УДК 004.043

ПОДХОД К ИНТЕГРАЦИИ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ В РЕЛЯЦИОННУЮ СУБД НА ОСНОВЕ ГЕНЕРАЦИИ ТЕКСТОВ ХРАНИМЫХ ПРОЦЕДУР

Т.В. Речкалов

Представлен подход к интеграции интеллектуального анализа данных (ИАД) в реляционную СУБД. Подход предполагает использование XML-разметки алгоритма ИАД, выраженного на языке SQL. Разметка позволяет выполнить автоматическую генерацию хранимых процедур на языке SQL, реализующих данный алгоритм, в зависимости от специфицированных пользователем таблиц исходных данных и параметров алгоритма. Приведено описание предложенного языка разметки. Если для решения задачи ИАД имеется несколько алгоритмов, подход предполагает генерацию SQL-кода, реализующего наиболее эффективный из них. Выбор наиболее эффективного алгоритма осуществляется на основе использования имеющейся в современных СУБД команды EXPLAIN, позволяющей получить оценку времени исполнения запроса SQL без его фактического выполнения. Описана модульная структура и интерфейс программной системы, реализующей данный подход.

Ключевые слова: интеллектуальный анализ данных, реляционная СУБД, SQL.

Введение

Интеграция алгоритмов интеллектуального анализа данных (ИАД) в реляционные системы управления базами данных (СУБД) представляет собой одну из актуальных задач аналитической обработки данных [1]. Это обусловлено тем, что существующие алгоритмы ИАД как правило предполагают размещение анализируемых данных в оперативной памяти (см. например [2, 3]), и их совместное использование с СУБД требует значительных накладных расходов, связанных с экспортом исходных данных задачи во внешнюю аналитическую утилиту и импортом результатов работы обратно в базу данных. Реализация алгоритмов ИАД на языке SQL позволяет избежать накладных расходов на экспорт-импорт данных и обеспечивает анализ данных, не помещающихся в оперативную память, на основе стандартных механизмов СУБД.

Исследования по реализации алгоритмов ИАД на языке SQL представлены следующими работами. Известны алгоритмы решения задачи рыночной корзины SETM [4] и ScanOnce [5]. Реализация алгоритма кластеризации K-Means на языке SQL рассмотрена в [1]. Реализация алгоритма нечеткой кластеризации Fuzzy c-Means описана в [6, 7]. Фреймворк для проведения ИАД в СУБД представлен в работе [8].

В данной работе описаны предварительные результаты по разработке подхода к интеграции ИАД в реляционную СУБД на основе использования XML-разметки алгоритма ИАД, выраженного на языке SQL. Разметка позволяет выполнить автоматическую генерацию хранимых процедур на языке SQL, реализующих данный алгоритм, в зависимости от указанных пользователем таблиц исходных данных и параметров алгоритма.

Работа организована следующим образом. В первом разделе на примере задачи рыночной корзины описан подход к автоматизированной генерации хранимых процедур ИАД на основе использования языка разметки алгоритма, а так же описан механизм выбора наиболее эффективного алгоритма ИАД для заданных исходных данных. Во втором разделе описана программная реализация предложенного подхода. В заключении представлена сводка основных результатов сообщения.

1. Автоматизированная генерация текстов хранимых процедур для интеллектуального анализа данных

В данном разделе описан подход к интеграции ИАД и реляционной СУБД на основе автоматизированной генерации текстов хранимых процедур на языке SQL, реализующих алгоритмы ИАД. Изложение проводится на примере алгоритма SETM решения задачи рыночной корзины [9]. Задача *рыночной корзины (market-basket problem)* заключается в нахождении всех наборов (множеств) товаров, которые часто приобретаются совместно и формально определяется следующим образом.

Пусть даны множество товаров $I=\{i_1, i_2, \ldots, i_m\}$, множество транзакций D, где каждая транзакция представляет собой множество товаров T **из** I. Каждая транзакция имеет уникальный идентификатор *TID*.

Поддержка (support) набора товаров X определяется как количество транзакций из D, каждая из которых содержит данный набор товаров X. Пусть *minsup* – минимальное значение поддержки, при котором набор товаров считается часто встречающимся.

