УДК 004.627, 004.656 DOI: 10.14529/cmse140404

АЛГОРИТМ ФРАКТАЛЬНОГО ПОИСКА В РЕЛЯЦИОННЫХ БАЗАХ ДАННЫХ

Т.Ю. Лымарь, Т.С. Мантрова, Н.Ю. Староверова

Статья посвящена вопросам разработки алгоритмов фрактального анализа реляционных баз данных. Дается обзор и сравнительный анализ известных приложений теории фракталов к обработке данных. Предложен новый алгоритм фрактального поиска в реляционной базе данных, позволяющий обнаруживать повторяющиеся группы данных. Приведена общая схема алгоритма. Рассмотрена реализация для СУБД Oracle. Представлена реализация с использованием модели распределенных вычислений MapReduce. Приводятся примеры использования разработанного алгоритма для сжатия и анализа содержимого базы данных

Ключевые слова: реляционные базы данных, теория фракталов, фрактальный анализ баз данных, сжатие данных.

Введение

Корпоративная база данных любого современного предприятия обычно содержит набор таблиц, хранящих огромное количество записей о тех или иных фактах либо объектах. Как правило, каждая запись в подобной таблице описывает какой-то конкретный объект или факт и, как правило, структуры этих записей идентичны. В связи с тем, что современные базы данных содержат огромное количество данных, которые необходимо не только компактно хранить, но и анализировать, осуществлять поиск. Для извлечения необходимой информации разрабатываются разнообразные теории и алгоритмы. Одной из подобных теория является фрактальный анализ.

«Все фигуры, которые я исследовал и называл фракталами, в моем представлении обладали свойством быть нерегулярными, но самоподобными», — писал Бенуа Мандельброт, который в 1975 г. ввел термин фрактал (от латинского fractus — дробный) [16]. Такое определение позволяет охватить широкое множество объектов, которые подпадают под понятие фрактала. Применение теории фракталов в различных областях сводится к поиску самоподобных простых частей (*доменов*), применив к которым определенную итерационную функцию, можно получить всю систему. В информационных технологиях наиболее распространенным вариантом применения теории фракталов является ее приложение к графической информации [19], что вполне естественно и понятно, однако можно рассмотреть с точки зрения данной теории и системы баз данных. Совокупность большого количества записей таблиц, может стать источником дополнительной, гораздо более ценной информации, которую нельзя получить на основе одной конкретной записи. Вполне возможно, что анализируемая информация может быть самоподобна и может отражать определенную зависимость не только между записями таблиц, но и внутри самой записи. Подобного рода информация обычно используется при прогнозировании, планировании, анализе, и ценность ее очень высока, поэтому выявление структуры данных — ключевой аспект эффективного представления и хранения этих данных.

Целью данного исследования является рассмотрение реляционных таблиц с точки зрения теории фракталов, определение подходящих методов фрактального анализа и разработка алгоритма поиска доменов в реляционных таблицах. Кроме того, необходимо

продемострировать применение полученных результатов, например, при поиске функциональных зависимостей в таблицах и сжатии хранимых данных.

В разделе 1 представлен обзор фрактальных методов интеллектуального анализа данных. Раздел 2 посвящен построени. алгоритма поиска самоподобных частей в реляционной таблице. Раздел 3 описывает ключевые аспекты реализации разработанной системы. В разделе 4 представлены результаты экспериментов для определения эффективности разработанного алгоритма. В заключении суммируются основные результаты работы.

1. Обзор методов Fractal Data Mining

Технология Data Mining основана на статистических методах и служит для выявления в «сырых» данных ранее неизвестных, нетривиальных, практически полезных и доступных интерпретаций знаний, необходимых для принятия решений. Применение фрактальных преобразований и самоподобия также относится к методам интеллектуального анализа данных [1]. Наиболее часто данная техника используются в алгоритмах сжатия данных без потери информации. Рассмотрим некоторые методы интеллектуального анализа данных, которые могут реализовываться с помощью фрактальных техник.

Кластеризация выполняет разбиение элементов некоторого множества на группы в зависимости от их схожести. У кластеризации существует большое количество практических применений, как в информатике, так и в других областях. Примерами применения могут служить: анализ данных, извлечение и поиск информации, группировка и распознавание объектов. Также кластеризация сама по себе является важной формой абстракции данных [17].

Использование самоподобия в кластеризации обеспечивает очень естественный способ определения кластеров и не ограничивается какой-либо конкретной формой кластера. В работе [2] проведен эксперимент с использованием алгоритма фрактальной кластеризации, основанный на самоподобии свойств наборов данных в кластере точек. Данный алгоритм назван «Фрактальная кластеризация» (ФК), постепенно определяет место точки в кластере, для которого может происходить изменение фрактальной размерности после добавления точки. Это очень естественный способ кластеризации точек, так как точки в одном кластере имеют большую степень самоподобия чем вне него. При добавлении новых точек вычисляется новая фрактальная размерность кластера. При этом уже образованные кластеры могут разбиваться или соединяться [2]. В работе [10] предложены алгоритмы обработки изображений, которые позволяют избежать проблем многократного повторения сканирования больших наборов данных. В работе [3] предложен метод кластеризации многомерных массивов данных с использованием MapReduce. Основные рассмотренные задачи: минимизация затрат ввода и вывода, способ разделения точек данных и объединение результатов.

