УДК 004.94

Е.В. Ивохин, К.Э. Юштин, В.С. Архипов

Киевский национальный университет имени Тараса Шевченко, Киев

СОЗДАНИЕ МЕХАНИЗМОВ РЕПЛИКАЦИИ НА ОСНОВЕ СТОРОННИХ ПРИЛОЖЕНИЙ

В работе рассмотрены основные современные механизмы реализации гомогенных моделей репликации ведущими производителями СУБД. На их основе предложены принципы создания собственных решений для гетерогенной репликации. Приведены примеры технической реализации в случае использования технологии .NET Framework.

Ключевые слова: *репликация снимками, репликация слиянием, транзакционная репликация*.

Введение

Репликация представляет собой набор технологий копирования и распространения данных и объектов между экземплярами серверов баз данных, а также синхронизации баз данных для поддержания согласованности информационного наполнения. Используя репликацию, можно распространять данные в различные расположения, а также удаленным или мобильным пользователям по локальным или глобальным сетям, посредством удаленного доступа, по беспроводным соединениям и через Интернет. Именно этот принцип заложен в основу тенденций развития современных СУБД (например, лозунгом SQL Server 2008 является «Your data is everywhere») [2].

В современных приложениях существует три основных причины необходимости реализации репликации. Первой причиной является требование повышения производительности распределенных систем. Если между узлом, на котором хранятся данные (Data Storage) и узлом, который их обрабатывает (Application Server) медленный (или ненадежный) канал, это может приводить к чрезмерно большому времени работы системы. Если потребность в такого рода операциях возникает часто, то нужно задумываться либо о переносе данных «ближе» к узлам, в них нуждающихся, либо о создании локальных копий данных, т.е. их репликации. Вторая причина, по которой часто используют репликацию, состоит в требовании повышения надежности системы путем резервного копирования части критичных данных. Третья причина – это агрегация данных из нескольких различных источников для совместной обработки их в одном месте (например, несколько филиалов и центральный офис).

Типы репликаций

Современные СУБД поддерживают различные типы репликаций данных [5, 6]. При выборе типа репликации нужно учитывать такие факторы как:

- Автономность (при разработке приложения необходимо продумать уровень автономности или независимости сервера для каждого узла).
- Задержки (время, затрачиваемое на внесение изменений на публикующем сервере, до того как они станут доступными для сервера подписчика).
- Непротиворечивость данных:
  - сходимость данных (все узлы заканчивают операции с одинаковыми результатами);
  - транзакционная непротиворечивость (данные на любом сервере идентичны, так, будто все транзакции выполнялись на одном сервере).
- Непротиворечивость схем.

Но главная задача состоит в выборе между двумя основными критериями нахождения золотой середины:

- Необходимостью связности данных в ущерб независимости.
- Необходимостью минимизации возможности конфликтов и максимальная автоматизация решений конфликтов, если такие все-таки возникнут.

Некоторые сценарии репликации не предусматривают связности данных кроме единичных случаев, другие не предусматривают связности данных в принципе. Остальные сценарии репликации требуют абсолютной непротиворечивости данных с недопустимостью их потери. В порядке уменьшения степени серверной зависимости выделяют следующие типы репликации:

- Мгновенная (snapshot) репликация.
- Репликация слиянием (merge replication).
- Транзакционная репликация.

Далее мы собираемся просмотреть потенциальные преимущества и недостатки различных типов репликаций. Существует множество стандартных решений на уровне конкретных СУБД для репликации, однако они применимы в основном только для гомогенной репликации (то есть, когда сервера

© Е.В. Ивохин, К.Э. Юштин, В.С. Архипов

репликации находятся под управлением одной и той же СУБД). Единственным приятным исключением является сейчас репликация между Oracle и Microsoft SQL Server. Однако, достаточно высокая стоимость подобных решений и отсутствие кросс-СУБД реализаций для репликаций с Open Source базами данных приводит к необходимости разработки собственных решений реализации репликаций. Таким образом, целью этой работы является обзор методов создания собственных решения для реализации трех видов репликаций без привязки к конкретным СУБД.

Полная репликация или мгновенная репликация или репликация на основе моментальных снимков (snapshot)

Репликация моментальных снимков – это наиболее простой тип репликации. В случае мгновенной репликации из сервера-источника берется «образ» данных, подлежащий копированию. Этот образ используется для замещения данных на сервере-получателе. Полные таблицы или их части передаются подписчикам во время процесса репликации. Так как обновления происходят периодически, большую часть времени требуются минимальные издержки в сети и сервере для поддержки репликации. В простейшем случае мгновенная репликация используется для обновления таблиц, доступных только для чтения на системе подписчика. Такой способ предусматривает максимальный уровень автономности сервера, но за счет относительно большой задержки.

