Министерство образования Российской Федерации
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет информатики
Кафедра прикладной информатики

УДК 681.03

ДОПУСТИТЬ К ЗАЩИТЕ В ГАК
Зав. кафедрой, проф., д.т.н.
_____ С.П.Сущенко
«___» _____ 2010 г.

Исмаилов Евгений Ринатович

РАЗРАБОТКА ПОДСИСТЕМЫ СТАТИСТИКИ ДЛЯ ВЫСОКОНАГРУЖЕННОГО ПРОЕКТА WWW.MOJIVA.COM

Дипломная работа

Научный руководитель С. А. Цой

Исполнитель, Е. Р. Исмаилов
студ. гр. 1431

Электронная версия дипломной работы помещена
в электронную библиотеку. Файл
Администратор

Томск – 2010

Реферат

Дипломная работа 37 с., 6 рис., 6 источников, 2 прил.

БАЗА ДАННЫХ, OLTP, OLAP, MAP/REDUCE, WEB-ПРИЛОЖЕНИЕ

Объект исследования – разработка подсистемы работы со статистикой высоконагруженного web-приложения.

Цель работы – реализация подсистемы статистики, обеспечивающей хранение данных и возможность получения статистической информации по одному или нескольким критериям.

Методы исследования – моделирование, эксперимент.

Результат работы – разработана подсистема работы со статистикой кликов и запросов к приложению мобильной рекламы www.mojiva.com.

Содержание

Введение

Проект mojiva.com представляет собой площадку для мобильной рекламы, где рекламодатели (адвертайзеры) размещают свои объявления, которые, впоследствии, показываются на сайтах, зарегистрированных их владельцами (паблишерами). Многие из сайтов паблишеров являются очень посещаемыми и на каждый запрос пользователя их сайта, передается запрос и на серверную часть нашего проекта. Таким образом, хотя количество пользователей интерфейсной части mojiva.com не является большим, этим пользователям необходима статистическая информация о происходящем в системе. Так, например, адвертайзерам необходимо знать, сколько они потратили на рекламную кампанию, в каких странах и для пользователей каких провайдеров мобильной связи были показана их реклама и т.д. В конечном счете - насколько удачно позиционированы их объявления (то есть, настроен таргетинг). Аналогичная информация нужна и для паблишеров.

С момента запуска проекта поток данных, по которому необходимо строить статистические выкладки, постоянно рос. Сейчас серверная часть приложения обслуживает порядка 250-300 миллионов запросов в сутки. Изменялись и требования пользователей в плане разнообразия запросов к системе, что увеличивало ее сложность.

В рамках данной работы решаются следующие задачи:

1. Описывается реализация подсистемы статистики и ее изменение, в соответствии с изменениями потребностей пользователей.
2. Проводится обзор существующих OLAP-решений.
3. Осуществляется выработка требований и применение этих требований к реализации.
4. Реализация выбранного решения на базе OLAP-системы Pentaho Mondrian
5. Реализация пользовательского интерфейса
6. Описываются планы эволюции решения в связи с ожидаемым увеличением нагрузки.

1. Современные подходы к обработке данных

1.1. Системы OLTP

OLTP (Online Transaction Processing) — обработка транзакций в реальном времени. Способ организации базы данных, при котором система работает с небольшими по размерам транзакциями, но идущими большим потоком, и при этом клиенту требуется от системы максимально быстрое время ответа.

Термин OLTP применяют также к системам (приложениям). OLTP-системы предназначены для ввода, структурированного хранения и обработки информации (операций, документов) в режиме реального времени.

OLTP-приложениями охватывается широкий спектр задач во многих отраслях — банковские и биржевые операции, в промышленности — регистрация прохождения детали на конвейере, фиксация в статистике посещений очередного посетителя веб-сайта, автоматизация бухгалтерского, складского учёта и учёта документов и т. п. Приложения OLTP, как правило, автоматизируют структурированные, повторяющиеся задачи обработки данных, такие как ввод заказов и банковские транзакции. OLTP-системы проектируются, настраиваются и оптимизируются для выполнения максимального количества транзакций за короткие промежутки времени. Как правило, большой гибкости здесь не требуется, и чаще всего используется фиксированный набор надёжных и безопасных методов ввода, модификации, удаления данных и выпуска оперативной отчётности. Показателем эффективности является количество транзакций, выполняемых за секунду. Обычно аналитические возможности OLTP-систем сильно ограничены (либо вообще отсутствуют).

OLTP-системы оптимизированы для небольших дискретных транзакций. А вот запросы на некую комплексную информацию (к примеру, поквартальная динамика объемов продаж по определённой модели товара в определённом филиале), характерные для аналитических приложений (OLAP), породят сложные соединения таблиц и просмотр таблиц целиком. На один такой запрос уйдет масса времени и компьютерных ресурсов, что затормозит обработку текущих транзакций.

1.2. Системы OLAP

1.2.1 Общий обзор OLAP-систем

OLAP (англ. online analytical processing, аналитическая обработка в реальном времени) — технология обработки информации, включающая составление и динамическую публикацию отчётов и документов. Используется аналитиками для быстрой обработки сложных запросов к базе данных. Служит для подготовки бизнес-отчётов по продажам, маркетингу, в целях управления, т. н. data mining — добыча данных (способ анализа информации в базе данных с целью отыскания аномалий и трендов без выяснения смыслового значения записей).

Основоположник термина OLAP, Эдгар Кодд, предложил в 1993 году «12 законов аналитической обработки в реальном времени». Найджел Пендс переформулировал 12 правил Кодда в более ёмкий тест FASMI (Fast Shared Multidimensional Information). По его определению, OLAP система должна быть:

- Fast – быстрой, обеспечивать почти мгновенный отклик на большинство запросов;
- Shared – многопользовательской, должен существовать механизм контроля доступа к данным и возможность одновременной работы многих пользователей;
- Multidimensional – многомерной. Данные должны представляться в виде многомерных кубов;
- Information – данные должны быть полны с точки зрения аналитика, т.е.содержать всю необходимую информацию.

Причина использования OLAP для обработки запросов — это скорость. Реляционные БД хранят сущности в отдельных таблицах, которые обычно хорошо нормализованы. Эта структура удобна для операционных БД (системы OLTP), но сложные многотабличные запросы в ней выполняются относительно медленно. Хорошей моделью для запросов, а не для изменения, является Пространственная БД.

OLAP делает мгновенный снимок реляционной БД и структурирует её в пространственную модель для запросов. Заявленное время обработки запросов в OLAP составляет около 0.1 % от аналогичных запросов в реляционную БД.

OLAP-структура, созданная из рабочих данных, называется OLAP-куб (рис 1). Куб создаётся из соединения таблиц с применением схемы звезды или схемы снежинки. В центре схемы звезда находится таблица фактов, которая содержит ключевые факты, по которым

делаются запросы. Множественные таблицы с измерениями присоединены к таблице фактов. Эти таблицы показывают, как могут анализироваться агрегированные реляционные данные. Количество возможных агрегирований определяется количеством способов, которыми первоначальные данные могут быть иерархически отображены.

