ПОСТРОЕНИЕ СОВРЕМЕННЫХ КОРПОРАТИВНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Логиновский О. В.[1], Шестаков А. Л.[2], Шинкарев А. А.[3]
(ФГАОУ ВО «ЮУрГУ (НИУ)», Челябинск)

Представлен анализ современных подходов и практик, которые применяются при построении корпоративных информационных систем. Выявляются проблемы оторванности практики от теоретических изысканий. Рассматриваются основные характеристики больших данных, решений для хранения корпоративной информации, применимость машинного обучения, его инструменты и актуальность в современных условиях. В том числе рассматриваются такие необходимые аспекты разработки информационных систем, как надежность, масштабируемость, поддерживаемость и безопасность. В ныне действующих системах, как правило, хранится только текущее состояние. Предлагается рассматривать использование потока событий в качестве адекватного современным потребностям бизнеса способа хранения, обработки и представления данных для принятия управленческих решений в условиях нестабильности. Раскрываются причины сложности перехода к потоку событий в качестве системы хранения информации. В свете меняющегося набора используемых технологий и новом фокусе на анализе данных формулируются базовые требования рынка труда к специалистам на стыке программирования и науки о данных. Среди основных навыков и требований рынка выделяется работа на стыке нескольких направлений: экспертиза в развертывании приложений в облачных сервисах, навыки написания чистого кода, владение математическим аппаратом на уровне, необходимом для осмысленного использования моделей машинного обучения, а также грамотное умение выстроить параллельную обработку данных в несколько потоков. В качестве главных конкурентных преимуществ бизнеса выделяются гибкость и способность извлечь максимум из информации.

Ключевые слова: корпоративные информационные системы, большие данные, машинное обучение, масштабируемость, поток событий, анализ.

1. Введение

Современные информационные системы и подходы к их построению значительно отличаются от тех, что создавались

[1] *Олег Витальевич Логиновский, д.т.н., профессор* (*loginovskiyo@mail.ru*).
[2] *Александр Леонидович Шестаков, д.т.н., профессор* (*admin@susu.ru*).
[3] *Александр Андреевич Шинкарев, к.т.н.* (*sania.kill@mail.ru*).

в прошлом и даже менее десяти лет назад. Проникновение программного обеспечения (ПО) с открытым исходным кодом (Open Source) в корпоративные приложения, усиление влияния мнения сообщества разработчиков на крупных производителей платного закрытого ПО, таких как Microsoft, переход к децентрализованным паттернам построения информационных систем – все это в значительной степени отличает и характеризует вектор развития архитектуры и разработки сегодня.

Во многом то, что мы видим сейчас, объясняется ростом доступности облачных вычислительных ресурсов, развитием таких провайдеров, как Azure от Microsoft и AWS от Amazon.

Сейчас каждый может попробовать работу с удаленными серверами. Это просто, быстро и сравнительно недорого. За счет этого децентрализованная архитектура стала краеугольным камнем развития разработки корпоративных информационных систем.

Создание такого сайта, как GitHub, сделало куда более популярным использование хорошо зарекомендовавших себя открытых библиотек в серьезной корпоративной разработке, что снижает зависимость от производителей платных инструментов и в конечном итоге улучшает качество и скорость совершенствования инструментария.

Развитие технологий, уменьшение размеров датчиков, распространение подключения к Интернету, рост скорости и пропускной способности, в свою очередь, определили появление так называемого интернета вещей (Internet of Things). Объем данных, которые генерируются сейчас, растет с каждым годом весьма значительно.

Появление возможности создавать цифровую тень объектов производства, генерировать большой непрерывный поток данных дополнительно подстегнуло развитие NoSQL-решений (Not Only SQL) для распределенного хранения данных. Эти решения в большинстве своем обладают лучшими возможностями горизонтального масштабирования, чем традиционные реляционные базы данных. Также NoSQL-решения в среднем рассчитаны на большее количество операций записи в секунду.

В свою очередь, наличие большого количества данных поставило вопрос о том, как их использовать, что нового можно узнать о предприятии, клиентах и конкурентах. И здесь на сцену выходит направление науки о данных (Data Science) с возможностями применения моделей машинного обучения (Machine Learning) к имеющимся большим данным (Big Data).

Несмотря на развитие сообщества, инструментария, изменения фокуса решаемых бизнес-задач, инертность корпоративных процессов и длина цепочки принятия решений во многом не позволяет совершить качественный скачок и идти в ногу со временем, чтобы решать задачи на должном уровне эффективности, не проигрывая более подвижным конкурентам.

Помимо закостенелости организационной структуры также имеет место определенный кадровый голод. На рынке труда сложно отыскать инженеров, которые могли бы не только справляться с уже типичными задачами разработки, но и обладали бы знаниями и навыками для решения новых задач в области науки о данных. Проблема заключается в том, что программисты не могут легко освоить сложное направление, требующее воскрешения забытого математического аппарата, а специалисты науки о данных не являются инженерами в полной мере и нуждаются в помощи в решении задач развертывания, поддержки и выстраивании грамотной архитектуры.

В итоге мы имеем значительный разрыв между последними наработками и тем, что в реальности используется в корпоративной разработке. И этот разрыв существует не только между используемыми инструментами, фреймворками и подходами, но и между теоретическими изысканиями в целом и тем, что в реальности необходимо на практике.

И все-таки, несмотря на все проблемы переходного периода в отрасли, основную ценность представляют именно данные, накапливаемые о пользователях систем, потенциальных покупателях, рынках сбыта и т.д. В контексте решаемых задач именно большие данные имеют ключевое значение для получения конкурентных преимуществ.

2. Большие данные

2.1. ОСНОВНЫЕ ХАРАКТЕРИСТИКИ

Понятие больших данных в каком-то смысле в последнее время стало чересчур общеупотребительным, и его значение сейчас довольно сильно размыто.

Приведем одно из существующих определений больших данных, довольно обобщенное, как и термин, которое оно описывает.

Большие данные – направление в науке и практике, связанное с разработкой и применением методов и средств оперирования большими объемами неструктурированных данных [9].

Такое понимание больших данных, как феномена, обусловленного развитием современных информационных технологий, ставит несколько глобальных вопросов. Первым из них является определение критериев, при удовлетворении которым данные переходят в категорию больших. Вторым же вопросом является отнесение только неструктурированных данных (Schemaless Data) к рассматриваемой категории на данном этапе развития технологий, когда происходит постепенное совмещение реляционных и NoSQL баз данных и подходов к хранению и индексации [15].

Приведем широко известные характеристики больших данных [11]:

- объем (Volume) – сколько данных содержится в наборе;
- разнообразие (Variety) – насколько отличаются друг от друга разные типы данных;
- скорость (Velocity) – какова скорость генерации новых данных;
- достоверность (Veracity) – насколько точны данные.

Существует еще несколько дополнительных характеристик, таких как: жизнеспособность (Viability), ценность (Value), переменчивость (Variability) и визуализируемость (Visualization) [1].

Перечислим типовые источники больших данных [4]:

- интернет вещей;
- социальные сети;
- телекоммуникационные спутники;
- интернет-магазины;
- интернет-энциклопедии;
- различного рода отладочные логи (Logs).

Безусловно, помимо представленных существует еще множество других источников больших данных, и их число продолжит расти.

