УДК 004.657

Захаров В.Н.[1], Мунерман В.И.[2], Самойлова Т.А.[2]

[1] Федеральный исследовательский центр «Информатика и управление» РАН, г. Москва, Россия
[2] Смоленский государственный университет, г. Смоленск, Россия

ПАРАЛЛЕЛЬНЫЕ МЕТОДЫ ВЫВОДА АССОЦИАТИВНЫХ ПРАВИЛ В ТЕХНОЛОГИЯХ IN-DATABASE И IN MEMORY*

Аннотация

В статье предложен подход к разработке параллельного программного обеспечения для решения задач data mining. Рассмотрена задача подготовки данных для вывода ассоциативных правил. Большинство современных алгоритмов для этой задачи имеют экспоненциальную сложность. Дополнительную сложность создает необходимость обработки больших данных. Сам вывод ассоциативных правил на основе полученных данных реализуется полиномиальным алгоритмом на малом объеме полученных данных. Доказано, что использование теоретико-множественной (файловой) и многомерно-матричной моделей данных позволяет разработать для ее решения полиномиальные алгоритмы. Предложены методы реализации этих алгоритмов в технологиях in-database и in-memory. В технологии in-database приведены схема базы данных и совокупность запросов к базе данных, с помощью которых формируются данные, необходимые для этапа вывода ассоциативных правил. Предложен метод ускорения обработки за счет использования принципа симметричного горизонтального распределения таблиц и параллельного полиномиального алгоритма реализации операции Join. В технологии in-memory предложен метод решения задачи подготовки данных на основе алгебры многомерных матриц. Показано, что эта задача может быть решена последовательностью (0, 1)- и (1, 0)-свернутых произведений многомерных матриц. Эти операции могут быть распараллелены аналогично распараллеливанию операции умножения обычных матриц. Приведены результаты вычислительного эксперимента, в котором предложенные методы сравнивались с алгоритмом Apriory. Результаты эксперимента подтвердили эффективность предложенных методов.

Ключевые слова

Ассоциативные правила, параллельное программирование, big data.

Zakharov V.N.[1], Munerman V.I.[2], Samoilova T.A.[2]

[1] Federal Research Center Computer Science and Control of the Russian Academy of Sciences, Moscow, Russia
[2] Smolensk State University, Smolensk, Russia

PARALLEL METHODS FOR DERIVING ASSOCIATIVE RULES WITH THE USAGE IN-DATABASE AND IN-MEMORY TECHNOLOGIES

Abstract

The approach to the development of parallel software for solving data mining problems is proposed in the article. The problem of data preparation for the derivation of associative rules is considered. Most modern algorithms for this problem have exponential complexity. An additional complication is the need to process large data. The derivation of associative rules on the basis of the obtained data is realized by a polynomial algorithm on a small volume of the obtained data. It is proved that the use of the set-theoretic (file) and multidimensional-matrix data models makes it possible to develop polynomial algorithms for solving it. Methods for implementing these algorithms in in-database and in-memory technologies are proposed. In the in-database technology, the database schema and the set of database queries are listed. With the help of these queries, the data necessary for the stage of the associative rules derivation are formed. The method of accelerating processing due to the use of

* **Труды II Международной научной конференции «Конвергентные когнитивно-информационные технологии» (Convergent'2017), Москва, 24-26 ноября, 2017**

Proceedings of the II International scientific conference "Convergent cognitive information technologies" (Convergent'2017), Moscow, Russia, November 24-26, 2017

the symmetric horizontal distribution of tables and the parallel polynomial algorithm for implementing the Join operation is proposed. In-memory technology offers a method for solving the problem of data preparation, which is based on the algebra of multidimensional matrices. It is shown that this problem can be solved by the sequence of (0, 1)- and (1, 0)-converted products of multidimensional matrices. These operations can be parallelized in the same way as the parallelized of the ordinary matrices multiplication. The results of a computational experiment in which the proposed methods were compared with the algorithm Apriory are presented. The results of the experiment confirmed the effectiveness of the proposed methods.

