SaaS模式多租户数据存贮模型的研究与实现

周文琼[1] 李庆忠[2] 范路桥[1] 郑述招[1]
(广东科学技术职业学院计算机工程技术学院 珠海 519080)[1]
(山东大学计算机科学与技术学院 济南 250000)[2]

摘 要 SaaS模式引入了多租户环境特征,在新的环境下,数据库层存贮设计面临租户数据隔离问题和租户数据弹性扩展问题。主要研究了多租户环境下的数据存储模式,提出了"共享数据库共享Schema存贮数据、独立Schema访问"的多租户数据存贮与访问模型,该模型将SaaS应用的数据存贮和数据访问的Schema进行分离,有效解决了"租户数据隔离性低"的问题;同时,提出了一种以XML为基础的多租户数据扩展模型,该扩展模型很好地解决了"租户数据弹性扩展"的问题。在此基础上,详细描述了这两个模型在SQL Server数据库的实现方案,实例结果证明了所述方法的灵活性和可行性。

关键词 SaaS,多租户,数据层,数据存贮
中图法分类号 TP311 **文献标识码** A

Research and Realization of Data Storage Model for Multi-tenant under SaaS Mode

ZHOU Wen-qiong[1] LI Qing-zhong[2] FAN Lu-qiao[1] ZHENG Shu-zhao[1]
(School of Computer Engineering Technology, Guangdong Institute of Science and Technology, Zhuhai 519080, China)[1]
(College of Computer Science and Technology, Shandong University, Jinan 250000, China)[2]

Abstract SaaS mode introduces the multi-tenant environment, which may cause following two concerns raised for database storage: isolation of tenant's data and expandability of tenant's data. With research of the data storage and access mode for multi-tenant environments, this paper pointed out the model of 'shared schema for data storage and independent schema for data access on shared database'. This model separates the schema for data storage and access for SaaS application, which would resolve the problems of 'isolation of tenant's data'. Meantime, the paper pointed out another XML-based model for multi-tenant data expansion, which would resolve the problem of 'expandability of tenant's data'. Furthermore, the report described the detailed solutions for the two models, which are proven practical and flexible.

Keywords SaaS, Multi-tenant, Data layer, Data storage

作为云计算的一个重要服务模式,SaaS(Software as a Serice,软件即服务)[1,2]是一种通过因特网提供软件服务的模式。SaaS应用需要在多租户架构下,实现SaaS应用系统的高性能、可配置性和伸缩性。

为了满足"多租户共用一个代码实例"的需求,SaaS应用的多租户数据存储方式一般有3种:租户专有数据库模式、共享数据库独立Schema模式、共享数据库共享Schema模式[3,4]。目前大多数SaaS应用的数据存贮模型采用第3种即"共享数据库共享Schema"模式,在该模式下,所有租户的数据以同一个数据库、同一套数据表来存贮。该数据存贮模型虽然具有"可在一台数据库服务器支撑海量租户、共享级别高"等特点,但也存在"数据隔离低、全性差"的问题,要实现租户数据隔离,增强SaaS应用的安全性,需要大量额外工作。

另外,为了实现SaaS应用的数据可配置,使租户数据既有共同的数据结构又有独特的数据模型需求,SaaS应用实现数据定置一般有3种解决方案:定制字段、预分配字段、名称值对。名称值对方案解决了前两个方案中空间资源浪费的问题,给数据扩展提供了非常大的灵活性,但数据处理时十分复杂,需要复杂的处理才能实现客户原数据和扩展数据的映射,检索数据时,需要多次访问元数据才能获取到完整的业务数据,这会大大影响数据访问的性能。

针对以上问题,本文提出了一种"共享数据库共享Schema存贮数据、独立Schema访问"的数据存贮与访问模式,主要贡献如下:

(1)在该模式下,不同租户使用同一个数据库、同一套数据表来存贮数据,但使用独立Schema进行数据访问,这样既可支持海量租户,又可在数据库层实现用户隔离。

(2)在此模型上,设计了动态数据扩展模型,使用XML描述租户的扩展数据,既可在原有数据结构的基础上满足租户对的数据扩展需求,又不会给数据访问性能带来较大的影响。

1 相关工作

目前,国内外学者和研究人员针对SaaS多租户应用的数据存贮做了大量工作,旨在为传统应用向SaaS应用转移提供

到稿日期:2013-01-18 返修日期:2013-03-27 本文受国家自然科学基金(61003253),广东省自然科学基金项目(S2011010001841)资助。
周文琼(1969—),女,硕士,副教授,高级工程师,主要研究方向为信息系统、数据库应用等,E-mail:zwq368@21cn.com;李庆忠(1965—),男,博士,博士生导师,主要研究方向为大规模网络数据管理与Web数据集成。