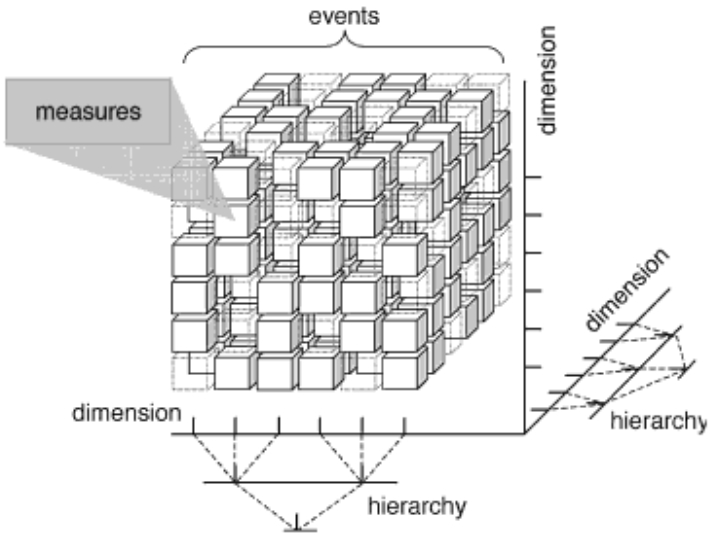

Рисунок 1. Визуальное представление OLAP-куба

Например, все клиенты могут быть сгруппированы по городам или по регионам страны (Запад, Восток, Север и т. д.), таким образом, 50 городов, 8 регионов и 2 страны составят 3 уровня иерархии с 60 членами. Также клиенты могут быть объединены по отношению к продукции; если существуют 250 продуктов по 2 категориям, 3 группы продукции и 3 производственных подразделения, то количество агрегатов составит 16560. При добавлении измерений в схему, количество возможных вариантов быстро достигает десятков миллионов и более.

OLAP-куб содержит в себе базовые данные и информацию об измерениях (агрегатах). Куб потенциально содержит всю информацию, которая может потребоваться для ответов на любые запросы. Из-за громадного количества агрегатов, зачастую полный расчёт происходит только для некоторых измерений, для остальных же производится «по требованию».

Вместе с базовой концепцией существуют три типа OLAP — OLAP со многими измерениями (Multidimensional OLAP — MOLAP), реляционный OLAP (Relational OLAP — ROLAP) и гибридный OLAP (Hybrid OLAP — HOLAP). MOLAP — это классическая форма

OLAP, так что её часто называют просто OLAP. Она использует суммирующую БД, специальный вариант процессора пространственных БД и создаёт требуемую пространственную схему данных с сохранением как базовых данных, так и агрегатов. ROLAP работает напрямую с реляционным хранилищем, факты и таблицы с измерениями хранятся в реляционных таблицах, и для хранения агрегатов создаются дополнительные реляционные таблицы. HOLAP использует реляционные таблицы для хранения базовых данных и многомерные таблицы для агрегатов.

Каждый тип хранения имеет определённые преимущества, хотя есть разногласия в их оценке у разных производителей. MOLAP лучше всего подходит для небольших наборов данных, он быстро рассчитывает агрегаты и возвращает ответы, но при этом генерируются огромные объёмы данных. ROLAP оценивается как более масштабируемое решение, использующее к тому же наименьшее возможное пространство. При этом скорость обработки относительно снижается. HOLAP находится посреди этих двух подходов, он достаточно хорошо масштабируется и быстро обрабатывается. Архитектура R-ROLAP позволяет производить многомерный анализ OLTP-данных в режиме реального времени.

Сложность в применении OLAP состоит в создании запросов, выборе базовых данных и разработке схемы, в результате чего большинство современных продуктов OLAP поставляются вместе с огромным количеством предварительно настроенных запросов. Другая проблема — в базовых данных. Они должны быть полными и непротиворечивыми.

Для того чтобы получить информацию из OLAP-системы используется специальный язык запросов MDX (Multidimensional Expressions). Его назначение - предоставить в распоряжение разработчиков средство для более простого и эффективного доступа к многомерным структурам данных. На этом языке можно указать срез куба с помощью измерений (например двумерный - т.е. таблицу), и те меры, которые нас интересуют.

1.2.2 ROLAP, Relational OLAP – реляционный OLAP

В реляционных OLAP-системах структура куба данных хранится в реляционной базе данных. Меры самого нижнего уровня остаются в реляционной витрине данных (таблице фактов), служащей источником данных для куба. Предварительно обработанные агрегаты также хранятся в реляционной таблице.

Когда человек, принимающий решение, запрашивает значение меры для определенного набора элементов измерения, ROLAP-система проверяет, указывают ли эти элементы на агрегат или на значение самого нижнего уровня иерархии (листовое значение). Если указан агрегат, то значение выбирается из реляционной таблицы. Если выбрано листовое значение, то значение берется из витрины данных.

Благодаря реляционным таблицам, архитектура ROLAP позволяет хранить большие объемы данных. Поскольку в архитектуре ROLAP листовые значения берутся непосредственно из витрины данных, то возвращаемые ROLAP-системой листовые значения всегда будут соответствовать актуальному на данный момент положению дел. Другими словами, ROLAP-системы лишены запаздывания в части листовых данных.

Достоинства этого класса систем:

- возможность использования ROLAP с хранилищами данных и различными OLTP-системами;
- возможность манипулирования большими объемами данных; объем данных могут ограничивать только лежащие в основе ROLAP системы реляционных баз данных, подход ROLAP сам по себе не ограничивает объем данных;
- безопасность и администрирование обеспечивается реляционными СУБД.

Недостатки:

- получение агрегатов и листовых данных происходит медленнее, чем, например, в MOLAP и HOLAP (см. ниже);
- функциональность систем ограничивается возможностями SQL, так как аналитические запросы пользователя транслируются в SQL-операторы выборки;
- сложно пересчитывать агрегированные значения при изменениях начальных данных;

сложно поддерживать таблицы агрегатов.

Представители: Пионером ROLAP был продукт Metaphor компании Metaphor Computer Systems, появившийся в 80-х годах. Также выделим DSS Suite фирмы MicroStrategy, MetaCube фирмы IBM Informix, Platinum Beacon от Platinum, Brio, Business Objects, DecisionSuite компании Information Advantage. На современном этапе развития ROLAP отметим Mondrian, JasperAnalysis, MicroStrategy 9, Tableau Software, Cognos Powerplay, Microsoft Analysis Services.

1.2.3 MOLAP, Multidimensional OLAP – многомерный OLAP

В многомерных OLAP-системах структура куба хранится в многомерной базе данных. В той же базе данных хранятся предварительно обработанные агрегаты и копии листовых значений. В связи с этим все запросы к данным удовлетворяются многомерной системой баз данных, что делает MOLAP-системы исключительно быстрыми.

Для загрузки MOLAP-системы требуется дополнительное время на копирование в многомерную базу всех листовых данных. Поэтому возникают ситуации, когда листовые данные MOLAP-системы оказываются рассинхронизированными с данными в витрине данных. Таким образом, MOLAP-системы вносят запаздывание в данные нижнего уровня иерархии.

Архитектура MOLAP требует большего объема дискового пространства из-за хранения в многомерной базе копий листовых данных. Но, несмотря на это, объем дополнительного пространства обычно не слишком велик, поскольку данные в MOLAP хранятся исключительно эффективно.

Достоинства MOLAP-систем:

- все данные хранятся в многомерных структурах, что существенно повышает скорость обработки запросов;
- доступны расширенные библиотеки для сложных функций оперативного анализа;
- обработка разреженных данных выполняется лучше, чем в ROLAP.

