# 一种基于组通信的数据库复制的实现*)

王 勇 焦丽梅

(国家智能计算机研究开发中心 北京100080) (中国科学院计算技术研究所 北京100080)

(中国科学院研究生院 北京100039)

**摘 要** 基于组通信系统的数据库复制将一些单个的数据库组织成一个逻辑数据库,可以提高其可用性和产量并支持更多用户。中间层数据复制用一种中间层的C/S模式来截取客户端和数据库交互的数据流,将用户请求送到所有节点执行。组通信系统和2PC被用来保证复制数据库严格的一致性。本文提出了一种面向操作的复制协议和一种面向事务的记录-重放协议。基于组通信系统的发展和国产的DBMS of Kingbase模式,面向操作的复制服务得到了实现。本文的工作表明利用组通信可以在中间层实现透明复制并可以取得较好的性能。

**关键词** 数据库复制,组通信,原子广播,复制服务

## An Implementation of Database Replication Based on Group Communication

WANG Yong JIAO Li-Mei

(National Research Center for Intelligent Computing System,Beijing100080)

(Institute of Computing Technology,Chinese Academy of Sciences,Beijing100080)

(Graduate School of Chinese Academy of Science,Beijing100039)

**Abstract** Database replication based on group communication system that organizes several single databases as a logical database can improve availability,throughput and support more clients. Replication middleware can catch data stream between server and client with an additional middle layer of C/S mode,send request to multiple database nodes to execute while group communication system and 2PC are used to guarantee strict data consistency. This paper presents an operation-oriented replication protocol and a transaction-oriented Record-Replay Protocol. Based on group communication system of Spread and homemade DBMS of Kingbase,operation-oriented replication server has been implemented. Measurement conducted on two nodes shows that the performance and stability of replicated database are better than non-replicated database in read intensive environment remarkably.

**Keywords** Eager,Database replication,Group communication,Atomic multicast,Replication server

## 1 引言

数据复制可以提高数据库系统的可用性和性能[1]。目前的复制协议可以分成同步复制和异步复制[1]。不加任何限制(update anywhere,anytime,anyway)的同步复制方式当负载增大时,死锁概率、更新操作的延时迅速上升,使系统变得很不稳定[1]。因此,大多数的商业数据库系统实现的是异步的复制方案。

虽然异步复制方案有更好的性能,但是不能支持严格的数据一致性。同时,一般的商业复制方案面向高端应用,价格昂贵,不适合一般的中小企业使用。使用组通讯系统可以以中间件的方式实现数据库的透明复制,同时能够保证数据库的严格一致性[2]。在单节点数据库能力不够的中等应用规模下,通过组通信进行数据库复制能以较低的代价提高系统吞吐率,满足更多用户的访问请求。

机群环境下,网络稳定,网络延时小并且比较固定,节点往往是同构的,便于实现对等的复制。如果将配置DBMS的若干个对等节点通过复制协议整合成一个逻辑数据库系统对外提供统一的服务,就能有效地解决单个数据库能力不够的中型应用的需求。近年来,出现了一些利用组通信来实现数据库复制协议的研究[3~6]。利用组通信提供的原子广播,可靠数据传送,组成员管理等服务,结合两阶段提交来实现同步的数据库复制。有结果[3]显示:在机群环境中实现同步的数据库复制除了能保证严格的数据一致性,也能获得较好的性能。

复制协议的实现方法分两种。一种是修改DBMS[3],在DBMS内实现复制协议,从而获得一个事务要更新的数据并传播到其它节点,同步更新,这样做的好处是可以方便地支持事务,性能也较高,但需要知道具体DBMS的细节,实现复杂。另一种方法是在中间层上实现[4],通过截取客户端和数据库交互的数据流。将用户请求分布到所有节点上同步执行,可以实现透明的数据库复制。由于复制协议不依赖于数据库管理系统,可以实现异构数据库的复制。但是在中间层上实现复制有两个难点:一是如何透明地获得事务并实现面向事务的复制,另一个是如何降低复制中的通信开销和死锁概率,提高复制的性能。

针对第一个问题,本文提出了记录-重放协议(Record-Replay Protocol,简称RRP)。先允许事务在本地执行,执行的过程中记录下事务,当事务请求结束时,将整个事务通过组通信系统全序地广播到各个节点,都执行完成后,通过一轮消息广播,一起提交。如果有节点执行不成功,则在所有节点上回滚。对于第二个问题,采取了一系列的技术,除了文[3]中提到的写本地化外,对记录下来的事务内容进行上下文分析,去掉

