Системное программное обеспечение

Л.В. Антонов
Научный руководитель – М.И. Ткачук
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл., ул. Орловская, д. 23

Анализ рынка современных пользовательских операционных систем

Целью настоящей работы является проведение комплексного анализа рынка современных операционных систем, с целью выявления основных тенденций развития и новых возможностей программных платформ.

Сегмент рынка операционных систем для стационарных компьютеров по-прежнему уверенно занимает Microsoft. Даже, несмотря на провал продукта “Windows Vista”, который сейчас активно теряет пользователей, благодаря набирающей популярность Windows 7, по данным авторитетного аналитического портала Net Applications, Microsoft занимает 93,5% мирового рынка ОС для персональных компьютеров [1].

Month	Windows	Mac	Linux
March, 2011	93.51%	5.49%	1.00%
April, 2011	93.34%	5.67%	0.99%
May, 2011	93.43%	5.60%	0.95%
June, 2011	93.32%	5.67%	1.00%
July, 2011	93.07%	5.96%	0.97%
August, 2011	92.90%	6.03%	1.07%
September, 2011	92.44%	6.45%	1.11%
October, 2011	91.86%	6.94%	1.19%
November, 2011	92.23%	6.46%	1.31%
December, 2011	92.23%	6.36%	1.41%
January, 2012	92.05%	6.39%	1.56%

Рис. 1. Статистика рынка операционных систем для персональных компьютеров на 2011-12 годы

Тем не менее, будущее лидера рынка не так безоблачно, как может показаться. Microsoft упустила момент выхода на рынок мобильных операционных систем, сочтя его не вполне перспективным, и сейчас её покрытие этого сегмента рынка составляет всего 4%[1].Также не совсем понятно стремление Microsoft в один момент изменить пользовательский интерфейс своих платформ, отказаться (частично) от концепции “Desktop” в пользу совершенно нового интерфейса под названием “Metro”, представляющего из себя совокупность прилегающих друг к другу значков-плиток, похожему на расписание подземного метро. Речь идёт о совершенно новой линейке продуктов под названием Windows 8. Таким образом, компания Стива Балмера, в которой даже существует специальное подразделение для сбора статистики работы пользователей в операционной системе, отказывается от всего накопленного за десятилетия опыта, сделав ставку на концептуально новый подход к построению интерфейса и пользовательской работы. Но генерального директора Microsoft Стива Балмера можно понять, ведь Metro–интерфейс, подходящий в большей степени для сенсорных экранов мобильных и планшетных компьютеров, где доля рынка корпорации – минимальна, а разработчики отказываются писать приложения для новой платформы, так как не считают её серьёзным игроком рынка. Выпуская новую платформу на основе Metro, Microsoft планирует интегрировать платформы для мобильных устройств и персональных компьютеров, сделать их полностью (или максимально) совместимыми и тем самым увеличить свое влияние на мобильном рынке, не теряя при этом позиции в сегменте программного обеспечения для настольных компьютеров.

В 2011-ом году Apple запустила линейку операционных систем под кодовым названием Mac OS X“Lion”. Она рассчитана на персональные компьютеры, и ноутбуки компании. В феврале 2012 вышла вторая версия операционной системы под именем “Mountain Lion”,

которая фактически тоже стирает грань между мобильной и настольной платформой. За счёт привлекательного дизайна, собственного надёжного аппаратного обеспечения, поставляемого вместе с программной платформой, Apple занимает приблизительно 5% рынка систем управления персональными компьютерами.

В современном мире пользовательских информационных технологий всё больше и больше уходят в область мобильных и планшетных устройств. Это легко объяснить тем фактом, что компьютер, интернет, повседневные приложения и программы для использования всё чаще требуются под рукой вне офиса и дома. Соответственно, мобильные гаджеты становятся адекватной заменой громоздким стационарным компьютерам и даже ноутбукам, использование которых на улице не всегда удобно. Рынок планшетов и мобильных устройств развивается активнейшим образом и по данным самого авторитетного информационно-аналитического портала TechCrunch в последнем квартале 2011 года покупатели в США стали приобретать гаджеты чаще, чем стационарные ПК [2]. Но по сути своей, переносные устройства используются во многом как платформа для развлечений и интересного время препровождения.

Крупных игроков на рынке мобильных операционных систем три: Apple, Google и Nokia. Что касается лидера сегмента рынка, то им, безусловно, является Apple, с 50% покрытия [1]. Операционная система iOS предназначена только для аппаратных продуктов компании Apple, а именно для планшетных компьютеров iPad и смартфонов iPhone. Компания создала замкнутую структуру сервисов устройств, включающую телефон, планшетный компьютер, настольный ПК, несколько интернет-магазинов и облачное хранилище данных, для синхронизации всех устройств. Благодаря удачно продуманной маркетинговой стратегии, качества и надёжности программного и аппаратного обеспечения Apple заслуженно является лидером рынка.

Но лидерство Apple не столь однозначно, как лидерство Microsoft на рынке персональных ПК. За каждую нишу в этом сегменте рынка конкуренты бьются отчаянно, используя при этом все методы, включая суды и пиар. Компания Google, выпускающая операционные системы Android не производит собственного аппаратного обеспечения, а предоставляет свой продукт другим компаниям, занимающимся сугубо аппаратной частью. Это принесло свой успех и Android на данный момент занимает 18% рынка, постоянно увеличивая аудиторию и предлагая сервисы, способные наравне конкурировать с iOS. Сейчас Google выпустил четвёртую версию продукта Android, под кодовым названием “IceCreamSandwich”, а также активно сотрудничает с такими гигантами IT-рынка как Samsung, HTSи Sony. Но в 2011 году корпорация Googleприобрела компанию Motorola, которая находилась на грани банкротства и не приносила никакой прибыли. Это говорит о том, что Googlев серьёз взялся за разработку собственного аппаратного обеспечения и готов приложить все усилия, чтобы догнать Apple.

Ещё один гигант телефонии Nokia заметно сдал позиции на рынке в 2011 году. Операционная система Symbian уступает своим основным конкурентам практически по всем показателям и стремительно теряет рынок. Если в начале марта 2011 года процент присутствия на рынке был равен 9, то в январе 2012 года цифра опустилась до 5%, что является ничтожным числом для некогда лидера рынка по продажам смартфонов. Такое положение дел вынудило Nokiaпойти на вынужденную сделку с Microsoft, присутствие которой на данном рынке также стало к настоящему времени чисто символичным. Совместная деятельность двух компаний породила в конце 2011 года смартфон “NokiaLumia 800”, на базе операционной системы WindowsPhone 7 с интерфейсом “Metro”. По данным экспертов и аналитиков портала TechCrunch телефон обладает отличным дизайном, надёжной и качественной аппаратной частью от Nokia и довольно интересным цифровым решением от Microsoft внутри телефона[2]. Аппарат не вызвал взрыва продаж телефонов Nokia, как когда-то это было с Apple. Также серьёзные нарекания вызывает факт отсутствия даже самых необходимых приложений для платформы, так как сторонние разработчики не торопятся переходить на написание программ именно для этой платформы, считая консорциум Nokia и Microsoft не самым сильным и перспективным игроком на рынке. Таким образом, к концу 2011 года WindowsPhone 7 имела все 2%рынка смартфонов.

