ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 2 за 2012 г.

Китаев Е.Л., Кузьмичев Д.Л.,
Слепенков М.И., Богданова В.М.

Мониторинг изменений в базах данных на основе анализа журнала транзакций

Рекомендуемая форма библиографической ссылки: Мониторинг изменений в базах данных на основе анализа журнала транзакций / Е.Л.Китаев [и др.] // Препринты ИПМ им. М.В.Келдыша. 2012. № 2. 14 с. URL: http://library.keldysh.ru/preprint.asp?id=2012-2

О р д е н а Л е н и н а
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В.Келдыша
Р о с с и й с к о й а к а д е м и и н а у к

Е.Л.Китаев, Д.Л.Кузьмичев, М.И.Слепенков, В.М.Богданова

Мониторинг изменений в базах данных на основе анализа журнала транзакций

Москва — 2012

Китаев Е.Л., Кузьмичев Д.Л., Слепенков М.И., Богданова В.М.

Мониторинг изменений в базах данных на основе анализа журнала транзакций

В современных информационных системах протоколирование выполняемых пользователями операций обычно решается на уровне сервера приложений с помощью журнала событий, регистрирующего события в терминах предметной области, а не на уровне SQL-операций. Однако существуют события, которые в таком журнале не отражаются, но способны нарушить целостность критически важной для бизнеса информации, поэтому необходим аудит изменений на уровне базы данных. В статье представлен подход к реализации системы мониторинга изменений в базе данных, основанный на возможности извлечения и последующей обработки информации о выполненных операциях, имеющейся в журнале транзакций.

Ключевые слова: аудит базы данных, журнал транзакций, информационная безопасность, Oracle Streams, сервер контроля изменений.

Evgeny Lvovich Kitaev, Dmitry Leonidovich Kuzmichov, Mikhail Ivanovich Slepenkov, Vera Mikhailovna Bogdanova

Monitoring changes in database by analyzing transaction log

In modern information systems, logging operations performed by users is usually solved at the application server using an event log that records the event in terms of subject area, rather than the SQL-level operations. However, there are events that are not reflected in this log, but they can damage the integrity of business-critical information, so you need to audit changes on the database level. The paper presents an approach to the implementation of monitoring changes in the database, based on the possibility of extraction and subsequent processing of information on completed transactions available in the transaction log.

Key words: Database audit, transaction log, information security, Oracle Streams, data change control server.

Введение

Современный подход к реализации информационных систем, как правило, основывается на использовании трехуровневой архитектуры, в которой предполагается наличие Клиентского рабочего места (обычно это «тонкий» или «толстый» Клиент – терминал), подключенного к Серверу приложений, и Базы данных (БД), где хранится информация. В такой архитектуре задача протоколирования выполняемых пользователями операций решается на уровне Сервера приложений. Для этого, например, в прикладной системе может быть реализована служебная компонента для ведения Журнала событий (операций), в котором регистрируются все значимые действия по просмотру и модификации информации. Преимущество такого подхода к протоколированию состоит в том, что он позволяет регистрировать события в терминах предметной области, а не на уровне SQL-операций над таблицами БД.

Казалось бы, что подобная компонента, фиксирующая действия пользователей, решает все проблемы мониторинга операций, выполняемых в БД информационной системы, и нет необходимости включения дополнительных средств мониторинга на уровне СУБД. Однако это не так, поскольку существуют события, которые в таком журнале не отражаются.

Цели мониторинга на уровне СУБД

Во-первых, у любой прикладной системы обычно имеется администратор, сопровождающий ее с помощью SQL-интерфейса, и его действия, как правило, в журнале событий прикладной системы не отражаются. В результате на уровне приложения не остается никаких следов, позволяющих обнаружить возможные несанкционированные действия.

Во-вторых, в самой прикладной системе могут быть уязвимости, позволяющие ее взломать, например, при помощи популярного среди хакеров метода SQL-injection, что может привести к непредсказуемым изменениям в базе, которые не будут зафиксированы в журнале событий.