Keywords

Associative rules, parallel programming, big data.

В статье рассматриваются два метода повышения эффективности обработки данных при решении задач вывода ассоциативных правил. В отличие от большинства работ в этой области, в которых предлагаются методы улучшения запросов конечных пользователей, занимающихся анализом данных, здесь речь идет о методах, ориентированных на программиста разработчика аналитических информационных систем [1].

Ассоциативные правила в настоящее время превратились в мощный инструмент аналитических информационных систем. Поскольку в основе любых информационных систем лежат базы данных, весьма актуально направление, связанное с реализацией алгоритмов интеллектуального анализа на языках манипулирования данными. Это направление получило название "аналитика в базе данных" ("in-database analytics"). Оно предполагает разработку технологий, позволяющих осуществлять обработку данных в базе данных путем построения аналитической логики в самой базе данных. В большинстве современных СУБД эти языки, подобные Transact SQL или PL/SQL, прибрели основные черты процедурно-ориентированных языков программирования. Запросы к БД, записываются на этих языках как программы, содержащие переменные разных типов и операторы управления. Программы запросов оформляются в виде хранимых процедур, которые транслируются и поэтому выполняются значительно быстрее, чем интерпретируемые запросы. Кроме того, большинство современных СУБД оптимизируют и распараллеливают SQL-запросы. Поэтому реализации вывода ассоциативных правил средствами СУБД посвящено достаточно много исследований, например, [2-7]. В большинстве работ в этой области в основу вычислений положена операция JOIN, причем делается это запросами вида

SELECT <список полей> FROM T_1, …, T_k.

Такой подход хорош для профессионального аналитика, имеющего навыки прикладного программирования. Однако для разработчика аналитических информационных систем он неприемлем. Это объясняется тем, что эффективность запросов, в которых операция JOIN задается неявно, полностью зависит от возможностей системы управления базами данных. Вместе с тем, программист-разработчик, владеющий методами параллельной и распределенной обработки данных, может существенно повысить производительность разрабатываемой системы за счет применения этих методов.

С другой стороны, большинство известных алгоритмов вывода ассоциативных правил используют технологии вычислений в оперативной памяти (in-memory computing). Эти технологии ускоряют обработку больших объемов данных (Big Data).

Оба подхода к разработке алгоритмов вывода ассоциативных правил при проектировании аналитических информационных систем требуют применения эффективных моделей данных, которые обеспечат программиста-разработчика необходимыми методами распределения данных и распараллеливания их обработки.

В общем виде задача вывода ассоциативных правил выглядит следующим образом. Имеется множество объектов $I=\{i_1, \ldots, i_n\}$ и множество свойств $P=\{p_1, \ldots, p_k\}$. Каждому объекту из I соответствует некоторое непустое подмножество свойств из P. Вывод ассоциативных правил состоит из двух этапов. На первом этапе для каждого подмножества свойств определяется количество объектов, которые обладают всеми свойствами этого подмножества, и только этими свойствами. На втором этапе, на основе полученных статистических данных и требований пользователя-аналитика, выводятся сами ассоциативные правила.

Далее рассматривается только первый этап, который можно определить как этап подготовки данных. Этот этап имеет высокую, как правило, экспоненциальную, вычислительную сложность. Повышение эффективности алгоритмов вывода ассоциативных правил достигается за счет распараллеливания известных алгоритмов, например, алгоритма Apriory [8]. Другой способ повышения эффективности алгоритмов состоит в использовании параллелизмов, реализованных в СУБД [9]. Эти способы хороши для оптимизации запросов конечных пользователей, но они могут не учитывать всех возможностей вычислительных систем, на которых эти пользователи решают свои задачи. Для того, чтобы в конкретных

случаях и на конкретных предприятиях эти задачи решались эффективно, разрабатываются индивидуальные аналитические информационные системы. В этом случае программист-разработчик имеет возможность добиваться высокой эффективности за счет построения программно-аппаратного комплекса, ориентированного на конкретный класс задач. Далее рассматриваются методы достижения этой цели, основанные на теоретико-множественной (файловой) и многомерно-матричной моделях данных. Обе эти модели изоморфны реляционной модели данных.

