https://doi.org/10.25559/SITITO.020.202403.609-618
УДК 004.0272.26

Оригинальная статья

Анализ одного алгоритма операции Join

В. И. Мунерман*, Д. В. Мунерман
ФГБОУ ВО «Смоленский государственный университет», г. Смоленск, Российская Федерация
Адрес: 214000, Российская Федерация, г. Смоленск, ул. Пржевальского, д. 4
* vimoon@gmail.com

Аннотация

Реляционный подход к организации баз данных за время своего существования пережил достаточно коллизий и неоднократно подвергался серьезной, и часто, необоснованной критике. Одним из положений критики было возражение против операции Join, которая считалась крайне неэффективной. Однако эта операция существовала в самых ранних системах обработки данных. В ранних публикациях она относится к классу слияния нестрого упорядоченных информационных массивов. В дальнейшем она была названа операцией слияния нестрого упорядоченных файлов. В последних работах в области машинного обучения стало активно развиваться направление, связанное с реляционными базами данных. Современные аналитические системы в производственной, финансовой, банковской, медицинской и многих других сферах базируются на больших объемах, структурированных данных, которые хранятся в, как правило, реляционных базах данных. Важное направление ускорения реализации запросов в таких базах данных заключается в использовании методов параллельной обработки данных. А так как операция Join – самая сложная из всех остальных операций, составляющих запросы, вопросам выбора для ее реализации наиболее эффективных алгоритмов, которые, в свою очередь, могут быть просто и эффективно. В статье предложен алгебраический метод формализации операции Join. На основе этого метода предложен алгоритм ее реализации с использованием специфической структуры очереди, которая называется «черпак». Приведены результаты экспериментального анализа алгоритма, подтвердившие его преимущество перед алгоритмом, который реализован в использованной системе управления базами данных Microsoft SQL Server. Приведены методы параллельной реализации предложенного алгоритма. Для этой цели использован традиционный поточный подход и подход, основанный на архитектуре вычислительной системы с ассоциативным распределением ресурсов.

Ключевые слова: реляционная модель, операция Join, алгоритм слияния, параллельное программирование

Конфликт интересов: авторы заявляют об отсутствии конфликта интересов.

Для цитирования: Мунерман В. И., Мунерман Д. В. Анализ одного алгоритма операции Join // Современные информационные технологии и ИТ-образование. 2024. Т. 20, № 3. С. 609-618. https://doi.org/10.25559/SITITO.020.202403.609-618

Original article

Analysis of Some Algorithm of the Join Operation

V. I. Munerman*, D. V. Munerman
Smolensk State University, Smolensk, Russian Federation
Address: 4 Przhevalsky St., Smolensk 214000, Russian Federation
* vimoon@gmail.com

Abstract

The relational approach to database organization has faced enough collisions during its existence and has often been subjected to serious and, at times, unjustified criticism. One of the points of criticism was the objection to the Join operation, which was considered extremely inefficient. However, this operation existed in the earliest data processing systems. In early publications, it belongs to the class of merging non-strictly ordered information arrays. Later, it was called the operation of merging non-strictly ordered files. In recent works in the field of machine learning, the direction associated with relational databases has begun to actively develop. Modern analytical systems in manufacturing, finance, banking, medicine and many other areas are based on large volumes of structured data that are stored, as a rule, in relational databases. An important direction for accelerating the implementation of queries in such databases is the use of parallel data processing methods. And since the Join operation is the most complex of all other operations that make up queries, the issues of choosing the most effective algorithms for its implementation, which, in turn, can be simple and effective, are of great importance. The article proposes an algebraic method for formalizing the Join operation. Based on this method, an algorithm for its implementation using a specific queue structure called a “scoop” is proposed. The results of an experimental analysis of the algorithm are presented, confirming its advantage over the algorithm implemented in the used Microsoft SQL Server database management system. Methods for parallel implementation of the proposed algorithm are given. For this purpose, a traditional flow approach and an approach based on the architecture of a computing system with associative resource distribution are used.

Keywords: relational model, Join operation, merge algorithm, parallel programming

Conflict of interests: The authors declares no conflict of interest.

For citation: Munerman V.I., Munerman D.V. Analysis of Some Algorithm of the Join Operation. *Modern Information Technologies and IT-Education.* 2024;20(3):609-618. https://doi.org/10.25559/SITITO.020.202403.609-618

Введение

Реляционный подход к организации баз данных за время своего существования пережил достаточно коллизий и неоднократно подвергался серьезной, и часто, необоснованной критике. Одним из положений критики было возражение против операции Join, которая считалась крайне неэффективной. Однако эта операция существовала в самых ранних системах обработки данных. В книге[1] она относится к классу слияния нестрого упорядоченных информационных массивов. В дальнейшем в работе[2] она названа операцией слияния нестрого упорядоченных файлов. В последних работах в области машинного обучения стало активно развиваться направление, связанное с реляционными базами данных [1-6]. Время распознавания «кошечек и собачек» уходит в прошлое. Современные аналитические системы в производственной, финансовой, банковской, медицинской и многих других сферах базируются на больших объемах, структурированных данных, которые хранятся в, как правило, реляционных базах данных. Важное направление ускорения реализации запросов в таких базах данных заключается в использовании методов параллельной обработки данных. А так как операция Join – самая сложная из всех остальных операций, составляющих запросы, вопросам выбора для ее реализации наиболее эффективных алгоритмов, которые, в свою очередь, могут быть просто и эффективно распараллелены[3] [7-11]. В статье предложен алгебраический метод формализации операции Join. На основе этого метода предложен алгоритм ее реализации с использованием специфической структуры очереди, которая называется «черпак». Приведены результаты экспериментального анализа алгоритма, подтвердившие его преимущество перед алгоритмом, который реализован в использованной системе управления базами данных Microsoft SQL Server. Приведены методы параллельной реализации предложенного алгоритма. Для этой цели использован традиционный поточный подход и подход, основанный на архитектуре вычислительной системы с ассоциативным распределением ресурсов.

