При агрегировании исключаются повторяющиеся новости, полученные из разных источников, в зависимости от их приоритета.

Фильтр новостей по категориям, разделам, темам или тегам позволяет ограничить сбор данных по установленным критериям, подгрузив только необходимую информацию.

Последним шагом настройки данной системы является установление связи между структурой полученных данных и структурой новостей на сайте – так называемый маппинг.

Полученная система – довольна гибкая и легко дополняемая. В целом можно выделить следующие преимущества:

–- значительная экономия времени в процессе сбора новостей;

–- уменьшение загруженности команды контент-менеджеров путём автоматизации части их рабочего процесса или исключение потребности в наёме отдельных работников, чьей обязанностью был бы сбор данных из внешних новостных источников;

–- расширяемость, возможность доработки функционала, добавления сбора информации из другого типа источников или внедрение анализа получаемого контента.

Таким образом, разработанная система позволяет значительно облегчить работу с контентом информационно-новостных порталов городов. Она может использоваться для сбора информации другого рода с целью публикации полученных данных на любом портале. Также актуально внедрение данной программы в виде отдельного модуля для систем управления содержимым сайта.

Список использованных источников:

1. Информационный онлайн-портал: цели и особенности – [Электронный ресурс] – http://jnetwork.kz/blog/informacionnyy-onlayn-portal-celi-i-osobennosti

2. Новостные агрегаторы как инструмент оптимизации – [Электронный ресурс] – http://xn--h1aieep.xn--p1ai/novostnye-agregatory-v-pomoshh-optimizatoru

АРХИТЕКТУРА СОБЫТИЙНОГО ПРЕДСТАВЛЕНИЯ ДАННЫХ

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

Кравцов Д.В.

Волорова Н.А. – к.т.н., доцент

В настоящее время большинство приложений работают с данными, и типичный подход заключается в том, чтобы приложение хранило текущее состояние данных, обновляя его тогда, когда пользователи взаимодействуют с ним. Например, в традиционной модели создания, чтения, обновления и удаления (CRUD – Create, Read, Update, Delete) типичным процессом обработки данных является чтение данных из хранилища, внесение некоторых изменений и обновление текущего состояния данных новыми значениями - часто используя транзакции, которые блокируют данные. Такой подход имеет ряд ограничений:

- системы CRUD выполняют операции обновления непосредственно в хранилище данных, что может замедлить производительность и оперативность, а также ограничить масштабируемость из-за требуемых затрат на обработку;
- при совместной работе со многими параллельными пользователями конфликты обновления данных более вероятны, поскольку операции обновления выполняются на одном элементе данных;
- если нет дополнительного механизма аудита, который записывает детали каждой операции в отдельный журнал, история теряется.

Архитектура EventSourcing определяет подход к обработке операций над данными, которые являются результатом последовательности событий, каждое из которых записывается в хранилище. Приложение отправляет ряд событий, которые однозначно описывают каждое действие, которое произошло в системе, в хранилище событий, где они сохраняются [1, 2].

Типичное использование событий заключается в том, чтобы генерировать представления сущностей одновременно с изменением их в системе, а также для интеграции с внешними системами.

Архитектура Event Sourcing предоставляет следующие преимущества:

- события неизменяемы и могут только добавляться в хранилище, что позволяет значительно повысить производительность и масштабируемость приложений, особенно для уровня представления или пользовательского интерфейса;
- события- это простые объекты вместе с любыми связанными данными, необходимыми для описания действия, представленного событием. События не обновляются непосредственно в хранилище данных, а просто записываются для обработки в соответствующее время, что может упростить внедрение и управление;
- события обычно имеют смысл для эксперта домена, тогда как несоответствие объектно-реляционного представления может затруднить понимание сложных таблиц базы данных;
- хранилище событий обеспечивает журнал, который может использоваться для мониторинга действий, выполненных в хранилище данных, восстановления текущего состояния в виде

материализованных представлений или прогнозов путем повторного воспроизведения событий в любое время и оказания помощи в тестировании и отладке системы. Список событий также может использоваться для анализа производительности приложений и выявления тенденций поведения пользователей или для получения другой полезной бизнес-информации.