Недостатки:

- данные куба «оторваны» от базовой таблицы; необходимы специальные инструменты для формирования кубов и их пересчёта в случае изменения базовых значений;
- сложно изменять измерения без повторной агрегации.

Представители: Cognos Powerplay, Oracle OLAP Option, Oracle Essbase, Microsoft Analysis Services, TM1, Palo, IdeaSoft O3.

1.2.4 HOLAP, Hybrid OLAP – гибридный OLAP

В гибридных OLAP сочетаются черты ROLAP и MOLAP, отсюда и название – гибридный. В моделях HOLAP используются преимущества и минимизируются недостатки обеих архитектур.

В HOLAP-системах структура куба и предварительно обработанные агрегаты хранятся в многомерной базе данных. Это позволяет обеспечить быстрое извлечение агрегатов из структур MOLAP. Значения нижнего уровня иерархии в HOLAP остаются в реляционной витрине данных, которая служит источником данных для куба.

HOLAP не требует копирования листовых данных из витрины, хотя это и ведет к увеличению времени доступа при обращении к листовым данным. Данные в витрине доступны аналитику сразу после обновления. Таким образом, HOLAP-системы не вносят запаздывания в работу с данными нижнего уровня иерархии. По сути, HOLAP жертвует скоростью доступа к листовым данным ради устранения запаздывания при работе с ними и ускорения загрузки данных. В связи с этим HOLAP проигрывает по скорости MOLAP.

К достоинствам подхода можно отнести комбинирование технологии ROLAP для разреженных данных и MOLAP для плотных областей, а к недостаткам – необходимость поддерживания MOLAP и ROLAP.

Представители: Microsoft Analysis Services, MicroStrategy, IBM DB2 OLAP Server, Sagent Holos.

1.3. Использование технологий OLAP/OLTP в проекте mojiva.com

Как и большинство веб-приложений пользовательская часть mojiva.com является OLTP-системой, т.е. обрабатывает действия пользователей транзакционно. Примерами таких действий могут быть:

- Создание рекламной кампании
- Регистрация сайта в системе
- Добавление нового пользователя и т.д.

В качестве базы данных используется СУБД PostgreSQL.

Со временем стало очевидно, что наличие статистической информации в той же базе данных, где хранятся реляционные модели предметной области, приводит к значительному

замедлению работы системы. В связи с этим, а также с тем, что пользователям понадобились новые типы отчетов, было принято решение вынести подсистему статистики как отдельное приложение на выделенный сервер.

В настоящее время подсистема статистики основана на широко используемом ROLAP-решении Pentaho Mondrian.

2. Эволюция подсистемы статистики

2.1. Статистика биллинг-данных

На начальном этапе подсистема статистики основывалась на данных, генерируемых сервером обработки запросов в виде информации биллинга. Эта информация включала в себя данные о:

1. Кликах
2. Показах
3. Доходах паблишеров
4. Расходах адвертайзеров
5. Рекламных кампаниях
6. Сайтах

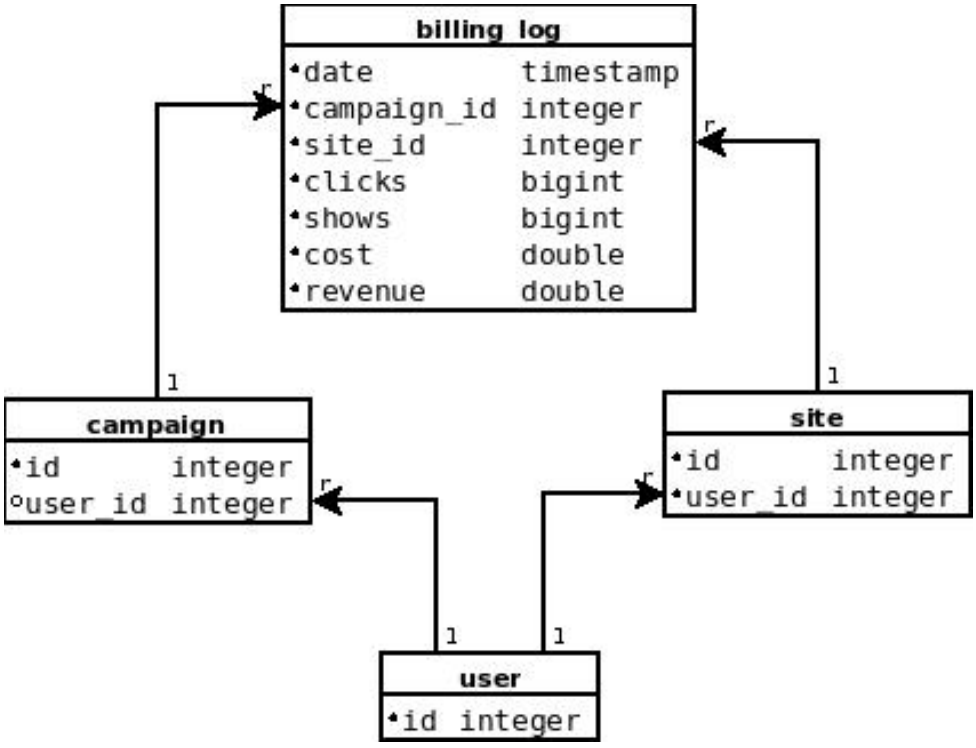

Рисунок 2. Статистика биллинга

В таблицу billing_log данные сбрасывались с периодичностью в несколько минут и затем, агрегировались с помощью sql-запроса в более сжатые таблицы. Для запросов к системе использовались как раз эти сжатые таблицы.

Постепенно количество данных увеличивалось многократно, и использование этого подхода стало затруднительно с точки зрения производительности БД. Было принято решение построить иерархию таблиц биллинга с разделением по времени создания записи.

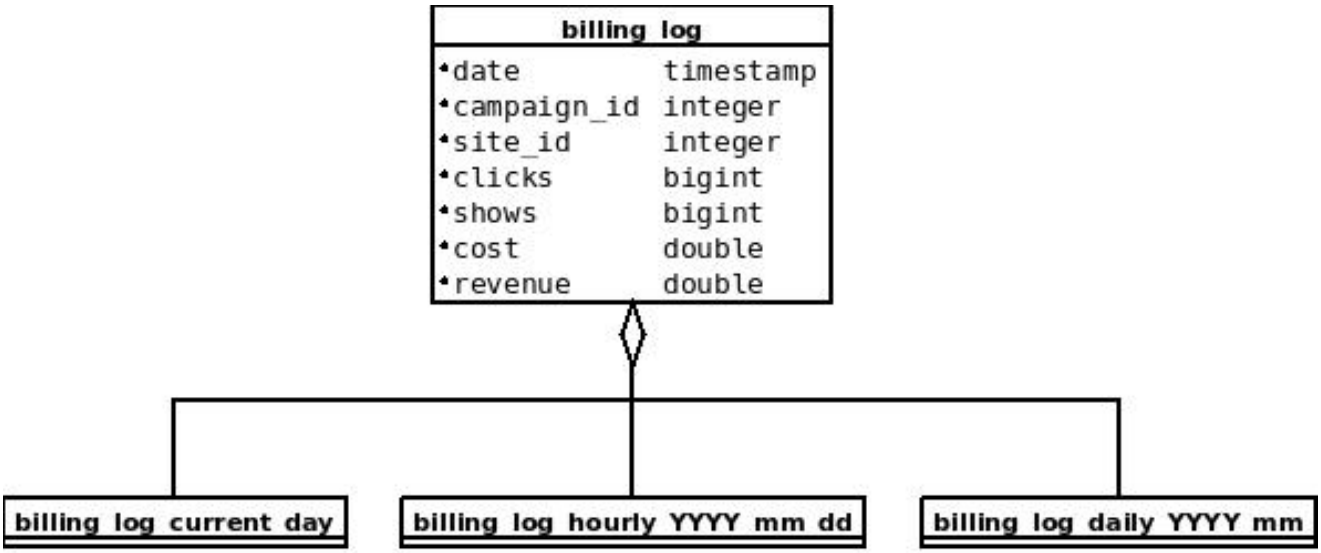

Рисунок 3. Иерархия таблиц биллинга

В данной реализации таблица billing_log используется как базовая и в ней ничего не хранится. От нее, средствами PostgreSQL наследуются таблицы, имеющие идентичную структуру, но с разной степенью детализации по дате.