Перечислим основные типы генерируемых больших данных [11]:

- Структурированные – данные, обладающие определенной схемой (Schema) и фиксированным набором атрибутов.
- Неструктурированные – данные, не имеющие постоянной структуры, зачастую такими данными являются различные текстовые документы.
- На естественном языке – особая разновидность неструктурированных данных, включает в себя все тексты на всех языках мира.
- Машинные – данные, создаваемые компьютерами, приложениями или иными устройствами без вмешательства человека.
- Графовые – данные, чьим естественным представлением является граф, моделирующий попарные отношения между объектами.
- Потоковые – данные, которые создаются в ответ на наступление какого-либо события.
- Аудио, видео и графика.

После выявления источников генераторов больших данных и типов информации, в них содержащихся, следует рассмотреть процесс работы науки о данных в целом. Приведем описание процесса, который позволяет систематизировать работу аналитика [11]:

- Назначение цели исследования – определение объекта исследования, пользы от проведения исследования, выявление необходимых данных и ресурсов.

- Сбор данных – определение источников, необходимых для исследования данных, оценки качества и доступности данных.
- Подготовка данных – повышение качества и однородности данных, их нормализация и очистка от некорректных записей.
- Исследование данных – выявление скрытых связей в данных, лучшее понимание природы исследуемой информации и явлений.
- Моделирование данных – итеративный процесс построения модели с целью получения ответа на главные вопросы исследования с использованием результатов предыдущих этапов.
- Отображение и автоматизация – реализация удобного варианта предоставления результатов работы, автоматизация процесса для обновления результатов при обновлении исходных данных.

Представленный типовой процесс помогает систематически решать как задачи на данных, которые могут уместиться в памяти одного компьютера, так и отвечать на вопросы, заданные к намного большему объему данных, для обработки которых может потребоваться целая группа серверов (кластер). На решение проблем распределенной обработки больших массивов данных и обучения моделей по ним тратится внушительная часть времени ученых и программистов.

Важность решения проблемы эффективной распределенной обработки данных обуславливается объемом генерируемых данных в мире. Он, конечно, поражает воображение. Google обрабатывает 3,5 миллиарда поисковых запросов в день [22], количество подключенных к интернету вещей устройств по разным оценкам может достигнуть 22–50 миллиардов к 2020 году [4, 23], пользователи Instagram создают 95 миллионов постов в день [49].

Однако же сами по себе объемы данных не могут дать бизнесу конкурентное преимущество. Только умение правильно задавать вопросы к этим данным, а также оперативно извлекать из ответов выгоду, перестраивая свой бизнес, позволяют боль-

шим данным оправдывать те накладные расходы и всевозрастающую сложность информационных систем, которые они привносят. Чтобы эффективно работать с данными, аналитику или руководителю, принимающему решения (Stakeholder), необходим инструментарий, и важно, чтобы он подходил для решения имеющихся задач.

2.2. ИНСТРУМЕНТЫ РАБОТЫ С БОЛЬШИМИ ДАННЫМИ

Одним из факторов, который делает методы работы с большими данными более доступными для широкого круга специалистов, является наличие большого количества библиотек, которые скрывают всю сложность работы моделей за своим программным интерфейсом.

С одной стороны, это оказывает положительное влияние на скорость внедрения и распространения новых подходов к ведению бизнеса и их интеграции в большее число ключевых бизнес-процессов. Однако с другой стороны, у специалистов, которые только вчера занимались разработкой таких систем, как, например, документооборот, возникает разрыв между теорией и практикой применения моделей машинного обучения. Это может негативно влиять на конечный результат их работы в случае, когда становится недостаточно лишь стандартных моделей, которые работают «из коробки» и необходима их калибровка или даже каскадирование с другими моделями.

Поэтому для развития корпоративной IT-инфраструктуры в направлении интеллектуального анализа данных и автоматизации поддержки принятия управленческих решений задействованным специалистам необходимо изучение корневых технологий и математического аппарата, которые лежат в основе любой библиотеки моделей машинного обучения. Не удастся решать задачи на больших данных без освоения паттернов параллельной обработки данных, распределенного их хранения, а также масштабируемых многопоточных алгоритмов. MapReduce, предложенный Google и уже считающийся классическим паттерном, может стать хорошей отправной точкой, хотя в Google уже не так распространен [42].

В перспективе подготовка новых кадров должна быть значительно более сконцентрирована на перечисленных «основах», без чего видится невозможным необходимый прорывной рост автоматизации и информатизации российских компаний на уровне западных конкурентов и более высоком.

На сегодняшний день бизнес подталкивает технологии к их развитию в сторону методов и систем работы с большими данными.

Можно выделить несколько наиболее важных причин, которые могут обуславливать развитие технологий по этому пути [29]:

- Бизнесу необходимо быть гибким, иметь возможность приспосабливаться к изменяющимся условиям за счет быстрой и дешевой проверки гипотез и короткого цикла разработки и TTM (Time to Market).
- Компаниям необходимо иметь возможность осуществлять собственную доработку используемых программных средств и систем, что полностью вписывается в концепцию программного обеспечения с открытым исходным кодом (Open Source Software), использование которого широко распространено.
- Возможности значительного роста частоты ЦПУ на данном этапе развития технологий практически исчерпали себя, в отличие от возможности наращивания количества ядер в многопроцессорных системах, что вместе с увеличивающейся пропускной способностью сетей будет лишь и дальше увеличивать роль параллельных вычислений.
- Во многих случаях компаниям выгодно осуществлять аутсорс услуг по предоставлению серверных мощностей. Сервисы Amazon Web Services и Microsoft Azure предлагают востребованные облачные инфраструктуры, такие как сервис IaaS (Infrastructure as a Service).

Создание программных систем является сложной задачей. А переход к параллельному паттерну работы с данными требует особых навыков и внимания к разрабатываемому программному обеспечению. С этой точки зрения особенно важны такие характеристики систем, как:

- надежность (Reliability);
- масштабируемость (Scalability);
- поддерживаемость (Maintainability).

Остановимся на каждом из этих аспектов подробнее, дадим им определения и обоснуем их важность для успешной работы корпоративных информационных систем в целом.

3. Характеристики корпоративных систем

3.1. НАДЕЖНОСТЬ (RELIABILITY)

Понятие надежности агрегирует в себе такие характеристики, как способность восстанавливаться после сбоев, устойчивость к хакерским атакам и непротиворечивость состояния при пользовательских, программных и аппаратных ошибках.

Таким образом, система надежна, если работает правильно. Нет смысла создавать приложение, которое будет способно пережить уничтожение всех серверов, на которых оно развернуто. Однако необходимо стремиться обрабатывать известные виды сбоев на ранних этапах разработки системы. Осмыслить и переписать большую систему сложнее, чем заложить в фундамент долю здоровой паранойи на старте работ.

С точки зрения сбоев аппаратного обеспечения, надежность тесно связана с масштабируемостью, которая обсуждается далее. Масштабирование хранилища данных, помимо повышения пропускной способности, также защищает от потери данных из-за сбоя жестких дисков. Кроме дублирования данных между центрами обработки данных (ЦОД), также используется технология RAID в масштабе одного сервера. Проблема выхода жесткого диска из строя кажется не столь важной при вероятности выхода из строя 1% за первые 3 года эксплуатации [2]. Но если взглянуть на массив из 10000 дисков, то окажется, что в день ожидается выход из строя одного из них.

