基于 SaaS 模式公共服务平台多用户数据结构的研究

The multi－tenant data architecture study based on the SaaS model for the public service platform

昌中作　徐 悦　戴 钢　（北京交通大学计算机学院网络管理中心　100044）

摘　要： 本文在研究 SaaS 模式特点的基础上，首先提出了 SaaS 应用的体系架构，然后提出了基于 SaaS 模式的企业公共服务平台的三个紧密相连的模型——多用户数据模型、元数据管理模型和安全服务模型。

关键词： SaaS　多用户数据模型　元数据　数据扩展　安全服务。

1　引言

SaaS（Software as a service 软件即服务）是一种通过 Internet 提供的新的软件使用模式，它消除了企业购买、构建与维护基础设施和应用程序的需要，并通过登录互联网的账号来使用软件。软件的开发、运营和维护由服务提供商负责管理，服务提供商以租赁的形式向用户提供软件的在线使用。它能使用户在任何地方，只要能接入 Internet，就能方便的使用软件来管理企业。

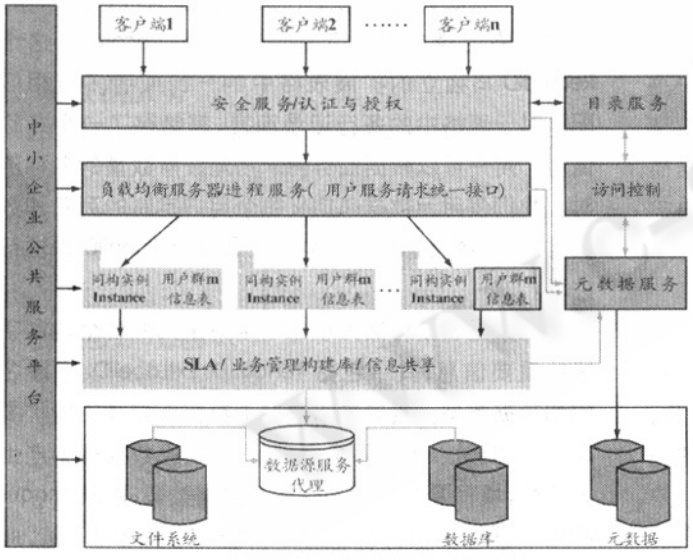

图 1

2　SaaS 应用体系结构

首先，我们大致的介绍一下整个 SaaS 应用的体系结构。SaaS 的特点（数据非本地）和性质（单一代码库，多应用实例）决定了元数据管理服务和安全服务既是客户的最大关注点，又是平台架构考虑的重点。普通的 B/S 平台架构不能确保客户数据的安全性，也不能满足企业业务多样化的要求。为此我们采取如下系统体系结构，如图 1。

在此架构中，软件服务供应商在负载均衡的服务器上为不同的客户提供主机服务，在同一代码库上运行多个相同的应用实例（同构实例），每个应用实例服务于一定数量不同需求的客户，通过授权和安全策略来确保不同的客户访问各自权限范围内的数据，以及区分不同客户的数据。为了屏蔽服务层对底层多个数据库的数据访问操作，我们采用面向对象

的"数据源代理"机制,这样能大大的提高系统的性能。另外,系统采用可配置的元数据为不同的用户提供个性化的服务。负载均衡的服务器群作为和用户交互的统一接口,并且向下管理这些同构实例。同构实例能够最大化不同用户间的资源共享,并且从最终用户的角度来看,不会察觉到应用是与多个用户共享的。用户使用软件的性能由服务水平协议 SLA 来监管。

企业用户登录之后,网络操作系统 NOS 根据负载均衡的原则从实例群中分配一个同构实例为其服务,并且把用户的目录服务结构和元数据配置信息写入该实例的用户群信息表。同构实例和用户群信息表绑定在一起,用于数据安全认证和资源授权。这种绑定能够避免"受信任的数据连接"的资源权限验证瓶颈的产生。关于受信任的数据连接将在 5.3 节中详细论述。关于认证机制的详细讲解,请见 5.1 节。

为了满足 SaaS 应用的需求——多用户、多应用实例、服务可定制性、数据安全性以及信息共享性等特点,本文将从数据的角度——多用户数据结构、元数据服务以及安全服务等三个方面,给出 SaaS 应用在数据访问层上的一些设计策略,希望能对整个 SaaS 公共服务平台的设计与实现起到一些作用。

3 多用户数据模型

在 SaaS 应用中,在满足资源的共享性与多用户高效性的同时,必须区分属于不同客户的数据。系统采用"共享数据库,独立架构"方法来实现多用户数据模型,并且采用"名称值对"模式来实现数据模型的扩展,以满足用户数据的动态增长。

