электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

**УДК 004.422**

# Объединение Cache с реляционной базой данных для осуществления обмена в режиме реального времени

***Оситняжская И.Е.,*** *студент*
*кафедра «Системы обработки информации и управления»,*
*Россия, 105005, г. Москва, МГТУ им. Н.Э.Баумана*

***Цатурян Т.Ш.,*** *студент*
*кафедра «Системы обработки информации и управления»,*
*Россия, 105005, г. Москва, МГТУ им. Н.Э.Баумана*

*Научный руководитель: Виноградова М.В., к.т.н., доцент*
*Россия, 105005, г. Москва, МГТУ им. Н.Э.Баумана*
*vinogradova.m@bmstu.ru*

## Постановка задачи

Cache – иерархическая СУБД производства компании InterSystems, позиционирующаяся производителем как «объектно-реляционная» или «постреляционная». [1, 2]

В Cache отсутствует импорт, экспорт и осуществление связи с другими СУБД. Также в Cache плохо организована кроссплатформенность: на разных машинах база данных может работать некорректно из-за разных версий ODBC драйвера.

В связи со всеми этими проблемами и с недостатками, представленными далее, возникла идея создать систему, которая позволит переносить данные из Cache в другие СУБД, в частности MySQL.

## Преимущества и недостатки работы с Cache

Каждая программа имеет свои преимущества и недостатки. Сначала расскажем о преимуществах. Ядро Cache имеет следующие особенности:

- Высокоэффективная поддержка обработки транзакций.
- Быстрая разработка прикладного обеспечения.
- Модель базы данных, основанная на модели данных ODMG и расширяющая ее.
- Новый, объектно-ориентированный язык программирования Cache ObjectScript.

- Быстрое и простое моделирование приложений с использованием объектной технологии.
- Резервное копирование в реальном масштабе времени.
- Поддержка для приложений многоуровневой архитектуры клиент/сервер.
- Полная поддержка Unicode.
- Низкие эксплуатационные расходы (простая настройка и администрирование базы данных).
- Автоматическое восстановление в случае сбоев сети.
- Автоматическое восстановление в случае аварии сервера.
- Автоматическое обслуживание базы данных.

Но есть и недостатки. В Cache используется реляционно-процедурный язык MUMPS.[3] Определение MUMPS, как языка программирования условно, потому что он зародился давно и, по сути, является устаревшим языком. Сфера его применения простирается от работы с медицинским оборудованием до операционных СУБД и экспертных систем. В данном языке все записи хранятся как строки, присутствует низкий уровень абстракции и низкая читабельность, особенно при использовании сокращений, сложный синтаксис. В ходе исследований в языке MUMPS было выявлено, что:

- отсутствует обязательное объявление переменных;
- отсутствует поддержка привычных приоритетов арифметических операций;
- лишний пробел или разрыв строки может совершенно изменить смысл кода;
- ключевые слова языка не зарезервированы и могут широко использоваться в качестве идентификаторов;
- весь код хранится в базе данных и для его написания надо использовать специальное приложение.

Важно отметить, что в руководстве пользователя представлена лишь небольшая толика информации. Также присутствует большая проблема интеграции Cache в существующие системы. По умолчанию невозможен экспорт/импорт данных в CSV, Excel или PDF.

**Обоснование связи Cache с другими СУБД**

Современные системы становятся все более и более интегрированными друг в друга. Миграция данных, пользовательские преференции (работа с несколькими системами одновременно), наличие уже созданного программного обеспечения оставляет на рынке только кроссплатформенные системы, имеющие отличную документацию,

знакомых для программистов, разработчиков и специалистов по внедрению. Однако старые системы не умирают, если стоимость перехода (включающая необходимость переписывание существующего кода) значительно выше получаемых преимуществ.

Приведем несколько примеров задач, где может потребоваться интеграция Cache с реляционными БД:

1. Переход на SQL и отказ от DBMS с переписыванием программного обеспечения. В этом случае требуется перенести базу данных, а так же какое-то время поддерживать целостность данных в обеих системах: DBMS(как рабочую систему на данный момент) и SQL(как потенциально новую, находящуюся в режиме тестирования и перехода).

