УДК 004.658.6

ОРГАНИЗАЦИЯ ХРАНИЛИЩ ДАННЫХ ДЛЯ РАСПРЕДЕЛЕННОЙ СИСТЕМЫ АНАЛИЗА ВИБРАЦИОННЫХ СИГНАЛОВ

В.Э. БАЗАРЕВСКИЙ

Белорусский государственный университет информатики и радиоэлектроники
П. Бровки, 6, Минск, 220013, Беларусь

Поступила в редакцию 27 июня 2013

Рассмотрены способы организации хранения длинных реализаций вибрационных сигналов, а также способы комплексного анализа больших объемов сигнальных данных. Приведены способы и методы организации сигнального и аналитического хранилищ данных для системы поддержки принятия решений, оценивающей вибрационные характеристики исследуемых объектов. Рассмотрены способы расширения и встраивания сторонней функциональности в СУБД MS SQL Server.

Ключевые слова: цифровая обработка сигналов, базы данных, аналитические хранилища данных, извлечение и очистка данных, MS SQL Server.

Введение

Развитие средств сбора и передачи информации позволяет хранить и обрабатывать все большие объемы данных. Удешевление устройств хранения данных (таких как жесткие диски) позволяет осуществлять практически непрерывный мониторинг исследуемых объектов или явлений. Однако хранение все большего объема данных напрямую не решает задач диагностики и анализа состояния исследуемых объектов, так как зачастую данные, получаемые в режиме реального времени, не могут быть полноценно проанализированы в процессе оцифровки и записи на носитель информации. А их неструктурированность, в свою очередь, не позволяет производить комплексный анализ всех собранных данных ввиду их большого объема.

Существует несколько подходов для решения проблемы комплексного анализа больших объемов данных:

– MapReduce – модель распределенных вычислений, ориентированная на комплексный анализ больших объемов данных в асинхронном режиме [1];

– аналитическое хранилище данных – набор моделей обработки и организации структурированных данных для комплексного анализа больших объемов данных [2].

Каждый из подходов имеет свои преимущества, если для модели MapReduce – это отсутствие жестких требований к структурированности данных и практически неограниченная масштабируемость решения, то для аналитического хранилища данных – это возможность анализа данных в реальном времени и отсутствие необходимости разработки и программирования сложных алгоритмов анализа данных для аналитика (но при предварительной необходимости настройки и оптимизации аналитического хранилища данных).

Так как вибрационные сигналы, подлежащие обработке и анализу более структурированы (в сравнении с текстовыми данными, аудио- или видеосигналами), и по ним можно сравнительно быстро построить вектор признаков (среднее квадратическое значение, максимальное значение сигнала и т. д.), а объем получаемых данных в рамках одной организации все же не достигает петабайт, то наиболее приемлемым и адекватным решением представляется разработка аналитического хранилища данных [3].

Разработка аналитического хранилища данных для вибрационных сигналов предполагает разработку ряда компонентов, взаимодействующих между собой:

1) компонент первоначального сбора исходных сигналов;
2) хранилище исходных сигналов (файловое хранилище);
3) компонент преобразования сигналов в векторы признаков;
4) хранилище векторов признаков;
5) компонент анализа векторов признаков;
6) компонент визуализации и просмотра отдельного сигнала;
7) компонент визуализации результатов анализа векторов признаков.

Диаграмма компонентов и их взаимодействия между собой показана на рис. 1.