Формализация операции Join

Несмотря на то, что реляционная алгебра есть формализация методов обработки данных, в публикациях всех уровней, начиная от популярных статей и заканчивая серьезными и глубокими исследованиями, отсутствуют формальные определения операций. Как правило, операции задаются словесными описаниями реализующих их алгоритмов. Поэтому далее предлагается формализация операции Join.

В реляционном подходе данные представляются в виде отношений, в терминологии большинства современных систем управления базами данных (СУБД) – таблиц, то есть подмножеств декартовых произведений доменов атрибутов[4]. В тех случаях, когда эти домены, универсальные одноосновные алгебраические системы[5], в декартовом произведении участвуют их основные множества [12, 13]. Операция Join бинарная, то есть в ней участвуют два отношения со схемами $R_1(A_1, ..., A_p)$ и $R_2(B_1, ..., B_q)$. С точки зрения операций, реализованных в СУБД, эти атрибуты можно разделить на три группы:

1. Ключи отношения – атрибуты, однозначно идентифицирующие кортежи. Они необходимы для быстрого поиска и корректировки данных.
2. Ключи операций – атрибуты, идентифицирующие не только отдельные кортежи, но и их группы, обладающие некоторым общим свойством. Они необходимы для запросов, в которые включены групповые операции с использованием агрегатных функций и операция Join.
3. Атрибуты, не участвующие в идентификации кортежей, но характеризующие объекты предметной области и участвующие в вычислениях.

Тогда схемы таблиц можно представить следующим образом: $R_1(K_{11},..., K_{1l}, C_{l+1},..., C_{l+s}\ F_{l+s+1}..., F_p)$ и $R_2(K_{21},..., K_{2m}, C_{m+1},..., C_{m+s}\ F_{m+s+1}..., F_q)$, где атрибуты обозначенные буквой K – атрибуты группы 1, буквой C – атрибуты группы 2, буквой F – атрибуты группы 3. В дальнейшем рассматриваются свойства только атрибутов группы 2 – $C_1, ..., C_s$, которые будут называться ключами операции Join. Эти ключи полностью совпадают в обоих отношениях, а на их доменах заданы отношение эквивалентности и, при необходимости отношение, порядка.

Замечание. В реляционной модели данных не принято атрибуты $C_1, ..., C_s$ называть ключами. Но поскольку они используются для однозначной идентификации связанных по смыслу операции *Join* групп кортежей, в дальнейшем они будут называться ключами.

Поскольку на множестве ключей $\boldsymbol{C} = \{C_1, ..., C_s\}$ задано отношение эквивалентности, то отношения R_1 и R_2 представляют собой фактормножества по отношению эквивалентности порожденному множеством $\boldsymbol{C}$ и будут обозначаться $R_1/_C$ и $R_2/_C$. Каждому классу эквивалентности этих фактормножеств соответствует фиксированный набор $c_1^*, ..., c_s^*$ экземпляров (значений) ключей $C_1, ..., C_s$, который будет называться экземпляром множества ключей и обозначаться $C_i^*\ (i = 1, ..., K)$, где K – количество всех возможных классов эквивалентности. Классы эквивалентности отношений R_1 и R_2 будут обозначаться $R_1/_{C^*}$ и $R_2/_{C^*}$, $(i, j = 1, ..., K)$. На практике обычна ситуация, когда отношения-операнды содержат не все возможные экземпляры множества ключей. Это затрудняет формализацию рассматриваемой операции. Поэтому целесообразно предположить, что все возможные экземпляры множества ключей логически присутствуют в отношениях, но те экземпляры множества ключей, которые физически отсутствуют в отношении, содержат универсальный неопределенный кортеж Θ. Такие классы эквивалентности будут обозначаться Θ_i^*.

[1] Обработка информационных массивов в автоматизированных системах управления / В. М. Глушков, В. П. Гладун, Л. С. Лозинский, С. Б. Погребинский ; под общ. ред. акад. В. М. Глушкова. Киев: Наукова думка, 1970. 181 с.

[2] Гендель Е. Г., Мунерман В. И. Применение алгебраических моделей для синтеза процессов обработки файлов // Управляющие системы и машины. 1984. № 4. С. 69-72.

[3] Гарсиа-Молина Г., Ульман Д., Уидом Д. Системы баз данных. Полный курс.: Пер. с англ.: М.: Изд. дом «Вильямс», 2004. 1088 с.

[4] Дейт К. Дж. Введение в системы баз данных. Восьмое издание. Вильямс, 2008. 1328 с.; Мальцев А. И. Алгебраические системы. М.: Наука, 1970. 392 с.

[5] Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. М.: Мир, 1979. 536 с.

Пусть отношение R получается в результате выполнения следующей операции Inner Join над отношениями R_1 и R_2:

INSERT INTO R SELECT <список атрибутов из R_1 и R_2, [формулы]>
FROM R_1 Inner Join R_2 ON $R_1.C_1 = R_2.C_1$ AND ... AND $R_1.C_s = R_2.C_s$ (1)
[GROUP BY <список атрибутов>] [ORDER BY <список атрибутов>]

Общая формула этой операции имеет вид: $R = \bigcup_{i=1}^{K} F(R/_{C_i^*})$.