Основными требованиями для мгновенной репликации является синхронизация данных, а также единственный доступ к ней агентов репликации во время получения снимка. Мгновенная репликация чаще всего используется для обновления данных, предназначенных для просмотра, и копий данных, предназначенных только для чтения на удаленных серверах.

Для того чтобы создать собственную реализацию репликации на основе моментальных снимков, необходимо использовать встроенные средства или создать свое приложение для получения копии данных исходной базы данных. Единственным замечанием является то, что разные типы СУБД используют разные механизмы генерации нового значения ключа (например, автонаращиваемые identity или использование последовательностей), а для восстановления данных достаточно отключить автогенерацию значений при вставке в данные источник. Поскольку при восстановлении снимка данных на целевом сервере предыдущие данные обычно не представляют ценности, то этот тип репликации является наиболее простым для сторонней реализации.

Репликация слиянием (Merge replication)

Это самый сложный тип репликации. При репликации слиянием изменения можно вносить как на издателе, так и на подписчиках. Все изменения сводятся воедино на издателе, который разрешает конфликты в случае их возникновения. Для разрешения конфликтов репликации можно определить свою собственную программную логику. После этого итоговый вариант данных передается подписчикам.

Репликация слиянием, как и транзакционная репликация, как правило, начинается с моментального снимка объектов и данных базы данных публикации. Последующие изменения данных и схемы, произведенные на стороне издателя и подписчиков, обычно отслеживаются при помощи триггеров. Подписчик синхронизируется с издателем при подключении к сети и обменивается с ним всеми строками, которые изменились со времени последней синхронизации издателя и подписчика.

Репликация слиянием обладает высоким уровнем автономности, но характеризуется наибольшими задержками и рискует быть транзакционно противоречивой, поскольку различные серверы могут начать обновления одной и той же строки одновременно. Противоречивость не всегда может быть устранена в полной мере, но может быть устранена по каким-то выбранным пользователем критериям для большей надежности. Репликация слиянием работает через агента слияния (Merge agent). Он копирует изменения со всех подписчиков и вносит их в данные, хранящиеся на публикующем сервере. Потом копирует все изменения (включая изменения которые были вызваны им самим) с публикующего сервера на остальные сервера-подписчики.

Репликация слиянием используется в основном тогда, когда нужна поддержка разделенных таблиц, когда пользователь может просматривать содержимое всех узлов, но изменять лишь непосредственно свой. Используя этот тип репликации, необходимо соблюдать проверки готовности данных для копирования.

В качестве собственного решения можно использовать два решения: создавать триггеры, которые автоматически будут записывать измененные и исходные данные модифицированных строк реплицируемых таблиц в дополнительный источник. Далее, специальное приложение издателя будет собирать информацию об изменениях и объединять их у себя. Для применения изменений на подписчиках необходимо генерировать итоговый транзакционный скрипт, адаптированный для типа СУБД каждого подписчика. Для избежания зацикливания на время выполнения скрипта необходимо будет временно отключать триггеры, которые будут генерировать информацию об изменениях.

Транзакционная репликация

Как следует из названия, транзакционная репликация используется для репликации исполняемых команд на сервере. При использовании этого метода репликации база данных подписчика создается первоначально с помощью моментального снимка, а последующие изменения, которые вносятся в базу данных издателя, реплицируются подписчику в соответствии с новыми исполненными транзакциями. Все обновления, вставки и удаления, выполняемые в базе данных, распространяются именно таким образом.

Разница между транзакционной и мгновенной репликацией состоит в том, что при транзакционной репликации на серверы-подписчики копируются данные не целиком, а только измененные данные. Все примененные к опубликованным данным запротоколированные изменения (например, операторы INSERT, UPDATE, DELETE) отслеживаются и затем копируются абонентам. Все транзакции, примененные к данным на публикующем сервере подписки, в той же самой последовательности применяются к серверам-подписчикам.

Далее, сравним реализации построения процесса репликации для различных СУБД. В качестве примера возьмем Microsoft SQL Server и систему репликации Open Source базы данных Slony-I. После чего предложим и обсудим собственное решение для реализации процесса репликации между гетерогенными источниками данных для двух случаев репликации (snapshot и merge).

