2: если существует такая строка $s_2 \in T_2$, что $s_1 = s_2$, то строку s_2 удаляем из таблицы T_2 и переходим к следующей строке таблицы T_1, если же такой строки не существует, то строку s_1 добавляем в новую таблицу T.

Далее рассмотрим модификацию алгоритма А 3.1, который выдает результат разности таблиц T_1 и T_2 в исходную таблицу T_1.

Алгоритм А 3.3. С каждой строкой $s_1 \in T_1$ сравниваем все строки таблицы T_2: если существует такая строка $s_2 \in T_2$, что $s_1 = s_2$, то строку s_1 удаляем из таблицы T_1.

Далее рассмотрим модификацию алгоритма А 3.3, которая заключается в том, что ввиду уникальности строк в таблицах, при нахождении в таблице T_2 строки, совпадающей с рассматриваемой строкой таблицы T_1, строку из таблицы T_2 можно удалить.

Алгоритм А 3.4. С каждой строкой $s_1 \in T_1$ сравниваем все строки таблицы T_2: если существует такая строка $s_2 \in T_2$, что $s_1 = s_2$, то строку s_1 удаляем из таблицы T_1, а строку s_2 удаляем из таблицы T_2.

Теперь рассмотрим модификацию алгоритма А 3.3, которая заключается в изменении порядка сравнения строк в исходных таблицах: в этом алгоритме строки таблицы T_2 сравниваются со строками таблицы T_1.

Алгоритм А 3.5. С каждой строкой $s_2 \in T_2$ сравниваем все строки таблицы T_1: если существует такая строка $s_1 \in T_1$, что $s_2 = s_1$, то строку s_1 удаляем из таблицы T_1 и переходим к следующей строке таблицы T_2.

Вычисление сложности предложенных алгоритмов. Была найдена сложность в худшем случае для всех предложенных алгоритмов, в которых полагали $|T_1| = m_1$ и $|T_2| = m_2$. Поскольку в каждом из предложенных алгоритмов количество вычислительных действий зависит от количества строк, одновременно принадлежащих обеим исходным таблицам (т.е. строк таблицы $T_1 \bigcap\limits_R T_2$), в процессе нахождения сложности каждого алгоритма сначала была найдена ее функциональная зависимость от числа строк таблицы $T_1 \bigcap\limits_R T_2$, а затем, в качестве сложности в худшем случае — максимум найденной функции. Заметим, что для каждой строки $s_1 \in T_1$ при поиске равной ей строки s_2 наибольшее количество вычислений происходит в случае, когда было перебрано максимально возможное количество строк таблицы $T_2(m_2)$, а при сравнении с s_1 других (отличных от s_2) строк таблицы T_2 было выполнено максимально возможное количество сравнений значений атрибутов (n). Также заметим, что в случае сравнения строк таблицы T_1 со всеми строками таблицы T_2, если заранее известно, что $\left|T_1 \bigcap\limits_R T_2\right| = j$, то необходимо сделать не более, чем $m_2 + (m_2 - 1) + \ldots + (m_2 - j + 1) = j \cdot m_2 - \dfrac{j \cdot (j-1)}{2}$ операций сравнения строк.

Сложностные характеристики для каждого из предложенных алгоритмов приведены в соответствующей табл. 1. Сложность в среднем алгоритмов А 1.1 — А 3.5 зависит от многих параметров исходных таблиц. Пусть $|T_1| = m_1$, $|T_2| = m_2$, $m = \min\{m_1, m_2\}$, T_1, T_2 — таблицы

схемы $R = \{A_1, ..., A_n\}$ мощности доменов атрибутов $A_1, ..., A_n$ равны соответственно $q_1, ..., q_n$, $q_1 \cdot \ldots \cdot q_n = Q$. Таким образом, существует ровно Q фиксированных строк, каждая из которых независимо от остальных может входить или не входить в каждую из исходных таблиц. Будем считать, что распределение значений по каждому атрибуту в обеих таблицах равномерное. Несложно видеть, что если $m_1 + m_2 \leqslant Q$, то минимальное количество строк таблицы $T_1 \bigcap\limits_R T_2$ равно 0, в противном случае минимальное количество строк таблицы $T_1 \bigcap\limits_R T_2$ равно $(m_1 + m) - Q$, это минимальное количество строк обозначим z; из определения пересечения таблиц очевидно вытекает, что максимально возможное количество строк таблицы $T_1 \bigcap\limits_R T_2$ равно m. Поскольку все предложенные алгоритмы используют сравнение строк в таблицах T_1 и T_2 на предмет их равенства и сложность всех этих алгоритмов зависит от количества строк

