doi:10.3969/j.issn.1000-436x.2012.z1.015

基于共享模式的 SaaS 多租户数据划分机制研究

李晓娜[1,2]，李庆忠[1]，孔兰菊[1]，庞成[1]
（1. 山东大学 计算机科学与技术学院，山东 济南 250101；2. 青岛大学 软件技术学院，山东 青岛 266071）

摘 要：对 SaaS 模式下，对共享模式的多租户数据在云中多节点环境的划分问题进行了研究，提出一种支持 SaaS 应用的多租户数据划分模型和算法。与目前主要面向分析型应用并且缺乏事务支持的分区技术和云数据库解决方案进行比较，理论分析和实验结果同时表明，该划分机制能有效地实现云环境中系统规模动态扩展，同时最大限度地降低分布式事务代价。

关键词：SaaS；多租户数据库；共享模式；数据划分
中图分类号：TP311 **文献标识码**：A **文章编号**：1000-436X(2012)Z1-0110-11

Research on multi-tenant data partition mechanism for SaaS application based on shared schema

LI Xiao-na[1,2], LI Qing-zhong[1], KONG Lan-ju[1], PANG Cheng[1]
(1. School of Computer Science and Technology, Shandong University, Jinan 250101, China;
2. School of Software Technology, Qingdao University, Qingdao 266071, China)

Abstract: In SaaS model, the multi-tenant data using shared schema partitioned in multiple nodes of the cloud environment was put forward in advance. Based on this, a data partition model and algorithm for multi-tenants to support SaaS application was proposed. Through comparsion with many data partition technology and cloud database solutions mainly for analysis application and lack of transaction support, the theoretical analysis and experimental results reveal that the proposed data partition mechanism can effectively realize the dynamic scalability of the system in cloud environment, at the same time to minimize the cost of the distributed transaction.

Key words: SaaS; multi-tenant database; shared schema; data partition

1 引言

软件即服务（software as a service）是云计算环境下一种新兴的商业模式。它基于互联网平台，将软件由商品托管为服务。SaaS 最大的特征之一就是单实例多租赁，即一个应用可以被多个租户租赁，并且支持租户的按需定制。由此，基于 SaaS 应用的一个标准共享模式，可以形成满足租户个性化需求的多个独立模式下的多租户数据。为了实现对多租户数据的存储，文献[1]提出了 3 种解决思路：独立数据库、共享数据库独立模式和共享数据库共享模式。其中，共享数据库共享模式（又称共享模式）能更好地体现 SaaS

收稿日期：2012-08-06
基金项目：国家自然科学基金资助项目（90818001）；国家科技支撑计划基金资助项目（2009BAH44B02）；山东省自然科学基金资助项目（2009ZRB019YT, ZR2010FQ026, ZR2010FQ010）；山东省科技攻关计划基金资助项目（2010GGX10105, 2010GGX10107, 2012GGE27036）
Foundation Items: The National Natural Science Foundation of China (90818001); The National Key Technologies R&D Program (2009BAH44B02); The Natural Science Foundation of Shandong Province (2009ZRB019YT, ZR2010FQ026, ZR2010FQ010); The Key Technology R&D Program of Shandong Province (2010GGX10105, 2010GGX10107, 2012GGE27036)

应用基于共享实现定制的特点。因此，本文主要对共享模式的多租户数据库展开研究。

随着 SaaS 应用租户的数目及信息数据量呈几何曲线增长，云中多租户数据管理需求迫切。现有云数据管理策略通常基于数据模式、时间、地域进行数据划分及管理，对于 SaaS 行业应用来说，租户基于同一套应用运行各自的业务，依赖上述特征进行分割，会造成租户数据杂乱地放置在各个节点，导致大量分布式事务的出现。如何在云数据管理策略中引入 SaaS 应用租户特征，在保证多租户数据共享存储的同时，租户数据以合理的方式存放于云中多节点，实现系统动态扩展并且降低分布式事务处理的代价是提高 SaaS 应用运行效率亟待解决的问题。

本文基于共享模式的多租户数据，提出了一种支持 SaaS 应用的多租户数据划分模型及算法。通过统计事务访问信息，建立相关性矩阵，当系统规模进行动态扩展时，对某租户定制的标准模式下所有表进行相关性聚类，形成多个不同“度”的分组。在此基础上，计算权值，产生相关性聚类分组图。按照图的最小分割对其进行划分，形成数据分区，使得跨数据分区的权值最小。从而确保分布式事务代价的最小化。

2　相关工作

分区（分片、划分）技术是实现数据库扩展的主要技术。传统的关系数据库通过水平分区，实现有限个节点的扩展。尽管有许多自动分区方案被提出[2~4]，但使用最广泛的方法主要有循环法（把连续的元组分配到不同的分区）、范围划分法（根据一组谓词将元组进行划分）、散列法（通过计算元组的散列值将其分配到相应分区）。这些数据划分方法主要面向分析型应用，对查询大的数据集非常有效，然而针对事务型工作负载的支持却非常有限，不适合处理事务在多个表上的操作，不具备生成具有 N-to-N 关系模式的合理分区方案能力。对于 SaaS 应用中事务型工作负载，如果访问的元组集合跨越多个节点，会带来大量分布式事务代价，影响系统性能。

当前的云数据库以现有的网络环境和硬件设备为依托，形成了多节点分布式的云环境。相应地云数据管理，为了实现多个节点的弹性扩展和高可用性，也提出了许多数据分区的解决方案。

基于关系模型的云数据管理通常以 DaaS (database as a service)[11,12]模式对外提供数据服务。微软云数据库 SQL Azure[9,10]将逻辑数据库定义为 table group，根据 table 之间的关系划分为 row group 作为数据分区，事务在 row group 内提供 ACID 保证，并且以 row group 为单位，在多节点上存储，在保证可靠性的同时避免了分布式事务的代价。SQL Azure 支持较好的数据外包服务，但是 SaaS 应用租户并不直接租赁数据空间，SQL Azure 数据库无法根据 SaaS 租户信息进行相应的数据划分、冗余管理等，难以保证 SaaS 应用租户的独立性。

麻省理工大学的 Carlo Curino 设计的 Schism[8]提出了事务驱动划分的策略，通过将数据记录映射成顶点，事务访问映射成边来构建图模型，通过图划分来进行数据分割，通过数据关联事务的数量建立副本，能较好地避免分布式事务，确保应用的高效运行。但是，以记录为粒度建立图模型的代价太大，没有考虑 SaaS 应用租户数量多、数据量大的特征。

