ПРИМЕНЕНИЕ ЛОГ-ОРИЕНТИРОВАННОГО ПОДХОДА ПРИ ПОСТРОЕНИИ ВЫСОКОНАГРУЖЕННЫХ СИСТЕМ ДИСТАНЦИОННОГО ОБУЧЕНИЯ

Н.А. Хмурович[1], С.С. Куликов[2]

[1] Белорусский государственный университет информатики и радиоэлектроники, Минск, Беларусь, dev.nkh@gmail.com
[2] Белорусский государственный университет информатики и радиоэлектроники, Минск, Беларусь, kulikov@bsuir.by

Abstract. This article describes goals and objectives of educational platforms, their classic storage implementation and the log-oriented storage alternative usage with event sourced domain logic implementation.

Создание универсальных платформ для управления дистанционным обучением имеет большие перспективы на рынке программных продуктов. Дистанционное обучение предполагает самостоятельное изучение материала удалённо и наличие периодического контроля процесса и результатов освоения. Действия пользователей в такой платформе представляют собой ценность с точки зрения анализа процесса обучения как отдельного учащегося, так и тенденций в обучении групп людей.

Современные веб-приложения обрабатывают большое количество запросов и действий пользователей, в таких приложениях принято оперировать последним состоянием сущностей конкретной предметной области. В то же время, в большом количестве предметных областей, предполагающих использование веб-приложений, количество и сложность операций чтения многократно превышает аналогичные показатели операций записи. Большинство операций записи обновляют лишь несколько полей сущности, или добавляет/удаляет сущность. Аналитические же запросы, предполагающие чтение данных, являются значительно более сложными с точки зрения количества операций, необходимых СУБД для их выполнения.

До сих пор общепринятым подходом является создание и реализация архитектуры веб-приложений с ориентацией на особенности поведения конкретной выбранной СУБД. Явным примером является повсеместное использование реляционных баз данных с высокими уровнями нормализации данных, что в свою очередь является оптимизацией записи и замедляет чтение. Денормализация данных для оптимизации чтения не имеет единых формализованных подходов и поэтому часто приводит к искажению исходных данных. Использование единственной базы данных часто ведёт к повышению стохастической сложности [1] системы, связанной с ограничениями выбранных платформ и моделей, возникающими на пути её решения, и не связанной непосредственно с решением ключевых проблем предметной области.

В современных веб-приложениях мы имеем дело с множеством запросов и действий пользователей. Такого рода информация имеет ценность с точки зрения поведенческого анализа субъекта.

При росте количества пользователей важной задачей становится скорость реакции приложений на действия пользователей и способность приложений строить произвольные аналитические запросы. Отсюда проистекает требование максимальной производительности современных веб-приложений при работе с большими объёмами информации. В то же время, задача удовлетворения запросов по историческим данным классически решается с помощью дополнительной поддержки побочного аудит-лога по определённым действиям пользователей в системе.

Актуальным решением для работы с большими объёмами данных является принцип Polyglot Persistence [2], пропагандирующий использование множества

хранилищ различного типа для решения конкретных задач. Однако при таком подходе проблемой становится синхронизация данных и поддержание их в целостном виде в различных хранилищах. В распределённых системах с множеством копий данных эта задача обретает ещё большую актуальность. Целесообразным решением этой задачи является наличие единого источника истины [3] – эталонного источника происходящих в системе событий, данные которого можно однозначно спроецировать в базу данных произвольного типа для дальнейшей работы с данными.

Целесообразным является применение подхода регистрации событии [4] и проектирование слоя бизнес-логики приложения таким образом, чтобы все мутации данных выражались через явные типизированные события и могли быть воспроизведены позже. При таком подходе все события сохраняются в виде последовательного потока данных в едином логе. Сохранение исходных причин изменения состояния сущностей системы также даёт возможность строить произвольные проекции по историческим данным. Наличие исходных данных позволяет строить данные для запросов, неизвестных на момент разработки бизнес-логики приложения. Функции аудита операций могут быть реализованы без изменения формата хранения данных.

В такой архитектуре выбор, добавление и удаление конкретной СУБД из инфраструктуры может происходить постепенно с течением времени и с изменениями требований. Бизнес-логика приложения зависит от упрощённого контракта хранилища событий, который может реализовывать хранилища различных поставщиков. Для конкретно выбранной СУБД реализуется модель чтения, которая представляет собой логику проецирования зарегистрированных событий в данную СУБД и реализацию запросов, которую данная модель может удовлетворить. В подобной архитектуре все хранилища становятся согласованными в конечном счёте [5].

На основании действий пользователей могут быть определены различные метрики, такие как количество времени, проведённого обучаемым за изучением теоретического материала по дисциплине, количество и распределение потраченного времени на выполнение различного рода заданий, своевременность начала выполнения учебных заданий. Становится возможным анализ вовлечения обучаемых различных групп в образовательный процесс.

Вопрос применения лог-ориентированного подхода при разработке систем дистанционного обучения зависит от поставленных задач и конечной аудитории пользователей. Регистрация событий в едином логе предоставляет неоспоримые преимущества с точки зрения долгосрочного развития продукта.

Литература

1. Essential and accidential complexity - Mark Needham, http://codebetter.com/markneedham/2010/03/18/essential-and-accidental-complexity
2. Polyglot Persistence - Martin Fowler, http://martinfowler.com/bliki/PolyglotPersistence.html
3. Single Source of Truth - Wikipedia, https://en.wikipedia.org/wiki/Single_Source_of_Truth
4. Why use event sourcing - Greg Young, http://codebetter.com/gregyoung/2010/02/20/why-use-event-sourcing/
5. BASE: An ACID alternative - Dan Pritchett, http://queue.acm.org/detail.cfm?id=1394128