2. Использование обеих баз данных по экономическим или структурным причинам (Cache является платным ПО, в то время как MySQL является бесплатным ПО). Допустим, госпиталь имеет уже существующую систему и хочет хранить большой объем данных, которые слабо связанны с существующей системой (например, архивные данные). В данном случае можно предложить осуществить перенос некоторой части БД (а именно архивной части) в отдельную базу MySQL. При этом будет достигнута ценовая эффективность (снижение стоимости лицензий Cache) и размещение данных в MySQL, где есть расширенный набор инструментов, например, для Data Mining (добыча данных, интеллектуальный анализ данных, глубинный анализ данных) и анализа. При возникновении потребности в архивных данных из Cache, они могут быть прозрачно для конечной системы перенесены из MySQL.

3. Использование программных решений работающих с MySQL. Например, есть пакеты, работающие с MySQL и значительно облегчающую работу (формирование и правку) Excel, Word, PDF, построение отчетов. Тут возможно имеет смысл выгрузить данные в MySQL и провести анализ через нее, нежели писать дополнительное ПО.

**Обзор существующих решений.**

Рассмотрим пару решений конвертирования данных из Cache в другие базы данных:

1. Утилита Full Convert Enterprise. Удобная и мощная утилита для конвертирования данных из одной базы данных в другую. Быстро и эффективно конвертирует уже готовую базу данных в Cache в любую другую СУБД.

Техническая поддержка является бесплатной в течение 12 месяцев. Ограничение на бесплатную триальную версию в 30 дней. При помощи этой утилиты можно произвести только миграцию данных из Cache в другие СУБД. Не поддерживает обмен в режиме реального времени.

2. Утилита The Evolve Suite.[4] Представляет собой комплекс программных инструментов, позволяющий перейти от Cache к реляционным базам данных. Включает в себя три продукта: Evolve Data Migration (осуществляет перенос данных из базы данных из Cache в стороннюю СУБД), Evolve Replication (осуществляет перенос данных и отслеживает изменения в базе данных и применяет эти изменения к сторонней СУБД) и Evolve System Migration (преобразует всю базу данных из Cache к формату, идентичному Java приложению). Является платной. Как и предыдущая утилита производит перенос данных только в одну сторону, но Evolve Replication поддерживает обмен данных в режиме реального времени. Найти информацию о данной утилите и скачать ее возможно только на официальном сайте разработчиков.

**Обмен в режиме реального времени**

Некоторые современные задачи требуют поддержки обмена в режиме реального времени.

Режим реального времени [5] - такой режим обработки данных, при котором обеспечивается взаимодействие вычислительной системы, в отношении к выполняемым действиям, в темпе, соизмеримым со скоростью самого процесса.

Один из действительно крупных ученых в этой области, Дональд Гиллельс, дал такое определение: «Системой реального времени является такая система, корректность функционирования которой определяется не только корректностью вычислений, но и временем, в которое получается требуемый результат».

Основная особенность – вычислительная система обеспечивает своевременную реакцию на все произошедшие события даже при максимальной загрузке.

Назначение данных систем – взаимодействие с объектами внешнего (по отношению к системе) мира в темпе процессов, протекающих в этих объектах.

Для того, чтобы обеспечить такой режим, управляющая ЭВМ наделена способностью «обрабатывать» внешние сигналы, возникающие в случайный момент времени.

**Реляционных и объектно-ориентированных СУБД**

В Cache строятся объектно-ориентированные СУБД, а мы же намеренны, осуществить перенос данных в реляционную СУБД.

В реляционных СУБД пользователь создает внешние ключи, через которые и происходит управление связями. Слабой стороной реляционной технологии является объединение (когда система просматривает несколько таблиц, сравнивая внешние ключи, до нахождения соответствий), которое плохо работает при двух или более уровнях объединений. В объектно-реляционной СУБД пользователь только объявляет связь, а СУБД уже сама придумывает, как правильно и лучше эти связи организовать. Ссылки генерируются прямые, так как много затратные операции по просмотру и сравнению индекса отсутствуют. [6, 7]

Объектно-ориентированные СУБД поддерживают объектно-ориентированные языки программирования. В то время как реляционные СУБД требует, чтобы разработчик транслировал объектную модель к поддерживаемой модели данных и включил подпрограммы, чтобы обеспечить это отображение во время выполнения.

В реляционных БД хранятся только голые данные.[8, 9] Что с ними будет делать приложение, зависит уже от приложения. В ООБД же, напротив должны храниться объекты, а объект это совокупность его свойств и методов.

**Протокол данных SOAP**

Cache имеет встроенную реализацию протокола SOAP, который обеспечивает связь с веб-сайтом. Любой метод класса Cache может быть представлен как веб-сервис посредством простого наследования из соответствующих классов, встроенных в Cache. При пересылке сообщения между сервером Веб-служб и клиентом создаются SOAP-пакеты. В этих сообщениях содержится или запрос на выполнение какого-либо действия, или ответ на этот запрос. Конверт и его содержимое кодируются на XML.