3.1 共享数据库,独立架构

该数据模型的示意图如图 2 所示。

一定数量的不同的用户使用同一个的数据库,每个用户都拥有自己的表集,形成用户各自专门的架构,如上图中的 Tenant BJTU 与 Tenant NIAT。当客户最初成为此类服务的用户时,配置子系统会自动为该用户创建离散的表集,并将其与用户自己的架构相关联。客户可用 SQL CREATE SCHEMA 命令来创建方案,并授权能够存取该方案的用户账号。离散表集创建后,用户可根据需要添加或者修改列,甚至是表格。

共享数据库,独立架构方法能够保证用户数据的高安全性,确保数据的逻辑隔离性,同时使得每个数据库资源得到了最为充分的利用。

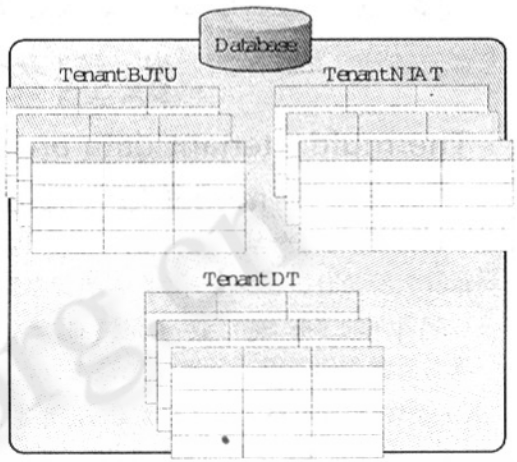

图 2

3.2 名称值对

由于不同的企业会有着各自独特的业务需求,而固定的没有延伸性的默认数据模型是无法满足企业业务的动态变化和增长的。为此,我们采用"名称值对"[1]模式来实现系统的数据模型的扩展,如图 3 所示。

客户在独立的表格中存储定制数据,包含定制数据的表格记录中包含了一个唯一的 RecordID,这个 RecordID 与独立的扩展表格中的一行或者多行相匹配。对于表格中的各行记录而言,都储存了一个名称值对。用户可以根据业务需求创建任意数量的名称值对。我们使用元数据来定义每个用户定制的列名和数据类型,元数据表与扩展数据表通过 ExtLabelID 来进行关联。

元数据表格存储关于每个用户定义的各个定制字段的重要信息,其中包括 TenantID、ExtLabelID、字段名称和数据类型。当最终用户对主数据表记录进行扩展时,应用要做以下四件事情。第一,在原数据表中产生一新行,填写要扩展列的列名和数据类型,以及 TenantID 值。第二,应用为记录创建唯一的 RecordID 值,此 RecordID 用来关联主数据表记录与延伸表。然后记录本身在主数据表中要保存此 RecordID 值。第三,在延伸表中创建一个包含以下信息的新的行[1]:

- 与主数据表记录相关联的 RecordID;
- 与定制字段定义相关联的 ExtLabelID;

· 将要定制字段的值转换成字符串。

第四，应用还要将 ExtLabelID 值和 RecordID 值填入元数据新行中。

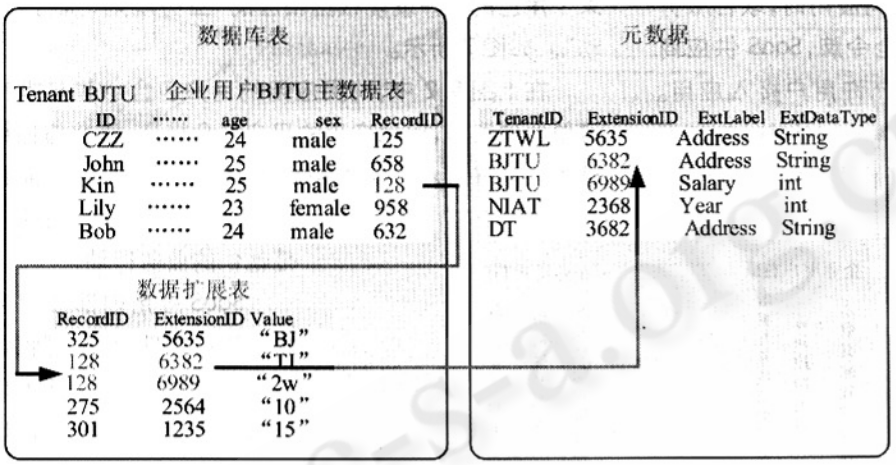

图 3

上述方案使每个用户都能够根据需要创建尽可能多的定制字段，以满足业务需求。当应用检索客户记录时，会在延伸表中进行查找，选择与 RecordID 相对应的所有行，并为所用的每个定制字段返回一个值。