1. billing_log_current_day - данные за текущий день; складываются, как только на сервер начинает использовать больше определенного лимита памяти (обычно раз в секунду)
2. billing_log_hourly - данные по часам. Таких таблиц семь. Т.е. мы храним и выдаем пользователям данные с почасовой детализацией не далее чем за прошедшую неделю.
3. billing_log_daily - данные по дням за прошедшие месяцы. На каждый месяц одна таблица.

Поскольку таблица иерархична, СУБД PostgreSQL при SQL-запросе определяет какие из подтаблиц необходимо использовать. Так, например, если спросить данные недельной давности, то задействуется только таблица billing_log_daily, что существенно экономит ресурсы сервера.

Таким образом, ежедневно объем данных в системе в биллинг-статистике увеличивается не более чем на количество данных, агрегированных за один день (обычно не более нескольких мегабайт).

2.2. Гео-статистические данные

Геостатистические данные представляют собой две таблицы:

1. Страна/регион + id сайта.
2. Стран/регион + id рекламной кампании.

Первоначально эти данные агрегировались внутри бекэнда и записывались с определенной частотой в БД. При увеличении нагрузки этот подход оказался неприемлемым, т.к. либо не справлялся сервер приложения, либо сервер БД. В связи с этим было принято решение сохранять данные о кликах и показах и производить заполнение гео-статистических таблиц раз в сутки несколькими агрегирующими sql-запросами.

2.3. Хранение логов кликов и показов

Для хранения информации о кликах и показах, произведенных пользователями на сайтах паблишеров, используется выделенный сервер БД PostgreSQL. В текущее время система обрабатывает примерно 250 миллионов запросов в сутки. При этом занимаемое место при хранении этих данных составляет порядка 100Гб/сутки. Непосредственно в базе данных хранятся логи последних нескольких дней. Предыдущие дни хранятся в сжатых файлах и занимают около 5Гб/сутки.

2.4. Таргетинг-статистики

2.4.1. Описание статистических данных по таргетингам

Наряду с гео-статистическими данными пользователей также интересуют данные по моделям телефонов, с которых был произведен запрос, провайдерам мобильной связи, был ли сделан запрос из приложения или с мобильного браузера. Поскольку социальные сети предоставляют интерфейс для разных языков программирования, приложения, созданные на их основе и использующие mojiva.com могут осуществлять предельно точное позиционирование рекламы, передавая такие параметры как возраст, пол, количество друзей и т.д. Все эти данные являются очень полезными как для рекламодателей, так и для владельцев сайтов. Таким

образом, для предоставления такой возможности было реализовано заполнение таблиц таргетинг-статистик.

Таблицы таргетинг-статистик для адвертайзера состоят из мер (кликов/показов/расходов) и ключа (таргетинг + id кампании). Для паблишера эти таблицы выглядят аналогично, за исключением того, что в качестве ключа используется пара (таргетинг + id сайта). Таргетингом может быть провайдер, id телефона, страна, регион, и т.д.

При использовании произвольных таргетингов, стало невозможно производить агрегацию используя sql-запросы. Также в системе стали возникать ситуации, при которых происходило расхождение группированных статистик и биллинга. Помимо этого, все таргетинг-статистики было решено перенести на отдельный сервер. Для решения возникших проблем была реализована подсистема агрегации.

2.4.2. Подсистема агрегации

Подсистема агрегации представляет собой библиотеку, реализованную на языке PHP, предоставляющую возможность агрегации таргетинг-статистик при одновременной синхронизацией с данными биллинга.