Здесь $R_1/_{C_i^*}$ – класс эквивалентности отношения R, а $F(R/_{C_i^*})$ – функция, реализующая групповую операцию над кортежами, полученными в результате декартова произведения классов эквивалентности $R_1/_{C_i^*}$ и $R_2/_{C_i^*}$ по следующим правилам:

$$R/_{C_i^*} = \begin{cases} \Theta_i^*, \text{ если } R_1/_{C_i^*} = \Theta_i^* \text{ или } R_1/_{C_i^*} = \Theta_i^*, \\ R_1/_{C_i^*} \times R_2/_{C_i^*}, \text{ если } R_1/_{C_i^*} \neq \Theta_i^* \text{ и } R_1/_{C_i^*} \neq \Theta_i^*. \end{cases}$$

Алгоритм операции Join с использованием очереди типа «черпак»

Известно достаточно много реализаций рассматриваемой операции, например, алгоритм, основанный на вложенных циклах (Nested Loop), алгоритм слияния упорядоченных отношений (Sort-Merge Join), алгоритм, использующий хэш-функцию на множестве значений ключей для идентификации кортежей (Hash Join). Далее предлагается алгоритм схожий с алгоритмом слияния, но использующий специфический вид очереди, который называется «черпак» [14]. Черпак имеет следующую структуру:

{<Экземпляр множества ключей класса эквивалентности $C_i^* = (c_1^*, ..., c_s^*)$ >;
<Массив структур/записей, содержащих те элементы кортежей класса эквивалентности, которые переносятся в отношение результат и/или участвуют в вычислениях>}

Сущность алгоритма, реализующего операцию Inner Join из запроса (1) состоит в следующем:

1. Оба отношения R_1 и R_2 лексикографически упорядочиваются по множеству ключей операции Join – C_1, ..., C_s.
2. Одно из отношений, как правило, это отношение, классы эквивалентности которого содержат меньшее количество кортежей, определяется как ведущее, а второе – как ведомое.
3. Классы эквивалентности ведущего отношения поочередно считываются («зачерпываются») целиком в оперативную память и размещаются в черпаке. То есть, чтение отношения рассматривается не как простая операция чтения отдельных кортежей (записей), а как макрооперация считывания класса эквивалентности целиком.
4. После заполнения черпака очередным классом эквивалентности начинается/продолжается чтение ведомого отношения.
5. Ключ каждого прочитанного кортежа ведомого отношения сравнивается с ключом черпака.
6. Если значения ключей совпадают, производится совместная обработка кортежа ведомого отношения со всеми элементами массива черпака.
7. Если задана агрегатная функция $F(R/_{C_i^*})$, реализующая групповую операцию, то производится очередная итерация ее вычисления.
8. Этот процесс заканчивается, когда одно из отношений будет полностью прочитано.

Предложенный алгоритм реализует операцию Inner Join из запроса (1). Его достоинство состоит в том, что он реализует ее за один проход исходных отношений. Но, как и большинство алгоритмов этой операции, он требует достаточно много оперативной памяти для хранения класса эквивалентности в черпаке. Однако, в общем случае, он хранит не все атрибуты кортежей ведущего отношения, что уменьшает расход памяти. Это свойство, в сочетании со страничной организацией оперативной памяти в современных операционных системах и с быстродействующими внешними запоминающими устройствами типа твердотельных дисков (SSD), существенно снижает затраты времени при совместной обработке содержимого черпака с кортежем ведомого отношения.

Экспериментальный анализ алгоритма черпака

Эксперимент проводился в следующих условиях:

1. Вычислительная система – рабочая станция на основе процессора Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz 3.70 GHz, оперативная память 32 Гб, 64-разрядная операционная система Windows 10 Корпоративная.
2. Система управления базами данных – Microsoft SQL Server, SQL Server Management Studio 20.
3. Язык программирования для реализации алгоритма – C#.

В эксперименте обрабатывались два исходных отношения R_1(A, B) и R_2(A, C), в которых текстовый атрибут A типа nchar(10) – ключ операции Join, а числовые атрибуты B и C типа int используются в вычислениях. Результат – отношение R(A, sBC), содержащее ключ A и атрибут sBC, который вычисляется по формуле

$$sBC = \sum_{i=1}^{\mu(R_1/c^*)\times\mu(R_2/c^*)} B \times C$$

(μ – количество кортежей в классе эквивалентности). Фрагменты этих отношений приведены в таблице 1.

Программа, реализующая алгоритм черпака, работает со следующими данными:

1. Структура и переменная черпака

```
public struct TBucketR1
 { public string A;
   public int n;
   public int[] B;}
TBucketR1 Bucket;
```

2. Тексты команд для чтения отношений R_1 и R_2
“SELECT * FROM R1 ORDER BY R1.A” и “SELECT * FROM R2 ORDER BY R2.A”;
3. Рабочая таблица, соответствующая отношению R, которая будет заполнена программой и перенесена в базу данных посредством SqlBulkCopy

```
DataTable wR = new DataTable();
 wR.Columns.Add(“A”, typeof(string));
 wR.Columns.Add(“sBC”, typeof(int));
```

4. Структура и переменная записи – кортежа отношения R

```
public struct TR
 { public string A;
   public int sumBC;}
TR recR;
```

Таблица 1. **Фрагменты исходных и выходного отношений**

Table 1. **Fragments of the initial and output relations**

R_1		R_2		R	
A	B	A	C	A	sBC
BBBBBBAABA	2	BBBAABBAAB	8	AAAAAABAAB	297918
ABABABBBBA	8	AABBBBBBAB	3	AAAAABAABA	252862
BBBAABBAAB	3	AAAABAAAAA	6	AAAAABBBAA	206150
AABBBBBBAB	2	BBBAABABAB	8	AAAABAAABA	184851
AAAABAAAAA	1	AABABAAABB	4	AAAABAABAA	231570

Источник: здесь и далее в статье все рисунки и таблицы составлены авторами.

Source: Hereinafter in the article, all tables and figures are compiled by the authors.

На листинге 1 приведен фрагмент программы, реализующий пункты 6 и 7 алгоритма.

