A NOVEL MULTI-TENANT ARCHITECTURE DESIGN FOR SOFTWARE AS A SERVICE APPLICATIONS

Haitham Yaish, Madhu Goyal and George Feuerlicht[1,2]
[1]*Faculty of Engineering and Information Technology, University of Technology, Sydney, P.O. Box 123, NSW 2007, Broadway, Australia*
[2]*Faculty of Information Technology, University of Economics, Prague, Czech Republic*

Keywords: Software as a Service, Multi-tenancy, Database, User Interface, Access Control, Configurability, Metadata, SaaS Architecture.

Abstract: Software as a Service (SaaS) is a web based delivery model which permits a third party provider offering software services to unlimited number of tenants based on multi-tenant architecture design. Multi-tenancy is the primary characteristic of SaaS, it allows SaaS vendors to run a single instance application which supports multiple tenants on the same hardware and software infrastructure. This application should be highly configurable to meet tenants' expectations and business requirements. Nevertheless, configuring multi-tenant application is hard and complex task to accomplish. This paper is generally focusing on SaaS multi-tenancy in the context of providing a configurable architecture design to support multi-tenant applications in designing and developing a configurable multi-tenant database schema, User Interface, and access control. This novel configurable multi-tenant architecture design provides critical and fundamental solution to the development of multi-tenant SaaS applications, once it is achieved it will be a vital contribution to SaaS industry.

1 INTRODUCTION

Multi-tenancy is the primary characteristic of SaaS, it allows SaaS vendors to run a single instance application which supports multiple tenants on the same hardware and software infrastructure. It can be applied in four software layers: application, middleware, virtual machine, and operating system (Kwok et al. 2008). The application layer of SaaS has a maturity model consists of four levels: (1) Ad Hoc/Custom, (2) Configurable, (3) Configurable and Multi-Tenant-Efficient, and (4) Scalable, Configurable, and Multi-Tenant-Efficient (Frederick and Carraro 2006; Hudli et al. 2009; Kwok et al. 2008). This paper will focus on the Configurable and Multi-Tenant-Efficient Level. This maturity level requires a multi-tenant aware design with single code base and metadata service, which allows sharing of resources across tenants and configuring how the application appears and behaves, with the ability of isolating and differentiating data, information, configurations, and settings which belong to different tenants. Multi-tenant aware application allows each tenant to design different parts of application, and automatically adjust its behaviour during runtime without redeploy the application (Chong 2006). Consequently, configuring multi-tenant aware application is tenant self-service that typically performs while the application in operation, to minimize the system downtime, and allows tenant to feel as if he/she is the only one using the application (Mietzner et al. 2009).

Nevertheless, it is not always the case with SaaS vendors to have a proper configuration capability in multi-tenant SaaS applications. Therefore, it might be ad-hoc and manual configuration practices (Kuo et al. 2007) Moreover, developing SaaS multi-tenant application is a complex process which requires extra development work to satisfy each tenant requirements and personalized needs (Yaish et al. 2011; Ying et al. 2010). Configuration is the most challenging issue in multi-tenant architecture, because it introduces significant complexities in SaaS development, and involves longer development lifecycle including implementation, testing, and deployment (Chang et al. 2007; Wei et al. 2010; Wei et al. 2008). Moreover, the current studies of SaaS configurations are not sufficient in allowing SaaS

Yaish H., Goyal M. and Feuerlicht G..
A NOVEL MULTI-TENANT ARCHITECTURE DESIGN FOR SOFTWARE AS A SERVICE APPLICATIONS.
DOI: 10.5220/0003915800820088
In *Proceedings of the 2nd International Conference on Cloud Computing and Services Science* (CLOSER-2012), pages 82-88
ISBN: 978-989-8565-05-1

vendors to create a multi-tenant application which facilitates configuring SaaS tenants' applications (Hongbo et al. 2009; Kuo et al. 2007).

