*Н. С. Дудаков, Н. Е. Пирогов, Ю. Ю. Шумилов*

# КРОССПЛАТФОРМЕННАЯ СИСТЕМА БЕЗОПАСНОГО УПРАВЛЕНИЯ ХРАНЕНИЕМ ДИНАМИЧЕСКИХ ДАННЫХ

Одним из наиболее важных вопросов взаимодействия в информационной среде является управление доступом и использованием информации. Обработка и хранение данных также являются важнейшими задачами. Для оперативного, гибкого и эффективного управления работой с информацией широко внедряются системы автоматизированного управления, ядром которых являются базы данных. При большом объеме информации и сложности производимых операций проблема эффективности средств организации хранения, доступа и обработки данных приобретает особое значение. Современная система управления хранением данных характеризуется следующими ключевыми параметрами [1]:

- скорость доступа к данным на чтение и запись;
- отказоустойчивость;
- удобство доступа к данным;
- безопасность работы с информацией.

В настоящее время безопасность и контроль целостности данных приобретают все большее значение. В современной распределенной системе данных доступ к информации одновременно получает большое число пользователей, наделенных различными правами. Система защиты хранилища данных должна решать две принципиальные задачи:

- обеспечивать разделение прав пользователей при доступе к информации баз данных;
- осуществлять безопасную передачу данных по локальной сети.

На настоящий момент существует большое количество систем управления данными, удовлетворяющих различным запросам. При решении конкретной задачи наиболее целесообразным является выделение подходящих существующих систем, анализ и определение наиболее подходящей системы и ее дальнейшая доработка для полного соответствия требованиям задачи.

В то же время, проектируя автоматизированные системы управления данными, зачастую невозможно обеспечить все требования к готовому продукту одной готовой системой управления хранением данных. В этих случаях целесообразно разделять данные и использовать взаимодействующие в той или иной степени существующие программные решения.

Используемая в большинстве информационных систем архитектура «клиент—сервер», предполагающая централизованное хранение данных, зачастую имеет существенные недостатки. К ним можно отнести большую нагрузку на сервер и высокие требования к надежности и пропускной способности каналов связи между клиентами и сервером. Подход, реализуемый в локальных базах данных, — хранение копии базы данных на каждом рабочем месте — решает указанные проблемы, однако появляется необходимость решения задачи синхронизации информации в клиентских и серверных копиях базы данных, а также обеспечения мониторинга работы системы, т. е. возможности отслеживать производимые в базе данных изменения.

В системе, построенной на локальных базах данных, при подключении к серверу клиентское рабочее место зачастую запрашивает полную копию базы данных, что при наличии на клиенте устаревшей копии базы данных является избыточным, так как большая часть информации базы данных статична и не меняется в течение длительного периода времени.

Актуальным и эффективным является разделение данных по темпам обновления и доступа на сравнительно более медленную «статическую» и быструю «динамическую» части. Статические данные характеризуются низкой частотой обновления и доступа, к ним можно отнести классификационные

данные, различную справочную информацию. Как правило, информация запрашивается разово при инициализации задач, использующих ее. В то же время статическая информация имеет достаточно большой объем, и хранить ее целесообразно на серверах баз данных. Динамические данные, в свою очередь, характеризуются высоким темпом обновления, а также необходимостью отслеживания изменений в режиме реального времени. При этом динамическая часть имеет сравнительно меньший объем данных и хранится как на сервере, так и на локальных рабочих местах. При такой конфигурации все запросы на чтение информации производятся из локальной копии базы данных, что существенно разгружает канал передачи информации к серверу.

В работе решается задача разработки системы безопасного управления хранением динамических данных. Систему можно разделить на несколько компонентов, «ядром» проектируемой системы является база данных, непосредственно содержащая хранимую информацию. В качестве основы для проектируемой системы предлагалось использовать готовое программное решение, которое будет являться «фундаментом» для хранилища динамических данных. Необходимо дополнительно реализовать механизмы резервирования, разграничения доступа, защиты данных, журналирования и документирования изменений, тиражирования данных по рабочим местам.

Для работы с динамическими данными в качестве базового механизма предлагается использовать одну из локальных систем управления базами данных или одну из баз данных, хранящихся только в оперативной памяти. Соответственно, можно выделить две основные задачи:

1. Выбор локальной базы данных в качестве основы для проектируемой системы;
2. Реализация взаимодействия баз данных при работе в локальной сети.

Для решения первой задачи необходимо осуществить обзор локальных баз данных, формирование требований к программным решениям и выбор наиболее подходящего в качестве основы для проектируемой системы. Выбор локальной базы данных обусловлен, в первую очередь, большой загруженностью канала передачи информации в системе «клиент—сервер». Работа проводилась с динамическими данными, имеющими высокие темпы обновления и доступа: предполагалось, что периодичность запросов на запись составляет порядка 5 секунд, частота запросов на чтение не ограничена. При клиент-серверной архитектуре системы база данных хранится на сервере, соответственно, серверное программное обеспечение обрабатывает запросы как на чтение, так и на модификацию информации и передает запрошенные данные на клиентские рабочие места. При большом количестве подключенных клиентов обработка запросов на чтение информации из базы данных может занять недопустимое время, а пересылка данных по запросам — увеличить нагрузку на сеть, снижая эффективность работы всей системы, а также делая ее более уязвимой для потенциальных атак.

