УДК 004.658

А. Н. Кот
Институт проблем регистрации информации НАН Украины
ул. Н. Шпака, 2, 03113 Киев, Украина

Вопрос построения распределенных информационных систем на основе репликации баз данных

Рассмотрено состояние механизмов репликации в основных промышленных СУБД. Показаны недостатки в существующих реализациях асинхронной репликации с возможностью модификации реплик во всех узлах системы. Предложены направления усовершенствования данного механизма, уменьшения накладных расходов и повышения производительности.

Ключевые слова: *распределенная СУБД, синхронная репликация, асинхронная репликация, транзакции, конфликт модификации данных.*

Введение

Для автоматизации корпоративных организаций актуально построение распределенных информационных систем. Как правило, данные системы строятся на основе СУБД. С этой целью могут использоваться централизованные СУБД, доступ к которым осуществляется по надежным высокоскоростным каналам передачи данных. Однако гораздо чаще такие системы строятся на основе распределенных СУБД (РСУБД). Это является следствием плохих каналов передачи данных, необходимостью приближения данных к месту их преимущественного использования и т.п. Одним из основных механизмов, обеспечивающих функционирование РСУБД, является репликация.

Репликация — это процесс создания и синхронизации нескольких копий информационных объектов, в частном случае реляционных отношений — таблиц, хранимых одновременно на нескольких узлах распределенной системы [1]. Реплицироваться может как вся таблица, так и некоторый ее фрагмент (горизонтальный либо вертикальный). Реплики некоторой таблицы могут быть организованы как на всех узлах распределенной системы, так и на некотором их подмножестве. Репликатор — это программный продукт, осуществляющий репликацию. Репликатор может быть как составной частью СУБД, так и независимым продуктом.

Одна из основных задач, которая должна быть решена при организации репликации — это эффективная конкурентная работа с реплицируемыми данными в разных узлах системы, т.е. необходимость обеспечения возможности модификации реплик во всех узлах системы и при этом достижения максимально возмож-

© А. Н. Кот

ной степени масштабируемости системы, а, именно, возможности функционирования системы на максимально возможном количестве узлов. Для этого надо решить вопрос организации распределенных блокировок и/или разрешения конфликтов модификации, а также улучшить производительность механизмов репликации, так как с увеличением размера систем растет и объем данных, обрабатываемый в них.

Обзор механизмов репликации, используемых в современных программных продуктах

Все основные промышленные реляционные СУБД (Oracle, MSSQL, DB2, Informix) обеспечивают репликацию средствами, встроенными в СУБД, или при помощи вспомогательного продукта, разрабатываемого производителем СУБД. Имеются также независимые программные продукты PeerDirect, GoldenGate, HVR и др., предназначенные, в основном, для обеспечения гетерогенной репликации, т.е. репликации между разными СУБД, а также для репликации больших объемов данных (HVR). Существование таких независимых продуктов свидетельствует о наличии недостатков в средствах репликации, реализованных в СУБД.

Одним из основных способов классификации механизмов репликации является классификация по способу распространения изменений. По этому критерию выделяется два основных класса механизмов репликации:

— с синхронным распространением обновлений, предназначенные для построения сильно связанных систем на основе надежных высокоскоростных каналов связи. Такие механизмы предназначены, в основном, для построения сверхбольших БД;

— с асинхронным распространением обновлений, предназначенные для построения распределенных систем с большим количеством мобильных, периодически подключаемых узлов. Такие механизмы предназначены для построения персональных систем, взаимодействующих с централизованной корпоративной системой.

При использовании синхронных механизмов репликации изменения, вносимые в реплику на одном узле, сразу распространяются на все другие узлы системы. Сразу после фиксации транзакции во всех узлах системы становится доступным актуальное состояние измененных данных. При использовании асинхронных механизмов, после изменения данных в некотором узле, транзакция в нем фиксируется, и данные становятся доступными другим пользователям. После этого система в целом на некоторое время утрачивает свойство целостности, т.е. одни и те же строки таблиц в разных узлах системы имеют разное состояние, и более того, могут быть одновременно модифицированы в разных узлах, что приведет к возникновению конфликтной ситуации. Системы с синхронным механизмом репликации для предотвращения конфликтных ситуаций используют механизм распределенных блокировок, при этом возникают проблемы связанные с ожиданием блокировок, взаимных блокировок и т.д., что приводит к существенному ограничению степени масштабируемости систем, использующих синхронные механизмы репликации. Системы с асинхронным механизмом допускают возникновение конфликтов, которые должны быть определены и разрешены при последующем распространении обновлений. Блокировки и конфликтные ситуации являются

платой за избыточность данных при конкурентном доступе. По сути, это два способа обработки конфликтных ситуаций: пессимистический (блокировки) — не дать конфликту произойти, и оптимистический (разбор конфликтов) — предположить, что конфликта не будет, а если он произойдет, то тогда разрешить его на более поздней стадии [2].

