У роботі вдосконалюється технологія автоматизованого створення матеріалізованих представлень. Запропонована оцінка запиту, що враховує статистичні і часові показники його виконання. Розроблено алгоритм розбиття запиту на лексеми і модель подання запитів у вигляді числових векторів. Введено етап кластеризації запитів для скорочення обсягів оброблюваних даних. Розроблено алгоритм групування запитів, що уточнює результати попередньої кластеризації

Ключові слова: матеріалізоване представлення, оцінка запиту, групування запитів, центральний запит

В работе усовершенствуется технология автоматизированного создания материализованных представлений. Предложена оценка запроса, учитывающая статистические и временные показатели его выполнения. Разработаны алгоритм разбиения запроса на лексемы и модель представления запросов в виде числовых векторов. Введен этап кластеризации запросов для сокращения объемов обрабатываемых данных. Разработан алгоритм группировки запросов, уточняющий результаты предварительной кластеризации

Ключевые слова: материализованное представление, оценка запроса, группировка запросов, центральный запрос, кластеризация

УДК 004.658.2
DOI: 10.15587/1729-4061.2015.50892

РАЗРАБОТКА ТЕХНОЛОГИИ АВТОМАТИЗИРОВАННОГО СОЗДАНИЯ МАТЕРИАЛИЗОВАННЫХ ПРЕДСТАВЛЕНИЙ

Е. А. Новохатская
Аспирант*
E-mail: katherinaniv@gmail.com
А. Б. Кунгурцев
Кандидат технических наук, профессор*
E-mail: abkun@te.net.ua
*Кафедра системного программного обеспечения
Одесский национальный политехнический университет
пр. Шевченко, 1, г. Одесса, Украина, 65044

1. Введение

Материализованные представления (МП) представляют собой физические объекты базы данных (БД), содержащие результаты выполнения запросов. Они позволяют многократно ускорить выполнение запросов, обращающихся к большим объемам данных, за счет использования заранее вычисленных итоговых данных и результатов соединений таблиц.

Одной из актуальных задач при оптимизации работы системы управления БД (СУБД) посредством МП является выбор запросов, результат которых может быть материализован. Актуальность проблемы обосновывается тем, что во многих СУБД МП создаются вручную администратором согласно формальным требованиям к архитектуре информационной системы (ИС) либо неформальным требованиям к ее производительности. При ручном подходе нет возможности проанализировать все запросы, поступающие в систему, учесть их статистические и временные показатели выполнения, и выбрать МП, позволяющие максимально эффективно оптимизировать работу конкретной ИС. Таким образом, созданные вручную МП решают лишь точечные проблемы производительности СУБД.

При автоматизированном создании МП необходимо проанализировать журнал транзакций СУБД за длительный период и выделить группу запросов, результат которых может быть материализован. Задача отличается высокой вычислительной сложностью. Современные СУБД обрабатывают сложные с точки зрения синтаксиса запросы. Учет различных конструкций языка SQL является еще одной проблемой при автоматизации создания МП. Помимо этого, при поиске запросов-кандидатов на материализацию необходимо учитывать различные статистические и временные показатели выполнения запросов.

2. Анализ литературных данных и постановка проблемы

В работе [1] разрабатывается набор инструментов для выбора запросов кандидатов на материализацию, использующих техники анализа данных (в частности, кластеризацию) для определения групп схожих запросов. Недостатком решения является его применимость только к запросам, содержащим соединения, агрегации и группировки. Другие синтаксические конструкции не поддерживаются.

В работе [2] рассматривается методика выбора МП с использованием алгоритма параллельной имитации отжига (Parallel Simulated Annealing, PSA) – общего алгоритмического метода решения задачи глобальной оптимизации, имитирующего физический процесс отжига металлов. Использование PSA обеспечило

масштабируемость задачи. Однако в работе решается только ее частный случай, когда множество запросов-кандидатов на материализацию уже известно, но требуется из этого множества выбрать такое подмножество МП, при котором время выполнения запросов и затраты на обслуживание МП будут минимальны.

В работе [3] были предложены две разновидности динамического интеграционного алгоритма выбора МП:

– GASAA_VSP использует генетический алгоритм для выбора локальных решений и алгоритм имитации отжига для выбора из локальных решений оптимального;

– GAACA_VSP использует генетический алгоритм для поиска начального набора решений, затем муравьиный алгоритм для уточнения результата.

Как и в предыдущей работе, авторы данной статьи рассматривают только случай выбора оптимального набора МП из существующего множества запросов-кандидатов на материализацию.

В составе СУБД Oracle присутствует инструмент SQL Access Advisor, автоматизирующий процесс создания МП [4]. SQL Access Advisor может предоставлять как точечные, так и комплексные решения по оптимизации работы ИС, включающие в себя создание или оптимизацию МП, инкрементальное обновление МП и преобразование запросов к БД в аналогичные запросы в терминах МП. Инструмент предоставляет решения на основании различных статистических параметров выполнения запросов. SQL Access Advisor также предоставляет возможность объем дискового пространства, необходимый для МП, автоматически сгенерировать SQL скрипты для создания МП, оптимизировать существующие МП таким образом, чтобы к ним был применим метод инкрементального обновления. Недостатками инструмента являются отсутствие группировки запроса, низкая производительность, отсутствие учета частоты обновления таблиц, на основе которых создано МП.

В работе [5] основной акцент делается на автоматический выбор разбиений таблиц (partitioning) в СУБД DB2, в том числе для МП.

В работе [6] предложена техника использования таблиц изменений (change-table technique), значительно повышающая производительность при обновлении агрегаций и объединений. Недостатком этого метода является его реализация на языке логической выборки (deductive query language).

В работе [7] был проведен анализ возможности применения МП в различных ИС. Была разработана технология, которая позволила автоматизировать процесс анализа журнала транзакций СУБД и процесс создания МП.

Разработанная авторами технология создания МП включает в себя следующие этапы (рис. 1).

Для анализа журнала транзакций ИС с целью выявления возможности оптимизации ее работы посредством использования МП было предложено выделить запросы вида SELECT, INSERT, UPDATE, DELETE с указанием временных и статистических показателей выполнении этих запросов, собираемых встроенными инструментами СУБД.

Период сбора данных для анализа определяется полным циклом работы предприятия (год, квартал, месяц). Выбор такого временного периода позволил получить распределение интенсивности появления одинаковых запросов в разные моменты работы ИС.

