BAKI UNİVERSİTETİNİN XƏBƏRLƏRİ
№3 Fizika-riyaziyyat elmləri seriyası 2006

İNFORMATİKA

ПРОГРАММНЫЙ МОДУЛЬ ЖУРНАЛИЗАЦИИ СИСТЕМЫ УПРАВЛЕНИЯ ТРАНЗАКЦИЯМИ В РАСПРЕДЕЛЕННОЙ БАЗЕ ДАННЫХ

Р.Э.ФАТУЛЛАЕВ
Бакинский Государственный Университет

В работе рассматривается программный модуль журнализации системы управления транзакциями, которая обеспечивает, с одной стороны, взаимодействия прикладных процессов пользователя с распределенной базой данных, а с другой стороны, непротиворечивость распределенной системы. Приводятся назначение, функции и интерфейсы модуля журнализации. Предлагается алгоритм ведения системного журнала. Приводится и доказывается утверждение о том, что предложенный алгоритм ведения системного журнала обеспечивает отказоустойчивость системы и, тем самым, сохраняет целостность информации в распределенной базе данных.

Введение. Распределенные системы (РС) обладают весьма высокой надежностью и устойчивостью к отказам отдельных аппаратных и программных средств. Высокой надежности можно добиться за счет распределения и дублирования ресурсов РС. Однако, для того, чтобы действительно достичь высокой надежности, необходимо обеспечить соответствующее управление распределенными и дублированными ресурсами. Высокая устойчивость обеспечивается за счет введения разнообразных форм избыточности: аппаратной, программной и временной, обеспечиваемой как на аппаратном, так и на программном уровне. Другим важным свойством РС является параллелизм, в связи с чем сокращается время обработки отдельных процессов, что в свою очередь повышает эффективность функционирования системы.

Наиболее успешным и часто применяемым на практике типом РС является распределенная база данных (РБД), представляющая собой интеграцию автономных локальных баз данных, географически распределенных и связанных посредством компьютерной сети [1]. Управление РБД осуществляется соответственно системой управления распределенной базой данных (СУРБД). Важнейшей проблемой, возникающей в любой РБД, является предохранение информационных ресурсов, хранящихся в ней, от

некорректных действий. Для этого в РБД вводятся ограничения целостности — множество формальных правил, устанавливающих и регламентирующих связи между ее ресурсами. Эти правила представляют собой некоторые предикаты, которым должны удовлетворять корректные данные РБД. РБД говорят, целостна (или находится в непротиворечивом состоянии), если значения ее данных удовлетворяют всем ограничениям целостности.

Каждая прикладная программа, предполагающая взаимодействие с РБД, содержит одно или более элементарных действия (чтение, запись и т.д.), переводящих ее из одного состояния в другое. В результате выполнения некоторые из этих действий могут временно нарушить целостность РБД. Для предотвращения таких нежелательных последствий действия над данными РБД группируются в транзакции — неделимые операции, которые являются логическими единицами работы системы и переводят ее из одного непротиворечивого состояния в другое непротиворечивое состояние.

Координация параллельной обработки транзакций в рамках всей РС проводится системой управления транзакциями (СУТ). Система управления транзакциями представляет собой независимую от конкретно применяемой СУБД программу. СУТ выполняется во всех узлах сети, обеспечивающей взаимодействие прикладных процессов пользователя с РБД. Главной задачей СУТ является обеспечение непротиворечивости РБД [2]. Воспринимая обширный поток входной информации от конечных пользователей, такая система должна при любом отказе технических средств и/или ошибочных действиях операторов сохранить целостность РБД, правильно прекратить выполнение начатых транзакций и привести систему к такому состоянию, с которого можно продолжить нормальную работу.

Назначение, функции и интерфейсы модуля журнализации. При совместной работе транзакций содержимое РБД может оказаться противоречивым. Для обеспечения надёжного внесения изменений в базы данных и поддержания (восстановления) непротиворечивого состояния РБД используется системный журнал, в котором в течение всего времени работы системы непрерывно регистрируются изменения, вносимые в РБД. Системный журнал должен быть построен таким образом, чтобы по каждому действию, связанному с изменением РБД, в нём фиксировалась информация, достаточная:

- для внесения изменений в базу данных;
- для аннулирования ветви транзакций при откате.

Каждая запись в системном журнале содержит:

- идентификатор транзакций (TSTR);
- тип выполняемого действия (OPER)
- идентификатор записи в базе данных (IDRC);
- старое значение изменяемого компонента (OLDV);

- новое значение изменяемого компонента (NEWV);
- ссылка на предшествующую запись, относящуюся к той же самой транзакции (REFN).

Каждая транзакция в общем случае состоит из ряда запросов (ветвей) к РБД, реализация которых является распределенной. В выполнении этих запросов участвуют несколько узлов (DM). Каждая DM имеет свой системный журнал, в котором регистрируются запросы, относящие только к данной DM. Кроме того, в системном журнале отмечаются события, связанные с инициализацией, фиксацией или аварийным завершением транзакции.