Таблица 1. **Сложностные характеристики предложенных алгоритмов**

Алгоритм	Сложность в худшем случае	Значение $W(j)$ для алгоритма
A 1.1	$m_1 \cdot m_2 \cdot n + n$	$\left(m_1 \cdot m_2 - \frac{j}{2} \cdot (m_2 + 1)\right) \cdot W' + 2 \cdot j \cdot n$
A 1.2	$m_1 \cdot m_2 \cdot n + 3 \cdot n$	$\left(m_1 \cdot m_2 - \frac{j}{2} \cdot \left(m_1 + m_2 + \frac{1}{2} - \frac{j}{2}\right)\right) \cdot W' + 3 \cdot j \cdot n$
A 1.3	$m_1 \cdot m_2 \cdot n + \min\{m_1, m_2\} \cdot n + m_1 + m_2 + 1$	$\left(m_1 \cdot m_2 - \frac{j}{2} \cdot (\max\{m_1, m_2\} + 1)\right) \cdot W' + \min\{m_1, m_2\} \cdot n + m_1 + m_2 + 1$
A 1.4	$m_1 \cdot m_2 \cdot n + \min\{m_1, m_2\} \cdot n + m_1 + m_2 + 1$	$\left(m_1 \cdot m_2 - \frac{j}{2} \cdot \left(m_1 + m_2 + \frac{1}{2} - \frac{j}{2}\right)\right) \cdot W' + \min\{m_1, m_2\} \cdot n + j \cdot n + m_1 + m_2 + 1$
A 2.1	$m_1 \cdot m_2 \cdot n + m_1 \cdot n + m_2 \cdot n$	$\left(m_1 \cdot m_2 - \frac{j}{2} \cdot (m_2 + 1)\right) \cdot W' + m_1 \cdot n + m_2 \cdot n$
A 2.2	$m_1 \cdot m_2 \cdot n + m_1 \cdot n + m_2 \cdot n$	$\left(m_1 \cdot m_2 - \frac{j}{2} \cdot \left(m_1 + m_2 + \frac{1}{2} - \frac{j}{2}\right)\right) \cdot W' + m_1 \cdot n + m_2 \cdot n + m_1 + j \cdot n$
A 2.3	$m_1 \cdot m_2 \cdot n + \min\{m_1, m_2\} \cdot n + m_1 + m_2 + 1$	$\left(m_1 \cdot m_2 - \frac{j}{2} \cdot \left(m_1 + m_2 + \frac{1}{2} - \frac{j}{2}\right)\right) \cdot W' + \min\{m_1, m_2\} \cdot n + j \cdot n + m_1 + m_2 + 1$
A 3.1	$m_1 \cdot m_2 \cdot n + m_1 \cdot n$	$\left(m_1 \cdot m_2 - \frac{j}{2} \cdot (m_2 + 1)\right) \cdot W' + m_1 \cdot n$
A 3.2	$m_1 \cdot m_2 \cdot n + m_1 \cdot n$	$\left(m_1 \cdot m_2 - \frac{j}{2} \cdot \left(m_1 + m_2 + \frac{1}{2} - \frac{j}{2}\right)\right) \cdot W' + m_1 \cdot n + j \cdot n$
A 3.3	$m_1 \cdot m_2 \cdot n + n$	$\left(m_1 \cdot m_2 - \frac{j}{2} \cdot (m_2 + 1)\right) \cdot W' + 2 \cdot j \cdot n$
A 3.4	$m_1 \cdot m_2 \cdot n + 3 \cdot n$	$\left(m_1 \cdot m_2 - \frac{j}{2} \cdot \left(m_1 + m_2 + \frac{1}{2} - \frac{j}{2}\right)\right) \cdot W' + 3 \cdot j \cdot n$
A 3.5	$m_1 \cdot m_2 \cdot n + n$	$\left(m_1 \cdot m_2 - \frac{j}{2} \cdot \left(m_1 + m_2 + \frac{1}{2} - \frac{j}{2}\right)\right) \cdot W' + 2 \cdot j \cdot n$