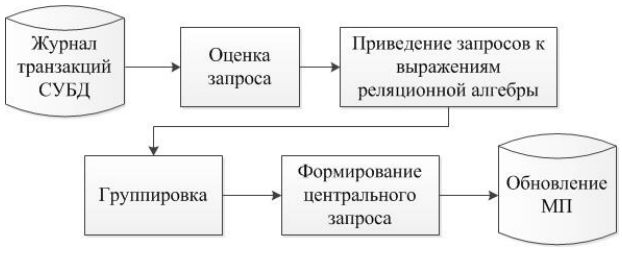

Рис. 1. Предшествующая технология автоматизированного создания МП

Для анализа запросов вида SELECT в качестве входных данных были выбраны следующие параметры:

– текст запроса;

– момент времени, когда СУБД получила запрос;

– продолжительность выполнения запроса;

– источник запроса;

– используемые ресурсы (память, время процессора) на сервере.

Для запросов типа INSERT, UPDATE, DELETE было предложено дополнительно фиксировать тип изменения и элемент данных, который был изменен.

Для оценки запроса было введено понятие стоимости, которое может быть рассчитано, как суммарное время выполнения всех одинаковых запросов:

$$S_{Z_k} = \sum_{i=1}^{N_k} T(Z_{ki}),$$

где N_k – количество появлений запроса Z_k; $T(Z_{ki})$ – время выполнения запроса Z_k в его i-ое появление в системе; k=1,K – мощность множества входных запросов.

Следующим этапом после оценки запросов в технологии [7] является синтаксический анализ запросов вида SELECT и приведение их к в виду тройки:

$$Z_k(F_k, T_k, C_k),$$

где F_k – вектор полей таблиц, участвующих в запросе Z_k; T_k – вектор таблиц; C_k – вектор условий выбора фразы WHERE.

Вектор условий C_k преобразовывается в «каноническую форму», т. е. устраняются пробелы, выравнивается регистр, имена приводятся к единой форме отображения. В таком виде вектор C_k может быть преобразован к логической форме записи, где каждый элемент условия (кроме логических операций и скобок) представлен некоторым идентификатором. Логическая форма может иметь дизъюнктивный или конъюнктивный вид записи:

(v and v and v) or (v and v and v) or (v and v).

Предложенная авторами группировка состоит из следующих этапов:

1. Выделение из множества запросов Z_k подмножеств, использующих одинаковые таблицы (анализ T_k).

2. Выделение из полученных на предыдущем этапе групп подмножеств запросов, использующих одинаковые наборы полей (анализ F_k).

3. Выделение из полученных на предыдущем этапе групп подмножеств запросов, имеющих одинаковые условия выборки полей (анализ C_k).

4. Выделение из полученных на этапе 2 групп подмножеств запросов, обладающих условиями с «перекрытием» (частичное пересечение C_k).

Последним шагом в разработанной авторами технологии является расчет центрального запроса для каждой сформированной группы однотипных запросов. Центральным был назван запрос, результат выполнения которого максимального, но возможно не полностью агрегирует в себе результаты выполнения всех запросов соответствующей группы.

Для формирования центрального запроса для каждой пары запросов группы рассчитывается стоимость взаимного пересчета одного запроса в другой. Затем отбрасываются те преобразования, стоимость которых превышает некоторое пороговое значение. Формируется граф (рис. 2), вершины которого представляет собой запросы, а ребра описывают возможные преобразования одного запроса в другой с указанием стоимости этого пересчета. Центральным будет запрос, стоимость пересчета которого в остальные запросы группы будет минимальной.

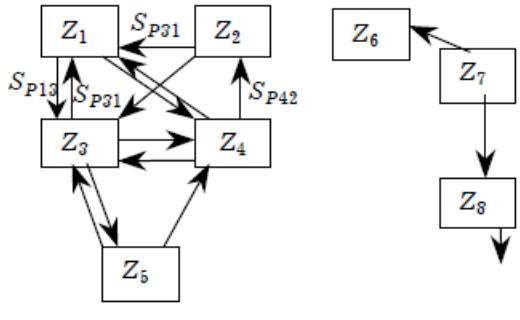

Рис. 2. Граф взаимного пересчета запросов

Выявленные недостатки:

1. Технология [7] при оценке запросов не учитывает статистические и временные показатели выполнения запросов: объем ресурсов, затраченных при их вычислении, число выполнений, частоту обновления базовых таблиц (БТ). Таким образом, МП могут не покрыть наиболее ресурсоемкие запросы или их частое обновление может сделать использование МП неэффективным.

2. Все этапы технологии [7] выполняются на полных массивах входных данных, что делает задачу вычислительно сложной. При выполнении анализа за длительный период времени (месяц, квартал, год) число запросов может достигать значительных порядков (тысяч и десятков тысяч значений).

3. Алгоритм группировки реализован на базе операций сравнения текстовых данных, что делает его ресурсоемким.

4. Группы формируются только с учетом схожести запросов. Нет возможности сформировать группу с учетом некоторого критерия, например, вокруг наиболее ресурсоемкого или часто выполняемого запроса. Т. е. не удовлетворяется основная цель создания МП – оптимизация тяжелых запросов.

5. При группировке запросов учитываются только таблицы, поля и логическая структура условных предикатов, участвующие в запросе. Отсутствует учет синтаксиса языка, например, группировок, агрегирующих функций и т. д. Это делает группы широкими и снижает их качество: увеличивает среднее расстояния между группами и снижает показатель схожести элементов внутри них.

6. Алгоритм группировки и формирования центрального запроса требуют полного перебора входных данных на каждой итерации алгоритма. Таким образом, их сложность достигает значения $(1–N)^N$, где N – число уникальных запросов вида SELECT журнала транзакций.

7. Технология [7] не принимает во внимание многие конструкции современного языка SQL, что делает ее неточной. Алгоритмы не учитывают иерархические, коррелированные запросы, подзапросы, фразы WITH и т. д., что существенно сокращает область использования данной технологии.

3. Цель и задачи исследования

Целью данной работы является усовершенствование технологии автоматизированного создания МП путем выбора более производительных и менее ресурсоемких алгоритмов анализа запросов, а также сокращения объемов обрабатываемых данных на промежуточных этапах технологии. Последнее достигается путем введения этапа предварительной группировки запросов с использованием алгоритмов кластеризации текстовых данных.

