Организация баз данных
2005. №2(10)

УДК 620.179.16

© 2005 г. **А.А. Ситников**
(Амурский государственный университет, Благовещенск)

МОДЕЛИРОВАНИЕ ПРОЦЕССА РЕПЛИКАЦИИ ДАННЫХ В СУБД ПО СОКРАЩЕННОМУ ЖУРНАЛУ

В работе приводится схема реализации процесса репликации, имеющая более высокую производительность по сравнению с традиционным методом репликации по журналу.

Введение

Репликацией называется процесс приведения неодинаковых состояний двух и более баз данных со схожей структурой в одинаковое состояние, удовлетворяющее заданным критериям процесса репликации с сохранением условия правильности БД. Состояние базы данных при этом определяется набором ее данных и структур. Применимо к реляционным СУБД можно сказать, что состояние БД описывается структурой таблиц и содержащимися в них данными в определенный период времени.

В настоящее время репликация широко используется в большинстве распределенных систем как основной механизм распространения данных между узлами. Существует большое количество разнообразных методов реализации процесса репликации, каждый из которых обычно использует следующие принципы.

Во-первых, узлы распределенной базы данных логически рассматриваются как обособленные базы данных с устанавливаемыми связями (постоянными или временными) с другими базами данных – участниками репликационного обмена.

Во-вторых, узлы обмена обмениваются сообщениями (репликами), содержащими в какой-либо форме информацию о произошедших изменениях.

В-третьих, система репликации должна реализовывать в явной или неявной форме следующие механизмы:

механизм передачи информации между узлами распределенной БД, когда может быть и непрерывно установленная связь, и связь в определенные моменты времени, и пакетный режим, при котором соединение с двумя БД одновременно не устанавливается;

механизм разрешения конфликтных ситуаций: так как система рас-

пределенная, то конфликты будут обязательно, и нужно предусмотреть способы их разрешения;

механизм обеспечения целостности базы данных в результате репликации, – например, если в БД есть подчиненные таблицы, то их обработка должна производиться «сверху-вниз», то есть сначала изменяется запись в родительской таблице, затем в связанных с ней; если же не предусмотреть разрешения этой ситуации, то целостность данных будет нарушена;

механизм обеспечения уникальности идентификаторов записей для различных узлов распределенной БД; когда записи добавляются в таблицу, нужно предусмотреть, чтобы не получилось такой ситуации, при которой на двух узлах появились две различные версии записи в таблице, но под одним идентификатором.

Существует классификация распространенных моделей репликации данных в СУБД по таким признакам как время распространения изменений, модель передачи данных, способы разрешений конфликтов и.т.п. По времени распространения изменений репликационные модели подразделяются на *немедленное* (синхронное) и *отложенное* (асинхронное).

При немедленном распространении изменений транзакция, вызвавшая какое-либо изменение в данных, не подтверждается до тех пор, пока оно не будет произведено на всех партнерах репликации. При этом если произойдет какая-то ошибка, то не будет подтверждена и исходная транзакция. Такой механизм позволяет обнаруживать конфликты немедленно при изменении данных, однако немедленное распространение предъявляет очень высокие требования к надежности линий связи. Обычно при этом широко используется механизм двухуровнего подтверждения транзакции в СУБД (2PC – Two Phase Commit). На практике системы немедленного (синхронного) распространения репликации применяются редко, именно из-за их высоких требований к каналам связи.

При отложенном времени распространения изменений (асинхронном режиме) проведение транзакции никак не связано со временем проведения репликации. На этом принципе построено большинство систем репликации данных, так как они позволяют наиболее полно реализовать такие принципы репликации как относительная независимость узлов данных, низкие требования к линиям связи между узлами и др. Предлагаемый метод репликации по сокращенному журналу основан именно на принципе отложенного времени репликации.