Обозначим за F *(frequent itemsets)* множество часто встречающихся наборов. Множество F_k содержит все возможные наборы товаров из k элементов, поддержка которых не меньше *minsup*. Решением задачи является множество $F=\bigcup_k F_k$. Пример задачи рыночной корзины и ее решения приведены на рис. 1.

TID	Items
10	{A, C, D}
20	{B, C, E}
30	{A, B, C, E}
40	{B, E}

а) исходные данные

k	F_k	Support
1	{A}	2
	{B}, {C}, {E}	3
2	{A, C}, {B, C}, {B, E}, {C, E}	2
3	{B, C, E}	2

б) решение

Рис. 1. Пример задачи рыночной корзины *(minsup=2)*

При реализации алгоритма ИАД в рамках реляционной СУБД исходные данные и решение задачи должны быть представлены в виде реляционных таблиц, а алгоритм записан на языке запросов SQL. На рис. 2 представлены преобразованные исходные данные и решение задачи рыночной корзины из рис. 1.

TID	Item
10	A
10	C
10	D
...	...
40	E

а) исходные данные

F1

item1	Support
A	2
B	3
C	3
E	2

F2

item1	item2	Support
A	C	2
B	C	2
B	E	3
C	E	2

F3

item1	item2	item3	Support
B	C	E	2

б) решение

Рис. 2. Преобразование данных задачи рыночной корзины к реляционному формату

Идея алгоритма заключается в итеративной генерации множества C_k кандидатов в частые наборы и последующем отборе кандидатов с подходящим значением поддержки множества F_k. Итерации алгоритма являются рекуррентными. Пример текста запросов для итераций 2 и 3 приведен на рис. 3, изменяющиеся части запросов подчеркнуты.

```
-- Генерация кандидатов в частые
-- 2-наборы
INSERT INTO C2 as
SELECT p.tid, p.item1,
q.item1 as item2
FROM F1 p, F1 q
WHERE q.tid = p.tid
and q.item1 > p.item1;
-- Нахождение частых 2-наборов
SELECT p.item1, p.item2, COUNT(*) as
support
FROM C2 p
GROUP BY p.item1, p.item2
HAVING COUNT(*) >= minsup;
```

а) k=2

```
-- Генерация кандидатов в частые
-- 3-наборы
INSERT INTO C3 as
SELECT p.tid, p.item1, p.item2,
q.item1 as item3
FROM F2 p, F1 q
WHERE q.tid = p.tid
and q.item1 > p.item2;
-- Нахождение частых 3-наборов
SELECT p.item1, p.item2, p.item3, COUNT(*)
as support
FROM C3 p
GROUP BY p.item1, p.item2, p.item3
HAVING COUNT(*) >= minsup;
```

б) k=3

Рис. 3. Фрагмент алгоритма SETM решения задачи рыночной корзины

Заметим, что в текстах запросов для различных итераций меняются только имена используемых таблиц и списки столбцов, а структура запросов остается прежней.

В соответствии с этим нами предлагается язык разметки SQL-алгоритмов ИАД, названный *AEML (Algorithm Explain Markup Language)*. С помощью AEML можно описать строки запросов SQL, зависящие от входных данных. Описание на языке AEML преобразовывается в текст хранимой процедуры на языке SQL.

На рис. 4 представлен пример *AEML* разметки для запросов, приведенных на рис. 3. Язык AEML имеет следующие основные теги.

Тег `<text>` вставляет заключенное в нем содержимое без изменений. Тег `<list>` используется для генерации списков значений с разделителями. Например, списков колонок в запросе или значений для вставки в таблицу (рис. 3, строки 5, 15). Тег `<listParam>` используется для получения элементов списка (рис. 3, строки 6, 15). Например, в списке перечисляются значения для вставки в таблицу.

Тег `<loop>` используется для генерации наборов строк. Например, для генерации набора запросов в зависимости от параметра (рис. 3, строка 1). Тег `<loopParam>` используется для получения переменных цикла. Тег `<externalParam>` используется для разметки

входных параметров алгоритма. Например, значение минимальной поддержки (рис. 3, строка 23).