Для достижения поставленной цели необходимо решить следующие задачи:

– предложить оценку запроса, которая бы учитывала статистические и временные показатели выполнения запроса;

– разработать алгоритм разбиения запроса на лексемы для учета его синтаксических особенностей при дальнейшем анализе;

– привести текстовые запросы к форме записи в виде числовых векторов, что позволит сократить ресурсоемкость алгоритмов группировки запросов;

– ввести этап предварительной группировки запросов для сокращения объемов обрабатываемых данных;

– разработать алгоритм группировки запросов и формирования центрального запроса, уточняющий результаты предварительной кластеризации.

4. Материалы и методы исследований автоматизированного создания МП

4. 1. Материалы исследования

В качестве входных данных предлагается использовать множество неуникальных запросов разных типов (SELECT, INSERT, UPDATE, DELETE) из журнала транзакций СУБД с указанием времени их выполнения и затраченных ресурсов:

$$Q = \bigcup q < \text{text}, \tau, b >,$$

где text – текст запроса; τ – время выполнения запроса; b – суммарное число блоков данных, прочитанных с диска или буфера.

4. 2. Технология автоматизированного создания МП

В общем виде усовершенствуемая авторами технология автоматизированного создания МП представлена на рис. 3.

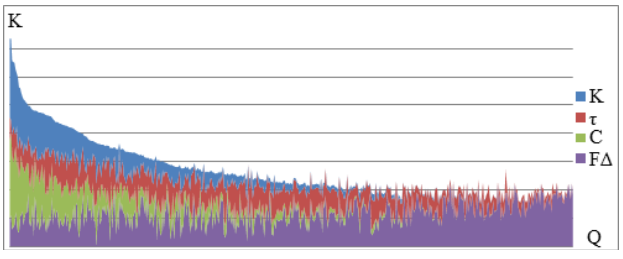

Рис. 3. Усовершенствованная технология автоматизированного создания МП

4. 3. Оценка запроса

Идея материализации запросов состоит в оптимизации наиболее ресурсоемких и часто выполняемых запросов путем сохранения предварительно вычисленных результатов выполнения этих запросов на диск. Т. к. каждая модификация данных, на основании которых МП было рассчитано, требует пересчета МП, была поставлена цель исключить создание МП для часто обновляемых данных.

В работе [8] были сформулированы следующие критерии выбора запроса-кандидата на материализацию:

1. Время выполнения запроса стремится к максимуму.

2. Частота выполнения запроса стремится к максимуму.

3. Число ресурсов, затраченных при выполнении запроса, стремится к максимуму.

4. Частота обновления БТ стремится к минимуму.

Проблему отсутствия учета статистических и временных показателей предшествующей технологии предлагаем решить путем введение коэффициента материализации, учитывающего время выполнения запроса, число потребленных ресурсов и частоту обновления БТ. Подробно алгоритм расчёта коэффициента материализации описан в работе [10]. В общем виде, его можно описать следующим выражением:

$$K = \frac{\tau cn}{F_{\Delta}}, \quad (1)$$

где n – число появлений запроса; c – стоимость выполнения запроса. Рассчитывается как суммарное число блоков данных b, затраченных за n-выполнений запроса; F_{Δ} – частота обновления БТ, участвующих в данном запросе.

Пример распределения коэффициента материализации представлен на рис. 4.

Исходя из рис. 4, коэффициент материализации K будет стремиться к максимуму для следующих типов запросов:

1. Запросы выполняются редко, но потребляют большое число ресурсов при вычислении. К таким запросам относятся различного вида отчеты о работе предприятия за некоторый период (день, неделя, месяц, квартал).

2. Запросы выполняются со средней частотой и потребляют среднее число ресурсов. В этом случае оптимизация посредством МП наиболее эффективна, т. к. данная группа запросов составляет основную нагрузку СУБД в рабочее время предприятия.

3. Запросы выполняются очень часто, но потребляют небольшое число ресурсов. Материализация данной группы запросов менее эффективна, но в целом позволит повысить производительность работы СУБД.

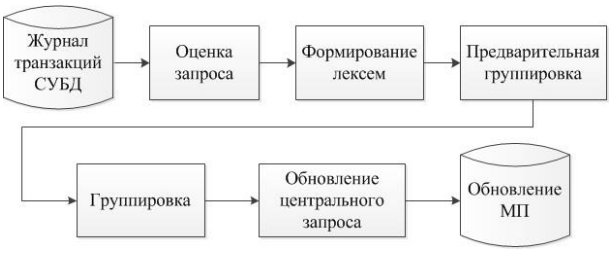

Рис. 4. Пример распределения коэффициента материализации для журнала транзакций исследуемой СУБД

Для следующих типов запросов коэффициент материализации K будет стремиться к минимуму, что позволит исключить их из числа кандидатов на материализацию:

1. Запросы, рассчитанные на основании часто обновляемых данных. Исключение данной группы позволит снизить затраты ресурсов на обслуживание МП.

2. Запросы, которые были выполнены один или несколько раз. Т. к. данная группа запросов не появляется в системе циклически, их материализация не требуется.

3. Запросы появляются часто, но выполняются предельно быстро. Их дальнейшая оптимизация не имеет смысла.

4. 4. Формирование лексем

Логическая форма записи запросов, используемая в предшествующей технологии, позволила покрыть соединения таблиц по условию равенства/неравенства и фильтрацию с помощью простых логических отношений. Также были учтены некоторые строковые и числовые функции. Однако логическая форма записи не покрыла более сложные синтаксические конструкции современного языка SQL, такие как иерархии, аналитические функции, функции агрегирования, оконные функции, различного вида подзапросы, корреляции и т. д. Еще одним недостатком данного решения является его представление данных в текстовом виде. Алгоритмы, используемые на последующих этапах технологии, оперируют строковыми данными, что делает их неоптимальными и ресурсоемкими.

Предлагается отказаться от представления данных в текстовом виде на начальных этапах обработки запросов, где объем входных данных очень велик, и перейти на представление запроса в виде числового вектора.