Листинг 1. Цикл обработки черпака и соответствующего класса эквивалентности отношения R_2

```
recR.sumBC = 0;
Bucket.n = 0;
while (readR1.Read())
{
   Bucket.n = Bucket.n + 1;
   Array.Resize(ref Bucket.B, Bucket.n);
   Bucket.B[Bucket.n-1] = readR1.GetInt32(1);
}
while (readR2.Read())
   for (i = 0; i < Bucket.B.Length; i = i + 1)
      recR.sumBC = recR.sumBC + Bucket.B[i] * readR2.GetInt32(1);
recR.A = Bucket.A;
wR.Rows.Add(recAB.A, recAB.sumBC);
```

Программа, реализующая алгоритм черпака выполнялась последовательно одним потоком, что иллюстрирует рисунок 1. При всех запусках программы, она занимала одно ядро процессора.

Запрос, реализующий операцию, аналогичную запросу (1) для отношений R_1, R_2 и R, имеет следующий вид

```
INSERT INTO R SELECT R1.A, SUM(R1.B*R2.C) AS sBC FROM R1
INNER JOIN R2 ON R1.A=R2.A GROUP BY R1.A ORDER BY R1.A.
```

Он реализован хранимой процедурой, что ускоряет процесс его выполнения, так как процедура компилируется, и поэтому исключается процесс интерпретации запроса. План запроса приведен на рисунке 2. На рисунке таблицы A, B и JAB соответствуют отношениям R_1, R_2 и R.

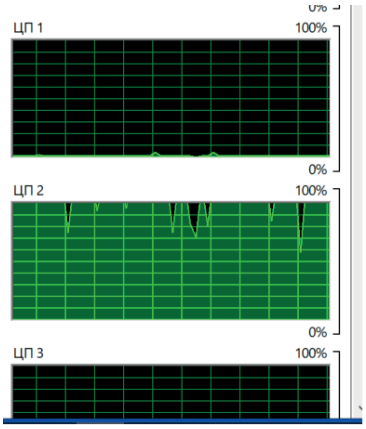


Рис. 1. Выполнение программы слияния алгоритмом черпака

Fig. 1. The execution of the merge program using the scoop algorithm

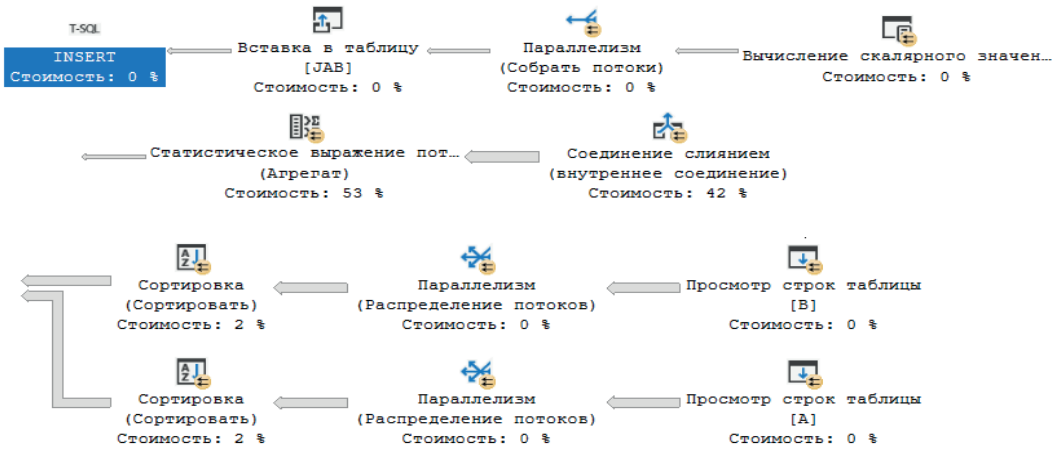
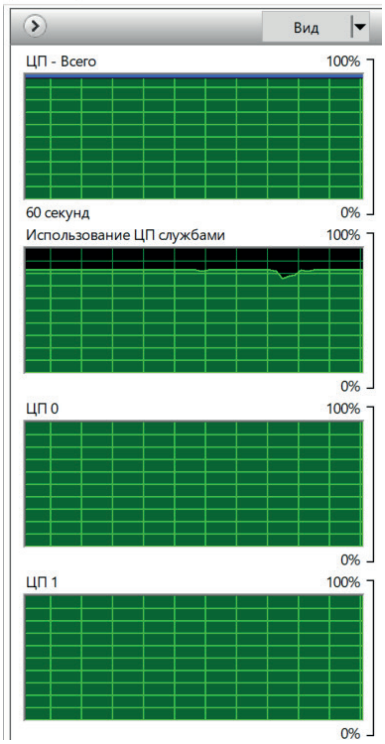


Рис. 2. План запроса, реализующего слияние операцией Join

Fig. 2. Query plan implementing merge using Join operation

План запроса свидетельствует о том, что используемая операция выполняется параллельно работающими потоками, причем были задействованы все двенадцать виртуальных ядер процессора. Это подтверждается рисунком 3.