Определение факторов для оптимизации процесса журнальной репликации

Репликация по журналу – это наиболее простой и один из наиболее эффективных по реализации метод. Принцип журнальной репликации сводится к следующему – в базе данных ведется журнал, который через определенные промежутки времени передается другой базе для повторного

воспроизведения всех выполненных в первой базе данных действий. Все действия, производимые в БД (создания, удаления и изменения записей), сохраняются в некотором специально организованном журнале (обычно это отдельная таблица или набор таблиц в самой БД, хотя журнал может быть организован любым образом). Заполнение журнала производится триггерами таблиц, участвующих в репликации, которые записывают в журнал информацию о действии над БД.

Из этого вытекает одно из основных преимуществ синхронизации по журналу – нет необходимости в постоянном соединении с удаленной базой данных во время передачи изменений – часть журнала является автономной единицей и ее можно передать по любым каналам связи в любое время. Журнальная синхронизация реализуется достаточно просто, но коллизии (конфликты) при этом неизбежны, так как процесс, проводящий репликацию, обычно не имеет одновременного доступа к двум базам данных.

Схема репликации по журналу представлена на рис. 1.

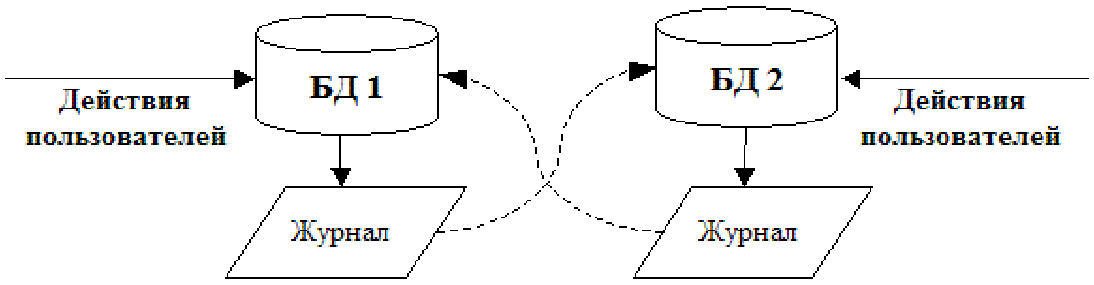

Рис. 1.

Однако метод журнальной синхронизации имеет ряд недостатков. Отметим наиболее существенные из них.

Каждое действие пользователя записывается в журнал, то есть на каждую вставку, изменение, удаление записи в реплицируемой таблице производится вставка одной записи в журнал. Если данные изменяются нечасто, этот метод хорошо работает, но при увеличении интенсивности работы с базой данных происходит очень быстрое разрастание журнала. Это можно легко представить на примере: допустим, была добавлена запись в реплицируемую таблицу, затем эта запись была изменена 10 раз и потом удалена. Логически рассуждая, при связи с БД-приемником репликации информацию об этой записи можно было бы и не передавать – ведь запись добавлена и уже удалена, ее не существует. Однако при репликации по журналу будет повторены ВСЕ действия по созданию, изменению 10 раз записи и в конечном итоге удалению. И журнал будет содержать 12 записей, касающихся этих изменений, совершенно ненужных: ведь объект, представленный записью, уже не существует. А если этих изменений окажется еще больше, журнал будет разрастаться чрезвычайно быстро.

Существует сложность отслеживания произошедших конфликтов. Из-за самой природы журнальной репликации, произошедшие изменения

повторяются в том же порядке, в каком были созданы на узле-источнике. При этом, если произошла неудача и какое-то из изменений не прошло, нужно проводить анализ – из-за чего, нужно дополнительно держать какую-то подсистему, которая информировала бы об ошибках процесса репликации и принимала меры по разрешению ошибок.

При воспроизведении этого журнала узлом-приемником выполняются все записанные действия, которые были выполнены на узле- источнике. Если узел-приемник выступает в роли также узла-источника, то данные, выполненные в процессе воспроизведения журнала, также попадут в журнал узла-приемника и, соответственно, при следующем сеансе связи передадутся обратно отправителю. То есть может возникнуть процесс неконтролируемого лавинообразного процесса накопления репликационной информации, когда узлы передают друг другу информацию об одной и той же записи.