Структура системного журнала в DM приведена в таблице 1.

Таблица 1

TSTR		OPER	IDRC	OLDV		NEWV		REFN
NOTM	LCCL			OADR	OLEN	NADR	NLEN	

TSTR- временная метка транзакции;
NOTM- номер узла инициирующую транзакцию (TM);
LCCL- локальное время;
OPER- тип действия(операции);
IDRC- идентификатор записи в отношении;
OLDV- старое значение записи;
OADR-адрес старого значения;
OLEN- длина старого значения;
NEWV- новое значение записи;
NADR- адрес нового значения;
NLEN- длина нового значения;
REFN- ссылка на предшествующую запись, относящуюся к этой же транзакции.

Кроме DM, TM также имеет свой системный журнал, в котором помещаются записи о ветвях транзакции. Введение системного журнала в TM необходимо, так как, дает возможность решить вопросы восстановления при некоторых возможных типах отказов. Структура системного журнала в TM приведена в таблице 2.

Таблица 2

TINF	TSTR		NBRN	NODM	TIME
	NOTM	LCCL			

TINF- тип сообщения (PREPARE, COMMIT, ABORT);
TSTR- временная метка транзакций;
NOTM- номер TM;

LCCL- локальное время;
NBRN- номер ветви;
NODM- номер DM;
TIME- тайм-аут.

Чтобы избежать случай, когда возникают отказы с потерей содержимого основной памяти, содержимое системного журнала дублируется в стабильную память. Для обеспечения достаточной надежности стабильной памяти следует предусмотреть введение копий системного журнала.

Всякое изменение БД типа PUT, UPDATE, DELETE выполняется только на заключительной фазе фиксации результата, т.е. после того, как DM-исполнители получат сообщение COMMIT от TM, TM посылая всем DM –исполнителям сообщение PREPARE, опрашивает, готовы ли они зафиксировать изменения. Если при работе модуля журнализации по данной транзакции не было ошибок, то в TM отсылается сообщение READY, в противном случае сообщение NO-READY. На сообщение DM COMMIT или ABORT DM-исполнитель отсылает соответственно сообщение ACK-COMMIT или ACK-ABORT. Во всех случаях соответствующие записи заносятся в системный журнал.

Процедура модуля журнализации вызывается модулем выполнения транзакций всякий раз, когда процессор данных встречает оператор PUT, UPDATE, DELETE и работает до фактического выполнения оператора.

Модуль фиксации результатов вызывает удаление записей системного журнала, относящиеся к транзакции с временной меткой TSTR.

Модуль отката взаимодействует с процедурой модуля журнализации, которая по системному журналу восстанавливает старые значения с помощью процессора данных.

Алгоритм ведения системного журнала

Шаг1. После поступления в DM подтранзакции соответствующей одной из ветвей, в системном журнале DM регистрируется информация об этих подтранзакции. А именно, в журнале фиксируется временная метка транзакции, тип действия, идентификатор записи в базе данных, ссылка на предшествующую запись, относящуюся к той же самой транзакции, адрес и длина старого и нового значения записи, а сами значения записи будут храниться в специальном файле. Выделение самих значений записи в специальный файл удобен тем, чтобы системный журнал был компактен.

Шаг2. Перед посылкой первого сообщения о подготовке к фиксации TM помещает в свой системный журнал запись «подготовка», в которой находятся идентификаторы всех подтранзакций-участников, а также запускает механизм тайм-аута, который будет прерывать координатор по истечению заданного интервала времени.

Шаг3. После получения сообщения PREPARE в стабильную память записывается вся информация необходимая для локальной фиксации

подтранзакций и сведения о готовности фиксации (в случае, когда участник готов фиксировать).

Шаг4. Получив от всех DM-исполнителей сообщение READY TM принимает решение о фиксации транзакции и записывает в стабильную память свое решение. Это означает, в конечном итоге, независимо от отказов, распределенная транзакция будет зафиксирована или аварийно завершена.

Шаг5. Если кто-либо из DM-исполнителей ответил «аварийное завершение» или не ответил до истечения времени тайм-аута, TM принимает решение об аварийном завершении транзакции.

Шаг6. TM информирует всех DM-исполнителей о своем решении и все DM-исполнители помещают в свой системный журнал запись о фиксации или аварийном завершении.

Шаг7. После фиксации результатов в базе данных или аварийного завершения подтранзакции DM-испольнители посылают TM подтверждение и все записи из системного журнала относящие к данной транзакции стираются, а записи в стабильной памяти остаются.

Шаг8. Получив подтверждение от всех DM-исполнителей, т.е. сообщение ACK-COMMIT (ACK-ABORT) TM помещает в свою стабильную память итоговую запись о завершении транзакции, после чего все записи из системного журнала TM, относящие к данной транзакции, стираются.

Утверждение Предложенный алгоритм ведения системного журнала корректен, обеспечивает отказоустойчивость системы и тем самым сохраняет целостность информации в РБД.