Известны различные варианты реализации алгоритма Apriori средствами языка SQL [9, 10]. В дальнейшем рассматривается реализация, основанная на построении на каждой итерации промежуточной таблицы, которая используется на следующей итерации.

База данных для решения задачи вывода ассоциативных правил, как правило содержит следующий набор таблиц:

$R_0(I, P_1)$ – исходные данные, каждая строка содержит идентификатор объекта и одно из его свойств;

$CopyR_0(I, P_1)$ – копия таблицы R_0, она не обязательна и может использоваться для ускорения процесса обработки данных;

$R_1(I, P_1, P_2)$, …, $R_{k-1}(I, P_1, …, P_k)$ – таблицы, получаемые на итерациях и содержащие данные для вывода ассоциативных правил на очередной итерации и используемые на следующей итерации как исходные данные.

Запрос, исполняемый на l-той итерации имеет, следующий общий вид:

Q_l = INSERT INTO R_l (I, P_1, …, P_{l+1})

SELECT $R_{l-1}.I$, $R_{l-1}.P_1$, …, $R_{l-1}.P_l$, $R_0.P_1$ AS P_{l+1}

FROM R_{l-1} INNER JOIN R_0 [на первой итерации вместо R_{l-1} используется $CopyR_0$]

ON $R_{l-1}.I = R_0.I$ AND $\pi(R_{l-1}.I, R_{l-1}.P_1, …, R_{l-1}.P_l, R_0.P)$.

$\pi(R_{i\,l-1}.I, R_{l-1}.P_1, …, R_{l-1}.P_l, R_0.P)$ – предикат, запрещающий дублирование комбинаций свойств.

Таким образом, если для построения ассоциативных правил необходимо знать количества всех объектов, содержащих комбинации свойств от одного до $k \le s$, выполняются запросы Q_1, …, Q_{k-1}. Из таблиц R_0, …, R_{k-1}, исходной и полученных в результате запросов, на втором этапе выводятся ассоциативные правила.

Ускорение подготовки данных возможно за счет того, что современные СУБД имеют возможность параллельного выполнения запросов, в том числе, и запросов, содержащих операцию JOIN. Файловая модель данных обеспечивает возможность применения симметричного горизонтального распределения [11] таблицы R_0 по ключу I между несколькими базами данных, расположенными на физически разных серверах. Это усилит эффект ускорения за счет существенного, практически кратного числу серверов, уменьшения объемов фрагментов таблиц $R_0, …, R_{k-1}$. Следует отметить, что для решаемой задачи целесообразно использовать однопроходный алгоритм операции JOIN, который основан на вычислении декартовых произведений классов эквивалентности фрагментов таблиц R_0 и R_{l-1}, которые обрабатываются на l-той итерации. Каждый такой класс эквивалентности содержит строки таблиц с одинаковыми значениями идентификатора объекта. Для реализации алгоритма требуется упорядоченность таблиц R_0 и R_{l-1} по ключу идентификатор объекта. Это незначительно повлияет на его производительность, так как таблица R_0 упорядочивается один раз при ее создании, а таблица R_{l-1} в этом случае создается упорядоченной. Для выполнения алгоритма в оперативной памяти выделяются две буферные области, в которые будут считываться из базы данных классы эквивалентности фрагментов таблиц R_0 и R_{l-1}. Размер буферных областей определяется количеством доступных процессоров (ядер) и объемом доступной оперативной памяти. Считывание классов эквивалентности осуществляется двумя параллельными потоками, которые взаимодействуют друг с другом для того, чтобы заполнить буферные области соответствующими друг другу классами эквивалентности. После заполнения буферных областей запускаются параллельные потоки, вычисляющие декартовы произведения (ДП-потоки) соответствующих пар классов эквивалентности, свертку и вывод результатов в базу данных в фрагмент таблицы R_l, которая будет использоваться на следующей итерации. На практике, количество потоков может оказаться значительно меньше пар классов эквивалентности, обрабатываемых этими потоками. В этом случае распределение пар классов эквивалентности между потоками можно сделать, используя алгоритм бустрофедона [12]. Программно-аппаратный комплекс для реализации этого алгоритма приведен на рисунке 1. Здесь R_0^i, R_l^i – фрагменты таблиц R_0 и R_{l-1} $R_{0_j}^i, R_{l_j}^i, (j = 1, ..., s)$ – совокупности соответствующих классов эквивалентностей в буферных областях.

