СОДЕРЖАНИЕ

ВВЕДЕНИЕ3

1 Методы и средства интеллектуального анализа данных5

1.1 Основные свойства полезных знаний5

1.2 Краткий обзор методов и задач Data Mining6

1.3 Визуализация Data Mining7

2 Использование методов Data Mining в билетной платформе BIL2412

2.1 Описание и архитектура платформы BIL2412

2.1.1 Термины и определения принятые в курсовой работе12

2.1.2 Назначение платформы BIL2414

2.1.3 Архитектура платформы15

2.1.4 Преимущества системы16

2.2 Actian NoSQL Object Database (ANOD)18

2.3 Порядок решения задач, связанных с интеллектуальным анализом данных BIL2418

3. Решение задач по выявлению аномалий при взаимодействии центрального сервера BIL24 и интерфейсов агентов20

3.1 Задача по устранению ошибок и оптимизации работы интерфейсов BIL24 по «Протоколу билетной системы»20

3.2 Задача по выявлению простоев внешних билетных систем и интернет-эквайрингов, используемых платформой BIL2426

ЗАКЛЮЧЕНИЕ31

Список используемых источников литературы33

Приложение А34

Приложение Б35

ВВЕДЕНИЕ

Из-за огромного количества информации очень малая её часть будет когда-либо увидена человеческим глазом. Единственная надежда понять и найти что-то полезное в этом океане информации — широкое применение методов Data Mining.

Технология Data Mining изучает процесс нахождения новых, действительных и потенциально полезных знаний в базах данных. Data Mining лежит на пересечении нескольких наук, главные из которых — это системы баз данных, статистика и искусственный интеллект.

*Data Mining* — исследование и обнаружение "машиной" (алгоритмами, средствами искусственного интеллекта) в сырых данных скрытых знаний, которые ранее не были известны, нетривиальны, практически полезны, доступны для интерпретации человеком.

В системах электронного бизнеса, где особую важность имеют вопросы привлечения и удержания клиентов, технологии Data Mining часто применяются для построения рекомендательных систем интернет-магазинов и для решения проблемы персонализации посетителей Web-сайтов. Рекомендации товаров и услуг, построенные на основе закономерностей в покупках клиентов, обладают огромной убеждающей силой. Статистика показывает, что почти каждый посетитель магазина Amazon не упускает возможности посмотреть на то, что же купили "Customers who bought this book also bought: ... " ("Те, кто купил эту книгу, также купили ..."). Персонализация клиентов или, другими словами, автоматическое распознание принадлежности клиента к определенной целевой аудитории позволяет компании проводить более гибкую маркетинговую политику.

Поскольку в электронной коммерции деньги и платежные системы тоже электронные, то важной задачей становится обеспечением безопасности при операциях с пластиковыми карточками. Data Mining позволяет обнаруживать случаи мошенничества (fraud detection). В области электронной коммерции также остаются справедливыми все методологии Data Mining, разработанные для обычного маркетинга. С другой стороны, эта область тесно связана с понятием Web Mining.

Для успешного продвижения товаров Data Mining позволяет решать задачи выделения групп потребителей со схожими стереотипами поведения, т. е. сегментировать рынок. Для этого можно применять такие технологии Data Mining, как кластеризация и классификация. Сиквенциальный анализ помогает торговым предприятиям принимать решения о создании товарных запасов. Он дает ответы на вопросы типа "Если сегодня покупатель приобрел видеокамеру, то через какое время он вероятнее всего купит новые батарейки и пленку?".

В силу приведенных аргументов, применение методов Data Mining в сфере электронной коммерции является важной и актуальной задачей, позволяющей повысить эффективность и увеличить доход от этого вида деятельности. В этой работе мы рассмотрим методологию и применение Data Mining на практике, исследуем сырые данные лог файлов билетной платформы BIL24, проанализируем результаты этой работы, опишем создание алгоритмов и модулей интеллектуального анализа данных для билетной платформы BIL24.

1. Методы и средства интеллектуального анализа данных 
   1. Основные свойства полезных знаний

Свойства обнаруживаемых знаний должны удовлетворять следующим требованиям:

а) Знания должны быть новые, ранее неизвестные. Затраченные усилия на открытие знаний, которые уже известны пользователю, не окупаются.

Поэтому ценность представляют именно новые, ранее неизвестные знания.

б) Знания должны быть нетривиальны. Результаты анализа должны отражать неочевидные, неожиданные закономерности в данных, составляющие так называемые скрытые знания.

в) Знания должны быть практически полезны. Найденные знания должны быть применимы, в том числе и на новых данных, с достаточно высокой степенью достоверности. Полезность заключается в том, чтобы эти знания могли принести определенную выгоду при их применении.

г) Знания должны быть доступны для понимания человеку. Найденные

закономерности должны быть логически объяснимы, в противном случае

существует вероятность, что они являются случайными. Кроме того, обнаруженные знания должны быть представлены в понятном для человека виде.

В Data Mining для представления полученных знаний служат модели. Виды моделей зависят от методов их создания. Наиболее распространенными являются: правила, деревья решений, кластеры и математические функции.

* 1. Краткий обзор методов и задач Data Mining

Методы Data Mining помогают решить многие задачи, с которыми сталкивается аналитик. Из них основными являются: обнаружение аномалий, классификация, кластеризация, регрессия, поиск ассоциативных правил и обобщение. Перечисленные задачи можно определить следующим образом.

Задача обнаружения аномалий идентифицирует предметы, события или наблюдения, которые не соответствуют ожидаемому шаблону или другим элементам в различных наборах данных. Задача применима в обнаружении неисправностей, мошенничества, при мониторинге работоспособности систем и т.д. Часто используется в предварительной обработке данных для удаления аномальных элементов из общего набора.

Задача классификации сводится к определению класса объекта по его характеристикам. Необходимо заметить, что в этой задаче множество классов, к которым может быть отнесен объект, известно заранее.

Задача кластеризации заключается в поиске независимых групп (кластеров) и их характеристик во всем множестве анализируемых данных. Решение этой задачи помогает лучше понять данные. Кроме того, группировка однородных объектов позволяет сократить их число, а, следовательно, и облегчить анализ.

Задача регрессии подобно задаче классификации позволяет определить по известным характеристикам объекта значение некоторого его параметра. В отличие от задачи классификации значением параметра является не конечное множество классов, а множество действительных чисел.

При поиске ассоциативных правил целью является нахождение частых зависимостей (или ассоциаций) между объектами или событиями. Найденные зависимости представляются в виде правил и могут быть использованы как для лучшего понимания природы анализируемых данных, так и для предсказания появления событий.

Обобщение обеспечивает компактное представление набора данных, включая визуализацию и генерацию отчетов.

Перечисленные задачи по назначению делятся на описательные и предсказательные.

Описательные (descriptive) задачи уделяют внимание улучшению понимания анализируемых данных. Ключевой момент в таких моделях — легкость и прозрачность результатов для восприятия человеком. К такому виду задач относятся кластеризация и поиск ассоциативных правил.

Решение предсказательных (predictive) задач разбивается на два этапа. На первом этапе на основании набора данных с известными результатами строится модель. На втором этапе она используется для предсказания результатов на основании новых наборов данных. К данному виду задач относят задачи классификации и регрессии. Сюда можно отнести и задачу поиска ассоциативных правил, если результаты её решения могут быть использованы для предсказания появления некоторых событий.

1.3. Визуализация в Data Mining

С возрастанием количества накапливаемых данных, даже при использовании сколь угодно мощных и разносторонних алгоритмов Data Mining, становится все сложнее "переваривать" и интерпретировать полученные результаты. А, как известно, одно из положений Data Mining - поиск практически полезных закономерностей. Закономерность может стать практически полезной, только если её можно осмыслить и понять.

В 1987 году по инициативе ACM SIGGRAPH IEEE Computer Society Technical Committee of Computer Graphics, в связи с необходимостью использования новых методов, средств и технологий данных, были сформулированы соответствующие задачи направления визуализации.

К способам визуального или графического представления данных относят графики, диаграммы, таблицы, отчеты, списки, структурные схемы, карты и т.д.

Визуализация традиционно рассматривалась как вспомогательное средство при анализе данных, однако сейчас все больше исследований говорит о её самостоятельной роли.

Традиционные методы визуализации могут находить следующее применение:

1) представлять пользователю информацию в наглядном виде;

2) компактно описывать закономерности, присущие исходному набору данных;

3) снижать размерность или сжимать информацию;

4) восстанавливать пробелы в наборе данных;

5) находить шумы и выбросы в наборе данных.

C помощью визуализации данных можно вывести закономерности и сделать наблюдения, которые не столь очевидны при рассмотрении обычной статистики. На данный момент существует огромное количество ПО и онлайн-ресурсов, которые могут в этом помочь.