Google 的云平台[13~15]中，以 Bigtable[5]为代表的 key-value 数据模型，将以 key-value 形式存储的 Table 划分为多个子表，以子表为单位实现数据分布和负载均衡。Megastore[6]在 Bigtable 的基础上加入了事务特征，在自定义的数据模型基础上，把数据分区为 entity group 集合，实现了跨数据中心的数据副本之间强一致性保证。ElasTraS[7]针对共享数据库独立模式下的多租户数据，设计了模式层的分区技术，将跨越多个表的相关数据行划分到同一个分区，并限制事务更新范围在一个分区来获得扩展性和高可用性。

现有的云数据管理策略并未引入 SaaS 单实例多租户的特征，容易导致租户数据混乱放置在多个节点，难以有效降低事务代价。随着系统规模的动态增长，某个租户的数据可能存在于一个节点内，也有可能跨越云中多个节点。在这种背景下，如何将多租户数据在多节点进行合理划分，在实现系统动态扩展的同时最大限度地减少分布式事务代价是需要研究的一项重要工作。

3　多租户数据划分模型

本节首先介绍支持 SaaS 应用、基于共享模式的多租户数据。以此为基础，提出了为实现多租户数据分区而建立的数据划分模型。

3.1　支持 SaaS 应用的多租户数据

独立软件开发商（ISV）面向传统的数据库开发应用软件，经过 SaaS 平台的交付部署，成为支持“共享数据库、单实例多租赁”的基于云计算的 SaaS 应用，并且满足不同租户通过平台对应用进行租赁和个性化定制，获取属于自己的虚拟应用同时确保其定制结果的正确性。SaaS 模式下，租户所有的业务数据存储在 SaaS 平台服务提供商的共享数据库中，最终用户感受不到使用的实例在同一时间也为其他客户所共享。由此形成的多租户数据如图 1 所示（只画出一个应用）。

多租户数据采用共享模式，以关系数据库实现存储，一方面能避免数据表格剧增引起的数据库性能下降，另一方面能更好地支持 SaaS 应用基于共享实现定制的特点。图 1 所示的多租户数据，自上而下分为 4 个层次，分别为租户逻辑视图、标准数据模式、全局存储和节点存储。相关定义如下。

定义 1　标准数据模式　开发商关于某应用的数据定义。通常由多个关系及某些关系实例的集合构成。App_i=<∪{T_i}，DS>，其中，App_i 表示某开发商定义的标准应用，T_i 表示此应用中定义的编号为 i 的关系，DS 表示在定义应用时，所定义的初始数据。

定义 2　租户逻辑视图　租户通过对标准数据模式 App_i 进行定制得到的数据定义。$TxLV$= <App_i，∪{R_i}，TxD，$TxDESC$>，其中，App_i 表示租户 Tx 定制的应用，R_i 表示编号为 i 的关系，TxD 表示租户的初始数据及在运行过程中产生的数据，$TxDESC$ 表示租户的定制描述信息。

假设应用 $App1$=<{department，student，course}，DS>。Department=(编号，学院)，Student=(编号，姓名，学院)，Course=(编号，课程名，学分)。基于标准数据模式，租户 A、B、C 分别进行定制。其中，灰色字段表示标准字段，白色表示定制字段。如图 2 所示，形成了各个租户的逻辑示意图。

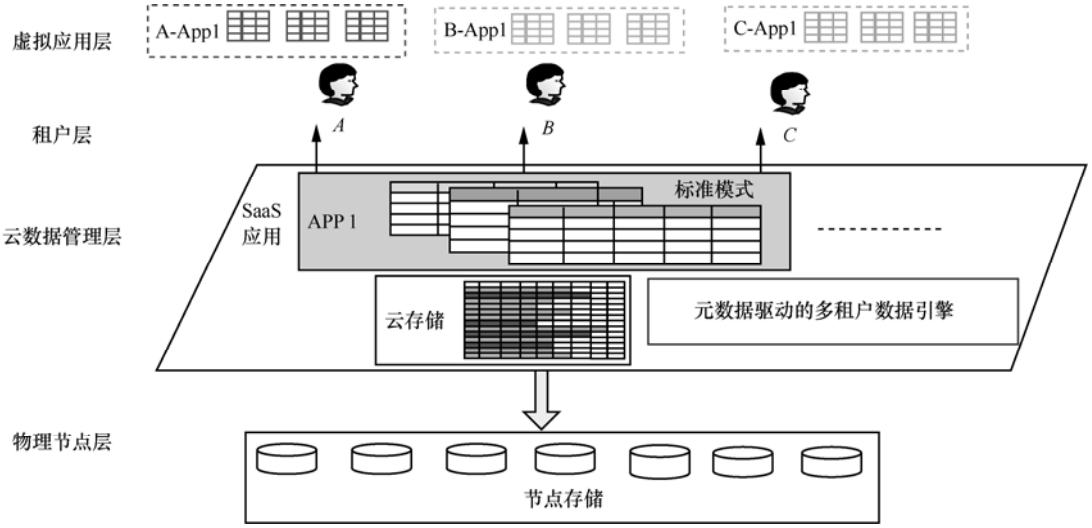


图 1　支持 SaaS 应用的多租户数据

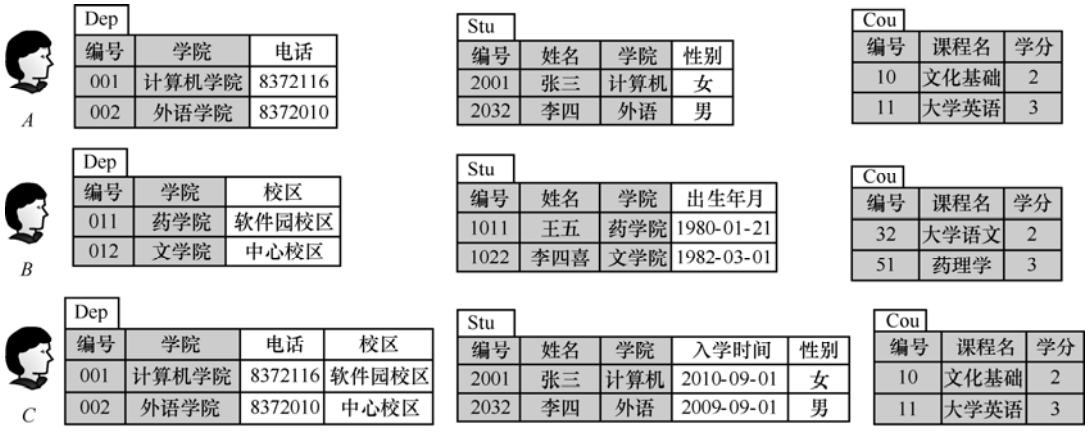


图 2　租户逻辑示意

定义 3　全局存储 SaaS 平台中数据存储的整体模型，它包括平台上所有应用的租户视图存储信息。CSM =<∪ {$CTable_j$}，$TDESC$，SF >，其中，$CTable_j$ 为编号等于 j 的存储视图。其结构可以为稀疏表、键值对、稀疏表与键值对相结合等模型。$TDESC$ 为租户视图的存储描述信息集合，SF 是租户视图和全局存储之间的映射集合。