Не все аппаратные сбои связаны с проблемой выхода из строя хранилища данных, но именно потеря информации несет серьезные репутационные и финансовые риски [16].

Программные ошибки можно найти, всего лишь изучив кодовую базу приложения, но дьявол кроется в деталях. Факторы

размера кода приложения, квалификации и усидчивости тех, кто этот код проверяет, интеграция со сторонними сервисами и проблема гонок в автоматах (Race Conditions) [3] сводят теоретическую возможность выявить все ошибки еще до запуска кода к нулю. Часто программные ошибки могут оставаться незамеченными в течение долгого времени, потому что требуется определенное стечение обстоятельств, условия для их срабатывания.

Таким образом, для обеспечения надежности разрабатываемой системы как минимум необходимо тестирование логики работы приложения и самого исходного кода, а также резервирование данных и хранение полного лога их изменений. Оба этих требования следует выполнять, начиная с ранних сроков разработки, и встраивать их в общий процесс на постоянной основе.

3.2. МАСШТАБИРУЕМОСТЬ (SCALABILITY)

Выделяют два вида масштабируемости информационных систем: вертикальная (Vertical) и горизонтальная (Horizontal, или Shared Nothing) [32].

При вертикальной масштабируемости возможное повышение пропускной способности (Throughput) и производительности достигается за счет повышения мощности отдельно взятого сервера, например, за счет увеличения объема оперативной памяти или замены процессора на более мощный.

Горизонтальная же масштабируемость предполагает, что при добавлении нового сервера возрастает нагрузка, с которой может справляться система. Вариант английского названия Shared Nothing хорошо отражает, что серверы не разделяют между собой общие ресурсы, такие как время центрального процессора, оперативную память или жесткие диски, т.е. независимы друг от друга.

С одной стороны, давно считается, что вариант повышения производительности одного-единственного сервера, вместо использования нескольких относительно маломощных станций, изжил себя, потому как частота отдельного ядра процессора уже не повышается в значительной степени, как раньше [6].

С другой стороны, сейчас можно наблюдать новые линейки процессоров для настольных компьютеров, которые имеют по 16–18 ядер и соответственно 32–36 потоков выполнения при средней частоте одного ядра 3 ГГц, что значительно превосходит «классические» четырехъядерные процессоры с частотой 3,2–3,6 ГГц. Таким образом, можно заметить, что даже классическое вертикальное масштабирование идет по пути горизонтального масштабирования процессора за счет увеличения количества ядер, нежели увеличения частоты их работы, которая имеет физическое ограничение при текущей архитектуре [21].

Так или иначе рассматривать вариант вертикального масштабирования в текущих реалиях возрастающих требований к отказоустойчивости, пропускной способности, возможности осуществлять обновления без прекращения обслуживания (Rolling Upgrades) становится техническим долгом с самого начала. Что и говорить, если еще Мартин Фаулер в 2002 году писал в своей книге об архитектуре горизонтально масштабируемых систем как о наиболее предпочтительной [20].

Таким образом, разработку монолитных приложений, которые изначально не предполагают возможности своего масштабирования, уже можно смело считать анти-паттерном (Anti-pattern). Сервисная же архитектура, обладающая независимостью развертывания (Deployment) блоков и их горизонтальной масштабируемостью, становится общепринятым подходом при разработке сложных информационных систем. Стоит отметить, что не нужно выстраивать сервисную архитектуру в простых приложениях, где такая «гибкость» принесет больше сложностей, чем пользы.

3.3. ПОДДЕРЖИВАЕМОСТЬ (MAINTAINABILITY)

Помимо критериев надежности и масштабируемости, то, насколько гибка архитектура системы, насколько легко адаптируется к изменяющимся функциональным и техническим требованиям со стороны бизнеса, определяет, сможет ли бизнес быстро перестраиваться и оставаться конкурентоспособным там, где жизненный цикл проектов измеряется годами и даже десятилетиями.

Когда проект переходит в стадию поддержки на 3, 5, 10 лет и редкого добавления нового функционала, становится легаси (Legacy) или же Brownfield [8], у разработчиков, которым приходится поддерживать проект, появляется желание переписать его заново. В некоторых случаях это оправдано. Когда проект использует технологический стек десятилетней давности, становится сложнее найти людей, которые могут и хотели бы с ним работать. Бывает, что проект, живущий в течение длительного времени, требуется дополнить новым функционалом, изменить старомодно выглядящий интерфейс, что влечет переработку сценариев работы серверной части приложения.

Не всегда необходимо переписывать все с нуля. Когда дело касается проектов с монолитной архитектурой [27], то точечная доработка с использованием новых технологий сложна из-за хрупкости дизайна и кода, отсутствия модульности частей системы. В основе сложно расширяемой системы может лежать реляционная база без репликации, где логика приложения опирается на гарантии ACID (Atomicity, Consistency, Isolation, Durability) [7]. Также идеальным условием функционирования такой системы является минимальное количество одновременных пользовательских сессий.

В такой ситуации можно пробовать аккуратно разделить монолит на отдельные сервисы с ограниченной зоной ответственности и возможностью их независимого развертывания. Нет смысла разбивать все сценарии использования приложения на сервисы, но поэтапно, по мере затрагивания частей приложения новым функционалом, это делать необходимо.

Модульная система сервисов или же микросервисный подход [35] обладает несколькими преимуществами, такими как независимое развертывание, слабая связанность, возможность реализовывать решение каждой задачи, используя подходящие инструменты, языки программирования, способы взаимодействия и базы данных [41].

Однако не следует дробить приложение на маленькие сервисы только ради соблюдения принципа единственности ответственности (Single Responsibility Principle) или потому, что так требует подход. Излишняя гранулярность приводит к новым

проблемам поддержки, которые не свойственны монолитной архитектуре. Становится сложно отследить то, как связаны сервисы, потоки выполнения пользовательских запросов, инварианты взаимодействия между версиями развернутых служб, а также обратную совместимость контрактов.

При использовании методологии разработки программного обеспечения DDD (Domain Driven Design) и делении приложения на ограниченные контексты (Bounded Context) возможен переход к одной базе данных на один сервис [33].

На поддерживаемость приложения влияют различные факторы: это и используемые технологии, и культура написания кода, и покрытие тестами, и архитектурные решения, принятые на ранних этапах жизненного цикла проекта. Общего рецепта, как сделать систему поддерживаемой, расширяемой и управляемой, пожалуй, нет. Но управлять сложностью проекта, его техническим долгом, адекватно оценивать будущие точки расширения, проводить непрерывное тестирование абсолютно необходимо для того, чтобы иметь шансы не переписывать код раз в два года.

4. Хранение данных

4.1. NOSQL И SQL

Когда языки программирования, платформы и общий стек технологий проекта определены, встает вопрос выбора подходящего для задачи хранилища данных. Так или иначе, приходится делать выбор между реляционными и NoSQL базами данных.

Принято отсчитывать возникновение понятия реляционной модели данных с момента первого описания в статье Эдгара Кода в 1969 году [12]. Таким образом, вот уже 50 лет ведутся исследования и разработка в области реляционных баз данных. В своей основе они имеют хорошо проработанный теоретический аппарат и язык построения запросов к данным SQL (Structured Query Language), впервые стандартизованный в ANSI (American National Standards Institute) и в ISO (International

Organization for Standardization) в 1986 и 1987 годах соответственно [46].