Остается проблема учета языковой структуры запросов. Она может быть решена использованием синтаксического анализатора. Данное решение наиболее точно, но оно не может быть применено на первых этапах технологии, когда необходимо оперировать полным объемом входных данных. Синтаксический анализатор является сложным в реализации, поскольку требует разработки грамматики. Помимо этого,

он является очень медленным и ресурсоемким, т. к. каждый запрос необходимо представить в виде дерева. В нашей задаче требуется решение, допускающее некоторую погрешность, но потребляющее минимальное число ресурсов для представления результата, чтобы снизить нагрузку на этапе группировки. Кроме того, необходимо добиться максимального быстродействия для того, чтобы обработать журнал транзакции СУБД за длительный период времени.

Предлагается разбить запросы на атомарные лексемы с учетом синтаксиса языка SQL по средствам использования строковых грамматических шаблонов. Атомарными лексемами будем называть одно или более выражений языка SQL такие, как названия полей, имена и псевдонимы таблиц, константы, функции, операторы, являющиеся минимальными смысловыми единицами при формировании фраз SELECT, FROM, WHERE, условий сортировки, группировки и т. д. [9].

К грамматическим шаблонам будем относить языковые конструкции вида:

<SUBQUERY#1> {(UNION ALL|UNION|INTERSECT|MINUS) <SUBQUERY#2>,
<TABLE#1>[INNER] JOIN < TABLE#2>,

позволяющие вычленить из запроса отдельные фразы и лексемы посредством применения регулярных выражений.

Введем понятие словаря лексем. Словарем лексем V будем называть множество уникальных лексем $L=\bigcup_{n=1}^{N} l_n$, где N – общее число найденных лексем при разборе запросов типа SELECT из журнала транзакций. Каждая запись словаря V описывается двойкой вида $<l_n, n>$, n=1..N, где l_n – текущая лексема, n – ее идентификатор (порядковый номер) в словаре V.

Для каждого запроса S_m, m=1..M, где M – число запросов типа SELECT из журнала транзакций СУБД, составим вектор D_m, содержащий множество лексем из словаря V, найденных при разборе данного запроса. Вместо строковых лексем при составлении вектора D_m будем оперировать их порядковыми номерами из словаря V. Число вхождений лексемы l в вектор D_m может быть больше 1 [9].

Подробный алгоритм формирования лексем при группировке запросов изложен в работе [9]. Кратко он может быть описан следующим образом:

1. Подготовка запроса к разбору: удаление комментариев, замена констант на шаблоны, исключение сортировок и т. д. Данный шаг позволяет на первом же этапе идентифицировать одинаковые по синтаксической структуре запросы, отличающиеся лишь значениями констант и переменных.

2. Поиск конструкций WITH и блоков, объединенных операциями над множествами (UNION, INTERSECT, MINUS и т. д.). Каждый найденный блок будет интерпретироваться при дальнейшем разборе как отдельный запрос.

3. Поиск коррелированных запросов и подзапросов во фразе SELECT. Аналогичным образом, найденные подзапросы будут анализироваться как отдельные запросы.

4. Разбиение простых фраз FROM без соединений на лексемы.

5. Выделение из фразы FROM конструкций JOIN и разбиение их на лексемы.

6. Анализ фразы WHERE и ее разбиение на множество простых условных предикатов с последующим их включением в список лексем.

7. Разбиение иерархической функции на атомарные составляющие.

8. Выделение лексем из фраз GROUP BY и HAVING.

9. Разбиение фраз SELECT на лексемы.

10. Выделение аналитических и оконных функций из фразы SELECT.

11. Добавление найденных лексем в словарь V.

12. Представление разобранного запроса в виде числового вектора на основании идентификаторов лексем из словаря V.

С развитием языка SQL алгоритм может легко расширяться путем добавления новых грамматических шаблонов.

4. 5. Предварительная группировка

Как уже было сказано ранее, группировка выполняется на полных массивах входных данных. В случаях, когда необходимо анализировать журнал транзакций за длительный период времени, задача группировки становится вычислительно сложной, т. к. число запросов в журнале транзакций может достигать значительных порядков.

Для достижения высокого качества группировки необходимо использовать синтаксический разбор запросов, что делает задачу ресурсоемкой.

Для решения этих проблем предлагается разбить группировку на два этапа. Первый этап будет применен к начальным объемам данным. Его целью является как можно производительнее обработать большие массивы данных с меньшей точностью, но тем самым сократив объемы обрабатываемых данных на последующих этапах.

Основные идеи предварительной группировки заключаются в следующем:

1. На этапе предварительной группировки предлагаем отказаться от синтаксического анализа запросов. Анализ сформированных на предыдущем этапе лексем при незначительных потерях точности группировки дает существенный выигрыш в производительности.

2. Предварительная группировка оперирует числовыми данными вместо анализа строк и формирования деревьев. Таким образом, снижается ресурсоемкость вычислений.

3. На этапе предварительной группировки предлагаем использовать алгоритмы кластеризации. Они являются более производительными и менее ресурсоемкими, чем строковые алгоритмы, используемые в предшествующих решениях.

4. Были выбраны гистограммные алгоритмы кластеризации. Они позволяют учесть синтаксис запроса, тем самым достичь высокого качества сформированных групп.

5. Усовершенствованный алгоритм кластеризации CLOPE [10] позволяет сформировать группы вокруг наиболее ресурсоемких и часто выполняемых запросов, тем самым сгруппировать запросы не только по принципу их схожести, как это делалось в предшествующей технологии, но и с учетом цели материализации – оптимизации работы СУБД.

6. Алгоритм кластеризации позволяет ограничить число групп, подлежащих дальнейшему анализу. Это требование обусловлено условиями администрирования СУБД, а именно числом ресурсов, выделенных для обслуживания МП в конкретной ИС.

7. Результатом алгоритма CLOPE являются не только сформированные группы, но и число вхождений каждой лексемы в запросы данной группы (гистограмму группы), которое будет использовано при формировании центрального запроса. Лексемы с высоким процентом вхождения должны попасть в центральный запрос. Лексемы с единичным числом вхождений потенциально могут быть исключены из группы.

8. Дальнейшая группировка будет выполняться для ограниченного числа групп запросов, выбранных в результате предварительной группировки.

Как было определено ранее, S<text, D, K> – множество запросов вида SELECT, полученных на этапе формирования лексем.

Под кластером будем понимать группу запросов, которая может рассматриваться как самостоятельная единица с общими характеристиками и удовлетворяющая следующим критериям:

1. Оператор манипуляции данными (SELECT, INSERT, UPDATE, DELETE) идентичен для всех запросов группы.