*)本文得到北京科技计划项目H030130020330"大型通用数据库管理系统研究"及其子课题"机群系统(Cluster)并行数据库技术研究"的支持。王 勇 直博生,研究方向为计算机系统的性能评价。焦丽梅 博士生,副研究员,研究领域为:体系结构、性能评测、应用负载特性。

读操作，构成一个事务的远地版本，减少其他节点执行该事务的代价；采用冲突队列和数据库结合的并发控制，避免了死锁概率随节点数和负载增大的3次方升高[1]的弱点；利用组通信软件 Spread[7]实现的 EVS 模型[8]和日志，有效地处理节点失效和恢复的问题。

本文第2节讨论相关的工作，第3节实现了面向操作的复制协议，第4节给出了面向事务的复制协议；第5节是面向单操作复制服务器的性能测试；最后总结全文并介绍未来的研究工作。

## 2 相关工作

在分布式系统中，分散在不同节点的多个进程构成一个组，通过相互传送消息来协同工作。但是由于各个节点上的物理时钟不一致和网络延迟的不确定性，各节点间无法确定消息发送的先后顺序。

组通信系统采用 Lamport[9]逻辑时钟，在 VS (Virtual Synchrony)[10]模型的基础上，实现了分布式环境下的一种通信中间件。它维护一个进程组，提供原子多播，数据安全传送和组成员管理服务。应用程序使用它可以方便地实现分布式环境下的容错，数据和关键服务的复制及系统的动态管理。主要的系统有 IsIs，Totem，Horus，Transis 等。近期为解决网络分割和性能等问题，又出现了 Spread[7]和 Ensemble[11]等系统。

2PC[12]仍然是保证复制数据库严格一致性的主要手段。对等的复制要求高速的广播协议和各个节点上事务的执行次序一致。组通讯系统恰好提供这样的服务，所以出现了不少基于组通信的同步数据库复制协议。

早期的工作有的只是提供了一个概念性的协议，没有考虑和 DBMS 相结合出现的具体问题[6]。文[6,13]提出的复制协议局限于存储过程，类似于本文中的面向操作的复制，不适用于一般的事务复制。有人[14]对利用组通信的同步复制进行了分类。

近年来的工作[3,4]结合实际的 DBMS，提出了实用的复制协议，并在文[15]等项目中实现。实现方式主要有两种：在 DBMS 内部实现复制协议和在中间层复制。Bettina Kemme 等人[3]采用前者，复制的对象是事务执行得到的更新数据，用分布式锁和索引锁解决并发冲突。Yair Amir 等人[4]在中间件层上实现了广域网上的数据库复制，复制的对象是每一个数据库操作，侧重于解决广域网上网络分割的问题[2]。本文提出了在中间层上面向操作和面向事务两种复制协议，借用了文[4]的复制模型，重点是解决机群环境下实用的同步复制，并结合其它技术来提高复制的性能。在国产 DBMS----Kingbase 的基础上，我们已经实现了面向操作的协议并进行了性能测试。

## 3 面向操作的复制协议

本协议适合非交互式的事务。在这种事务中，客户端一次将事务的全部内容提交给服务器，单操作事务和存储过程属于该类型。

### 3.1 复制服务器的结构

中间层进行数据库复制的目标是实现一个复制服务器，结构如图1所示。

复制服务器是一个中间件，由三部分组成：事务拦截器，事务分析器和复制引擎。事务拦截器截取客户端和服务器交互的数据流，获取事务内容。事务分析器对数据流进行处理，分析是只读事务还是更新事务(包含更新操作的事务)。复制引擎实现复制协议：只读事务本地执行并提交，更新事务通过组通信系统广播到所有节点上作为一个分布式事务执行。采用两阶段提交保证数据的严格一致性。