Данный подход обычно используется тогда, когда необходимо минимизировать или полностью избежать конфликтов при обновлении данных, а также когда особую важность играет возможность воспроизводить происходящие в системе события, чтобы восстановить состояние системы, отменить изменения или сохранить журнал истории и аудита.

Очень часто EventSourcing используется вместе с шаблоном CQRS (Command-QueryResponsibilitySegregation– разделение ответственности на команды и запросы), что позволяет разделить процессы записи и чтения данных, увеличивая производительность и надежность системы [3].

Список использованных источников:

1. Martin Fowler, Development of Further Patterns of Enterprise Application Architecture

2. Event Sourcing Pattern [Электронный ресурс] / Christopher Bennag. – 23 июня 2017. – Режим доступа: https://docs.microsoft.com/en-us/azure/architecture/patterns/event-sourcing.

3. Betts, D. Exploring CQRS and Event Sourcing: A journey into high scalability, availability, and maintainability with Windows Azure. / D. Betts. – Microsoft, 2013. – 213 p.

ПРОГРАММНОЕ СРЕДСТВО ОБФУСКАЦИИ VHDL-КОДА

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

Крагель Н.А.

Иванюк А.А.. – профессор, д.т.н., доцент

Одной из важных проблем, возникающих при проектировании цифровых устройств является защита интеллектуальной собственности. Существуют различные способы ее обеспечения и доказательства авторских прав: лексические и функциональные обфускации проектных описаний, внедрение водяных знаков и др.

Обфускация - приведение исходного текста проектных описаний к виду, сохраняющему функциональность, но затрудняющему анализ, понимание алгоритмов работы и модификацию.

Цели обфускации:

Затруднение изучения и модификации проектных описаний;

Усложнение реверс инжиниринга проектных описаний.

Целью создания программы была автоматизация обфускации VHDL-кода с поддержкой основных конструкций языка.

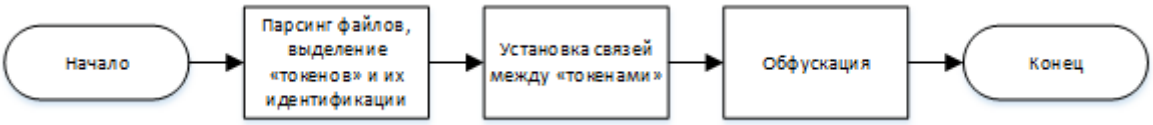

Рис. 1. Основные этапы работы программы.

Процесс работы программы представлен на рисунке 1. На первом этапе исходные файлы разбиваются на «токены». На втором этапе происходит инициализация «токенов» и установление связей между ними. Таким образом после этого этапа имеется полное программное представление загруженных исходных файлов. Рассмотрим подробнее шаг обфускации с внедрением водяного знака. Для обфускации генерируются новые имена для переменных, содержащие только знаки «1», «0», «O», «I» и «L». Можно выделить 2 группы элементов:

Элементы визуально похожие на «1»: «L», «I»;

Элементы визуально похожие на «0»: «O».

Распределение данных групп в рамках генерируемых названий нормальное. Однако можно управлять распределение элементов в рамках группы, увеличивая частоту использования одного символа и уменьшая частоту другого.

В рамках реализованной программы обфускации подлежат все переменные, сигналы, сущности и порты. Перед процессом обфускации были выделены все связи, что теперь позволяет пройтись по всем элементам, объявленным в загруженных файлах. Такой подход позволяет избежать проблем, возникающих, когда были загружены не все исходные файлы и некоторая часть структурных элементов была объявлена в отсутствующих файлах.

Ниже представлен пример работы программы.