This paper proposes a novel multi-tenant SaaS architecture design that in turn will provide a programming model to simplify and speed up the development of multi-tenant SaaS applications. This architecture aims to provide: (1) a configurable multi-tenant database schema design to enable SaaS tenants create their own elastic database schema to satisfy their business needs, (2) a method to solve integrating shared tenant tables and virtual tenant tables problem by making them work together to act as one database, (3) a query optimizer method to optimize the execution of virtual multi-tenant database queries, (4) a configurable multi-tenant user interface method for SaaS tenants to configure their user interfaces according to their business needs, (5) a user interface rendering method to facilitates designing and developing run-time execution solution that generates dynamic user interface screens for particular tenant business domain, and (6) an authentication and access control method to authenticat the users of multi-tenant application, as well as isolate and diffrentiate accessing their user interfaces and databases.

The paper is structured as follows: Section 2 explores SaaS Multi-tenancy. Section 3 describes our overview architecture. Section 4 describes our conceptual architecture design. Section 5 is a conclusion and discussion of future work.

2 MULTI-TENANCY

Multi-tenancy is the fundamental design approach that improves the manageability of SaaS application (Jansen et al. 2010), and allows its tenant to have his user-experience as he/she is working on his own system (Bezemer et al. 2010). It is an important feature of SaaS application that can be used by multiple tenants at the same time on a single instance of software code (Menken & Blokdijk 2009; Samuel & Lopes 2009; Shao 2011).

2.1 SaaS Maturity Model

Multi-tenancy can be applied in four software layers: application, middleware, virtual machine, and operating system (Kwok et al. 2008). The application layer has four levels of SaaS maturity model. Level 1 - Ad Hoc/Custom: each tenant in this level has a separate custom instance of SaaS application which is hosted on the vendor servers, the source code of this instance can be customized according to a tenant needs. Level 2 - Configurable: in this level the vendor hosts separate instances for each tenant, each of these instances using the same source of code. However, each instance may be configured differently to meet tenant's needs. Level 3 – Configurable and Multi-Tenant-Efficient: this level allows the vendor to runs a single instance to support multiple tenants, this single instance can be configured differently by each single tenant and each configuration will be designated for a tenant who created it .Level 4 – Scalable, Configurable, and Multi-Tenant-Efficient: in this level, the vendor hosts multiple tenants with a high level of scalability (Frederick and Carraro 2006; Hudli et al. 2009; Kwok et al. 2008).

2.2 Multi-tenancy Architecture

Shao (2011) SaaS tenants usually share the same software and in some cases the same database. Accordingly, once multi-tenant vendors need to increase multi-tenant architecture they should look after Quality of Service (QoS) for one tenant from being affected from the rest of the tenants. Shao's study describes multi-tenant architecture in four aspects including: Resource isolation, Configuration, Security, and Scalability. First aspect, resource isolation is significant for multi-tenant application, because tenants sharing the same infrastructure and software code. Second aspect, ensuring multi-tenant application is highly configurable. Third aspect, security is an issue due to sharing software code and data between tenants. Last aspect, as discussed in the SaaS Maturity Model in section 2.1, in order to make level 1 and level 2 a scalable levels, a significant software design and implementation need to be done. This is not the case for Level 3, since it allows all SaaS tenants to use the same single instance. Nevertheless, if this level doesn't have an infrastructure to dynamically create multiple copies to provide multi-tenant service, then its scalability will be limited. In our proposed multitenant architecture design we are focusing on Level 3 and covering the following aspects: resource isolation, configuration, and security.

2.3 Multi-tenant Configuration

Configuration in multi-tenant applications allows SaaS vendors to run a single instance to support multiple tenants with configurable metadata, which provides means of configuration for multi-tenant application, to satisfy their business needs, and to

resolve the problem of different requirements for several tenants who may use a particular business domain application. This maturity level requires a multi-tenant aware design with single code base and metadata service, which allows sharing resources across tenants, and configuring how multi-tenant application appears and behaves with the ability of isolating and differentiating data, information, configurations, and settings which belong to different tenants. Multi-tenant aware application allows each tenant to design different parts of application, and automatically adjust its behaviour during run-time execution without redeploy the application (Chong 2006).

3 OVERVIEW ARCHITECTURE

The overview architecture of the system proposed in Figure 1 depicts the main five artifacts including: web user interface, user interface services, database services, access control services, and database. The database comprises of three types of tables: Common Tenant Tables (CTT), Elastic Extension Tables (EET), and Virtual Extension Tables (VET).