上述简单示例中，对应的全局存储模型为 CSM =<∪ {$CTable_j$}，$SDESC$，SF >，其中，$CTable_j$ 采用一个关系存储模型，命名为 *DataTable*。*DataTable* 利用预留的 500 个“flex 列”，以 varchar 形式来承载不同类型的数据。其中，*GUID* 表示记录的唯一编号，*ObjID* 表示租户定制的表对象编号。*DataTable* 的结构如图 3 所示。

定义 4　节点存储 节点存储是租户关系及实例的具体存储视图，是数据存储最低一级的逻辑描述。节点存储视图 NSM =<∪ {$TxPartition_i$} ，*METADATA*>，其中，$TxPartition_i$ 是隶属于该节点的某租户 Tx 的数据分区，METADATA 描述了上述各类描述数据的集合。

图 1 所示系统运行时通过基于元数据驱动的多租户数据引擎产生虚拟应用组件，如租户逻辑视图的表、记录等。基于共享的存储模式对于单个租户来说，其数据分散粒度大且相邻记录之间非同质的概率较高，对数据查询等操作的效率产生了影响。如何将共享模式下的多租户数据以合理的方式划分到云中多节点，在实现系统动态扩展的同时尽量减少分布式事务的代价是要研究的重点。

3.2　多租户数据划分模型

基于共享模式和 SaaS 应用操作特征，建立多租户数据划分模型，能够建立统一的多租户数据管理机制和数据划分算法。多租户数据划分模型如图 4 所示。

数据划分模型分为 2 层，分别定义为 Tenant 层、Group 层。

Tenant 层　*DataTable*=<∪ {*TxD*}>,*TxD* 表示属于租户 *Tx* 的所有应用数据的集合。通过 *DataTable* 的 *TenantID* 列，能迅速判断出某条数据记录所属的租户集合。

Group 层　*TxD*=<∪ {*Group*(*i*)}>,根据本文建立的相关性聚类算法，租户 *Tx* 的数据划分为多个分组，*Group*(*i*)表示租户 *Tx* 具有某种相关性的一组定制表形成的集合，即 *Group*(*i*)=<∪ {*Table*(*j*)}>。在系统维护负载均衡和数据迁移的过程中，以 *Group*(*i*) 作为数据管理的单位，实现动态扩展。

通过多租户数据划分模型，将共享模式下最初分散且不同质的多租户数据以 Group 为粒度，且相关性程度高的数据聚集在一起，在各个物理节点实现多租户共享存储。满足系统动态扩展的同时，最大限度地减少分布式事务代价。

4　基于模型的多租户数据划分算法

基于建立的多租户数据划分模型，通过统计事务信息，建立相关性矩阵。当系统规模进行动态扩展时，对某租户定制的标准模式下的所有表进行相关性聚类，形成多个不同“度”的分组。在此基础上，计算权值并产生相关性聚类分组图。按照图的最小分割对其进行划分，形成数据分区，使得跨数据分区的权值最小，从而确保分布式事务代价的最小化。

4.1　数据划分步骤

Step1　初始状态，假设系统有 1 个结点（或者小规模的几个分布式节点），*A*、*B*、*C* 3 个租户，租户的数据量较小，在此环境下没有分布式事务。如图 5 中的初始状态所示。

GUID	*ObjID*	*TenID*	*Value* 0	*Value* 1	*Value* 2	*Value* 3	*Value* 4	*Value* 5	…
0001	1	*A*	001	计算机学院	8372116				
0002	1	*B*	011	药学院	软件园校区				
0003	1	*C*	001	计算机学院	8372116	软件园校区			
0004	1	*C*	002	外语学院	8372010	中心校区			
0005	2	*A*	2001	张三	计算机	女			
0006	1	*A*	002	外语学院	8372010				
0007	2	*B*	1011	王五	药学院	1980-01-21			
0008	3	*C*	10	文化基础	2				
…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…