Литература

1. Режим доступа: http://www.netapplications.com/
2. Режим доступа: http://techcrunch.com

С.А. Бабченко
Научный руководитель – доцент, канд. техн. наук К.В. Макаров
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл., ул. Орловская, д. 23

Разработка интерпретатора программ на G-коде для станков с ЧПУ

При обработке заготовки на станке инструмент совершает относительные перемещения (ходы). Совокупность перемещений, повторяющихся при изготовлении каждой детали, называется циклом обработки. Каждый цикл характеризуется величиной ходов и их последовательностью. В общем случае программа управления станком с программным управлением (ПУ) — это последовательность команд, обеспечивающих заданное функционирование его рабочих органов станка. Программа содержит размерную информацию и команды.

Наибольшей гибкостью и быстротой переналадки обладают станки с ПУ, управляемые системами, задающими программу работ в алфавитно-цифровом коде.

По виду управления станки с ПУ подразделяют на станки с системами циклового программного управления (ЦПУ) и станки с системами числового программного управления (ЧПУ) [1].

Принципиальная особенность станка с ЧПУ - это работа по управляющей программе (УП), на которой записаны цикл работы оборудования для обработки конкретной детали и технологические режимы. При изменении обрабатываемой на станке детали необходимо просто сменить программу, что сокращает трудоемкость переналадки по сравнению с трудоемкостью этой операции на станках с ручным управлением.

На производстве, где работают различные станки с числовым программным управлением, используется множество различного программного обеспечения, но в большинстве случаев весь управляющий софт использует один и тот же управляющий код. Программное обеспечение для любительских станков, так же базируется на аналогичном коде. В обиходе его называют «G-код».

G-code это условное именование языка для программирования устройств с ЧПУ (CNC). Был создан компанией Electronic Industries Alliance в начале 1960-х. Финальная доработка была одобрена в феврале 1980 года как RS274D стандарт. Комитет ИСО утвердил G-code, как стандарт ISO 6983-1:1982, Госкомитет по стандартам СССР — как ГОСТ 20999-83. В советской технической литературе G-code обозначается, как код ИСО-7 бит [2].

Управляющая система считывает инструкции программы на G-коде, которые затем интерпретатором системы ЧПУ переводятся из входного языка в команды управления главным приводом, приводами подач, контроллерами управления узлов станка (например, включить/выключить подачу охлаждающей эмульсии).

Программа, написанная с использованием G-кода, имеет жесткую структуру. Все команды управления объединяются в кадры-группы, состоящие из одной или более команд. Кадр завершается символом перевода строки (CR/LF) и имеет номер, за исключением первого кадра программы и комментариев. Первый (а в некоторых случаях ещё и последний) кадр содержит только один символ «%». Завершается программа командой M02 или M30. Комментарии к программе размещаются в круглых скобках, как после программных кодов, так и в отдельном кадре.

Команды G-кода подразделяются на:

1). Основные (называемые в стандарте подготовительными) команды языка начинаются с буквы G:

- Перемещение рабочих органов оборудования с заданной скоростью (линейное и круговое).
- Выполнение типовых последовательностей (таких, как обработка отверстий и резьба).
- Управление параметрами инструмента, системами координат, и рабочих плоскостей.

Коды	Описание
G00-G03	Позиционирование инструмента
G17-G19	Переключение рабочих плоскостей (XY, ZX, YZ)
G20-G21	Не стандартизовано
G40-G44	Компенсация размера различных частей инструмента (длина, диаметр)
G53-G59	Переключение систем координат
G80-G85	Циклы сверления, растачивания, нарезания резьбы
G90-G91	Переключение систем координат (абсолютная, относительная)

максимум 4 команды в кадре

2). Технологические команды языка начинаются с буквы M. Включают такие действия, как:

- Сменить инструмент.
- Включить/выключить шпиндель.
- Включить/выключить охлаждение.
- Работа с подпрограммами.

Код	Описание
M00	Приостановить работу станка до нажатия кнопки «старт» на пульте управления, так называемый «безусловный технологический останов»
M01	Приостановить работу станка до нажатия кнопки «старт», если включён режим подтверждения останова
M02	Конец программы, без сброса модальных функций
M03	Начать вращение шпинделя по часовой стрелке
M04	Начать вращение шпинделя против часовой стрелки
M05	Остановить вращение шпинделя
M06	Сменить инструмент
M07	Включить дополнительное охлаждение
M08	Включить основное охлаждение.
M09	Выключить охлаждение
M13	Включить охлаждение и вращение шпинделя по часовой стрелке
M14	Включить охлаждение и вращение шпинделя против часовой стрелки

не больше одного кода в кадре

Для некоторых команд необходимы дополнительные параметры. Параметры команд задаются буквами латинского алфавита

Код	Описание
X	Координата точки траектории по оси X
Y	Координата точки траектории по оси Y
Z	Координата точки траектории по оси Z
P	Параметр команды
F	Скорость рабочей подачи
S	Скорость вращения шпинделя
R	Параметр стандартного цикла или радиус дуги (расширение стандарта)
H	Параметр коррекции выбранного инструмента
P	Число вызовов подпрограммы
I, J, K	Параметры дуги при круговой интерполяции
L	Вызов подпрограммы с данной меткой

Порядок команд в кадре строго не оговаривается, но традиционно предполагается, что первыми указываются подготовительные команды, (например, выбор рабочей плоскости), затем команды перемещения, затем выбора режимов обработки и технологические команды.

Подпрограммы могут быть описаны после команды M02, но до M30. Начинается подпрограмма с кадра вида Lxx, где xx — номер подпрограммы, заканчивается командой M17.

В качестве средства разработки интерпретатора автором выбрана среда Lazarus, так как она имеет все необходимые компоненты, а используемый в ней язык программирования Pascal, довольно прост. Интерпретатор будет содержать несколько модулей: модуль, содержащий

форму приложения, модуль, отвечающий за анализ исходного текста программы, модуль для проверки правильности написания программы и модуль, сопоставляющий проанализированный текст программы с таблицей команд и переводящий из входного языка в команды управления станком.

Принцип работы интерпретатора выглядит таким образом: в модуле анализа текста происходит поочередный перебор символов текста, которые затем объединяются в лексемы. Полученные лексемы сопоставляются с таблицей команд G-кода, если установлено соответствие, то выполняется соответствующая команда, если соответствие не найдено появляется сообщение об ошибке.

Подобное построение интерпретатора целесообразно, так как появляется возможность довольно просто расширить базу операторов при необходимости, расширяя возможное применение, особенно учитывая тот факт, что производители систем управления используют G-код в качестве базового подмножества языка программирования, расширяя его по своему усмотрению.