Учитывая сравнительно небольшой объем динамических данных, при проектировании системы предлагалась конфигурация, согласно которой копии базы данных хранятся одновременно как на серверах, так и на рабочих местах. Запросы на чтение осуществляются из локальных баз данных, а все изменения согласовываются с сервером и тиражируются по подключенным рабочим местам. В этом случае серверное программное обеспечение обрабатывает только запросы на модификацию данных, но более трудоемкой является задача поддержания целостности данных и синхронизации данных на рабочих местах.

Соответственно, при выборе «фундамента» — базы данных — необходимо учитывать особенности проектируемой системы и формировать требования к рассматриваемым базам с учетом этих особенностей.

Были сформированы следующие требования к программному решению:

- поддержка нескольких платформ, операционных систем;
- возможность доступа к хранилищу несколькими процессами, потоками одновременно;

• удобная и гибкая модель доступа к данным и интерфейс управления хранилищем;

• наличие исходных кодов проекта.

В соответствии с предъявленными требованиями, а также по результатам серии исследований и тестов была выбрана база данных SQLite — встраиваемая библиотека, поддерживающая стандарт SQL 92. Интерфейс SQLite реализован в одной библиотеке, а сами данные можно хранить в единственном файле. При этом SQLite является производительной базой данных благодаря высокоупорядоченной внутренней архитектуре.

Особенностью SQLite является также отсутствие встроенных средств администрирования. Первичные средства администрирования поддерживаются организацией прав доступа в самой файловой системе. Кроме того, SQLite является открытой базой данных и используется во многих свободно распространяемых программах, например в интернет-браузерах [2].

При создании прототипа системы, основанной на локальной базе данных SQLite, авторами разработана схема взаимодействия баз данных в локальной сети. Предлагалась двух- или трехуровневая система хранения данных. Первым уровнем являются подключенные рабочие места, вторым — серверная часть хранилища. В случае группы серверов можно выделить третий уровень — центральный сервер баз данных, динамически определяемый в процессе работы.

База данных хранится в одном файле, параллельная запись в базу менее надежна в связи с возможными блокировками процессов, поэтому более целесообразно осуществлять запись в один поток с сохранением очередности. Также в целях большей безопасности при передаче информации по сети было решено минимизировать количество каналов передачи данных в системе.

Реализацию общего взаимодействия локальных баз данных и поддержку клиент-серверной архитектуры хранилища осуществляют модули менеджера хранилища динамических данных. Модули — работающие приложения, которые запускаются на каждой машине системы в единственном экземпляре. Обмен информацией по сети осуществляется только между ними. Основная задача модулей менеджера хранилища — синхронизация информации локальных баз данных и поддержка ее целостности.

Все сетевое взаимодействие в системе осуществляется через сокетные каналы связи. При этом модули менеджера данных выполняют централизующие функции, пользовательские приложения подключаются только к менеджерам данных, функционирующим на рабочих местах. В свою очередь, они обмениваются информацией с аналогичными модулями, работающими на серверах. Передача информации осуществляется по защищенным каналам связи, также обмен информацией между модулями системы позволяет применять любые дополнительные алгоритмы кодирования данных для обеспечения большей безопасности соединения.

Реализация механизмов доступа к данным и разграничение прав доступа осуществляются за счет использования клиентского интерфейса доступа к динамическим данным. Интерфейс обеспечивает единообразный доступ к данным независимо от методов выбранной базы данных SQLite, что также позволяет изменять системную часть управления хранением данных без модификации созданных модулей. Для предотвращения рассогласования форматов структура базы данных вынесена в отдельный файл и загружается при запуске системы. Кроссплатформенность созданной системы достигается за счет использования библиотеки Qt при проектировании модулей [3].

Таким образом, рассмотрена система безопасного управления хранением динамических данных. В качестве «фундамента» системы предложен вариант использования локальной базы данных SQLite, разработаны модули системы, осуществляющие взаимодействие локальных баз данных и реализующие клиент-серверную архитектуру системы. Передача информации осуществляется только между модулями системы, пользователи не имеют прямого доступа ни к файлу базы данных, ни к серверу, что позволяет минимизировать количество защищенных каналов связи. Разграничение

прав доступа реализуется за счет единого интерфейса доступа к хранилищу данных, библиотеки, содержащей все необходимые методы. При этом система сохраняет всю функциональность и производительность исходной базы данных SQLite и является эффективным решением для хранения быстро меняющихся «динамических» данных с поддержанием политики безопасности.

## СПИСОК ЛИТЕРАТУРЫ:

1. *Дейт К. Дж.* Введение в системы баз данных. М., 2001.
2. Документация по базе данных SQLite. URL: www.sqlite.org.
3. *Шлее М.* Qt4: Профессиональное программирование на C++. СПб., 2007.