Некоторые примеры ПО для визуализации данных:

1) «Data Driven Documents» – D3.js, (рисунок 1) – первое, что приходит на ум, когда мы думаем о визуализации данных. D3.js использует HTML, CSS и SVG для графиков и диаграмм. Если разработчик может представить какую-либо визуализацию, D3 поможет воплотить её в жизнь. Многофункциональный инструмент интерактивный и красиво выполненный, с бесплатными функциями (по большей части) и открытым исходным кодом. Готовые диаграммы не встроены по умолчанию, однако можно воспользоваться [галереей](https://github.com/mbostock/d3/wiki/Gallery), демонстрирующей основные возможности D3.

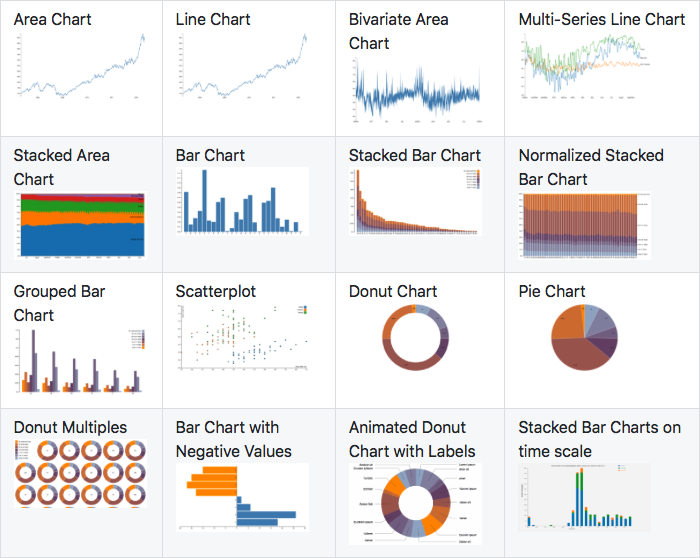


Рисунок 1 – Пример галереи графиков в D3.js

2) Google Charts (рисунок 2) создаёт диаграммы в HTML5/SVG, чтобы обеспечить кроссбраузерную совместимость и переносимость на iPhones и Android. Приложениепредлагает порядочное количество диаграмм с наиболее часто используемыми типами диаграмм (штриховая, комбинированная, круговая и т.д.)

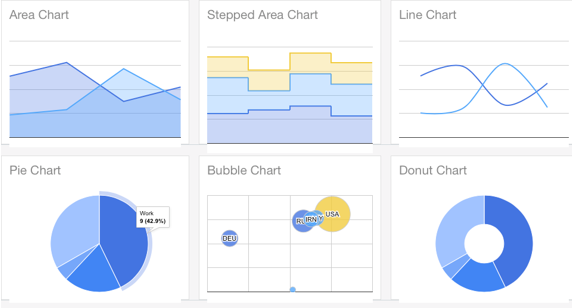


Рисунок 2 – Пример галереи графиков в Google Charts

3) RawGraphs (рисунок 3) характеризует себя как «недостающее звено между электронными таблицами и векторной графикой». Этот инструмент разработан на основе D3.js, имеет интуитивный интерфейс, который даёт возможность быстро привыкнуть к его использованию. У RawGraphs открытый исходный код и он не требует никакой регистрации. Пользователям предлагается библиотека и 16 типов диаграмм для выбора. RawGraphs легко настраиваемая и расширяемая система. Она позволяет создавать новые макеты диаграмм и графиков.



Рисунок 3 – Пример галереи графиков в Raw Graphs

2. Использование методов Data Mining в билетной платформе BIL24

2.1 Описание и архитектура платформы BIL24

2.1.1 Термины и определения, принятые в курсовой работе

Билетная платформа BIL24 (БП) – комплекс программных и аппаратных средств, предназначенных для организации доступа на развлекательные мероприятия (Представления) с помощью продажи билетов на них через широкую агентскую сеть. БП реализована на программной платформе JAVA и использует объектную СУБД промышленного уровня.

Билет – электронный документ, формируемый БП и предоставляющий Покупателю право посетить Представление и занять определенное место в зале. Билет содержит всю необходимую информацию о Представлении, о выделенном Покупателю месте и стоимости услуги.

Заказ – один или несколько билетов, выбранных покупателем для единовременной оплаты. Один заказ может содержать билеты на разные Представления. Сформированный и переданный на оплату Заказ нельзя изменить.

Покупатель - физическое или юридическое лицо, которое приобрело право (билеты) посетить определенный Сеанс.

Оператор – роль пользователей БП, которые являются сотрудниками компании, эксплуатирующей и управляющей БП. Пользователи с этой ролью обладают максимальными правами в системе.

Агент– роль пользователей БП - сотрудников компании, являющейся билетным Агентом. Агент реализует билеты через собственные Интерфейсы (сайты, мобильные приложения, кассы), отчитывается перед Организатором о собственных продажах.

Центральный сервер (ЦС) – центральный сервер билетной системы обслуживающий все клиентские приложения и взаимодействующий с СУБД. Для надежности, ядро системы может быть построено на двух Центральных серверах, между СУБД которых организована журнализация транзакций в режиме реального времени.

Интерфейс – интерфейс (FRONTEND) между покупателем и БП. Каждый FRONTEND принадлежит Агенту и имеет уникальный идентификатор FRONTEND ID (FID) и соответствующий ему токен. Эти данные используются для взаимодействия интерфейса и ЦС по Протоколу билетной системы. В основном Интерфейсами являются вебсайты, мобильные приложения, кассы. Все билеты в системе продаются через один из имеющихся интерфейсов.

ВБС - Внешняя Билетная Система, с которой БП может взаимодействовать через шлюз. Шлюз реализует протокол или API, представленные ВБС. Общей задачей взаимодействия с ВБС является возможность выбирать и резервировать места, продавать и возвращать билеты, источником которых является ВБС.

Протокол билетной системы (ПБС) – протокол обмена данными между Интерфейсами и Центральным сервером билетной системы. Протокол реализован в формате JSON и в формате Java Serialization API.

ОМП – Основное Мобильное Приложение билетной системы предназначено для продажи билетов Покупателям через мобильные устройства с операционными системами семейств Android и IOS. При оплате билетов используется интернет-эквайринг банка.

Редактор БП – приложение, используемое пользователями с ролями Оператор и Организатор для создания Представлений и вывода их в продажу.

Отчеты БП (Reporter) – приложение, используемое пользователями с ролями Оператор, Организатор и Агент для получения информации о проданных и возвращенных билетах, о пропуске Покупателей в зал через Систему Контроля Доступа (СКД). Данная информация может быть представлена в виде отчетов различной структуры в формате Microsoft Excel.

Менеджер БП – приложение, используемое пользователями с ролями Оператор, Организатор и Агент для управления учетными записями их сотрудников в БП, а также для управления Интерфейсами.

МСКД – Мобильная Система Контроля Доступа является мобильным приложением для Android и использует возможности мобильных устройств для проверки криптозащищенных QR-кодов проданных билетов на входе в зал. Так же может работать в режиме сканирования штрихкодов.

МРМК – Мобильное Рабочее Место Кассира является дополнительным режимом работы ОМП для Android. После перевода ОМП в режим МРМК кассир с мобильным устройством может продавать билеты за наличный расчет.

2.1.2 Назначение платформы BIL24

Билетная платформа предназначена для реализации билетов на зрелищные мероприятия через различные каналы продаж. Система ориентирована как на покупателей, активно использующих для покупки электронные каналы продажи билетов (сайты, мобильные приложения, социальные сети), так и на обеспечение продаж через традиционные билетные кассы. С использованием МРМК билеты могут реализовываться через новые каналы продаж, например, через туристические агентства, отели, сеть реализаторов в публичных местах.

BIL24 использует концепцию «единого билетного поля», в которой все билеты постоянно доступны всем Агентам, их продажа происходит без выдачи и возврата квот. В то же время система поддерживает режим работы с квотами, получаемых и возвращаемых по накладным.

Также Платформа предназначена для автоматизации работы контролеров на входе Представления с помощью Мобильной Системы Контроля Доступа (МСКД). Использование МСКД и размещенных на билетах криптозащищенных QR-кодов сводит на нет попытки подделки билетов и других характерных злоупотреблений.

В силу использования высокоскоростной объектной NoSQL СУБД промышленного уровня BIL24 в состоянии не только накапливать и хранить большие объемы данных, но и предоставлять к ним доступ в режиме реального времени. Хранение данных в том же виде, в котором они используются в оперативной памяти, предоставляет широкие возможности. Система отчетности БП позволяет в режиме реального времени получать и анализировать большие объемы статистической информации, создавать и получать отчеты различной формы.

2.1.3 Архитектура системы

Билетная платформа построена на единственном ядре и высокоскоростной объектной NoSQL СУБД (рисунок 4).

