第 35 卷 第 1 期 计 算 机 工 程 2009 年 1 月
Vol.35 No.1 Computer Engineering January 2009
·软件技术与数据库· 文章编号：1000—3428(2009)01—0074—02 文献标识码：A 中图分类号：TP311

对等数据管理系统中模式映射的备份机制

赵智超，赵 政
(天津大学计算机科学与技术学院，天津 300072)

摘 要：针对对等数据管理系统中节点因自治性而动态离开和返回网络造成模式映射路径频繁断裂的问题，提出一种基于模式映射备份机制的离开节点绕过方法，向下游节点提供以树状结构组织的上游映射信息。当映射路径中有节点离开时，其下游节点的查询以此映射信息和备份的上游映射，绕过离开节点。仿真和分析的结果表明，映射备份机制的引入提高了查询的可达性，从而可以从更多的局部数据库中获得返回结果。

关键词：对等计算；数据管理系统；查询处理

Backup Mechanism of Schema Mapping in Peer-to-peer Data Management System

ZHAO Zhi-chao, ZHAO Zheng
(School of Computer Science and Technology, Tianjin University, Tianjin 300072)

【Abstract】 To address the problem of frequently broken mapping paths caused by peer dynamically departing and returning the network due to their autonomous nature in PDMS, an absent peer bypassing method is proposed based on schema mapping backup mechanism, which provides tree-structure organized mapping information to the downstream peers. When a peer departs in the mapping path, queries of its downstream peers rely on the mapping information and the backup upstream mappings to bypass the absent peer. Simulation and analysis result shows that the in mapping backup mechanism increases the query achievability, so that results can be returned from more local databases.

【Key words】 peer-to-peer computing; data management system; query processing

1 概述

对等数据管理系统是无中心的分布式数据管理系统，参与的每个节点即本地数据库是自治的，可以有不同的模式，根据需要在模式间建立的映射关系可以指导它们之间的数据共享[1]。模式映射的方向通常设为数据的流动方向。某一模式上的查询参照射入模式映射可再形式化为另一模式上的查询。查询的再形式化在同方向连接的模式映射上可以连续进行，使所有映射可达的模式间实现数据共享[2]。某一模式和其任意映射可达模式之间存在一条或多条映射路径。模式的所有映射路径构成其数据共享的拓扑结构[3]。

对等节点的自治特性使其能够自由地离开和加入数据共享网络。当其离开时，经由它的所有映射路径就会断裂，下游模式上的查询无法继续向上游模式再形式化[4]。上游节点虽然没有离开网络，但其中的数据无法得以共享。对等数据管理系统必须具备一种机制，用来应对这一情况。本文提出的映射备份机制可以使映射路径动态地绕过离开的节点，形成强健的数据共享拓扑。

2 基础设施

对等节点的异质性有 2 个层次，即模型和之上的模式。节点可以有不同的模型，如关系模型、对象模型、XML模型等。具有相同模型的节点可以有不同的模式。模式映射只指出了相同模型上不同模式的映射关系，没有指导模型转换的作用。不同模型的节点需要增加包装器，使其从外部看来具有相同的模型，然后才可以依据相同模型上的模式映射在不同模式间共享数据[5]。模型包装器技术已有文献提供，不在本文讨论范围之内。本文设定的相同模型为XML模型。XML语言是为不同机构间的信息交流而设计，因此，以其作为共同模型更有利于共享数据。其他模型到XML模型的包装器已有技术提供。XML模型上的模式可用XML Schema表示，使用XSLT作为模式映射描述语言。XSLT被设计用来指导XML文档从一种模式变形为另一种模式，因此，可以担任模式映射的描述语言。查询语言也可使用XSLT，这样可以使用与模式映射相同的解析器。

查询发起者异步接收结果，对查询编号可以防止多个查询返回结果的混淆。查询可能经过多次再形式化转发，返回结果的节点不能以查询发送方为结果返回目标。因此，查询消息除了查询自身，还要包括查询编号和结果返回地址。对等数据管理系统的服务程序收到查询后，将其送入本地数据库进行评价，将结果和查询编号一起发送到结果返回地址，并按照该节点的每一射入模式映射将查询再形式化，然后将再形式化后的查询与查询编号、结果返回地址一起发送到对应节点进行评价。节点的射入模式映射通过一张索引表来管理，表中有 2 个属性，即上游节点的地址和映射文件的保存位置。对索引表的每一元组，服务从保存位置取得映射文件，参照它再形式化查询，将再形式化后的查询发送到对应的上游节点地址。此过程的伪代码如下：

```
PeerQuery(QueryNo, InitAddr, Query){
```

作者简介：赵智超(1980－)，男，博士研究生，主研方向：分布式数据库；赵 政，教授、博士生导师

收稿日期：2008-05-17 **E-mail：**zhaozhichaomail@gmail.com

```
LocalQuery(QueryNo, InitAddr, Query);
foreach UpstreamPeer in IndexTable{
 ReformedQuery=Reformulation(Query,
UpstreamPeer.SchemaMapping);
 SendQuery(UpstreamPeer.Addr, QueryNo, InitAddr, Reformed
Query);  }
}
```

