

УДК 004.657

Тищенко Н.А.<sup>1,2</sup>, Ипполитов В.Д.<sup>1,2</sup>, Бельский А.Ю.<sup>1,2</sup>, Кременная О.А.<sup>1,2</sup>, Вельдяков В.Н.<sup>1,2</sup>, Баженов Н.А.<sup>1,2</sup>

<sup>1</sup>Новосибирский государственный университет

<sup>2</sup>Экспериментальная лаборатория ФИТ. ООО «Parallels»

## **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 метаинформации 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 (или групп объектов, связанных определенным отношением). 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 с оригинальным. Суть этой модификации заключается в отслеживании обращения к этой библиотеке на запись в базу и сохранении необходимых для репликации метаданных 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».

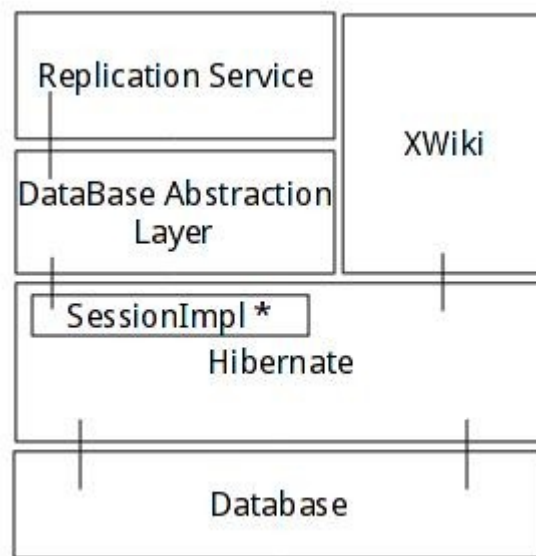


Рис. 1. Структурная организация репликатора.

Разрабатываемый прототип репликатора, использующий Hibernate для хранения информации в реляционной СУБД, состоит из следующих частей (рис. 1): прослойка между СУБД и репликатором (**Database Abstraction Layer**), механизм, отслеживающий обращения к Hibernate на запись в БД (модифицированная версия класса **SessionImpl**), клиент-серверное приложение для обмена метаинформацией и репликационным материалом (**Replication service**), интерфейс пользователя для управления механизмом репликации.

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 модифицировать для интеграции в приложения, написанные на других языках.

Более детальное изучение проблем репликации данных готовых информационных систем показало, что репликация обособленных объектов зачастую неприменима, так как не предусматривает зависимость между объектами. Она может быть представлена наличием внешнего ключа или условием бизнес-логики на сохранение определенных групп объектов в одной транзакции и, возможно, в определенном порядке.

Существующая реализация предъявляет к приложению ряд требований. 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. К тому же появилась необходимость наличие механизма для управления содержимым кэша приложения, в котором, для увеличения производительности, хранятся те объекты, которые когда-либо уже были загружены из БД.

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.

Реализация приведенной модели репликатора, адаптированная к XWiki, на настоящий момент поддерживает репликацию узлов в режиме «один к одному», управляемую при помощи выделенного web-интерфейса.

## 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 Кирклэнд.* Domino версий 5 и 6. Администрирование сервера. — ДМК пресс, 2003.