图 3　DataTable 结构

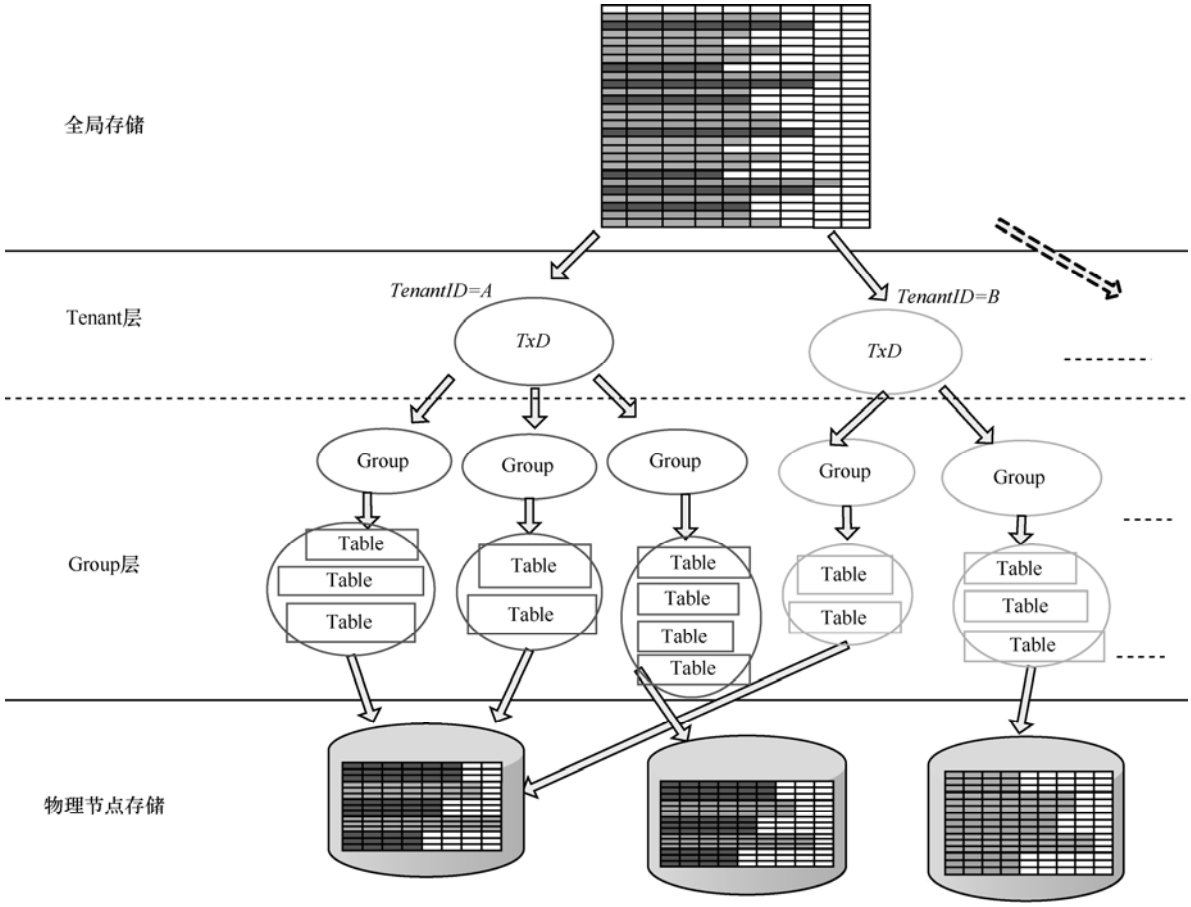


图 4　多租户数据划分模型

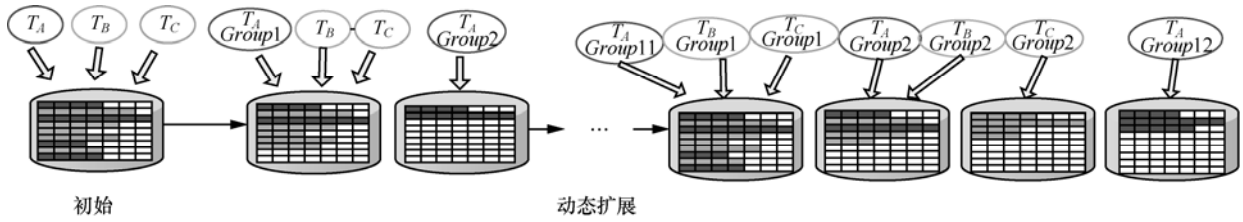


图 5　数据划分过程

租户基于标准数据模式进行定制扩展，相关参数定义如下。

TSet：标准模式下的表集合，*TSet*={$table_1$, $table_2$,…,$table_n$}；

TxTSet：租户 *Tx* 定制的表集合，*TxTSet*={$table_{1x}$, $table_{2x}$,…,$table_{nx}$ }。

租户 *Tx* 定制的每一个表 $table_{ix}$ 有 2 个属性来表示<$TranS_iSet$,*si*>，其中，$TranS_iSet$ 代表一段时间内访问 $table_{ix}$ 的事务集合，*si* 表示 $table_{ix}$ 的大小。

定义 5　相关性　如果事务同时访问几个表，表明它们之间有相关性。

定义 6　相关值　同时访问指定表的事务数目。

2 个表之间的相关值 $Relevance_{ij}$：$Relevance_{ij}$=Count($TranS_iSet \cap TranS_jSet$)。

多个表（表集合 Tset）的相关值：*RelevanceTset*=Count($TranS_iSet \cap TranS_jSet \cap TranS_mSet \cap \cdots \cap TranS_nSet$)，其中，$table_m$, $table_n$ 属于表 *Tset* 集合。

Step2　统计事务信息，建立相关性矩阵。根据一段时间事务访问的情况，统计每个租户基于标准模式定制的表之间的相关值。租户 *Tx* 的相关性矩阵 ***TxRM***，矩阵项 $RM_{ij}=Relevance_{ij}$，其值为同时访问 2 个表的事务数目。***TxRM*** 对角线上的数值，表示访问某一个表的事务数目。

TxRM 是一个 $n \times n$ 的矩阵，n 表示租户 *Tx* 定制的表的数目。图 6 简单表示出系统自初始状态，经过一段时间事务访问，租户 T_A 的相关性矩阵。

$T_A table_1.TranS_1Set = \{t_1, t_2, t_3, t_4, t_5\}$
$T_A table_2.TranS_2Set = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$
$T_A table_3.TranS_3Set = \{t_1, t_2, t_6, t_8, t_9\}$
$T_A table_4.TranS_4Set = \{t_6, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}\}$
$T_A table_5.TranS_5Set = \{t_1, t_2, t_3, t_4, t_8, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}\}$

$$T_x RM_{ij} = Relevance_{ij} = Count(TranS_i Set \cap TranS_j Set)$$

$T_A RM$=

	$table_1$	$table_2$	$table_3$	$table_4$	$table_5$
$table_1$	5	…	…	…	…
$table_2$	5	7	…	…	…
$table_3$	2	3	5	…	…
$table_4$	0	1	3	8	…
$table_5$	4	4	3	6	10

图 6　相关性矩阵

Step3　系统规模动态扩展时，基于相关性矩阵及元数据信息，对租户表集合进行聚类分组，形成多个度为 i 的相关性分组 GR^i 组成的集合 $GSet$。i 代表分组的度。

定义 7　分组的度　分组 RG^i 中，包含多个表集合，每个表集合的相关值都为 i，i 称为此分组的度。如图 7 最下部所示。

GSet 集合　经过聚类后形成的分组集合。$GSet=<\cup RG^i \cup RG_{solo}>$。$RG^i$ 表示度为 i 的分组。若某个表与其他表之间没有相关性，则聚类到 RG_{solo} 分组。初始情况下，$GSet=\Phi$。

SaaS 模式下，多租户数据源于一个标准数据模式进行定制扩展，根据系统运行过程中事务访问的特征及定制产生的元数据信息，对租户表集合进行聚类分组后，产生的 $GSet$ 集合有可能被多个租户共享。$GSet$ 集合中的表 $table_1$'代表了源自标准数据模式的 $table_1$ 被租户集合$\{T_A, T_C, T_D, \cdots\}$中的多个租

