BAKI UNİVERSİTETİNİN XƏBƏRLƏRİ

№1 ***Fizika-riyaziyyat elmləri seriyası*** ***2012***

УДК 681.3

МУЛЬТИВЕРСНАЯ СЕРИАЛИЗУЕМОСТЬ ТРАНЗАКЦИЙ С ПРИМЕНЕНИЕМ МЕТОДА ВРЕМЕННЫХ МЕТОК В РАСПРЕДЕЛЕННЫХ БАЗАХ ДАННЫХ

Н.А.ГАСАНОВА
Бакинский Государственный Университет
n_gasanova@hotmail.com

В работе рассматриваются вопросы сериализуемости параллельно выполняемых транзакций в распределенных базах данных. Описывается мультиверсная система для управления параллелизмом в распределенных базах данных. Предлагается алгоритм мультиверсного управления с применением метода временных меток, обеспечивающий сериализуемость параллельно выполняемых транзакций.

Ключевые слова: транзакции, распределенные системы, сериализуемость, распределенная база данных, временные метки, мультиверсная модель, параллельное управление

Системы распределенной обработки или распределенные системы (РС), функционирующие в компьютерных сетях, являются одной из наиболее перспективных и быстро развивающихся областей информатики. Такое место они заняли благодаря их существенным преимуществам по сравнению с изолированными системами, функционирующими на базе отдельных компьютеров: РС характеризуются потенциально более высокой надежностью, гибкостью использования вследствие представления пользователю широкого спектра информационных и вычислительных услуг, высокой степенью параллелизма обработки данных. Однако достижение отмеченных достоинств сопряжено с решением комплекса проблем, связанных со значительным усложнением механизмов управления информационными процессами. При этом особое значение имеет обеспечение целостности и непротиворечивости распределенных баз данных (РБД) – одного из основных функциональных компонентов РС [1].

Общепринятым методом обработки информации в РС является транзактная обработка. Это означает, что весь процесс обработки состоит из множества заданий, называемых транзакциями. Транзакцией называ-

ется последовательность операций, производимых над базой данных и переводящих базу данных из одного непротиворечивого (согласованного) состояния в другое непротиворечивое (согласованное) состояние. Транзакция рассматривается как некоторое неделимое действие над базой данных, осмысленное с точки зрения пользователя. В то же время это логическая единица работы системы. Транзакция порождается в некотором узле (назовем его TM_i) и выполняется в одном или нескольких других узлах (назовем их DM^k). Часть работы, проводимой в одном узле DM^k, будем называть подтранзакцией [2].

Проблема обеспечения целостности РБД существенно усложняется при параллельном выполнении транзакций. Поэтому необходимы средства, обеспечивающие сериализуемость множества параллельных транзакций, то есть эквивалентность результата выполнения данной совокупности параллельных транзакций некоторому последовательному выполнению того же множества транзакций. Ввиду фундаментальной важности указанные проблемы привлекли внимание ведущих зарубежных и отечественных специалистов в области распределенной обработки информации.

Мультиверсная сериализуемость. Один из методов решения задачи сериализуемости основан на классическом результате и называется метод блокировки. В указанном методе блокировка на ресурсы устанавливается постепенно при выполнении транзакций, и только после терминации (завершения) транзакции все ресурсы разблокируются.

Этот метод позволяет выполнять вместе только совместимые операции. Однако совместное выполнение транзакций может привести к тупиковой ситуации, когда две или более транзакции одновременно блокируют некоторые ресурсы и каждой из них для полного завершения нужен ресурс, заблокированный другой транзакцией. Разрешить такую ситуацию можно с помощью отката одной или нескольких транзакций, вызвавших конфликт. В результате оставшаяся транзакция получает освобожденный откатываемой транзакцией ресурс и завершает свою отработку. Отметим, что этот метод предполагает значительную централизованность РС, т.е. среди узлов РС выделяется один, обладающий практически всей информацией о выполняемых транзакциях во всех остальных узлах и координирующий их выполнение. Такая централизация не всегда эффективна, так как, например, возможна сильная загруженность центрального узла, что снижает общую производительность РС.