Тег `<internalParam>` используется для разметки внутренних параметров алгоритма, значение которых определятся в процессе генерации текста алгоритма. Внутренние параметры могут быть производными от внешних параметров, (например, значение минимальной поддержки, увеличенное на единицу) или от контекста генерации (например, имя таблицы, зависящее от номера итерации алгоритма (строки 2 и 9 на рис. 3).

```
1.  <loop name=‘‘iterator_K’’>
2.    <text> INSERT INTO </text> <internalParam name=‘‘candidateTableName’’/>
3.    <text> as </text>
4.    <text> SELECT p.tid, </text>
5.    <list separator=‘‘,’’ index=‘‘1’’ generator=‘‘colList’’>
6.      <text>p.item</text> <listParam index=‘‘1’’/>
7.    </list>
8.    <text> q item1 as item </text> <internalParam name=’’newItemColumnIndex’’/>
9.    <text> FROM </text> <internalParam name=‘‘freqItemsetTableName’’/>
10.   <text> p, F1 q </text>
11.   <text> WHERE q.tid = p.tid and
12.     q.item1 &gt; p.item </text> <internalParam name=‘‘itemNum’’/> <text>;</text>
13.   <text> SELECT </text>
14.   <list separator=‘‘,’’ index=‘‘1’’ generator=‘‘colList’’>
15.     <text>p.item</text> <listParam index=‘‘1’’/>
16.   </list>
17.   <text>, COUNT(*) as support</text>
18.   <text> FROM </text> <internalParam name=‘‘freqItemsetTableName’’/> <text> p</text>
19.   <text> GROUP BY </text>
20.   <list separator=‘‘,’’ index=‘‘1’’ generator=‘‘colList’’>
21.     <text>p.item</text> <listParam index=‘‘1’’/>
22.   </list>
23.   <text>HAVING COUNT(*) &gt;= </text> <externalParam name=’’minsup’’/>
24.   <text>;</text>
25. </loop>
```

Рис. 4. Разметка запросов, приведенных на рис. 3

Помимо алгоритма SETM, существуют другие алгоритмы решения задачи рыночной корзины, например, алгоритм ScanOnce [5]. Если для решения задачи ИАД имеется несколько алгоритмов, осуществляется генерация SQL-кода для наиболее эффективного из них.

Выбор наиболее эффективного алгоритма выполняется на основе использования утилиты EXPLAIN [10], входящей в состав современных СУБД. EXPLAIN позволяет получить оценку времени исполнения запроса SQL без его фактического выполнения. Утилита, используя технику оптимизации запросов, строит наиболее эффективный план запроса и выдает оценку времени выполнения в условных единицах.

Оценка алгоритма ИАД производится путем суммирования оценок EXPLAIN для всех запросов, входящих в реализацию алгоритма. Алгоритм с наименьшей оценкой является предпочтительным.

2. Программная реализация подхода

Подход, описанный в разделе 1, реализован в виде сторонней по отношению к СУБД утилиты, названной *DM-EXPLAIN*. Варианты использования утилиты DM-EXPLAIN приведены на рис. 5.

Администратор базы данных осуществляет добавление в систему описания алгоритма ИАД на языке AEML. Программист с помощью системы выполняет генерацию хранимой процедуры на языке SQL, реализующей этот алгоритм, в соответствии с заданными параметрами. В процессе генерации используются результаты оценки времени выполнения запросов, получаемых с помощью стандартной утилиты EXPLAIN.