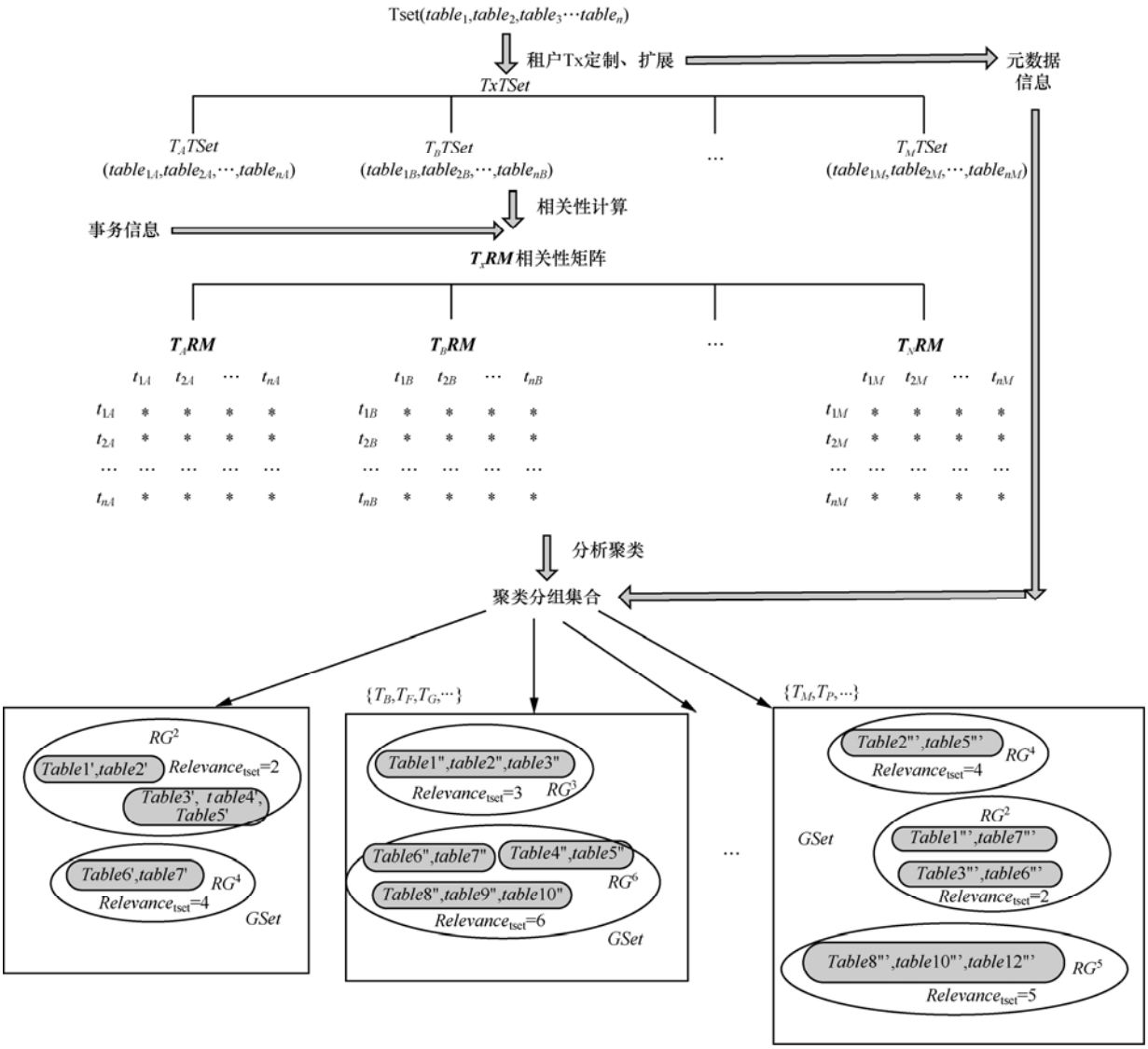


图 7　相关性分组

户进行同样的定制扩展后，在事务运行过程中具有相同的数据特征。*GSet* 中的 $table_1$"、$table_1$"'亦是如此。多租户数据基于相关性矩阵，形成聚类分组的过程如图 7 所示。

参数定义如下。

$|RG^i|$：分组 RG^i 的相关值。$|RG^i| = \sum_{\text{tset}\in RG^i} (Relevance_{\text{tset}} + |table_i| + |table_j| + \cdots + |table_m|)$。其中，$table_i$, $table_j$, $table_m$,…是属于分组 RG^i 中某个 *Tset* 集合的表。$|table_i|$为相关性矩阵对角线数值。

$|RG_{solo}|$：RG_{solo} 分组的相关值。$|RG_{solo}| = \sum_{table_i\in RG_{solo}} table_i$。

$|GSet|$：*GSet* 的相关值。$|GSet| = \sum_{RG^i\in GSet} RG^i + |RG_{solo}|$。

Ω：$table_i$ 加入 *GSet* 之前，整体的相关值$\Omega = |GSet| + |table_i|$。

Ω'：$table_i$ 加入 *GSet*，形成新的聚类分组集合 *GSet'*，整体的相关值$\Omega' = |GSet'|$。

聚类分组过程，以租户 T_A 数据为例，首先将租户 T_A 定制的 $table_{1A}$ 放入分组 RG_{solo} 中。其次，依次对租户 T_A 定制的每个表 $table_{iA}$，计算它与已经形成的相关性分组 RG^i 及 RG_{solo} 中表的相关性值，寻找出与 $table_{iA}$ 相关性值最高的表集合 *S*,相关值为 *d*。然后，检测集合 *S* 中的表是否能与 $table_{iA}$ 一起形成新的相关性分组 RG^d。通过比较Ω与Ω'的大小，若$\Omega < \Omega'$，表明新形成的分组能提高整体的相关性，则 RG^d 加入成功。否则，$table_{iA}$ 加入 RG_{solo} 中。

以 $table_5$ 为例，假设已经形成的 *GSet* 包括 2 个聚类分组，如图 8(a)所示。通过计算 $table_5$ 与其他表的相关值，得知 $table_5$ 与 $table_4$ 的相关值最大。将 $table_5$ 和 $table_4$ 形成一个新的聚类分组后，原 RG^3 分组只剩余一个元素 $table_3$，则还要继续聚类，令 $table_3$ 与其他分组的表进行比较，寻找最大相关值。最终 $table_3$ 聚类到 RG_{solo} 分组中，则 *GSet* 转变为图 8(b)实线所示。此时，两部分的整体的相关值$\Omega < \Omega'$，表明新的聚类的相关值比原聚类相关值要高，所以 $table_4$ 与 $table_5$ 聚为一类。

Step4 对聚类图进行最小分割。

Step3 形成的聚类分组之间存在一定联系。以分组为顶点，分组之间表的相关性为边，表的相关值的和作为权值，建立聚类图。根据图的最小分割算法，对聚类图进行划分，形成数据分区。使得划分后，跨越分区的权值最小，从而保证跨分区的事务代价最少。