Термин же NoSQL в его актуальной интерпретации был сформирован в 2009 году [37]. Это направление баз данных ориентировано на масштабируемость, отказоустойчивость, возможность осуществлять большое количество операций записи, и понятие Eventual Consistency (EC) [17] для них имеет определяющее значение. EC подразумевает под собой то, что данные системы при отсутствии обновлений со временем будут согласованы во всех копиях, и сервисы доступа к данным будут возвращать одно и то же последнее записанное значение из любой копии (реплики). Такая гарантия куда слабее, чем гарантии ACID, но подобное поведение позволяет достичь быстрой записи, масштабируемости и отказоустойчивости.

В качестве одного из значимых различий между реляционными и документно-ориентированными базами данных выделяют наличие строгой схемы данных в первом случае и отсутствием таковой во втором [47]. Однако это не совсем так. Реляционные базы данных действительно обладают обязательной схемой, так называемой схемой на запись (Schema-on-Write), в то время как документно-ориентированные базы данных позволяют хранить в себе данные в любом виде, или же неструктурированные данные. Но последнее не предполагает, что в таком виде с данными можно продуктивно работать. Так или иначе, схема присутствует, но в виде схемы на чтение (Schema-on-Read), когда клиентский код ожидает получать данные в определенном формате. Таким образом, схема на чтение является более мягким ограничением, чем схема на запись [29]. Такая свобода с одной стороны открывает возможность поддерживать несколько версий схем данных, осуществлять горячую замену версии развернутых сервисов, а с другой таит в себе опасность несогласованности данных. Более развернутое сравнение рассматриваемых типов баз данных представлено в статье [47].

Однако, как и во многих сферах, рассматриваемых в статье, в области хранения и обработки данных присутствуют тенденции брать лучшее от нескольких миров, создавая гибридные решения. В частности, подход Polyglot Persistence [48] позволяет

не осуществлять выбор только в пользу реляционной базы данных или документно-ориентированного хранилища, а напротив, подбирать различные хранилища данных в рамках одного проекта, исходя из критерия целесообразности. Такой подход убирает строгие рамки и позволяет достичь гибкости, используя для частей приложения лучше всего соответствующие их нуждам и задачам инструменты. Параллельно с Polyglot Persistence свое развитие получает внедрение NoSQL-техник в классические реляционные базы данных, такие как MS SQL Server и PostgreSQL [24]. В частности, появляются новые типы данных, которые могут храниться в колонках, например JSON-документы (JavaScript Object Notation) [25]. Но есть и обратная тенденция, а именно создание SQL-подобных языков доступа к данным в NoSQL-решениях, например, для MongoDB [34].

4.2. OLTP И OLAP

Для того чтобы осознанно выбрать тип хранилища данных, необходимо знать сценарии его использования. Количество операций записи в секунду, основные типы запросов на чтение, типы связей сущностей между собой, цена потери данных, критерии отказоустойчивости – все это ключевым образом влияет на выбор между базами данных и способом организации данных, в частности между OLTP и OLAP.

OLTP (Online Transaction Processing) – способ организации баз данных, при котором система работает с небольшими по размерам транзакциями, но идущими большим потоком, и при этом клиенту требуется от системы минимальное время отклика [39].

OLAP (Online Analytical Processing) – технология обработки данных, заключающаяся в подготовке суммарной (агрегированной) информации на основе больших массивов данных, структурированных по многомерному принципу [38].

Исходя из приведенных базовых определений, для оперативной обработки запросов пользователей используются OLTP-базы данных, а для анализа снимка состояния системы на момент времени (Snapshot) используются OLAP-решения, такие как OLAP-куб [36], Data Warehouse [44] или Data Lake [28].

Использование OLTP-решений характерно для сценариев бизнес-транзакций [20], когда пользовательский ввод инициирует запись в базу данных и выполнение запросов на чтение.

Необходимость построения аналитических отчетов со сканированием большого количества данных, возможно целых таблиц, привело к появлению OLAP-решений, которые поддерживают другой паттерн взаимодействия, нежели OLTP-решения.

Приведем таблицу, позволяющую сравнить паттерны доступа для 2 классов решений [29]:

Таблица 1. Сравнение свойств транзакционной и аналитической систем

Свойство	OLTP	OLAP
Основной паттерн операций чтения	Небольшое количество записей в ответ на запрос по ключу	Агрегирование большого количества записей
Основной паттерн операций записи	Случайный доступ, запись с низкой задержкой в ответ на пользовательский ввод	Импорт большого количества данных или потока событий
Основные клиенты	Конечные пользователи, посредством Web-приложений	Внутренний анализ для поддержки принятия решений
Что представляют собой данные	Последнее состояние данных (в данный момент)	История событий, произошедших с течением времени
Размер массива данных	Гигабайты или терабайты	Терабайты или петабайты

Изначально одни и те же базы данных использовались как для сценариев транзакций, так и для построения аналитики. В этом плане SQL оказался мощным и гибким инструментом. Однако со временем построение аналитик было вынесено в отдельные хранилища данных, так называемые Data Warehouse [14]. Data Warehouse получает данные, собранные из всех имеющихся в компании источников, которые агрегируются, очищаются, преобразуются в удобный для построения ана-

литики формат и загружаются в хранилище без возможности осуществлять редактирование, т.е. в read-only режиме, только для чтения. Описанный процесс загрузки данных называется Extract-Transform-Load (ETL), т.е. делится на этапы: извлечение данных, преобразование и загрузка [29].

В то время как OLTP содержит в основном нормализованные данные без дублирования информации, OLAP-решения достигают своих целей именно за счет денормализации данных. Тем самым количество операций объединения таблиц (Join) сокращается.

Рассмотренные системы призваны решать разные задачи. Объединение задачи построения аналитики и отчетов с задачей выполнения бизнес-транзакций пользователей может быть проще на старте проекта, но давать меньше гибкости в долгосрочной перспективе и сильно влиять на производительность обоих сценариев использования с ростом количества данных. Тем не менее на рынке есть системы, объединяющие в себе оба решения, например, Microsoft SQL Server и SAP Hana, дающие доступ через общий SQL-интерфейс [19, 30].

Может показаться, что деградация производительности системы с ростом количества данных при использовании одного решения вместо двух допустима в разумных пределах. Однако исследования компании Amazon говорят о том, что увеличение времени ответа от сервера лишь на 100 мс снижает выручку на 1% [31], другие исследования указывают на то, что замедление на 1 с снижает уровень удовлетворения покупателей на 16% [10, 18].

Необходимо заботиться о людях, которые работают с вашими системами каждый день в течение всего своего рабочего времени – повышать эффективность, снижая время отклика и генерации отчетов, не блокировать работу системы модальными окнами загрузки и т.д. и т.п.

Принципиально другим вариантом хранения данных, в сравнении с хранением только текущего состояния системы, является хранение потока событий, которые в ней происходили. Этот вариант требует другого мышления и проектирования, но в результате дает возможность получать состояние системы на

любой момент времени, дополнять аналитику, делать ее сколь угодно сложной и глубокой.

4.3. ХРАНЕНИЕ ПОТОКА СОБЫТИЙ ТАБЛИЦЫ

Машинному обучению в целом и его моделям в частности необходимы исходные данные, и чем они обширнее и качественнее, тем результативнее работа. Поэтому важен исходный материал, которого может и не быть при привычном подходе к хранению только последнего состояния имеющихся бизнес-сущностей.