Понижение размерности понимается как исключение коррелирующих атрибутов отношения. Атрибуты, которые могут быть рассчитаны из других по известному методу, можно исключить, применив методы Data Mining без ущерба для результатов [11]. Таким образом, задача превращается в выявление корреляции между атрибутами в наборе данных, и подсчет количества избыточных атрибутов, находящихся в наборе. Данный подход можно рассматривать как способ сжатия: рассматривать только те атрибуты, которые поддерживают основные характеристики хранящихся данных.

Для выявления повторяющихся данных из информации используется такое свойство «фрактальная размерности» (ФР) данных. ФР, по сути, есть оценка степени свободы набора данных. Она дает нам представление о том, каким образом данные распространяются в пространстве данных. Использование ФР набора данных может сократить время анализа данных. Основная идея, изложенная в статье [8], заключается в использовании ФР данных, и отказе от атрибутов, которые не влияют на нее. ФР является относительно устойчивой к воздействию избыточных атрибутов. Таким образом, предлагается своего рода алгоритм обратной ликвидации, который использует быстрое вычисление ФР. Этот алгоритм последовательно удаляет атрибуты, которые способствуют минимуму на ФР. В работе [8] представлен быстрый, масштабируемый алгоритм для быстрого выбора наиболее важных атрибутов (размеров) для данного набора n-мерных векторов.

2. Алгоритм фрактального поиска в реляционной БД

Мандельброт о своей теории фракталов отозвался так: «Рискнув ответить на вызов, я задумал и разработал новую геометрию природы, а также нашел для нее применение во многих разнообразных областях. Новая геометрия способна описать многие из неправильных и фрагментированных форм в окружающем нас мире и породить вполне законченные теории, определив семейство фигур, которые я называю фракталами. Наиболее полезные фракталы включают в себя элемент случайности; как правильность, так и неправильность их подчиняется статистическим законам» [16].

Покажем, что реляционная таблица также является фракталом. Для начала приведем основные определения теории реляционных баз данных. Схемой отношения R называется конечное множество имен атрибутов $\{A_1, A_2, \ldots, A_n\}$. Каждому имени атрибута A_i ставится в соответствие множество D_i — множество значений атрибута A_i, $1 \le i \le n$, D — объединение D_i, $1 \le i \le n$. Отношение r со схемой R — это конечное множество отображений $\{t_1, t_2, \ldots, t_n\}$ из R в D; причем каждое отображение t должно удовлетворять следующему ограничению: $t(A_i)$ принадлежит D_i, $1 \le i \le n$. Эти отображения называются кортежами [12].

Из определения отношения r можно отметить, что кортеж — это составная часть, которая несет в себе основную информацию о структуре отношения. Добавляя каждый последующий кортеж, мы строим отношение.

С точки зрения фрактального анализа доменом для отношения может выступать кортеж целиком или некоторая его часть. Таким образом, доменом может являться некоторая комбинация атрибутов одного кортежа, тем самым мы можем разложить отношение на еще более мелкие части, сохранив при этом сведения о его структуре.

Каждый атрибут ограничен определенным множеством значений, значит, и пара атрибутов будет также ограничена некоторым множеством. Тогда можно предположить, что домены в отношении могут быть не только идентичными по структуре, но и иметь одинаковые значения (рис. 1).

Стоит отметить тот факт, что первичный ключ отношения может выступать как некоторая функция, которая позволит определить соответствие между значением домена и кортежем.

Имя_Д	**Город_Д**	**Объем_Д**	**Цвет_Д**
Гайка	Тула	5	Красный
Гайка	Тула	10	Синий
Болт	Сургут	5	Красный
Болт	Сургут	10	Синий
Болт	Сургут	20	Черный

Болт	Сургут
Болт	Сургут

5	Красный
10	Синий
20	Черный

Рис. 1. Домены на уровне значений

Таким образом, выделяя домены в отношении, можно определить взаимосвязь атрибутов и фрактально закодировать таблицу, сохраняя сведения о ее структуре. Что позволит не только извлечь новые полезные и нетривиальные знания, но и представить отношение в более компактном виде. Полученная информация может использоваться при прогнозировании, планировании, анализе, и ценность ее очень высока, поэтому выявление структуры данных — ключевой аспект эффективного представления и хранения этих данных.

Основная идея алгоритма фрактального поиска сформулирована в [15] и заключается в подборе такого множества доменов, которое полностью опишет структуру отношения и позволит представить его содержимое минимально возможным количеством записей.

В общем виде алгоритм можно представить следующим образом (рис. 2):

Рис. 2. Общий вид алгоритма поиска доменов

Каждая часть алгоритма представляет собой отдельную задачу, которую можно решить различными способами. Далее опишем каждый шаг общего алгоритма поиска доменов более подробно.

2.1. Список возможных структур доменов

Как было сказано, доменом мы будем считать некоторую комбинацию атрибутов одного кортежа. Размером домена мы будем называть количество входящих в него атрибутов.