Step5 数据迁移。相关参数定义如下。

NiCap: 数据节点 *i* 的容量。

α：数据节点容量的上限百分比。

NiSize：数据节点 *i* 的当前数据量。

TxSize：租户 *Tx* 当前的数据量。

当某数据节点的当前数据量 $NiSize > NiCap\cdot\alpha$, 系统申请新节点进行动态扩展，通过上述划分算法，聚类图经过最小分割后，租户 *Tx* 定制的表集合分为 2 部分，假设 $Txt1=\{table_{1A}, table_{2A}, \cdots, table_{pA}\}$, $Txt2=\{table_{p+1A}, table_{p+2A}, \cdots, table_{nA}\}$。每一部分代表了一个数据分区，分区的大小分别为 $Partition_1Size = \sum_{i=1}^{p} S_i$，$Partiton_2Size = \sum_{i=p+1}^{n} S_i$。

系统控制器通过某种策略确定迁移租户 *Tx* 的最小分区数据到新节点。被迁移数据的大小不能超过节点容量的上限值 $NiCap\cdot\alpha$。迁移后，*group* 与节

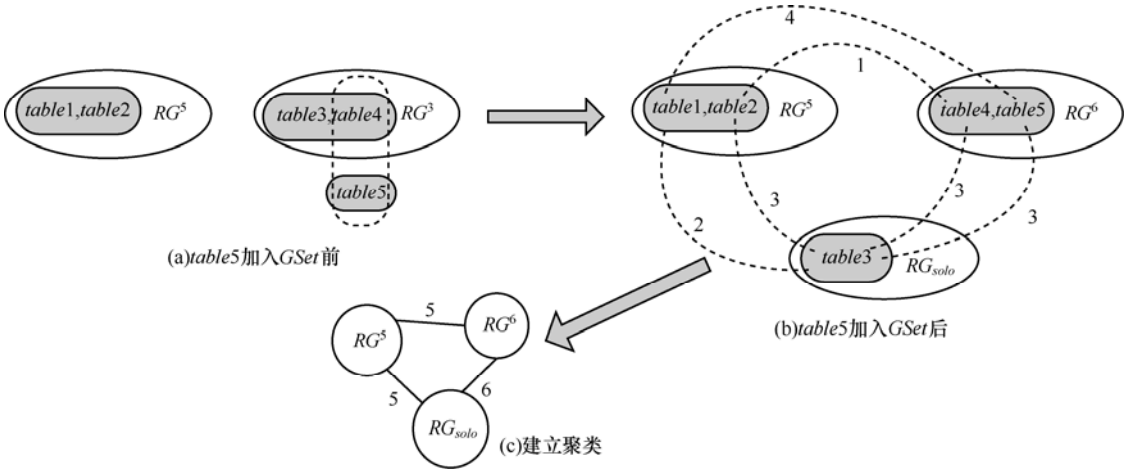


图 8 聚类过程

点的映射信息将加入元数据中。

4.2 划分算法

算法 多租户数据划分算法

Input: 数据节点 N 所有租户定制的表集合：$TxTSet=\{table_{x1},table_{x2},\cdots,table_{xn}\}$（例如：租户 TA 定制的表集合 $T_ATSet=\{table_{1A},table_{2A},\cdots,table_{nA}\}$,租户 TB 定制的表集合 $T_BTSet=\{table_{1B}, table_{2B}, \cdots, table_{nB}\}$,…)事务集合

Output: 相关性分组集合 $GSet$、数据分区

1) $GSet=\Phi$，$RG_{solo}=\{table_{1X}\}$

2) While (数据节点当前容量 $N_iSize < N_iCap\cdot\alpha$)

3) {统计事务访问信息，更新 Tx 的相关性矩阵 $TxRM$;

4) If (事务属于 read) 根据多租户划分模型及元数据信息访问数据；

5) If (事务属于 update) 根据多租户划分模型及元数据信息，访问节点更新数据；

6) 继续接受新事务访问；}

7) $TxRM=Relevance_{ij}=Count(TranS_iSet\cap TranS_jSet)$;

8) For $table_{2X}$ to $table_{nX}$ do //在 $GSet$ 中寻找与 $table_{ix}$ 具有最高相关值的表集合 S

9) $S=\{table_{1X}\}$

10) For each $table_{jX}\in GSet$ or$\in RG_{solo}$ do

11) Calculating $Relevance_{ij}$.

12) If $Relevance_{ij}> Relevance_{ik}$ Where $table_{kX}\in S$ then

13) $S=\Phi$ and add $table_{jX}$ to S.

14) Else if $Relevance_{ij}= Relevance_{ik}$ Where $table_{kX}\in S$ then

15) Add $table_{jX}$ to S.

16) End if

17) End for

18) If $Relevance_{ij}=0$ where $table_{xk}\in S$ then Add $table_{iX}$ to RG_{solo}.

19) ELSE {$Sr=\Phi$. //在 $GSet$ 中寻找与 $table_{iX}$ 具有最高相关值的表集合 S

20) For each $table_{mX}\in S$ do

21) If $table_{mX}\in RG_{solo}$ then add $table_{mX}$ to Sr

22) Else { 形成新的相关性分组 RG^d;

23) For each $table_{pX}\in S$ do

24) { $table_{pX}$ 从原分组 RG^x 删除后，若 RG^x 只剩一个元素 $table_{qX}$, 则 $table_{qX}$ 继续与其他分组中的表比较，重建分组，使分组的相关值最大。若 RG^x 包含两个及两个以上表，则不需进行分组重建。}

25) }

26) Calculating Ω and Ω'

27) If $\Omega<\Omega$' then add $table_{mX}$ to Sr

28) End if

29) End if

30) End for

31) If $Sr=\Phi$ then add $table_{iX}$ to RG_{solo}

32) Else moveing Sr and $table_{iX}$ to a new Group RG^d according to the highest degree of relevance

33) End if

34) End if

35) End for

36) 以 RG^d 为顶点，以 RG^d 表的相关性为边，计算边的权值和，形成图 G

37) 对 G 最小分割，划分为 2 个分区 $Partition_{1X}=\{table_{1X}, table_{2X}, \cdots, table_{pX}\}$,

$Partition_{2X}=\{table_{p+1X}, table_{p+2X},\cdots, table_{nX}\}$

$Partition_{1X}Size=\sum_{i=1}^{p} S_i$

$Partiton_{2X}Size=\sum_{i=p+1}^{n} S_i$

38) 申请新节点 N_j;