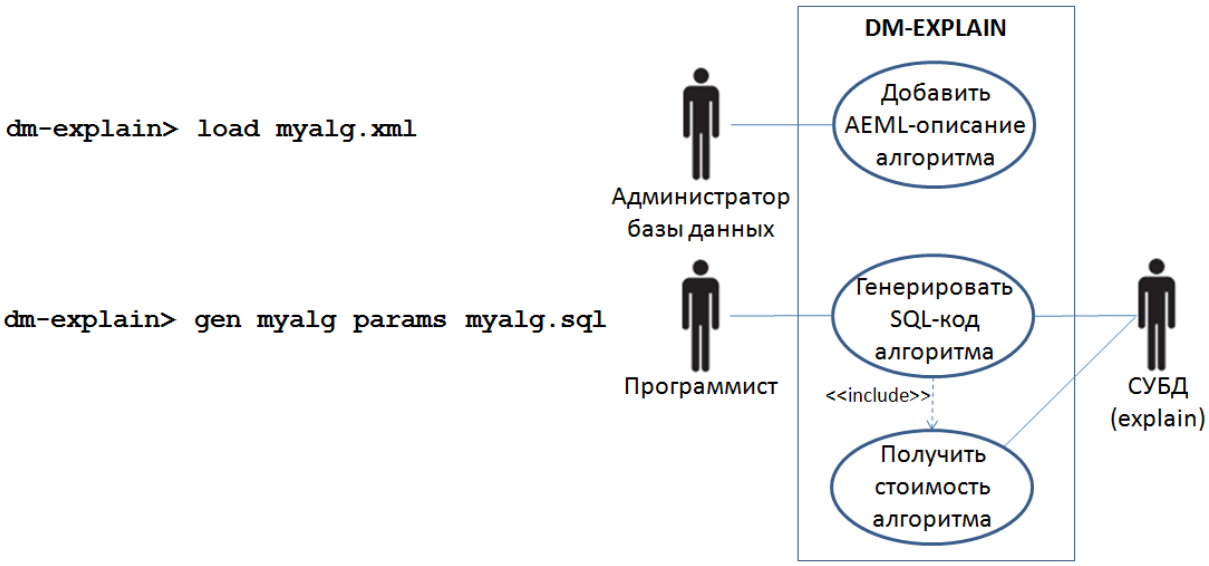

Рис. 5. Диаграмма вариантов использования утилиты DM-EXPLAIN

Модульная структура утилиты DM-EXPLAIN представлена на рис. 6.

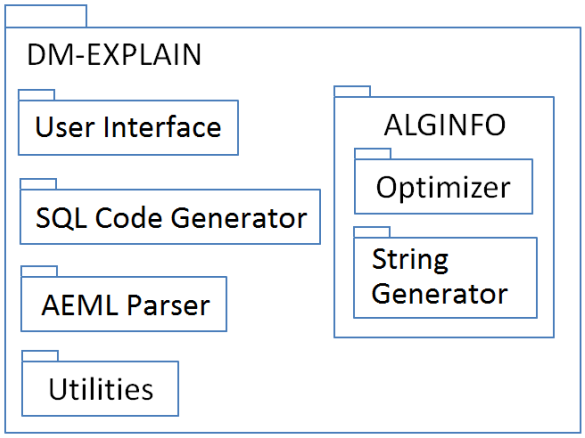

Рис. 6. Диаграмма пакетов утилиты DM-EXPLAIN

Пакет User Interface реализует пользовательский интерфейс утилиты.

Пакет SQL Code Generator обеспечивает функции автоматической генерации текстов хранимых процедур на языке SQL.

Пакет AEML Parser отвечает за разбор AEML разметки.

Пакет ALGINFO представляет собой API для разработки алгоритмов и состоит из двух вложенных пакетов: Optimizer и String Generator. Пакет Optimizer предоставляет

API для оценки сложности алгоритмов. Пакет String Generator предоставляет API для генерации подстановочных значений алгоритмов.

Пакет Utilities содержит классы, отвечающие за взаимодействие утилиты с СУБД.

Заключение

В сообщении описан подход к интеграции интеллектуального анализа данных в реляционные СУБД на основе автоматизированной генерации алгоритмов ИАД, реализованных в виде хранимых процедур. Рассмотрена проблема автоматизированной генерации хранимых процедур. Приведено описание языка разметки Algorithm Explain Markup Language. Описан механизм выбора наиболее эффективного алгоритма ИАД из нескольких реализаций. Описана модульная структура и интерфейс программной системы, реализующей данный подход.

Перспективным направлением продолжения исследований по описанной тематике является адаптация предложенного подхода к использованию в контексте параллельных СУБД [11–13], в частности, для параллельной СУБД PargreSQL [14].