Пусть n — это количество атрибутов в таблице, а k — это размер домена, причем $1 \le k \le n$. Тогда количество возможных доменов размера k в таблице составляет:

$$C_n^k = \frac{n!}{k!(n-k)!}$$

Тогда общее количество всевозможных структур доменов будет:

$$\sum_{k=1}^{n} C_n^k = \sum_{k=1}^{n} \frac{n!}{k!(n-k)!}$$

При фрактальном кодировании изображения возникал вопрос об оптимизации поиска и подбора доменных областей таким образом, чтобы сохранить точность изображения. В результате под структурой домена в задаче сжатия изображения рассматривался не только размер домена, но и его вид: прямоугольник, треугольник, шестиугольник и т.д. В силу того, что в отношении кортеж есть конкретный объект, то можно считать, что иное определение домена для отношения не представляется подходящим. Таким образом, говоря о структуре домена, мы ограничиваемся лишь его размером. Оценим примерное количество всех возможных структур доменов отношения. При формировании отношения количество входящих в него атрибутов стараются ограничивать 15, поэтому количество возможных структур доменов не должно превышать 32767. Однако если при проектировании схемы базы данных не придерживаться данного ограничения, то количество атрибутов в таблице может быть намного больше, следовательно, резко возрастет количество возможных структур доменов. Тем самым сложность во многом зависит от количества атрибутов в таблице.

В связи с описанной выше проблемой, возникает вопрос о времени поиска и достаточном множестве структур доменов, которые позволят наиболее полно представить описываемую систему. Основываясь на статистических сведениях о данных, содержащихся в базе, становится возможным сократить список структур доменов. Чем меньше количество рассматриваемых структур, тем меньше времени необходимо на поиск различных значений и на формирование оптимального множества доменов.

2.2. Количество различных значений домена

Количество различных значений домена является важнейшей характеристикой структуры, по сути, она является критерием того, насколько данная структура нам подходит. Чем меньше количество различных значений домена, тем лучше, тем больше информации о взаимосвязи атрибутов структура домена отображает.

Имея сведения о количестве различных значений атрибутов, мы всегда можем максимально оценить количество различных значений для произвольной структуры домена. Количество различных значений домена всегда меньше или равно произведения количеств различных значений атрибутов, входящих в домен. Обозначим оценку количества различных значений домена KD. Значение KD может быть как больше количества кортежей в таблице, так и меньше. Если KD окажется намного больше количества кортежей в отношении, то структуру, количество различных значений которой ограничивает данное значение, рассматривать смысла не имеет. Однако если KD окажется намного меньше

количества кортежей в отношении, то данная структура, возможно, позволит подобрать оптимальное множество для кодирования отношения или разбить объекты таблицы на более крупные классы.

Исходя из изложенных выше утверждений, не вполне очевидно, что именно может являться ограничением величины KD. Выбор можно осуществлять в зависимости от поставленной задачи и максимально допустимого для нее количества различных значений. Например, сжатие в определенное число раз потребует выбора параметра KD, который будет равен части количества кортежей таблицы. Так, чтобы рассмотреть все возможные домены, значение параметра KD не должно превышать половины количества всех кортежей отношения. Если же решается задача сжатия отношения, то значение параметра KD можно ограничить меньшими значениями, например $1/3$ или $1/4$ от мощности отношения. Поведение алгоритма при различных значениях параметра KD будет исследовано в экспериментах.

Алгоритм поиска точного количества различных значений весьма прост: мы просто сравниваем каждую считанную комбинацию с уже имеющимися значениями.

Поиск количества различных значений домена является самой затратной частью алгоритма. Поэтому очень важно заранее отбросить некоторое количество структур доменов, чтобы сократить время выполнения данной части. Исключение неподходящих структур доменов может проводиться на основании сведений, хранящихся в словаре базы данных. Оценивая количество различных значений доменов, можно исключить те структуры, которые содержат большое число отличных данных. Кроме того, алгоритм поиска количества различных значений легко поддается распараллеливанию на всем множестве структур доменов, так как для каждой структуры домена алгоритм может выполняться независимо.

2.3. Оптимальное множество доменов

После того, как было получено количество различных значений доменов, можно сформировать множество доменов, которые полностью описывают таблицу. *Оптимальным множеством доменов* D будем называть совокупность структур доменов $D = D_1 \cup ... \cup D_m$, которое содержит в совокупности все атрибуты таблицы в единственном экземпляре, и сумма количества различных значений доменов будет минимальна.

Алгоритм поиска оптимального множества доменов можно представить в виде рекурсивной функции, последовательно добавляющей новые домены до тех пор, пока не просмотрены все атрибуты и общее количество различных значений доменов не превышает текущего минимума. При первом запуске значение минимума определяется как *(количество атрибутов в отношении)*(количество строк в отношении)*.

Найденное оптимальное множество доменов позволит получить список различных значений, которых достаточно, чтобы восстановить таблицу, задав определенную систему функций.