Альтернативный метод сериализации транзакций, хорошо работающий в условиях редких конфликтов транзакций и не вовлекающий транзакции в тупиковые ситуации, основан на использовании метода временных меток. Временная метка (timestamp)— это уникальное число,

которое присваивается транзакции или объекту и выбирается из монотонно возрастающей последовательности, которая часто является функцией от времени. В РС, где каждый узел имеет свой собственный уникальный идентификационный номер, временная метка образуется конкатенацией локального времени и идентификационного номера узла. Упорядочение по временным меткам — это подход, посредством которого последовательность выполнения транзакций подчиняется некоторой приоритетной дисциплине в соответствии их временным меткам. Поскольку каждая транзакция имеет уникальную временную метку, то упорядочение по временным меткам является сериальным и позволяет обеспечить квазипоследовательное выполнение транзакций.

Существуют моноверсная и мультиверсная системы РБД. В мультиверсной системе РБД каждая операция записи элемента создает новую версию элемента данных. Таким образом, последующая операция чтения элемента данных может читать какую-либо из текущих существующих версий. Как моноверсные планы, так и мультиверсные планы делятся на серийные и параллельные. Мультиверсный план серийный, если две транзакции не выполняются параллельно, в противном случае, план параллельный. Серийный мультиверсный план является стандартным, если каждая операция чтения доступна версии элементов данных, созданных последней операцией записи, предшествующей чтению. Отметим, что стандартный серийный план в мультиверсных системах РБД соответствует серийному плану в моноверсных системах РБД. Из свойства согласованности для каждой транзакции, то есть из предположения, что каждая транзакция отдельно сохраняет целостность РБД, следует, что стандартный серийный мультиверсный план должен также сохранять целостность РБД. На основе вышеуказанного можно дать определение мультиверсного критерия сериализуемости [3].

Мультиверсный план является корректным, если он эквивалентен любому стандартному серийному мультиверсному плану множества транзакций. Этот критерий можно интерпретировать следующим образом. Параллельный план множества транзакций в РБД корректен, если он эквивалентен серийному плану транзакции, в котором данные явно ограничены версиями. Мультиверсный план эквивалентный любому стандартному серийному мультиверсному плану называется мультиверсной сериализуемостью [4].

Алгоритм. Узел TM_i инициирует транзакцию, присваивает ей временную метку *ts,* которая образуется конкатенацией значений локального времени и идентификационного номера узла. При этом временные метки переносятся на все операции чтения *(Read)* и записи *(Write*) данной транзакции и обозначаются ts_R и ts_W, соответственно.

Ниже предлагается алгоритм мультиверсного управления с применением метода временных меток, обеспечивающий сериализуемость параллельно выполняемых транзакций.

Шаг 1. Узел TM_i, инициировавший транзакцию T_i с временной меткой ts^*, рассылает узлам – исполнителям DM^k сообщение $req(T_i(ts^*))$ с требованием получить от них распределенный план всех версий для данного элемента V.

Шаг 2. Узлы исполнители DM^k посылают узлу – инициатору ответное сообщение $rep(P_{WR}(V))$, где $P_{WR}(V) = \{..., (ts_W(V^n), ts_R(V^n)), ...\}$

Шаг 3. Узел TM_i проверяет следующее условие: если хотя бы для одного плана $ts_W(V^n) \le ts_W^* \le ts_R(V^n)$, то узел TM_i будет генерировать эту же транзакцию с новой временной меткой, в противном случае узел отправляет сообщение о выполнении транзакции, соответственно, с временными метками ts_W^* или ts_R^*.