Система журнальной репликации уже используется на практике более 6 лет для передачи данных от дополнительных офисов банка к головному офису. В последнее время, с увеличением интенсивности работы с узлами БД, отмечается очень частое появление описанных проблем. Поэтому было решено разработать новую систему репликации с учетом данных проблем, с упором на решение первой – разрастания журнала, так как ведение полного журнала занимает очень много ресурсов. Было предложено реализовать репликацию на основе метода сокращенного журнала, или маркера измененной записи. Ниже приводим описание метода.

Метод репликации по сокращенному журналу (по маркерам измененных записей)

Сначала определимся, каким образом мы можем сократить объем данных в журнале и, соответственно, сам журнал, а также объем передаваемых между узлами данных. Как мы помним, в журнал попадают все изменения, сделанные с любой записью реплицируемой таблицы в хронологическом порядке. При получении этого журнала БД-приемником в том же порядке, как и в исходной БД, повторяются все действия, совершенные каждой из записей. В конечном итоге запись в БД-приемнике приводится к такому же состоянию, что и в исходной БД. Возникает вопрос – зачем хранить в журнале все промежуточные состояния записи, если в конечном итоге после выполнения сеанса репликации запись приводится к состоянию в исходной БД? Не проще ли в журнале хранить только последнее состояние записи? Если запись будет потом изменяться, то нужно записывать последнее состояние в конец журнала и стирать предыдущие состояния. Для увеличения быстродействия можно не удалять старые записи, а просто сдвигать их в конец журнала. Рассмотрим предлагаемую схему работы на примере выполнения одного действия над записью репликационной таблицы. Принцип работы данного метода показан на рис. 2.

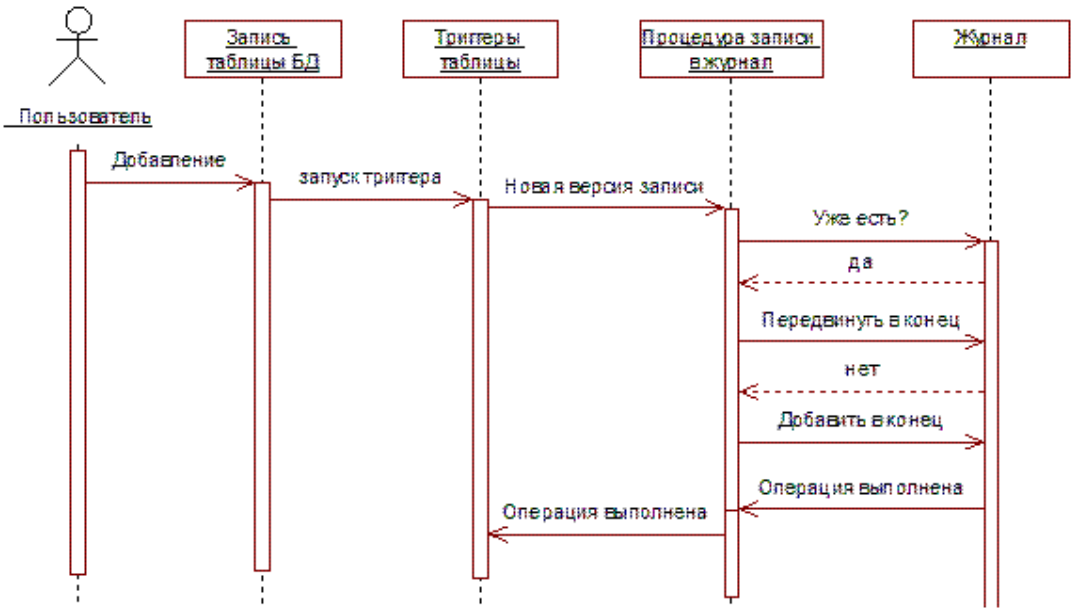

Рис. 2.