Заметим, что не для всех структур доменов количество различных значений может быть существенно меньше количества кортежей в отношении. Однако, получив сведения о количестве различных значений доменов, можно сформировать алгоритмы для извлечения новых, нетривиальных знаний из рассматриваемой таблицы:

- Понижение размерности. Полученный набор различных значений позволит определить наличие функциональных зависимостей в таблице, тем самым появляется возможность вычислить коррелирующие атрибуты.
- Классификация. Основываясь на структурах доменов с меньшим количеством различных значений, можно явно выделить классы, на которые разбиваются объекты в таблице.
- Кластеризация. Выделение доменов позволяет выполнять последовательную кластеризацию объектов, тем самым появляется возможность некоторого регулирования размера кластера и признаков разбиения, то есть разместить объект в кластере, основываясь на некоторой значимой части.

Основываясь на полученных количествах различных значений доменов, становится возможным не только закодировать отношение, но и получить достаточно информации о зависимостях между атрибутами. Основной сложностью данной части алгоритма является большое количество полученных структур доменов. Чем больше структур, тем больше информации необходимо обработать и проанализировать, поэтому время выполнения последней части во многом зависит от мощности списка возможных структур доменов.

3. Реализация алгоритма

Для реализации программной системы, был выбран язык программирования Java. Определим основные объекты, которые определяют основную логику программной системы и, соответственно, реализуют описанные выше алгоритмы. Диаграмма классов представлена на рис. 3.

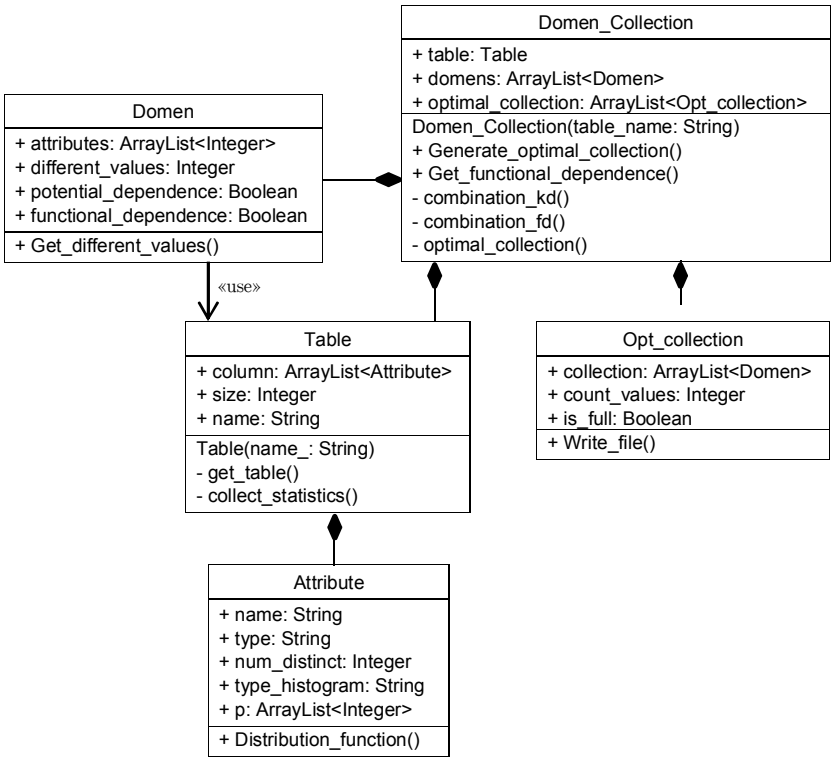

Рис. 3. Диаграмма классов

Класс «Atribute» описывает столбец отношения. Класс содержит набор атрибутов: имя столбца; тип данных; тип гистограммы; функция распределения, и метод построения функции распределения. Данные о распределении значений в столбце извлекаются из словаря СУБД. В качестве СУБД в данной реализации была использована Oracle 10g. Словарь СУБД Oracle содержит огромное количество таблиц с различной информацией о структуре базы данных, настройках системы [13], в том числе необходимые нам инструменты для сбора статистической информации и таблицы словаря данных, в которых собирается информация о распределении значений в столбцах таблицы: количество различных значений и гистограмма.

Класс «Table» задает структуру, согласно которой будут храниться сведения об анализируемом отношении. Класс содержит набор атрибутов: список атрибутов отношения; мощность отношения; имя таблицы, и методы построения статистики и получения отношения, реализующие взаимодействие с СУБД Oracle.

Класс «Domen» описывает структуру наименьшего основного фрактального объекта — домена. Атрибутами данного класса являются: список индексов, которые являются ссылками на атрибуты таблицы, составляющих домен; количество различных значений; признак наличия потенциальной функциональной зависимости; признак наличия функциональной зависимости. У класса только один метод — подсчет количества различных значений.

Класс «Domen_Collection» реализует основные алгоритмы для получения списка доменов, который позволит построить оптимальное множество и выделить функционально зависимые атрибуты. Атрибутами данного класса являются: список доменов; объект «Table», список оптимальных множеств (так как разные наборы доменов можеут давать одинаковое количество записей). Интерфейсные методы: определения функциональных зависимостей и построение оптимального множества.