Реализация в Microsoft SQL Server

Все вышеописанные репликационные действия в Microsoft SQL Server реализуются отдельными службами, которые называются агентами [1, 3].

Мгновенная репликация (Snapshot Replication) осуществляется через агентов репликации (рис. 1):

- Агент мгновенной репликации (Snapshot agent). Поддерживает мгновенную репликацию и начальную синхронизацию данных. Он получает копию всех данных и хранит ее на распределяющем сервере.
- Распределяющий агент (Distribution agent). Выполняет запуск транзакций и перенос мгновенных копий с публикующего сервера на сервер-подписчик.

Для мгновенной репликации используются периодические обновления. Это происходит следующим образом:

- Агент мгновенной репликации устанавливает блокировку на все статьи публикации для гарантии непротиворечивости данных.
- Копия схемы каждой таблицы записывается в рабочий каталог на распределяющем сервере.
- Мгновенная копия табличных данных записываться в соответствующий каталог.
- Агент мгновенной репликации снимает блокировку со всех статей публикации.
- Распределительный агент создает целевые таблицы и объекты базы данных, такие как индексы у подписчика. Затем проводится копирование мгновенных данных с перезаписью уже существующих таблиц.

Транзакционная репликация (рис. 2) выполняется агентами репликации, а это:

- Агент чтения протокола (Log Reader Agent). Агент чтения протокола отвечает за копирование отмеченных транзакций с публикующего сервера на распределяющий сервер. Для ускорения процесса агент читает данные во внутреннем бинарном формате.
- Распределяющий агент. Он отвечает за передачу транзакций от распределяющего сервера к подписчикам.

После проверки начальной синхронизации транзакционная репликация выполняется по следующему сценарию:

- Внесенные изменения данных записываются в журнал транзакций на публикующем сервере подписки.
- Агент чтения протокола просматривает журнал транзакций и находит изменения, помеченный для копирования.
- Изменения прочитываются из журнала транзакций и вносятся в распределяющую базу данных хранящуюся на распределяющем сервере (distribution).

Распределяющий агент вносит изменения в соответствующие таблицы базы данных.

Транзакции считываются из журнала транзакций агентом чтения журнала, который запускается на дистрибьюторе и подсоединяется к издателю. Эти транзакции обрабатываются и помещаются в базу данных distribution на распределяющем сервере. Позже агенты распространения (Distribution Agents) считывают эту информацию из дистрибутивной базы данных и применяют транзакции к базе данных подписчика.

Каждая база данных, использующая репликацию транзакций, имеет собственного агента Log Reader Agent, который запускается на распределяющем сервере и следит за журналом транзакций этой базы данных на издателе. На каждую базу данных используется только один агент Log Reader Agent – независимо от количества публикаций, определенных по соответствующей базе данных.

Репликацию слиянием (рис. 3) реализует агент слияния (Merge agent), который копирует изменения со всех подписчиков и вносит их в данные, хранящиеся на публикующем сервере. Затем агент слия-

ния копирует все изменения (включая изменения, которые были произведены самим агентом) с публикующего сервера на остальные серверы-подписчики. После завершения начальной синхронизации далее в течение репликации слиянием происходят следующие события:

- Триггеры отслеживают изменения опубликованных данных.
- Изменения, произведенные на публикующем сервере, копируются на серверы-подписчики.
- Изменения, произведенные на серверах-подписчиках, вносятся в данные на публикующем сервере и разрешаются всевозможные конфликты.