В-третьих, часто возникает необходимость выполнения прямых изменений в базе данных посредством SQL-команд, минуя интерфейс прикладной системы. Например, имеются прикладные системы, в задачи которых входит ежедневное выполнение расчетов определенных показателей (финансовых обязательств, биржевых котировок, прогнозных показателей и т.п.) на основе нормативных и\или оперативных данных, загружаемых из внешних источников. Бывают ситуации, когда перед расчетом выясняется, что поступившие в систему данные некорректны, а времени на их повторную загрузку уже нет, и в этом случае ошибки обычно исправляются вручную путем прямых изменений в базе. Очевидно, что такие действия необходимо контролировать, поскольку велика вероятность, что исправление будет неверным. В самих системах ведение этих данных не предусмотрено, ведь они поступают извне. Учтем также, что время проведения расчета строго

регламентировано, и он обязательно должен быть выполнен ежедневно к определенному часу. При этом важно отслеживать, что все изменения произведены до запуска расчета, а не в процессе его выполнения или окончания.

Перечисленные выше случаи (а любой профессионал в этой области легко может дополнить данный список) свидетельствуют о том, что одного журнала событий прикладной системы недостаточно — необходим аудит изменения данных на уровне базы.

Несмотря на инструментальную роль, отводимую базам данных в архитектуре современных информационных систем, когда подавляющее большинство пользователей не используют SQL и вообще ничего не знают о схеме базы и структуре таблиц, с которыми они работают, применение средств мониторинга изменений на уровне базы оказывается полезным дополнительным инструментом для обеспечения безопасности и исключения ошибок при эксплуатации системы. Основным критерием для выбора механизма, обеспечивающего мониторинг, является минимальное влияние внедряемого решения на производительность системы.

Мониторинг с помощью триггеров

Наиболее известный способ мониторинга изменений — использование механизма триггеров, с помощью которых можно осуществлять перехват и протоколирование операций, производимых в базе: модификацию данных (DML-операции: insert, update, delete) и модификацию схемы (DDL-операции: create table, alter table add column и т.п.). Этот метод также действует во встроенных механизмах аудита современных СУБД (например, при использовании команды AUDIT в Oracle), где система в автоматическом режиме создает и актуализирует набор триггеров, соответствующий текущим правилам контроля объектов базы, типов операций и т.п.

Основным недостатком применения триггеров и встроенного аудита для мониторинга изменений является заметное падение производительности системы, поскольку в этом случае каждая операция в транзакции дополнительно сопровождается добавлением записи в таблицу аудита. Реально использовать встроенный аудит СУБД возможно только в том случае, когда удается заранее установить селективные условия аудита (например, настроить фильтр на протоколирование данных из небольшого подмножества таблиц базы с целью обеспечения обозримого объема выборки записей для анализа). Если же подключить к аудиту все операции над базой или большую их часть, то это сильно повлияет на производительность прикладной системы, которая начинает резко падать. Выполнять подобные операции на «боевой» системе, находящейся под постоянной или временами пиковой нагрузкой, обычно неприемлемо.

Мониторинг по журналу транзакций

С учетом названных ограничений получил развитие альтернативный подход к реализации мониторинга изменений, основанный на возможности извлечения и последующей обработки информации об операциях DML и DDL, имеющейся в журнале транзакций базы. Для «большой тройки» СУБД — Oracle, MS SQL Server и IBM DB2 — можно назвать следующие промышленные системы аудита, предлагаемые на рынке в виде отдельных продуктов: Oracle Audit Vault [1], Apex SQL Log [2] и IBM Audit Management Expert [3]. Эти системы обладают возможностями захвата (capture) изменений из журналов транзакций и позволяют аудитору со своего рабочего места удаленно подключаться к журналам транзакций целевых баз (включая архивы журналов), задавать разнообразные фильтры на выборку интересующих его записей, получать удобные для анализа отчеты в различных форматах и пр.

Общим недостатком этих систем, несмотря на различия в их архитектуре, является то, что захват изменений осуществляется непосредственно на целевой базе. Этот процесс является достаточно «тяжелым», поскольку связан с парсингом и трансформацией данных. Хотя по сравнению с подключением встроенного аудита СУБД способ захвата нагружает систему не так значительно, тем не менее и это не всегда можно делать на исходной базе, особенно когда прикладная система активно выполняет свою целевую функцию.