Р и с. 3. Выполнение запроса, реализующего слияние операцией Join

F i g. 3. Executing a query that implements a merge using the Join operation

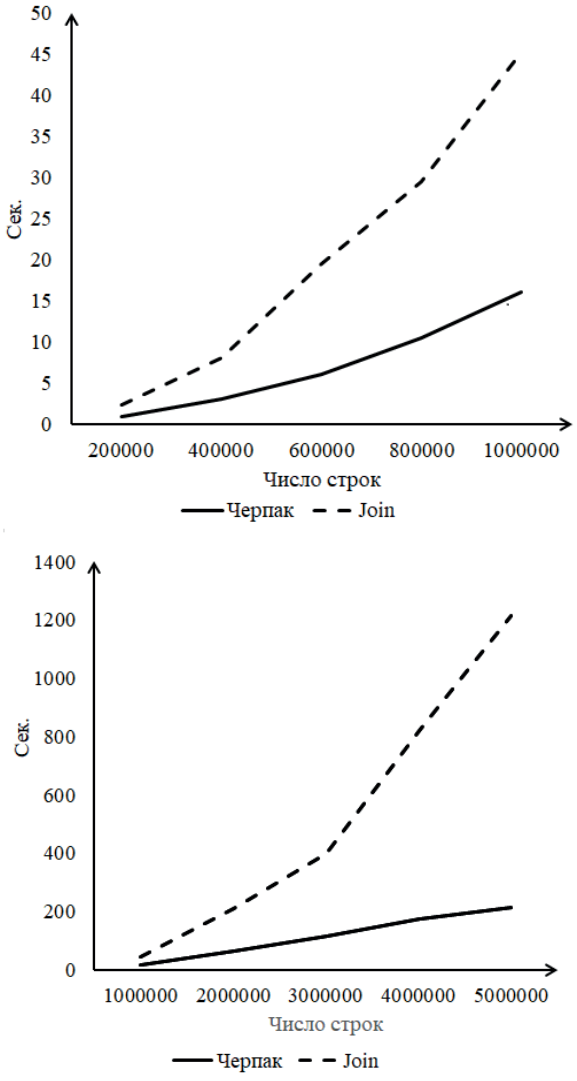
Используемые в эксперименте таблицы базы данных, соответствующие отношениям R_1 и R_2, содержали один и тот же набор из 1024 классов эквивалентности. Число строк в этих классах эквивалентности было случайным и зависело от общего числа строк в таблице. Эксперимент походил в два этапа. На первом этапе число строк в таблицах изменялось от 200000 до 1000000 с шагом 200000. На втором – от 1000000 до 5000000. Результаты эксперимента, представленные в таблице 2 и, для наглядности, на рисунке 4, отражают зависимость времени выполнения операции слияния алгоритмом черпака и стандартной операцией Join, реализованной в СУБД Microsoft SQL Server.

Эксперимент проводился для случая высокоактивных данных. Это данные, для которых коэффициент активности – отношение числа обрабатываемых данных к общему числу данных близок к 1 [15]. В рассматриваем случае он был равен 1. Полученные результаты позволяют утверждать, что предложенный алгоритм при массовой обработке высокоактивных данных имеет большую эффективность, чем используемые в СУБД алгоритмы реализации операции Join. Причем, чем больше объем данных, тем эта эффективность выше.

Т а б л и ц а 2. **Зависимость времени выполнения слияния от числа строк в таблицах-операндах**

T a b l e 2. **Merge execution time dependence on the number of rows in the operand tables**

Число строк	Черпак	Join
200000	0,83	2,36
400000	2,97	8,03
600000	6,06	19,52
800000	10,47	29,54
1000000	16,05	45,28
2000000	63,3	210,8
3000000	116,45	400,72
4000000	172,55	819,12
5000000	213,32	1214,46



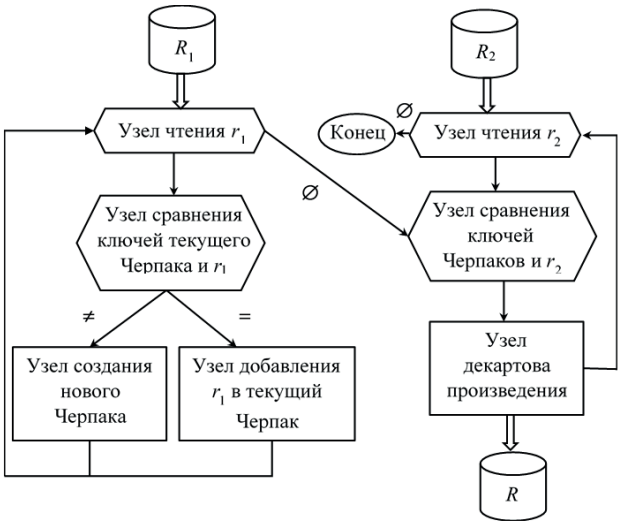
Р и с. 4. Зависимость времени выполнения слияния от числа строк в таблицах-операндах

F i g. 4. Merge execution time dependence on the number of rows in the operand tables

Параллельная реализация алгоритма черпака

Большинство алгоритмов операции Join эффективно распараллеливаются. Поскольку эта операция выполняется тем быстрее, чем меньше отношения-операнды, то один из способов распараллеливания заключается в их распределение между несколькими вычислительными узлами (процессорами). Вопросам распределенной обработки посвящено много исследований, например, [16-20]. Авторами был предложен для этой цели принцип симметричного горизонтального распределения, подробно описанный в работах [21-25]. Также возможно распараллеливание предложенного алгоритма в тех случаях, когда ведущее отношение достаточно велико и классы эквивалентности содержат большое число кортежей. Тогда, например, цикл ***for***, приведенный в листинге 1, может быть параллельно реализован в языке программирования C# с использованием метода Parallel.For. Также для этого может быть эффективно использована архитектура вычислительной системы с ассоциативным распределением ресурсов (ВСАРР), реализующая технологию управления процессом вычисления потоком данных (*data flow*) [16]. Такая вычислительная система состоит из узлов, каждый из которых реализует одну из функций, составляющих алгоритм, и активизируется в тот момент, когда ему на вход поступают все необходимые данные. Пример такой ВСАРР приведен в виде data flow диаграммы на рисунке 5. Здесь узлы чтения выполняют чтение отношений R_1 и R_2 до тех пор, пока одно из отношений не будет прочитано до конца. Узел сравнения ключа прочитанного кортежа и ключа черпака в начале работы алгоритма и, в дальнейшем, при несовпадении ключей активизирует узел создания нового черпака, передавая ему прочитанный кортеж ведущего отношения. Кроме того, полностью заполненный предыдущий черпак переводится в состояние доступности для обработки. Активизация узла создания черпака состоит в создании потока, реализующего экземпляр этого узла. Этот поток будет выполняться независимо от других потоков. Если ключи совпали, то активизируется узел, также реализованный отдельным потоком, добавляющий кортеж в доступный для добавления черпак. Узел сравнения ключей ведомого отношения и черпаков активизирует узел (поток) реализации декартова произведения передавая ему прочитанный кортеж и ссылку на соответствующий ему черпак.