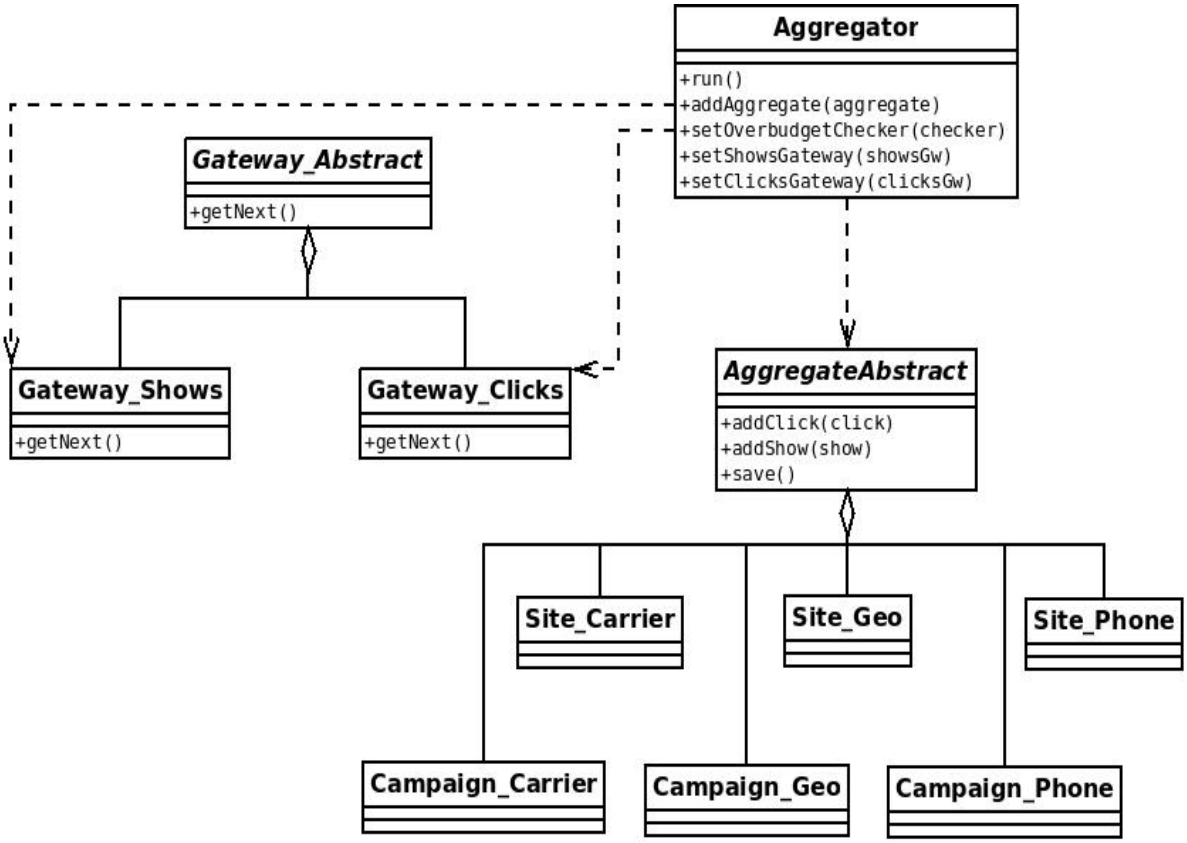

Рисунок 4. Диаграмма классов подсистемы агрегации

Классы Gateway_Shows и Gateway_Clicks представляют собой интерфейс к таблицам clicks, shows в которых хранятся данные логов кликов и показов. Эти таблицы очень большого объема, и использовать запросы к базе данных напрямую в невозможно, так как select * from shows вытаскивает данные из этой таблицы целиком в память. При использовании ограничений типа limit/offset скорость выполнения запросов к таблице резко снижается. Опытным путем было выяснено, что наиболее производительным способом является открывание транзакции и использование курсора над select * from shows при вытаскивании в память кусков по 250-300 тысяч строк. Таким образом, вышеупомянутые классы являются реализацией этого подхода.

Подклассы AggregateAbstract реализуют механизм агрегации таргетинг-статистик в памяти с последующим сохранением данных на диск.

В классе Aggregator регистрируются те подклассы AggregateAbstract, ассоциированные таблицы которых необходимо заполнить.

После запуска метода run() класса Aggregator весь процесс происходит следующим образом:

1. Из clicksGateway последовательно читаются клики и передаются каждому из зарегистрированных подклассов AggregateAbstract. Перед передачей клик синхронизируется с биллингом.
2. Аналогичный процесс для показов.
3. После обработки всех логов за один день у всех подклассов AggregateAbstract вызывается метод save(), который инициализирует запись накопленных в памяти таргетинг-статистик в таблицы БД.

3. Задачи, стоявшие при реализации подсистемы статистики

3.1. Формулировка требований

При реализации подсистемы статистики стояло несколько задач.

Первой задачей было предоставление пользователям возможность производить не только запросы по двумерным таргетинг-статистикам, но, также, задавать любые наборы фильтров. Так, в новой системе пользователь может одновременно задать несколько фильтрующих полей и несколько целевых полей. В качестве примера можно привести запрос типа “распределение запросов от телефона Apple iPhone по странам, с фильтром по нескольким сайтам”. В данном случае фильтрами являются iPhone и сайты, а целевым полем - страны. Также можно узнать каким моделям телефонов отдают предпочтения пользователи, живущие в США. Здесь фильтр - США, целевое поле - модели телефонов.

Второй, не менее важной задачей, стояло обеспечение высокой производительности при работе пользователей с системой. Проблема низкой производительности была одной из основных причин, по которой пришлось отказаться от старой подсистемы статистики. Реализовывать новое решение, напрямую используя реляционную БД, представлялось нецелесообразным.

Третьей задачей стояло предоставление комфортной работы пользователей системы в многопользовательском режиме. Т.е. возможность производить запросы к системе одновременно без существенного замедления производительности.

Все эти задачи легко выполнимы на небольших объемах данных, однако, поскольку количество запросов уже сейчас достаточно велико и продолжает расти, все заданные задачи достаточно проблемны. К тому же построенное решение должно обеспечивать незначительное снижение производительности при увеличении объемов данных.

Исходя из спецификации указанных задач, и их сходства с правилами FASMI (Fast Shared Multidimensional Information), сформулированными Найджелом Пендсом для систем аналитической обработки в реальном времени, было принято решение подобрать решение из систем этого класса.

3.2. Реализации OLAP-систем

Существует несколько реализаций OLAP-систем.

Таблица 1. Сравнительная таблица популярных OLAP-решений

OLAP сервер	Вендор	ROLAP	HOLAP	MOLAP	Лицензия ПО
Microsoft Analysis Services	Microsoft	Да	Да	Да	Проприетарное ПО
Essbase	Oracle	Да	Да	Да	Проприетарное ПО
TM1	IBM	Нет	Нет	Да	Проприетарное ПО
Mondrian OLAP server	Pentaho	Да	Нет	Нет	EPL
Palo	Jedox	Нет	Нет	Да	GPL v2
Oracle OLAP Option	Oracle	Да	Да	Да	Проприетарное ПО
Microstrategy OLAP Services	Microstrategy	Да	Нет	Да	Проприетарное ПО
SAS OLAP Server	SAS Institute	Да	Да	Да	Проприетарное ПО

Среди коммерческих решений наиболее привлекательным является решение от Oracle, поскольку оно может быть установлено на linux-сервера, которые есть у нас в наличии и позволяет осуществить выбор технологии хранения куба (ROLAP/HOLAP/MOLAP).

Однако решение от Oracle является очень дорогостоящим, и было принято решение опробовать свободно-распространяемый OLAP-сервер, и, при успешном внедрении в проект, возможно, перейти на коммерческое решение, предоставляющее больше опций.

Среди существующих свободных OLAP-серверов единственно подходящим по типу лицензии является Pentaho Mondrian. Помимо свободного распространения это решение также

применяется в популярном коммерческом продукте Pentaho Buisness Inteligence,таким образом, доказывая свою состоятельность в плане использовании в Enterprise-окружении.

4. ROLAP-решение на базе Pentaho Mondrian

4.1. Описание сервера Pentaho Mondrian

С точки зрения пользователя OLAP-система Pentaho Mondrian состоит из четырех уровней:

1. Уровень представления (the presentation layer)
2. Уровень схемы (the dimensional layer)
3. Уровень кэширования (the star layer)
4. Уровень хранения (and the storage layer)

