2. **IDF** (*inverse document frequency* — обратная частота документа) — инверсия частоты, с которой некоторое слово встречается в документах коллекции.

$$idf(t, D) = \frac{|D|}{|\{t \in d_i\}|}, \quad (2)$$

где |D| - количество документов в коллекции, $|\{t \in d_i\}|$ – число документов из коллекции D, в которых встречается слово t.

Как результат выполнения функции TF-IDF получаем связный список типа «ключ-значение», где каждому слову присвоен определенный коэффициент. Полученные значения используются для определения сходства текстов с помощью косинус-функции: $cos(a, b) = \frac{(\underline{a},\underline{b})}{|\underline{a}|\times|\underline{b}|}$ (3). Полученное значение и определяет сходство текстов, если оно слишком велико, новый файл не проходит валидацию.

Аналогичные проверки на плагиат планируется реализовать и для остальных типов данных.

Список использованных источников:

1. Jones K. S., A statistical interpretation of term specificity and its application in retrieval.
2. Saul Schleimer, Daniel S. Wilkerson, Alex Aiken Winnowing: Local Algorithms for Document Fingerprinting.

ОБРАБОТКА ПОСТОЯННОГО ПОТОКА ДАННЫХ С АППАРАТНЫХ И ВИРТУАЛЬНЫХ ПЛАТФОРМ CISCO

Губин И.Ю.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Иванов Н.Н. – к.т.н.,доцент

В настоящее время потребности абонентов к пропускной способности сети растут с каждым днём, так стараясь удовлетворить эти потребности, операторы связи ведут активное внедрение новый аппаратных и виртуальных решений, которые могли бы обеспечить обработку большого объема трафика. А при большом количестве оборудования, как аппаратного, так и виртуального, за ним надо наблюдать, получать статистические данные, обрабатывать их и строить прогнозы для лучшей балансировки и реакции на проблемы в моменты пиковой нагрузки. Все это вытекает в сбор, обработку и хранение огромного количество данных.

В ходе доклада будет рассказано, каким образом и какие данные собираются, как с аппаратных, так и виртуальных платформ. Как происходит предобработка и обработка статистических данных оборудования. Как хранятся эти данные и каким образом получается к ним доступ.

Так же будет изложено о пользе сбора и анализа данных, какие выводы можно сделать на основе полученных отчетов, как это может повлиять на коммерческую составляющую, а также как статистические данные помогают избежать аварий и вовремя среагировать при достижения определенных лимитов по ресурсам и много другое.

Будут рассмотрены варианты визуализации данных, посредством Grafana – открытой платформы для анализа и мониторинга, на которой отображаются нужные графики с использованием обработанных данных из баз данных.

Кроме вариантов визуализации, так же будут рассмотрены алгоритмы получения данных напрямую с аппаратной платформы, а также сбор данных после предобработки специализированным ПО.

Будет затронута тема обработки данных на языке программирования Python 3, а также библиотека, предназначенная для работы с большими потоками и объемами данных pandas – это программная библиотека на языке Python для обработки и анализа данных. Работа pandas с данными строится поверх библиотеки NumPy, являющейся инструментом более низкого уровня. Предоставляет специальные структуры данных и операции для манипулирования числовыми таблицами и временными рядами.

В итоге будут сделаны выводы о проделанной работе, полученном опыте при разработке и полезности в применении данного подхода для продакшн проекта. Будет рассказано о плане будущего развития проекта, какие модули будут добавлены по возможности в будущем и простоте расширения.

И в заключении будет сказано об аналитике состоянии сети в целом, о возможности прогнозирования ситуаций и нагрузок на сеть благодаря наличию большого количества данных, применении современных алгоритмов из обработки и машинного обучения. И какую пользу все это может принести как владельцу сервиса, так и пользователям.

Список использованных источников:
1. Маккинни У., Python и анализ данных, 2015. – 482с
2. Cisco ASR 5x00 System Administration Guide, StarOS Release 2x.x, 2018. – 682s
3. Вандер Плас Дж., Python для сложных задач. Наука о данных и машинное обучение, 2018. – 336с
4. Daniel Lee, Visualize Your Data With Grafana, 2017. – 57s

СПОСОБЫ МИГРАЦИИ СХЕМ ДАННЫХ С СОБЛЮДЕНИЕМ ПРИНЦИПОВ ИДЕМПОТЕНТНОСТИ И КОНВЕРГЕНТНОСТИ

Данильчик В.В.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Прытков В.А. – проректор по уч. работе, к.т.н., доцент

Основным направлением изучения способов миграции схем данных из одной структуры в другую является определение оптимальных принципов модификации структур и алгоритма перестроения объектов с учетом существующих зависимостей.

Актуальность анализа способов миграции схем данных обусловлена необходимостью постоянной поддержки структуры базы данных и ее последующей оптимизации при выходе новой версии программного продукта. В настоящее время процессу миграции данных часто сопутствуют проблемы их дублирования и возникновения ошибок, связанных с изменением структуры данных.

При разработке программного продукта и, соответственно, написании кода разработчики пользуются системами контроля версий, такими как Git или Subversion. Данные системы позволяют вести совместный процесс разработки несколькими сотрудниками, хранить только разницу между версиями кода, отслеживать и, при необходимости, возвращаться к конкретной версии исходного кода. В то же время применение систем контроля версий при работе со структурами схем данных довольно ограничено.

В результате в процессе развития программного продукта отследить изменения структуры данных весьма проблематично ввиду того, что DDL, язык определения схем данных, не предусматривает документирование изменения структуры на каждом шаге [1].

Одним из методов решения данной задачи является создание в каталоге проекта пронумерованных скриптов модификации структуры базы данных при этом, не прибегая к использованию дополнительных инструментов. В таком случае все изменения будут накапливаться в проекте при условии соблюдения заранее определенных правил участниками проекта.

Одним из наиболее современных и практичных способов контроля структуры схем данных является использование дополнительного инструмента Liquibase. Основной задачей системы является контроль номера последнего выполненного скрипта изменения структуры данных и обеспечение единовременного линейного выполнения скриптов по изменению структуры данных. Однако, использование инструментов, в основе которых лежит принцип накопления change-скриптов, не позволяет решить задачу так же легко, как написать программный код приложения. Главным минусом данного подхода является необходимость хранения огромного количества неактуальных change-скриптов, что затрудняет получение представления о фактически используемой структуре схемы данных.

Для решения поставленных задач исследованы такие принципы, как конвергентность и идемпотентность. Соблюдение приведенных принципов в разработанном модуле миграции позволяет наиболее эффективно контролировать состояние схемы данных и не допускать повторного выполнения скрипта миграции данных при его успешном запуске ранее.

Наиболее оптимальным решением задачи миграции схем данных и является следование данным принципам: идемпотентности и конвергентности [2]. Отличительной особенностью такого подхода является то, что скрипт, который удовлетворяет условиям принципов, описывает состояние, к которому объект должен быть приведен, а не действия, которые требуется произвести над ним.

Идемпотентность обеспечивает отсутствие изменений объекта при выполнении скрипта повторно в случае, если скрипт был выполнен успешно при первом его запуске. Соблюдение данного принципа позволяет избежать дублирования и искажения данных. Однако, в случае невыполнения скрипта или неудачного завершения его выполнения, система, при повторном выполнении данного