Таким образом, возникает несколько параллельно реализуемых действий. Пока один заполненный черпак обрабатывается совместно с кортежами ведомого отношения другой или несколько других черпаков могут находится в режиме заполнения. Также в то время, когда выполняется декартово произведение черпака с последним кортежем соответствующего ему класса эквивалентности ведомого отношения, может быть запущено в новом потоке декартово произведение другого готового к обработке черпака с прочитанным и вызвавшим несовпадение кортежа.



Р и с. 5. Диаграмма ВСАРР для реализации алгоритма черпака

F i g. 5. Diagram of the Automatic resource allocation computing system for the implementation of the scoop algorithm

Заключение

Из сказанного в статье можно сделать следующие выводы.

1. Предложенный алгоритм достаточно прост и может быть легко реализован средствами любого языка программирования, содержащего средства взаимодействия с базами данных.
2. Это свойство простоты позволяет легко разработать генератор программ, позволяющий настраивать алгоритм для любых конкретных отношений на основе метаданных, содержащих схемы этих отношений.
3. Такие программы, независимо от ручного или автоматизированного способа их разработки могут быть добавлены в ядро СУБД, например, как это делается для Microsoft SQL Server посредством CLR-функций.
4. Алгоритм достаточно эффективен, что доказано результатами эксперимента.
5. И, наконец, он допускает достаточно простое распараллеливание.

Список использованных источников

[1] Цымблер М. Л. Обзор методов интеграции интеллектуального анализа данных в СУБД // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2019. Т. 8, №. 2. С. 32-62. https://doi.org/10.14529/cmse190203

[2] Chaudhuri S., Motwani R., Narasayya V. On random sampling over joins // Proceedings of the 1999 ACM SIGMOD international conference on Management of data (SIGMOD '99). New York, NY, USA: Association for Computing Machinery, 1999. P. 263-274. https://doi.org/10.1145/304182.304206

[3] Random Sampling over Joins Revisited / Z. Zhao [et al.] // Proceedings of the 2018 International Conference on Management of Data (SIGMOD '18). New York, NY, USA: Association for Computing Machinery, 2018. P. 1525-1539. https://doi.org/10.1145/3183713.3183739

[4] Deng S., Lu S., Tao Y. On Join Sampling and the Hardness of Combinatorial Output-Sensitive Join Algorithms // Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS '23). New York, NY, USA: Association for Computing Machinery, 2023. P. 99-111. https://doi.org/10.1145/3584372.3588666

[5] Joins on samples: a theoretical guide for practitioners / D. Huang [et al.] // Proceedings of the VLDB Endowment. 2019. Vol. 13, No. 4. P. 547-560. https://doi.org/10.14778/3372716.3372726

[6] Chen Y., Yi K. Random Sampling and Size Estimation Over Cyclic Joins // 23rd International Conference on Database Theory (ICDT 2020). Leibniz International Proceedings in Informatics (LIPIcs). 2020. Vol. 155. P. 7:1-7:18. https://doi.org/10.4230/LIPIcs.ICDT.2020.7

[7] Vengerov D., Menck A. C., Zait M., Chakkappen S. P. Join size estimation subject to filter conditions // Proceedings of the VLDB Endowment. 2015. Vol. 8, No. 12. P. 1530-1541. https://doi.org/10.14778/2824032.2824051

[8] Тимофеева Н. Е., Дмитриева К. А. Универсальный алгоритм обработки запросов с использованием технологии параллельных вычислений // Научно-технический вестник Брянского государственного университета. 2018. № 2. С. 211-217. https://doi.org/10.22281/2413-9920-2018-04-02-211-217

[9] Benchmarking Stream Join Algorithms on GPUs: A Framework and its Application to the State-of-the-art / D. P. A. Nugroho [et al.] // Proceedings of the 27th International Conference on Extending Database Technology (EDBT). 2024. P. 188-200. https://doi.org/10.48786/edbt.2024.17

[10] BS-Join: A novel and efficient mixed batch-stream join method for spatiotemporal data manage-ment in Flink / H. Ji [et al.] // Future Generation Computer Systems. 2023. Vol. 141. P. 67-80. https://doi.org/10.1016/j.future.2022.11.016

[11] Wang J., Trummer I., Kara A., Olteanu D. ADOPT: Adaptively Optimizing Attribute Orders for Worst-Case Optimal Join Algorithms via Reinforcement Learning // Proceedings of the VLDB Endowment. 2012. Vol. 16, No. 11. P. 2805-2817. https://doi.org/10.14778/3611479.3611489

[12] Мунерман В. И., Мунерман Д. В. Параллельная реализация операций обработки файлов // Современные информационные технологии и ИТ-образование. 2016. Т. 12, № 2. С. 84-90. EDN: XSATGX

[13] Мунерман В. И. Реализация обработки больших объемов данных на симметричных мультипроцессорных системах // Системы высокой доступности. 2013. Т. 9, № 2. С. 036-039. EDN: QOVNNR

[14] Левин Н. А., Мунерман В. И. Метод логически последовательного доступа к данным // Системы высокой доступности. 2011. Т. 7, № 4. С. 65-67. EDN: PEZEGP