Вычислительная сложность обработки буферных областей определяется временем считывания классов эквивалентности из базы данных и сложностью вычисления декартовых произведений классов эквивалентности. В общем случае, когда число пар классов эквивалентности больше числа обрабатывающих их процессоров (ядер), сложность определяется значением $\underset{j=1}{\overset{s}{Max}} \sum_{k=1}^{n_s} (\mu R_{0_{jk}}^i \times \mu R_{l_{jk}}^i)$.

$\mu R^i_{0_{jk}}, \mu R^i_{l_{jk}}$ – количество строк в *k*-том классе эквивалентности, обрабатываемым *j*-тым потоком, n_s – количество классов эквивалентности, обрабатываемых *j*-тым потоком.

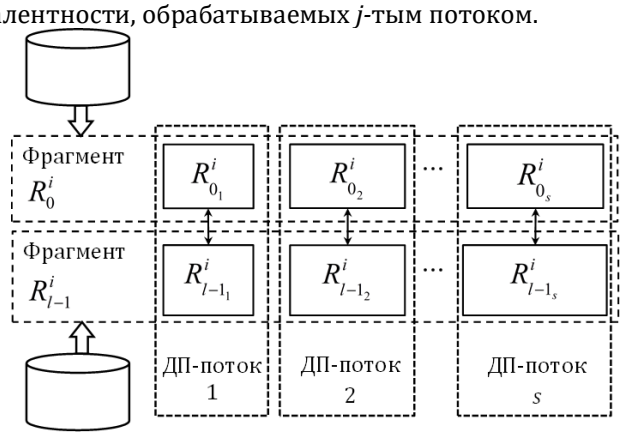

Рис. 1. Программно-аппаратный комплекс для вывода ассоциативных правил

Кроме того, дополнительного эффекта можно достигнуть за счет применения конвейерного метода для цепочки операций JOIN, которые реализуют вычисления на итерациях [13]. Это потребует дополнительных усилий программиста-разработчика, но обеспечит существенное ускорение этапа подготовки данных, особенно в сочетании с симметричным горизонтальным распределением таблицы R_0.

Применение многомерно-матричной модели позволяет существенно улучшить временные характеристики этапа подготовки данных. Если номера объектов и свойств использовать в качестве индексов, то исходные данные в этой модели – матрица $M_0 = \left(m^0_{ij}\right), i = 1,..., n, j = 1,..., s.$ При условии, что $m^0_{ij} = 1,$ если объект I_i обладает свойством P_j и 0 в противном случае, матрица M_0 взаимно однозначно соответствует таблице R_0. Операции JOIN в алгебре многомерных матриц соответствует операция (λ, μ)-свернутого произведения, которая позволяет умножать матрицы с произвольным числом измерений, получая многомерные матрицы. В рассматриваемом случае индекс *i*, соответствующий идентификатору объекта, выполняет две функции:

- при вычислении матрицы для следующей итерации он используется как скоттов индекс для совмещения умножаемых элементов матриц, то есть реализуется (1, 0)-свернутое произведение;
- при вычислении матрицы для вывода ассоциативных правил он используется как кэлиев индекс для совмещения умножаемых элементов матриц и суммирования произведений, то есть реализуется (0, 1)-свернутое произведение.