理论指导和专业经验，本文相关工作大致可以分为以下方面。

为了满足多租户应用需求，Craig D Weissman[1]提出了元数据驱动的多租户数据共享存储结构，租户数据采用元数据表示，通过运行引擎基于元数据产生虚拟应用组件，例如基于租户逻辑视图的数据表、租户记录等。其核心思想是建立一张稀疏数据表，该表有 50 个字段，每个字段类型均为 varchar，用于存储所有租户业务数据。这种多租户存贮结构会出现大量空值，导致存储空间浪费、存取效率较低。

微软的 SQL Azure[5]是以微软 SQL Server 为主，建构在 Windows Azure 云操作系统之上，运行云计算的关系数据库服务(Database as a Service)，它提供网络型的应用程序数据存储的服务。SQL Azure 架构在数据中心可分为 3 个部分：服务提供层 (Service Layer)、平台提供层 (Platform Layer)、基础建设层 (Infrastructure Layer)，它提供了与微软 T-SQL 传统开发相一致的开发工具。其数据模式为“多租户共享数据库、独立模式”，单一数据节点上只支持 150 个独立数据库，不适合 SaaS 应用租户多、每个租户数据量小的特征。

Stefan Aulbach 等[6]论证了过量数据表将引起数据库管理系统资源竞争加剧，导致系统性能严重下降甚至崩溃。为了既满足租户个性化的数据需要，又不能让数据表随租户的增加而激增，提出了 Chunk Folding 算法，该算法把租户的数据表垂直划分为多个块，将它们折叠存储至不同的 Chunk 表。但是，目前该算法仅仅处于理论研究阶段，缺乏一种行之有效的垂直划分算法来获得较满意的效果。

2006 年，Frederick Chong[2]提出了多用户数据体系结构，并针对多租户提出了名称值对的数据存储方式，该方式把租户共享数据放在稀疏表中，把租户个性化数据存放在键值对中，名称值对存储方式可以有效地存贮租户的数据，并很好地支持租户数据模式的扩展。但是，随着租户数目增多，共用数据越来越少，使得稀疏表中只存放少量数据，而键值对存放大量数据，导致系统总体访问性能下降。

R. Agrawal[7]提出了垂直存储的策略。在垂直存储方法中，稀疏数据存储为{对象标示，对象属性，属性值}。垂直存储模式有效地减少了数据表的稀疏度，为了有效地查询，文献提出建立视图将垂直数据表转换为水平视图的方法。但是，由于水平视图中存在大量的左连接，因此该解决方法存在“对某些查询会非常费时”的问题。

亚马逊的 Dynamo[8]是基于 key-value 模式的存储平台，按分布式系统常用的哈希算法切分数据，并将其分放在不同的 node 上。Read 操作时，根据 key 的哈希值寻找对应的 node，可用性和扩展性都很好，存储内容是非结构化数据，存储数据值的原始形式，不识别任何数据结构。

在国内，叶伟等人[9]提出了基于元数据管理的模型驱动结构，从性能、灵活性、可扩展性、实现复杂度和空间利用率等各方面对比预分配字段、定制字段和名称值对的数据存储配置方式。但是，该文献未能提出切实可行的数据存储方案，难以指导 SaaS 应用的实际开发。

目前大多数 SaaS 交付平台尚不能完全支持“用户自定制”，使 SaaS 交付平台的应用领域主要集中在通用软件领域，对多数行业领域应用缺乏支持。在本文中，我们关注一种切实可行的多租户数据存贮模型，一方面，可以使基于 B/S 计算模式的行业应用系统向 SaaS 应用快速转移，另一方面，对 SaaS 应用的数据定制提出了解决方案。