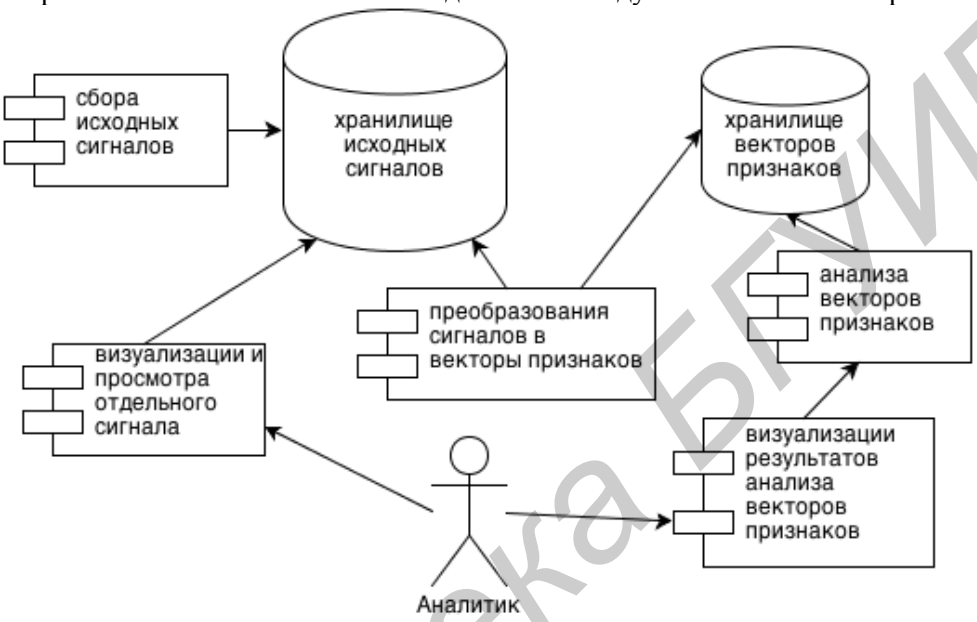

Рис. 1. Компонентная диаграмма аналитического хранилища данных с надстройкой для хранения, обработки и анализа вибросигналов

Следует обратить внимание, что большинство описываемых компонентов системы (хранилище исходных сигналов, компонент преобразования сигналов в векторы признаков, хранилище векторов признаков, компонент анализа векторов признаков и частично компонент визуализации результатов анализа векторов признаков) в значительной мере реализованы в рамках BI-систем ведущими поставщикам программного обеспечения, такими как Microsoft, Oracle, IBM, а также рядом открытых свободно распространяемых проектов (Mondrian, PALO и др.).

Одним из наиболее популярных и функциональных решений на рынке BI систем является комплекс MS SQL Server [4], представляющий помимо реляционной модели хранения данных также ряд компонентов для загрузки, очистки, трансформации и анализа данных. К преимуществам решения на базе MS SQL Server можно отнести нацеленность данной разработки непосредственно на сектор корпоративных приложений, интегрированность с рядом других программных разработок от Microsoft (Excel, Sharepoint, Dynamics CRM), кроме того, ряд предприятий уже имеет лицензии для данной программной системы, соответственно внедрение еще одного решения на базе данной системы не повлечет за собой дополнительных финансовых издержек. Кроме того, существует бесплатная версия MS SQL Server, предоставляющая хоть и усеченный набор функционала, но достаточный для минимальной работы предложенной системы. В сравнении с открытыми BI-решениями, выбор MS SQL Server обусловлен большей надежностью данной многократно проверенной и отлаженной системы по сравнению с зачастую недостаточно зрелыми открытыми разработками, и наличием исчерпывающей документации по архитектуре и настройке системы. В табл. 1 указано соответствие между функциональными компонентами системы, показанными на рис. 1 и компонентами MS SQL Server.

Таблица 1. **Соответствие между функциональными компонентами системы и компонентами MS SQL Server**

Компонент системы	Компонент MS SQL Server
Компонент первоначального сбора исходных сигналов	Отсутствует
Хранилище исходных сигналов (файловое хранилище)	FileStream
Компонент преобразования сигналов в векторы признаков	Integration Services
Хранилище векторов признаков	MS SQL Server database
Компонент анализа векторов признаков	Analysis Services
Компонент визуализации сигналов	Отсутствует
Компонент визуализации результатов анализа векторов признаков	Частично представлен компонентами визуализации алгоритмов Analysis Services

Хранилище бинарных сигнальных данных

При хранении неструктурированного содержимого (видео, аудио, документы и т.д.) в реляционной базе классически рассматриваются две опции:

– хранение содержимого в бинарных объектных хранилищах;

– разделенное хранение содержимого, когда скалярные атрибуты содержимого (тип, название, автор, путь в файловой системе) хранятся в реляционной таблице, а само содержимое находится в файловой системе.