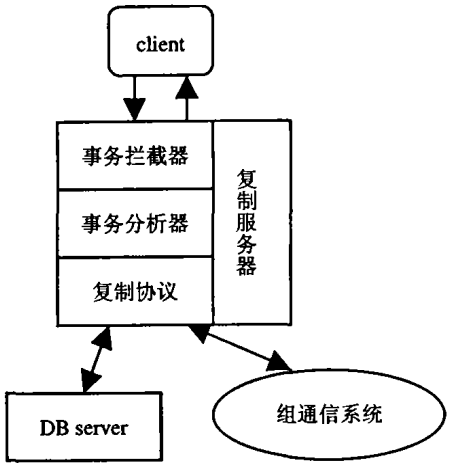


图1 复制服务器结构

### 3.2 面向操作的复制协议

机群系统中由 $n$ 个节点组成，$S=\{s1,s2,\cdots,sn\}$，所有节点是同构的，每个节点配置一个完全复制的数据库和复制服务器，初始状态时，各节点的数据库是一致的，客户端随机选择一个节点为其服务。

经过原子广播后，每个节点得到相同次序的事务序列。但是如果两个事务 $A$，$B$ 对相同的数据项操作，必须强制按全局次序执行，否则，会出现分布式死锁。例如：在节点 $Si$ 上，$A$ 的执行先于 $B$，即 $B$ 等待 $A$ 释放该数据项，在 $Sj$ 上，$B$ 先于 $A$ 执行，$A$ 等待 $B$，这样就形成一个等待回路。为解决这个问题，我们采用冲突队列来保证冲突事务按全局次序执行，在我们的实现中，考虑数据冲突的最小单位是表。

冲突队列是一种死锁避免的方法。在我们的实现中为每一个表建立一个队列，如果事务 $T$ 对表 $A$ 进行更新，则 $T$ 进入 $A$ 的队列。如果两个事务对同一个表进行操作，则它们在同一个队列中排队，按序执行。称这种队列为冲突队列。这种方法避免了死锁，同时也避免了数据库的附加的冲突处理和 Abort 操作。代价是降低了并发程度。考虑更小的数据能提高并发性，代价是增加中间件的复杂程度。

面向操作的复制协议如下：

Phase 1：如果是只读事务，执行并提交；包含更新操作的事务(以下称更新事务)进入下一阶段。

Phase 2：通过组通信系统把事务广播到所有节点(包括广播事务的节点)，每个节点接收到相同次序的事务序列。将事务按先后次序放入相应的冲突队列。

Phase 3：调度处于冲突队列第一个的事务执行。全序地广播执行结果(执行成功或失败)消息到每一个节点。

Phase 4：如果收到所有节点执行成功的消息，提交之，否则，只要一个不成功，则将其回滚。

## 4 面向事务的复制协议

### 4.1 交互式事务复制中的冲突

在事务执行过程中，客户端需要根据服务器的返回结果才能确定下一步操作的事务称作交互式事务，这种事务需要明确的 Commit 或 Rollback 标识事务的结束。在多用户情况下，同一个事务在不同时刻执行，会有不同的执行序列，从而可能得到不同的结果。

非交互式的事务，一次就可以得到整个事务，可直接广播到各节点上按全局一致的次序执行。在交互式事务中，在接收到事务结束标志之前，不能知道事务何时结束，因此不能确定在系统中的执行次序，面向操作的复制协议不再适应。

文[3]中实现了将复制协议加入到 DBMS 的实现中，并且本地事务先在 shadow copy 中执行，执行结束，即收到 Commit 或 Rollback 的时候，将 write set(更新的数据集)和相应的锁传播到各个站点上去执行。在中间件系统中，很难获得 wirteset，只能在节点间广播查询操作。要实现面向事务的复制，就要获取完整的事务并确定事务在系统中的执行顺序。而要获得完整的事务必须先要在本地执行事务。

在面向操作的复制协议中，更新事务是在获得全局一致的顺序之后执行，只要考虑定序后的事务冲突，如上所述，可以用冲突队列解决。

面向事务的协议中，事务的本地执行带来了新的冲突：本地事务之间的冲突以及本地事务和全局事务的冲突。为了降低事务被撤消的频率，要尽可能地利用 DBMS 本身的冲突处理能力。本地事务被中间件接收在本地并发执行，由 DBMS 保证是本地可串行化的，按照接收到 Commit 的先后确定一个本地执行次序，将冲突事务按这个次序全序广播就保证了全局的执行次序和本地一致。当已经广播的事务和本地正在执行的事务发生冲突时，如果本地事务处在本地执行阶段，要将其进行撤销操作。如果事务已经发送，则说明事务已经被完全获取，并确定了全局的次序。文[2]的措施是也执行 Abort 操作，我们的解决方法是执行撤销之后，按它的全局顺序重做该事务，这样可以减少撤销的比率。