Переход от хранения текущего состояния без истории изменения, которая позволяла бы восстановить данные по состоянию на любой момент существования системы, к хранению потока событий, влияющего на данные, как первоисточника данных приложения, кажется весьма логичным шагом развития технологии и мировоззрения сообщества программистов, архитекторов и бизнеса в широком смысле этого слова. Хранение полной истории изменений дает инструментам работы с данными «пищу для размышления».

Однако помимо перехода к системам, хранящим свое состояние в виде потока событий, также необходим переход от единственного реляционного хранилища данных, либо от хранилища, имеющего синхронную репликацию, к которому сообщество привыкло и от которого даже обрело пагубную зависимость, к распределенному хранилищу с репликацией (Replication) и партиционированием (Partitioning, Sharding), а также к асинхронному распространению изменений и отказу от распределенных транзакций, как средству, обеспечивающему определенные гарантии целостности операций, однако имеющему свою цену в плане общей производительности системы и ее пропускной способности на чтение и запись.

Этот сдвиг массовой парадигмы в разработке приложений кажется таким же необходимым и неизбежным, как когда-то переход от однопроцессорных систем с одним потоком и отсутствием состояния гонок в автоматах к многопоточной модели, которая налагает определенные ограничения, требует большей

внимательности и аккуратности, а также использования средств синхронизации потоков и процессов выполнения.

Стоит заметить, что несмотря на то, что мультипроцессорные системы и программные модели работы с ними существуют уже достаточно давно по меркам информационных технологий, их эффективное освоение разработчиками в массе своей все еще находится на довольно низком уровне. Отчасти это может объясняться инертностью процесса высшего образования, который еще не включает это направление по умолчанию, как обязательную компьютерную грамотность, и отчасти тем, что во множестве компаний потребность в быстрых решениях, рассчитанных лишь на несколько пользователей, с определенными рисками потери данных и высоким временем отклика, которое может считаться не критичным для внутренних корпоративных пользователей, превалирует над более затратными актуальными подходами. И поэтому сотрудникам трудно и не нужно осваивать новое при отсутствии внутреннего спроса на такие навыки.

Безусловно, идею хранения истории изменения данных и получения унаследованного представления из них нельзя отнести к новым. Аналогичный подход имеет место, например, в реляционных базах, где хранится лог транзакций (Transaction Log) и WAL (Write Ahead Log) для индексов. Они позволяют восстанавливать состояние базы данных после сбоев и взаимных блокировок (Deadlocks), а также проводить аудит, но имеют ограниченный размер и зачастую очищаются после определенного промежутка времени, т.е. не хранятся «вечно».

Идея же хранения всего потока изменений и событий, в частности Event Sourcing, не предполагает удаление старых событий для экономии дискового пространства. Напротив, случившиеся события считаются неизменяемыми (Immutable), и это имеет ряд преимуществ. Среди таких преимуществ можно выделить возможность глубокого анализа истории событий в системе, построение аналитики любой сложности за счет наличия полной истории, а не только последнего актуального состояния системы, возможность кеширования событий и т.д.

Переход к накоплению огромных массивов данных поставил задачу их умного анализа, выявления закономерностей

и предсказания поведения систем, на которые прямое воздействие оказывает обратная связь конечных пользователей. Решением этих задач стало практическое применение возрожденного и получившего второе дыхание направления машинного обучения, без которого видится невозможным дальнейшее эффективное ведение бизнеса.

5. Машинное обучение

5.1. АКТУАЛЬНОСТЬ

Применение машинного обучения в буквальном смысле захватило умы в IT. Предсказание спроса на товары, персонализированная таргетированная реклама, выявление мошенничества – это только несколько сфер применения, которые у всех на слуху. Чтобы понять, насколько высок спрос на специалистов в этой области, достаточно анализа данных с сайта hh.ru. Он показывает, что наиболее востребованы сейчас на рынке труда именно навыки и технологии, напрямую связанные с областями больших данных, анализа и машинного обучения [5]. Среди лидеров этого рейтинга: Python, Big Data, Machine Learning, Hadoop, Spark, Data Mining, Deep Learning, Scala.

Ажиотаж вокруг этого, по меркам IT, нового и увлекательного направления, привлекает в него все больше молодых специалистов. Но помимо вчерашних студентов, разработчики со стажем также горят желанием попробовать себя на ниве интеллектуальных систем.

Интерес к очередному «прорывному» фреймворку для построения серверной или клиентской части приложений со временем угасает, появляются новые технологии, и не видно конца и края этой чехарде. Однако в случае с большими данными и машинным обучением это тот новый, недоступный ранее бизнесу инструмент, который дает недосягаемый ранее бонус – новые знания. Поэтому актуальность этого направления будет и дальше возрастать по мере развития моделей и методов.

Анализ исторических данных позволяет понять, на чем бизнес теряет деньги, выявить скрытые тенденции и взаимосвя-

зи, избежать убыточных решений, используя уже имеющийся опыт.

Потоковая обработка новых данных позволяет принимать тактически более взвешенные решения, увеличивать прибыль в ближайшей перспективе, выявлять и предотвращать злонамеренные действия, тем самым сокращая убытки.

Любая новая область требует работников нового вида, и сейчас это специалисты Data Science. Но специалисты из области науки о данных не программные инженеры [13]. Ключевыми навыками инженеров являются программирование и создание программных систем, в то время как базовые компетенции специалиста Data Science (DS) – это математическая статистика, математика в целом, машинное обучение и анализ. Больше всего эти два профиля пересекаются в области больших данных.

Попытки возложить обязанности инженеров на DS-специалистов приводят к потере эффективности. Снижение скорости решения задач анализа данных может достигать 70–80% [13]. Поэтому компании нуждаются в разделении этих сфер, распределении ответственности и набора компетенций, которыми должен обладать сотрудник. Но необходимо связующее звено, и таким соединителем становится инженер машинного обучения. Такого рода специалистами чаще становятся программные инженеры, имеющие склонность к математике и имеющие опыт разработки программного обеспечения и дизайна потоков обработки данных 3–6 лет. Им становится тесно в рамках ставших рутиной задач, и они видят возможность заняться тем, что их всегда манило, но было сложно начать.

Инструментарий сегодняшнего дня позволяет осуществлять переход из программной инженерии в науку о данных постепенно, шаг за шагом углубляя свои познания теории, которая лежит в основе. Процесс такого обучения может занимать годы, но возможно в конечном итоге именно к такому виду программистов мы придем через 5–10 лет.

Во многом взрывной рост популярности различных моделей машинного обучения и распределенных систем работы с большими объемами подчас неструктурированных данных обуславливается прогрессом производительности вычислитель-

ной техники, увеличением количества ядер процессоров, снижением стоимости средств хранения информации, в том числе технологии SSD, а главное широким распространением облачных сервисов, как основы доступных распределенных вычислений.

5.2. ПРИМЕНЕНИЕ

Проблема многих компаний заключается в том, что нет достаточного понимания того, где и как возможно и целесообразно применить методы и модели машинного обучения к имеющимся данным о бизнес-процессах. Также нетривиальной задачей может стать поиск источников информации, из которых необходимо писать историю изменений в долгосрочное хранилище для дальнейшего ее анализа, выявления трендов, опять-таки формирования правильных вопросов, ответы на которые помогут сократить издержки и повысить эффективность работы компании в целом.