Алгоритм, основанный на умножении многомерных матриц, так же как и алгоритм, основанный на операции JOIN, состоит из последовательности итераций.

На первой итерации вычисляется трехмерная матрица $M_1 = ^{1,0} (M_0 \times M_0) = \left(m^1_{ijj}\right), i = 1,..., n, j = 1,..., s,$ где $m^1_{ijj} = m^0_{ij} \times m^0_{ij}$ и двумерная матрица $T_1 = ^{0,1} (M_0 \times M_0) = \left(t^1_{jj}\right), j = 1,..., s,$ где $t^1_{jj} = \sum_{i=1}^{n} m^0_{ij} \times m^0_{ij}.$ Матрица T_1 также может быть получена из матрицы M_1 операцией свертки, однако, обе матрицы можно вычислять одновременно.

На *l*-той итерации вычисляются *l*+1-мерная матрица $M_l = ^{1,0} (M_{l-1} \times M_0) = \left(m^l_{i\underbrace{j...j}_{l}} \right), m^l_{i\underbrace{j...j}_{l}} = m^{l-1}_{i\underbrace{j...j}_{l-1}} \times m^0_{ij},$ которая будет использована на следующей итерации, и матрица $T_l = ^{0,1} (M_{l-1} \times M_0) = \left(t^l_{\underbrace{j...j}_{l}} \right), t^l_{\underbrace{j...j}_{l}} = \sum_{i=1}^{n} m^{l-1}_{i\underbrace{j...j}_{l-1}} \times m^0_{ij},$ которая будет использована для вывода ассоциативных правил. Также, как и при использовании реляционной модели, для получения всех возможных комбинаций свойств от 1 до *k* необходимо выполнить *k* итераций. На последней итерации нет необходимости в построении матрицы M_{k+1}.

При последовательной реализации алгоритма методом умножения многомерных матриц вычислительная сложность определяется значением: $k \times (n + n^2 + ... + n^k)$, *n* – количество объектов, *k* – количество свойств.

Таким образом, можно утверждать, что оба рассмотренных алгоритма имеют полиномиальную сложность.

Для ускорения решения задачи вывода ассоциативных правил можно использовать параллельный алгоритм умножения многомерных матриц [14]. Кроме того, симметричное горизонтальное распределение в данном случае применимо для реализации рассматриваемой задачи в многомерно-матричной модели данных.

Для оценки рассмотренных в статье методов был проведен вычислительный эксперимент. Анализу были подвергнуты три алгоритма вывода ассоциативных правил:

- алгоритм Apriory, взятый из пакета "Apriori 1.1.1", который находится в центральном репозитарии модулей языка Python – Python Package Index [15];
- алгоритм, реализованный средствами СУБД Microsoft Sql Server 2016 (на основе операции JOIN);
- алгоритм на основе умножения многомерных матриц.

Программное обеспечение разрабатывалось в Microsoft Visual Studio 2017. Для алгоритма 1 исходные данные преобразовались из текстового файла в списковую структуру в оперативной памяти, необходимую для работы алгоритма. Для алгоритма 2 исходные данные располагались в таблице R_0 а подготовка к работе заключалась в удалении всех строк из промежуточных и результирующих таблиц. Для алгоритма 3 исходные данные преобразовывались из таблицы R_0 в матрицу M_0 в оперативной памяти. Программа подготовки данных и вызова алгоритма 1 написана на языке Pyton. Алгоритм 2 реализован хранимыми процедурами, написанными на языке Transact Sql и вызываемыми из программы, написанной на языке C#. Алгоритм 3 реализован как процедура динамической библиотеки на языке C++, подготовка данных для которой и ее вызов осуществляются программой на язык C#.