Таким образом, журнал может быть существенно сокращен. Для любых действий над записью в реплицируемой таблице запись об этом событии просто переносится в конец журнала, и не может быть более одной строки журнала на запись в реплицируемой БД. И сама строка записи уже приобретает новое свойство – «маркера измененной записи», так как строка журнала определяет, «что произошло» с записью, а ее положение определяет, «когда» это произошло.

Рассмотрим, как можно создать систему репликации, реализующую данные принципы. Сначала определимся, как будет организован сам журнал. Он должен обладать следующими свойствами:

упорядочен по времени выполнения операций над репликационными таблицами;

каждая запись должна определять, в какой реплицируемой таблице изменена запись;

каждая запись хранит информацию о состоянии записи (добавлена/изменена, удалена). Обратите внимание, что нет необходимости разделять состояние «Запись добавлена» от состояния «Запись изменена». По сути это одно и то же событие, показывающее, что появилась новая версия записи в реплицируемой таблице.

Журнал можно реализовать в виде дополнительной таблицы в базе данных. Первое необходимое свойство реализуется добавлением самовозрастающего поля – идентификатора, который будет нумеровать записи журнала. Второе свойство реализуется добавлением полей ссылки на имя таблицы, в которой произошло изменение, и идентификатора записи, которая изменилась. Заметим, что для реализации системы репликации по маркерам измененных записей необходимо, чтобы все реплицируемые табли-

цы имели первичный ключ, и желательно одного типа. Тогда будет просто вести журнал изменений, так как добавляются лишь имя таблицы и значение ключа. Третье свойство реализуется введением поля с состоянием записи в таблице, которое принимает одно из двух значений: либо появилась новая версия записи, либо запись была удалена. На рис. 3 представлена схема таблицы с описанием полей, которая позволит реализовать ведение журнала, где ID – автоинкрементное поле, сквозная нумерация строк журнала, по нему клиент определяет, какую часть журнала забрать с момента последней синхронизации; DB_ID – идентификатор базы данных, в в которой была создана запись для предотвращения зацикливания репликации; TABLE_NAME – в какой таблице произошло изменение; TABLE_ID – идентификатор измененной записи; REC_ACTION – тип изменения (либо появилась новая версия записи, либо запись была удалена).

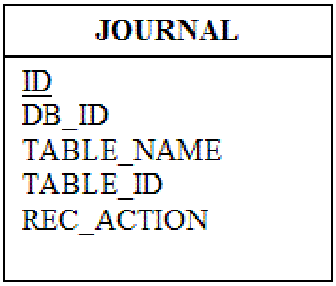

Рис. 3.

При добавлении, изменении, удалении записи в репликационной таблице специальные триггеры, сгенерированные на эту таблицу, добавляют запись в журнал с именем таблицы, идентификатором записи и состоянием записи. При этом если в журнале уже есть записи с предыдущим состоянием этой же записи репликационной таблицы, то они удаляются – они уже больше не нужны. И логически можно представить, что запись репликационной таблицы «помечена» для последующего проведения процесса самой репликации при связи одной БД с другой, а если есть уже старая отметка, то она обновляется. Идентификатор записи в журнале определяет степень «новизны» произошедших событий.

Реализовав запись событий в наш сокращенный журнал, определимся, каким образом будет происходить процесс синхронизации между двумя базами данных.

Рассмотрим простой процесс односторонней синхронизации данных между двумя базами данных, в котором одна база всегда играет роль БД-источника, а другая – роль БД-приемника данных. Для БД-приемника необходимо предусмотреть хранение идентификатора последней записи от БД-источника, чтобы получать только те события, которые произошли с момента последней связи с БД-источником.

Пусть в БД-источнике есть реплицируемая таблица (TABLE), организована запись событий в сокращенный журнал (JOURNAL). Изначально и TABLE и JOURNAL пусты. БД-приемник тоже не содержит данных (рис. 4).

Пользователь, работая с БД-источником, добавляет две записи в TABLE (рис. 5). Соответственно добавляются две записи в JOURNAL с типом состояния «+», что мы условились принимать за новую версию записи.