Существующие модели машинного обучения, такие как регрессия, классификация, кластеризация и нейронные сети, хорошо зарекомендовали себя сами по себе, но, как и во многих направлениях науки и техники, значительный интерес представляют гибридные модели, те, что создаются на стыке, вбирая в себя все лучшее от нескольких семейств моделей и привнося нечто новое, повышая точность и скорость формирования прогнозов.

В западных и европейских странах, в частности в Нидерландах, уже довольно давно существуют специальности подготовки специалистов именно в области науки о данных. В России же пока не так хорошо акцентирован практический аспект применения получаемых в университете знаний по математической статистике, теории вероятностей, эконометрике, математическому анализу. Такой разрыв между подготовкой специалистов и быстроменяющимися потребностями рынка труда негативно сказывается на темпах автоматизации бизнес-процессов. Кроме этого, не все специалисты IT-сферы на данном этапе ощущают острую необходимость в переобучении, осваивании навыков работы с машинным обучением, большими данными, распреде-

ленными вычислениями, алгоритмами, используемыми в криптовалютах.

Такая закостенелость разработчиков в большей степени определяется инерционностью процессов и однотипностью задач, которые в большинстве своем стоят перед программистами в IT-подразделениях крупных компаний в течение последних 10 лет. Развивающиеся компании, напротив, могут позволить себе быстрее адаптироваться и пробовать внедрять новые модели. Важно отметить, что вчерашние студенты быстрее осваивают новые знания, быстрее добиваются карьерного роста и иногда уже в возрасте 21 года работают на позиции Senior Developer.

Крупные корпорации, в которых устоялись сложные и обладающие большой инертностью бизнес-процессы, для которых информационные технологии и консалтинг в сфере анализа данных не являются основной сферой деятельности, могут испытывать значительные трудности при переходе к новым моделям ведения бизнеса. Несмотря на это, видится жизненно важным переход на качественно иной уровень владения информацией.

Своевременное обновление позволит сохранить конкурентоспособность на рынке, актуализировать используемые технологии и знания для расширения своих возможностей и, в том числе, снижения издержек на поддержку морально устаревших подходов и инструментов.

6. Безопасность

6.1. АКТУАЛЬНОСТЬ

Несмотря на то, что тема безопасности компьютерных систем не привязана лишь к распределенным системам обработки данных, вопросы того, как защитить данные и пользователей от злоумышленников, не только актуальны, но и подкреплены тем, что именно безопасности зачастую уделяется катастрофически мало внимания.

Сжатые сроки и наличие длинного списка новых функциональных требований к создаваемым информационным системам определяют приоритеты при планировании работ. Тема же без-

опасности не часто обладает высоким приоритетом. Также заблуждение о том, что никому в голову не взбредет взламывать создаваемый продукт, приводит к тому, что даже самые простые, хорошо известные уязвимости веб-сервисов, как SQL-инъекции, XSS и CSRF-атаки, не учитываются при разработке.

Хранение паролей в открытом виде в базах данных, неправильное использование хэша и соли пароля (Hash and Salt) [43] и даже вопиющее развертывание боевых баз данных, незащищенных логином и паролем [50], – все это говорит не только о низкой квалификации разработчиков, но и о проблеме в управлении процессом разработки корпоративных приложений.

Хорошим ориентиром для определения того, на какие уязвимости в веб-приложениях необходимо обратить внимание в первую очередь, является Open Web Application Security Project (OWASP) [40].

В случае когда тестирование безопасности приложения имеет самый низкий приоритет, как зачастую и бывает, необходимо коренным образом менять корпоративную культуру и отношение менеджмента к подобным вопросам, до того, как случится серьезное происшествие с утечкой конфиденциальной информации.

Срочной мерой в данном случае может стать приглашение консультанта, который поможет провести аудит разрабатываемых и эксплуатируемых систем. Но что более важно, необходимо обучение собственных специалистов, грамотная расстановка акцентов и ежедневное внимание к аспекту безопасности разрабатываемого функционала и отдельное тестирование, нацеленное не на соответствие функциональным требованиям, а на уязвимость к известным угрозам.

6.2. АУТЕНТИФИКАЦИЯ

Аутентификация, как и авторизация, является неотъемлемой частью любой системы, будь то публичный ресурс или внутрикорпоративный интранет-сервис.

Сложность вопроса реализации входа пользователя в систему и проверки прав доступа к тем или иным ресурсам заключается в том, чтобы обеспечить бесперебойную работу сервисов

при нагрузке, значительно превышающей десять пользователей, работающих в одно и то же время. Иногда и десяти может быть достаточно, чтобы перегрузить плохо спроектированную систему авторизации.

Сейчас довольно сильное распространение получает не самая новая техника входа пользователя в систему SSO (Single Sign On). Это обуславливается тем, что при наличии большого количества сервисов внутри одной компании встает задача использования пользователями лишь одного аккаунта для всех с одним входом и выходом из любого приложения, который бы обеспечил одновременный выход из всех.

Здесь использование технологии Cookies с общим доменом может оказаться весьма полезным, хотя есть и другие техники, такие как переадресация между всеми сервисами на страницы выхода из всех сервисов при выходе из одного или же LDAP (Lightweight Directory Access Protocol) [45].

На данный момент значительную популярность обрела технология аутентификации с использованием JWT-токенов (JSON Web Tokens) [26], которые не должны хранить состояния. Различают два вида токенов: токен доступа (Access) и токен обновления (Refresh).

Токен доступа имеет короткий период жизни, например, 15 минут, в то время как токен обновления может использоваться в течение месяца. Токен обновления передается только в точку доступа (Endpoint) для создания нового токена доступа, когда время его жизни подходит к концу. Кроме метода выпуска нового токена доступа, токен обновления больше не должен никуда передаваться.

JWT не должны хранить состояние пользовательской сессии, т.е. должны быть иммутабельными (Immutable), не изменяемыми после создания. За счет короткого времени жизни токена доступа при использовании защищенной точки доступа можно не сверяться с базой на предмет того, что пользователь не лишен прав или удален. Это значительно снижает накладные расходы и нагрузку на сервис авторизации пользователя. Таким образом можно разрешать одну из основных проблем произво-

дительности при попытке реализации собственного сервиса SSO.

На данный момент, если говорить о Web-приложениях, популярным остается подход с использованием Cookies, также как и сравнительно новый вариант с использованием JWT. Хотя, конечно, JWT могут храниться в Cookies, – это лишь формат описания токена с подписью, гарантирующей его подлинность.

Использование Cookies с общим доменом, хранящего JWT, видится наиболее перспективным вариантом при разработке систем, использующих SSO. При аккуратной настройке и использовании TLS, HttpOnly-Cookies, а также механизмов защиты от CSRF-атак и отсутствии XSS-уязвимостей подобный подход может давать хорошие гарантии безопасности.

7. Вывод

Какие бы направления развития разработки корпоративных информационных систем не рассматривались, будь то разнообразие решений для хранения данных, развитие процессоров, внедрение новых программных архитектур, везде основным трендом является уход от вертикальной масштабируемости и переход к горизонтальной. Масштабируемость вычислений и хранилищ данных, с ростом объемов данных и сложности задач по их анализу, становится краеугольным камнем дальнейшего развития разработки программного обеспечения.