2 提出的解决方案

2.1 “共享数据库共享 Schema 存贮数据、独立 Schema 访问”的多租户数据存贮与访问模型

目前主流大型关系数据库系统都支持 Schema 的概念，Schema 是形成单个命名空间的数据库实体的集合。“共享数据库共享 Schema 存贮数据、独立 Schema 访问”的多租户数据存贮与访问模型如图 1 所示，将 SaaS 应用的数据存贮和数据访问的 Schema 进行分离；使用一个 Schema 创建数据表等数据库对象、保存所有数据；使用各租户 Schema 进行数据访问。在云数据存贮与管理层，首先，按照“共享数据库共享 Schema”数据存贮模式，在一个 Schema 下建立元数据表和业务数据表等数据库对象框架，并在每张业务数据表中增加一个租户编号 tenantID 字段；其次，为每个租户建立独立的 Schema，在租户 Schema 下，建立满足该租户逻辑的业务数据视图，视图名与存贮 Schema 下的业务数据表采用相同的名称；再次，为每个租户建立数据库连接身份 DBUserX，每个 DBUserX 只拥有本租户 Schema；最后，在租户层，当租户访问 SaaS 应用服务时，需要连接身份认证中心，在集中式账户数据库中进行身份鉴定和识别，获取该租户的实际数据库连接身份 DBUserX 和身份密码，将其返回给 SaaS 应用服务器，SaaS 应用服务器再以获得的 DBUserX 的身份连接数据库，进行相关业务处理和分析。

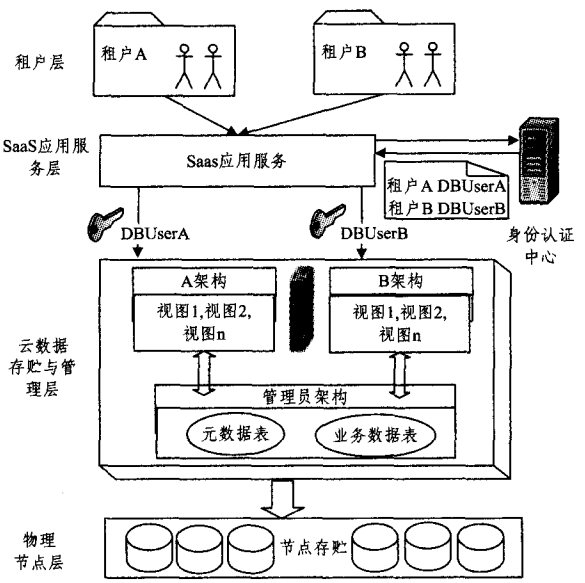

图 1 多租户数据存贮与访问模型

定义 1 Saas 应用的业务数据模型是一个五元组 $N=(V,T,E,O,F)$，其中，V 是租户 Schema 下的逻辑数据对象；T 是存贮 Schema 下共用的业务数据对象；E 是存贮 Schema 下由租户定制、独占使用的数据对象；O 是定制对象和其属性进行的定制行为；F 是映射，$F:\delta(T,E)\rightarrow V$。

在该数据存贮与访问模型下，每个租户只能使用自己的 Schema 下的视图进行数据存取，从而在数据库层实现了租户

数据隔离；由于全部租户的数据实际存贮于同一个 Schema 的数据表中，因此新增租户不会增加数据表的数量；在基于 B/S 计算模式的行业应用向 SaaS 应用转移时，由于租户 Schema 的视图名称与存贮 Schema 的数据表名称相同，因此研发人员可以不修改程序代码。

但是，在该数据存贮与访问模型下，存在租户的实际数据库连接身份 DBUserX 和身份密码可能暴露在网络中的问题，需要对连接身份密码进行加密，可使用不可逆加密算法 MD5 对身份密码进行加密，从而增强系统的安全性；另外，该模式下，新增租户会导致数据库连接数增大的问题，但与独立 Schema 模式下数据表数量增大相比较，这种增大的级别是可以接受的。

2.2 基于 XML 的数据扩展模型

SaaS 应用拥有众多的租户，租户在同一应用下的模式类似，但业务又不尽相同。因此，SaaS 平台需要支持数据存储模型的弹性扩展，即数据定置。在新模式下，将租户异构的数据存入到固定的数据视图中，需要对异构的部分数据进行处理。

定义 2 租户定制的业务数据对象是一个二元组 $E=(M,D)$，其中，M 是租户定制的业务对象元数据；D 是租户定置的业务数据集合。

定义 3 租户定制的业务对象元数据是一个四元组 $M=(X,C,T,R)$，其中，X 是业务数据对象；C 是定置属性名；T 是属性类型，属性类型包括字符串、数字、日期、布尔等和计算类型(计算类型需指明该属性的计算公式)；R 是规则。