Каждый из указанных подходов имеет свои недостатки. В первом случае – это тяжелая нагрузка на журнал сервера баз данных, ограниченность размера хранимого содержимого в 2 ГБ; во втором случае нет никакой связи между таблицей со скалярными атрибутами и файлами, на которые она ссылается. Файлы можно переносить, переименовывать, удалять, при этом сервер баз данных не может отслеживать указанные изменения – происходит потеря целостности.

Появившийся в SQL Server 2008 функционал FileStream позволяет совместить оба подхода, преодолев недостатки каждого: хранить бинарные объекты в файловой системе, обеспечив клиенту к ним доступ в виде потоков. С другой стороны, вся работа с файлами происходит под полным контролем сервера баз данных, который обеспечивает по ним транзакционность, резервное копирование, полнотекстовый поиск (при необходимости), репликацию, логирование, кластеризацию и др. FileStream доступен во всех изданиях MS SQL Server, включая Express.

Помимо указанных преимуществ, за счет того, что вся работа с хранимым содержимым происходит непосредственно через средства MS SQL Server, механизм FileStream позволяет встроить ряд специфичных сигнальных преобразований непосредственно в сервер баз данных. Такой подход унифицирует процесс анализа хранимых сигналов, а также облегчает получение векторов признаков сигналов, а соответственно и непосредственное создание аналитического хранилища данных.

Механизм встроенных функций MS SQL Server позволяет разрабатывать собственные пользовательские функции на любом из языков, поддерживаемых платформой .NET с возможностью их дальнейшего использования в SQL-запросах к базе данных. На рис. 2 показан пример использования встроенной библиотеки сигнальных преобразований DST (Digital Signal Transforms) для получения среднеквадратического значения сигнальной выборки.

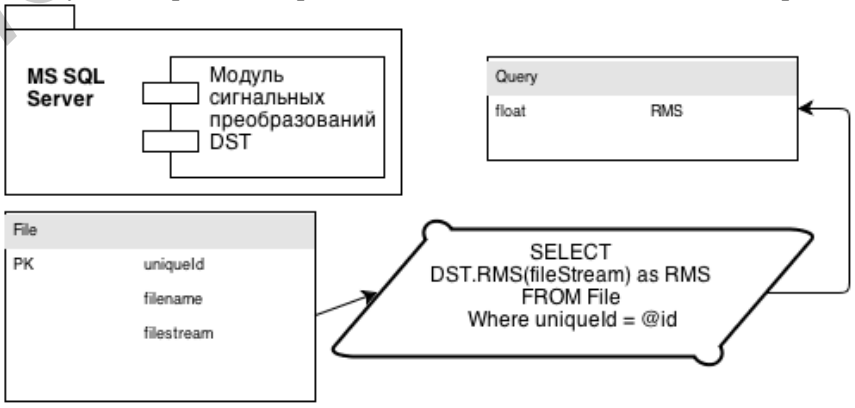

Рис. 2. Получение среднеквадратического значения сигнальной выборки средствами MS SQL Server

На рис. 2 показана встроенная библиотека DST, написанная на C#, и содержащая функции обработки бинарных сигнальных выборок. Одной из таких функций является RMS – получение среднеквадратичного значения выборки [5].

Все выборки хранятся в таблице File и содержат помимо бинарных данных так же сведения об имени и формате хранимых данных: частоту дискретизации сигнала, точность хранимых отсчетов, количество хранимых каналов и т.д. С помощью SQL-запроса, содержащего вызов пользовательской функции, можно получить вектор признаков сигнала, содержащий с том числе и среднеквадратическое значение сигнала, как показано на рисунке.

Организация взаимодействия хранилищ данных системы

Одной из особенностей процесса построения хранилищ данных является способ организации данных в них, в зависимости от их целей. Существует два основных способа организации данных.