На следующем этапе разработки планируется создание программного модуля, обеспечивающего непосредственную связь интерпретатора с контроллером станка. Результаты интерпретации команд записанных с помощью G-кода будут преобразовываться в соответствующие управляющие импульсы, передаваемые через LPT-интерфейс на контроллер станка.

Литература

1. Ловыгин А.А., Васильев А.В. Современный станок с ЧПУ и CAD/CAM система. – М.: «Эльф ИПР», 2006, 286 с.

2. G-code [Электронный ресурс] Материал из Википедии — свободной энциклопедии. - - Режим доступа: http://ru.wikipedia.org/w/index.php?title=G-code&stable=0&redirect=no

А.Е. Баринов
Научный руководитель – доцент, канд. техн. наук Р.А. Симаков
Муромский институт Владимирского государственного университета
|602264 г. Муром, Владимирской обл., ул. Орловская, д. 23
e-mail: alex@f5f5.ru

Разработка настольной СУБД

В современном мире невозможно найти область, в которой не применялись бы компьютеры. Без них невозможно представить офисы, торговые предприятия, крупные корпорации и даже складские помещения. Но до сих пор главными задачами, решаемыми только с помощью вычислительной техники, являются сбор, обработка и хранение различных данных. Структурировать и использовать совокупность полученных данных позволяют системы управления базами данных (СУБД) [1].

СУБД как программный комплекс для управления и ведения баз данных, является составляющей частью практически любой информационной системы. Самые известные продукты СУБД: Oracle, Firebird, MS SQL Server, MySQL и др. Все они обладают большим набором функций и возможностей, которые можно применять при решении различных задач. Но бывают случаи, когда появляется необходимость разработать собственную СУБД, нацеленную решение определенных проблем или узкоспециализированную в какой либо области.

В данной научной работе разрабатывается спроектированная настольная СУБД. Предпосылкой к ее созданию являлась необходимость в решении некоторых задач:

- использование в качестве стенда для обучения и отработке навыков создания собственных СУБД;
- реализация, оптимизация и тестирование новых приемов в разработке СУБД;
- оптимальное использование новых технологий и современного оборудования, таких как: многоядерные процессоры, многопоточность, доступность больших объемов для хранения данных на внешних носителях [2, 3];
- выбор структуры хранения и представления данных: табличная, иерархическая, сетевая и др.

При реализации необходимой СУБД соблюдается ряд требований, наличие которых являются базовым практически для любых аналогичных продуктов:

- составление механизм страничной организации доступа к дисковому файлу базы данных;
- реализация алгоритма битовой карты выделения памяти, в которой при помощи бинарного массива сообщается состояние свободной или занятой страницы
- написание механизма транзакций;
- составление механизма описания и реализации структуры;
- разработка механизма получения требуемых данных по средствам специализированных запросов [4, 5].

Разработка СУБД является сложным процессом, которому предшествовал не менее важный этап – процесс проектирования, где были установлены окончательные требования, выявлены явные нюансы, возможные проблемы и трудности. Наряду с этим непосредственно в дальнейшем процессе реализации, некоторые уже установленные аспекты могут претерпеть изменения, корректироваться, дополняться и дорабатываться.

Литература

1. Comparison of different SQL implementations against SQL standards. Includes Oracle, DB2, Microsoft SQL Server, MySQL and PostgreSQL. (08/Jun/2007)
2. Илюшечкин В.М. Основы использования и проектирования баз данных. «Юрайт ИД Юрайт», 2011, 213с.
3. Смирнов А.В., Пугин Е.В. Разработка модели оптимизатора запрос / Алгоритмы, методы и системы обработки данных. 2010. №15
4. Gaven Powell, Beginning Database Design, 2005, 504p.
5. Симаков Р.А., Домнин Н.А. Моделирование пользовательской нагрузки как метод производительности сервера БД / Алгоритмы, методы и системы обработки данных. 2007. № 12.

Д.П. Гранченко
Научный руководитель – доцент, канд. техн. наук Д.Н. Стародубов
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл., ул. Орловская, д. 23
e-mail: kaf-eivt@yandex.ru

Разработка метода доступа соединения слиянием для односторонних внешних соединений в оптимизаторе СУБД Firebird

Алгоритм соединения занимает одно из центральных мест в оптимизаторе любой СУБД. Исходными данными для операции являются два отношения (таблицы) и описание условия соединения. Результатом операции является отношение (таблица), получаемая как декартово произведение исходных отношений, ограниченная условием соединения. В практических реализациях соединение обычно не выполняется, как ограничение декартова произведения. Имеются более эффективные алгоритмы, гарантирующие получение такого же результата.

При внутреннем соединении в случае наличия индекса по условию связи оптимизатор Firebird выбирает алгоритм соединения вложенными циклами. Упрощая до двух таблиц, алгоритм можно описать следующим образом: для каждой строки одной из таблиц (ведущей) выполняется поиск в другой таблице (ведомой) строк соответствующих условию соединения.

Это самый общий и поэтому незаменимый алгоритм соединения. При помощи соединения вложенными циклами можно реализовать любое условие соединения. Остальные алгоритмы имеют ограничения по реализуемым с их помощью условиям соединения (например, когда условие выражается неравенством). При использовании поиска по индексу алгоритм лучше всех масштабируется. То есть при увеличении объёма данных в соединяемых таблицах время выполнения запроса увеличивается практически линейно при тех же аппаратных ресурсах.

При отсутствии индекса выбирает алгоритм соединения слиянием. Для выполнения соединения потоков по этому алгоритму первоначально записи в них сортируются по ключу связи. При запросе на извлечение очередной записи, считываются записи из внешнего и внутреннего потока и происходит сравнение их ключей. В случае равенства запись отправляется на выход. В случае если ключ во внешнем потоке больше, чем ключ во внутреннем потоке, то из внутреннего потока извлекаются записи, пока ключ внутреннего потока не станет большим или равным ключу во внешнем потоке. Таким образом, записи во внутреннем потоке, для которых не нашлось соответствие, опускаются.

Если ключ во внешнем потоке меньше, чем ключ во внутреннем потоке, то запись из внешнего потока соединяется с null’ом и подается на выход. Алгоритм прекращается, когда заканчиваются записи во внешнем потоке.

Алгоритм выполняется в один проход, то есть записи из каждого потока читаются только один раз. Это возможно благодаря тому, что записи в потоках предварительно упорядочены.

Алгоритм используется только при сравнении ключей на строгое равенство. Условия связи с использованием неравенств не могут быть выполнены посредством однопроходного слияния, однако допускается слияние на основе выражений и обработка нескольких условий связи, соединенных через AND. В этом случае входные потоки сортируются по нескольким полям.

При отсутствии индексов алгоритм соединения слиянием значительно выигрывает по скорости у алгоритма соединения вложенными циклами, однако при наличии индексов он работает медленнее, поскольку затрачивается время на сортировку входных потоков.

Для внешних соединений алгоритм слияния не реализован и в случае отсутствия подходящих индексов все равно используется соединение рекурсивным перебором, что приводит к низкой скорости выполнения запроса и извлечения записей. Актуальной является задача разработки подобного метода доступа для односторонних внешних соединений.