Класс «Opt_collect» задает структуру для хранения оптимального множества. Класс содержит набор атрибутов: список доменов, составляющих множество; количество записей; признак полного заполнения, и метод записи хранящего оптимального множества в файл.

Как уже отмечалось выше, этап поиска количества различных значений домена является наиболее вычислительно сложным — порядка $O(m)$, где m — мощность отношения. Поэтому для реализации этого этапа алгоритма была выполнена дополнительная параллельная реализация с использованием технологии распределенных вычислений MapReduce [7]. В качестве платформы для реализации выбран проект с открытым исходным кодом Hadoop [9].

В соответствии с принципами концепции MapReduce [4] каждый домен обрабатывается Map-элементами. Пары <ключ, значение>, произведенные всеми Map-элементами, группируются по ключу и направляются в виде пары <ключ, массив значений> в адрес Reduce-элементов. Подсчитывается количество Reduce-элементов и выдается окончательный результат [14]. На рис. 4 показан пример обработки домена.

Полученное количество различных значений доменов служит основой для выбора оптимального множества доменов, которое полностью описывает таблицу.

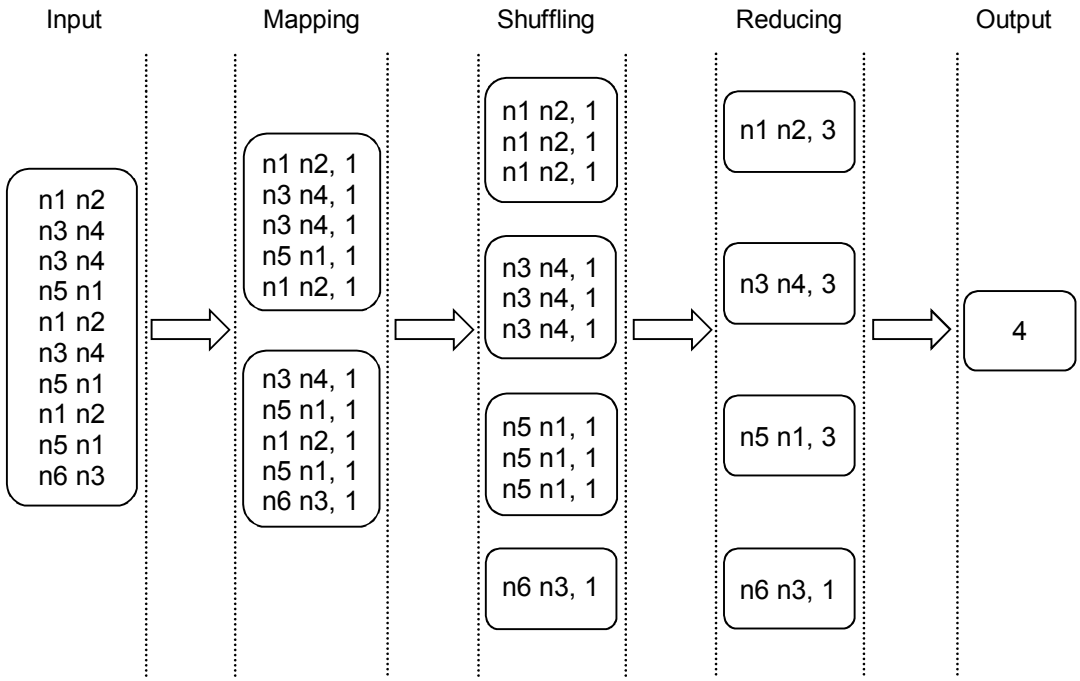

Рис. 4. Пример подсчета количества различных значений домена

Построенный алгоритм поиска оптимального множества можно отнести к алгоритму сжатия без потерь, тогда как обычно алгоритмы фрактального сжатия относят к алгоритмам с потерей информации. Сжатие с потерей информации для баз данных неприменимо. Немаловажным является и тот факт, что сжатую таким образом таблицу можно использовать для работы: выполнения запросов, вставки и удаления информации.

4. Эксперименты

Как было отмечено ранее, в отличие от большинства конкурирующих методов анализа, в которых большие наборы данных являются проблемой, метод на фрактальной основе страдает только тогда, когда данных слишком мало. Поэтому в качестве тестовой базы данных будут рассмотрены большие объемы данных реального программного комплекса «единый государственный реестр земель» (далее ПК ЕГРЗ) [18], предназначенный для ведения государственного земельного кадастра на уровне кадастрового района. Структура базы данных ПК ЕГРЗ менялась в зависимости от изменений в законе, появления приказов. Постоянное внесение изменений в структуру базы данных привело к появлению функциональных зависимостей и увеличению количества атрибутов в таблице. В качестве тестовых отношений будут рассмотрены таблицы OBJLOT, содержащая сведения о характеристиках земельных участков, и OBJ, содержащая список объектов. Таблица OBJLOT состоит из 53 атрибутов, количество кортежей 10568. Таблица OBJ состоит из 16 атрибутов, количество кортежей 63636. Таблицы заполнены сведениями, отраженными на публичной кадастровой карте [17].