1. OLTP (англ. Online Transaction Processing, транзакционная система) – обработка транзакций в реальном времени. Способ организации БД, при котором система работает с небольшими по размерам транзакциями, но идущими большим потоком, и при этом клиенту требуется от системы минимальное время отклика.

2. OLAP (англ. Online Analytical Processing, аналитическая обработка в реальном времени) – технология обработки данных, заключающаяся в подготовке суммарной (агрегированной) информации на основе больших массивов данных, структурированных по многомерному принципу.

Таким образом, если OLTP-подход характерен для процесса сбора данных, то использование OLAP-подхода необходимо для процесса комплексного анализа собранных данных (в основном векторов признаков). Таким образом, в качестве основных требований для организации единой системы сбора и анализа вибрационных характеристик объектов можно выделить следующие:

– система должна состоять из двух различных по архитектуре и назначению хранилищ данных;

– оба хранилища данных должны быть изолированы друг от друга (для того, чтобы повышенная нагрузка на одно из хранилищ не сказывалось на работе другого);

– должен быть разработан легковесный процесс выгрузки, очистки, преобразования и загрузки данных из OLTP в OLAP хранилище;

– должен быть разработан процесс обработки ошибок автоматической загрузки данных из одного хранилища в другое;

– должен быть разработан механизм определения временного среза, на основании которого впоследствии проводится загрузка данных из OLTP и в OLAP-хранилище.

Диаграмма потока данных, построенного на основе указанных требований, показана на рис. 3.

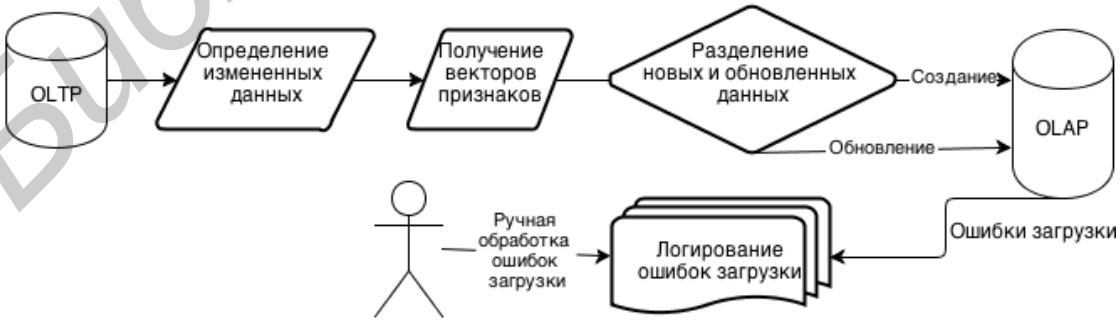

Рис. 3. Диаграмма потока данных между OLTP и OLAP хранилищами

Одной из наиболее сложных задач при выгрузке данных между OLTP и OLAP хранилищами является задача определения множества измененных данных со времени предыдущей успешной загрузки. Существует несколько способов определения данного множества:

– пересечение двух соответствующих множеств из двух хранилищ для поиска непересекающегося подмножества из OLTP-хранилища;

– очистка OLTP-хранилища после каждого процесса миграции данных;

– использование инкрементальных ключей и механизма истории в OLTP-хранилище вместе с временными метками миграций данных.

Первый способ чреват сильными нагрузками на сервер баз данных и, как следствие, вероятностью частых отказов системы при росте количества данных. К недостаткам второго способа можно отнести необходимость создания третьего хранилища, для обеспечения доступа ко всем исходным сигнальным выборкам конечному пользователю, что в свою очередь требует усилий по обеспечению согласованности данных между множеством хранилищ. Третий способ представляется наиболее оптимальным, хоть в общем виде и ставит сравнительно сложную задачу организации хранения истории всех изменений в OLTP-хранилище.

В программном стеке Microsoft существует несколько способов организации истории изменений данных пользователя в зависимости от уровня организации, нагрузки на ресурсы системы и полноценности предоставляемых данных. Преимущества и недостатки каждого способа описаны в табл. 2.