[15] Li M., Huang L., Xu G., Biao K. A parallel particle swarm optimization framework based on a fork-join thread pool using a work-stealing mechanism // Applied Soft Computing. 2023. Vol. 145, issue C. Article number: 110537. https://doi.org/10.1016/j.asoc.2023.110537

[16] Hu X., Yi K. Massively Parallel Join Algorithms // ACM SIGMOD Record. 2020. Vol. 49, No. 3. P. 6-17. https://doi.org/10.1145/3444831.3444833

[17] Efficient Massively Parallel Join Optimization for Large Queries / R. Mancini [et al.] // Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22). New York, NY, USA: Association for Computing Machinery, 2022. P. 122-135. https://doi.org/10.1145/3514221.3517871

[18] Massively Parallel NUMA-Aware Hash Joins / H. Lang [et al.] // Memory Data Management and Analysis. IMDM IMDM 2013 2014. Lecture Notes in Computer Science ; ed. by A. Jagatheesan, J. Levandoski, T. Neumann, A. Pavlo. Vol. 8921. Cham: Springer, 2015. P. 3-14. https://doi.org/10.1007/978-3-319-13960-9_1

[19] Distributed join algorithms on thousands of cores / C. Barthels [et al.] // Proceedings of the VLDB Endowment. 2017. Vol. 10, No. 5. P. 517-528. https://doi.org/10.14778/3055540.3055545

[20] Мунерман В. И. Построение архитектур программно-аппаратных комплексов для повышения эффективности массовой обработки данных // Системы высокой доступности. 2014. Т. 10, № 4. С. 3-16. EDN: TFQJWX

[21] Мунерман В. И., Мунерман Д. В. Один метод реализации симметричного горизонтального распределения данных // Системы компьютерной математики и их приложения. 2017. № 18. С. 97-100. EDN: ZQTUZT

[22] Мунерман В. И., Мунерман Д. В. Параллельная реализация симметричного горизонтального распределения данных на основе сетевых технологий // Современные информационные технологии и ИТ-образование. 2017. Т. 13, № 3. С. 38-43. https://doi.org/10.25559/SITITO.2017.3.521

[23] Мунерман В. И. Реализация параллельной обработки данных в облачных системах // Современные информационные технологии и ИТ-образование. 2017. Т. 13, № 2. С. 57-63. https://doi.org/10.25559/SITITO.2017.2.223

[24] Munerman V., Munerman D., Samoilova T. The Heuristic Algorithm For Symmetric Horizontal Data Distribution // 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus). St. Petersburg, Moscow, Russia: IEEE Press, 2021. P. 2161-2165. https://doi.org/10.1109/ElConRus51938.2021.9396510

[25] Провоторова А. О. Моделирование параллельной системы баз данных на вычислительной системе «ВСАРР» // Системы и средства информатики. 2006. Т. 16, № 1. С. 418-430. EDN: KZUWMZ

Поступила 21.05.2024; одобрена после рецензирования 28.07.2024; принята к публикации 19.09.2024.

Об авторах:

Мунерман Виктор Иосифович, доцент кафедры прикладной математики и информатики физико-математического факультета, ФГБОУ ВО «Смоленский государственный университет» (214000, Российская Федерация, г. Смоленск, ул. Пржевальского, д. 4), кандидат технических наук, доцент, **ORCID: https://orcid.org/0000-0002-9628-4049,** vimoon@gmail.com

Мунерман Даниил Викторович, лаборант-стажер кафедры прикладной математики и информатики физико-математического факультета, ФГБОУ ВО «Смоленский государственный университет» (214000, Российская Федерация, г. Смоленск, ул. Пржевальского, д. 4), **ORCID: https://orcid.org/0000-0002-5139-6645,** danvmoon@gmail.com

Все авторы прочитали и одобрили окончательный вариант рукописи.

References

[1] Zymbler M.L. Overview of Methods for Integrating Data Mining into DBMS. *Bulletin of South Ural State University: Computational Mathematics and Software Engineering*. 2019;8(2):32-62. (In Russ., abstract in Eng.) https://doi.org/10.14529/cmse190203

[2] Chaudhuri S., Motwani R., Narasayya V. On random sampling over joins. In: Proceedings of the 1999 ACM SIGMOD international conference on Management of data (SIGMOD '99). New York, NY, USA: Association for Computing Machinery; 1999. p. 263-274. https://doi.org/10.1145/304182.304206

[3] Zhao Z., Christensen R., Li F., Hu X., Yi K. Random Sampling over Joins Revisited. In: Proceedings of the 2018 International Conference on Management of Data (SIGMOD '18). New York, NY, USA: Association for Computing Machinery; 2018. p. 1525-1539. https://doi.org/10.1145/3183713.3183739

[4] Deng S., Lu S., Tao Y. On Join Sampling and the Hardness of Combinatorial Output-Sensitive Join Algorithms. In: Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS '23). New York, NY, USA: Association for Computing Machinery; 2023. p. 99-111. https://doi.org/10.1145/3584372.3588666

[5] Huang D., Yoon D.Y., Pettie S., Mozafari B. Joins on samples: a theoretical guide for practitioners. *Proceedings of the VLDB Endowment*. 2019;13(4):547-560. https://doi.org/10.14778/3372716.3372726

[6] Chen Y., Yi K. Random Sampling and Size Estimation Over Cyclic Joins. In: 23rd International Conference on Database Theory (ICDT 2020). Leibniz International Proceedings in Informatics (LIPIcs). 2020;155:7:1-7:18. https://doi.org/10.4230/LIPIcs.ICDT.2020.7

[7] Vengerov D., Menck A.C., Zait M., Chakkappen S.P. Join size estimation subject to filter conditions. *Proceedings of the VLDB Endowment.* 2015;8(12):1530-1541. https://doi.org/10.14778/2824032.2824051