Каждая из основных промышленных СУБД обеспечивает несколько вариантов репликации. Обычно это нижеперечисленные варианты, а также их комбинации:

1) репликация неизменяемых мгновенных снимков данных от узла владельца во все остальные узлы;

2) репликация мгновенных снимков с возможностью их предварительной модификации во всех узлах с накоплением лога изменений, который передается узлу владельцу для внесения в главную реплику;

3) равноправное владение репликами всеми узлами системы с возможностью модификации реплик во всех узлах.

Первые два варианта являются асинхронными по своей сути и реализуются на основе асинхронных обеспечивающих механизмов: установление связи по расписанию, передачи пакетов изменений через системы обмена сообщениями (МОМ).

Третий вариант в разных СУБД реализуется синхронным и/или асинхронным способом. Несмотря на большое количество работ, посвященных усовершенствованию синхронных механизмов репликации [3–6], их результаты пока практически не внедрены в промышленные СУБД. Для обновления реплик в промышленных СУБД до сих пор используются распределенные транзакции, с двухфазной фиксацией, включающие все узлы системы. Проблемы подобного подхода подробно описаны в [7].

Все промышленные СУБД тем или иным образом обеспечивают асинхронные механизмы репликации с возможностью модификации реплик во всех узлах системы. Однако во всех случаях для их реализации используются подходы, приводящие к высокой избыточности данных. В прикладных базах данных (БД) вводятся буферные таблицы или другие структуры хранения для накопления всех изменений данных, которые необходимо реплицировать. Для накопления эти изменения нужно некоторым образом захватывать. Продукты производителей СУБД, обеспечивающие репликацию для захвата изменений, обычно используют анализ лога транзакций СУБД. Продукты сторонних разработчиков, в основном, используют для этих целей триггера, внедряемые БД. Последние захватывают каждую изменяемую строку и записывают ее в структуру, где накапливаются изменения. Наличие триггеров для захвата изменений может оказывать существенное негативное влияние на производительность прикладной системы, особенно при проведении массовой модификации данных. Кроме того, для обеспечения функционирования в данном режиме в таблицы прикладной системы внедряются дополнительные поля, такие как: уникальный идентификатор записи, метка времени изменения, идентификатор узла и т.п. Для актуализации данных полей, в свою очередь, используются дополнительные триггера. Накопленные таким образом изменения в дальнейшем передаются в остальные узлы системы, где применяются для внесения изменений в соответствующие реплики, а также для определения и разрешения конфликтов. Необходимо отметить, что количество накопленных дан-

ных в описанных вспомогательных структурах прямо пропорционально количеству изменений, произведенных в узле системы. В случае массовой модификации данных, возможна генерация значительного объема вспомогательных данных, сопоставимых по объему с полезными данными.

Таким образом, описанный подход к реализации асинхронных механизмов репликации применим для реализации систем с относительно небольшим количеством модификаций данных и относительно частых сеансов связи между узлами. Однако он практически не применим для систем, в которых узлы могут работать автономно длительный период времени и обрабатывать большие объемы данных.

Как уже упоминалось, при использовании асинхронных механизмов репликации с возможностью модификации реплик во всех узлах неизбежно наличие конфликтов модификации данных. Все рассматриваемые СУБД обеспечивают эквивалентные методы разрешения конфликтов:

— стандартные, встроенные методы — актуальным признается изменение строки по приоритету времени изменения, приоритету узла и т.п.

— специальная пользовательская хранимая процедура для разбора конфликтов, оформленная по специальным правилам.

Во втором случае разрешение конфликта переносится на уровень прикладной системы, что существенно усложняет ее разработку.

Практически во всех продуктах, поддерживающих асинхронную репликацию, определение и разрешение конфликтов осуществляется на уровне отдельных строк таблицы. В то же время большинство приложений оперирует информационными объектами, которые сохраняются в виде множества взаимосвязанных строк в разных таблицах. Модификация реплик данных объектов в разных узлах приводит к возникновению межтабличных конфликтов [8]. В лучшем случае, в документации разработчика СУБД приводятся краткие рекомендации по проектированию БД для распределенных прикладных систем, направленные на снижение вероятности конфликтов. В то же время существуют еще более сложные виды конфликтов, например, конфликты, определяемые на основе бизнес-правил системы. Для задания многих из этих правил не хватает декларативных возможностей БД (ограничений ссылочной целостности и т.п.), и конфликт вообще не обнаруживается на уровне базы данных. Разрешение всех нетривиальных конфликтов полностью переносится на разработчиков прикладных систем.