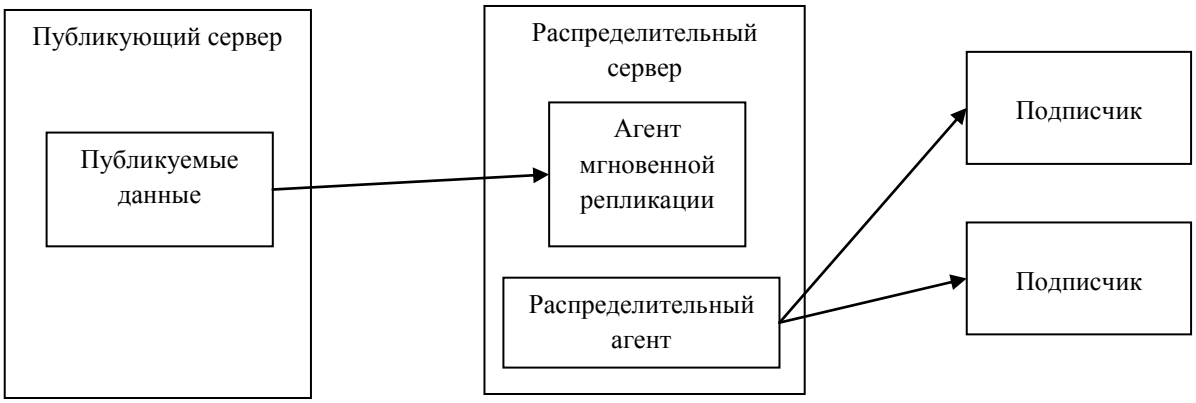

Рис. 1. Модель работы репликации моментальных снимков

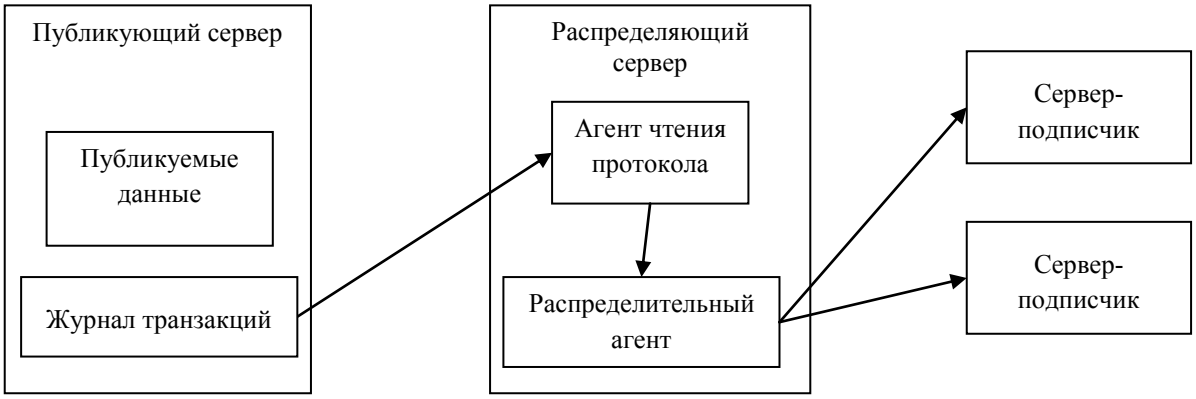

Рис. 2. Модель работы транзакционной репликации

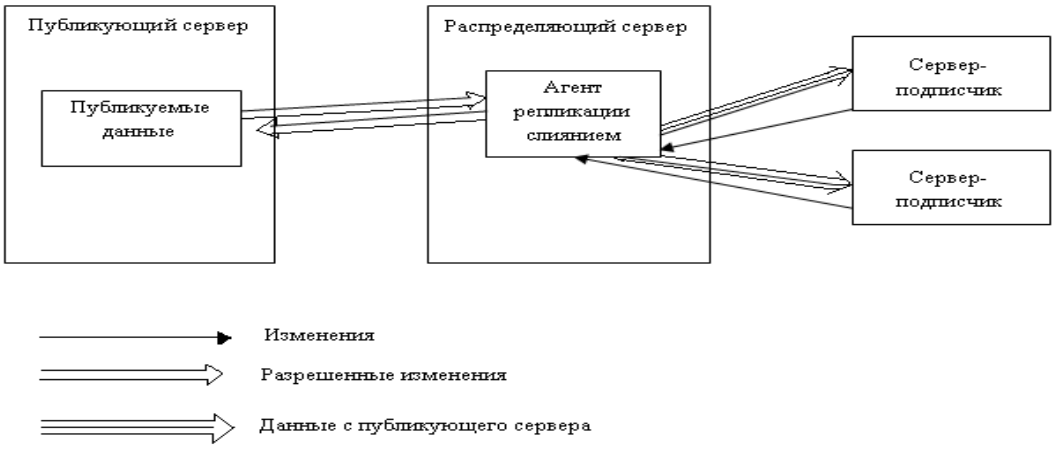

Рис. 3. Метод работы репликации слиянием

Агент слияния вносит изменения не зависимо от того, где они произошли – на публикующем сервере или же на серевер-подписчике. Конфликты разрешаются автоматически агентом слияния.