таблицы $T_1 \bigcap\limits_R T_2$, были найдены вероятности $P(j)$ того, что таких строк будет в точности j ($j \in \{z, ..., m\}$):

$$P(j)=\frac{C_Q^j \cdot C_{Q-j}^{m_1-j} \cdot C_{Q-m_1}^{m_2-j}}{C_Q^{m_1} \cdot C_Q^{m_1 2}}=\frac{m_1! \cdot m_2! \cdot (Q-m_1)! \cdot (Q-m_2)!}{j! \cdot (m_1-j)! \cdot (m_2-j)! \cdot (Q-m_1-m_2+j)! \cdot Q!}.$$

Далее, для каждого из предложенных алгоритмов было определено значение $W(j)$ среднего количества вычислительных действий при условии $\left|T_1 \bigcap\limits_R T_2\right| = j$. Сложностью в среднем считаем величину $\sum\limits_{j=z}^{m} P(j) \cdot W(j)$.

Для нахождения $W(j)$ было найдено среднее количество вычислительных действий W', необходимое для сравнения строк $s_1 \in T_1$ и $s_2 \in T_2$ в случае, когда $s_1 \neq s_2$:

$$W' = \frac{Q}{Q-1} \cdot \left(1 \cdot \frac{q_1 - 1}{q_1} + 2 \cdot \frac{q_2 - 1}{q_1 \cdot q_2} + \ldots + n \cdot \frac{q_n - 1}{Q}\right).$$

Для сложности в среднем в табл. 1 указаны только значения $W(j)$, поскольку значения $P(j)$ во всех алгоритмах одинаковы и зависят только от параметров исходных таблиц.

Из предложенных алгоритмов для операции пересечения наименьшую сложность в худшем случае имеет алгоритм А 1.1, а в среднем — у алгоритма А 1.4; для объединения наименьшая сложность в худшем случае и в среднем у алгоритма А 2.3; для разности наименьшая сложность в худшем случае у алгоритмов А 3.3 и А 3.5, а в среднем — у алгоритма А 3.5.

Для экспериментального подтверждения результатов была разработана программа в среде Lazarus, которая вычисляет фактическое количество выполненных вычислений для каждого из предложенных алгоритмов и сравнивает их с предложенными теоретическими оценками; также программа может вычислять среднее значение фактического количества выполненных вычислений для любой серии опытов, в каждом из которых исходные таблицы формируются случайным образом. Проведенные эксперименты показали, что уже для относительно небольшого количества опытов (от 100) средние значения фактического количества выполненных вычислений отличаются от расчетных на величину, не превосходящую 0,1 %, а при увеличении количества опытов различие между фактическим количеством выполненных вычислений и расчетным уменьшается.

В дальнейшем предполагается аналогичным образом исследовать алгоритмы реализации остальных сигнатурных операций табличных алгебр, а также алгоритмы реализаций операций над таблицами в виде мультимножеств, у которых строки могут повторяться.

ЦИТИРОВАННАЯ ЛИТЕРАТУРА

1. *Codd E.F.* A Relational Model of Data for Large Shared Data Banks // Communications of the ACM. — 1970. — **13**, No 6. — P. 377–387.
2. *Редько В.Н., Буй Д.Б.* К основаниям теории реляционных моделей баз данных // Кибернетика и систем. анализ. — 1996. — **32**, № 4 — С. 3—12.
3. *Knuth D.E.* The Art of Computer Programming, Volume 4, Fascicle 0: Introduction to Combinatorial Algorithms and Boolean Functions — Boston: Addison-Wesley Professional, 2008. — 240 p.
4. *Jarke M., Koch J.* Query Optimization in Database Systems // ACM Computing Surveys. — 1984. — **16**, No 2. — P. 111—152.