### 4.2 面向事务的复制协议

下面的协议实现了上面的设计思路。我们称之为记录-重放协议 Record-Replay Protocol (RRP)。首先说明几个概念。

本地事务：处于本地执行阶段的事务。

全局事务：经过全序广播，确定了全局次序的事务。

事务的标识符：由节点 IP 和本地序列号组成，整个系统唯一。用来区别不同的事务和判断是否是本地事务。

原子性入队：一次性进入所有冲突队列。

同时，假定系统是静态的，即没有节点失效。

面向事务的复制协议如下：

·记录和本地执行：事务先在本地执行，每执行一个操作，检查是否跟全局事务冲突，冲突则撤消该事务，否则记录该操作，直到接收到 Commit 或 Rollback。如果是后者，结束。否则将和更新有关的操作组织成事务的远地版本，按收到 Commit 的先后确定一个本地次序。

·广播：如果是只读事务，提交并终止。如果被撤销，终止。否则将远地版本按照本地次序通过组通信系统全序地广播到其他节点。同时本地事务标识为 executed 状态。

·重放：全局事务的执行。全局事务原子性进入所有冲突队列，如果处于所有相关队列的第一个，调度执行。其执行分三种情况：

1：如果不是本地事务，并且和正在执行的本地事务冲突，本地事务执行撤销操作，如果本地事务已经广播，则标识为该本地事务为 abort 状态。

2：如果是本地事务，且处于 abort 状态，重做该事务。否则必定处于 executed 状态，发送执行成功的消息 Executed Message。

3：如果不冲突，或冲突已经按上面的步骤解决冲突，则执行，如果成功发送 Executed 消息，否则发送 Abort 消息。

·写回：如果收到所有节点执行成功的消息，提交。只要收到一个 Abort 消息，将事务回滚。从相应队列删除。

按照 Spread[7]提供组成员管理，当节点失效时，Spread 会侦测到并发送通知。在事务提交之前记日志，当节点恢复时，根据日志将节点失效之前提交的事务恢复。其它仍然连接的节点提交已经准备提交的事务，回滚已广播但没有确定提交的事务，之后的事务按新的组来继续执行。

## 5 性能测试

### 5.1 性能分析

在面向操作的协议中，读操作和写操作执行方式不同，其比例和执行时间是影响复制服务器性能的主要因素。因此测试中只采用两种代表性的数据库操作：select 和 update，控制它们的执行时间是固定的，分别为 $ts$ 和 $tu$。select 的比例为 $ps$。设未复制的数据库时间 $T$ 内能执行 $X$ 个操作，则有：

$$T=X\times[ts\times ps+(1-ps)tu] \tag{1}$$

update 需要广播事务和消息同步，设增加的时间开销为 $tl$，所以 update 的执行时间成为 $tl+tu$；如果系统由 $N$ 个节点组成，则

select 的比例：$ps'=\dfrac{ps}{1+(N-1)(1-ps)}$

update 的比例：$\dfrac{N\times(1-ps)}{1+(N-1)(1-ps)}$

复制服务器单个节点 $T$ 时间内实际执行操作数是 $X'$，代入式(1)中得：

$$T=X'[ts\times ps'+(tl+tu)\times(1-ps')] \tag{2}$$

复制系统完成的总的操作数为：

$N\times X'\times ps'+X'\times(1-ps')$

复制系统的加速比

$$so=\frac{X'}{x}[1+(N-1)\times ps']$$

$$=N\times\frac{ps+\dfrac{tu}{ts}(1-ps)}{ps+\dfrac{tl+tu}{ts}\times N\times(1-ps)} \tag{3}$$

令 $t'=\dfrac{tu}{ts}$，$t''=\dfrac{tl+tu}{ts}$ 则

$$so=N\times\frac{ps+t'(1-ps)}{ps+Nt''\times(1-ps)} \tag{4}$$

由上面的公式可以看出：三个参数都会对复制服务器的性能产生影响：

1. 读写比例。若读写的执行时间相同，且不考虑复制延迟的影响，有：

$$so=N\times\frac{ps+(1-ps)}{ps+N\times(1-ps)}=\frac{1}{1-ps+\dfrac{ps}{N}} \tag{5}$$