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 12-07-00443-а.

Литература

1. Ordonez, C. Integrating K-Means Clustering with a Relational DBMS Using SQL // IEEE Transactions on Knowledge and Data Engineering. – 2006. – Vol. 18, No. 2. – P. 188–201.
2. Berthold, M.R. KNIME: The Konstanz Information Miner / M.R. Berthold, N. Cebron, F. Dill, et al. // Proceedings of the 31st Annual Conference of the Gessellschaft fur Klassifikation. – Springer Berlin Heidelberg, 2008. – P. 319–326.
3. Пан, К.С. Параллельный алгоритм решения задачи анализа рыночной корзины на процессорах Cell / К.С. Пан, М.Л. Цымблер // Вестник ЮУрГУ. Серия «Математическое моделирование и программирование». – 2010. – № 16(192). – Вып. 5. – С. 48–57.
4. Chung, S.M. Mining Association Rules from Relations on a Parallel NCR Teradata Database System / S.M. Chung, M. Mangamuri // Proc. of Int. Scientific Conference on Inf. Technology: Coding and Computing, 2004 (ITCC 2004). – Vol. 1. – P. 465–470.
5. Wang, F. SQL Implementation of a ScanOnce Algorithm for Large Database Mining / F. Wang, J. Gordon, N. Helian // Proceedings of the 5th Workshop on Engineering Federated Information Systems (EFIS 2003). – IOS Press, 2003. – P. 43–45.
6. Миниахметов, Р.М. Интеграция алгоритма кластеризации Fuzzy c-Means в PostgreSQL / Р.М. Миниахметов, М.Л. Цымблер // Вычислительные методы и программирование: Новые вычислительные технологии (Электронный научный журнал). – 2012. – Т. 13. – С. 46–52.
7. Miniakhmetov, R.M. Integrating Fuzzy c-Means Clustering with PostgreSQL // Труды Института системного программирования РАН. – 2011. – Т. 21. – С. 263–276.

8. Ordonez, C. A Data Mining System Based on SQL Queries and UDFs for Relational Databases // Proceedings of the 20th ACM International Conference on Information and Knowledge Management. – ACM, 2011. – P. 2521–2524.
9. Agrawal, R. Fast Algorithms for Mining Association Rules in Large Databases / R. Agrawal, R. Srikant // Proceedings of the 20th International Conference on Very Large Data Bases. – Morgan Kaufmann, 1994. – P. 487–499.
10. Ioannidis, Y.E. Query Optimization // ACM Computing Surveys. – 1996. – Vol. 28, No. 1. – P. 121–123.
11. Лепихов, А.В. Обработка запросов в СУБД для кластерных систем / А.В. Лепихов, Л.Б. Соколинский // Программирование. – 2010. – № 4. – С. 25–39.
12. Соколинский, Л.Б. Организация параллельного выполнения запросов в многопроцессорной машине баз данных с иерархической архитектурой // Программирование. – 2001. – № 6. – С. 13–29.
13. Соколинский, Л.Б. Обзор архитектур параллельных систем баз данных // Программирование. – 2004. – № 6. – С. 49–63.
14. Пан, К.С. Разработка параллельной СУБД на основе последовательной СУБД PostgreSQL с открытым исходным кодом / К.С. Пан, М.Л. Цымблер // Вестник ЮУрГУ. Серия «Мат. моделирование и программирование». – 2012. – № 18(277). – Вып. 12. – С. 112–120.

Тимофей Валерьевич Речкалов, аспирант кафедры системного программирования, Южно-Уральский государственный университет (г. Челябинск), trechkalov@gmail.com.

AN APPROACH TO INTEGRATION OF DATA MINING WITH RELATIONAL DBMS BASED ON AUTOMATIC SQL CODE GENERATION

T.V. Rechkalov, South Ural State University (Chelyabinsk, Russian Federation)

The paper introduces an approach to integration of data mining algorithms with relational DBMS. Approach assumes using XML-based markup for data mining algorithms implemented in SQL. User specifies input tables and algorithm parameters. After that system generates persistent stored procedures for specific data mining algorithm. Proposed markup language is described with examples. If there are several algorithms for some data mining problem the most effective algorithm implementation in SQL is generated by means of DBMS command EXPLAIN which shows estimated execution cost. Module structure and API for proposed program system is described.