Необходимо отметить, что для обеспечения возможности работы репликации каждая таблица должна иметь первичный ключ или столбец типа уникальный идентификатор Unique Identifier). Без этого процесс репликации слиянием невозможен, так как невозможно характеризовать строку, в которой произошли изменения. Таким образом, необходимо отметить, что подобный процесс репликации может быть использован и для собственной реализации (за исключением чтения протокола изменения данных, однако вместо него можно использовать триггеры).

Модель Slony-I

Slony – это система репликации в режиме реального времени, позволяющая организовать синхронизацию нескольких серверов PostgreSQL по сети. Slony использует триггеры Postgre для привязки к событиям INSERT/DELETE/UPDATE и хранимые процедуры для выполнения действий.

Основным рабочим элементом Slony-I является многопоточная служба Slon. Поток синхронизации через установленные промежутки времени проверяет наличие репликационной активности, генерируя события SYNC. Локальный поток прослушивает наличие новых конфигурационных событий и модифицирует соответственно конфигурацию других потоков.

Поток очистки репликации выполняет такие действия как удаление устаревших событий и изменения таблиц. Удаленный прослушивающий поток подключается к удаленному узлу в базе данных для получения изменений из провайдера событий (event provider). При получении информации о событиях или подтверждениях, он выбирает соответствующие данные и передает их рабочему потоку целевого сервера. Репликационные данные объединяются в группы по транзакциям. Удаленные рабочие потоки, по одному для каждого целевого сервера, выполняют репликацию данных, сохранение событий и генерацию подтверждений успешного выполнения. В любой момент времени подчиненный сервер гарантированно знает текущую точку обработки транзакций.

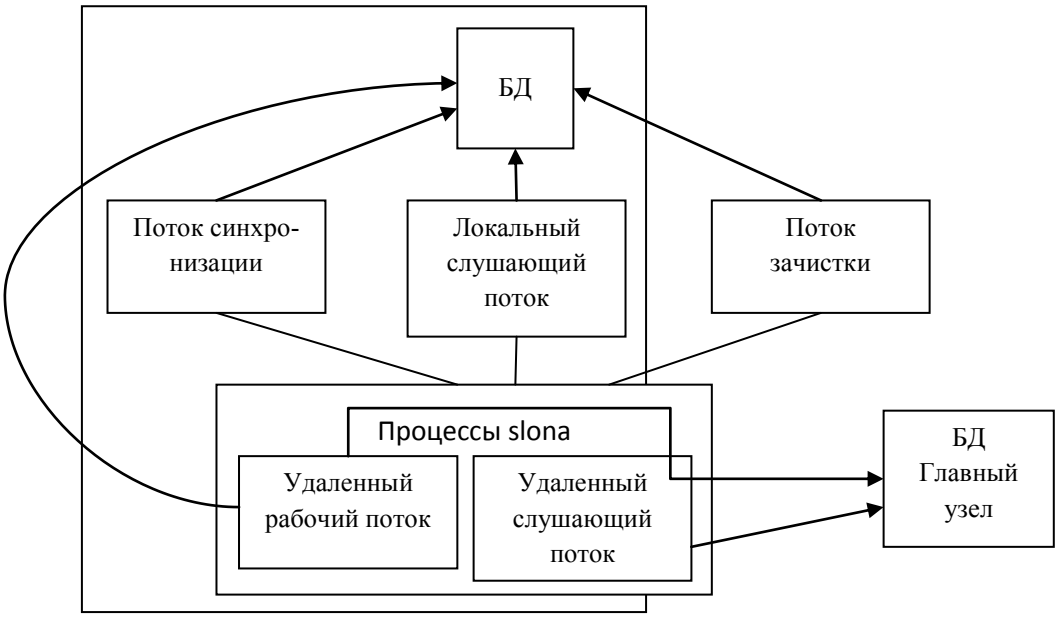

Рис. 4. Модель работы Slony-I

Таким образом, необходимо отметить, что подобный процесс репликации также может быть использован и для собственной реализации. Далее, рассмотрим механизм создания репликации между двумя гетерогенными источниками с использованием технологий .NET Framework.

Использование собственных решений

Рассмотрим классическую задачу репликации слиянием. Пусть имеется две таблицы с идентичной структурой в разных СУБД. Как определить различия в данных, которые содержатся в этих таблицах? Мы можем перебрать все строки, но для этого необходимо иметь колонку, уникально характеризующую строку (первичный ключ или уникальный идентификатор).