Дополнительно следует отметить, что мониторинг, выполняемый по журналу транзакций, позволяет извлекать информацию о давно произведенных операциях в базе. Если изменения по каким-то объектам БД до этого не отслеживались, то аудитор, тем не менее, может запросить историю модификации любого из них за прошлый период по журналу транзакций. Очевидно, что эта функциональность очень полезна и востребована, но ее поддержка на сервере прикладной системы в критической степени сказывается на производительности системы, поскольку исполнение такого рода запросов вызывает необходимость ресурсоемкого сканирования большого количества (до нескольких тысяч) архивных файлов журнала транзакций.

Архитектура Сервера контроля изменений

Более рациональным решением является вариант, когда серверы с базами данных информационной системы полностью избавляются от «непрофильной» работы, а выполнение задач по захвату, обработке и сохранению информации об изменениях осуществляется на специально выделенном сервере. Предлагаемая нами архитектура системы такого класса схематически представлена на рис. 1.

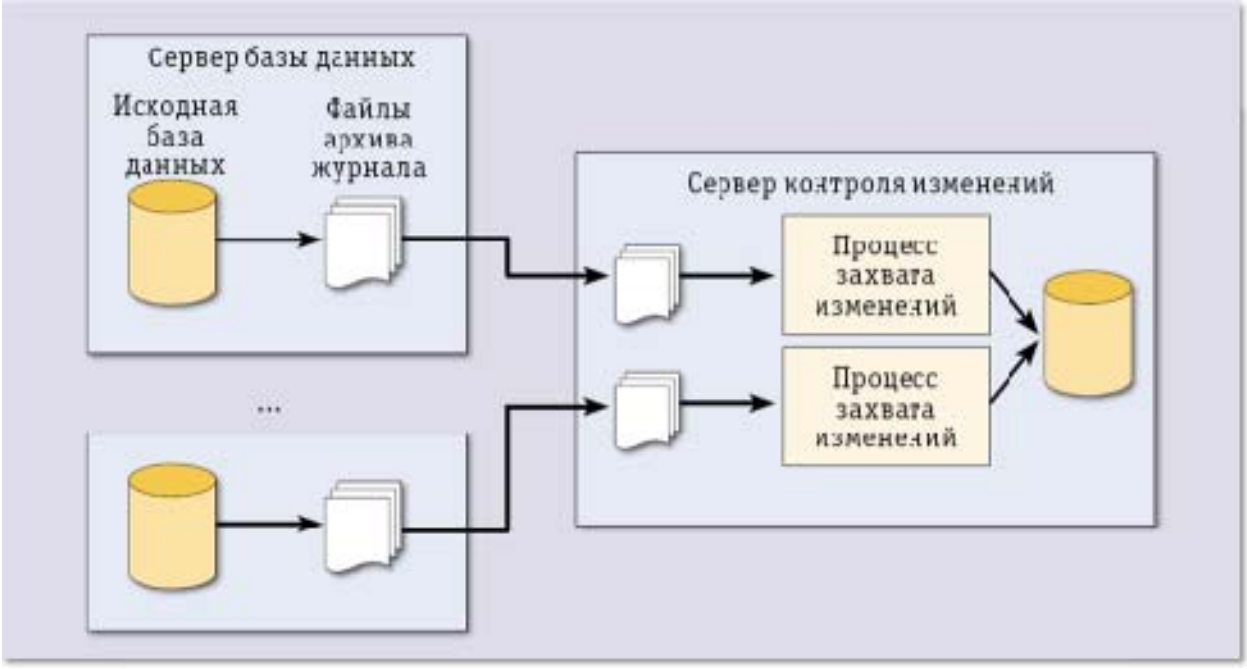

Рис. 1. Архитектура Сервера контроля изменений (СКИ)

Центральным звеном представленной архитектуры является Сервер контроля изменений (СКИ), на котором выполняется процесс захвата и обработки записей из архивных журналов, поступающих с серверов, где размещены исходные базы. Предполагается, что любая СУБД поддерживает механизм архивирования журналов транзакций, применяемый для решения задач резервного копирования и восстановления данных. Именно эти архивные журналы передаются на СКИ и там обрабатываются, формируя структурированное представление о прочитанном из журнала изменении, которое в виде отдельной записи (или группы записей) может быть сохранено в базе СКИ.