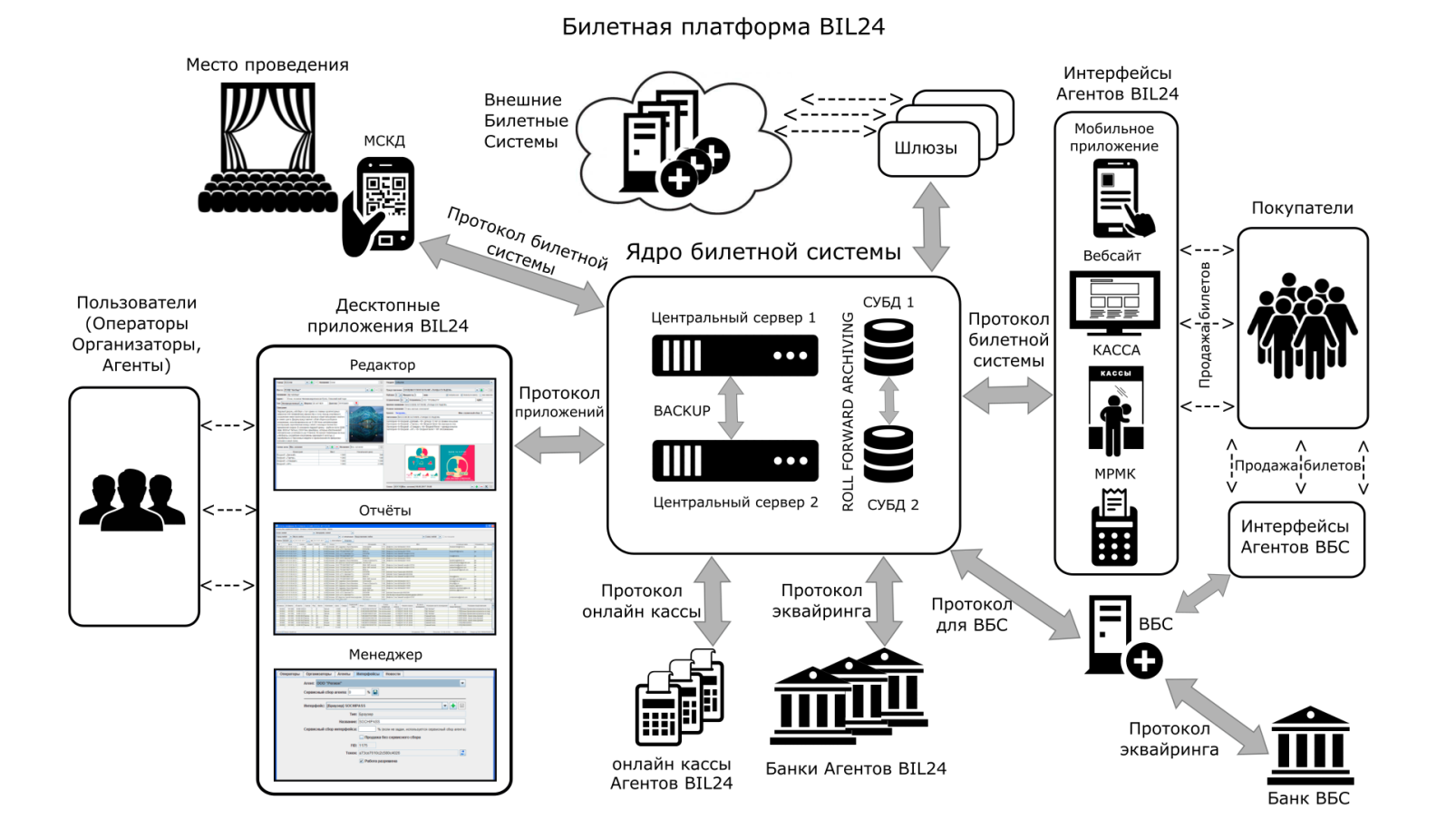


Рисунок 4 – Архитектура системы BIL24

2.1.4 Преимущества системы

BIL24 обладает рядом несомненных преимуществ перед системами конкурентов.

Платформа создана в 2016-2017 годах на основе современных технологий. Использование программной платформы Java и объектной СУБД промышленного уровня позволяют системе хранить, обрабатывать и анализировать большие объемы сложноструктурированных данных в режиме реального времени. BIL24 использует: JSON для обмена данными с Frontend-интерфейсами, SVG для работы со схемами залов, Roll Forward Archiving (журнализацию транзакций) для восстановления базы данных в случае логических или физических отказов. Платформа является современной разработкой, свободной от груза технологий многолетней давности.

При создании BIL24 использована прогрессивная архитектура. Центральное ядро системы состоит из двух серверов, каждый из которых использует свой экземпляр СУБД. Данные этих СУБД согласованы с помощью журнализации транзакций в режиме реального времени. Это делает возможным «горячую» замену отказавшего сервера. Размещение серверов ядра на разных площадках и резервирование каналов связи делает систему максимально надежной. Широкая сеть Агентов, Организаторов и площадок обслуживается единственным центральным ядром системы, соответственно, нет необходимости размещать у них множество локальных серверов. Открытость системы позволяет быстро и эффективно подключать как Frontend-интерфейсы Агентов, так и Внешние билетные системы (ВБС).

Система проста и удобна для покупателей билетов. Она сочетает необходимый уровень безопасности и возможность покупки билетов за минимум «кликов». Покупателю необходимо только подтвердить свой адрес email, и через какой бы интерфейс он не купил билеты, они появятся в соответствующем разделе сайта или мобильного приложения. Система не применяет к покупателям сложный механизм регистрации, отсутствует тяжеловесный «Личный кабинет». Все направлено на то, чтобы можно было купить билеты и воспользоваться ими, совершая минимум действий.

В силу используемых технологий и примененной архитектуры система способна обрабатывать большие объемы данных и предоставлять подробную отчетность в режиме реального времени в рамках одной объектной СУБД. Формы отчетности создаются в формате Microsoft Excel.

Пользователи системы (Операторы, Организаторы, Агенты) работают с Java-приложениями, обладающими дружественным, стандартизированным, интуитивно понятным графическим интерфейсом. Приложения BIL24 ориентированы на создание и вывод в продажу большого числа Представлений, с множеством Сеансов на разных площадках, в разных городах. На всех этапах, от создания схемы зала, до вывода сеансов в продажу пользователь совершает минимум возможных действий, эффективно расходует свое время.

Система ориентирована на использования всех имеющихся каналов продаж билетов – сайтов, мобильных приложений, социальных сетей, обычных и мобильных касс и т.д. В системе широко применяются мобильные технологии, что позволяет использовать мобильные устройства для разных задач, от проверки билетов (МСКД) до получения отчетности (мобильная версия приложения Отчеты).

Платформа предоставляет инструментарий, подходящий для решения самых разных задач. Различные комбинации инструментов БП позволяют, например, продавать билеты на экскурсии, транспорт, реализовать туристические продукты и много другое.

Билеты BIL24 надежно защищены от подделок и мошенничества криптозащищенными QR-кодами, и собственной технологии их проверки с помощью МСКД.

2.2 Actian NOSQL Object Database (ANOD)

ООСУБД ANOD – коммерческая объектно-ориентированная СУБД, разработанная для использования в крупномасштабных информационных системах соответствующая стандартам ODMG (Object Data Management Group), JDO (Java Data Objects) и ориентированная на платформы Windows и Unix. ANOD обеспечивает эффективную параллельную обработку данных, на базе современных многоядерных процессоров и многопроцессорных серверов при этом зачастую превосходя реляционные СУБД по надёжности функционирования. Целостный объектно-ориентированный подход позволяет до двух раз сократить расходы времени на разработку, отладку и тестирование прикладных информационных систем, снизить их стоимость. ANOD поддерживает все базовые понятия и концепции объектно-ориентированного подхода такие, как объект и идентификатор объекта, атрибут и метод класса, иерархия и наследование классов и обеспечивает непревзойденный уровень работы приложений, базирующихся на Java, .NET, C++, C#, C и других современных объектно-ориентированных средств разработки приложений.

2.3 Порядок решения задач, связанных с интеллектуальным анализом данных BIL24

Билетная система BIL24 является высоконагруженной системой. Центральный сервер BIL24 ведет несколько журналов событий (log-файлов), куда записываются большие объемы сырых данных с детальной информацией событиях в системе. Анализ этих данных полезен для улучшения работы платформы, прогнозирования её поведения и увеличения экономических показателей. Со временем, для удобства анализа часть этих данных размещается в СУБД. При размещении данных о событиях в СУБД необходимо соблюсти баланс между практической ценностью этих данных и результатов их анализа, и объемом этих данных, увеличивающих размер и нагрузку на СУБД.

Для решения задач, связанных с анализом больших объемов сырых данных (Data mining) в проекте BIL24 установлен следующий порядок:

1) Составление задачи, для решения которой необходимы методы Data Mining.

2) Определение класса задачи, способов ее решения. Выполнение пробного анализа большого объема сырых данных из лог-файлов BIL24 с помощью отельной утилиты.

3) Определение практической ценности результатов анализа для решения задачи из пункта 1.

4) Определение и создание объектов с оптимальной структурой и объемом данных в СУБД.