Таким образом, задача сводится к генерации скрипта изменений на языке SQL для СУБД целевого сервера. Для ускорения этого процесса [4] в .NET Framework для работы с данными есть объекты типа DataTable, в которых данные в строках представлены в виде объектов DataRow. Для каждого столбца объекты DataRow содержат в себе два вида данных: исходные (Original) и текущие (Current). Таким об-

разом, если заполнить исходные данные в строках информацией из целевого сервера, а текущие данные в строках – информацией из публикующего сервера, то можно быстро получить разницу данных. Для применения изменений на сервер в .NET Framework в классе DataAdapter есть 3 вида шаблонных команд, показывающих как надо добавлять (InsertCommand), модифицировать (UpdateCommand) и удалять (DeleteCommand) данные на целевом сервере. В результате, для разработчиков конкретной модели репликации остаются задачи правильного построения потоков информации и, следовательно, фильтров для запросов. Таким образом, с применением .NET Framework самая сложная модель репликации между различными источниками реализуется стандартно. При этом существует возможность проведения отложенной репликации без онлайнового протоколирования производящихся изменений, когда изменения определяются по конечным состояниям данных на целевом и публикующем серверах.

На основе данного подхода было разработано собственное приложение для репликации данных между тремя серверами системы «Студент» Киевского национального университета имени Тараса Шевченко. Было показано, что можно проводить эффективно репликацию данных даже в случае, когда изменения определяются по конечным состояниям данных на целевом и публикующем серверах.

Для реализации модели мгновенной репликации исходные версии данных в строках заполнять не надо, тогда для всех данных публикующего сервера будет сгенерирован вставочный скрипт. Для транзакционной модели необходимо в клиентском приложении логгировать все запросы на модификацию данных в отдельную таблицу. Далее необходимо из этой таблицы отдельным приложением читать данные (и отмечать прочитанное) на распределяющий сервер и потом применять изменения на все сервера подписки.

Для реализации классической модели репликации слиянием необходимо триггерами протоколировать все изменения в отдельную таблицу с удвоенным количеством колонок (по 2 версии данных из каждой колонки для начальной и конечной версий строк кроме первичного ключа и указанием типа операции: Insert, Update, Delete). Далее из этой таблицы отдельным приложением читать данные (и отмечать прочитанное) на распределяющий сервер и потом применять изменения на все сервера подписки (предварительно отключив протоколирующие триггеры на время внесения пакетов изменений).

Выводы

Таким образом, в данной статье рассмотрены собственные механизмы создания репликационных гетерогенных приложений между различными видами СУБД и показан опыт практического применения в случае репликации данных значительного объема.

Список литературы

1. Шпеник М. Администрирование MS SQL Server 2000: Официальный учебный курс Microsoft / М. Шпеник, О. Следж. – 2-е издание. – 2006. – 640 с.

2. Репликация Microsoft SQL Server 2005/2008 / Под ред. А. Гладченко и В. Щербинина. – 2009. – 452 с.

3. Microsoft Server 2005. Руководство администратора. – М.: Вильямс, 2008. – 546 с.

4. Duffy Joe. Professional .NET Framework 2.0 / Joe Duffy. – April, 2006. – 478 p.

5. *Коннолли Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли, К. Бегг. – М.: Вильямс, 2003. – 762 с.*

6. *Дейт К. Дж. Введение в системы баз данных / К. Дж. Дейт. – 8-е издание. – М.: Вильямс, 2008. – 672 с.*

Поступила в редколлегию 5.03.2009

Рецензент: д-р. физ.-мат наук, проф. Д.Я. Хусаинов, Киевский национальный университет им. Тараса Шевченко, Киев.

СТВОРЕННЯ МЕХАНІЗМІВ РЕПЛІКАЦІЇ НА БАЗІ СТОРОННІХ ПРИКЛАДЕНЬ

Є.В. Івохін, К.Е. Юштін, С.В. Архіпов

В работе рассмотрены основные современные механизмы реализации гомогенных моделей репликации ведущими производителями СУБД. На их основе предложены принципы создания собственных решений для гетерогенной репликации. Приведены примеры технической реализации в случае использования технологии .NET Framework.

Ключевые слова: *репликация снимками, репликация слиянием, транзакционная репликация.*

CUSTOM APPLICATION REPLICATION IMPLEMENTATION

Ye.V. Ivohin, K.E. Yushtin, V.S. Arhipov

In work are considered main modern mechanisms to realization of the homogeneous models replication leading producer DBMS. On their base are offered principles of the making the own decisions for heterogenous replication. Cite an instance technical realization in the event of use of technologies .NET Framework.

Keywords*: репликация picture, replication merging, transaction replication*.