Сервер контроля изменений может обслуживать несколько исходных баз, для каждой из которых порождается свой процесс захвата и обработки изменений. Основной особенностью предлагаемой архитектуры является то, что сами исходные базы на сервер СКИ не перемещаются, а метаинформация о структуре базы, составляющая содержимое словаря данных (место размещения названий таблиц, полей и т.п.) может быть получена из архивного журнала или каким-то иным способом передана на СКИ. Не имея этой метаинформации, а она является неотъемлемой частью любой БД и используется при чтении файлов журнала, анализировать записи журналов не представляется возможным, так как они закодированы (например, вместо имен полей используются их номера).

К числу дополнительных преимуществ, вытекающих из описанной архитектуры, относятся: ее экономическая целесообразность (для реализации подобной системы достаточно одного современного сервера обработки данных); возможность получения интегральной информации (например, в виде

отчетов) об изменениях, произведенных в нескольких источниках; организация централизованного долговременного хранения файлов архивных журналов и т.п.

Пример реализации СКИ средствами Oracle

Далее, чтобы подтвердить работоспособность предлагаемых идей, мы представим возможную реализацию СКИ на базе технологий, которые предоставляет Oracle (в версии 11g R2). На сегодняшний день это, пожалуй, единственная компания, обеспечивающая разработчиков практически полным набором штатных инструментальных средств для решения задач мониторинга. Алгоритмы консолидации изменений, происходящих и транслируемых из исходных баз данных, основываются на механизме Oracle Streams, предназначенном для распространения данных на базе очередей сообщений Oracle Advanced Queuing (AQ). Этот механизм достаточно универсален, с его помощью можно выполнять: репликацию данных; резервное копирование (Standby); загрузку хранилищ данных из прикладных систем; управление событиями и т.п.

В СУБД Oracle используется журнал транзакций фиксированного размера, называемый также оперативным журналом повтора (online redo log). Рабочий объем журнала определяется при конфигурировании базы путем указания количества файлов журнала (не менее двух) и их размера. Выделение свободного места для записи генерируемого системой потока транзакций осуществляется по циклической схеме. Записи последовательно помещаются в один из файлов журнала (активный файл), по достижении конца которого производится переключение (logfile switching) на следующий по порядку файл, либо на начало первого файла, если был заполнен последний из файлов журнала. Таким образом, оперативный журнал транзакций в Oracle может содержать только небольшой временной диапазон записей, а информация о транзакциях, выходящих за границы этого диапазона, теряется — новые транзакции записываются «поверх» старых.

Для реализации возможности сохранения информации о транзакциях за продолжительный период времени, а также для передачи этой информации в другие системы, в базе необходимо включить режим ARCHIVELOG, активирующий механизм генерации архивных журналов (archived logs). При работе в таком режиме система осуществляет копирование заполненных файлов оперативного журнала в архивные журналы. Гарантируется, что до перезаписи файла система сделает копию его предыдущего содержимого.

В Oracle для любой базы может быть порождено несколько архивных журналов (но не более 10) для решения различных прикладных задач. Каждый архивный журнал представляет собой папку в файловой системе, куда сохраняются копии заполненных файлов оперативного журнала транзакций. Папки архивных журналов могут размещаться не только в локальной файловой

системе, а и на удаленных устройствах. Обычно архивные журналы создаются для бэкапирования или для зеркального отображения данных на резервный сервер (standby-сервер). Для передачи файлов журнала между основным и резервным серверами в Oracle имеется встроенный сервис Log Transport Service (процесс, осуществляющий транспортировку файлов журнала), который выполняет синхронизацию между серверами путем копирования log-файлов в определенную папку на резервном компьютере. Для подключения базы к Oracle Streams следует создать отдельный архивный журнал, который будет передаваться на целевой downstream-сервер СКИ. Это рекомендуется сделать, чтобы избежать конфликтов со штатными процедурами архивирования, если таковые имеются.