2. Обращение происходит к одинаковым таблицам.

3. Множество условий фильтрации данных одного запроса покрывает аналогичное множество другого запроса.

4. Множества полей результатов запросов пересекаются.

5. Запросы имеют схожие синтаксические конструкции (группировки, агрегации, аналитические функции и т. д.).

Множеством кластеров C является разбиение множества запросов S такое, что:

$$\left\{ C = \bigcup_{j=1}^{E} c_j = S \,|\, c_k \bigcap c_j = \varnothing, k = 1..E, k \neq j \right\}.$$

Каждый из кластеров $c_j, j = 1..E$ характеризуется следующими параметрами:

– $W(c_j)$ – ширина кластера. Соответствует числу уникальных лексем L в кластере c_j;

– $O(c_j)$ – число запросов, вошедших в кластер c_j;

– $Occ(L_n,c_j)$ – количество вхождений уникальной лексемы $L_n, n = 1..W(c_j)$ в кластер c_j;

– $P(c_j)$ – мощность кластера c_j, соответствует общему числу лексем в кластере c_j:

$$P(c_j) = \sum_{n=1}^{W(c_j)} Occ(L_n, c_j);$$

– $H(C_j)$ – высота кластера c_j:

$$H(c_j) = \frac{P(c_j)}{W(c_j)}.$$

Гистограммой кластера c_j называют графическое отображение его характеристик (рис. 5): по оси абсцисс откладываются лексемы $L_n, n = 1..W(c_j)$, в порядке уменьшения величины $Occ(L_n, c_j)$, по оси ординат – сама величина $Occ(L_n, c_j)$. Мощность кластера $P(c_j)$ геометрически соответствует площади гистограммы [10].

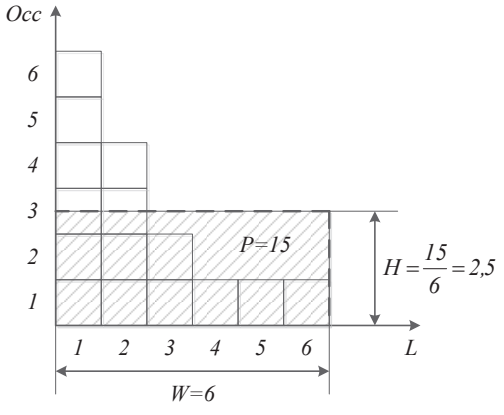

Рис. 5. Иллюстрация гистограммы кластера c=[{1,6},{2,4},{3,2},{4,1},{5,1},{6,1}]

Для начальной инициализации кластеров выберем V запросов, соответствующих максимальному значению коэффициента материализации K. Это позволит сформировать кластеры вокруг наиболее ресурсоемких и часто выполняемых запросов.

При включении s_i запроса в кластер c_j значение высоты кластера изменится следующим образом:

$$H(c_j)^{new} = \frac{\sum_{v=1}^{O(c_j)} K_v P(D_v) + K_i \sum_{y=1}^{W(D_i)} Occ(L_y, D_i)}{W(c_j \cup D_i)}.$$

Коэффициент K при этом регулирует влияние включения s_i запроса в кластер c_j соответственно «полезности» запроса с точки зрения материализации.

Функция прироста Q_{ij} не претерпела изменений и имеет такой же вид, как в классической интерпретации алгоритма CLOPE [11]. Данная функция показывает, как включение запроса s_i в кластер c_j влияет на характеристики данного кластера. Чем выше значение Q_{ij}, тем больше общих лексем было найдено при сравнении запроса с гистограммой кластера. Таким образом, цель кластеризации – максимизация функции Q_{ij}:

$$Q_{ij} = \frac{H(c_j)^{new}(O(c_j)+1)}{W(c_j \cup D_i)} - \frac{H(c_j)O(c_j)}{W(c_j)}.$$

В общем виде, алгоритм можно представить следующим образом:

I. Инициализация:

1. Создаем начальное число кластеров на основе V запросов, соответствующих MAX(K).

2. Для оставшихся (M-V) запросов, считываем очередной запрос s_i<text, D, K>.

3. Включаем запрос s_i в существующий или новый кластер c_{new}, для которого значение Q_{ij} максимально.

II. Итерация:

1. Инициализируем счетчик перестановок Z=0.

2. Считываем запрос s_i, i=1, M.

3. Исключаем запрос s_i из его текущего кластера c_{old}.

4. Включаем запрос s_i в существующий или новый кластер c_{new}, для которого значение $Q_{i\,new}$ максимально.

5. Если c_{new} и c_{old} – разные кластеры, Z=1.

6. Если значение Z=0 и итератор прошел по всем запросам множества S, выводим результат. Иначе переходим к п. 1.

В результате группировки М-запросов вида SELECT будут распределены по Е-кластерам. На следующем шаге происходит отбор некоторого процента кластеров (заданного администратором СУБД с учетом требований, предъявляемых к конкретной ИС), которые подлежат дальнейшему анализу в соответствии с одним или нескольким из нижеуказанных критериев:

1. Выбранные кластеры соответствуют максимальному значению коэффициента К. В таком случае оптимизации подлежат наиболее ресурсоемкие и часто выполняемые запросы, поступающие в систему.

2. Выбранные кластеры покрывают максимальное число запросов, поступающих в систему. Таким образом, улучшается общая производительность работы СУБД.

3. Выбранные кластеры имеют наилучшие показатели качества группировки: число общих лексем среди запросов кластера и расстояние между кластерами максимально, число единичных лексем в кластере минимально.

Рассмотрим пример кластера, полученного на этапе предварительной группировки. Параметры кластера сведены в табл. 1.

Таблица 1

Параметры рассчитанного кластера

Размер кластера	12
Гистограмма кластера	{0(8), 35(4), 32(9), 71(4), 39(21), 6(15), 8(4), 349(4), 469(10), 464(1), 104(5), 252(4), 353(1), 53(7), 22(17), 52(13), 121(2)}
Показатель схожести элементов в кластере	80.665 %
Процент единичных лексем	11.764 %

Гистограмма кластера изображена на рис. 6.