Несмотря на обилие терминов и клишированных фраз, таких как большие данные, машинное обучение, интернет вещей, асинхронность, параллельность, распределенность, их объединяют между собой общие базовые концепции, идеи и проблемы, которые были известны еще в двадцатом веке. Например, такии абстракции, как подтверждение (Acknowledgement) или проблема состояния гонок в автоматах (Race Condition Automaton), существуют в схемотехнике достаточно давно, но современные информационные технологии продолжают их переоткрывать вновь и вновь при появлении новых модных технологий и парадигм разработки.

В этом контексте современные «прорывные» или же «хайповые» направления видятся отчасти переосмыслением уже существующего багажа знаний и оформлением его в новом сверкающем варианте. Отсюда и отсутствие той «серебрянной пули», которая бы позволяла решать все проблемы и задачи в определенной области легко и непринужденно. Ведь создание такого универсального инструмента может потребовать значительного переосмысления подходов к построению информационных систем.

Такая инертность внедрения новых подходов к разработке объясняется сложностью сдвига мышления в сторону потоков событий как основного варианта хранения данных, хотя и этот вариант имеет под собой хорошо известные примеры из прошлого. В частности, таким примером является бухгалтерская книга, в которой записи лишь добавляются, но не исправляются после внесения, а ошибки в предыдущих записях лишь компенсируются новыми строками.

Из-за сложностей идеологического, методологического и технического характера в подавляющем большинстве компаний построение корпоративных систем происходит с использованием подходов, которые апробировались последнее десятилетие.

Можно сказать, что существует целый ряд стандартных задач, таких как документооборот, аутентификация, авторизация, CRUD-операции для различных бизнес-сущностей (Create, Read, Update, Delete), построение отчетов и аналитики и т.д. И решение этих задач тоже уже во многом устоялось, хоть и существует множество вариаций не только с точки зрения технологического стека, но и качества исполнения конечной реализации.

Несмотря на развитие аппаратного обеспечения и информационной инфраструктуры, математическая интерпретация процессов предметной области изменилась не так значительно, по сравнению с изменениями в техносфере.

Например, сегодня много говорится о сенсорных сетях и результатах, которых можно достичь с их использованием для описания тех или иных объектов управления. Но в то же время каким бы ни было количество датчиков на объектах, которыми мы хотим управлять более эффективно, это лишь констатация

динамики изменения состояния объектов управления, не более того. Сами же управляющие механизмы при этом не получили новых прорывных технологий. В сущности этот процесс отображения характеристик объекта управления во времени представляет собой ничто иное, как цифровую тень. И по-прежнему повсеместно лица, принимающие решения, несут ответственность по формированию этих решений, основываясь на опыте и интуиции, а не на адекватных моделях, которые бы позволили эти решения обосновать.

Современные академические изыскания сильно оторваны от практики промышленных предприятий. Теоретические исследования важны, но, к сожалению, очень далеки от нужд практики.

Выход же на сцену такого направления подготовки специалистов IT-сферы, как наука о данных, вполне вероятно поменяет, а точнее уже меняет, направление вектора разработки информационных систем, подготовки специалистов по их созданию и сопровождению, а главное те задачи, решение которых позволяет сделать бизнес более конкурентоспособным, хотя и более уязвимым к атакам, целью которых является получение конфиденциальной информации, которой становится попросту больше за счет увеличения потока генерируемых данных и внедрения новых технологий их хранения. Именно из-за роста уровня гетерогенности инфраструктуры тестирование и обеспечение безопасности являются важными факторами, обеспечивающими работоспособность и устойчивость в современных условиях.

Можно однозначно сказать, что большие данные, наука о данных и развитие направлений масштабируемости и отказоустойчивости информационных систем предопределили развитие информационных технологий в долгосрочной перспективе. Компании, которые не воспользуются всем спектром предоставляемых возможностей, могут остаться на плаву, но максимальную выгоду и лидирующие позиции обеспечат себе лишь те, что инвестируют ресурсы и смогут перестроиться под стремительно меняющиеся реалии.

Литература

1. *Большие данные* https://ru.wikipedia.org/wiki/%D0%91%D0%BE%D0%BB%D1%8C%D1%88%D0%B8%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5 (дата обращения: 11.04.2019).
2. *Время работы жесткого диска* https://www.videomax-server.ru/support/articles/vremya-raboty-zhestkogo-diska (дата обращения: 11.04.2019).
3. *Гонки в автомате* https://studopedia.ru/2_34642_gonki-v-avtomate.html (дата обращения: 12.04.2019).
4. НОВИКОВ Д.А. *Большие данные и большое управление* [Электронный ресурс]. – Режим доступа: https://mipt.ipu.ru/sites/default/files/page_file/BigDataBigControl.pdf (дата обращения: 03.01.2018).
5. *Обзор рынка труда в области Big Data и Data Science* https://habr.com/company/newprolab/blog/320336 (дата обращения: 12.04.2019).
6. *Почему не растет частота* https://habr.com/ru/company/intel/blog/194836 (дата обращения: 11.04.2019).
7. *ACID* https://en.wikipedia.org/wiki/ACID_(computer_science) (дата обращения: 14.04.2019).
8. BALEY K., BELCHAM D. *Brownfield Application Development in .NET* // Manning. – 2010. – P. 416.
9. *Big Data: the Next Google* https://www.nature.com/news/2008/080903/full/455008a.html (дата обращения: 12.04.2019).
10. BRUTLAG J. *Speed Matters for Google Web Search* [Электронный ресурс]. – Режим доступа: http://googleresearch.blogspot.com/2009/06/speed-matters.html (дата обращения: 12.04.2019).
11. CIELEN D., MEYSMAN A., ALI M. *Introducing Data Science* // Manning. – 2016. – P. 320.
12. CODD E. *Derivability Redundancy and Consistency of Relations Stored in Large Data Banks* // IBM Research Report, San Jose, California – 1969. – Vol. RJ599. – P. 15. – DOI: 10.1145/1558334.1558336.

13. *Data Engineers vs. Data Scientist* https://www.oreilly.com/ideas/data-engineers-vs-data-scientists (дата обращения: 12.04.2019).
14. *Data Warehouse* https://en.wikipedia.org/wiki/Data_warehouse (дата обращения: 14.04.2019).
15. *Do NoSQL Database Tend to Become More Relation Over Time or Vice Versa* https://www.quora.com/Do-NoSQL-database-tend-to-become-more-relational-over-time-or-vice-versa (дата обращения: 11.04.2019).
16. *Does a Data Breach Really Affect Your Firm's Reputation* https://www.csoonline.com/article/3019283/does-a-data-breach-really-affect-your-firm-s-reputation.html (дата обращения: 11.04.2019).
17. *Eventual Consistency* https://en.wikipedia.org/wiki/Eventual_consistency (дата обращения: 12.04.2019).
18. EVERTS T. *The Real Cost of Slow Time vs Downtime* [Электронный ресурс]. – Режим доступа: http://www.webperformancetoday.com/2014/11/12/real-cost-slow-time-vs-downtime-slides (дата обращения: 16.02.2018).
19. FARBER F., MAY N., LEHNER W. *The SAP HANA Database – An Architecture Overview* // IEEE Data Engineering Bulletin. – 2012. – Vol. 3, No. 1. – P. 28–33.
20. FOWLER M. *Patterns of Enterprise Application Architecture*. – Addison-Wesley Professional, 2002. – 560 p.
21. GAPNER P., KOWALIK M.F. *Multi-Core Processors: New Way to Achieve High System Performance* // Int. Symposium on Parallel Computing in Electrical Engineering (PARELEC). – 2006. – P. 9–13. – DOI: 10.1109/PARELEC.2006.54.
22. *Google Search Statistic* http://www.internetlivestats.com/google-search-statistics (дата обращения: 11.04.2019).
23. *How Much Data Will the Internet of Things Generate by 2010* https://www.versatek.com/blog/how-much-data-will-the-internet-of-things-iot-generate-by-2020 (дата обращения: 11.04.2019).