Над разработанной системой было проведено несколько различных серий экспериментов. В данной статье рассмотрим два основных блока:

1) исследование эффективности кодирования таблицы оптимальным множеством доменов,
2) анализ быстродействия параллельной реализации.

Для проведения первого блока экспериментов был использован следующий подход: если сохранить в простой текстовый файл таблицу, закодированную с помощью оптимального множества доменов, и таблицу с полными записями, то сравнением размеров файлов будет получен коэффициент сжатия информации, содержащейся в таблице.

Рассмотрим построение оптимального множества для таблицы OBJ. Сравним размеры закодированных файлов при различных значениях параметра KD (табл. 1).

Таблица 1

Результаты сжатия таблицы OBJ

№ п/п	Содержимое файла	Размер файла	Коэффициент сжатия
1	Исходная таблица	5,02 Mb	1
2	Закодированная таблица, $KD = 25$ % строк таблицы	1,9 Mb	2,7
3	Закодированная таблица, $KD = 33$ % строк таблицы	1,9 Mb	2,7
4	Закодированная таблица, $KD = 50$ % строк таблицы	1,5 Mb	3,34

В результате получается, что в лучшем случае мы сможем сжать таблицу OBJ в 3,34 раза, в худшем — в 2,7 раз.

Рассмотрим построение оптимального множества для таблицы OBJLOT. Аналогично сравним размеры закодированных файлов (табл. 2).

Таблица 2

Результаты сжатия таблицы OBJLOT

№ п/п	Содержимое файла	Размер файла	Коэффициент сжатия
1	Исходная таблица	3,25 Mb	1
2	Закодированная таблица, $KD = 25$ % строк таблицы	1,3 Mb	2,50
3	Закодированная таблица, $KD = 33$ % строк таблицы	1,25 Mb	2,60
4	Закодированная таблица, $KD = 50$ % строк таблицы	1,23 Mb	2,64

В результате получается, что в лучшем случае мы сможем сжать таблицу $OBJLOT$ в 2,64 раза, в худшем — в 2,5 раза.

Таким образом, очевидно, что представление таблицы базы данных посредством оптимального множества доменов весьма эффективно.

В заключение рассмотрим результаты экспериментов над параллельной версией алгоритма. Эксперименты проводились узле суперкомпьютера «СКИФ-Аврора», характеристики которого приведены в табл. 3.

Таблица 3

Аппаратная платформа экспериментов

Характеристика	Значение

Число выч. процессоров/ядер	2/12
Тип процессора	Intel Xeon X5680 (Gulftown, 6 ядер по 3,33 ГГц)
Оперативная память	12 Гб
Тип управляющей сети	InfiniBand QDR (40 Гбит/с, макс. задержка 2 µs)

Была исследована зависимость времени выполнения алгоритма от количества используемых узлов. График ускорения выполнения алгоритма приведен на рис. 5.

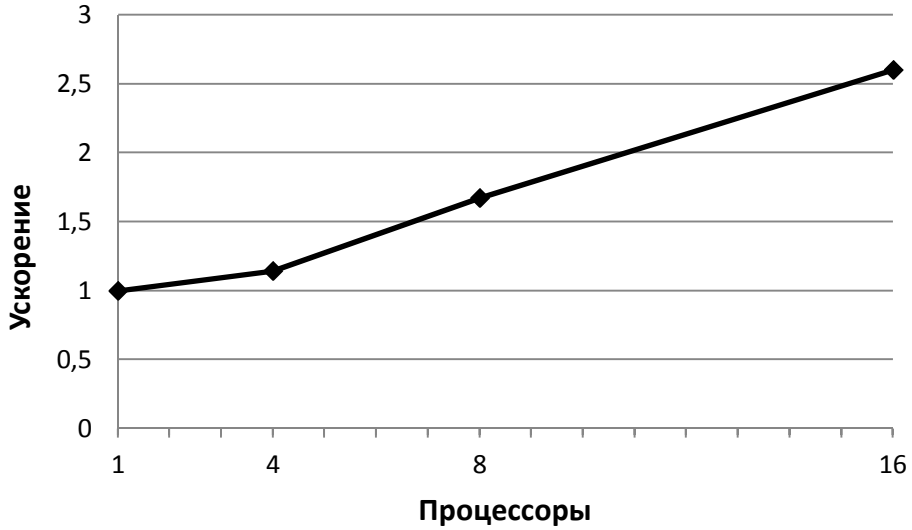

Рис. 5. Ускорение параллельного алгоритма

В дальнейшей работе планируется проведение более масштабных экспериментов.

Заключение

В статье были рассмотрены вопросы, связанные с переносом понятий и алгоритмов, разработанных в математической теории фракталов, на приложения реляционных баз данных. Исследованы известные приложения теории фракталов к обработке данных. Предложен новый алгоритм фрактального поиска в реляционной базе данных, позволяющий обнаруживать повторяющиеся группы данных. На основе предложенного алгоритма разработана программная система в контексте приложений СУБД Oracle 10g. Для наиболее трудоемкого этапа алгоритма выполнена параллельная реализация в рамках парадигмы MapReduce. Проведены вычислительные эксперименты, показавшие достаточно высокую эффективность предложенного решения.