39) MINI= 取最小值（$Partition_{1X}Size$, $Partition_{2X}Size$）；$T=Tx$

40) If (MINI< $N_jCap\cdot\alpha$) then {控制器根据迁移策略迁移某租户 Tx 的最小分区的数据元素. 更新元数据}

41) Else {继续对最小数据分区形成的聚类图进行最小分割，直到分区数据量小于 $N_jCap\cdot\alpha$;

42) 控制器根据迁移策略进行数据迁移。}

5 实验

如上文所述，基于共享模式、支持 SaaS 应用的多租户数据划分模型及算法在理论上有较好的性能。在此部分，对多租户数据划分机制进行实验评估。本文提及的多租户数据库环境，由于没有考虑数据副本信息，各个数据库节点并不存在数据交集。同时系统动态扩展过程中，单个租户数据允许跨节点存储，在这种实验环境中，集中评估多租户

数据划分算法带来的分布式事务数量及对系统扩展性能的影响。

实验数据采用了课题组的项目—职业资格服务体系公共基础支撑平台中公共服务应用的数据，主要应用包括机构管理服务、支付管理服务、专家管理服务和考试管理服务。各应用通过 PaaS 平台部署成 SaaS 应用，被诸多租户租赁使用。应用数据采用前文提到的共享模式进行存储。本实验部分将在开源数据库 MySQL 的环境中进行。为了实现通过数据分区技术提升数据库的扩展性，本文利用 Amoeba For MySQL 进行数据切分及整合。本文的实验环境如下。

数据引擎服务器：联想 PC，CPU 为 2×2.26GHz，内存为 8G，硬盘 500G；

客户机：联想 PC 机，CPU 为 2×2.26G Hz，内存为 2G，硬盘 100G；

数据节点：HP 服务器/DELL 服务器，CPU 为 4×2.26G Hz/2×2.26GHz，内存为 4G/3G，硬盘 300G/100G；

初始状态：10 000~50 000 条/租户。

数据划分策略。

循环法：把连续的元组分配到不同的分区。本实验中采用循环法划分，造成同一个租户的数据无序地分布在多个节点的分区中。

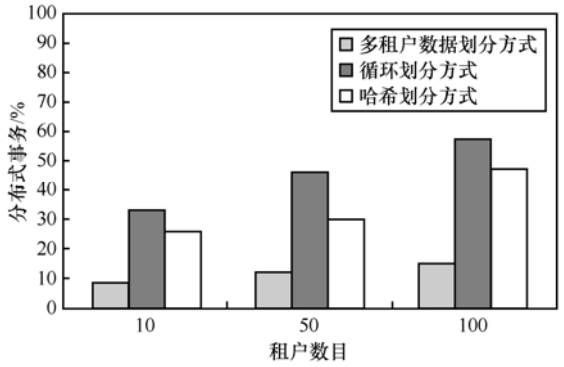
散列法：通过计算元组的散列值将其分配到相应分区。本实验中采用计算 TenantID 列的函数值进行分区。

多租户数据划分算法：通过相关性矩阵进行聚类，将聚类产生的某些相关性分组分配到相应分区。

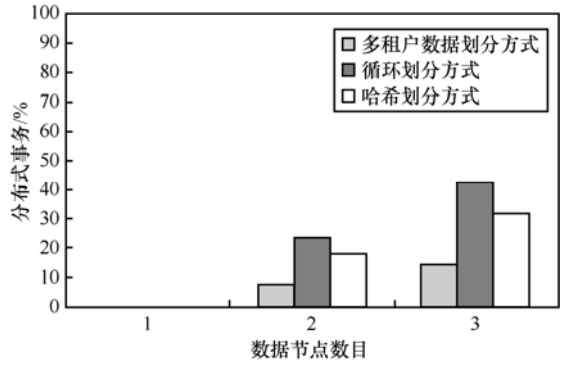
5.1 数据划分对分布式事务的影响

本实验中，集中评估数据划分算法对分布式事务的影响。本文提出的多租户数据划分算法充分考虑 SaaS 应用特点及事务特征，当系统规模扩展为多个节点时，能确保减少分布式事务代价，使分布式事务数目最小。数据划分对分布式事务的影响是通过比较不同的数据划分算法来分析它们的性能。

实验分别检测了租户数目和节点数目发生变化时，3 种数据划分策略所产生的分布式事务的结果。租户数目为 10、50、100 的情况下，初始状态下数据量较小，所有租户数据共享存储于一个节点内，没有分布式事务产生。随着事务访问的变化，租户数据量增加，系统动态扩展形成分布式多节点环境。理论上若节点容量足够大，则可通过一次数据迁移的代价将一个租户的所有数据迁移到新节点，避免后期分布式事务的产生。但在实际的云环境中，租户数据呈几何数量增加，一个租户数据分布于多节点的情况不可避免。为此，在本实验中，多租户数据划分方式采用一次迁移最小数量且相关性较高的分区数据到新节点，来降低后期大量分布式事务代价的产生。几种数据分区算法对分布式事务的影响如图 9 所示。



(a)租户数目改变



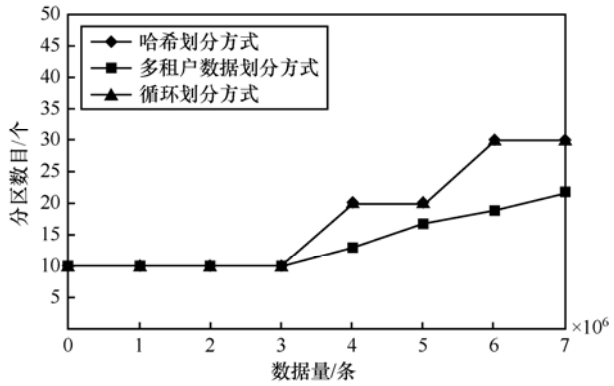
(b)节点数目改变

图 9 3 种数据划分方式产生的分布式事务结果比较

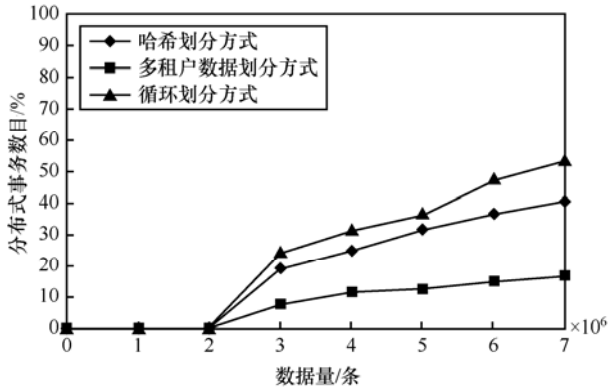
实验结果表明，随着租户数目或数据节点数目的增长，分布式事务的代价会增加。初始状态系统只有一个数据节点时，不会产生分布式事务。租户数目增加，数据量的变化使系统引入新节点后，带来分布式事务代价。相对于循环划分方式和散列划分方式，本文提出的支持 SaaS 应用的多租户数据划分方法，由于充分考虑了事务访问过程中数据相关性特点，能大大降低分布式事务的产生数目，使系统效率得到提高。实验通过执行 1 000~10 000 个不同事务，分析其中产生分布式事务的概率。横向比较，数据划分策略采用多租户数据划分方式时，分布式事务的增长不会太大。