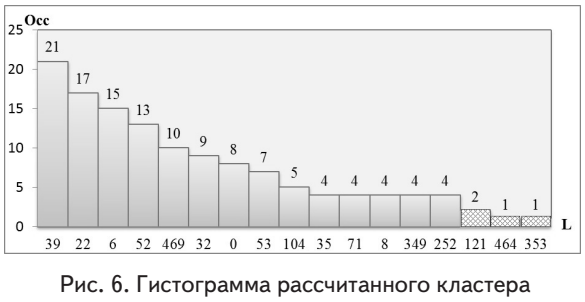

Рис. 6. Гистограмма рассчитанного кластера

Как было сказано ранее, на основе гистограммы, полученной в результате формирования кластера, можно оценить, какие лексемы потенциально должны войти в центральный запрос. Лексемы с высоким значением Occ входят в большинство запросов кластера и, следовательно, должны попасть в центральный запрос. Лексемы с низким значением Occ потенциально могут быть исключены из кластера. Единичные лексемы (Occ=1) рекомендуется не включать в центральный запрос кроме случаев, когда они принадлежат запросу с максимальным значением К, вокруг которого кластер сформирован.

Для анализируемого кластера на рис. 6 сплошным цветом отмечены лексемы, которые вошли в центральный запрос, штриховкой обозначены лексемы, которые были исключены из рассмотрения.

Примеры запросов, сформировавших полученный кластер, приведены в табл. 2.

Таблица 2

Примеры запросов, формирующих кластер

Запрос 1	Запрос 2
SELECT 1 FROM objects t, references r, objects c, params d WHERE t.parent_id=@NUMBER AND t.object_type_id=@NUMBER AND t.object_id=r.object_id AND r.attr_id=@NUMBER AND r.reference=c.object_id AND c.parent_id=@NUMBER AND c.object_type_id=@NUMBER AND c.name=@BIND AND t.object_id=d.object_id AND d.attr_id=@NUMBER AND d.value =@BIND;	SELECT i.object_id FROM references o, objects i , params t, params e WHERE i.parent_id=@NUMBER AND i.object_type_id=@NUMBER AND o.attr_id=@NUMBER AND o.object_id=i.object_id AND o.reference=@BIND AND t.object_id=i.object_id AND t.attr_id=@NUMBER AND t.list_value_id=@NUMBER AND e.object_id=i.object_id AND e.attr_id=@NUMBER AND e.list_value_id=@NUMBER;

4. 5. Группировка

Вторая стадия группировки позволяет уточнить результаты, полученные на предыдущем этапе. Она применяется к готовым кластерам, число которых ранее ограничено. Тем самым ресурсоемкость группировки компенсируется малыми объемами обрабатываемых данных.

Задачей группировки является разбиение готовых кластеров на подгруппы посредством анализа синтаксических деревьев запросов, входящих в анализируемые кластеры.

Основные идеи группировки заключаются в следующем:

1. Группировка применяется к малым объемам данных.

2. Анализу подлежат заранее сформированные кластеры. Таким образом, перебор и сравнение запросов осуществляется только в рамках анализируемой группы, что сокращает число операций сравнения.

3. На данном этапе осуществляется синтаксический разбор запросов и представление их в виде абстрактных синтаксических деревьев (АСД).

5. Группировка запросов реализуется путем сравнения деревьев.

Для синтаксического анализа запросов будем использовать контекстно-свободную грамматику языка SQL типа LL(1). Для данного класса грамматик применяются предиктивные синтаксические анализаторы, осуществляющие сканирование входного потока слева направо. На каждом шаге предпросмотра используется один символ для принятия решения о действиях синтаксического анализатора. Фразы на основе LL-грамматики формируются путем получения левых порождений [12].

Для LL(1)-грамматики языка SQL построим синтаксический анализатор. Для этого используем ANTLR [13] – генератор парсеров, позволяющий автоматически создавать программу-парсер по описанию LL-грамматики.

Каждый запрос s_i, i=1..O(c) кластера с представим в виде АСД путем его разбора средствами синтаксиче-

ского анализатора. АСД представляет собой конечное, ориентированное дерево, в котором внутренние вершины сопоставлены операторами языка SQL, а листья соответствуют константам, полям таблиц, переменным и т. д. Пример АСД для следующего ниже запроса изображен на рис. 7.

SELECT a+b FROM T WHERE c=9 AND d>3;

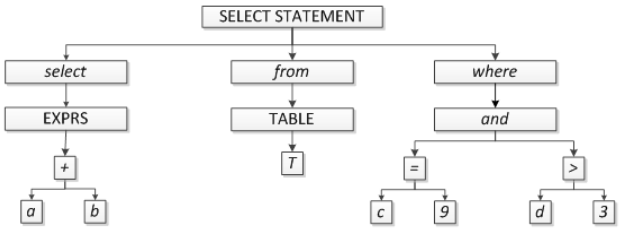

Рис. 7. Абстрактное синтаксическое дерево запроса

Опишем алгоритм группировки для некоторого кластера c. Выберем следующие параметры группировки: G – множество подгрупп кластера c; E – число подгрупп кластера c; $AST(G_i)$ – АСД подгруппы G_i, i=1..E.

Каждую вершину vg_{iy}, y=1..Y, где Y – число всех вершин дерева $AST(G_i)$, соответствующую лексеме L_n, n=1..W(c), маркируем показателем $Occ(L_n,c)$. Таким образом, в дальнейшем при формировании центрального запроса для $AST(G_i)$ из дерева будут исключены вершины, показатель $Occ(L_n,c)$ которых не превышает порогового значения.

Определим словарь преобразований запросов P, каждая запись p_z, z=1..Z которого может быть представлена тройкой <предикат, условие, операция>, где Z – число записей словаря P. Предикат описывает фразу языка SQL, которая подлежит преобразованию. Условие определяет, в каком случае преобразование может быть применено к данной фразе. Операция задает само преобразование.

Введем несколько возможных операций:

– “exclude” – преобразование невозможно. В таком случае, текущее АСД не может быть объедено с АСД подгруппы G_i, i=1..E;

– “include” – вершина текущего АСД должна быть включена в АСД подгруппы G_i;

– “modify” – вершина АСД подгруппы G_i должна быть изменена с учетом значения вершины текущего АСД;

– “skip” – включение вершины текущего АСД в АСД подгруппы G_i не требуется.

Неполный пример словаря преобразования запросов P приведен в табл. 3.