С.А. Жолобов
Научный руководитель – доцент, канд. техн. наук Р.А. Симаков
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл., ул. Орловская, д. 23

Разработка и исследование тестов производительности астрономических запросов на СУБД Firebird/Postgres

В настоящее время разрабатывается несколько проектов в астрономической сфере. Каждый из них обрабатывает или будет обрабатывать колоссальные объёмы данных [1, 2]. Задача исследования состоит в предоставлении тестов производительности различных СУБД, которые могли бы использоваться для хранения и обработки данных, либо служить в качестве нормативов при разработке специфичных СУБД.
Джим Грей предложил 20 астрономических запросов, помогающие астрономам в определении тех или иных объектов, их свойств, расстояния до них и между ними, а так же их траектории [3]. В некоторых случаях, запросы итеративные, то есть результат одного запроса передаётся в следующий. В полноразмерной базе данных многие запросы выполняются за несколько секунд. Выполнение запросов, включающих последовательное сканирование базы данных, занимает около трёх минут. Одно объединение таблиц может занять 10 минут [4].

В качестве исходных данных для предстоящего исследования используются 1% реальных данных SkyServer в формате MDF (MS SQL Server Database File), 20 астрономических запросов для оценки производительности, а также СУБД Firebird 2.5 и PostgreSQL 9.1.

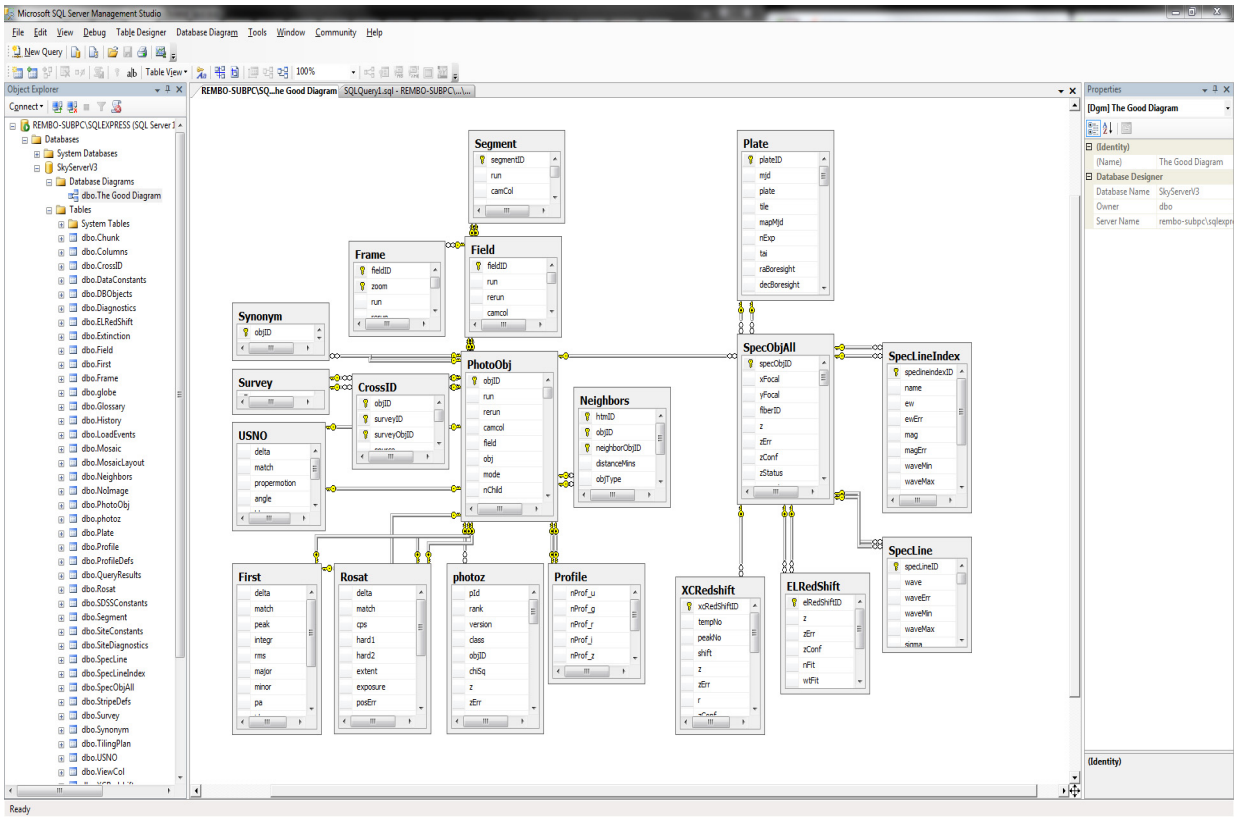

Рис. 1. Схема базы данных

На рисунке представлена схема базы данных тестового сервера. Как видно из диаграммы, большинство сущностей связаны с сущностью «PhotoObj», которая в свою очередь является хранилищем объектов, полученных из фотографий звёздного неба [4].

Суть исследования заключается в измерении времени выполнения каждого запроса в каждой СУБД. Перед началом исследования необходимо:

- воссоздать структуру базы данных в каждой СУБД;
- подключить пользовательские функции;
- добавить тестовые данные;
- преобразовать запросы в синтаксис каждой СУБД.

В таблице представлены результаты выполнения запросов в СУБД MS SQL Server. По результатам исследования, получившиеся данные будут сопоставлены с эталонными, чтобы проанализировать успешность выбранной системы управления базами данных. Корректные результаты исследования можно получить, только проведя тесты всех СУБД на одном и том же сервере.

Таблица 1

Эталонные данные

Запрос	Работа процессора, с	Время выполнения, с	Ввод / вывод	Число записей
Q01	0.25	8.08	603	19
Q02	0.17	1.94	218	21761
Q03	10.14	76.15	22146	0
Q04	0.06	0.59	69	13
Q05	1.09	13.10	3397	251
Q06	0.20	0.98	82	12495
Q07	0.42	0.94	0	19
Q08	0.00	0.04	1	3
Q09	0.00	0.00	0	1
Q10	0.01	2.28	73	54
Q10A	0.00	0.00	0	0
Q11	0.08	0.94	287	300
Q12	0.27	1.11	98	11
Q13	0.14	0.16	19	108
Q14	0.87	2.22	155	374
Q15A	0.44	4.63	419	6
Q15B	0.05	0.05	0	0
Q16	0.06	0.06	0	1
Q17	0.13	0.33	89	39
Q18	3.42	3.87	79	34
Q19	0.09	0.10	0	16
Q20	0.05	0.11	0	0

В результате были подготовлены исходные данные, получены эталонные результаты тестов производительности астрономических запросов в СУБД MS SQL Server. Проведение тестов в выбранных СУБД и соотнесение результата с эталонными, позволит сформировать рейтинг систем управления базами данных, в области обработки астрономических запросов.