5.2　数据划分对系统扩展性的影响

云环境要求系统具备良好的扩展性。随着系统数据量增加，数据划分方法能产生相应数目的分区。本实验将检测在租户数目固定的情况下（10 个租户），数据量从 1 个节点增加到 3 个分布式节点过程中，3 种不同数据划分方式所产生的分区数目和分布式事务的情况。实验结果如图 10 所示。



(a)数据量对分区数目的影响



(b)数据量对分布式事务影响

图 10　分区数目与分布式事务的比较结果

实验结果表明，系统在一个数据节点时，循环划分方式和散列划分方式都是以预先设定的数值 10 作为分区数目，多租户数据划分方法在初始状态没有分布式的情况下，将每个租户的数据作为一个分区。此时，3 种划分方式都不会带来分布式事务代价。当数据节点的数据量超过上限值（3×10^6 条），通过多租户数据划分方式，逐步迁移最小数量的一组相关性数据到新节点，在此过程中，分区数目逐渐增加，由于相关性强的数据聚集在一起不会带来过量的分布式事务。而循环划分方式和散列划分方式，在形成的分布式环境中，仍然以新设定的某个常数的分区数目进行数据划分。通过实验数据可以看出，3 种数据划分策略在系统规模动态扩展过程中，通过多租户数据划分方法，分区数目能取得较好的线性规模，同时所产生的分布式事务代价是最少的。

6　结束语

本文结合 SaaS 模式下，多租户基于共享模式的数据存储方式，提出了多租户数据划分模型和算法。该算法充分考虑事务访问特征，确保在云中系统动态扩展时，租户数据能够以“Group”为粒度且相关性高的数据聚集在一起，实现多租户数据共享存储，尽最大限度地降低了分布式事务代价。最后，对本文提出的理论进行了实验评估验证。

SaaS 模式的另外一个特点是允许租户在系统运行过程中按需定制，一旦发生定制模式的改变，对于新产生的表及其相应的数据如何划分以及为了获得云环境的高可用性而如何建立数据副本是在接下来的研究中需要深入探讨的问题。

参考文献：

[1] CHONG F, CARRARO G. Multi-tenant data architecture[EB/OL]. http://msdn.microsoft.com/en-us/library/aa479086.aspx, 2006.

[2] AGRAWAL S, NARASAYYA V, YANG B. Integrating vertical and horizontal partitioning into automated physical database design[A]. Proceedings of the ACM SIGMOD International Conference on Management of Data[C]. Paris, France, 2004.359-370.

[3] ZILIO D C. Physical Database Design Decision Algorithms and Concurrent Recorganization for Parallel Database Systems[D]. Canada: University of Toronto,1998.

[4] RAO J, ZHANG C, MEGIDDO N, *et al*. Automating physical database design in parallel database[A]. Proceedings of the ACM SIGMOD International Conference on Management of Data[C]. Madison, Wisconsin, USA, 2002.558-569.

[5] CHANG F, DEAN J, GHEMAWAT S, *et al*. Bigtable: a distributed storage system for structured data[J]. ACM Transactions on Computer Systems, 2008,6(2):4.

[6] BAKER J, BOND C, CORBETT J C, *et al*. Megastore: providing scalable, highly available: storage for interactive services[A]. Proceedings of the 5th Biennial Conference on Innovative Data Systems Research, Conference[C]. Asilomar, CA, United States, 2011.223-234.

[7] DAS S, AGARWAL S, AGRAWAL D, *et al*. ElasTraS: An Elastic, Scalable, and Self Managing Transactional Database for the Cloud[R]. http://www.cs.ucsb.edu/research/ tech_reports/,2010.

[8] CURINO C, JONES E, ZHANG Y, *et al*. Schism: a workload-driven

approach to database replication and partitioning[J]. The VLDB Endowment, 2010, 3(1),48-57.

[9] SQL azure[EB/OL].http://www.microsoft.com/en-us/sqlazure/default.aspx .

[10] CAMPBELL D G, KAKIVAYA G, ELLIS N. Extrem scale with full SQL language support in microsoft SQL azure[A]. Proceedings of the ACM SIGMOD International Conference on Management of Data[C]. Indianapolis, Indiana, USA, 2010.1021-1023.

[11] AGRAWAL D, ABBADI A E, ANTONY S, *et al*. Data management challenges in cloud computing infrastructures[A]. Databases in Networked Information Systems the 6th International Workshop, DNIS 2010[C]. Aizu-Wakamatsu, Japan, 2010.1-10.

[12] AULBACH S, JACOB D, KEMPER S A, *et al*. A comparison of flexible schemas for software as a service[A]. SIGMOD'09[C]. Providence, Rhode Island, USA,2009.881-888.

[13] BURROWS M. The chubby lock service for loosely-coupled distributed systems[EB/OL]. http://labs.google.com/papers/,2010.

[14] DEAN J, GHEMAWAT S. Distributed Programming with Mapreduce[R].2007.

[15] YANG F, SHANMUGASUNDARAM J, YERNENI R. A scalable data platform for a large number of small applications[A]. CIDR 2009 4th Biennal Conference on Innovative Data Systems Research[C]. Asilomar, CA, United States, 2009.129-138.

作者简介：

李晓娜（1980-），女，山东淄博人，山东大学博士生，青岛大学讲师，主要研究方向为云计算、多租户数据管理。

通讯作者**李庆忠**（1965-），男，河北威县人，山东大学教授、博士生导师，主要研究方向为大规模网络数据管理和 Web 数据集成。

孔兰菊（1978-），女，山东荷泽人，山东大学讲师，主要研究方向为数据库和数据管理。

庞成（1988-），男，山东济南人，山东大学硕士生，主要研究方向为数据库和云计算。