БД источник

TABLE

ID	Field1	Field2

JOURNAL

ID	DB_ID	TABLE_NAME	TABLE_ID	REC_ACTION

JRL_ID **0**

DB_ID **10**

БД приемник

TABLE

ID	Field1	Field2

LAST_JRL_ID **0**

Рис. 4.

БД источник

TABLE

ID	Field1	Field2
1	данные	данные
2	Зап2	Зап2

JOURNAL

ID	DB_ID	TABLE_NAME	TABLE_ID	REC_ACTION
1	10	TABLE	1	+
2	10	TABLE	2	+

JRL_ID **2**

DB_ID **10**

БД приемник

TABLE

ID	Field1	Field2

LAST_JRL_ID **0**

Рис. 5.

Наступает момент синхронизации БД-приемника с БД-источником. БД-приемник соединяется с БД-источником и запрашивает данные журнала с идентификатором больше чем LAST_JRL_ID (в текущем состоянии – 0). Затем по этому полученному журналу выбирает необходимые данные из таблицы TABLE. В нашем случае добавились две записи, и БД-приемник получил журнал, в котором сообщается, что есть новые версии записей 1 и 2 таблицы TABLE, и процесс синхронизации выбирает данные этих записей, необходимые для вставки в БД-приемник. Затем поочередно выполняются действия журнала у себя с использованием выбранных данных, необходимых для выполнения действий. После успешного выполнения всех полученных действий обновляется указатель LAST_JRL_ID и устанавливается на последнюю запись полученного журнала (в нашем случае LAST_JRL_ID становится равен 2). Рассмотрим, что получается после выполнения первой синхронизации, на рис. 6.

БД источник

TABLE

ID	Field1	Field2
1	данные	данные
2	Зап2	Зап2

JOURNAL

ID	DB_ID	TABLE_NAME	TABLE_ID	REC_ACTION
1	10	TABLE	1	+
2	10	TABLE	2	+

JRL_ID **2**

DB_ID **10**

БД приемник

TABLE

ID	Field1	Field2
1	данные	данные
2	Зап2	Зап2

LAST_JRL_ID **2**

Рис. 6.

Далее проследим, как передаются удаления. Пусть пользователи БД-источника удалили запись с идентификатором 2 и добавили запись 3. После этого в журнале будут сдвинуты маркеры измененных записей 1 и 2 на новые позиции. После чего состояние БД-источника окажется таким, как изображено на рис. 7.

БД источник

TABLE

ID	Field1	Field2
1	данные	данные
3	Новая	Запись

JOURNAL

ID	DB_ID	TABLE_NAME	TABLE_ID	REC_ACTION
1	10	TABLE	1	+
3	10	TABLE	2	-
4	10	TABLE	3	+

JRL_ID **4** *DB_ID* **10**

Рис. 7.

При синхронизации с БД-приемником будут выбраны записи журнала, начиная с LAST_JRL_ID=2, то есть 3 и 4, означающие, что надо удалить 2-запись и добавить третью. Обратите внимание, что исчезла запись журнала с идентификатором 2 по записи табл. 2. Она передвинулась на позицию 3 и будет захвачена БД-приемником при синхронизации. Состояние БД-приемника после синхронизации отображено на рис. 8.

БД приемник

TABLE

ID	Field1	Field2
1	данные	данные
3	Новая	Запись

LAST_JRL_ID **4**

Рис. 8.

Итак, при последующих модификациях записей реплицируемых таблиц их маркеры состояний будут постоянно проталкиваться в конец журнала и попадать под выборку записей с БД-приемника. При этом автоматически решается проблема разрастания журнала, так как на одну запись в реплицируемой таблице всегда существует одна и только одна запись журнала, которая может перемещаться по нему. Что если проследить ситуацию, когда пользователь поменял 10 раз запись номер 1? Рассмотрим это на рис. 9.

БД источник

TABLE

ID	Field1	Field2
1	10 раз	измен.
3	Новая	Запись

JOURNAL

ID	DB_ID	TABLE_NAME	TABLE_ID	REC_ACTION
3	10	TABLE	2	-
4	10	TABLE	3	+
14	10	TABLE	1	+

JRL_ID **14** *DB_ID* **10**

Рис. 9.