24. *Hybrid Databases: Combining Relational and NoSQL* https://www.stratoscale.com/blog/dbaas/hybrid-databases-combining-relational-nosql (дата обращения: 12.04.2019).
25. *JSON* https://www.json.org (дата обращения: 12.04.2019).
26. *JSON Web Token* https://tools.ietf.org/html/rfc7519 (дата обращения: 14.04.2019).
27. KALSKE M., MÄKITALO N., MIKKONEN T. *Challenges When Moving from Monolith to Microservice Architecture* // Int. Conference on Web Engineering (ICWE): Current Trends in Web Engineering. – 2018. – Vol. 10544. – P. 32–47. – DOI: 10.1007/978-3-319-74433-9_3.
28. KHINE P., SHUN WANG Z. *Data Lake: A New Ideology in Big Data Era* // 4th Int. Conference on Wireless Communication and Sensor Network (WCSN) – 2017. – P. 6.
29. KLEPPMANN M. *Designing Data-Intensive Applications*. – O'Reilly Media, 2017. – P. 590.
30. LARSON P., CLINCIU C., FRASER C. *Enhancements to SQL Server Column Stores* // ACM Int. Conference on Management of Data (SIGMOD). – 2013.
31. LINDEN G. *Make Data Useful* // Slides from presentation at Stanford University Data Mining class (CS345) – 2006.
32. LIU C., SHIE M., LEE Y., LIN Y., LAI K. *Vertical/Horizontal Resource Scaling Mechanism for Federated Clouds* // Int. Conference on Information Science & Applications (ICISA) – 2014. – P. 1–4. – DOI: 10.1109/ICISA.2014.6847479.
33. *Microservices* https://martinfowler.com/articles/microservices.html (дата обращения: 11.04.2019).
34. *MongoDB CRUD Operations – Query Documents* https://docs.mongodb.com/manual/tutorial/query-documents (дата обращения: 12.04.2019).
35. NAMIOT D., SNEPS-SNEPPE M. *On Micro-services Architecture* // Int. J. of Open Information Technologies – 2014. – Vol. 2, No. 9. – P. 24–27.

36. NAOUALI S., BEN SALEM S. *Towards Reducing The Multi-dimensionality Of Olap Cubes Using The Evolutionary Algorithms And Factor Analysis Methods* // Int. J. of Data Mining & Knowledge Management Process (IJDKP) – 2016. – Vol. 6, No. 1. – P. 27–38.
37. *NoSQL* https://en.wikipedia.org/wiki/NoSQL (дата обращения: 12.04.2019).
38. *OLAP* https://ru.wikipedia.org/wiki/OLAP (дата обращения: 12.04.2019).
39. *OLTP* https://ru.wikipedia.org/wiki/OLTP (дата обращения: 12.04.2019).
40. *OWASP Top Ten – 2017* https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf (дата обращения: 12.04.2019).
41. RICHTER J. *Architecting Distributed Cloud Applications* [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=xJMbkZvuVO0 (дата обращения: 04.01.2019).
42. ROBINSON H. *The Elephant Was a Trojan Horse: On the Death of Map-Reduce at Google* [Электронный ресурс]. – Режим доступа: https://www.the-paper-trail.org/post/2014-06-25-the-elephant-was-a-trojan-horse-on-the-death-of-map-reduce-at-google (дата обращения: 12.04.2019).
43. *Salted Password Hashing – Doing it Right* https://crackstation.net/hashing-security.htm (дата обращения: 14.04.2019).
44. SINGH S., MALHOTRA S. *Data Warehouse and Its Methods* // Journal of Global Research in Computer Science – 2011. – Vol. 2, No. 5. – P. 113–115.
45. *Single Sign On* https://en.wikipedia.org/wiki/Single_sign-on (дата обращения: 14.04.2019).
46. *SQL* https://en.wikipedia.org/wiki/SQL (дата обращения: 12.04.2019).

47. *SQL vs NoSQL Database Differences Explained with few Example DB* https://www.thegeekstuff.com/2014/01/sql-vs-nosql-db (дата обращения: 12.04.2019).
48. *The Future is Polyglot Persistence* https://martinfowler.com/articles/nosql-intro-original.pdf (дата обращения: 12.04.2019).
49. *Twenty Two Plus Instagram Stats That Marketers Can't Ignore This Year* https://blog.hootsuite.com/instagram-statistics (дата обращения: 11.04.2019).
50. *Uncovering the Impact of the MongoDB Vulnerability* https://www.bitsight.com/blog/uncovering-the-impact-of-the-mongodb-vulnerability (дата обращения: 14.04.2019).

BUILDING OF MODERN ENTERPRISE INFORMATION SYSTEMS

Oleg Loginovskiy, Federal State Autonomous Educational Institution of Higher Education "South Ural State University (national research university)", Chelyabinsk, Doctor of Science, professor (loginovskiyo@mail.ru).
Aleksandr Shestakov, Federal State Autonomous Educational Institution of Higher Education "South Ural State University (national research university)", Chelyabinsk, Doctor of Science, professor (admin@susu.ru).
Aleksandr Shinkarev, Federal State Autonomous Educational Institution of Higher Education "South Ural State University (national research university)", Chelyabinsk, Cand.Sc. (sania.kill@mail.ru).

Abstract: The paper presents an analysis of modern approaches and practices that are used in the building of enterprise information systems. The problems of the isolation of practice from theoretical research are revealed. The main characteristics of big data, solutions for storing enterprise information, the applicability of machine learning, its tools and relevance in modern conditions are considered. Among other things, such necessary aspects of development of information systems as reliability, scalability, maintainability and safety are considered. As a rule, existing systems store only the current state. The event flow is proposed as a method of storing, processing and presenting data for management decision-making in conditions of instability, which is relevant to the current business needs. The reasons behind the complexity of the transition to the flow of events as a storage system are revealed. In light of the changing set of used technologies and the new focus on data analysis, the basic labor market requirements for specialists at the intersection of

programming and data science are being laid down. The main skills and market requirements include work at the intersection of several areas, expertise in deploying applications in cloud services, writing clean code, knowledge of mathematical apparatus at the level necessary for rational use of machine learning models, as well as competent ability to build parallel processing in several streams. The main competitive advantages of business are flexibility and the ability to get the most out of information.

Keywords: enterprise information systems, big data, machine learning, scalability, events flow, analysis.

УДК 62-004.62 + 004.75
ББК 32.972
DOI: https://doi.org/10.25728/ubs.2019.81.5

Статья представлена к публикации
членом редакционной коллегии В.Н. Бурковым.

Поступила в редакцию 23.04.2019.
Опубликована 30.09,2019.