В нашем решении мы тоже планируем задействовать данный протокол.

Simple Object Access Protocol (SOAP) [10, 11] — это протокол на базе языка XML, который определяет правила передачи сообщений по Internet между различными прикладными системами. В основном он используется для удаленного вызова процедур. Изначально протокол SOAP разрабатывался с тем расчетом, что он будет функционировать «над» HTTP (дабы упростить интеграцию SOAP в Web-приложения), однако теперь могут быть задействованы и иные транспортные протоколы, например SMTP.

SOAP может использоваться с любым протоколом прикладного уровня: SMTP, FTP, HTTP, HTTPS и другие. Однако его взаимодействие с каждым из этих протоколов имеет свои особенности, которые должны быть определены отдельно.

С помощью SOAP разработчики могут создавать Web-службы настолько же оперативно, насколько быстро будут написаны сообщения SOAP для вызовов программ для существующих приложений, а затем добавить эти приложения в простые Web-страницы. Но, кроме того, у разработчиков есть возможности использовать вызовы SOAP в выделенных приложениях и создавать приложения, которые можно переносить на Web-страницы других людей, и тем самым избежать трудоемкого и дорогостоящего процесса разработки.

Однако присутствуют и минусы:

- Использование SOAP для передачи сообщений увеличивает их объём и снижает скорость обработки. В системах, где скорость важна, чаще используется пересылка XML-документов через HTTP напрямую, где параметры запроса передаются как обычные HTTP-параметры.
- Хотя SOAP является стандартом, некоторые программы часто генерируют сообщения в несовместимом формате.

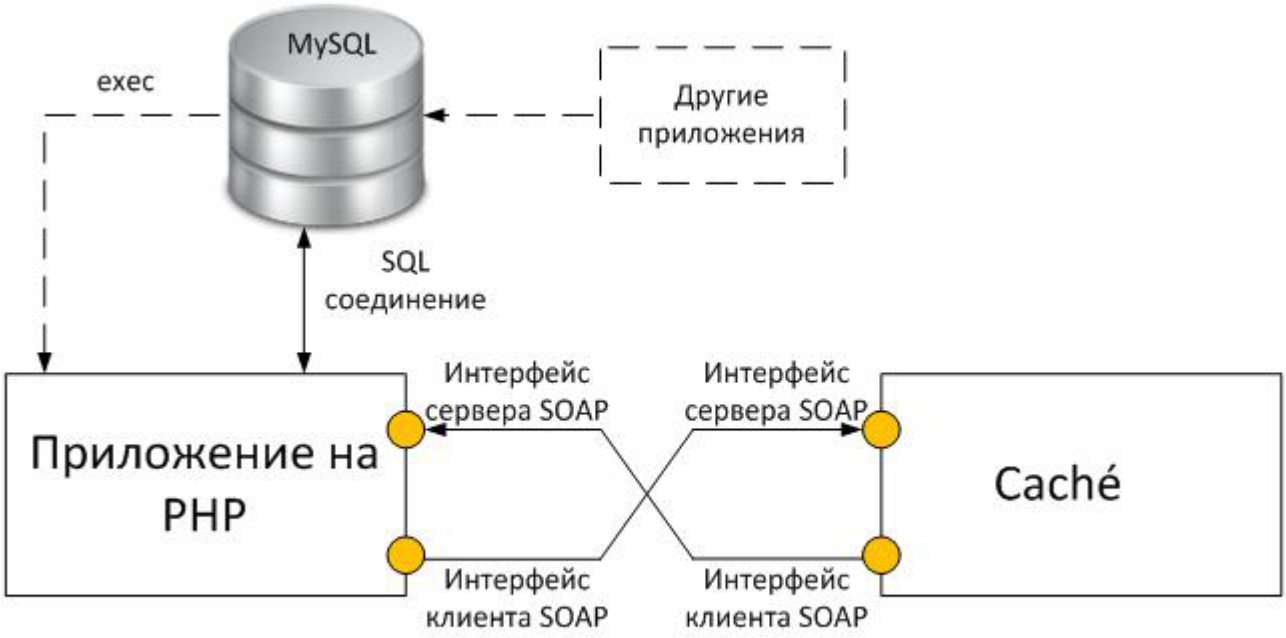
В Cache отсутствует обратная совместимость версий: могут возникнуть проблемы при работе с разными версиями ODBC драйвера. При возникновении ошибок никакие сообщения не выдаются, программа просто выводит невнятное уведомление где-то внутри программы.