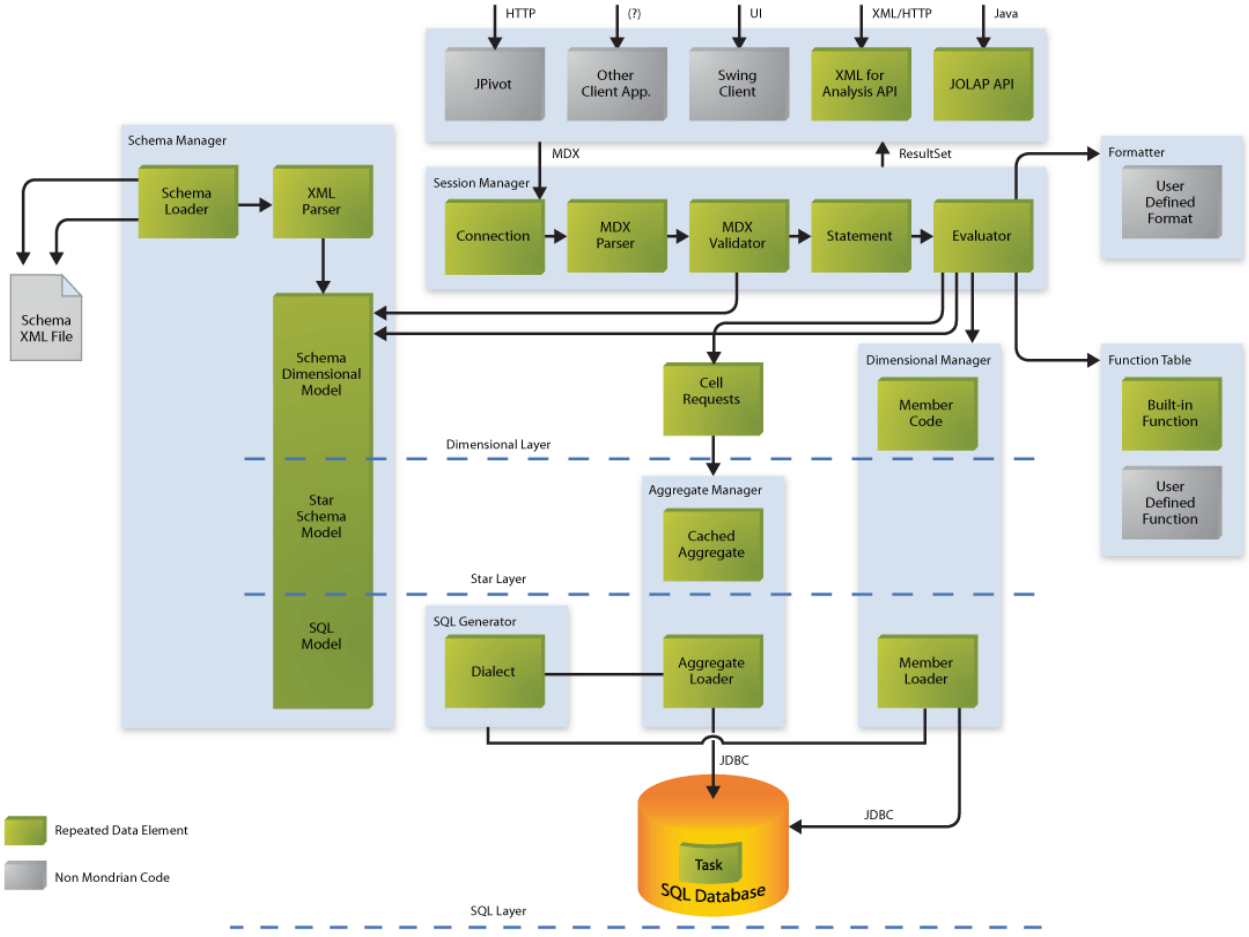

Рисунок 5. Структура OLAP-сервера Pentaho Mondrian

Уровень представления определяет, что конечный пользователь видит на своем мониторе. Существует множество способов представления многомерных структур данных, таких как гистограммы, графики, круговые диаграммы, линейчатые диаграммы и т.д. Все эти представления могут быть реализованы доступными программными средствами. В случае нашего проекта, была выбрана реализация, основанная на flash-библиотеке FusionChart, позволяющей рисовать интерактивные диаграммы.

Уровень схемы занимается разбором, проверкой и исполнением MDX-запросов к системе на основе заданной схемы OLAP-куба и агрегационных таблиц. В результате разбора запроса строится описания набора ячеек OLAP-куба, которые необходимо извлечь. Затем преобразованный запрос передается уровню кэширования.

Уровень кэширования проверяет, есть ли у него в памяти уже извлеченные ячейки OLAP-куба, по которым производится запрос. В случае отсутствия производится запрос к уровню хранения, но только для тех ячеек, которые еще не были помещены в кэш.

Уровень хранения выбирает агрегационную таблицу, заданную в схеме, с минимальным количеством данных в ней и с полями, достаточными для выполнения запроса. Если такой агрегационной таблицы не найдено, то для выборки используется базовая таблица фактов. Для выбранной таблицы строится и выполняется SQL-запрос к СУБД, выбранной в качестве среды хранения, используя диалект этой СУБД. В реализованной системе используется PosqtgreSQL, позволяющая осуществлять хранение очень большого объема данных и наследование таблиц, позволяющего разделить большие таблицы и за счет этого уменьшить нагрузку.

Pentaho Mondrian предоставляет доступ в виде SOAP-протокола XML/A (XML for Analysis), который используют почти все OLAP-решения. Этот протокол позволяет передавать MDX-запросы в систему и получать XML-ответ в соответствии со спецификацией.

4.2. Синтаксис и структура MDX-запроса

MDX (MultiDimentional eXpressions) представляет собой SQL-подобный язык, предназначенный для осуществления запросов к многомерным данным.

```
SELECT {[Measures].[Unit Sales]} on columns,
   TopCount([Product].[Brand].members,
     Parameter("Count", NUMERIC, 10, "Number of brands to show"),
     (Parameter("Region", [Store], [Store].[USA].[CA]),
```

```
        [Measures].[Unit Sales])) on rows
FROM Sales
```

Листинг 1. Пример MDX-запроса.

Приведенный выше запрос осуществляет выборку данных по продажам для 10 наиболее популярных производителей в магазинах, расположенных в Калифорнии.

4.3. OLAP-схема подсистемы статистики mojiva.com

Схема подсистемы статистики описывается в XML-формате, определенном спецификацией Pentaho Mondrian.