Эксперимент производился на исходном наборе данных, содержащем миллион объектов, каждому из которых соответствовал набор от одного до пяти свойств. Таблица R_0 содержала последовательно от одного до пяти миллионов строк. Алгоритмы 1 и 3 были реализованы в технологии in-memory, то есть использовали исходные данные, загруженные в оперативную память, промежуточные данные и результаты вычислений сохранялись там же. Алгоритм 2 был реализован в технологии in-database, причем использовался стандартный алгоритм операции JOIN, реализованный в СУБД Microsoft Sql Server 2016.

Для алгоритмов 2 и 3 было выполнено симметричное горизонтальное распределение данных между четырьмя серверами для алгоритма 2 и таким же количеством потоков для алгоритма 3. С учетом времени распределения и сбора результатов были получены результаты, приведенные в таблице 1.

Таблица 1. Временные характеристики алгоритмов подготовки данных

Таблица R_0 (миллионы строк)	Алгоритм Apriory	Алгоритм на основе JOIN		Многомерно-матричный алгоритм	
		последовательно	параллельно	последовательно	параллельно
1	29,87	23,61	7,44	5,05	1,09
2	56,12	40,69	12,82	11,27	2,43
3	84,79	54,96	17,31	18,35	3,96
4	130,54	74,95	23,61	23,42	5,05
5	209,56	105,19	33,14	33,49	7,22

На рисунке 2 приведены графики, отражающие соотношения времени подготовки данных для вывода ассоциативных правил алгоритмом Apriory, реализованном в технологии in-memory, и алгоритмом, реализованном в технологии in-database, последовательно и параллельно.

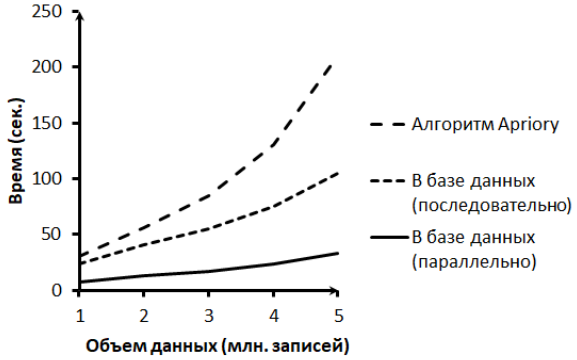

Рис. 2. Сравнение алгоритма Apriory и алгоритма на основе операции JOIN

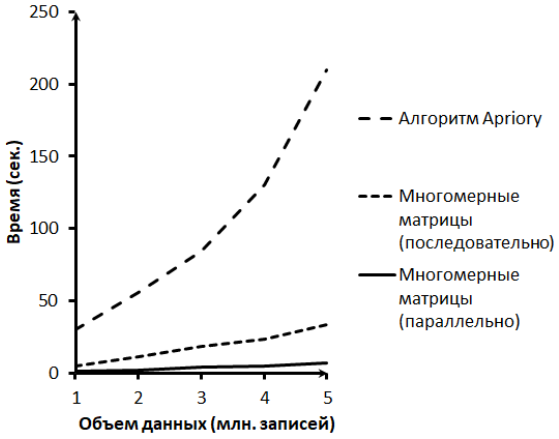

Рис. 3. Сравнение алгоритма Apriory и алгоритма на основе умножения многомерных матриц

На рисунке 3 приведены графики, отражающие соотношения времени подготовки данных для вывода ассоциативных правил алгоритмом Apriory, реализованном в технологии in-memory, и алгоритмом основанном на умножении многомерных матриц, также реализованном в технологии in-memory, последовательно и параллельно.

Из сказанного можно сделать следующие выводы:

В том случае, когда объемы данных настолько велики, что их невозможно разместить в оперативной памяти целесообразно использовать алгоритмы подобные алгоритму 2. Время решения задачи вывода ассоциативных правил таким способом сопоставимо с временем широко используемых в настоящее время алгоритмов типа алгоритма Apriory, а при использовании симметричного горизонтального распределения данных существенно меньше.

При возможности размещения данных в оперативной памяти алгоритмы, основанные на умножении многомерных матриц, значительно более эффективны и гораздо проще распараллеливаются, чем известные алгоритмы вывода ассоциативных правил.