5) Реализация в платформе BIL24 программного модуля для сбора, анализа и визуализации результатов решения задачи. Применение результатов в практической деятельности.

3. Решение задач по выявлению аномалий при взаимодействии центрального сервера BIL24 и интерфейсов агентов

3.1 Задача по устранению ошибок и оптимизации работы интерфейсов BIL24 по «Протоколу билетной системы»

Для продажи билетов Агенты используют интерфейсы. Каждый интерфейс (FRONTEND) имеет уникальный идентификатор FRONTEND ID (FID) и соответствующий ему токен. Эти данные используются для взаимодействия интерфейса и центрального сервера BIL24 по «Протоколу билетной системы». В основном, интерфейсами агентов являются различные вебсайты, мобильные приложения, кассы. Имеется отдельный тип интерфейса для подключенных к BIL24 клиентских билетных систем.

«Протокол билетной системы» (далее ПБС) содержит команды, необходимые для продажи билетов через интерфейс агента, основные из них:

- GET\_ACTIONS\_V2 – получение списка мероприятий в городе;

- GET\_ACTION\_EXT – получение информации о конкретном мероприятии и его сеансах;

- GET\_SEAT\_LIST или URL\_SEATING\_PLAN – получение списка мест на конкретный сеанс или его схемы зала с количеством мест;

- RESERVATION – резервирование мест на схеме зала;

- CREATE\_ORDER – создание заказа с билетами на зарезервированные места для оплаты;

ПБС и его команды реализуют разработчики интерфейсов агентов. Эффективная реализация ПБС подразумевает использование команд только по необходимости. Например, если клиент посещает страницу сайта Агента, на которой отображается выбор города и все события в нем, то необходимо выполнить команду GET\_ACTIONS\_V2. Если клиент выбирает конкретное мероприятие, то нужно выполнить команду GET\_ACTION\_EXT, чтобы получить все данные об этом мероприятии и его сеансах. Если клиент дошел до резервирования мест и покупки билетов на конкретный сеанс, то надо последовательно выполнить команды GET\_SEAT\_LIST, RESERVATION, CREATE\_ORDER. Эффективное использование ПБС подразумевает предоставление клиентам всех возможностей сервиса по покупке билетов при минимальной нагрузке на центральный сервер платформы. Сторонние разработчики, реализующие ПБС для агентов, часто обладают низкой квалификацией и программируют использование команд ПБС некорректно и неэффективно. Для улучшения их реализации ПБС и снижения бесполезной нагрузки на центральный сервер платформы необходимо определить шаблон нормального взаимодействие интерфейса и сервера, выявить аномальные отклонения от него и предоставить разработчикам интерфейсов рекомендации по устранению этих аномалий.

В соответствии с принятым в проекте BIL24 «Порядком разработки модулей для анализа сырых данных платформы BIL24» были предприняты следующие действия:

1) Сформулирована задача: «Определить шаблон нормального взаимодействие интерфейса и сервера с помощью ПБС, выявить аномальные отклонения от него и предоставить разработчикам интерфейсов рекомендации по устранению этих аномалий».

2) Определены следующие параметры задачи.

Класс задачи: обнаружение аномалий (anomaly detection) при использовании ПБС. Задача состоит в идентификации событий и параметров взаимодействия через ПБС, которые не соответствуют ожидаемому шаблону с нормальной активностью.

Метод решения: полуавтоматическое определение аномалий (Semi-supervised anomaly detection), в ходе которого создается шаблон, соответствующий нормальному (эффективному) использованию ПБС. Затем, сырые данные лог-файлов BIL24 проверяются на соответствие данному шаблону.

Наиболее протестированным интерфейсом (Frontend), подключенным через ПБС к билетной платформе, является «Основное мобильное приложение (ОМП) для Android – Билеты 24». Это приложение было выполнено разработчиком ПБС одновременно с созданием кода центрального сервера BIL24. ОМП предоставляет высокий уровень сервиса покупателям и взаимодействует с центральным сервером наиболее оптимально. Реализацию ПБС в приложении «Билеты 24» логично считать эталонной.

Для создания шаблона возьмем данные об использовании команд протокола интерфейсом «BIL24 ОМП Android» (1001) за 11 ноября 2017года (рисунок 4). В эти сутки с помощью ОМП продано 7 заказов (13 билетов).

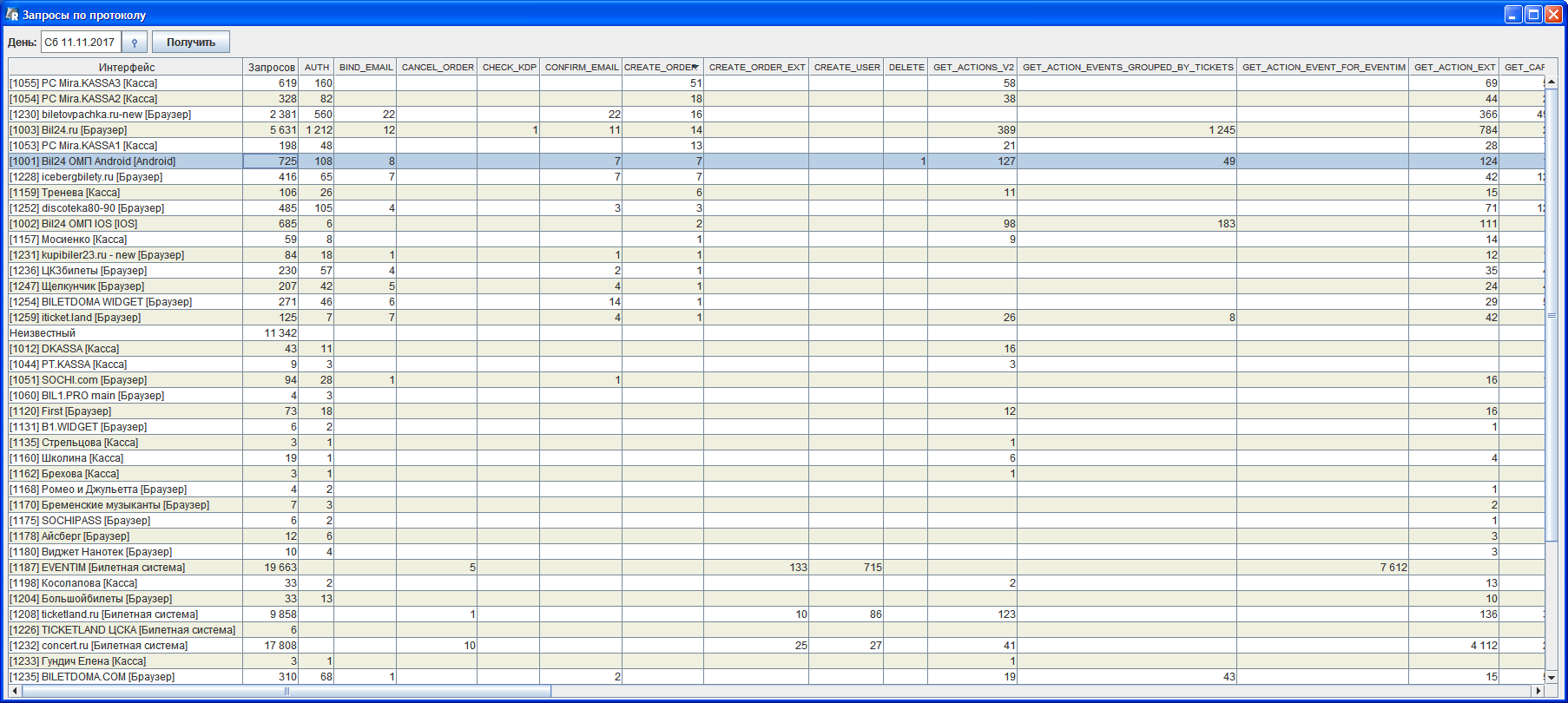


Рисунок 4 – Эталонный шаблон, интерфейс BIL24 ОМП Android от 11.11.17

Выполним сравнение продаж ОМП с интерфейсом BIL24.ru (1003) за эти же сутки. С помощью интерфейса BIL24 продано 14 заказов (21 билет).

3) Результаты пробного анализа (пункт 2) признаны ценными для оптимизации и снижения нагрузки на центральный сервер и выработке рекомендаций сторонним разработчикам.

4) По результатам пробного анализа данных, в СУБД ANOD реализованы объекты для хранения количества запросов за одни сутки. В разрезе каждой команды протокола и для каждого интерфейса. Это минимально необходимые данные для мониторинга и анализа работы интерфейсов агентов через ПБС.

5) В центральном сервере BIL24 реализован программный модуль для подготовки и сохранения в СУБД данных о запросах за сутки (п.4), в приложении «Отчеты BIL24» создан пункт меню статистика и окно с сортируемой таблицей «Запросы по протоколу».

По результатам анализа и визуализации данных таблицы «Запросы по протоколу» предприняты следующие действия:

- проведена работа с программистами, реализующими ПБС на стороне агентов, им предоставлены данные о некорректном и неоптимальном использовании протокола, выработаны рекомендации как улучшить их реализацию ПБС.