[8] Timofeeva N.E., Dmitrieva K.A. Universal algorithm of processing of requests with use of parallel technology. *Nauchno-tekhnicheskiy vestnik Bryanskogo gosudarstvennogo universiteta* = Scientific and Technical Journal of Bryansk State University. 2018;(2):211-217. (In Russ., abstract in Eng.) https://doi.org/10.22281/2413-9920-2018-04-02-211-217

[9] Nugroho D.P.A., et al. Benchmarking Stream Join Algorithms on GPUs: A Framework and its Application to the State-of-the-art. In: Proceedings of the 27th International Conference on Extending Database Technology (EDBT). 2024. p. 188-200. https://doi.org/10.48786/edbt.2024.17

[10] Ji H., et al. BS-Join: A novel and efficient mixed batch-stream join method for spatiotemporal data manage-ment in Flink. *Future Generation Computer Systems*. 2023;141:67-80. https://doi.org/10.1016/j.future.2022.11.016

[11] Wang J., Trummer I., Kara A., Olteanu D. ADOPT: Adaptively Optimizing Attribute Orders for Worst-Case Optimal Join Algorithms via Reinforcement Learning. *Proceedings of the VLDB Endowment*. 2012;16(11):2805-2817. https://doi.org/10.14778/3611479.3611489

[12] Munerman V.I., Munerman D.V. The parallel implementation of the files processing operations. *Modern Information Technologies and IT-Education*. 2016;12(2):84-90. (In Russ., abstract in Eng.) EDN: XSATGX

[13] Munerman V.I. Implementation of big data processing on symmetric multiprocessing systems. *Highly Available Systems.* 2013;9(2):036-039. (In Russ., abstract in Eng.) EDN: QOVNNR

[14] Levin N.A., Munerman V.I. Logically sequential data access method. *Highly Available Systems.* 2011;7(4):65-67. (In Russ., abstract in Eng.) EDN: PEZEGP

[15] Li M., Huang L., Xu G., Biao K. A parallel particle swarm optimization framework based on a fork-join thread pool using a work-stealing mechanism. *Applied Soft Computing*. 2023;145(C):110537. https://doi.org/10.1016/j.asoc.2023.110537

[16] Hu X., Yi K. Massively Parallel Join Algorithms. *ACM SIGMOD Record*. 2020;49(3):6-17. https://doi.org/10.1145/3444831.3444833

[17] Mancini R., Karthik S., Chandra B., Mageirakos V., Ailamaki A. Efficient Massively Parallel Join Optimization for Large Queries. In: Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22). New York, NY, USA: Association for Computing Machinery; 2022. p. 122-135. https://doi.org/10.1145/3514221.3517871

[18] Lang H., Leis V., Albutiu M.C., Neumann T., Kemper A. Massively Parallel NUMA-Aware Hash Joins. In: Jagatheesan A., Levandoski J., Neumann T., Pavlo A. (eds.) In: Memory Data Management and Analysis. IMDM IMDM 2013 2014. *Lecture Notes in Computer Science*. Vol. 8921. Cham: Springer; 2015. p. 3-14. https://doi.org/10.1007/978-3-319-13960-9_1

[19] Barthels C., Müller I., Schneider T., Alonso G., Hoefler T. Distributed join algorithms on thousands of cores. *Proceedings of the VLDB Endowment*. 2017;10(5):517-528. https://doi.org/10.14778/3055540.3055545

[20] Munerman V.I. Construction of hardware-software complexes architecture to improve massively data processing. *Highly Available Systems.* 2014;10(4):3-16. (In Russ., abstract in Eng.) EDN: TFQJWX

[21] Munerman V.I., Munerman D.V. *Odin metod realizacii simmetrichnogo gorizontal'nogo raspredelenija dannyh* [One method for implementing symmetric horizontal data distribution]. *Sistemy komp'yuternoj matem-atiki i ih prilozheniya* = Computer Mathematics Systems and Their Applications. 2017;(18):97-100. (In Russ., abstract in Eng.) EDN: ZQTUZT

[22] Munerman V.I., Munerman D.V. Parallel implementation of symmetric horizontal distributionof data on the network technologies basis. *Modern Information Technologies and IT-Education*. 2017;13(3):38-43. (In Russ., abstract in Eng.) https://doi.org/10.25559/SITITO.2017.3.521

[23] Munerman V.I. The implementation of parallel data processing in cloud systems. *Modern Information Technologies and IT-Education*. 2017;13(2):57-63. (In Russ., abstract in Eng.) https://doi.org/10.25559/SITITO.2017.2.223

[24] Munerman V., Munerman D., Samoilova T. The Heuristic Algorithm For Symmetric Horizontal Data Distribution. In: 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus). St. Petersburg, Moscow, Russia: IEEE Press; 2021. p. 2161-2165. https://doi.org/10.1109/ElConRus51938.2021.9396510

[25] Provotorova A. Parallel data base system simulating on the VSARR computing system. *Systems and Means of Informatics.* 2006;16(1):418-430. (In Russ., abstract in Eng.) EDN: KZUWMZ

Submitted 21.05.2024; approved after reviewing 28.07.2024; accepted for publication 19.09.2024.

About the authors:

Victor I. Munerman, Associate Professor of the Chair of Applied Mathematics and Informatics, Faculty of Physics and Mathematics, Smolensk State University (4 Przhevalsky St., Smolensk 214000, Russian Federation), Cand. Sci. (Eng.), Associate Professor, **ORCID: https://orcid.org/0000-0002-9628-4049,** vimoon@gmail.com

Daniel V. Munerman, Laboratory Assistant of the Chair of Applied Mathematics and Informatics, Faculty of Physics and Mathematics, Smolensk State University (4 Przhevalsky St., Smolensk 214000, Russian Federation), **ORCID: https://orcid.org/0000-0002-5139-6645,** danvmoon@gmail.com

All authors have read and approved the final manuscript.