При организации подключения базы к Oracle Streams требуется определить способ транспортировки файлов журнала на целевой downstream-сервер. Для этого можно использовать либо штатный сервис Log Transport Service, либо реализовать свое решение для передачи файлов.

Транспортировка архивных журналов

Наш опыт работы с Log Transport Service показал, что этот сервис имеет ряд принципиальных недостатков, которые ставят под сомнение целесообразность его использования в качестве транспортного средства для передачи файлов журнала транзакций. Во-первых, метод аутентификации, используемый сервисом, требует совпадения паролей для учетной записи SYS на исходной и целевой БД. Данное требование необходимо при использовании сервиса транспортировки для организации зеркалирования, поскольку очевидно, что в этом случае вся учетная информация должна быть идентична. С другой стороны, при подключении к downstream-серверу это ограничение нелогично и не отвечает требованиям безопасности.

Во-вторых, Log Transport Service не обеспечивает гарантированную доставку файлов, в результате чего в журнале на целевом сервере могут возникать разрывы (gap) в последовательности отгруженных файлов. Это объясняется push-режимом, лежащим в основе архитектуры данного процесса. Если по каким-то причинам ему не удается передать файл (например, целевой компьютер выключен или с ним разорвано соединение), то процесс после нескольких неудачных попыток просто «забывает» про передаваемый файл и переключается на следующий. Такие разрывы приходится устранять вручную с помощью копий журнала, в которых следует отыскать все потерянные файлы и скопировать их на целевой downstream-сервер.

Мы рекомендуем альтернативный подход для организации транспортировки файлов журнала на целевой downstream-сервер СКИ, который основан на использовании pool-технологии.

При подключении исходной базы к системе мониторинга в ней необходимо создать отдельный архивный журнал в папке файловой системы

этого сервера. Эту папку следует сделать разделяемой (shared folder) и открыть к ней доступ по сети. На целевом downstream-сервере СКИ должен быть реализован и запущен процесс Log_File_Reader, который опрашивает удаленную папку и «забирает» из нее новые файлы. При подключении других внешних источников с архивными журналами каждый из них будет копироваться в свою папку на целевом downstream-сервере. Таким образом, любой архивный журнал исходной базы будет представлен своей копией, размещаемой в отдельном каталоге downstream-сервера. После того как файл успешно передан, он удаляется из папки первоисточника, чтобы освободить место на диске и предотвратить его переполнение. Разумеется, что для работоспособности описанной схемы у разделяемой папки-первоисточника должны быть установлены права на чтение/запись/удаление файлов для процесса Log_File_Reader. Изложенная схема по транспортировке файлов журнала обладает одним существенным преимуществом — она устраняет возможность появления разрывов в последовательности загружаемых файлов.

Захват изменений процессами Oracle Streams

После того как вопрос с транспортировкой данных решен, требуется описать основные понятия и процессы Oracle Streams, которые позволяют выполнять обработку файлов журнала транзакций.

Oracle Streams состоит из трех основных процессов:

- Capture Process — захват изменений в рабочих и архивных журналах транзакций, выбор из них записей об изменениях в исходной базе (CR, Change Record) и формирование логических записей изменений (LCR, Logical Change Record), помещаемых в очередь сообщений AQ;
- Propagate Process — передача сообщений из очереди исходной базы в очередь на целевой базе;
- Apply Process — применение изменений из очереди LCR к таблицам целевой базы, либо их передача специальной обрабатывающей программе (handler) для выполнения необходимых преобразований.

В Oracle 11g R2 реализован удобный комбинированный способ обработки журналов транзакций Combined Capture Apply, позволяющий напрямую (через служебную очередь) организовать взаимодействие между процессами Capture и Apply (рис. 2).

После загрузки очередного log-файла в папку downstream-сервера процесс Log_File_Reader выполняет регистрацию этого файла для Capture процесса путем выполнения соответствующей команды: ALTER DATABASE REGISTER LOGICAL LOGFILE <имя файла> FOR <имя Capture процесса>.