定义 4 定制对象和其属性进行的定制行为 O 有 4 种定置操作：①添加一个定制对象；②删除一个定制对象；③修改一个定制对象；④查询定置对象的数据。

从以上定义可知，实现租户定置数据，关键是需要实现(共用业务数据对象 T，租户定置数据对象 E)与租户逻辑数据对象 V 的映射关系 F，F 用关系代数表示如下：

$F:\sigma_{tenantID}=$'租户编号'$(T \triangleright\lhd E)$

在传统的"名称值对"定置数据解决方案中，由于 T 与 E 的关系是 $1:n$，要实现 T 与 E 的等值连接，需要做大量工作。

XML(eXtensible Markup Language，可扩展标记语言)文档具有数据语义的自解释性和支持异构数据结构，通过将 XML 与关系型数据库的有机结合，既可满足租户对 SaaS 数据的扩展需求，又不改变原有的关系数据视图结构。使用 XML 数据类型可以在应用程序和关系型数据模型之间提供一个隔离层，该层允许数据以非关系的模型进行操作。

在存贮 Schema 下，利用 XML，使 T 与 E 的关系改变为 $1:1$，这样可以将两个数据对象合并为一个数据对象，因此，基于 XML 的数据扩展模型可以在原来的关系数据表中增加一个 XML 字段，如图 2 所示，该 XML 字段用来处理各租户间异构的数据。由于基于 XML 类型字段的存在，扩展字段的数量没有限制，扩展字段的数据类型也没有限制。因此，基于 XML 的数据扩展模型，可以灵活方便地应对不同租户的数据扩展需求；同时，主流关系数据库支持对 XML 字段创建索引，可以极大地提高扩展数据的查询性能。

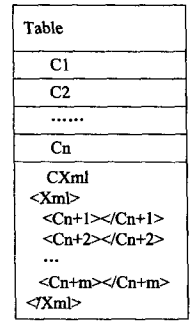

图 2 在数据表添加 XML 字段

在租户访问 Schema 下，为了存贮数据结构到租户数据的映射 F，需要创建租户业务视图，实现租户数据隔离和数据定置。

3 实例研究

下面实例以 SQL Server2010 为数据库平台。

3.1 "共享数据库共享 Schema 存贮数据、独立 Schema 访问"模式的实现

在基于 SaaS 模式的 CRM 系统中，客户 Customer、联系人 Contactor 是"客户"模块的主要业务对象，例如，新增租户 101 时，按照"创建 Schema→在新增架构下，创建客户模块相关视图→创建数据库用户，使用新增架构"的数据库层处理步骤，便可有效构建 SaaS 应用。

```
--创建架构
USE CrmDB
GO
CREATE SCHEMA sche101
--创建"客户"模块有关业务视图
create view sche101.Customer
as
select *,租户扩展数据
from dbo.Customer
where tenantID = 101
with check option
go
create view sche101.Contactor
as
select *,租户扩展数据
from dbo.Contactor
where tenantID = 101
with check option
go
--……创建其他视图
--创建登录、数据库用户，使用 MD5 生成身份密码
declare @pwd varchar(128)
set @pwd = hashbytes('MD5','123456')
exec('CREATE LOGIN user101 WITH PASSWORD='''+ @pwd
+'''')
USE test
GO
CREATE USER user101 WITH DEFAULT_SCHEMA= sche101
```

执行上述 SQL 代码后，以 user101 登录数据库，则只能查询租户 101 的数据，当基于视图做数据修改、删除、插入时，

由于使用 with check option 子句，确保了数据只能为租户 101 的数据。

3.2 基于 XML 的数据扩展模型的实现

当租户有数据配置需求时，例如，租户 101 针对 Contactor 数据定义时有两项扩展数据即“Email”和“职位 Job”，租户 102 针对 Contactor 数据定义时有一项扩展数据即“生日 Birthday”。按照“创建 XML Schema→类型化数据表的扩展 XML 字段→创建主 XML 索引→扩展字段的数据处理”的步骤进行。

①创建统一的扩展字段 XML Schema，定义扩展字段格式为〈extendCol colName="" colType=""〉值〈/〉

```
CREATE XML SCHEMA COLLECTION dbo. ExtendColSchema AS
N'〈xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www. w3. org/2001/XMLSchema"〉
〈xs:element name="extendCol"〉
〈xs:complexType〉
〈xs:simpleContent〉
〈xs:extension base="xs:string"〉
  〈xs:attribute name="colName" type="xs:string" /〉
  〈xs:attribute name="colType" type="xs:string" /〉
〈/xs:extension〉
〈/xs:simpleContent〉
〈/xs:complexType〉
〈/xs:element〉
〈/xs:schema〉'
GO
```