Присутствуют проблемы с пулом соединений ODBC: проблемы с работой драйверов и поддержкой соединений. Присутствует разрыв удаленного соединения при отсутствии действий со стороны пользователя.

**Предлагаемое решение по миграции данных**

Использование протокола SOAP поможет создать мост между Cache и приложениями, которые имеют встроенный SOAP, чтобы приложения на PHP могли использовать данные из базы данных. Дополнительно мы связали Cache с MySQL в режиме реального времени, что позволяет одновременно проводить транзакции в обоих базах данных или делить задачи и использовать обе базы данных или обеспечить частичный переход на Cache.

На рисунке представлена схема взаимодействий в режиме реального времени:




Обмен данными, который можно наблюдать на данной схеме, происходит в два этапа:

1) На Cache и PHP приложении поднимаются SOAP клиент и SOAP сервер. Инициатором запроса на изменение (запись, удаление, добавление) данных может выступать, как Cache, так и приложение.

Пример:

- инициатор Cache: в Cache были изменены данные, и их необходимо отправить в другую СУБД (MySQL);
- инициатор приложение: изменение данных происходит в MySQL, должны быть произведены изменения в Cache.

2) Приложение связывается с MySQL при помощи стандартного SQL-соединителя.

Cache имеет встроенные триггеры (on update, on delete, on insert), которые позволяю проводить изменения данных. При помощи данных триггеров осуществляется запрос к приложению.

MySQL не поддерживает инициацию SOAP соединения, но поддерживает запуск сторонних программ при срабатывании триггеров (exec).

При срабатывании триггера происходит подключение к приложению через SOAP,и там получают информацию обо всех изменениях в базе данных.PHP приложение обновляет эти данные через стандартное SQL соединение. Другие приложения могут

работать с MySQL через стандартную функцию exec. Приложение делает запрос к Cache через SOAP и обновляет или удаляет данные. Таким образом, обеспечивается согласование данных между двумя СУБД.

При необходимости мы можем осуществить экспорт в Excel. Есть два способа создания файлов в Excel:

- создание файлов, используя работу с XML в Cache
- воспользоваться специальным ПО (которое, кстати, есть и бесплатное).

На форумах поднимался данный вопрос, и специалисты остановились на первом варианте.

Первый вариант имеет один существенный плюс – это повышение навыков при формировании файлов Microsoft Office.

Но присутствуют и недостатки:

- Требуется основательно разобраться со структурой и ошибки могут привести к не читаемости файлов или к отсутствию кросплатформенности.
- Значительно повышенные расходы на разработку, где программисты, по сути, реализуют велосипед заново.
- Зависимость от обновлений. При очередной смене формата (или его расширении) функции перестанут работать.

Использование специализированного ПО позволяет программисту сфокусироваться лишь на извлечении данных, а файловое представление система будет стоить сама. При этом минусы исчезают, и возникает лишь задача выборки данных и интегрирования с существующими пакетами. Был взят PHP Excel как пакет формирования и применили ранее описанный метод извлечения данных (при помощи SOAP). В итоге файл, создающий Excel, состоит из несколько десяток строчек кода и базируется на функциях пакета PHP Excel, а не ручной создании схемы.

**Преимущества и недостатки данного решения**

Преимущества:

- Возможен импорт и экспорт данных в режиме реального времени;
- Подключение различных модулей для работы с базой данных.

Недостатки:

- Решение не является полностью автоматизированным. При интеграции из одной базы данных в другую сложно создать универсальный принцип перехода, который сможет правильно перекинуть базу данных.

В базе данных существует возможность получения противоречивости: запись в одну и ту же строку, но в разные базы данных. Шанс возникновения данной возможности мал, но ему подвержены все базы данных.