- создан и настроен многоуровневый кэш (промежуточный буфер с быстрым доступом) для кеширования множественных запросов ПБС.

- изменен ряд параметров запросов, для уменьшения их количества.

Сравним данные об использовании команд (запросов) ПБС пяти интерфейсов: BIL24 ОМП Android (рисунок 5), BIL24.ru [Браузер] (рисунок 6), EVENTIM [Билетная система] (рисунок 7), PC Mira.Kassa3 [Касса] (рисунок 8), ticketland.ru [Билетная система] (рисунок 9). В модуле «Статистика» с января по ноябрь 2017 года, в последнее число каждого месяца были выбраны сведения об общем количестве запросов и созданных заказов для каждого интерфейса.

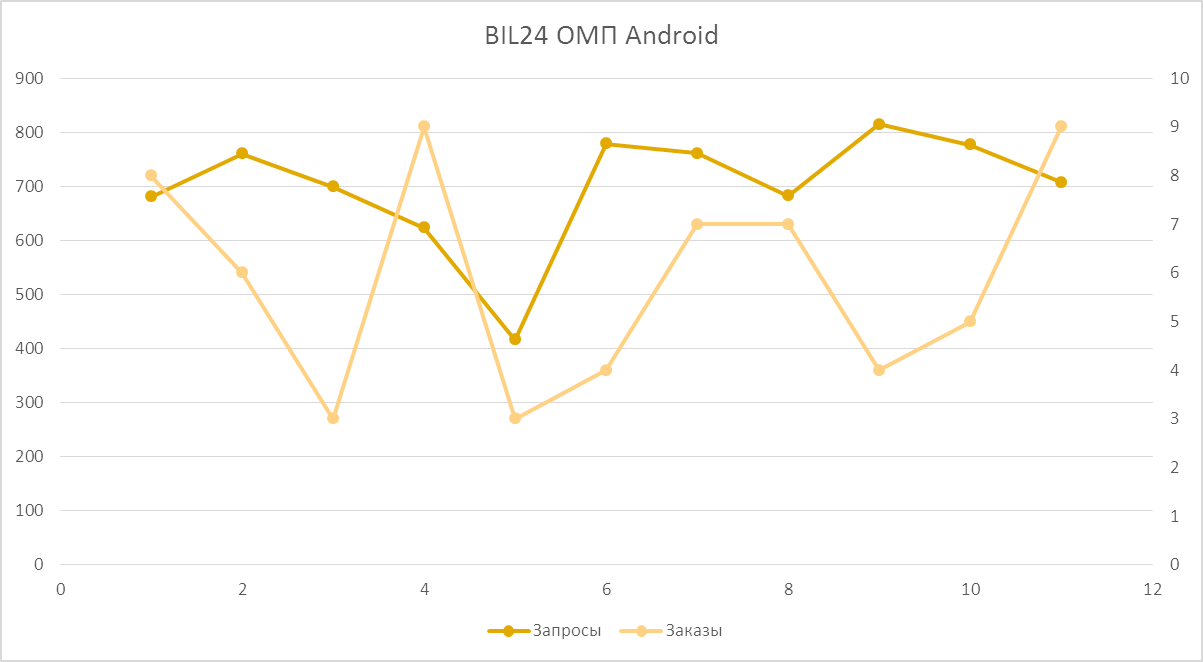


Рисунок 5 – График изменения общего числа запросов и созданных заказов интерфейса BIL24 ОМП Android

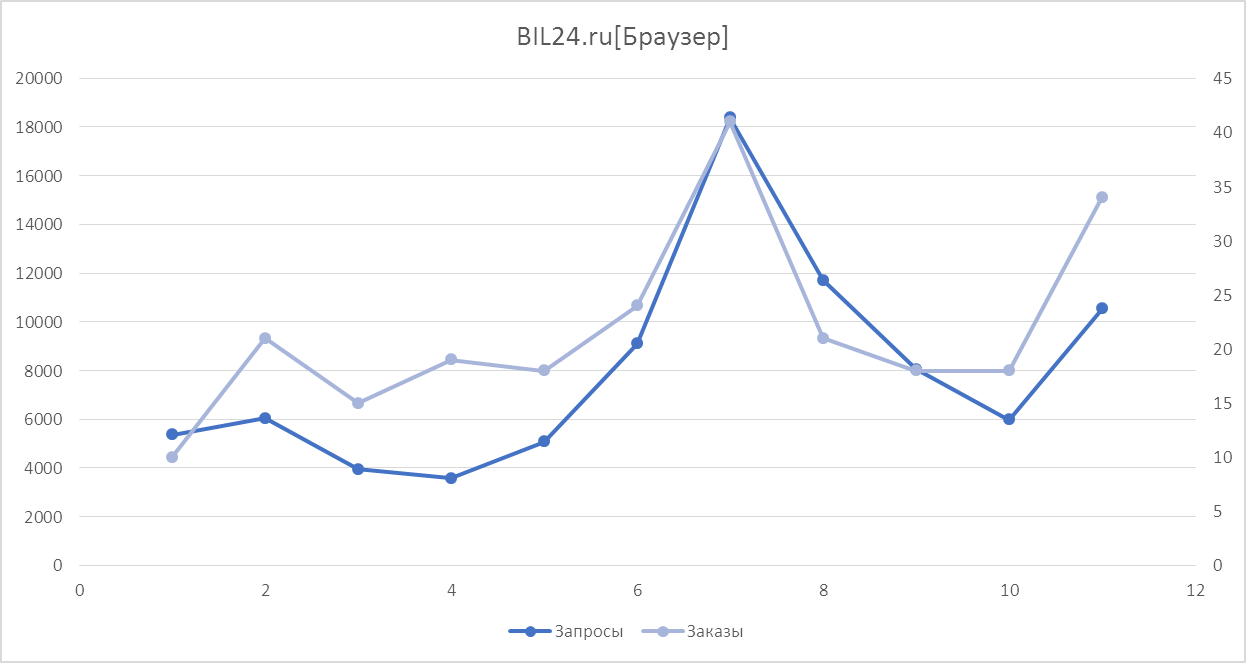


Рисунок 6 – График изменения общего числа запросов и созданных заказов интерфейса BIL24.ru [Браузер]

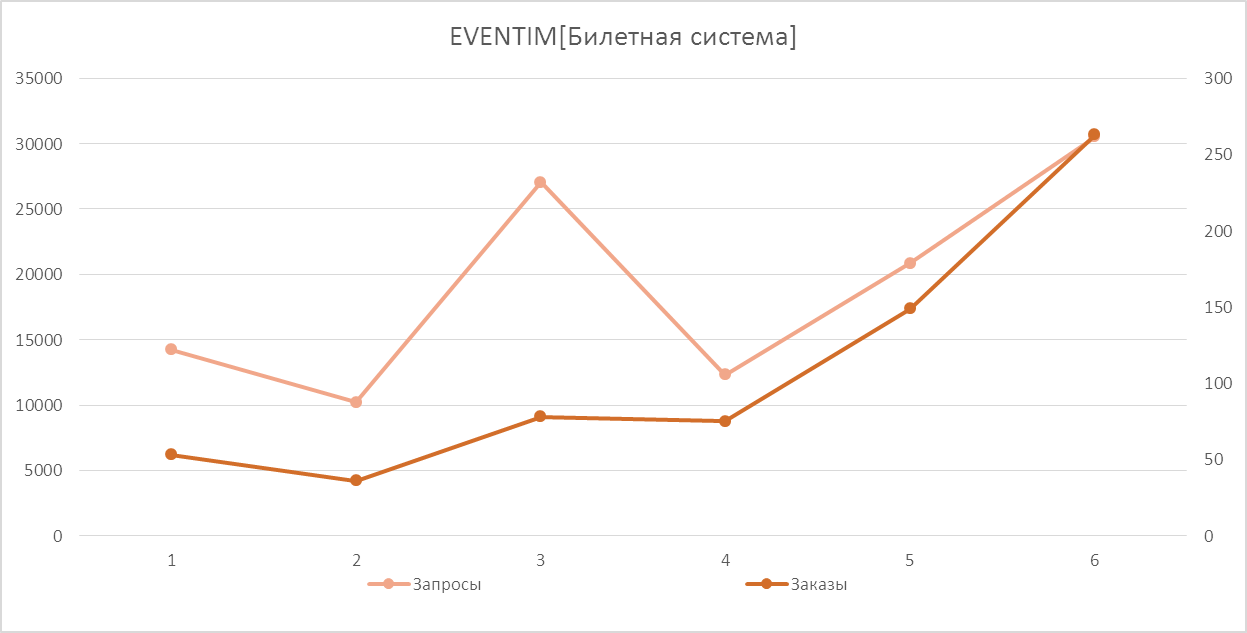


Рисунок 7 – График изменения общего числа запросов и созданных заказов интерфейса EVENTIM [Билетная система]