②类型化数据表的扩展 XML 字段

```
alter table Contactor alter column extendCol XML(ExtendColSchema)
```

③创建主 XML 索引

```
CREATE PRIMARY XML INDEX idx_extendCol ON Contactor (extendCol)
```

④创建包括定置数据的租户视图

```
create view sch101. Contactor
as
SELECT *
,extendCol. query('/extendCol[@colName="email"]'). value('/extendCol)[1]','varchar(100)') As email
,extendCol. query('/extendCol[@colName="job"]').
value('/extendCol)[1]','varchar(100)') As job
From dbo. Contactor
WHERE tenantID = 101
with check option
```

⑤扩展字段的数据处理

a)查询扩展字段的内容；

```
——以 user101 登录，查找该租户拥有的编号为 1 的联系人
SELECT CID,…,email,job
from Contactor
WHERE CID = 1
```

b)插入包括扩展字段的 XML 数据，例如，新增租户 101 的联系人 1 的 Email 是"joan. sina. com"、Job 是"副主任"；

```
INSERT INTO Contactor(tenantID,CID,…,extendCol)
VALUES(101,1,…,
'〈extendCol colName="email" colType="string"〉joan@sina. com〈/extendCol〉
〈extendCol colName="job" colType="string"〉
副主任〈/extendCol〉')
```

c)更新 XML 字段的内容，例如，将上面新增数据行的 EMAIL 修改为"joan168. sina. com"；

```
UPDATE Contactor
SET extendCol. modify(
'replace value of (/extendCol[@colName="email"])[1] with
"joan168@sina. com"')
WHERE Cid=1
```

d)删除 XML 字段的数据，则将其更新为空。

从以上实例可以看出，基于 XML 的数据扩展模型灵活性高，所占存储空间较少，数据查询简单快捷，由于租户视图中扩展字段来源于 XML 加工，数据处理需要稍微复杂的处理，能够满足 SaaS 应用的大规模数据定制需求。

结束语 本文提出一种 SaaS 平台多数据存贮模型“共享数据库共享 Schema 存贮数据、独立 Schema 访问”，该模型能有效支持数据层统一管理和隔离访问，为 SaaS 服务提供商屏蔽多租户数据管理技术细节；同时，提出基于 XML 的数据扩展模型，很好地支持了租户定制数据，为 SaaS 服务提供商提供了数据节点伸缩技术[10]。本文通过实例说明了该多租户数据存贮模型的有效性及高扩展性，表明本模型能高效支持 SaaS 应用开发及租户个性化定制。

参 考 文 献

[1] Weissman C D, Bobrowski S. The design of the force. com multitenant internet application development platform[C]// Proceedings of the ACM SIGMOD International Conference on Management of Data, 2009. Providence, Rhode Island, USA: Ugur Cetintemel, 2009: 889-896

[2] Frederick C, Gianpaolo C. Architecture strategies for catching the long tail[OL]. http://msdn2. microsoft. com/zh-n/architecture/ aa479069. aspx, 2006

[3] 李晓娜，李庆忠，孔兰菊，等. 基于共享模式的 SaaS 多租户数据划分机制研究[J]. 通信学报，2012，09：110-119

[4] 姚金成，张世栋，史玉良，等. 基于 Chunk Folding 的多租户数据库缓存管理机制[J]. 计算机学报，2011，12：2319-2330

[5] Microsoft Inc. Microsoft SQL Azure [EB/OL]. http://www. microsoft. com/cz- e/azure/sqlazure, 2012

[6] Aulbach S, Grust T, Jacobs D, et al. Multi-tenant data- bases for Software as a Service: Schema-mapping Techniques[C]// Proc. of ACM Conference on Management of Data. Vancouver, Canada: [s. n.], 2008: 1195-1206

[7] Agrawal R, Somani A, Xu Yi-rong. Storage and Querying of E-Commerce Data[C]// VLDB. 2001: 149-158

[8] DeCandia G, Hastorun D, Jampani M, et al. Dynamo: Amazon's Highly Available Key-value Store[C]// SOSP. 2007

[9] 叶伟. 互联网时代的软件革命-SaaS 架构设计[M]. 北京：电子工业出版社，2008

[10] 朱志良，苑海涛，宋杰，等. SOA 与云计算：竞争还是融合[J]. 计算机科学，2011，12：6-11

[11] 陶新民，郝思媛，张冬雪，等. 不均衡数据分类算法的综述[J]. 重庆邮电大学学报：自然科学版，2013，25(1)：101-110