4 元数据服务

为了满足企业业务需求的变化以及业务逻辑行为的差异性，SaaS 应用应该具有服务可定制性。这可以通过元数据服务来实现的。元数据服务为客户提供了服务定制和配置功能，以满足客户特定的需求。元数据服务可以进行以下四个领域的配置更改，形成四个层级的配置域，如图 4。

(1) 用户界面和风格

用户可以自定义自己页面的风格，如改变图形、色彩、字体等相关内容。

(2) 工作流程和业务规则

不同企业的工作流程和业务规则会有所差别的。客户可以根据实际需求情况自行配置应用的工作流程，以满足自己的业务需求。

(3) 数据模型的扩展

数据库数据模型的设计应该具有较好的扩展性，我们采用"共享数据库，名称值对扩展"策略，以满足客户业务的动态变化和增长。

(4) 存取控制

企业用户 TID(Tenant ID)负责创建多个最终用户的账户 ID，并建立相应的访问控制列表(ACL)来确定每个用户能够存取使用的资源和功能。

(5) 树形域定制

为了使客户能够灵活的根据需求进行软件配置，将上述选项组织成层级的配置域，形成一棵树状配置管理域，如图 5 所示。

每个配置域包含不同的选项，以反应上述四个域的配置信息。每个客户都拥有顶级配置域，使他们能够根据业务需求进行

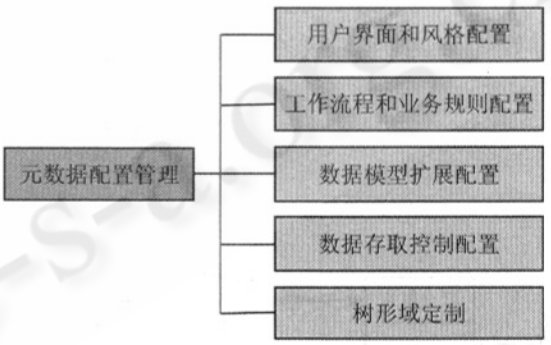

图 4

变更配置，并能在顶级配置域下构建任意层级的一个或者多个配置域。

5 安全服务

5.1 认证

认证主要是用于实现用户登录时的安全验证，本系统采用非集中认证模式[2]。其认证原理如图 6 所示。

在第二章中我们提到，SaaS 供应商在创建一个新

的应用之后，就授权客户管理员负责创建、管理和删除本地账户。在非集中认证系统中，为了使得 SaaS 服务器能够识别这些账户，我们采取的策略是：客户采用与其自己的本地目录服务证书相连的联合服务（Federation Service）。当最终用户尝试访问应用时，联合服务将用户进行本地认证，并发布安全令牌，SaaS 供应商的认证系统将接受安全令牌，并允许用户接入应用。我们采用目前发展成熟的 SSL（Secure Sockets Layer 安全套接层协议层）技术来实现这种非集中认证模式。

图 5

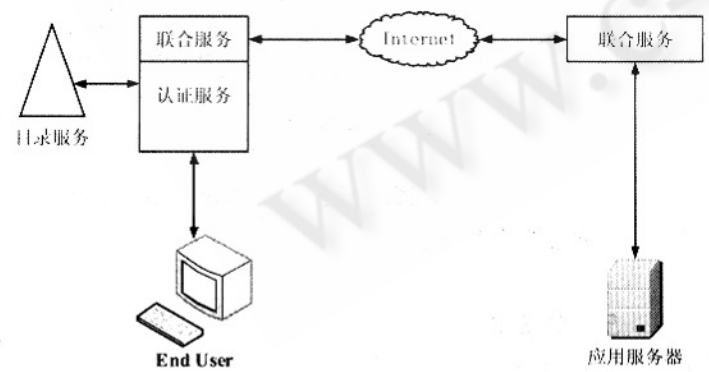

图 6

5.2 授权

授权主要是用于企业最终用户在系统资源的分配和商务功能的权限方面的管理，以确保内部业务流程的合理分工和协作。在本系统中的资源和商务功能都使用“角色”来进行管理。通过“角色”来实现数据资源的存取控制。角色与企业中的特定岗位功能进行映射。每个角色都被赋予一项或者更多“许可”，分配到某个角色的用户就能根据相应的“业务规则”执行“行动”。如图 7 所示。

在上图 5.2 中，购买应用的用户包含创建与提交订单两项许可，审批应用的用户包含提交、批准与拒绝订单三项许可。这两类用户的角色是不一样的，而且通过不同的业务规则来规范他们的业务行动。

SaaS 应用内部负责对角色进行管理，角色可包含单个用户的账户以及用户群组。不同用户账户和用户群组根据需要被分配到不同的角色。根据用户所分配到的角色，该用户可获得一项或者多项许可，以执行特定的操作活活动。这些活动通常直接与重要的商务功能或应用管理本身映射。