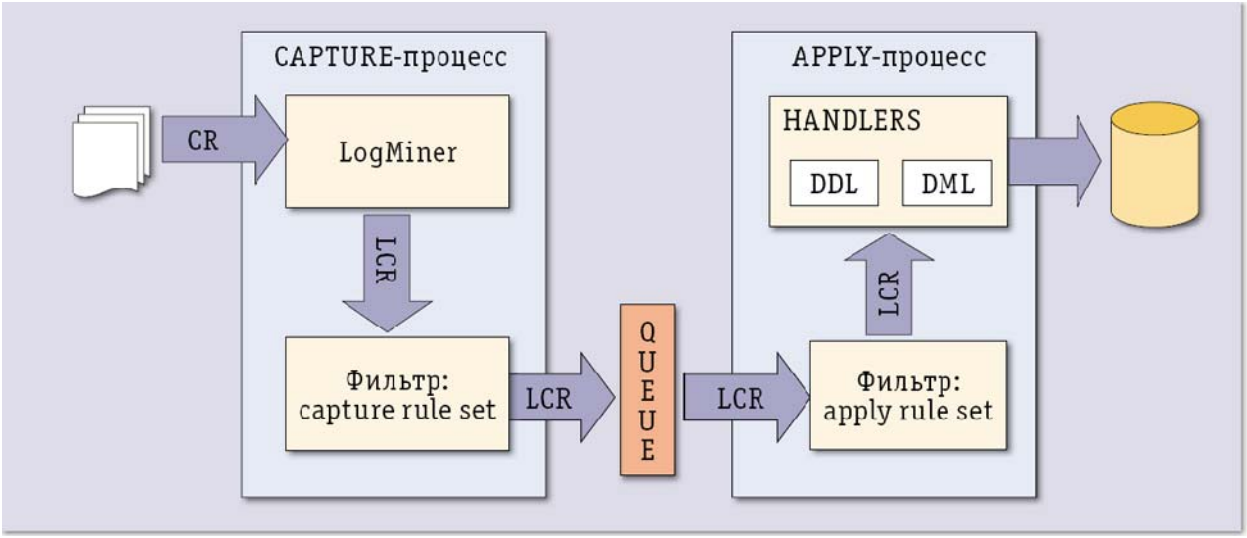

Рис. 2. Схема обработки журналов транзакций механизма Combined Capture Apply

Процесс захвата изменений (Capture Process) является фоновым процессом, который постоянно просматривает архивный журнал транзакций, выбирая оттуда вновь зарегистрированные log-файлы. Следует отметить, что для обработки каждого журнала в системе должен быть создан свой Capture процесс. Обработка log-файлов выполняется при помощи вызова штатной утилиты Oracle LogMiner, которая считывает записи об изменениях (CR) и преобразует их в логические записи об изменениях — LCR. Для выполнения этой трансформации LogMiner использует словарь данных исходной базы, выгружаемый в журнал транзакций специальной процедурой BUILD из системного пакета DBMS_CAPTURE_ADM. Дополнительно процедура BUILD возвращает SCN (System Change Number) — системный номер записи в журнале, начиная с которого в log-файле будет располагаться выгруженный словарь данных. Именно этот SCN указывается при регистрации Capture процесса. Можно считать, что именно с этого момента в СКИ будут доступны все изменения, которые производились в базе прикладной системы.

На уровне Capture процесса задаются правила, с помощью которых определяются схема и набор таблиц баз данных, подлежащих мониторингу. Соответственно, только изменения объектов, перечисленные в правилах, будут передаваться в очередь Apply процесса в виде DML и DDL операций модификации в данных и схеме базы. В процессе эксплуатации системы набор правил может изменяться путем добавления в них новых объектов, подлежащих аудиту, или исключения тех, мониторинг которых утратил актуальность.

Далее Apply процесс извлекает из очереди входящих сообщений LCR-записи и передает их обрабатывающей прикладной программе (handler). Предварительно выбираемые записи могут быть отфильтрованы по аналогичным правилам, которые задаются для Capture процесса. В зависимости

от типа операции вызывается либо DDL_handler, либо DML_handler. Обрабатывающая программа разбирает LCR, распознавая указанный там тип операции, и сохраняет ее в базе данных СКИ. Например, в обработчике DML_handler, в который передаются сообщения об изменении отдельных записей таблицы, следует различать и отдельным способом обрабатывать информацию о командах INSERT (когда в LCR представлены только новые значения), UPDATE (когда в LCR имеются как новые, так и старые значения) и DELETE.