Необходимо также отметить, что при использовании синхронных механизмов репликации во всех промышленных СУБД, аналогично асинхронным методам, используется блокировка данных на уровне отдельных строк таблиц. Здесь возникают абсолютно аналогичные проблемы, например взаимоблокировка при попытке модификации в разных узлах различных частей одного информационного объекта.

Методы повышения эффективности механизмов асинхронной репликации с возможностью модификации реплик во всех узлах системы

В предыдущем разделе описаны две проблемы, возникающие при реализации асинхронной репликации в современных СУБД. Первая — это высокие накладные

расходы, необходимые для обеспечения функционирования данных механизмов, связанные с дублированием данных (дополнительное дисковое пространство) и использованием триггеров, выполняющих модификацию дополнительных служебных данных (дополнительные вычислительные ресурсы). Вторая — это сложности, возникающие в результате разбора конфликтов на уровне отдельных строк в таблицах БД. Для ряда прикладных систем возможно усовершенствование механизмов асинхронной репликации, повышения их производительности и степени масштабируемости систем, построенных на их основе.

Одним из способов решения первой из названных проблем может быть отказ от использования буферных таблиц для накопления изменений. В этом случае для отслеживания измененных строк в реплицируемых таблицах можно использовать фиксированный объем вспомогательных данных для каждой строки, внедряемых в виде дополнительных полей в эту строку. Поддержание таких данных в актуальном состоянии триггерами не требует дополнительных вычислительных ресурсов, так как не будет необходимости в генерации новых данных при модификации данных реплицируемой таблицы.

Одним из возможных путей усовершенствования механизмов обработки конфликтных ситуаций может быть рассмотрение в качестве атомарных объектов, с которыми работают механизмы репликации, не отдельных строк в таблицах БД, а информационных объектов состоящих из множества взаимосвязанных строк. Подобный подход может позволить минимизировать ресурсы, необходимые для обеспечения блокировок, отсечь некоторые виды конфликтов, реализовать более высокоуровневые схемы блокировок, например, механизм передачи владения информационным объектом от одного узла другому по требованию.

Анализ механизмов репликации показал, что ее реализация оказывает существенное влияние на структуру прикладной системы, накладывает ограничения на допустимые операции с БД со стороны прикладной системы, структуру таблиц, ограничения целостности, которые можно использовать в системе. В этом плане для повышения эффективности механизмов репликации целесообразно накопление и систематизация образцов проектирования структур БД для использования в прикладных системах аналогично тому, как это делается для образцов проектирования объектно-ориентированного ПО [9].

1. *Дейт К. Дж.* Введение в системы баз данных: Пер. с англ. — 6-е издание. — М.: Диалектика, 1998. — 784 с.

2. *Wiesmann M., Pedone F., Schiper A., Kemme B., Alonso G.* Database Replication Techniques: a Three Parameter Classification // Proc. 19-th {IEEE} Symp. on Reliable Distributed Systems. — 2000. — P. 206–218.

3. *Holliday J., Steinke R.., Agrawal D., Amr E. A.* Epidemic Algorithms for Replicated Databases // IEEE Transactions on Knowledge and Data Engineering. — 2003. — Vol. 15, N. 3. — P. 1218–1238.

4. *Kemme B., Pedone F., Alonso G., Schiper A.* Processing Transactions over Optimistic Atomic Broadcast Protocols // 19-th IEEE International Conf. on Distributed Computing Systems. — 1999. — P. 424–432.

5. *Alonso G.* Partial Database Replication and Group Communication Primitives // 2nd Europ. Research Seminar on Advances in Distributed Systems. — 1997, March.

6. *Kemme B., Alonso G.* A New Approach to Developing and Implementing Eager Database Replication Protocols // ACM Transactions on Database Systems. — 2000. — Vol. 25, N. 3. — P. 333–379.

7. *Gray J. N., Helland P., Shasha D., O'Neil P.* The Dangers of Replication and a Solution // Proc. of the 1996 ACMSIGMOD International Conf. on Management of Data. — 1996, June. — P. 173–182.

8. *Goldring R.* Update Replication: What Every Designer Should Know. — http://scholar.google.com/url?q=http://www.minet.unijena.de/dbis/lehre/ss2002/seminar/3/Gol95.pdf.

9. *Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж.* Приемы объектно-ориентированного проектирования. Потерны проектирования. — СП.: Питер, 2001. — 368 с.

Поступила в редакцию 10.12.2004