В качестве направления дальнейшего развития работы можно рассмотреть разработку полностью параллельного алгоритма фрактального поиска.

Литература

1. Adibi, J. Fractals and Self-similarity in Data Mining: Issue and Approaches — KDD-2002 Workshop Report. — 2002. / J. Adibi, C. Faloutsos. URL: http://www.sigkdd.org/ sites/default/files/issues/4-2-2002-12/adibi.pdf (дата обращения: 1.08.2014).
2. Barbara D. Fractal Mining — Self Similarity-based Clustering and its Applications // Data Mining and Knowledge Discovery Handbook. — Springer. — 2010. — P. 573–589.

3. Ferreira Cordeiro, R.L. Clustering very large multi-dimensional datasets with MapReduce / R.L. Ferreira Cordeiro, C. Traina Jr, A.J. Traina, J. López, U. Kang, C. Faloutsos // Proceedings of the 17th International Conference on Knowledge Discovery and Data Mining (KDD '11). — 2011. — P. 690–698.
4. Lämmel, R. Google's MapReduce programming model / R. Lämmel // Science of computer programming. — 2008. — 70(1). — P. 1–30.
5. Peres, S.M. A fractal fuzzy approach to clustering tendency analysis / S.M. Peres, M. L. de Andrade Netto // Advances in Artificial Intelligence–SBIA 2004. – Springer Berlin Heidelberg. — 2004. — P. 395–404.
6. Sousa, E. A fast and effective method to find correlations among attributes in databases / E. Sousa, C. Traina, A. Traina // Data Mining and Knowledge Discovery. — 2007. — 14(3). — P. 367–407.
7. Stonebraker, M. MapReduce and parallel DBMSs: friends or foes? / M. Stonebraker // Communications of the ACM. — 2010. — P. 64–71.
8. Traina Jr, C. Fast feature selection using fractal dimension/ C. Traina Jr, A. Traina, L. Wu, C. Faloutsos // Journal of Information and Data Management. — 2010. —Vol. 1, No. 1. — P. 3–16.
9. White, T. Hadoop: The definitive guide. / T. White — O'Reilly Media/Yahoo Press — 2012. — 688 p.
10. Yan, G. The practical method of fractal dimensionality reduction based on Z-ordering technique / G. Yan, Z. Li, L. Yuan // Advanced Data Mining and Applications. — 2006. — P. 542–549.
11. Zmeškal, O. Fractal analysis of image structures. / O. Zmeškal, M. Veselý, M. Nežádal, M. Buchníček // Harmonic and Fractal Image Analysis. — 2001. — P. 3–5.
12. Дейт, К.Дж. Введение в системы баз данных, 8-е издание. / К.Дж. Дейт — М.: Издательский дом «Вильямс», 2006. — 1328 с.
13. Кайт, Т. Oracle для профессионалов. Архитектура, методики программирования и основные особенности версий 9i, 10g и 11g. / Т. Кайт — М.: Издательский дом «Вильямс», 2013. — 848 с.
14. Лымарь, Т.Ю. Фрактальный поиск в базе данных с применением модели распределенных вычислений / Т.Ю. Лымарь, Т.С. Мантрова // Параллельные вычислительные технологии (ПаВТ'2014): Труды международной научной конференции (1–3 апреля 2014 г., г. Ростов-на-Дону). — Челябинск: Издательский центр ЮУрГУ, 2014. — С. 369.
15. Лымарь, Т.Ю. Параллельный алгоритм фрактального поиска в базе данных / Т.Ю. Лымарь, Н.Ю. Староверова // Параллельные вычислительные технологии (ПаВТ'2011): Труды международной научной конференции (Москва, 28 марта – 1 апреля 2011 г.). — Челябинск: Издательский центр ЮУрГУ, 2011. — С. 703.
16. Мандельброт, Б. Фрактальная геометрия природы. / Б. Мандельброт — Москва: Институт компьютерных исследований, 2002. — 656 с.
17. Официальный Портал Росреестра. URL: http://rosreestr.ru (дата обращения: 07.05.2014).
18. Официальный сайт ФГУП ФКЦ «Земля». URL: http://www.fccland.ru (дата обращения: 07.05.2014).

19. Уэлстид, С. Фракталы и вейвлеты для сжатия изображений в действии. / С. Уэлстид — М.: Издательство Триумф, 2003. — 320 с.

Лымарь Татьяна Юрьевна, к.ф.-м.н., доцент кафедры системного программирования, Южно-Уральский государственный университет (Челябинск, Российская Федерация), lymarti@susu.ac.ru.

Мантрова Татьяна Сергеевна, магистрант, кафедра системного программирования, Южно-Уральский государственный университет (Челябинск, Российская Федерация), tatiana.mantrova@gmail.com.

Староверова Наталья Юрьевна, программист, ООО «БТ-Челябинск» (Челябинск, Российская Федерация), snu1988@yandex.ru.