Шаг 4. Узлы DM^k получив сообщение о выполнении транзакции проверяют следующие условия:

а) если $ts_R^* > ts_R(V^n)$, то $ts_R(V^n) := ts_R^*$;

б) если $ts_W^* > ts_R(V^n)$, то создается новая версия для данного элемента (V^{n+1}) с временной меткой $ts_W(V^{n+1}) = ts_R(V^{n+1}) = ts_W^*$.

Утверждение. Предложенный алгоритм корректен и распределенный план сериализуем.

Доказательство: Предложенный алгоритм корректен. На самом деле, инициированная транзакция T_i с временной меткой ts^* либо должна быть выполнена, либо если хотя бы для одного плана выполняется условие $ts_W(V^n) \le ts_W^* \le ts_R(V^n)$, то транзакция T_i откатывается и при следующей генерации, она будет иметь большую временную метку и выполнится. Таким образом, алгоритм результативен. С другой стороны, при выполнении транзакции T_i либо $ts_R(V^n)$ будет иметь временную метку ts_R^*, либо будет создана новая версия (V^{n+1}) с временной меткой ts_W^*. В обоих случаях распределенный план будет сериализуем, что и требовалось доказать.

Приведем пример выполнения алгоритма. Рассмотрим вероятность выполнения четырех операций над элементом данных V: две *read* операции транзакций T_1 и T_2 и две *write* операции транзакций T_3 и T_4. Допустим, что $ts(T_1) = 5$, $ts(T_2) = 12$, $ts(T_3) = 16$ и $ts(T_4) = 23$. Предположим,

что план элемента данных $P_{WR}(V)=\{(1,6),(9,11),(13,18),(20,22)\}$ имеет вид как на рисунке 1.

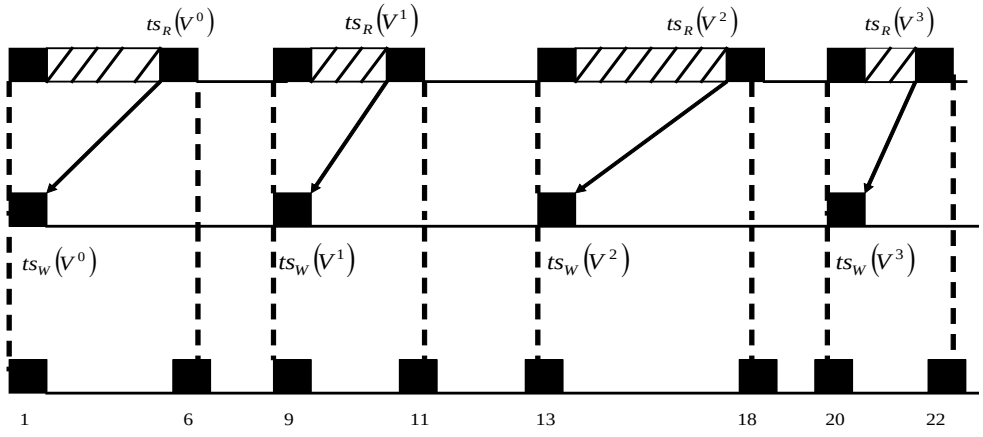

Рис. 1. Мультиверсный план элемента данных до выполнения алгоритма.

Выполнение операций происходит по следующему:

1. Транзакция T_1 будет генерироваться и по шагу 4 для *read* операции транзакции T_1 проверяется следующее условие:

$$ts_W(V^0)\le ts(T_1)\le ts_W(V^1)$$

и временная метка $ts_R(V^0)$ не меняется.

2. Транзакция T_2 будет генерироваться и по шагу 4 для *read* операции транзакции T_2 проверяется следующее условие:

$$ts_W(V^1)\le ts(T_1)\le ts_W(V^2)$$

и временной метке $ts_R(V^1):=ts(T_2)=12$ присваивается значение временной метки транзакции T_2.