Keywords: data mining, relational databases, SQL.

References

1. Ordonez C. Integrating K-Means Clustering with a Relational DBMS Using SQL. IEEE Transactions on Knowledge and Data Engineering, 2006. Vol. 18, No. 2. P. 188–201.
2. Berthold M.R., Cebron N., Dill F. et al. KNIME: The Konstanz Information Miner. Proceedings of the 31st Annual Conference of the Gessellschaft fur Klassifikation, 2008. Springer Berlin Heidelberg: 2008. P. 319–326.

3. Pan K.S. Parallel'nyj algoritm reshenija zadachi analiza rynochnoj korziny na processorah Cell [Parallel Algorithm for Market-Basket Problem on Cell Processors]. Vestnik JuUrGU. Serija «Mat. modelirovanie i programmirovanie» [Bulletin of the SUSU. Series «Mathematical modeling and programming»]. 2010. № 16(192). Vol. 5. P. 48–57.
4. Chung S.M., Mangamuri M. Mining Association Rules from Relations on a Parallel NCR Teradata Database System. Proceedings of International Scientific Conference on Information Technology: Coding and Computing, 2004 (ITCC 2004). Vol. 1. P. 465–470.
5. Wang F., Gordon J., Helian N. SQL Implementation of a ScanOnce Algorithm for Large Database Mining. Proceedings of the 5th Workshop on Engineering Federated Information Systems (EFIS 2003). IOS Press, 2003. P. 43–45.
6. Miniakhmetov R.M., Tsymbler M.L. Integracija algoritma klasterizacii Fuzzy c-Means v PostgreSQL [Integration of Fuzzy c-Means Clustering Algorithm with PostgreSQL Database Management System]. Vychislitelnye metody i programmirovanie: Novye vychislitelnye tehnologii (Jelektronnyj nauchnyj zhurnal) [Numerical Methods and Programming. Advanced Computing (Electronic Scientific Journal)]. 2012. Vol. 13. P. 46–52.
7. Miniakhmetov R.M. Integrating Fuzzy c-Means Clustering with PostgreSQL. Trudy Instituta sistemnogo programmirovanija RAN [Proc. of Institute for System Programming Russian Academy of Sciences]. 2011. Vol. 21. P. 263–276.
8. Ordonez C. A Data Mining System Based on SQL Queries and UDFs for Relational Databases. Proceedings of the 20th ACM International Conference on Information and Knowledge Management. ACM, 2011. P. 2521–2524.
9. Agrawal R., Srikant R. Fast Algorithms for Mining Association Rules in Large Databases. Proceedings of the 20th International Conference on Very Large Data Bases. Morgan Kaufmann, 1994. P. 487–499.
10. Ioannidis Y.E. Query Optimization // ACM Computing Surveys, 1996. Vol. 28, No. 1. P. 121–123.
11. Lepikhov A.V., Sokolinsky L.B. Query Processing in a DBMS for Cluster Systems. Programming and Computer Software. 2010. Vol. 36. No. 4. P. 205–215.
12. Sokolinsky L.B. Organization of Parallel Query Processing in Multiprocessor Database Machines with Hierarchical Architecture. Programming and Computer Software. 2001. Vol. 27. № 6. P. 297–308.
13. Sokolinsky L.B. Survey of Architectures of Parallel Database Systems. Programming and Computer Software. 2004. Vol. 30. № 6. P. 337–346.
14. Pan K.S., Zymbler M.L. Razrabotka parallel'noj SUBD na osnove posledovatel'noj SUBD PostgreSQL s otkrytym ishodnym kodom [Development of a Parallel Database Management System on the Basis of Open-Source PostgreSQL DBMS]. Vestnik JuUrGU. Serija «Mat. modelirovanie i programmirovanie» [Bulletin of the SUSU. Series «Mathematical modeling and programming»]. 2012. № 18(277). Vol. 12. P. 112–120.

Поступила в редакцию 6 ноября 2012 г.