

*Литвиненко И.А.¹, Тищенко Н.А.², Бельский А.Ю.³, Ипполитов В.Д.⁴, Баженов Н.А.⁵,
Афоница А.Д.⁶, Кременная О.А.⁷, Вельдяков В.Н.⁸*

^{1,2,3,4,5,6,7,8} Новосибирский Государственный Университет.

Реализация механизма репликации в среде с объектно-реляционным отображением

Под репликацией подразумевается полная или частичная синхронизация содержимого нескольких хранилищ данных [1]. Репликация может использоваться для организации распределенной работы с данными в сети, состоящей из обособленных хранилищ. С точки зрения каждого узла такой сети, его хранилище является локальной репликой, хранилища же других узлов – удаленными репликами. Реляционная база данных устроена таким образом, что, реплицируя лишь строки таблицы, невозможно обеспечить целостность отношений между таблицами. Таким образом, в распределенной реляционной СУБД репликация обеспечивается либо одновременным проведением транзакции на всех узлах, либо полным копированием всего содержимого одной реплики в другую. Оба метода репликации неприменимы в сети с низкой скоростью передачи данных и низкой отказоустойчивостью. Наша задача состоит в том, чтобы обеспечить репликацию в оффлайн-режиме, когда изменение данных происходит независимо в каждой реплике, а синхронизация содержимого обеспечивается за счет анализа служебной метаданных об этих изменениях, которой обмениваются узлы во время сеанса репликации. При этом возникает проблема поддержания целостности данных во время репликации [2], которая в общем случае неразрешима. Большинство современных приложений на основе РСУБД в своей реализации используют объектно-ориентированную модель и объектно-реляционное отображение (object-relational mapping, ORM) для хранения состояний объектов. При этом нередко целостность данных сводится к поддержанию целостности состояния отдельных объектов (многие реальные приложения осуществляют транзакции над группами объектов одновременно, но часто это делается скорее для повышения производительности, чем для обеспечения целостности [2]). В этих условиях, для обеспечения целостности данных может оказаться достаточно осуществлять репликацию на уровне объектов. Наша задача состоит в реализации механизма

репликации «распространением слухов» [1] для приложения с ORM-слоем.

Наиболее распространенным инструментом для реализации ORM является свободно распространяемая библиотека Hibernate [3] для языка Java. Наше приложение включает в себя модифицированную версию Hibernate. Модификация прозрачна с точки зрения бизнес-логики приложения, т.е. по всем API и поведению модифицированный Hibernate полностью совместим со стандартным. Механизм, обеспечивающий репликацию, отслеживает обращения к этой библиотеке на запись в базу и сохраняет необходимые для репликации метаданные в отдельной таблице той же СУБД. Метаданные включают в себя уникальный в рамках всей сети идентификатор объекта, время его фактического изменения (изменения его атрибутов) и время его сохранения в локальной реплике. Как показывает опыт Lotus Notes/Domino [4], данной информации достаточно для реализации репликации путем «распространения слухов».

Первый прототип репликатора, использующий Hibernate для хранения информации в реляционной СУБД, состоит из следующих частей (рис. 1): прослойка между СУБД и репликатором (Layer), механизм, отслеживающий обращения к Hibernate на запись в БД (Interceptor [3]), клиент-серверное приложение для обмена метаинформацией и репликационным материалом (Replication service), интерфейс пользователя для управления механизмом репликации.

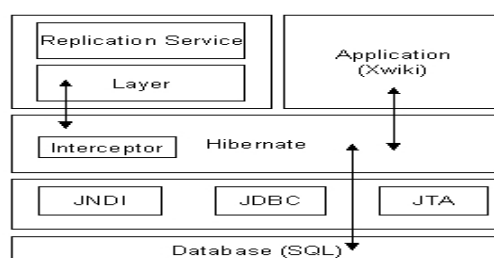


Рис. 1. Структурная организация репликатора.

Таким образом, в простейшем случае интеграция репликатора в существующее приложение подразумевает только сборку приложения с модифицированным Hibernate и включение в поставку приложения клиента и сервера репликатора. Строго говоря, только объектно-реляционный компонент и является языково-зависимым и только его и надо переделывать для интеграции в приложения, написанные на других языках.

Ограничение предлагаемого подхода состоит в том, что перенос каждого объекта представляет собой отдельную транзакцию. Поэтому в процессе репликации бизнес-логике приложения будет доступно частично отреплицированное множество объектов. Поэтому либо бизнес-логика приложения должна рассматривать практически любое сочетание объектов как консистентные данные, либо те объекты, для которых

это требование выполнить невозможно, должны реплицироваться другим способом.

Кроме этого требования, репликатор предъявляет к приложению еще два. Во первых, все подлежащие репликации объекты должны иметь уникальные идентификаторы, неизменные на протяжении всего жизненного цикла объекта. Изменение этого идентификатора репликатор рассматривает как создание нового объекта без уничтожения старого. Во вторых, все подлежащие репликации объекты должны быть каким-то образом сериализуемы, и должен быть простой способ получить сериализованный образ объекта на основе его уникального идентификатора.

Для тестирования и демонстрации возможностей прототипа репликатора, реализованного по данной модели, выбрано приложение XWiki (<http://www.xwiki.org>), использующее Hibernate для хранения информации в реляционной СУБД.

В данном решении узлы репликационной сети обмениваются информацией в режиме «один к одному», еще не поддерживается частичная репликация. Следующий прототип будет обеспечивать одновременную репликацию с несколькими узлами, полную автоматизацию управления репликацией, представленную выделенным узлом-менеджером и соответствующим протоколом обмена данными между менеджером и остальными узлами. Возможным будет как ручное редактирование топологии репликационной сети, так и автоматическое, основанное на статистике частоты изменений на узле, скорости обмена данными с узлом и средней нагрузки [5]. В дальнейшем планируется реализация частичной репликации.

СПИСОК ЛИТЕРАТУРЫ

1. *Иртегов Д.В.* Введение в сетевые технологии. — СПб.: БХВ-Петербург, 2004.
2. *Таненбаум А.Э., ван СТЕЕН М.* Распределенные Системы. Принципы и парадигмы. — СПб.: Питер, 2003.
3. *Christian Bauer, Gavin King* Java Persistence with Hibernate. — USA: Manning Publications Co., 2007.
4. *Роб Курклэнд* Domino версий 5 и 6. Администрирование сервера. — ДМК пресс, 2003.
5. *A. Rowstron, P. Druschel* Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems // IFIP/ACM International Conference on Distributed Systems Platforms – 2001.