3. Транзакция T_3 не будет генерироваться, потому что выполняется следующее условие:

$$ts_W(V^2)\le ts(T_3)\le ts_R(V^2).$$

По шагу 3 транзакция T_3 должна генерироваться с новой временной меткой.

4. Транзакция T_4 будет генерироваться и по шагу 4 для данного элемента создается новая версия $ts_W(V^4)=ts_R(V^4)=ts(T_4)=23.$

После выполнения всех операций план для данного элемента будет иметь вид как на рисунке 2

$$P_{WR}(V)=\{(1,6),(9,12),(13,18),(20,22),(23,23)\}.$$

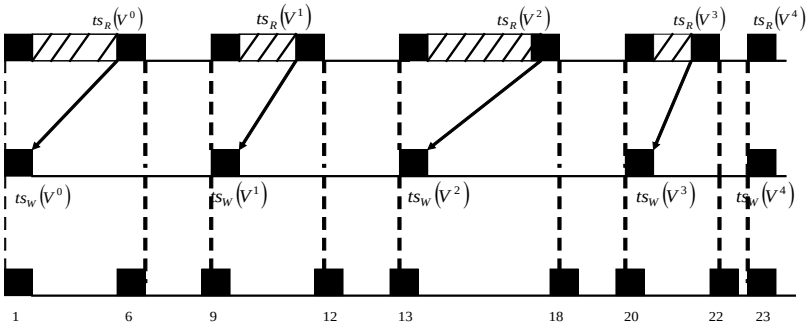

Рис. 2. Мультиверсный план элемента данных после выполнения алгоритма.

ЛИТЕРАТУРА

1. Andrew S. Tanenbaum, Maarten van Stehen. Distributed systems. Principles and paradigms. ISBN:5-272-00053-6, 2002, 880 p.
2. George Coulouris, Jean Dollimore, Tim Kindberg and Gordon Blair. Distributed systems. Concept and Design. ISBN 0-13-214301-1, 2011, 800 p.
3. Chengzheng Sun, David Chen. A Multiversion approach to Conflict Resolution in Distributed Groupware Systems, 20th IEEE International Conference on Distributed Computing Systems (ICDCS'00), 2000, p. 316-324
4. Herlihy M., Weihl W. Hybrid concurrency control for abstract data types. // In proceedings of the 7th ACM-SIGMOD-SIGACT Symposium on Principles of Database Systems (PODS) 1988, p. 201-210.

PAYLANMIŞ VERİLƏNLƏR BAZASINDA VAXT NİŞANI ÜSULUNUN TƏTBİQİ İLƏ TRANZAKSİYALARIN MULTİVERS SERİALLIĞI

N.Ə.HƏSƏNOVA

XÜLASƏ

Məqalədə paylanmış verilənlər bazasında paralelliyin idarə edilməsinin multivers modelinə baxılmışdır. Çox versiyalı «write» əməliyyatların üstünlükləri şəhr edilmişdir. Paralel yerinə yetirilən tranzaksiyaların seriallığını təmin edən multivers alqoritm vaxt nişanı üsulunun tətbiqi ilə təklif edilmişdir.

Açar sözlər: tranzaksiyalar, paylanmış sistemlər, seriallıq, paylanmış verilənlər bazası, vaxt nişanı, multivers model, paralelliyin idarə edilməsi

MULTIVERSION SERIALIZABILITY OF TRANSACTIONS USING THE METHOD OF TIMESTAMPS IN DISTRIBUTED DATABASES

N.A.HASANOVA

SUMMARY

In the given paper, the multiversion model for concurrency control in the distributed database is considered. The advantages of multiple versions of “write” operations are studied. An algorithm of the multiversion control method using timestamps that ensure serializability of concurrent transactions is proposed.

Key words: transactions, distributed systems, serializability, distributed data bases, timestamps, multiversion model, concurrency control

Поступила в редакцию: 29.04.2011 г.
Принята к печати: 05.03.2012 г.