Многие технические детали в изложении процесса захвата опущены (подробности можно найти в документации). Хотя реализация описанной схемы может потребовать определенных усилий, тем не менее, предложенный Oracle инструментарий вполне работоспособен. На наш взгляд, главным ограничением существующей версии Streams является отсутствие штатного доступа к словарю данных, который был получен из журнала транзакций. Из-за этого при построении пользовательских сервисов над базой СКИ приходится обращаться к недокументированным системным таблицам для обеспечения конечных пользователей возможностью выбора объектов схемы (названий таблиц, полей и т.п.) из списка.

Возможности управления мониторингом

Представленная технология открывает возможности для реализации на сервере СКИ корпоративного сервиса, предоставляющего конечным пользователям возможность самостоятельно управлять процессом аудита путем явного указания объектов, подлежащих мониторингу. На основании подключенных к аудиту баз данных пользователи сервиса смогут определять состав таблиц, для которых будут протоколироваться операции по изменению данных и схемы. С этой целью каждый аудитор создает свой профиль, содержащий набор правил, которые определяют интересующие его изменения. Система объединяет правила, указанные во всех профилях, формируя интегральный фильтр, который будет применяться для отбора данных из архивных журналов при захвате изменений.

При задании конкретного правила в своем профиле пользователь может дополнительно указать дату, начиная с которой он хочет получить соответствующие изменения, произведенные в базе прикладной системы. Если такая дата указана, то это инструктирует систему о необходимости перезагрузки файлов журнала задним числом, но не ранее момента подключения базы к мониторингу. Это возможно, если считать, что все архивные журналы-первоисточники сохранены в файловой системе СКИ.

Интересы аудиторов могут меняться в процессе эксплуатации системы мониторинга — в нее могут добавляться новые правила или удаляться существующие. Система должна отслеживать эти модификации, перестраивая интегральный фильтр, и если оказывается, что часть сохраненных в СКИ

данных больше не востребована, то они удаляются из базы. Кроме того, при определении любого правила можно указать период в виде количества дней до текущей даты, который определяет время хранения отобранной по данному правилу информации в базе СКИ. Автоматический способ сокращения объемов хранимой в базе СКИ информации заметно влияет на повышение производительности системы, особенно в части получения отчетов.

Роль администратора в такой системе сводится лишь к подключению исходных баз к системе мониторинга, а все задачи управления номенклатурой загружаемой в СКИ информации полностью возлагаются на пользователей-аудиторов, поскольку именно они обладают знанием предметной области и могут точно определить, какие изменения необходимо контролировать.

Заключение

Описанный подход к организации мониторинга изменений баз данных ориентирован на создание корпоративного сервиса, позволяющего производить централизованный аудит всех критических изменений, которые были выполнены в прикладных системах организации. Это особенно важно, когда прикладные системы тесно взаимодействуют друг с другом и изменения в структуре базы, составе данных одних из них могут повлечь изменения в работе, вплоть до отказа, смежных систем.

Несмотря на то, что все технические аспекты излагались нами на основе продуктов Oracle, представленное решение является универсальным и может быть реализовано и в других СУБД. Единственной проблемой, на которую следует обратить внимание разработчиков в этом случае, будет отсутствие в составе таких платформ как MS SQL Server и IBM DB2 инструментальных средств уровня LogMiner и Streams, позволяющих реализовать данное решение с минимальным объемом программирования.

Авторы выражают признательность Александру Лашманову, заместителю председателя правления по ИТ ОАО "Администратор торговой системы оптового рынка электроэнергии", являющемуся одним из авторов проекта по созданию сервера СКИ, за помощь в подготовке данной публикации.

Литература

[1] Oracle Audit Vault http://www.oracle.com/technetwork/database/audit-vault/index.html

[2] ApexSQL Log http://www.apexsql.com/sql_tools_log.aspx

[3] IBM DB2 Audit Management Expert http://www-01.ibm.com/software/data/db2imstools/ db2tools/db2ame/

Оглавление