式(5)实际上就是 Amdahl 定律。当全为只读事务时，即 $ps=1$；$so=N$ 系统具有线性加速比。当全为更新操作时，即 $ps=0$，这时系统的加速比是1，也就是相当于一个节点的性能。

2. 复制延时。包括通信延时和并发控制延时。延时越大，性能越差。RRP 协议执行每个事务需要一轮同步消息，文[3]同步消息个数和事务中操作数相同，复制延时明显大得多。和文[4]相比，RRP 减少了 Abort 的比例，有助于性能提高。

3. 读和更新操作的执行时间。$t'$ 越大，加速比越高。当 $t'\to 0$ 即 $ts>>tu$ 时，$so=N$，也能获得线性加速比。因此，在读比例比较大的应用环境中，中间件复制系统会有较好的性能。

### 5.2 测试环境

我们已经实现了面向操作的复制服务器，并在两个节点上进行了相关测试。每个节点是双 Xeon 1.8G；2G 内存；73G SCSI 硬盘；100Mb fast ethernet；操作系统是 Red Hat Linux8.0；内核是：2.4.18-smp；组通信系统是 Spread3.17.1。客户端是 PIII 866MHz；256M 内存；操作系统是 Red Hat Linux9.0；100Mb ethernet。客户端创建若干个线程，每个线程来模拟一个用户，不停地发送请求。

测试用40个表，所有字段相同：一个 integer 类型，做主键；一个 integer 型的随机数做数据；一个50个字节的串类型。初始时每个表有3000条记录，操作分为两种：elect avg(*) from table_i，update table_i set attr1=rand() where ID=rand()%3000；输出结果是每秒钟执行的语句的条数。在此基础上，进行了不同并发用户数和不同读比例下的性能测试。

### 5.3 性能测试

图2给出了400个并发用户时，在不同的读比例的情况下复制的数据库系统和单节点数据库的性能比较，读比例从0到1变化。很明显：复制协议将写操作复制到各个节点上执行，增加了更新操作的开销，因此，在读比例小的时候(≤0.5)，复制数据库的性能不如单节点的数据库系统。随着读比例的提高，复制数据库性能迅速提高，在读比例为1的情况下，性能是单节点数据库的2倍多，出现超线性加速比的原因是两个节点均分了用户，减少了系统调度开销。由此可见在读比例高的环境下，复制数据库有很好的性能。

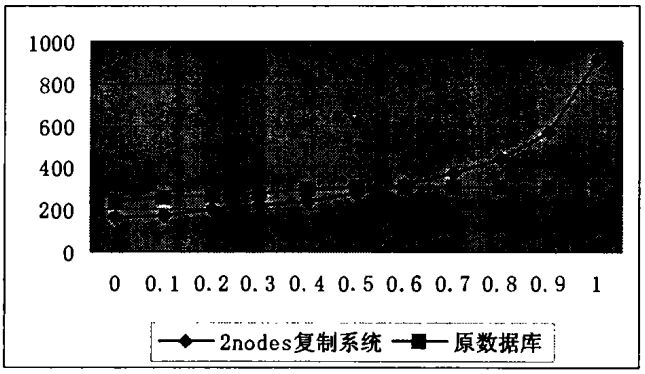


图2 不同读比例的性能比较

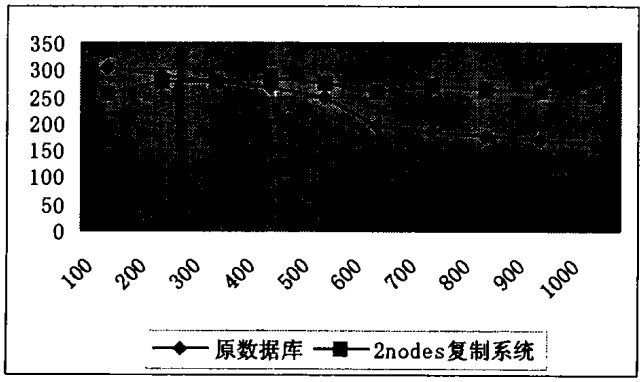


图3 不同并发客户数的测试

图3显示了在 select 的比例为0.7的情况下不同并发用户数的性能比较。100个并发用户的时候，一个节点也不会产生压力，因此复制数据库的优势没有表现出来。当并发用户数增多时，两个副本的复制数据库的性能明显要好于单节点的数据库，并且并发用户数继续增多时，复制的数据库吞吐率更平稳，这也是由于冲突队列避免了死锁，起到了平稳请求的作用。由于多个节点分担了用户请求，中间件系统比单节点的数据库能支持更多的并发用户。