**Примеры используемого кода**

1. Создание SOAP-клиента по WSDL-документу on PHP

```
$client = new SoapClientLogger(new SoapClient("test_some.wsdl", array('soap_version' => 3,
'encoding'   =>  'UTF-8')));
$result = $client->SayHello("222xxbb2221vv122");
```

2. Пример создания триггера Cache

```
ClassMethod %OnDelete(oid As %ObjectIdentity) As %Status [ Private, ServerOnly = 1 ]
{
	//Пример выполнения кода
	//set o=##class(Archive).%New()
	//set o.Log("Some data")
	//do o.%Save()
		write "Delete object "_oid
	//Метод возвращает код%Status , когда статус отказа останавливает удаление
	QUIT $$$OK
}
```

3. Пример простейшего сервиса Cache

```
/// MyApp.MyService
Class MyApp.MyService Extends %SOAP.WebService [ ProcedureBlock ]
{
///  Имя Веб-Сервиса
Parameter SERVICENAME = "MyService";
/// Область SOAP для Веб-Сервиса – ее нужно поменять на Вашу
Parameter NAMESPACE = "http://tempuri.org";
/// Пространство имен задействованных классов будет использовано в WSDL.
Parameter USECLASSNAMESPACES = 1;
Method Test() As %String [ WebMethod ]
{
	Quit "Test"
}
```

```
/// Этот метод возвращает завтрашнюю цену за указанный материал
Method Forecast(StockName As %String) As %Integer [ WebMethod ]
{
   // применяются запатентованные, нелинейные, эвристические методы, чтобы найти
новую
  //цену
   Set price = $Random(1000)
   Quit price
}
}
```

4. Пример создания SOAP клиента в Cache

```
ClassSoapDemo.SoapDemoSoapExtends %SOAP.WebClient [ ProcedureBlock ]
{
/// URL, используемый для доступа к веб-службе
Parameter LOCATION = "http://test1.ru/cache/php-wsdl-2.3/demo.php";
/// Пространство имен, используемое службой
Parameter NAMESPACE = "http://test1.ru/cache/php-wsdl-2.3/";
/// Использование атрибута типаxsi для точных типов
Parameter OUTPUTTYPEATTRIBUTE = 1;
/// Определяет обработку заголовка безопасности.
Parameter SECURITYIN = "ALLOW";
/// Имясервера
Parameter SERVICENAME = "SoapDemo";
/// версияSOAP, поддерживаемаяслужбой
Parameter SOAPVERSION = 1.2;
Method SayHello(name As %String) As %String(XMLNAME="return")
[ Final, ProcedureBlock = 1, SoapBindingStyle = rpc,SoapBodyUse = encoded, WebMethod ]
{
 Quit ..WebMethod("SayHello").Invoke($this,"http://test1.ru/cache/php-wsdl-
2.3/SayHello",.name)
}
}
```

**Выводы**

В ходе нашей работы мы реализовали приложение для переноса данных между двумя СУБД. Разработка данного приложения велась на PHP.

Для того, чтобы произвести корректный перенос данных между двумя базами данных необходимо участие программиста. Он должен будет провести несложные действия для правильной постройки принципа перехода.

В дальнейшем можно разработать механизм, чтобы исключить данных недостаток. Можно написать программу, которая будет осуществлять синтаксический анализ (парсинг) и производить автосоздание таблиц и обмен данными. Позволит полностью автоматизировать работу.

**Список литературы**


1. Inter Systems Cache. Available at: http://www.intersystems.ru/cache/, accessed 29.04.2014.
2. Cache . Available at: http://ru.wikipedia.org/wiki/Cach%C3%A9, accessed 27.04.2014).
3. Отличительные особенности СУБД Cache. Режим доступа: http://citforum.ru/database/articles/subdcache.shtml (дата обращения 29.04.2014).
4. The Evolve Suite. Available at: http://www.cavsystems.com/evolve/, accessed 30.04.2014.
5. Лекции «Системы реального времени». Режим доступа: http://as08sbmpei.wikidot.com/sistemy-realnogo-vremeni, (дата обращения 30.04.2014).
6. Реляционная база данных. Режим доступа: http://ru.wikipedia.org/wiki/%D0%E5%EB%FF%F6%E8%EE%ED%ED%E0%FF_%E1%E0%E7%E0_%E4%E0%ED%ED%FB%F5 (дата обращения 06.05.2014).
7. Реляционные базы данных и язык SQL. Режим доступа: http://citforum.ru/database/sql_kg/1-1.shtml (дата обращения 06.05.2014).
8. Общие понятия реляционного подхода к организации баз данных. Основные концепции и термины. Режим доступа: http://bazid.narod.ru (дата обращения 08.05.2014).
9. Балдин А.В., Елисеев Д.В., Агаян К.Г. Обзор способов построения темпоральных систем на основе реляционной базы данных // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2012. № 8. DOI: 10.7463/0812.0441884.
10. SOAP. Available at: http://www.hardline.ru/1/12/4102, accessed 10.05.2014.
11. Практическое использование SOAP в PHP 5. Режим доступа: http://phpclub.ru/detail/article/soap (дата обращения 10.05.2014).