5.3 受信任的数据连接

受信任的数据连接[1]主要是，用于确保不同企业用户间的数据安全与企业内部不同用户群间的数据安全，以防止用户非法访问他人的数据或者未授权的数据。在 SaaS 应用中，有两类不同的用户，即企业用户 TID（Tenant ID）和最终用户 ID。我们要区分这两类不同的用户。企业用户 TID 通过应用访问自己数据存储区的数据，其数据存储区在逻辑上通常与其他企业用户的数据相隔离。每个企业用户 TID 为一个或多个最终用户 ID 的应用存取授权，使最终用户 ID 能够在企业用户 TID 控制下，存取企业用户 TID 数据的某一部分。

目前，有两种方法经常用来确保安全存取数据库中存储的数据，一是模拟[1]，二是受信任的子系统账户[1]。我们采取综合上述两种方法的方案来确保数据的安全存取。该方案能够充分发挥数据库服务器

本机安全机制的作用，确保用户数据的逻辑隔离最大化。其原理如图 8 所示。

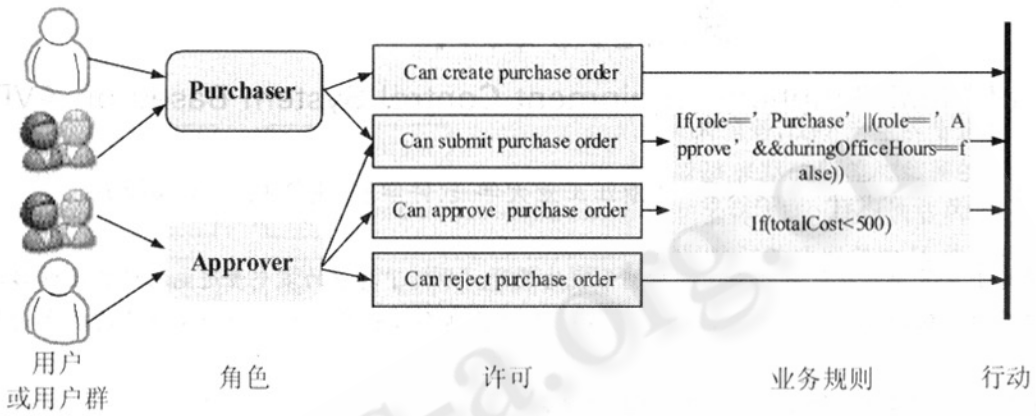

图 7

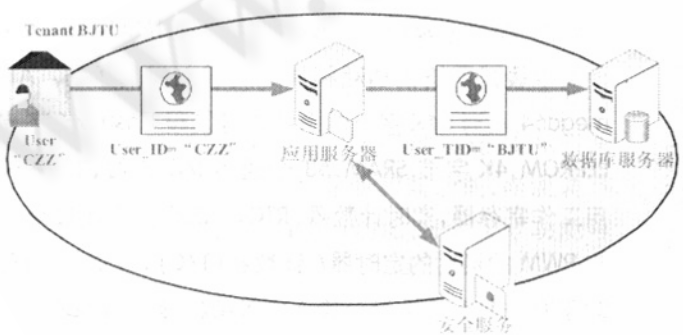

图 8

在该方案中，需要为每个企业用户 TID 建立数据库存取账户，并用 ACL 为每个企业用户账号授权，以使其能够访问被允许使用的数据库对象。如果最终用户执行的操作直接或间接要求调用数据库，那么应用会采用与企业用户账号 TID 相关联的凭证、而不是与最终用户 ID 关联的凭证。数据库服务器不区别区分来自相同企业用户 TID 的最终用户请求，而是直接允许这些最终用户访问该用户的数据。

6 结论

本文提出了基于 SaaS 模式企业公共服务平台的多用户数据体系架构，并从系统数据的角度，提出了多用户数据模型、元数据管理模型和安全服务模型。这三个模型是实现 SaaS 应用的核心关键技术，也是实现的难点。当然，大型数据中心模型的建设，以及应用和数据在服务器群上扩展也是 SaaS 应用实施的难点，在本文中未给出相应的探讨。最后，我们给出一点建议，为了更好的管理企业业务和适应业务的增长变化，可以在 SaaS 应用中融入 BPM 管理思想，使得整个体系架构变的更加健壮。

参考文献

1 Frederick Chong, Gianpaolo Carraro, Roger Wolter. Multi - Tenant Data Architecture. Microsoft Corporation. June 2006.

2 Frederick Chong , Gianpaolo Carraro. Architecture Strategies for Catching the Long Tail. Microsoft Corporation. April 2006.

3 刘启原著，数据库与信息系统的安全，出版社：科学出版社，2000 - 01 - 01.

4 张敏、徐震、冯登国著，数据库安全 - - - 信息安全国家重点实验室信息安全丛书 ，科学出版社，2005 - 07 - 01.