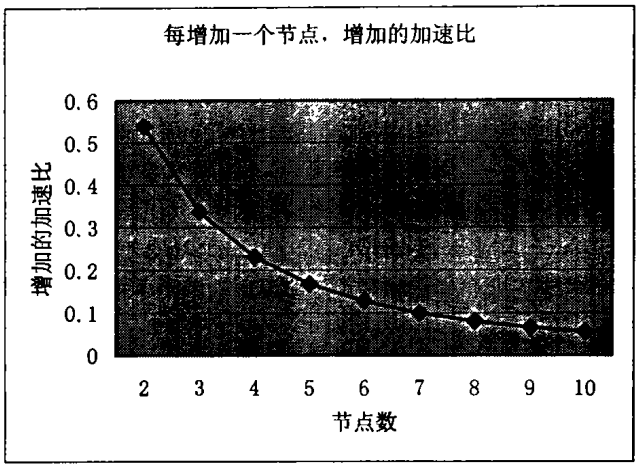


图4 加速比分析

上面的测试验证了5.1的性能分析。由于复制系统通过将读操作分配给多个节点共同执行，提高了读操作的性能。理论上节点越多，性能越好，但式(5)决定了：性能有多大的提高，还是要看读操作的比例。图4显示的是在70%读操作的情况下，每增加一个节点，性能提高的比例。可以看出：随着节点数增多，新增加一个节点获得性能的提高越来越小。同样是增加一个节点，2节点的复制系统比单节点数据库性能增加54%，从6个节点增加到7个时，性能仅增加单节点数据库的10%。考虑到增加节点的成本，不能一味增加节点，2节点的系统已经具有较高的实用价值。

**结论** 本文介绍了利用组通信实现数据库复制的进展，分析了在中间层上实现复制协议的关键问题，提出了面向操作和面向事务的复制协议。根据协议，我们已经实现了面向操作的复制服务器并进行了性能分析和测试。本文的工作表明：利用组通信可以在中间层上实现透明的复制，并可以取得较好的性能。显然，当前的复制协议还不够完善，需要更多地结合数据流的语义分析来确定更细致的冲突关系，这对性能的提高有很大帮助。目前我们正在实现面向事务的复制服务器。

## 参考文献


1 Gray J, Helland P, O'Neil P, Shasha D. The dangers of Replication and a Solution. In: Proc. of the SIGMOD Conf. 1996

2 Amir Yai, Tutu Ciprian. From total order to data replication. In: Intl. Conf. on Distributed Computing Systems, Vienna, Austria,. IEEE. July 2002. 494～503

3 Kemme B, Alonso G. Don't be lazy, be consistent: Postgres-R, A new way to implement Database Replication. In: Proc. of the 26th VLDB Conf. Cario, Egypt, 2000

4 Amir Y, Danilov C, et al. Practical Wide Area Database Replication. Johns Hopkins University: [Technical Report : CNDS-2002-1]

5 Kemme B, Pedone F, Alonso G, Schiper A. Processing transactions over optimistic atomic replication broadcast protocols. In: Proc. of the Int. Conf. on Distributed Computing Systems (ICDCS), 1999

6 Agrawal D, Alonso G, et al. Exploiting atomic broadcast in replicated databases. In: Proc. of Euro-Par, 1997

7 Amir Y, Stanton J. The spread wide area group communication system. Johns Hopkins University, Center for Networking and Distributed Systems: [Technical Report: CNDS 98-4]. 1998


(下转第113页)