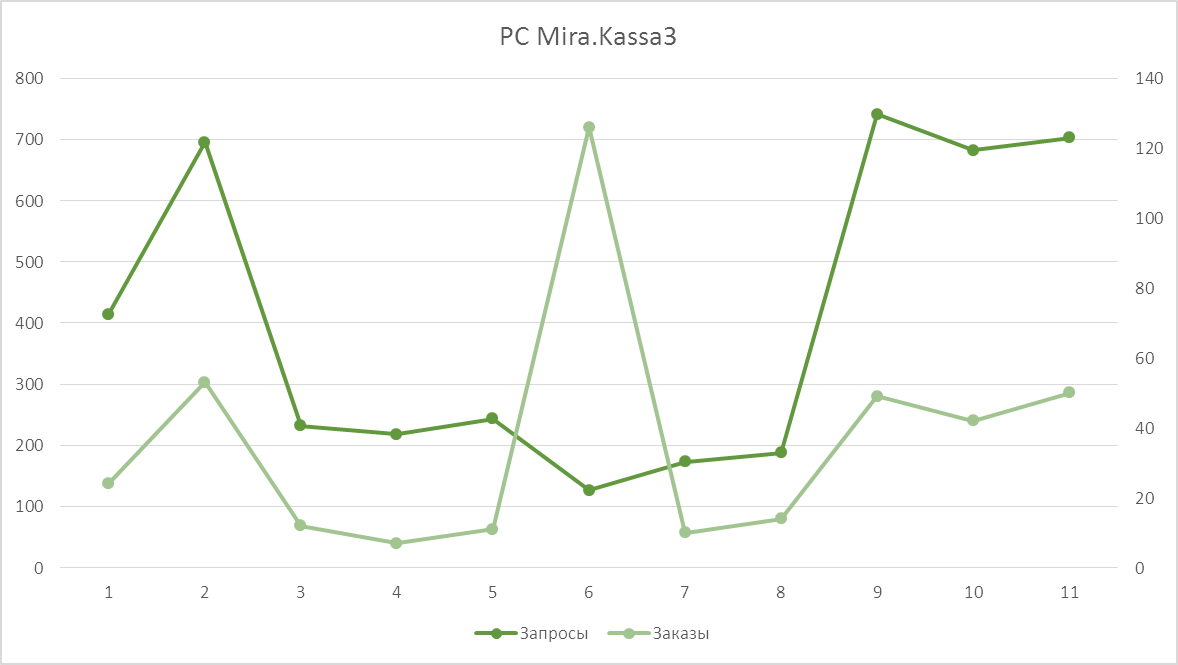


Рисунок 8 – График изменения общего числа запросов и созданных заказов интерфейса PC Mira.Kassa3 [Касса]

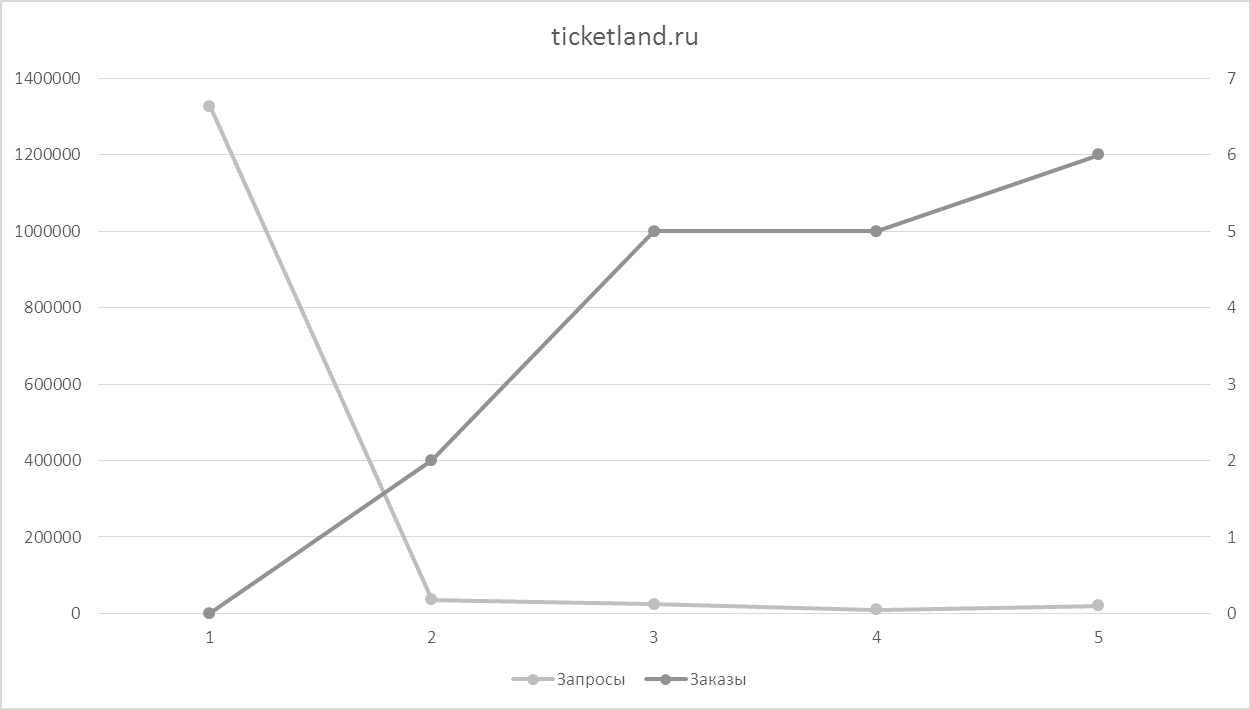


Рисунок 9 – График изменения общего числа запросов и созданных заказов интерфейса ticketland.ru [Билетная система]

На графиках трёх из пяти интерфейсов (BIL24 ОМП Android, EVENTIM [Билетная система], PC Mira.Kassa3 [Касса]) видно, что они близки к такой тенденции, что общее количество запросов снижается, а количество созданных заказов увеличивается.

3.2 Задача по выявлению простоев Внешних билетных систем и интернет-эквайрингов, используемых платформой BIL24

В связи с жалобами покупателей в службу технической поддержки платформы BIL24 на невозможность приобрести билеты на некоторые представлений и оплатить их банковскими картами, проведено исследование сырых данных лог-файлов центрального сервера BIL24. В результате исследования получена информация о наличие множества ошибок, свидетельствующих о недоступности некоторых внешних систем в течение значимого времени.

По принятому в проекте BIL24 «Порядку разработки модулей для анализа сырых данных платформы BIL24» были предприняты следующие действия:

1) Сформулирована задача: «Проанализировать статистику недоступности внешних билетных систем и интернет-эквайрингов и визуализировать итоговые данные с целью вывода неочевидных закономерностей и наблюдений».

2) Определены следующие параметры задачи.

Класс задачи: обнаружение и анализ простоев внешних систем.

Метод решения: разработана утилита GatewayDenial (программный код приведён в приложении А), анализирующая лог-файл Operabilties.txt, который состоит из записей об изменении статуса внешних систем.

Первый пример записи:

01.10.17 10:48:40.587|VTIX\_2|Доступен -> connect timed out.

В этой записи:

01.10.17 10:48:40.587 – время события;

VTIX\_2 – идентификатор внешней системы;

Доступен -> connect timed out – события изменения статуса внешней системы из доступной в недоступную.

Второй пример записи:

01.10.17 16:29:17.233|VTIX\_2|Недоступен -> Доступен

В этой записи:

01.10.17 16:29:17.233– время события;

VTIX\_2 – идентификатор внешней системы;

Недоступен -> Доступен *-* события изменения статуса внешней системы из недоступной в доступную.

Данные лог файла Operabilties.txt пишет модуль, выполняющий постоянный опрос внешних систем в случае отсутствия их активности.

Утилита GatewayDenial преобразует сырые данные лог-файлов в информацию о простоях внешних систем (таблица 1). Где первый столбец таблицы содержит идентификатор систем, второй – количество простоев каждой системы и третий – общее время простоев каждой системы в формате часы : минуты : секунды.

Таблица 1 – Пример заполнения таблицы данными, полученными в результате работы утилиты gatewayDenial.

|  |  |  |
| --- | --- | --- |
| ID ВБС | Количество простоев | Общее время простоев |
| infotec\_3 | 12 | 2h 0min 36sec |
| TICKETLAND\_1 | 47 | 0h 43min 11sec |
| VTIX\_2 | 15 | 0h 35min 7sec |
| Raduga\_1 | 9 | 1h 16min 50sec |

В ходе работы GatewayDenial считывает данные о событиях изменения статуса внешних систем в объект TreeMap oper, где данные представлены в виде ключ : значение. Ключом является идентификатор ВС, а значением - ArrayList из объектов типа Record, которые содержат данные из каждой строки лог-файла (время, статус события, описание ошибки).