Таблица 2. **Способы организации истории изменений данных пользователя**

Механизм	Уровень	Полноценность	Недостатки
Средства ORM	Высокий	Все необходимые данные	Несогласованность. Реализуется внешними средствами
Триггеры	Низкий	Все необходимые данные	Значительно снижает прозрачность работы базы данных
MS SQL Server Audit Log	Низкий	Все необходимые данные	Избыточность данных Повышенная нагрузка
Change Data Capture	Низкий	Все необходимые данные	Повышенная нагрузка
Change Tracking	Низкий	Только факт изменения данных	Недостаточен, если необходима миграция непосредственно истории изменений в OLAP-хранилище

Так как процесс миграции данных между хранилищами должен производиться регулярно, а OLTP-хранилище не предполагает частого редактирования хранимых данных, оптимальным способом определения множества измененных данных (новых и отредактированных) представляется механизм Change Tracking. Таким образом, процесс выгрузки данных представляет собой следующую последовательность:

– включение механизма Change Tracking для необходимых таблиц (выполняется один раз при настройке хранилища);

– создание таблицы синхронизаций в исходном хранилище, содержащей информацию о времени начала каждой выгрузки, успешности данной выгрузки данных и идентификаторе последнего изменения в системе (специфичный для данного механизма аналог временной метки);

– определение временной метки последней успешной выгрузки;

– выгрузка, очистка, преобразование данных;

– обновление флага успешной выгрузки в таблице синхронизаций в случае корректной выгрузки данных.

В случае, если в процессе миграции данных на каком-либо из этапов произойдет ошибка обработки части данных, не влияющая существенным образом на миграцию данных в целом, эти данные все равно должны быть сохранены в отдельный лог-файл для последующей ручной или полуавтоматической обработки. Указанные меры необходимы для устранения потерь данных при миграции, а так же для отладки процессов миграции данных, а так же как можно раннего определения изменений формата входных данных.

Заключение

Разработан ряд архитектурных подходов для организации системы поддержки принятия решений при обработке длинных реализаций вибросигналов на базе систем управления базами данных. Использование MS SQL сервера в качестве базиса для построения

такой системы поддержки принятия решения на основе анализа вибрационных сигналов, отражающих состояние исследуемых объектов, решает множество проблем, в числе которых:

– согласованное хранение сигнальных данных вместе с их атрибутами и характеристиками обследуемых объектов;

– поддержка встраиваемых функций сигнальных преобразований (что позволяет производить согласованный анализ данных и значительно упрощает процессы трансформации данных при их миграции);

– поддержка интеграции разделенных хранилищ сбора и анализа вибрационных характеристик.

Описанные подходы позволяют организовать распределенное хранение и анализ сравнительно больших объемов данных (несколько десятков миллионов векторов признаков), что представляется достаточным для области вибродиагностики объектов, но при возникновении необходимости анализа объемов данных, превышающих указанный предел на несколько порядков, рекомендуется использование асинхронных методов анализа данных, таких как MapReduce.

DISTRIBUTED WAREHOUSING FOR VIBRATIONAL SIGNALS ANALYSIS SYSTEM

V.E. BAZAREVSKY

Abstract

Article covers the methods of vibration signals long implementations analysis as well as complex analysis of large amounts of signal data. Several architectural pattern applications of signal and analytical data warehouses for decision support system that evaluates the vibration characteristics of the objects are proposed. The ways of extending and embedding third-party functionality in MS SQL Server engine are recommended.

Список литературы

1. *White T.* Hadoop: The Definitive Guide. Sebastopol, 2010.
2. *Codd Edgar F.* / Framingham, Computerworld. November 1998. Vol. 27. № 30.
3. *Kimball R, Ross M, Thornthwaite W.* The Data Warehouse Lifecycle Toolkit. New Jersey, 2008.
4. *Tok W.H., Parida R., Masson M. et. al.* Microsoft SQL Server 2012 Integration Services. Sebastopol, 2012.
5. *Гольденберг Л. М.* Цифровая обработка сигналов. Справочник. М., 1985.