Литература

1. SDSS SkyServer DR7 - http://skyserver.sdss.org/
2. SciDB - http://www.scidb.org/
3. Designing and Mining Multi-Terabyte Astronomy Archives: The Sloan Digital Sky Survey 4. http://research.microsoft.com/pubs/68553/ms_tr_99_30_sloan_digital_sky_survey.pdf
4. 20+ queries - http://research.microsoft.com/en-us/um/people/gray/Papers/MSR_TR_O2_01_20_queries.doc
5. SS-DB: A Standard Science DBMS Benchmark - http://www-conf.slac.stanford.edu/xldb10/docs/ssdb_benchmark.pdf

К.Д. Кокурин
Научный руководитель – доцент, канд. техн. наук Е.Е. Макарова
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл., ул. Орловская, д. 23

Программа учета работы пользователя в сети Internet

С развитием корпоративных информационных технологий и в связи с широким внедрением услуг Internet на предприятиях, встает вопрос о контроле работы пользователей в глобальной сети. В докладе рассматриваются вопросы разработки программы учета работы пользователей в сети Internet. Процесс сбора и обработки информации по посещаемым пользователями ресурсам вручную занимает большой объем времени, в особенности, если в сеть включено несколько десятков пользователей. Программа разрабатывается для того, чтобы автоматизировать сбор и обработку информации о работе пользователей в глобальной сети, что позволит ускорить и соответственно облегчить процесс контроля работы пользователей.

В контексте проекта актуально использовать понятие управленческого учета. Управленческий учёт — упорядоченная система выявления, измерения, сбора, регистрации, интерпретации, обобщения, подготовки и предоставления важной для принятия решений по деятельности организации информации и показателей для управленческого звена организации (внутренних пользователей — руководителей) [1].

Для того, чтобы эффективно вести учет работы пользователей в глобальной сети, администратору необходимо взаимодействовать со всей сетью компьютеров пользователей. Взаимодействие можно представить в виде схемы:

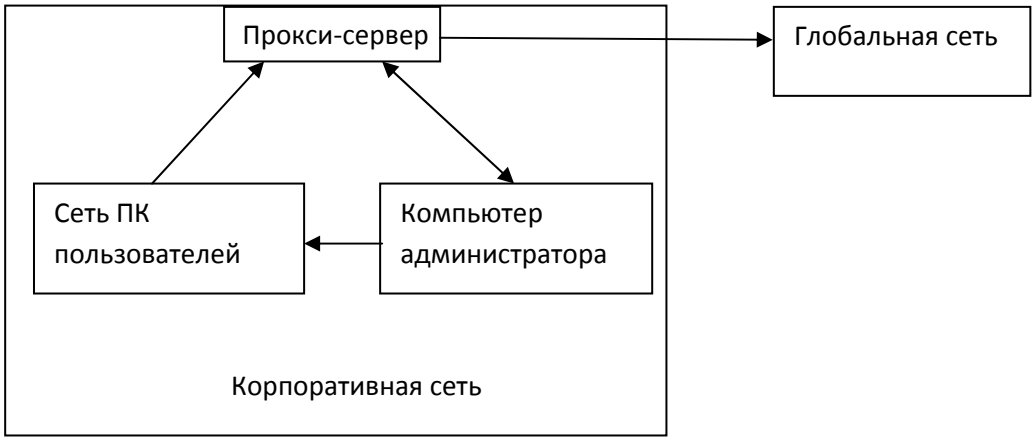

Рис. 1. Принципиальная схема взаимодействия сети пользователей и администратора

Важно отметить, что прокси-сервер может быть установлен как на отдельном компьютере-сервере, так и непосредственно на компьютере администратора. В рамках проекта было решено использовать наиболее распространенный прокси-сервер Squid. Это программный пакет, реализующий функцию кэширующего прокси-сервера для протоколов HTTP, FTP, Gopher и (в случае соответствующих настроек) HTTPS. Разработан как программа с открытым исходным кодом (распространяется в соответствии с GNU GPL). Все запросы выполняет как один неблокируемый процесс ввода/вывода [2].

Озвученная темой доклада программа, на основе лог-файлов прокси-сервера Squid, ведет учет работы пользователя в сети Internet. Из лог-файлов системы приложение извлекает следующие данные:

1) Имена пользователей сети
2) Перечень посещенных ресурсов
3) Объем входящего Internet-трафика

Так как в системе Squid имеется возможность изменять вид лог-файлов, в программе реализована функция работы с файлами разной структуры. Программа ведет отслеживание

изменения файла конфигурации Squid. Если дата изменения текущих настроек Squid не соответствует полученному значению, то необходимо перенастроить механизм обработки лог-файлов, что приложение выполняет автоматически. Лог-файл может иметь огромное количество записей (до нескольких миллионов за один месяц, в зависимости от масштабов предприятия), поэтому считывать лог-файл каждый раз сначала неприемлемо. Проблема решается путем указания в рабочем текстовом файле номера строки, с которой необходимо будет продолжить считывание информации.

Подводя итог, следует отметить, что существуют приложения, выполняющие функции анализа статистики работы пользователей в сети Internet, но особенностью разработанной программы является объединение важнейших достоинств такого рода приложений с исключением всех возможных их недостатков, таких как установка дополнительных библиотек, невозможность гибкой настройки периодов ведения учета, отсутствие возможности анализа статистики по пользователям, ресурсам и прочим параметрам за определенный период времени.

Литература

1. Управленческий учет – Режим доступа: http://ru.wikipedia.org/wiki/Управленческий_учет
2. Squid. – Режим доступа: http://ru.wikipedia.org/wiki/Squid

Е.В. Пугин
Научный руководитель – доцент, канд. техн. наук Р.А. Симаков
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл.,ул. Орловская, д. 23

Проблема подсчёта числа уникальных значений для больших объёмов данных

Часто возникает необходимость подсчёта числа уникальных значений во входном массиве данных. Примером может служить задачи определения количества уникальных посещений некоторого ресурса, уникального числа атакующих IP адресов при DDoS атаках. Также существуют аналогичные задачи в некоторых СУБД [2, 3]. В простейшем случае, если объём данных невелик, возможен подход, при котором уникальные значения просто сохраняются в память ЭВМ. Если число таких элементов велико, то существует вероятность исчерпания системных ресурсов машины. Также при сборе информации о числе уникальных элементов с нескольких датчиков возрастает нагрузка на сетевые компоненты информационной системы, которая расходуется на передачу очередных значений.

Решение данных проблем заключается в использовании приблизительной оценки числа уникальных элементов массива данных. Такая оценка основывается на теории вероятностей. Суть метода заключается в использовании не самих исходных значений, а их хешей, полученных путём использования некоторой хеш-функции. Основным преимуществом такого метода является снижение используемого объёма памяти до 32 КБ – 1 МБ.

Предположим, что имеется 32-битная хеш-функция без коллизий. Тогда вероятность того, что самый правый бит будет установлен в 0 равна 0.5. Вероятность, что 2 крайних бита будут установлены в 0 равна 0.25 и т.д. Следовательно, вероятность того, что i-ый справа бит будет установлен в 1, а все биты правее его в 0 равна 2^{-i}. На основе этой информации можно оценить число уникальных входных элементов, как некоторое число из промежутка 2^i и 2^{i+1}. Для уточнения результата используется некоторый алгоритм коррекции [1].