Доказательство В системе отказы могут возникнуть в различные моменты времени.

Если DM-исполнитель выходит из строя после получения PREPARE, т.е. до помещения в системный журнал записи о готовности, то в этом случае у TM истекает время тайм-аута, и он принимает решение об аварийном завершении. Все работоспособные DM-исполнители производят аварийное завершение своих подтранзакций. Когда восстанавливается отказавший узел, процедура восстановления производит аварийное завершение его подтранзакций. После чего все записи, относящие к данной транзакции из системного журнала, стираются.

Если DM-исполнитель выходит из строя после помещения в системный журнал записи о готовности, то в этом случае все работоспособные узлы продолжают выполнение транзакции. Отказавший узел восстанавливается, после чего и этот узел продолжает выполнение транзакции.

Если после помещения в системный журнал записи о глобальной фиксации или глобальном аварийном завершении выходит из строя TM, то в этом случае все DM-исполнители должны работать с TM –дублером. TM после восстановления продолжает работу.

Если после сообщения от DM –исполнителя READY или NO-READY нарушается связность сети, то в этом случае истекает время тайм-

аута ТМ, аварийно завершается вся транзакция и соответствующие записи из системного журнала стираются.

Если после внесения изменений в базе данных происходит отказ, то в этом случае, если не возможно восстановить отказавший узел или же нет необходимости продолжить работу транзакции, то выполняется откат транзакции. Процедура отката восстанавливает старые значения в базе данных, как если бы транзакции вообще не выполнялось. Восстановления значений объектов, к которым отказываемая транзакция имела доступ, включает две фазы:

- выявление объектов, значения которых необходимо восстановить;
- приведение выявленных объектов в состояния, которые имели до выполнения транзакции.

Для того чтобы сократить объем действий по восстановлению РБД при отказах, вводится в системе контрольные точки. В системный журнал помещается специальная запись, которая содержит текущее состояние всех компонентов в БД. Эта информация используется для восстановления состояния системы. Во время отката системный журнал считывается в обратном направлении и осуществляется аннулирование действий откатируемой транзакции до записи контрольной точки.

Во время отката транзакции:

1) в системном журнале находятся записи, относящиеся к данной транзакции;

2) находится последняя запись;

3) в обратном порядке, считывая записи изменения, сформируется макропрограмма отката, причем происходит следующая замена:

$$\text{PUT} \rightarrow \text{DELETE}$$
$$\text{UPDATE} \rightarrow \text{UPDATE}$$
$$\text{DELETE} \rightarrow \text{PUT}$$

4) выполняется макропрограмма отката;

5) после фиксации результата стираются все записи системного журнала, относящиеся к данной транзакции.

Этим утверждение доказано.

Заключение Таким образом, для обеспечения непротиворечивого состояния РБД используется системный журнал, в котором в течение всего времени работы системы непрерывно регистрируются изменения, вносимые в базу данных. Предложенный алгоритм ведения системного журнала обеспечивает отказоустойчивость системы и тем самым сохраняет целостность информации в РБД.

ЛИТЕРАТУРА

1. M.T.Ozsu, P.Valduries. Principles of Distributed Database Systems, Prentice-Hall, 1999.
2. Bernstein P., Shipman D., Rothnic J. Concurrency control in a system for distributed databases (SDD-1) // ACM trans. on database systems, 1980, v.5, No 1, p.18-51.

PAYLANMIŞ VERİLƏNLƏR BAZASINDA TRANZAKSİYALARIN İDARƏ OLUNMASI SİSTEMİNİN QEYDİYYAT PROQRAM MODULU

R.E.FƏTULLAYEV

XÜLASƏ

İşdə tranzaкsiyaların idarə olunması sisteminin qeydiyyat proqram moduluna baxılır. Bu sistem bir tərəfdən istifadəçinin tətbiqi prosesləri ilə paylanmış verilənlər bazası arasında əlaqəni, digər tərəfdən isə paylanmış sistemlərin ziddiyyətsizliyini təmin edir. Qeydiyyat modulunun təyini, funкsiyaları və interfeysləri işdə qeyd olunur. Sistem jurnalının aparılması üçün alqoritm təкlif olunur. Bu alqoritmin sistemin nasazlığa davamlılığını və paylanmış verilənlər bazasında informasiyanın tamlığını təmin etməsi haqqında höкm verilir və isbat edilir.

A SOFTWARE MODULE FOR SYSTEM LOGGING OF TRANSACTIONS MANAGEMENT SYSTEM IN THE DISTRIBUTED DATABASE

R.E.FATULLAYEV

SUMMARY

The logging module for transactions management system described in this paper maintains the interaction of user's applied processes with the distributed database system and assures the consistency of distributed system. The purpose, functions and interfaces of the logging module are described. An algorithm for system log maintenance is proposed. A statement that the proposed algorithm for system log maintenance assure the fault-tolerance of system and thus preserves the integrity of data snored in distributed database is proposed and proved.