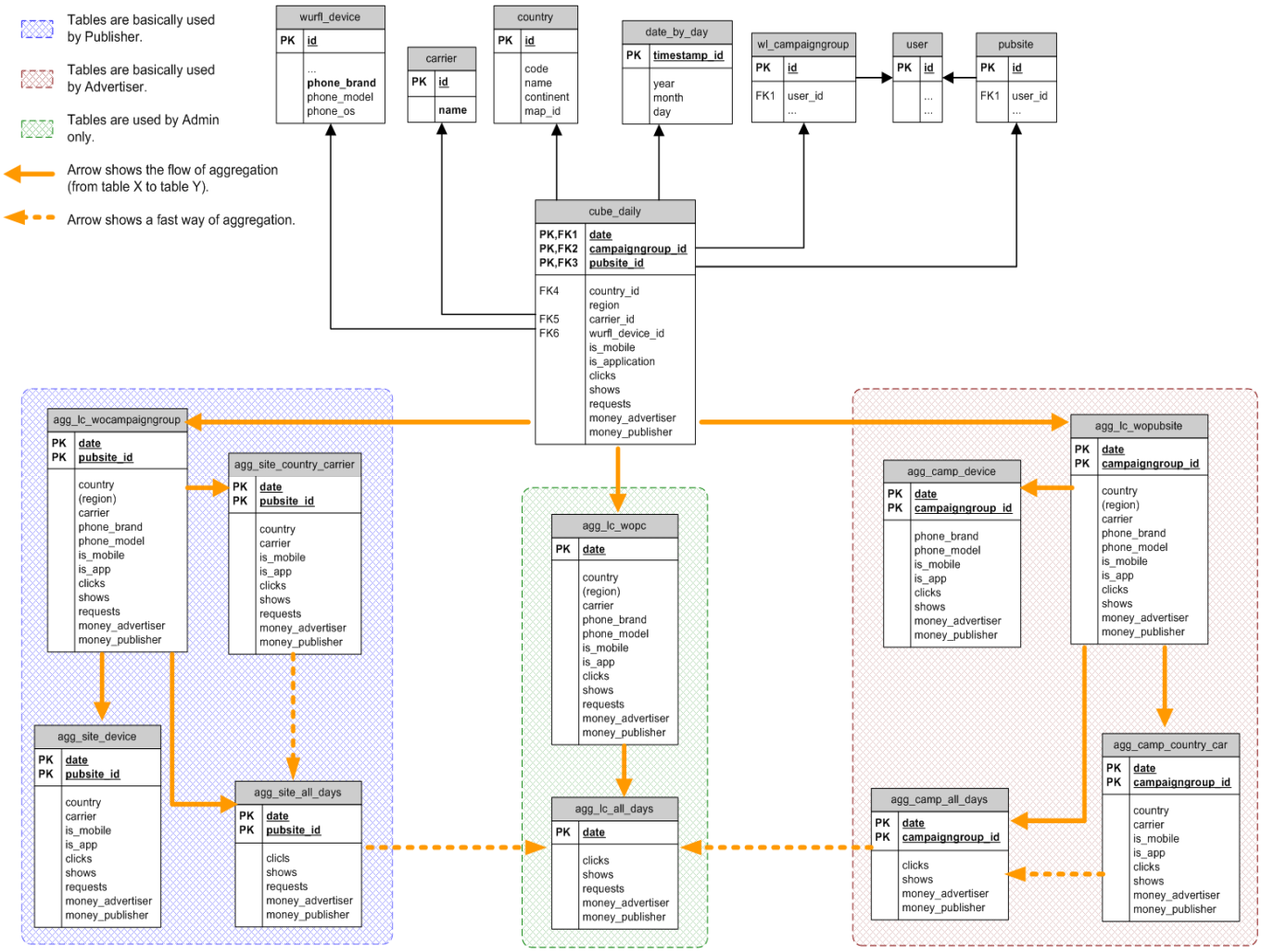

Рисунок 6. OLAP-схема подсистемы статистики

В схеме указывается базовая таблица фактов (cube_daily). В этой таблице находятся данные о кликах и показах сгруппированные по ключу (дата + кампания + сайт + страна + регион + устройство + провайдер). В качестве целевых значений (мер) выбраны : количество кликов, количество показов, потраченные на кампанию деньги, заработанные сайтом деньги.

Для каждого поля из ключа таблицы cube_daily существует соответствующая таблица измерений (dimension table).

Агрегационные таблицы могут исключать некоторые поля таблицы фактов, а могут включать необходимые поля из таблиц измерений. Это позволяет системе не использовать таблицу фактов, в которой обычно больше всего данных, а использовать ту, в которой данных меньше всего, но полей достаточно для выполнения запроса. Включение полей из таблиц измерений позволяет избавиться от дополнительных JOINов в запросах, однако увеличивает объем хранимых данных.

Полная схема находится в приложении А.

4.4. Импорт данных в OLAP-систему

Импорт данных в систему осуществляется аналогично процессу, описанному в п.2.4.2. Только вместо классов, агрегирующих тагретинг-статистики, используется класс для агрегации базовой таблицы фактов cube_daily. Затем, с помощью SQL-запросов происходит сборка агрегационных таблиц. Порядок сборки указан на Рисунке 6 стрелками. Сначала собираются таблицы с наименьшим числом редуцированных полей, а потом уже из них собираются остальные.

4.5. Библиотека PHP-классов для взаимодействия с Mondrian

Для того, чтобы использовать упрощенный интерфейс к OLAP-серверу, была реализована библиотека. Эта библиотека представляет собой набор классов на языке PHP и позволяет составлять запросы без прямого использования синтаксиса MDX, что позволяет разработчикам не заниматься изучением его синтаксиса.

4.6. Пользовательский интерфейс analytics.mojiva.com

Поскольку было принято решение предоставить доступ к аналитической информации всем зарегистрированным пользователям системы, был реализован графический интерфейс в виде современного веб-приложения.

Этот интерфейс позволяет задавать любые комбинации фильтров и просматривать результаты запросов по определенному набору целевых измерений.

Все манипуляции происходят без перезагрузки страницы. Возможно строить месячные и дневные отчеты, осуществлять с отчетами операции сохранения/удаления/обновления.

Внешний вид интерфейса представлен в приложении Б.

5.Проблемы и перспективы

5.1. Проблемы

1. Линейная зависимость скорости обработки от объемов данных. Скорость обработки суточных данных сейчас порядка 12 часов. При увеличении нагрузки в два раза, будет невозможно агрегировать собранные данные о кликах и показах за сутки. Т.е. скорость накопления данных может превысить скорость их обработки.
2. Низкая отказоустойчивость. При выходе из строя сервера хранения логов, новые данные некуда будет записывать и они просто потеряются.

5.2. Перспективы

1. Использование распределенного хранилища (Hadoop, Cassandra) обеспечивает масштабирование и отказоустойчивость хранения данных.
2. Использование map-reduce парадигмы для агрегации данных в analytics.mojiva.com обеспечивает прозрачное распараллеливание процесса обработки данных, что снимает линейную зависимость от их объема.

   Предложенные подходы решают исходные проблемы.

.

Заключение

В рамках данной работы были решены следующие задачи:

- Описана реализация подсистемы статистики и ее изменение, в соответствии с изменениями потребностей пользователей.
- Произведен обзор существующих OLAP-решений.
- Осуществлена выработка требований и применение этих требований к реализации.
- Выбранное решение реализовано на базе OLAP-системы Pentaho Mondrian
- Реализован пользовательский интерфейса, позволяющий делать запросы к OLAP-серверу без необходимости изучения синтаксиса MDX-запросов.

В связи с постоянным увеличением нагрузки на систему, была начата реализация функциональности агрегации с использованием распределенных хранилищ данных и применения подхода map-reduce, обеспечивающего прозрачное распараллеливание процесса.

Описанная система продемонстрировала свою состоятельность и успешно применяется на протяжении нескольких месяцев.

Список использованных источников

1. Андреев А.Н. Классификация OLAP-систем вида xOLAP // Рязанский государственный радиотехнический университет

2. Гамма Э., Хэлм Р., Дженсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб: Питер, 2006. – 366 с.

3.Кудрявцев Ю.А. OLAP технологии: обзор решаемых задач и исследований // Бизнес-информатика. – 2008. №1. – С. 66-70.

4. Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. Proceedings of the Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December, 2004