5. *Valduriez P.* Join Indices // ACM Transactions on Database Systems. — 1987. – **12**, No 2. — P. 218-246.
6. *Кузнецов С.Д.* Методы оптимизации выполнения запросов в реляционных СУБД // Итоги науки и техники. Вычислительные науки. 1. — Москва: ВИНИТИ, 1989. С. 76 153.
7. *Мендкович Н.А., Кузнецов С.Д.* Обзор развития методов лексической оптимизации запросов // Труды ИСП РАН. — 2012. — 23. — С. 195-214.
8. *Мейер Д.* Теория реляционных баз данных — Москва: Мир, 1987. — 608 с.
9. *Редько В.Н., Брона Ю.Й., Буй Д.Б., Поляков С.А.* Реляційні бази даних: табличні алгебри та SQL-подібні мови. — Київ: ВД Академперіодика, 2001. — 198 с.
10. *Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К.* Алгоритмы: построение и анализ. — Москва: Вильямс, 2013. — 1328 с.

REFERENCES

1. *Codd E.F.* Communications of the ACM, 1970, **13,** No 6: 377–387.
2. *Red'ko V.N., Bui D.B.* Cybernetics and Systems Analysis, 1996, **32,** No 4: 471-478.
3. *Knuth D.E.* The Art of Computer Programming, Volume 4, Fascicle 0: Introduction to Combinatorial Algorithms and Boolean Functions, Addison-Wesley Professional, 2008.
4. *Jarke M., Koch J.* ACM Computing Surveys, 1984, **16,** No 2: 111-152.
5. *Valduriez P.* ACM Transactions on Database Systems, 1987, **12,** No 2: 218-246.
6. *Kuznetsov S.D.* Itogi nauki i tekhniki. Vychislitelnye nauki, 1989. **1**: 76-153 (in Russian).
7. *Mendkovich N.A., Kuznetsov S.D.* Proc. of the Institute for System Programming, 2012, **23:** 195-214 (in Russian).
8. *Maier D.* The Theory of Relational Databases, Comp. Sc. Press, 1983.
9. *Red'ko V.N., Brona J.J., Buy D.B., Polyakov S.V.* Relational data base: table algebras and family of the SQL languages, Kiev, Akademperiodyka, 2001 (in Ukrainian).
10. *Cormen T., Leiserson, Ch., Rivest R. Stein C.* Introduction to Algorithms. 3rd edition, MIT Press, 2009.

Поступило в редакцию 01.03.2016

І.С. Канарська

Київський національний університет ім.Тараса Шевченка
E-mail: iren_kiss@mail.ru

ОЦІНКИ СКЛАДНОСТІ АЛГОРИТМІВ РЕАЛІЗАЦІЇ ТЕОРЕТИКО-МНОЖИННИХ ОПЕРАЦІЙ В ТАБЛИЧНИХ АЛГЕБРАХ

Досліджено алгоритми реалізації перетину, об'єднання та різниць таблиць в табличних алгебрах: спочатку розглядаються найбільш природні алгоритми, а потім пропонуються їх модифікації, які дозволяють зменшити кількість обчислень. Для усіх запропонованих алгоритмів знайдено точні оцінки складності в найгіршому випадку та у середньому, на основі яких було знайдено найбільш швидкі алгоритми для кожної операції. Проведені обчислювальні експерименти, які підтверджують теоретичні оцінки.

Ключові слова: *складність алгоритму, табличні алгебри, бази даних.*

I.S. Kanarskaya

Taras Shevchenko National University of Kiev
E-mail: iren_kiss@mail.ru

ESTIMATES OF THE COMPLEXITY OF ALGORITHMS OF IMPLEMENTATION OF SET-THEORETIC OPERATIONS IN TABLE ALGEBRAS

The algorithms of implementation of the intersection, union, and difference of tables in the table algebras are investigated. A modification of the most common algorithms reducing the amount of computation is proposed. Based on the evaluated complexities in the worst case and on the average for the modified algorithms, the fastest algorithms for each operation are found. The experiments, which confirm the theoretical estimates, are executed.

Keywords: *complexity of algorithms, table algebra, database.*