При малом числе уникальных элементов результат может иметь высокую погрешность из-за того, что хеш некоторого значения может оканчиваться большим числом битов, обращённых в ноль. В этом случае следует хранить хеши элементов массива данных в памяти. При превышении некоторого порога уникальных элементов необходимо осуществить переход от одного алгоритма к другому. На практике такой порог выбирается в промежутке между 4-10 тысячами различных значений.

Естественным является то, что все подобные алгоритмы приближённой оценки числа уникальных элементов дают некоторую погрешность, которая составляет 1-7%. Она зависит от двух факторов:

1. Стойкость хеш-функции к коллизиям. Чем меньше входных элементов будут давать одинаковые хеши, тем точнее будет оценка.
2. Длина генерируемого хеша. Если хеш-функция выдаёт только 32-битные хеши, то при её использовании можно подсчитать максимум 2^{32} уникальных значений. Поэтому следует заранее выбирать хеш-функцию, удовлетворяющую некоторым условиям. Оптимальной длиной хеша будет являться длина в 64 и 128 бита.
3. Объём используемой памяти. Существующие алгоритмы, такие как LogLog [1] и его производные (SuperLogLog, HyperLogLog) зависят от количества используемой памяти. Нормальная величина в данном случае хоть и не велика (64 КБ), но допускает некоторого наращивания. Если требуется получить результат с большей точностью, то в алгоритмах имеется возможность увеличивать данное значение до 128 КБ – 1 МБ и более. Всё зависит от имеющихся в распоряжении машины или пользователя ресурсов.

Таким образом, рассмотренный алгоритм позволяет во много раз сократить расход системных ресурсов ЭВМ и ускорить обработку входных данных. Использование небольшого количества памяти даёт возможность использования этого метода во встраиваемых системах и

датчиках. Одним из применений таких алгоритмов являются распределённые СУБД [3], обрабатывающие сверхбольшие объёмы научных данных.

Литература

1. Durand, M., Flajolet, P.: Loglog counting of large cardinalities. In: Proc. 11th European Symp. on Algorithms. (2003) 605–617
2. Flajolet, P., Martin, G.N.: Probabilistic counting algorithms for data base applications. J. of Computer and System Sciences 31 (1985) 182–209
3. Gibbons, P.B., Tirthapura, S.: Distributed streams algorithms for sliding windows. In: Proc. 14th ACM Symp. on Parallel Algorithms and Architectures. (2002) 63–72

М.А. Смяткин
Научный руководитель – доцент, канд. техн. наук Д.Н. Стародубов
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл.,ул. Орловская, д. 23

Алгоритм группировки с использованием B+ дерева в СУБД Red Database 2.5

СУБД Red Database построена на основе Firebird и имеет ряд расширений. Одно из реализованных на текущий момент расширений – алгоритм группировки данных на основе B+ дерева. Firebird в настоящее время поддерживает только один алгоритм группировки, выполняемый последовательно на заранее отсортированном потоке записей. Это традиционный алгоритм и он достаточно эффективен в случае, когда поля группировки (атрибуты оператора GROUP BY) можно отобразить на имеющиеся индексы или количество групп достаточно велико относительно размера группируемого потока. В последнем случае от накладных расходов на материализацию данных (одновременное представление необходимых данных в памяти) сложно избавится в альтернативных алгоритмах.

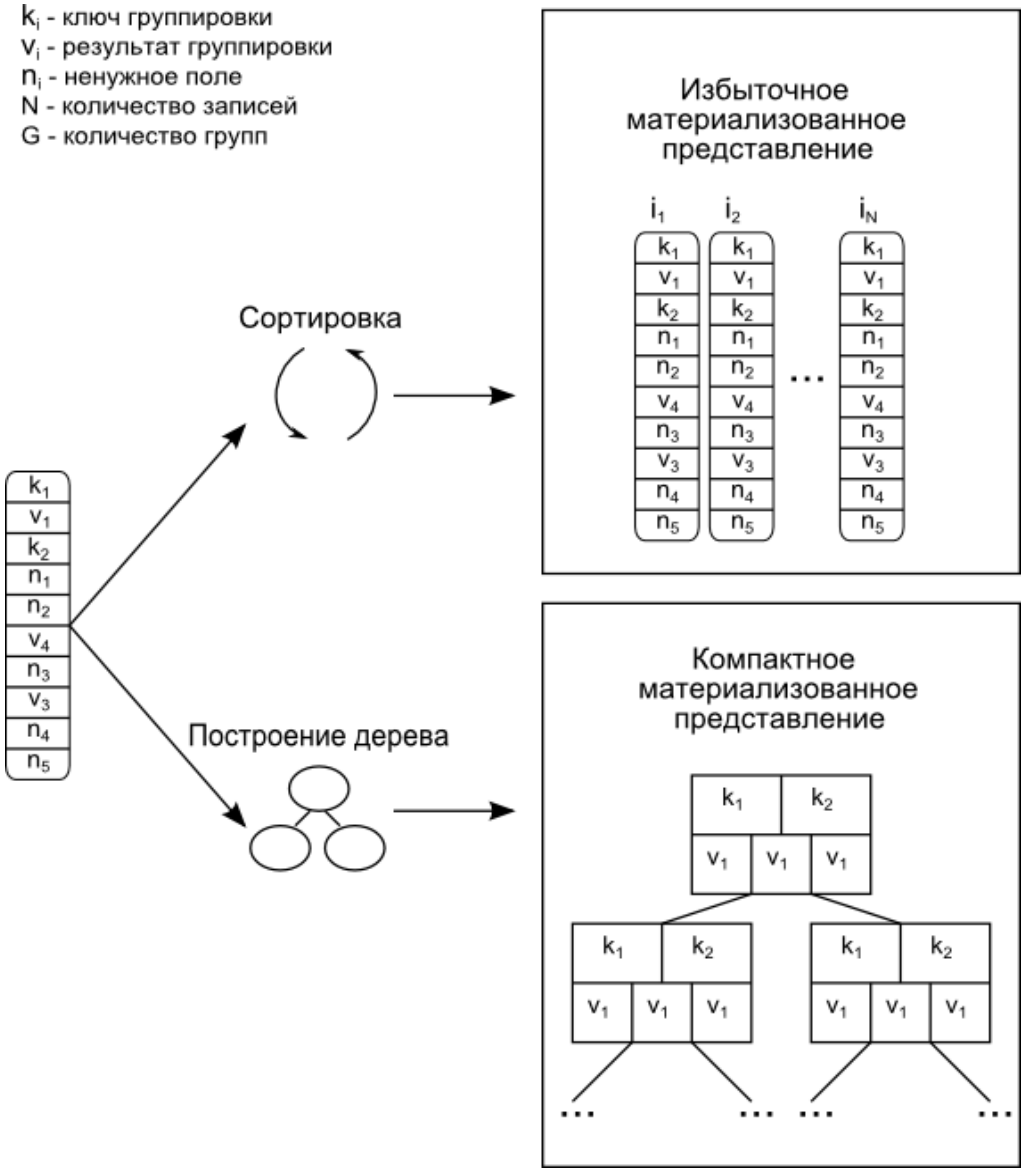

Рис. 1 Различия между группировкой на основе сортировки и группировкой на основе B+ дерева

Однако, если количество групп достаточно мало относительно общего количества записей, то можно применить алгоритм группировки хешированием. Этот алгоритм реализован в таких СУБД, как Oracle, PostgreSQL, Microsoft SQL и др. Для эффективной работы этого алгоритма требуется наличие статистики по распределению данных в полях таблиц, чтобы выбрать хеш-таблицу подходящего размера, иначе расходы на её перераспределение могут значительно превысить выигрыш от использования алгоритма.