Предложенные в статье методы могут быть эффективно использованы при разработке программного обеспечения аналитических информационных систем.

Литература

1. Самойлова Т.А. Разработка аналитических информационных систем средствами современных баз данных. Системы компьютерной математики и их приложения: материалы XVIII Международной научной конференции, – Смоленск: Изд-во СмолГУ, 2017, Вып. 18. – с. 111-114.
2. Houtsma M., Swami A. Set-oriented mining of association rules. Research Report RJ 9567, IBM Almaden Research Center, San Jose, California, October 1993.
3. Salim M, Yao X. Evolving SQL Queries for Data Mining. – Lecture Notes in Computer Science, 2002, 2412. – p. 62–67.
4. R. Agrawal, T. Imielinski, A. Swami. 1993. Mining Associations between Sets of Items in Massive Databases. In Proc. of the 1993 ACM-SIGMOD Int'l Conf. on Management of Data, 207-216.
5. Agrawal R., Srikant R. "Fast Discovery of Association Rules". – Proc. of the 20th International Conference on VLDB, Santiago, Chile, September 1994.
6. Srikant R., Agrawal R. "Mining Generalized Association Rules". – Proc. of the 21th International Conference on VLDB, Zurich, Switzerland, 1995.
7. R. Srikant, R. Agrawal. "Mining quantitative association rules in large relational tables". In Proceedings of the ACM SIGMOD Conference on Management of Data, Montreal, Canada, June 1996.
8. Pol U. Design and Development of Apriori Algorithm for Sequential to concurrent mining using MPI / U. Pol // International journal of Computers & Technology. – 2013. – Т. 10. – № 7. – С. 1785–1790.
9. Речкалов Т.В. Подход к интеграции интеллектуального анализа данных в реляционную СУБД на основе генерации текстов хранимых процедур. – Вестник Южно-Уральского государственного университета. Серия «Вычислительная математика и информатика», том 2, №1, издательство: Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск), 2013. – с. 114-121.
10. Sidló, C.I., Lukács, A. Shaping SQL-based frequent pattern mining algorithms (Revised Selected and Invited Papers). – Knowledge Discovery in Inductive Databases: 4th International Workshop, KDID 2005, Springer, Heidelberg. – p. 188–201.
11. Мунерман В.И. Модели обработки больших объемов данных в системах массового параллелизма . – М.: Системы высокой доступности, 2013, Т. 9, № 1. – С. 35-43.
12. Макаров Д.И., Мунерман В.И. Параллельная реализации операции соединения для массовой обработки данных Системы высокой доступности. 2012. Т. 8. № 3. С. 26-28.
13. Мунерман В.И., Мунерман Д.В. Алгебраический подход к построению программно-аппаратных комплексов для повышения эффективности массовой обработки данных. Современные информационные технологии и ИТ-образование. 2015. Т. 2. № 11. – С. 391-396.
14. Захаров В.Н., Мунерман В.И. Параллельный алгоритм умножения многомерных матриц Современные информационные технологии и ИТ-образование. 2015. Т. 2. № 11. С. 384-39.
15. https://pypi.python.org/pypi.