То есть запись журнала проталкивается в конец, и все. Процесс синхронизации сразу получит последнюю версию записи с идентификатором 1, без промежуточных ее состояний, что позволит увеличить скорость проведения репликации.

Необходимо отметить, что при приеме журнала БД-приемником, нужно отслеживать, изменения какой таблицы произвести в первую очередь. Это связано с тем, что между таблицами обычно существуют зависимости, или связи. И если пытаться, например, добавлять сначала подчиненную запись, затем родительскую, то добавится только родительская запись. Для решения этой проблемы БД-приемнику можно сверяться по схеме БД, отслеживая связи и сортируя журнал перед исполнением таким образом, чтобы сначала выполнялись действия над корневыми таблицами, а затем над подчиненными. Такого же эффекта можно достичь, если выполнять запись в журнал после всех других действий, заданных триггерами таблицы. Тогда изменения в БД-приемнике будут проходить поочередно,

как и в БД-источнике.

Вместо организации журнала маркеров измененных записей с тем же эффектом можно добавлять дополнительное поле к каждой из реплицируемых таблиц и проставлять там автоматически увеличивающийся номер при каждом изменении. Тогда не нужно будет держать для каждой записи в реплицируемой таблице свой маркер в журнале; снижается расход памяти. Процесс синхронизации просто мог бы собирать данные непосредственно из репликационных таблиц, выбирая их по этому дополнительному полю. Однако такой подход содержит в себе несколько существенных недостатков:

процесс синхронизации должен опросить се реплицируемые таблицы в базе данных в поиске новых версий записей, что очень ухудшает производительность;

не решается проблема удаленных записей, события об удалении нужно куда-то записывать и соответственно вводить дополнительные таблицы удаленных данных, что по сути также представляет собой предлагаемый метод по сокращенному журналу, но в журнал будут попадать только удаленные записи, а измененные помечаться непосредственно в репликационных таблицах;

добавление лишнего поля к каждой из реплицируемых таблиц ухудшает структуру таблицы, а по расходу памяти практически нет разницы – где хранить маркер: вместе с записью или в журнале;

усложняется выборка данных процессом синхронизации. Ему необходимо заранее знать, какие таблицы реплицирует БД-источник, и опрашивать их по очереди, а для этого ему необходимо хранить схему репликации и нет возможности гибкого изменения набора реплицируемых таблиц со стороны БД-источника.

Поэтому введение журнала маркеров измененных записей представляется более простым и оптимальным решением, чем пометка самих записей реплицируемых таблиц. Процесс синхронизации не должен быть связан поочередным опрашиванием всех таблиц в поисках новых данных; используется журнал для определения того, какие записи ему нужны.

Процесс синхронизации данных по журналу в случае односторонней репликации схематично представлен на рис. 10.

Для двусторонней репликации необходимо предусмотреть механизм разрешения конфликтов и разделения идентификаторов вновь создаваемых записей. Для разрешения конфликтов можно ввести дополнительно поле с временем создания записи в журнале и при синхронизации считать более значимым последнее изменение.

Для решения проблемы генерации ключей есть несколько вариантов. Основные два: либо использовать какой-то центральный сервер выдачи идентификаторов, которым будут пользоваться все участники информационного обмена для получения уникальных идентификаторов, либо обеспе-

чить каким-то математическим, логическим и иным образом генерацию уникальных идентификаторов в пределах распределенной базы данных, без связи баз данных друг с другом.