Данные из TreeMap oper анализируются для каждой из внешних систем. В результате анализа вычисляются простои каждой внешней системы с учётом неизвестного первоначального статуса и перезагрузок центрального сервера BIL24. Результаты анализа сохраняются в объект TreeMap idleMap в виде ключ : значение, где ключ – идентификатор внешней системы, а значение это ArrayList из объектов типа Idle. Объекты типа Idle содержат время начала и конца простоя, его длительность.

Информация из объекта TreeMap idleMap преобразуется в таблицу 1, которая является итогом работы утилиты. Эта таблица используется для визуализации данных о простоях в виде диаграмм.

Пример работы программы для файла Operabilties.txt за период с 1 по 28 октября (рисунок 11).

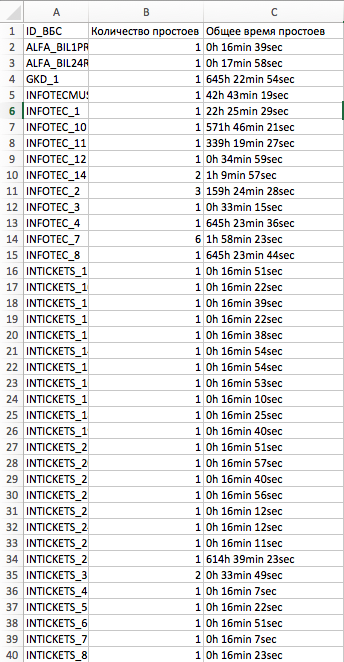


Рисунок 11 – Фрагмента из составленного отчёта в Excel

Визуализация результата проведённого анализа с использованием RawGraphs для файла Operabilties.txt за период с 1 по 28 октября представлена на рисунке 13.

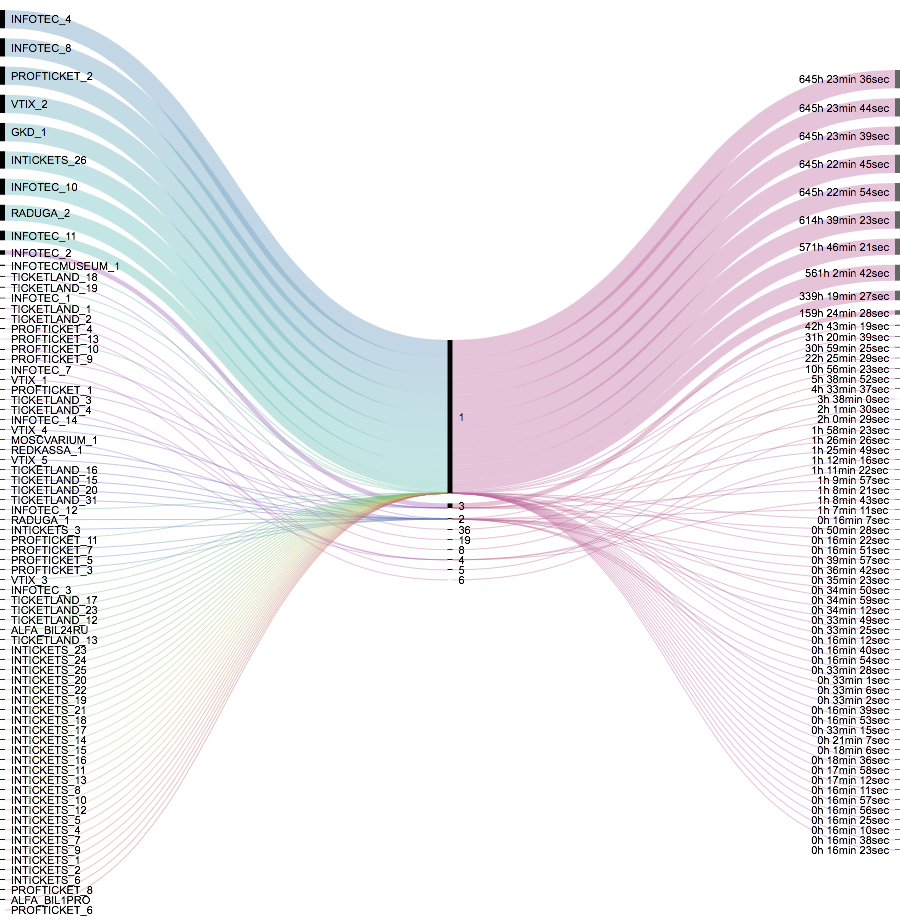
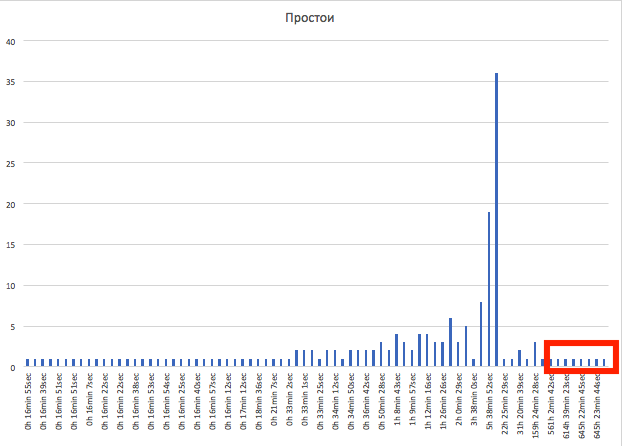


Рисунок 13 – Визуализация результата работы утилиты gatewayDenial

Из рисунка 13 видно, что внешние системы INFOTEC\_4, INFOTEC\_8, PROFITICKET\_2, VTIX\_2, GKD\_1, INTICKETS\_26, INFOTEC\_10, RADUGA\_2, INFOTEC\_11 были недоступны большую часть времени. Их можно считать выбросами, то есть значениями, резко отличающимися от других значений в собранном наборе данных.

На основе результата программы был создан график (рисунок 12), включающий в себя сведения о количестве и длительности простоев 76 внешних систем. Среднее время их недоступности – 74,5 часа. После исключения внешних систем, которые объявлены выбросами, среднее время простоев уменьшилось, примерно, в 2 раза – до 34 часов.

Рисунок 12 – График простоя внешних систем за период с 1 по 28 октября

3) Результаты пробного анализа и визуализации полученных данных помогли представить информацию о простоях в наглядном виде, определить практически полностью недоступные для BIL24 внешние системы. С помощью визуализации большой объём сырых данных из Operabilties.txt представлен в компактном, сжатом и удобном для понимания виде. Такие результаты не могли быть получены при просмотре сырых данных из лог-файла.

В соответствии с Порядком решения задач, описанном в пункте 2.3 второго раздела планируется создание в BIL24 модуля для постоянного анализа данных о простоях внешних систем.

ЗАКЛЮЧЕНИЕ

Подводя итоги работы по применению Data Mining в билетной платформе BIL24 следует отметить, что исследование сырых данных лог-файлов платформы позволило обнаружить в них важные и практически ценные знания, которые постоянно используются для улучшения работы системы, увеличения продажи билетов, и, в конечном счете, для роста доходов от этой деятельности.

Порядок решения задач, связанных с интеллектуальным анализом данных BIL24 (раздел 2.3), на практике доказал свою эффективность. В соответствии с этим Порядком созданы объекты СУБД и код для выявления аномалий в реализации «Протокола билетной системы», создаются необходимые структуры для визуализации данных о недоступности внешних систем, в перспективе – данных о продажах билетов.

В рамках курсовой работы выполнен интеллектуальный анализ данных об использовании запросов по «Протоколу билетной системы» (раздел 3.1) успешно решается задача по устранению ошибок и оптимизации работы интерфейсов BIL24 по данному протоколу. Для этой цели разработаны алгоритмы и программные модули, средства визуализации больших объемов данных. Работ по мониторингу и улучшению различных реализаций Протокола билетной системы ведется на постоянной основе.

Одной из важнейших задач Data Mining является обобщение и визуализация больших объемов данных в удобном для интерпретации виде. Для BIL 24 выполнена работа по выявлению простоев Внешних билетных систем и интернет-эквайрингов, визуализация данных о простоях в удобной для анализа и принятия решений форме (раздел 3.2). Изучение отчетов в виде графиков и диаграмм позволили провести работу с администрацией внешних систем, снизить количество простоев, и, как следствие, снизить количество незавершенных заказов. Ведется работа по созданию программных модулей для визуализации различных данных BIL24 в наглядной графической форме

В перспективе, в развитие проведенной работы планируется:

Создание документации о применении Data Mining в BIL24, о возможностях интеллектуального анализа данных для пользователей системы. Создание по заказам пользователей новых форм визуальных отчетов, для обобщений и интерпретации больших объемов данных, для принятия бизнес-решений на их основе.

Создание программных модулей и структур в СУБД для интеллектуального анализа данных о продажах билетов в разрезе представлений, интерфейсов, агентов, мест проведения и т.д., для выявления закономерностей в данных о незавершенных заказах.