References

1. Samojlova T.A. Razrabotka analiticheskih informacionnyh sistem sredstvami sovremennyh baz dannyh. Sistemy komp'juternoj matematiki i ih prilozhenija: materialy XVIII Mezhdunarodnoj nauchnoj konferencii, – Smolensk: Izd-vo SmolGU, 2017, Vyp. 18. – s. 111-114.
2. Houtsma M., Swami A. Set-oriented mining of association rules. Research Report RJ 9567, IBM Almaden Research Center, San Jose, California, October 1993.
3. Salim M, Yao X. Evolving SQL Queries for Data Mining. – Lecture Notes in Computer Science, 2002, 2412. – p. 62–67.
4. R. Agrawal, T. Imielinski, A. Swami. 1993. Mining Associations between Sets of Items in Massive Databases. In Proc. of the 1993 ACM-SIGMOD Int'l Conf. on Management of Data, 207-216.
5. Agrawal R., Srikant R. "Fast Discovery of Association Rules". – Proc. of the 20th International Conference on VLDB, Santiago, Chile, September 1994.
6. Srikant R., Agrawal R. "Mining Generalized Association Rules". – Proc. of the 21th International Conference on VLDB, Zurich, Switzerland, 1995.
7. R. Srikant, R. Agrawal. "Mining quantitative association rules in large relational tables". In Proceedings of the ACM SIGMOD Conference on Management of Data, Montreal, Canada, June 1996.
8. Pol U. Design and Development of Apriori Algorithm for Sequential to concurrent mining using MPI / U. Pol // International journal of Computers & Technology. – 2013. – T. 10. – № 7. – C. 1785–1790.
9. Rechkalov T.V. Podhod k integracii intellektual'nogo analiza dannyh v reljacionnuju SUBD na osnove generacii tekstov hranimyh procedur. – Vestnik Juzhno-Ural'skogo gosudarstvennogo universiteta. Serija «Vychislitel'naja matematika i informatika», tom 2, #1, izdatel'stvo: Juzhno-Ural'skij gosudarstvennyj universitet (nacional'nyj issledovatel'skij universitet) (Cheljabinsk), 2013. – s. 114-121.Sidló, C.I., Lukács, A. Shaping SQL-based frequent pattern mining algorithms (Revised Selected and Invited Papers). – Knowledge Discovery in Inductive Databases: 4th International Workshop, KDID 2005, Springer, Heidelberg. – p. 188–201.
10. Sidló, C.I., Lukács, A. Shaping SQL-based frequent pattern mining algorithms (Revised Selected and Invited Papers). – Knowledge Discovery in Inductive Databases: 4th International Workshop, KDID 2005, Springer, Heidelberg. – p. 188–201.
11. Munerman V.I. Modeli obrabotki bol'shih ob"emov dannyh v sistemah massovogo parallelizma . – M.: Sistemy vysokoj dostupnosti, 2013, T. 9, #1. – S. 35-43.
12. Makarov D.I., Munerman V.I. Parallel'naja realizacii operacii soedinenija dlja massovoj obrabotki dannyh Sistemy vysokoj dostupnosti. 2012. T. 8. # 3. S. 26-28.
13. Munerman V.I., Munerman D.V. Algebraicheskij podhod k postroeniju programmno-apparatnyh kompleksov dlja povyshenija jeffektivnosti massovoj obrabotki dannyh. Sovremennye informacionnye tehnologii i IT-obrazovanie. 2015. T. 2. # 11. – S. 391-396.
14. Zakharov V.N., Munerman V.I. Parallel'nyj algoritm umnozhenija mnogomernyh matric Sovremennye informacionnye tehnologii i IT-obrazovanie. 2015. T. 2. # 11. S. 384-39.
15. https://pypi.python.org/pypi.

Об авторах:

Захаров Виктор Николаевич, доктор технических наук, доцент, ученый секретарь, Федеральный исследовательский центр "Информатика и управление" РАН, vzakharov@ipiran.ru

Мунерман Виктор Иосифович, кандидат технических наук, доцент кафедры информатики, Смоленский государственный университет, vimoon@gmail.com

Самойлова Татьяна Аркадьевна, кандидат технических наук, доцент кафедры информатики, Смоленский государственный университет, tatsam@hotbox.ru

About the authors:

Zakharov Viktor N., Doctor of Technical Sciences, Associate Professor, Scientific Secretary, Federal Research Center Computer Science and Control of the Russian Academy of Sciences, vzakharov@ipiran.ru

Munerman Viktor I., Candidate of Technical Sciences, associate professor "Information technologie", Smolensk State University, vimoon@gmail.com

Samoilova Tatyana A., Candidate of Technical Sciences, associate professor "Information technologie", Smolensk State University, tatsam@hotbox.ru