3 映射备份机制

映射路径上的中间节点离开后，其下游节点只要获得上游节点到它的模式映射，按照该映射对查询再进行一次再形式化就可以绕过离开节点，共享上游节点的数据。为此，每个节点在上游节点没有离开时，就要取得它的模式映射进行备份。节点保存的模式映射由平等关系变为层次关系，直接射入该节点的原有映射是第 1 级，射入上游节点的映射是第 2 级，在上游节点传来的映射中还有它备份的射入它的上游节点的映射为第 3 级，以此类推。原有的索引表不能表达这样的层次关系，XML 的树状模型可以用来存储。索引树的 XML Schema 定义如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" target
Namespace="http://pdms.tju.edu.cn/IndexTree"
elementFormDefault="qualified"
>
<element name="peer" type="PeerType"/>
<complexType name="PeerType">
 <element ref="peer" minOccurs="0" maxOccurs="unbounded"/>
 <attribute name="address" type="string" use="required"/>
 <attribute name="schemaMapping" type="string" use=
"required"/>
</complexType>
</schema>
```

对等数据管理系统需要添加新的服务以支持下游节点取得其模式映射。该服务收到请求后，先返回其索引树。请求节点将收到的索引树作为一枝嵌套在其索引树对应上游节点下，然后按照这一枝依次取得该上游节点保存的模式映射。按照这一过程，模式映射逐步向下游传递备份。节点会与新的模式建立映射关系，原有映射也会作出调整。下游节点会每隔一段时间索取一次索引树，将其与本地索引树中的对应枝比较，找出新增和改变的映射，向上游节点索取。

原来的查询处理服务需要作出改变。当向上游节点发送再形式化后的查询失败时，可以在索引树对应节点的子节点找出其所有射入模式映射。对每个射入模式映射，参照其对再形式化后的查询再次进行再形式化，然后发往对应的上游的上游节点，这样就绕过了离开的上游节点。如果上游的上游节点也离开，对其采用上述操作，也可将其绕过。递归使用上述操作，可以绕过上游一连串离开的节点。此过程的伪代码如下：

```
BypassQuery(QueryNo, InitAddr, Query, Node){
 foreach ChildNode in Node.Children{
  ReformedQuery=Reformulation(Query, ChildNode. Schema
Mapping);
  if(!SendQuery(ChildNode. Addr, QueryNo, InitAddr,
ReformedQuery))
   BypassQuery(QueryNo, InitAddr, ReformedQuery,
ChildNode); }
}
PeerQuery(QueryNo, InitAddr, Query){
 LocalQuery(QueryNo, InitAddr, Query);
 BypassQuery(QueryNo, InitAddr, Query, IndexTree.Root);  }
```

4 仿真分析

为了检验上述映射备份机制解决节点离开，导致的映射路径断裂问题的能力而设计仿真程序。程序用一台机器上的多个线程来仿真多个节点，没有考虑网络状况对可达性造成的影响。仿真的场景是由 10 个节点组成的映射路径作为查询发起节点的上游。这 10 个节点独立决定何时离开和返回，但它们都以一定的概率在线。在相同节点的在线概率下，对比增加映射备份机制前后查询的可达性。查询发起节点发出 1 000 次查询，统计每次查询到达节点的数目，将其均值作为可达性的测度。在第 1 次仿真中，假设每个节点的在线概率相同。图 1 给出了查询可达节点数随节点在线概率的变化。实线表示的增加映射备份机制后的可达节点数高于虚线表示的原有系统。在第 2 次仿真中，各节点的在线概率随机产生。对于 10 组随机产生的在线概率，1 000 次查询的平均可达节点数的对比如图 2 所示。有映射备份机制的网络具有更高的查询可达性。

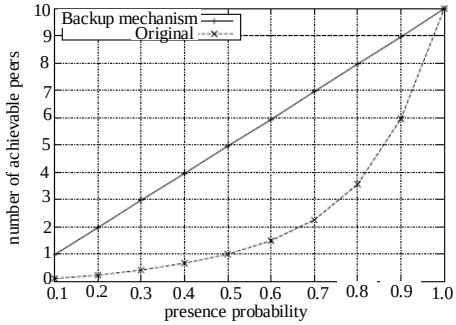

图 1　相同节点在线概率时查询可达性的比较

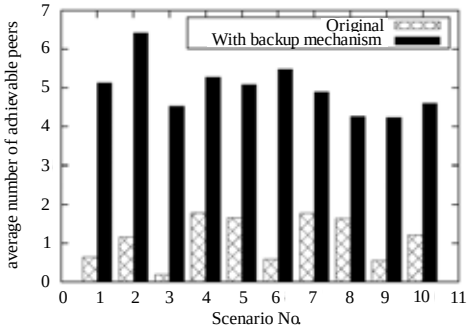

图 2　随机节点在线概率时查询可达性比较

在第 1 次仿真中，备份机制的可达节点数分布服从 10 重贝努利试验 $C_{10}^{n}p^{n}(1-p)^{10-n}$，$p$为每一节点的在线概率。1 000 次查询的平均可达节点数近似 10 重贝努利试验的期望 $10p$。原有系统的可达节点数分布类似几何分布，期望为 $\sum_{n=1}^{9}p^{n}(1-p)+10p^{10}=\frac{p}{1-p}(1-p^{10})$。因为 $\frac{p}{1-p}(1-p^{10})=p(1+p+p^{2}+\cdots+p^{9})$，而$p^{n}\le 1$，所以 $\frac{p}{1-p}(1-p^{10})\le 10p$。这个关系对于任意上游节点数成立。第 2 次仿真中，原有系统的可达节点数期望为 $\sum_{n=1}^{10}n(\prod_{i=1}^{n}p_i)(1-p_{n+1})$，其中，$p_i$为逆流方向第$i$个节点的在线概率。$n$=1 的项为$p_1(1-p_2)$，这时第 1 个节点在线，第 2 个节点离开。在有备份机制的系统中，这种

（下转第 78 页）