Поступила в редакцию 12 августа 2014 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2014, vol. 3, no. 4, pp. 61–74

DOI: 10.14529/cmse140404

FRACTAL SEARCH ALGORITHM IN RELATIONAL DATABASES

T.Yu. Lymar, South Ural State University (Chelyabinsk, Russian Federation),

T.S. Mantrova, South Ural State University (Chelyabinsk, Russian Federation),

N.Yu. Staroverova, ООО «BT-Chelyabinsk» (Chelyabinsk, Russian Federation)

The article deals with the development of algorithms of fractal analysis of relational databases. An overview and comparative analysis of the known applications of the theory of fractals in data processing is provided. A new algorithm of fractal search in a relational database, which allows detecting duplicate data group, is presented. Implementation of the proposed algorithm for the Oracle DBMS is considered. An implementation using distributed computing MapReduce paradigm is described. Examples of using the developed algorithm to compress and analyze the contents of the database are presented. The results of computational experiments are given.

Keywords: relational databases, the theory of fractals, fractal analysis of databases, data compression.

References

1. Adibi J., Faloutsos C. KDD-2002 Workshop Report. Fractals and Self-similarity in Data Mining: Issue and Approaches URL: http://www.sigkdd.org/sites/default/files/issues/4-2-2002-12/adibi.pdf (accessed: 1.08.2014).
2. Barbara D., Chen P. Fractal Mining — Self Similarity-based Clustering and its Applications // Data Mining and Knowledge Discovery Handbook. 2010. P. 573–589. DOI: 10.1007/978-0-387-09823-4_28.

3. Ferreira Cordeiro R.L., Traina Jr C., Traina A.J., López J., Kang U., Faloutsos C. Clustering very large multi-dimensional datasets with MapReduce // Proceedings of the 17th International Conference on Knowledge Discovery and Data Mining (KDD '11). 2011. P. 690–698. DOI: 10.1145/2020408.2020516.
4. Lämmel R. Google's MapReduce programming model // Science of computer programming. 2008. 70(1). P. 1–30.
5. Peres S.M., de Andrade Netto M.L. A fractal fuzzy approach to clustering tendency analysis //Advances in Artificial Intelligence — SBIA 2004. — Springer Berlin Heidelberg, 2004. P. 395–404. DOI: 10.1007/978-3-540-28645-5_40.
6. Sousa E.P., Traina Jr C., Traina A.J. A fast and effective method to find correlations among attributes in databases // Data Mining and Knowledge Discovery. 2007. 14(3). P. 367–407. DOI: 10.1007/s10618-006-0056-4.
7. Stonebraker M. MapReduce and parallel DBMSs: friends or foes? // Communications of the ACM, 2010. P. 64–71. DOI: 10.1145/1629175.1629197.
8. Traina Jr.C., Traina A., Wu L., Faloutsos C. Fast feature selection using fractal dimension // Journal of Information and Data Management. 2010. Vol. 1, No. 1. P. 3–16.
9. White T. Hadoop: The definitive guide. Yahoo Press, 2012. 688 p.
10. Yan G., Li Z., Yuan L. The practical method of fractal dimensionality reduction based on Z-ordering technique // Advanced Data Mining and Applications. 2006. P. 542–549. DOI: 10.1007/11811305_60.
11. Zmeškal O., Veselý M., Nežádal M., Buchníček M. Fractal analysis of image structures. // Harmonic and Fractal Image Analysis. 2001. P. 3–5.
12. Date C.J. An Introduction to Database System. Addison-Wesley, 2003. 1024 p.
13. Kyte T. Expert Oracle Database Architecture Oracle Database 9i, 10g, and 11g Programming Techniques and Solutions. Apress, 2010. 832 p.
14. Lymar T.Yu., Mantrova T.S. Fractalny poisk v base dannykh s primeneniem modeli raspredelennykh vychisleny [Fractal search the database using distributed computing]. Parallelnye vychislitelnye tekhnologii (PaVT'2014): Trudy mezhdunarodnoj nauchnoj konferentsii (Rostov-on-Don, 1–3 aprelya 2014) [Parallel Computational Technologies (PCT'2014): Proceedings of the International Scientific Conference (Rostov-on-Don, Russia, April, 1–3, 2014)]. Chelyabinsk, Publishing of the South Ural State University, 2014. P. 369.
15. Lymar T.Yu., Staroverova N.Yu. Parallelny algorithm fractalnogo poiska v base dannykh [Parallel algorithm of fractal database search]. Parallelnye vychislitelnye tekhnologii (PaVT'2011): Trudy mezhdunarodnoj nauchnoj konferentsii (Moscow, 28 marta – 1 aprelya 2011) [Parallel Computational Technologies (PCT'2011): Proceedings of the International Scientific Conference (Moscow, Russia, March, 28 – April, 1, 2011)]. Chelyabinsk, Publishing of the South Ural State University, 2011. P. 703.
16. Mandelbrot B. The Fractal Geometry of Nature. NY: W. H. Freeman and Company, 1982. 461 p.
17. Official Portal of Rosreestr. URL: https://rosreestr.ru (accessed: 07.05.2014).
18. Official site of the FCC FSUE «Zemlya». URL: http://www.fccland.ru (accessed: 1.02.2014).
19. Welstead S. Fractal and Wavelet Image Compression Techniques. SPIE Publications, 1999. 254 p.

Received August 12, 2014.