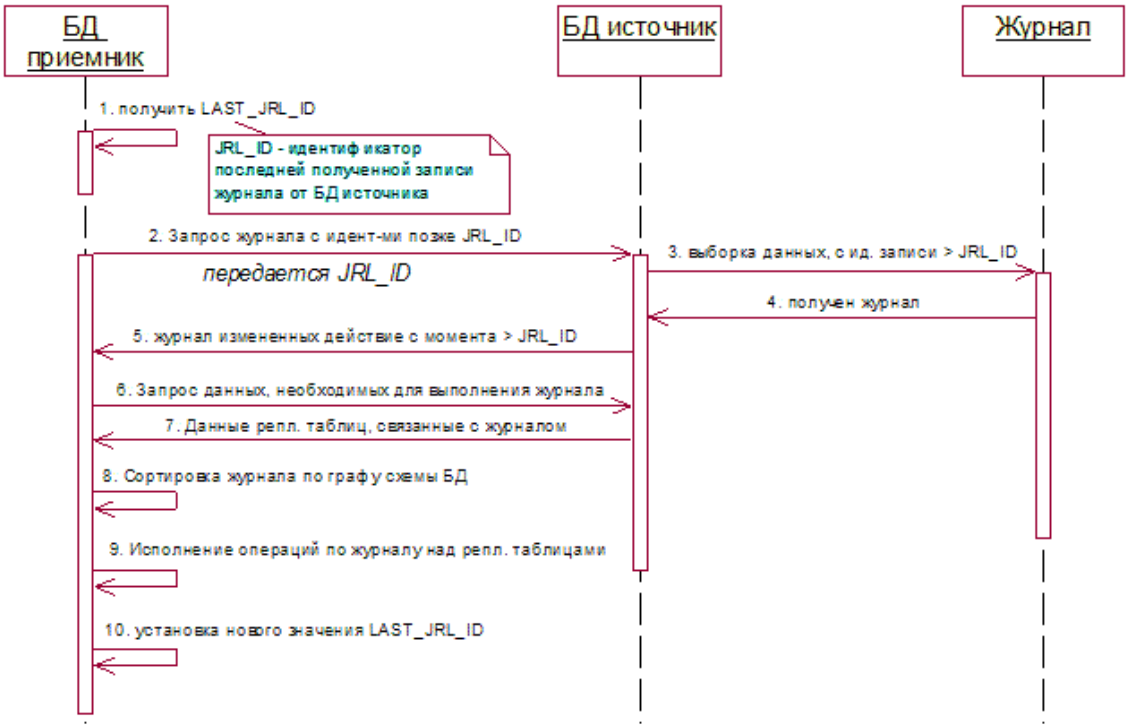

Рис. 10.

Заключение

Итак, предлагаемая схема репликации по сокращенному журналу решает проблему разрастания журнала в системах репликации по журналу за счет того, что на каждую запись приходится одна запись в журнале с указателем на исходную запись и состоянием (версией). Поэтому процесс синхронизации может быстро определить набор измененных записей, получить их с БД-источника и выполнить изменения на БД-приемнике. Данная схема реализации процесса репликации имеет более высокую производительность по сравнению с традиционным методом репликации по журналу за счет:

сокращения объема записи журнала (как следствие ускоряется работа пользователя с записями в БД-источнике);

сокращения объема самого журнала (при N изменениях хранится только одна запись журнала, которая может перемещаться по нему);

получения процессом синхронизации сразу последней версии записи, т.е. ему не нужно повторять все действия одно за другим для приведения записи в последнее состояние.

Таким образом, можно считать что приведенная схема более эффективна, чем репликация по полному журналу, за счет сокращения объема

журнала и повышения производительности процесса синхронизации. В настоящее время идет моделирование предлагаемого метода программными средствами.

ЛИТЕРАТУРА

1. *Григорьев Ю.А., Плутенко А.Д.* Жизненный цикл проектирования распределенных баз данных. Благовещенск: Изд-во Амурского гос. ун-та, 1999.
2. *Плутенко А.Д., Ситников А.А.* Моделирование процесса репликации данных в СУБД методом журнальной синхронизации // Информатика и системы управления. 2004. №1(7). С.16-26.
3. *Плутенко А.Д., Ситников А.А.* Проблемы имитационного моделирования процесса журнальной репликации данных в распределенных СУБД // Информатика и системы управления 2005. №1(9). С.3-15.

Статья представлена к публикации членом редколлегии А.Д. Плутенко.