Так как в СУБД Red Database (и Firebird) нет подобного рода статистических данных, то реализован альтернативный алгоритм – группировка на основе B+ дерева (Рисунок 1).

Этот алгоритм получает записи с диска, создавая дерево, ключами которого являются поля группировки, а значениями – результаты агрегирования. На рисунке 1 представлен упрощенный случай, когда один агрегат соответствует одному полю таблицы. Как видно из рисунка в случае сортировки материализуется N записей с полным набором полей, тогда как в случае группировки деревом материализации подлежит G <= N записей (как правило, G << N), и ограниченный набор полей, который также зачастую меньше исходного.

Основными преимуществами меньшей материализации являются:

- Меньшее время повторной итерации при выдаче результата (G элементов вместо N);
- Меньшее количество используемой памяти, что снижает вероятность выгрузки части данных на диск, приводящей к чрезвычайно медленному выполнению запросов.

Таким образом, имеющийся алгоритм группировки деревом эффективно можно использовать при соблюдении следующих условий:

- В базе данных отсутствует подходящий набор индексов, на который можно отобразить атрибуты оператора GROUP BY;
- Количество групп G (а соответственно и размер дерева) значительно меньше общего количества записей N, подходящих под условия запроса и/или средний размер элементов дерева будет значительно меньше среднего размера отсортированной записи.

В то время как соблюдение первого условия зависит от администратора баз данных, второе условие часто соблюдается в рабочих нагрузках, в особенности при выполнении аналитических запросов.

Также в заключение следует отметить, что на текущий момент реализация этого алгоритма для distinct-агрегатов отсутствует, так как пока не подтверждена её эффективность, в то время как сложность реализации достаточно высока.

М.А. Смяткин
Научный руководитель – доцент, канд. техн. наук Р.А. Симаков
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл.,ул. Орловская, д. 23
e-mail: smyatkinmaxim@gmail.com

Хранилище распределенной массивно-реляционной СУБД

Хранение и обработка больших объемов научных и аналитических данных требуют значительно более мощных программных средств, чем те, которые доступны на текущий момент. Более того, объемы таких данных постоянно растут. Их характер, как правило, многомерный с преобладающим чтением данных, и на текущий момент уже существует несколько проектов, фокусирующихся на решении этой проблемы [1, 2].

Проект, разрабатываемый в рамках научной работы авторов, также ставит в центре внимания решение обозначенной проблемы. Предпосылки к созданию и архитектура системы описаны в опубликованных ранее статьях [3, 4], а целью этой работы является обоснование и описание наиболее вероятной структуры распределенного хранилища, позволяющего удовлетворить поставленные к СУБД требования.

Существующие системы, предоставляющие обработку аналитических данных, поддерживают какую-то одну определенную модель данных. Существуют различные модели, такие как графовые, хранилища ключ-значение и другие, однако в данной работе особый интерес представляют реляционная модель, обладающая наибольшей популярностью, и многомерная, которая хорошо подходит для анализа больших объемов данных. Оба варианта имеют свои ограничения. При поддержке исключительно многомерной модели представление таких данных, как метаданные, таблицы-справочники, журналы и другие данные, имеющие форму множеств, становится не естественным и может оказаться не эффективным, так как на них, как правило, строятся запросы не аналитического характера. В тоже время поддержка исключительно реляционной модели, позволяет представить в виде множеств как привычные для реляционной модели наборы данных, так и аналитические (многомерные). Существуют специальные методы и алгоритмы представления многомерных данных в виде совокупности реляционных отношений, например [5]. Такой подход также имеет свои недостатки: он не всегда поддается автоматизации и требует дополнительных затрат на соединение отношений для выполнения аналитического запроса.

Для ликвидации этих недостатков в проектируемой системе планируется совместная поддержка обеих моделей. Это является наиболее важным (но не единственным) отличием от имеющихся проектов, которое делает разрабатываемую подсистему хранения уникальной.

В первую очередь в работе рассматриваются альтернативные подходы, позволяющие поддерживать гибридную модель хранения данных: автономная работа обеих моделей, реализация многомерной модели поверх реляционной и реализация реляционной модели поверх многомерной. Каждый подход имеет свои преимущества и недостатки, которые чрезвычайно важно тщательно взвесить перед принятием проектного решения. Под реализацией одной модели поверх другой понимается предоставление пользователю интерфейса обеих моделей, независимо от физического представления, т.е. хранение всех данных в какой-то одной модели, выбранной в зависимости от анализа альтернатив, не исключено.

Кроме того, необходимо рассмотреть более общие вопросы, касающиеся физического хранения данных:

- вертикальное (по столбцам) или горизонтальное (по записям) хранение;
- разреженное представление и сжатие данных;
- фрагментация данных на узлах;
- введение слоя файловой системы, инкапсулирующего отказоустойчивость, репликацию и работу с фрагментами данных;
- файловая структура баз данных;

- хранение статистической информации;
- хранение индексов.

Рассматривая эти вопросы, большое внимание также уделяется другим особенностям системы, которые необходимо учитывать при проектировании подсистемы хранения:

- append-only архитектура;
- стремление исключить централизованные элементы из системы (повышение масштабируемости и отказоустойчивости);
- отказ от глобальных блокировок и попытка уменьшения количества двухфазовых фиксаций/распределенных транзакций;

В общем случае, проектирование подсистемы хранения информации в СУБД является сложным процессом. В случае распределенной СУБД, совмещающей одновременно две модели представления информации, этот процесс усложняется ещё больше, и, несомненно, в ходе работы будут возникать вопросы, требующие более детального изучения и выводы, касающиеся уникальной части разработки.

Литература

1. *M. Stonebreaker, P. Velikov, R. Simakov, A. Smirnov* [и др.]. Overview of SciDB: large scale array storage, processing and analysis. //Proceedings of the ACM SIGMOD International Conference on Management of Data, 06.06.2010-11.06.2010, Indianapolis, IN. P. 963-968.

2. *P. Bauman, A. Dehmel, P. Furtado* [и др.]. The multidimensional database system RasDaMan. // Proceedings of the ACM SIGMOD International Conference on Management of Data, New York, NY, USA, 1998. P. 575 – 577.