Создадим также таблицу исключений Exp с сигнатурой, аналогичной сигнатуре словаря преобразований запросов P. Если во время группировки необходимое преобразование не было найдено в словаре P, формируем запись об исключении вида <предикат, условие, NULL> в таблице Exp. По завершению группировки для каждой записи таблицы исключений Exp необходимо провести анализ и добавить операцию преобразования. Затем содержимое таблицы Exp может быть добавлено в словарь P. Таким образом, алгоритм может расширяться на новые языковые конструкции, встреченные в анализируемых запросах.

Таблица 3

Пример словаря преобразования запросов P

Предикат $AST(G_i)$	Условие $AST(s_j)$	Операция
WHERE	новый фильтр существующей таблицы	skip
WHERE	новый фильтр новой таблицы	include
WHERE field>@NUMBER1	field>@NUMBER2	modify: field>MIN (@NUMBER1, @NUMBER2)
WHERE field=@NUMBER1	field=@NUMBER2	modify: field IN (@NUMBER1, @NUMBER2)
FROM	новое объединение: INNER JOIN	include: LEFT JOIN
FROM	новое объединение: LEFT JOIN	include: LEFT JOIN
FROM	новое объединение: RIGHT JOIN	include: FULL JOIN
SELECT	новое поле	include
GROUP BY	новое поле	include
QUERY	GROUP BY	skip
QUERY	WITH	include

Распределим O(c)-запросов кластера *c* по подгруппам следующим образом:

I. Инициализация

1. Отсортируем O(c)-запросов кластера c по убыванию значения коэффициента материализации K.

2. Сформируем стартовую подгруппу, включающую запрос кластера, соответствующий максимальному значению коэффициента K:

E=1;

$G_1 = \{s_{k\,max} \mid K=MAX(K)\}$;

$AST(G_1) = AST(s_{k\,max})$.

II. Итерация

1. Считываем $AST(s_j)$, j=2..O(c) из кластера c.

2. Инициализируем итератор подгрупп i=1.

3. Если i>E, формируем новую подгруппу и переходим к п. 1:

E=E+1;

$G_i=\{s_j\}$;

$AST(G_i) = AST(s_j)$.

4. Считываем G_i.

5. Объединяем вершины $AST(G_i)$ и $AST(s_j)$ в следующем порядке:

– фраза FROM;

– фраза WITH;

– условия иерархии START WITH и CONNECT BY;

– условия объединения и фильтрации фразы WHERE;

– условия группировки GROUP BY... HAVING;

– список полей результаты фразы SELECT.

5. 1. Считываем очередную дочернюю вершину vs_{jl}, l=1..L анализируемой фразы FROM, WITH и т. д. из дерева $AST(s_j)$, где L – число дочерних вершин данной фразы.

5. 2. Для каждого vs_{jl} считываем последовательно дочерние вершины vg_{im}, m = 1..M анализируемой фразы FROM, WITH и т. д. из дерева $AST(G_i)$, где M – число дочерних вершин данной фразы.

5. 3. Ищем в словаре P соответствующее преобразование p_z, z=1..Z:

<предикат> p_z соответствует vg_{im};

vs_{jl} удовлетворяет <условию> p_z;

5. 4. Если преобразование p_z найдено и <операция> p_z не равна "exclude", модифицируем вершину vg_{im} из $AST(G_i)$ согласно <операции> p_z на основании vs_{jl} из $AST(s_j)$ и переходим к п. 5. 1.

5. 5. Если <операция> p_z равна "exclude", формируем новую группу и переходим к п. 1:

$E=E+1$;

$G_i=\{s_j\}$;

$AST(G_i) = AST(s_j)$.

5. 6. Если преобразование p_z не найдено, считываем очередную вершину vg_{im} и переходим к п. 5. 3.

5. 7. Если m=M и преобразование p_z не найдено, считываем родительскую вершину rg_{im} вершины vg_{im}, добавляем запись <rg_{im}, vs_{jl}, NULL> в таблицу исключений Exp. Инкрементируем счетчик групп i=i+1. Переходим к п. 3.

В результате группировки кластер с будет разбит на E-групп, каждая из которых будет представлена обобщенным АСД.

4. 6. Формирование центрального запроса

Для каждой группы, сформированной на предыдущем этапе, средствами синтаксического анализатора выполним обратное преобразование АСД в SQL запрос. Сформированный запрос является центральным. На основании него может быть создано итоговое МП.

5. Результаты исследований автоматизированного создания МП

Для проведения эксперимента был использован журнал транзакций, который содержал более 1389 не уникальных запросов. В результате выделения уникальных запросов и расчета агрегированных показателей было получено 502 входных запроса. После лексической обработки был сформирован словарь данных, который содержит 393 лексемы.

Анализ потребляемых ресурсов был проведен для журнала транзакций СУБД Oracle в 64-битной JVM.

Во время экспериментов был проведен сравнительный анализ алгоритмов генерации лексем и группировки, предложенных в технологии [7] («Алгоритм 1»), и аналогичных алгоритмов, описанных в данной работе («Алгоритм 2»).

5. 1. Сравнительный анализ алгоритмов генерации лексем

Сравнительный анализ алгоритмов генерации лексем осуществлялся по двум критериям: по ресурсоемкости и по покрытию синтаксиса языка SQL. Результаты анализа сведены в табл. 4, 5 соответственно.

Таблица 4

Сравнительный анализ алгоритмов по ресурсоемкости

Параметр	Алгоритм 1	Алгоритм 2
Размер словаря лексем	–	42 230 байт
Размер структуры	482 100 байт	43 415 байт
Общий размер потребляемой памяти	482 100 байт	85 645 байт

Таблица 5

Сравнительный анализ алгоритмов по покрытию синтаксиса языка SQL

Вид запроса	Алгоритм 1	Алгоритм 2
Простые запросы на выборку	+	+
Простые запросы с группировкой	+	+
Простые многотабличные запросы	+	+
Запросы с подзапросами	частично	+
Запросы с блоками WITH	–	+
Запросы с простым объединением по равенству	+	+
Многотабличные запросы с объединениями и агрегациями	частично	+
Иерархические запросы	–	+
Запросы с аналитическими функциями	–	+

5. 2. Сравнительный анализ алгоритмов группировки

При сравнительном анализе алгоритмов группировки были введены два дополнительных критерия оценки:

Процент единичных кластеров – отношение числа кластеров, содержащих единственный запрос, к общему числу рассчитанных кластеров. С точки зрения группировки запросов данный показатель идентифицирует число запросов, которые не представляют интереса в задаче создания МП.