5. Pentaho Mondrian Project [Электронный ресурс]: Сайт – Режим доступа: http://mondrian.pentaho.com, свободный.

6. Wikipedia [Электронный ресурс]: Сайт – Режим доступа: http://en.wikipedia.org, свободный.

Приложение А. OLAP-схема подсистемы статистики

```
<Schema name="Analytics Schema">
<Cube name="Analytics" cache="true" enabled="true">
      <Table name="cube_daily" schema="public">
         <AggName name="agg_lc_wopc_cube_daily">
         <AggFactCount column="cube_daily_fact_count"/>
         <AggForeignKey factColumn="wurfl_device_id"
      aggColumn="wurfl_device_Phone_model" />
         <AggMeasure    column="cube_daily_Sum_cost"    name="[Measures].[Sum
cost]">
         </AggMeasure>
         <AggMeasure  column="cube_daily_Sum_revenue"  name="[Measures].[Sum
revenue]">
         </AggMeasure>
         <AggMeasure   column="cube_daily_Sum_clicks"   name="[Measures].[Sum
clicks]">
         </AggMeasure>
         <AggMeasure  column="cube_daily_Sum_requests"  name="[Measures].[Sum
requests]">
         </AggMeasure>
         <AggMeasure   column="cube_daily_Sum_shows"   name="[Measures].[Sum
shows]">
         </AggMeasure>
         <AggMeasure   column="cube_daily_Sum_unique"   name="[Measures].[Sum
visitors]">
          </AggMeasure>
          <AggLevel column="country_Country" name="[Country].[Country]">
          </AggLevel>
          <AggLevel column="carrier_Carrier" name="[Carrier].[Carrier]">
          </AggLevel>
          <AggLevel column="date_by_day_Year" name="[Time].[Year]">
          </AggLevel>
          <AggLevel column="date_by_day_Month" name="[Time].[Month]">
          </AggLevel>
```

```
        <AggLevel column="date_by_day_Day" name="[Time].[Day]">
        </AggLevel>
        <AggLevel                          column="true_false_AppVsMobile"
name="[AppVsMobile].[AppVsMobile]">
        </AggLevel>
        <AggLevel                          column="true_false_MobileVsWeb"
name="[MobileVsWeb].[MobileVsWeb]">
              </AggLevel>
       </AggName>
 <!-- описание остальных агрегационных таблиц-->
...
</Table>
<!-- измерения -->
<Dimension          type="StandardDimension"          foreignKey="country"
highCardinality="false" name="Country">
     <Hierarchy hasAll="true" allMemberName="All" primaryKey="code">
          <Table name="country" schema="public"></Table>
          <Level   approxRowCount="249"   name="Country"   table="country"
column="code"   type="String"   uniqueMembers="true"   levelType="Regular"
hideMemberIf="Never"></Level>
     </Hierarchy>
</Dimension>
<Dimension          type="StandardDimension"          foreignKey="carrier"
highCardinality="false" name="Carrier">
     <Hierarchy hasAll="true" allMemberName="All" primaryKey="id">
          <Table name="carrier" schema="public"></Table>
               <Level   approxRowCount="215"   name="Carrier"   column="id"
type="Numeric"          uniqueMembers="true"          levelType="Regular"
hideMemberIf="Never">
               </Level>
     </Hierarchy>
</Dimension>
<Dimension      type="StandardDimension"      foreignKey="wurfl_device_id"
highCardinality="false" name="Phone">
     <Hierarchy hasAll="true" allMemberName="All Brands" primaryKey="id">
     <Table name="wurfl_device_existing" schema="public">  </Table>
```

```
      <Level     name="Phone     brand"     column="brand"     type="String"
uniqueMembers="false" levelType="Regular" hideMemberIf="Never">
      </Level>
      <Level     name="Phone     model"     column="id"     type="Numeric"
uniqueMembers="true" levelType="Regular" hideMemberIf="Never">
      </Level>
      </Hierarchy>
</Dimension>
<Dimension      type="StandardDimension"      foreignKey="wurfl_device_id"
highCardinality="false" name="OS">
      <Hierarchy hasAll="true" allMemberName="All OS" primaryKey="id">
      <Table name="wurfl_device_existing" schema="public"></Table>
      <Level name="Phone os" column="os" type="String" uniqueMembers="true"
levelType="Regular" hideMemberIf="Never">
      </Level>
      </Hierarchy>
</Dimension>
<Dimension          foreignKey="pubsite_id"          highCardinality="false"
name="Publisher">
      <Hierarchy hasAll="true" allMemberName="All" primaryKey="id">
            <Table name="user"></Table>
            <Level     approxRowCount="17000"     name="User"     column="id"
type="Numeric"           uniqueMembers="true"           levelType="Regular"
hideMemberIf="Never">
            </Level>
      </Hierarchy>
</Dimension>
<Dimension foreignKey="pubsite_id" highCardinality="false" name="Pubsites">
      <Hierarchy hasAll="true" allMemberName="All" primaryKey="id">
            <Table name="pubsite"></Table>
            <Level   approxRowCount="9000"   name="Pubsite   id"   column="id"
type="Numeric"           uniqueMembers="true"           levelType="Regular"
hideMemberIf="Never">
            </Level>
      </Hierarchy>
</Dimension>
```

```
<Dimension foreignKey="campaigngroup_id" highCardinality="false"
name="Advertiser">
     <Hierarchy hasAll="true" allMemberName="All" primaryKey="id">
          <Table name="user">  </Table>
          <Level approxRowCount="17000" name="User" column="id"
type="Numeric" uniqueMembers="true" levelType="Regular"
hideMemberIf="Never">
          </Level>
     </Hierarchy>
</Dimension>
     <Dimension foreignKey="campaigngroup_id" highCardinality="false"
name="Campaigns">
     <Hierarchy hasAll="true" allMemberName="All" primaryKey="id">
          <Table name="wl_campaigngroup"></Table>
          <Level approxRowCount="12000" name="Campaigngroup id"
column="id" type="Numeric" uniqueMembers="true" levelType="Regular"
hideMemberIf="Never">
          </Level>
     </Hierarchy>
</Dimension>
<Dimension type="TimeDimension" foreignKey="date" highCardinality="false"
name="Time">
     <Hierarchy hasAll="true" allMemberName="All" primaryKey="date">
     <Table name="date_by_day"></Table>
     <Level name="Year" column="year" type="Numeric" uniqueMembers="true"
levelType="TimeYears" hideMemberIf="Never">
     </Level>
     <Level name="Month" column="month_of_year" type="Numeric"
uniqueMembers="false" levelType="TimeMonths" hideMemberIf="Never">
     </Level>
     <Level name="Day" column="day_of_month" type="Numeric"
uniqueMembers="false" levelType="TimeDays" hideMemberIf="Never">
     </Level>
</Hierarchy>
</Dimension>
```

```
<Dimension foreignKey="is_application" highCardinality="false"
name="AppVsMobile">
     <Hierarchy hasAll="true" allMemberName="All" primaryKey="id">
          <Table name="true_false"></Table>
          <Level approxRowCount="2" name="AppVsMobile" column="value"
type="Boolean" uniqueMembers="true" levelType="Regular"
hideMemberIf="Never">
          </Level>
     </Hierarchy>
</Dimension>
<Dimension foreignKey="is_mobile" highCardinality="false"
name="MobileVsWeb">
     <Hierarchy hasAll="true" allMemberName="All" primaryKey="id">
          <Table name="true_false"></Table>
          <Level approxRowCount="2" name="MobileVsWeb" column="value"
type="Boolean" uniqueMembers="true" levelType="Regular"
hideMemberIf="Never">
          </Level>
     </Hierarchy>
</Dimension>

<!-- меры -->
<Measure name="Sum cost" column="money_advertiser" datatype="Numeric"
aggregator="sum" visible="true">
</Measure>
<Measure name="Sum revenue" column="money_publisher" datatype="Numeric"
aggregator="sum" visible="true">
</Measure>
<Measure name="Sum clicks" column="clicks" datatype="Integer"
aggregator="sum" visible="true">
</Measure>
<Measure name="Sum requests" column="shows" datatype="Integer"
aggregator="sum" visible="true">
</Measure>
<Measure name="Sum visitors" column="unique_visitors" datatype="Integer"
aggregator="sum" visible="true">
```

```
</Measure>
<Measure name="Sum shows" datatype="Integer" aggregator="sum">
      <MeasureExpression>
            <SQL dialect="generic">(CASE WHEN (campaigngroup_id != 0) THEN
shows ELSE 0 END)</SQL>
      </MeasureExpression>
</Measure>
</Cube>
</Schema>
```

Приложение Б. Интерфейс analytics.mojiva.com