3. *М.А. Смяткин, Р.А. Симаков.* Предпосылки создания распределенной массивно-реляционной СУБД. Алгоритмы, методы и системы обработки данных. 2011. №18.

4. *Р.А. Симаков, М.А. Смяткин.* Особенности архитектуры распределенной массивно-реляционной СУБД. Алгоритмы, методы и системы обработки данных. 2011. №18.

5. *M. Varga*. A Procedure of Conversion of Relational into Multidimensional Database Scheme. Journal of Computing and Information Technology. – CIT 10, 2002, pp. 69-84.

Е.А. Соколов
Научный руководитель – доцент, канд. техн. наук Р.А. Симаков
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл., ул. Орловская, д. 23
e-mail: script1990@yandex.ru

Разработка библиотеки операторов обработки изображений для СУБД SciDB

На сегодняшний день информационные системы и базы данных применяются практически в любой области деятельности человека[1]. Хранение и анализ наблюдений в оптической и радиоастрономии, сейсмологии, генетике, океанографии, геологии, климатических и экологических наблюдениях требует хранения больших объемов информации[2]. Существующие СУБД не способны обработать такой большой объем данных. Вследствие этого был открыт проект SciDB по созданию свободной СУБД для использования в области обработки научных данных, полученных в результате экспериментов и наблюдений[2]. В отличие от традиционных реляционных баз данных, моделью данных в SciDB является многомерный массив[3]. Подобная структура обеспечивает компактное хранение данных и высокую производительность при работе с ними.

Одно из возможных вариантов использования СУБД – это хранение изображений больших объемов и обработка их для дальнейшего использования. Поэтому целью данной научной работы стала разработка библиотеки операторов обработки изображений. Создание подобной библиотеки позволит решить ряд задач:

- возможность сохранения изображений из файла большого объема непосредственно в СУБД;
- обработка изображений посредством математических функций, таких как бинаризация, скелетизация, сглаживание и прочих;
- высокая скорость обработки данных;
- выделение необходимых признаков изображений для дальнейшего анализа;
- классификация изображений по ряду признаков;
- сравнение изображений по определенным критериям;
- распознавание образов.

При разработке данной библиотеки необходимо соблюдать ряд требований, которые позволят решить поставленные задачи:

- создание оператора для записи в массив базы данных яркостных характеристик полученного изображения;
- последовательная обработка данных из массива для обеспечения максимальной производительности[4];
- масштабируемость алгоритмов обработки;
- предусмотреть возможность сохранения обработанных цветных и черно-белых изображений в файл.

Разработка подобной библиотеки операторов довольно непростая задача, но ее значение довольно велико, так как на данный момент в проекте SciDB нет возможности производить операции с изображениями. Создание данных операторов даст толчок к дальнейшему развитию СУБД и расширит ее использование для различных информационных систем, поскольку обработка изображений сейчас становиться все более актуальной задачей, тем более когда речь идет о больших объемах информации.

Литература

1. Comparison of different SQL implementations against SQL standards. Includes Oracle, DB2, Microsoft SQL Server, MySQL and PostgreSQL. (08/Jun/2007)
2. http://www.opennet.ru/opennews/art.shtml?num=30983
3. Симаков Р.А., Книжник К., Смирнов А.В.. SciDB - новая СУБД для больших объемов научных данных . Суперкомпьютеры, Москва, No.1(5), 2011, с. 32-34 (ЦП)
4. Симаков Р.А., Домнин Н.А. Моделирование пользовательской нагрузки как метод производительности сервера БД / Алгоритмы, методы и системы обработки данных. 2007. №12

С.П. Фомин
Научный руководитель – доцент, канд. техн. наук Р.А. Симаков
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл., ул. Орловская, д. 23
e-mail: SergeyFomin@f5f5.ru

Разработка и исследование тестов производительности астрономических запросов на языке SCIDB

Разработка и исследование проводилось на основе СУБД SciDB. СУБД специально разрабатывается для обработки, хранения и исследования больших объемов данных [1-2]. Данные представляют собой снимки космического пространства очень высокого разрешения. Снимки космоса занимают большой объем памяти, поэтому обработка этих данных является сложной задачей [4-5].

Исследование основано на двадцати основных астрономических запросах, применяемых к космическим снимкам, запросы разработаны и подробно описаны Джимом Греем [3]. При тестировании использовались запросы, ищущие определенные объекты, например, галактику или объект, похожий на звезду. При поиске объектов можно учитывать такие параметры, как размер объекта, степень размытости, уровень яркости, цвет, угол наклона, размер эллипса. Есть возможность поиска галактики, которая закрыта звездой, поиск квазаров, особо мощное и далёкое активное ядро галактики. Доступен поиск галактик с анимальными линиями излучения спектров, отслеживание перемещения объектов, схожих с астероидами, выявление “белых карликов”.

Для исследования изображения в SciDB был создан 2-мерный безразмерный 3-х атрибутный массив целых чисел, текст запроса: “CREATE ARRAY IM <R:int32,G:int32,B:int32> [x=0:*,1000,0, y=0:*,1000,0];”. Каждый атрибут отвечает за градацию одного из цветов RGB. В массив можно добавить дополнительные атрибуты, например атрибут прозрачности, расстояния до ближайших объектов на изображении. Чтобы загрузить изображение в массив SciDB, его необходимо преобразовать в текстовый формат csv. Для этого была написана программа-конвертор. Далее с помощью утилиты csv2scidb файл преобразуется в текстовый формат, поддерживаемый SciDB. Затем функция load() загружает полученные данные в массив. После этого с массивом можно работать, для выборки данных из массива используется язык AQL (Array Query Language). Синтаксис языка AQL схож с SQL.

Пример запроса к массиву данных: “AQL% SELECT R FROM IM WHERE R between 50 and 60;”. Здесь отбираются все элементы изображения со значением синей градации яркости в диапазоне от 50 до 60. Время выполнения запроса 10мс. По аналогичному принципу построены остальные запросы.

При выполнении каждого из астрономических запросов на разных снимках была получена информация о времени выполнения.

Литература

1. *M. Stonebreaker, P. Velikov, R. Simakov, A. Smirnov* [и др.]. Overview of SciDB: large scale array storage, processing and analysis. //Proceedings of the ACM SIGMOD International Conference on Management of Data, 06.06.2010-11.06.2010, Indianapolis, IN. P. 963-968.
2. *P. Bauman, A. Dehmel, P. Furtado* [и др.]. The multidimensional database system RasDaMan. // Proceedings of the ACM SIGMOD International Conference on Management of Data, New York, NY,
3. *Jim Gray, Don Slutz.* Data Mining the SDSS SkyServer Database, USA, 1998. P. 575 – 577.
4. *А.В. Смирнов.* Проблема оптимизации запросов в расширяемой СУБД SciDB. Алгоритмы, методы и системы обработки данных. 2011. №18.
5. М.А. Смяткин, Р.А. Симаков. Предпосылки создания распределенной массивно-реляционной СУБД. Алгоритмы, методы и системы обработки данных. 2011. №18.