Процент единичных лексем в кластере – отношение числа лексем в кластере, для которых значение Occ(L, C) равно единице. Показывает, какая часть кластера потенциально может не принимать участия в дальнейшей обработке группы.

Результат сравнения приведен в табл. 6.

Таблица 6

Результаты сравнительного анализа алгоритмов группировки

Параметр	Алгоритм 1	Алгоритм 2
Число сформированных кластеров	46	154
Средний показатель схожести элементов в кластере	61,44 %	70,8 %
Среднее расстояние между кластерами	75,73 %	84,89 %
Минимальное расстояние между кластерами	25 %	25,3 %
Максимальное расстояние между кластерами	100 %	100 %
Процент единичных кластеров	54,35 %	48 %
Процент единичных лексем в кластере	42,38 %	28,5 %
Количество итераций алгоритма	1	10
Время обработки данных	22181 мс	1493 мс
Размер потребляемой памяти	252 510 кбайт	11 370 кбайт

6. Обсуждение результатов исследований автоматизированного создания МП

Предложенный алгоритм формирования лексем позволил сократить размер памяти, потребляемой для хранения результатов разбора запросов, в 5,6 раз.

Учет синтаксиса языка SQL при формировании лексем позволил применить методику группировки к

запросам, содержащим блоки WITH, иерархии и аналитические функции. Также было достигнуто улучшение качества разбора запросов, содержащих подзапросы.

Быстродействие алгоритма группировки в 15 раз превзошло предшествующее решение; ресурсоемкость алгоритма снизилась в 22 раза.

Увеличено качество полученных групп:

– среднее расстояние между кластерами увеличилось на 9,16 %;

– средний показатель схожести элементов внутри кластера увеличился на 9,36 %;

– процент единичных кластеров сократился на 6,35 %;

– процент единичных лексем в кластере сократился на 13,88 %;

– число кластеров увеличилось в 3,16 раз.

7. Выводы

В данной работе была усовершенствована технология автоматизированного создания МП.

Был предложен числовой коэффициент оценки запросов с точки зрения возможности создания МП. Полученный коэффициент покрыл такие важные временные и статистические показатели выполнения запросов, как объем затраченных при формировании результата ресурсов, частоту появление запросов в ИС, а также частоту обновления БТ.

Предложенный алгоритм формирования лексем позволил существенно сократить ресурсоемкость методики группировки запросов путем оперирования числовыми данными. За счет учета синтаксических особенностей запросов помимо анализа их логической структуры удалось улучшить качество сформированных групп.

В работе алгоритмы кластеризации строковых данных были применены в задаче группировки запросов при поиске кандидатов на материализацию, что позволило снизить вычислительную сложность задачи.

Улучшены алгоритмы группировки запросов и генерации центрального запроса путем учета особенностей языка SQL.

Таким образом, разработанная технология покрыла полный цикл процесса создания МП.

Литература

1. Aouiche, K. Clustering-Based Materialized View Selection in Data Warehouses [Text] / K. Aouiche, P. Jouve, J. Darmont // Proc. East European conference on Advances in Databases and Information Systems, 2006. – P. 81–95. doi: 10.1007/11827252_9
2. Derakhshan, R. Parallel Simulated Annealing for Materialized View Selection in Data Warehousing Environments [Text] / R. Derakhshan, B. Stantic, O. Korn, F. Dehne // Proc. International Conference on Algorithms and Architectures for Parallel Processing, 2008. – P. 121–132. doi: 10.1007/978-3-540-69501-1_14
3. Zhou, L. An Improved Algorithm for Materialized View Selection [Text] / L. Zhou, H. Geng, M. Xu // Journal of Computers. – 2011. – Vol. 6, Issue 1 – P. 130–138. doi: 10.4304/jcp.6.1.130-138
4. Ashdown, L. Optimizing Access Paths with SQL Access Advisor [Electronic resource] / L. Ashdown, M. Colgan, T. Kyte // Redwood City, 2014. – Available at: https://docs.oracle.com/database/121/TGSQL/tgsql_sqlaccess.htm#TGSQL592
5. Rao, J. Automating physical database design in a parallel database [Text] / J. Rao, C. Zhang, N. Megiddo, G. Lohman // Proceedings of the 2002 ACM SIGMOD international conference on Management of data – SIGMOD '02, 2002. – P. 558–569. doi: 10.1145/564691.564757
6. Gupta, H. Incremental maintenance of aggregate and outerjoin expression [Text] / H. Gupta, I. S. Mumick // Information Systems. – 2006. – Vol. 31, Issue 6. – P. 435–464. doi: 10.1016/j.is.2004.11.011
7. Кунгурцев, А. Б. Метод анализа информационной системы для применения материализованных представлений [Текст] / А. Б. Кунгурцев, Куок Винь Нгуен Чан // Холодильна техніка і технологія. – 2005. – № 1(23). – С. 102–105.
8. Новохатская, Е. А. Расчет коэффициента материализации в оценке запросов при обслуживании материализованных представлений [Текст] / Е. А. Новохатская // Вестник ХНТУ. – 2015. – № 2 (53). – С. 128–133.
9. Новохатская, Е. А. Формирование лексем при группировке запросов в методе инкрементального обновления МП [Текст] / Е. А. Новохатская, А. Б. Кунгурцев // Вестник ЧГТУ. Серия «Технические науки». – 2014. – № 1(71) – С. 193–199.
10. Novokhatska, K. A. Application of clustering algorithm CLOPE to the query grouping problem in the field of materialized view maintenance [Text] / K. A. Novokhatska // CIT. Journal of Computing and Information Technology. – 2015. – Vol. 23, Issue 4.
11. Yang, Y. CLOPE: A Fast and Effective Clustering Algorithm for Transactional Data [Text] / Y. Yang, X. Guan, J. You // Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, 2002. – P. 682–687. doi: 10.1145/775047.775149
12. Ахо, А. Компиляторы. Принципы, технологии и инструментарий [Текст] / А. Ахо, Моника Лам, Р. Сети, Д. Ульман. – М.: Вильямс, 2002. – 1184 с.
13. Parr, T. The Definitive ANTLR Reference [Text] / T. Parr // San Francisco, 2013. – 328 p.