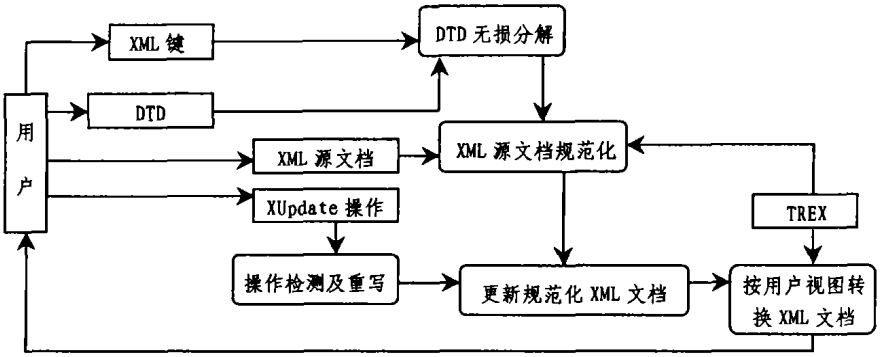


图7 XML 更新系统的结构

**总结和展望** 本文提出了特别适用于更新非规范化 XML 文档的一种新方法，基于规范化的 XML 文档——"XML 范式，XNF"，保持语义直接更新 XML 数据。该方法中，XML 文档的规范化、XL 数据的更新采用 XML 文档的等价转换技术 TREX 来实现，因此，XML 文档的整个更新过程既保持了 DTD 的结构信息、XML 键的语义约束，也有效地避免了 XML 数据的冗余以及更新而产生的各种异常，同时，DTD 的无损分解、待更新 XML 文档的规范化对用户完全透明，并且 XML 最终更新结果也符合用户视图——原 DTD，该过程中 XML 文档的预处理、中间结果和最终结果的生成均基于有效的 XML 等价转换技术 TREX。本文提出的方法对于规模不大的非规范化 XML 的更新操作具有较好的性能。从原更新操作到符合 XNF 操作的自动转换，更新过程中部分转换的等价性证明、更高效的 XML 文档转换技术、XML 文档规范化过程中的优化策略等方面，是我们正在进行的研究工作。

## 参 考 文 献


1 Tatarinov I, Ives Z, Halevy A, Weld D. Updating XML. SIGMOD, 2001

2 Yue K, Xu Z, Guo Z, Zhou A. Constraint Preserving XML Updating. APWeb, 2003. 47～58

3 Buneman P, Davidson S, Fan W, Hara C, Tan W C. Keys for XML. WWW10, 2001

4 岳昆，胥正川，周傲英，宫学庆. 用于更新 XML 文档的注释技术. NDBC, 2002. 47～51

5 Cobena G, Abiteboul S, Marian A. Detecting Changes in XML Documents. ICDE, 2002. 41～52

6 Arenas M, Libkin L. A Normal Form for XML Docments. PODS, 2002. 85～96

7 Fan W, Libkin L. On XML Integrity Constraints in the Presence of DTDs. JACM, 2001

8 Zhou A, Wang Q, Guo Z, et al. TREX: DTD-Conforming XML to XML Transformations. SIGMOD, 2003. 670

9 Robie J, Chamberlin D, Florescu D. Quilt: an XML query language. http://www.almaden.ibm.com/cs/people/chamberlin/quilt_euro.html. March 2000


（上接第108页）


8 Moser L E, Amir Y, Melliar-Smith P M, Agarwal D A. Extended virtual synchrony. In: Intl. Conf. Distributed Computing Systems, 1994. 56～65

9 Lamport L. Time, clocks, and the ordering of events in a distributed system. Communications of the ACM, 1978, 21(7): 558～565

10 Tanebaum A S, van Steen M. Distributed System -Principle and Paradigm. Beijing, Tsinghua University Press, 2002

11 Hayden M, Rodeh O. Ensemble Reference annual. http://www.cs.cornell.edu/Info/Projects/Ensemble/doc.html, Aug. 2003

12 Bernstein P, Hadzilacos V, Goodman N. Concurrency Control and Recovery in Database Systems. Addison-Wesley, 1987

13 Jimenez-Peris R, Patino-Martinez M, et al. Improving the Scalability of Fault-Tolerant Database Clusters. In: Proc. of 22nd IEEE Int. Conf. on Distributed Computing Systems (ICDCS'02), 2002. 477～484

14 Wiesmann M, Pedone F, Schiper A, Kemme B, Alonso G. Database Replication Techniques: a Three Parameter Classification. In: Proc. of the 19th IEEE Symposum on Reliable Distributed Systems (SRDS2000), Numberg, Germany, Oct. 2000

15 Information & Communcations Systems Research Group, ETH Z¨urich and Laboratoire de Syst`emes d'Exploitation (LSE), EPF Lausanne. DRAGON: Database Replication Based on Group Communication, May 1998. http://www.inf.ethz.ch/department/IS/iks/research/dragon.html