Применение методов и средств Data Mining позволит вывести билетную платформу BIL24 н новый уровень развития.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Описание BIL24 [электронный ресурс]. URL: <http://kernel.group/bil24/bil24.html> (дата обращения: 24.09.17).
2. Описание Actian NOSQL Object Database (Versant) [электронный ресурс]. URL: kernel.group/rpc/versant.htm (дата обращения: 24.09.17).
3. «Основное мобильное приложение (ОМП) для Android – Билеты 24» [электронный ресурс] .URL: <https://play.google.com/store/apps/details?id=com.bil24&hl=en> (дата обращения: 25.09.17).
4. Барсегян, А. А. Б26 Анализ данных и процессов: учеб. пособие / А. А. Барсегян, М. С. Куприянов, И. И. Холод, М. Д. Тесс, С. И. Елизаров. — 3-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2009.

ПРИЛОЖЕНИЕ А

Программный код утилиты gatewayDenial.

Класс MainVlass.java:

import java.io.\*;

import java.text.DateFormat;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.\*;

public class MainClass {

public static void main (String[] args){

Map<String,ArrayList<Record>> oper = new TreeMap<String,ArrayList<Record>>();

Record myRecord;

StringTokenizer st;

String strLine;

Date eventTime = null;

String gateway = null;

Boolean event;

String description;

Boolean firstDate = true;

Date logStartDate = new Date();//дата начала лога

Date logEndDate = new Date();//дата конца лога

try{

FileInputStream fstream = new FileInputStream("/Users/lizaorlova/IdeaProjects/gatewayDenial/src/logTest3.txt");

BufferedReader br = new BufferedReader(new InputStreamReader(fstream));

while((strLine = br.readLine()) != null){

st = new StringTokenizer(strLine,"|");

try {

eventTime = new SimpleDateFormat("dd.MM.yy HH:mm:ss.SS").parse(st.nextToken());

}

catch(ParseException p){

System.out.println(p);

}

if(firstDate){

logStartDate = eventTime;

firstDate = false;

}

gateway = st.nextToken();

description = st.nextToken();

if(description.indexOf("Недоступен -> Доступен") == -1){

event = false;

} else {

event = true;

}

myRecord = new Record(eventTime, event, description);

if(oper.containsKey(gateway) == false){

oper.put(gateway, new ArrayList<Record>());

}

oper.get(gateway).add(myRecord);

System.out.println("Key = " + gateway + " add record = " + myRecord.event + " begin = " + myRecord.eventTime + " " + myRecord.description );

}

logEndDate = eventTime;

fstream.close();

IdleCounter idelCount = new IdleCounter(oper, logStartDate, logEndDate);

}catch (IOException e){

System.out.println(e);

}

}

}

Класс IdleCounter.java:

import java.util.\*;

public class IdleCounter {

Map<String,ArrayList<Idle>> idleMap = new TreeMap<String,ArrayList<Idle>>();

ArrayList<Record> tempRecordArrayList;

ArrayList<Idle> tempIdleArrayList;

Boolean previousState = true; //предыдущее состояние внешней системы, true - работала, false - не работала

Date periodBegin;

Date startPeriodDate;

Date endPeriodDate;

Idle myIdle;

Boolean startRecord = true;

Boolean idleEndFound = false;

public IdleCounter(Map<String,ArrayList<Record>> tm, Date d, Date d2){

startPeriodDate = d;

endPeriodDate = d2;

System.out.println("");

for (Map.Entry e: tm.entrySet()){

idleMap.put(e.getKey().toString(), new ArrayList<Idle>());

tempRecordArrayList = tm.get(e.getKey());

tempIdleArrayList = idleMap.get(e.getKey());

startRecord = true;// для первого раза

idleEndFound = false;// найден конец простоя для записи в объект idle

previousState = true;//считаем, что первоначальное состояние вс рабочее

for (Record r: tempRecordArrayList){

//неожиданные записи и состояния, связанные с изменениями состояний внешней системы из-за выключения сервера

System.out.println("Key = " + e.getKey()+ " previousState = " + previousState + " r.event = " + r.event + " date = " + r.eventTime);

if(previousState == true && r.event == true){// внешняя система предположительно работала и запись включения

if (startRecord){

myIdle = new Idle(startPeriodDate);

myIdle.setEndDate(r.eventTime);

tempIdleArrayList.add(myIdle);

System.out.println("add Idle1 = " + myIdle);

startRecord = false;

}

previousState = true;

idleEndFound = true;

}

if(previousState == false && r.event == false){// внешняя система предположительно не работала и запись остановк

/\* myIdle = new Idle(r.eventTime);

tempIdleArrayList.add(myIdle);

System.out.println("add Idle2 = " + myIdle);\*/

}

// Ожидаемые состояния

if(previousState == true && r.event == false){ // внешняя система работала и запись остановки, начало простоя

myIdle = new Idle(r.eventTime);

previousState = false;

idleEndFound = false;

}

if(previousState == false && r.event == true){ // внешняя система не работала и запись запуска, конец простоя

myIdle.setEndDate(r.eventTime);

tempIdleArrayList.add(myIdle);

System.out.println("add Idle3 = " + myIdle);

// System.out.println("key " + e.getKey() + " add " + myIdle.begin + " " + myIdle.end);

previousState = true;

idleEndFound = true;

}

}

if(idleEndFound == false){

myIdle.setEndDate(endPeriodDate);

tempIdleArrayList.add(myIdle);

System.out.println("add Idle4Start = " + myIdle.begin + " Idle4End = " + myIdle.end);

}

}

CSVGen csvg = new CSVGen();

long summaryTime = 0;

System.out.println("");

for(Map.Entry entry: idleMap.entrySet()){

System.out.println(entry.getKey() + " : " + entry.getValue());

summaryTime = 0;

for (Idle i : idleMap.get(entry.getKey())){

summaryTime += i.idleResult(i.begin, i.end);

}

csvg.writeData(entry.getKey() + "," + idleMap.get(entry.getKey()).size() + "," + getTime(summaryTime));

}

csvg.close();

}

private String getTime(Long s){

Long hours = s / 3600000;

Long minutes = (s % 3600000)/ 60000;

Long seconds = s % 60;

//return hours + "h " + minutes + "min " + seconds + "sec";

return s.toString();

}

}

Класс CSVGen.java:

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.OutputStream;

public class CSVGen {

FileOutputStream os;

public CSVGen(){

try {

os = new FileOutputStream(new File("output.txt"));

} catch (IOException e) {

e.printStackTrace();

}

}

public void writeData(String data){

String dataOut = data + System.getProperty("line.separator");

try{

os.write(dataOut.getBytes(), 0, dataOut.length());

} catch (IOException e){

e.printStackTrace();

}

}

public void close(){

try {

os.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

Класс Idle.java:

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.Date;

import java.util.TimeZone;

public class Idle {

Date begin;

Date end;

public Idle(Date mBegin){

begin = mBegin;

}

public void setEndDate(Date mEnd){

end = mEnd;

}

public long idleResult(Date begin, Date end){

if (end != null){

long seconds = (end.getTime()- begin.getTime());

return seconds;

} else {

return -1;

}

}

}

Класс Record.java:

import java.util.Date;

public class Record {

Date eventTime;

Boolean event;// true - система запустилась, false - остановилась

String description;

public Record(Date myEventTime, Boolean myEvent, String myDescription){

eventTime = myEventTime;

event = myEvent;

description = myDescription;

}

}

ПРИЛОЖЕНИЕ Б

Пример фрагмента, анализируемого в курсовой работе лог-файла

01.10.17 10:47:39.216|INFOTEC\_4|Доступен -> com.sun.xml.internal.ws.client.ClientTransportException: HTTP transport error: java.net.SocketTimeoutException: connect timed out

01.10.17 10:47:40.948|INFOTEC\_8|Доступен -> [code=20993] Ошибка! (Код= 20993)

01.10.17 10:47:41.260|RADUGA\_2|Доступен -> Код ответа: 404|Текст ошибки: <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">|<html><head>|<title>404 Not Found</title>|</head><body>|<h1>Not Found</h1>|<p>The requested URL /events was not found on this server.</p>|</body></html>|

01.10.17 10:48:05.253|PROFTICKET\_2|Доступен -> com.sun.xml.internal.ws.client.ClientTransportException: HTTP transport error: java.net.SocketTimeoutException: connect timed out

01.10.17 10:48:40.587|VTIX\_2|Доступен -> connect timed out

01.10.17 10:49:05.718|GKD\_1|Доступен -> connect timed out

01.10.17 16:12:06.208|PROFTICKET\_10|Доступен -> com.sun.xml.internal.ws.client.ClientTransportException: HTTP transport error: java.net.SocketTimeoutException: connect timed out

01.10.17 16:12:29.203|VTIX\_1|Доступен -> connect timed out

01.10.17 16:29:13.458|PROFTICKET\_10|Недоступен -> Доступен

01.10.17 16:29:17.233|VTIX\_1|Недоступен -> Доступен

01.10.17 20:58:57.545|INTICKETS\_3|Доступен -> java.net.SocketTimeoutException: connect timed out