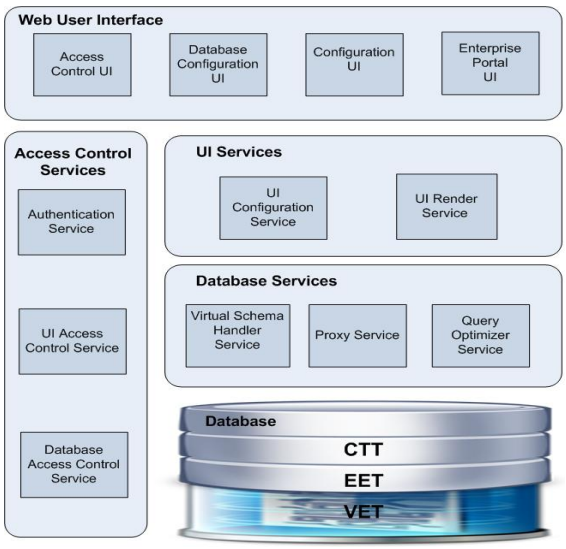

Figure 1: Overview Architecture.

This architecture designed based on the three-tier architecture design. The web user interface is the presentation tier, the services are the application tier, and the database is the data tier.

4 CONCEPTUAL ARCHITECTURE DESIGN

In this section we are proposing a conceptual architecture design which permits SaaS vendors to implement a single instance application which supports multiple tenants on the same hardware and software infrastructure. This single instance application has flexibility to configure three multi-tenant application aspects: database model, user interface, and access control. This architecture design can be used to implement any business domain in multi-tenant application like: Customer Relationship Management (CRM), Supply Chain Management (SCM), Human Resources (HR), or any other business domain. Figure 2 depicts the essential elements of the conceptual architecture design at a very high level.

4.1 Configuring Multi-tenant Database

In our conceptual architecture design we propose EET technique, and three database services that we will discuss in the following subsections.

4.1.1 Database Schema Design

Based on reviewing and understanding existing multi-tenant database schema designs, and based on our proposed novel multi-tenant database schema mapping technique that we published in a previous conference paper, we are proposing our multi-tenant architecture design to enable SaaS tenants create their own elastic database schema during multi-tenant application run-time execution to satisfy their business needs. Our schema mapping technique comprises of three types of tables Elastic Extension Tables (EET), Common Tenant Tables (CTT), and Virtual Extension Tables (VET) (Yaish et al. 2011).

- **Elastic Extension Tables (EET):**
  These tables propose a new way of designing and creating an elastic tenant database which consists of two types of tables, first type is Common Tenant Tables (CTT), and second type is Virtual Extension Tables (VET) which can be used to satisfy each specific tenant need. For example, let us assume that “tenant” table is one of CTT which can be used and shared by all tenants to store their information, and VET can be an extension to the “tenant” table that may include more columns which are not already included in the “tenant” common table (Yaish et al. 2011).

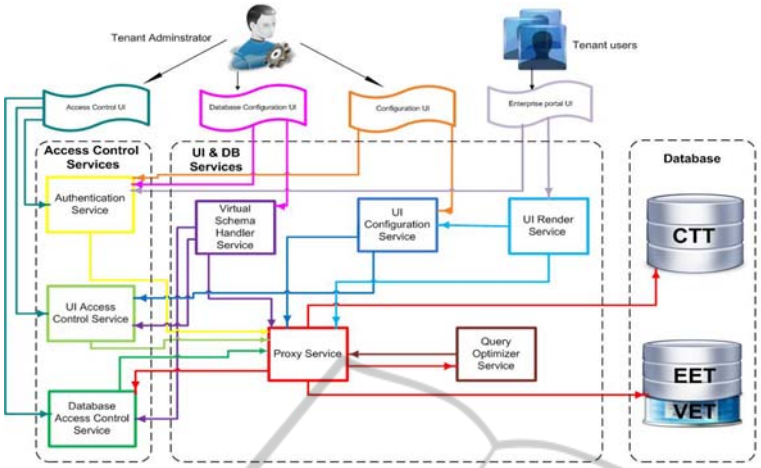

Figure 2: Conceptual Architecture Design.

- **Common Tenant Tables (CTT):**
  These tables are shared between tenants who are using a same single instance multi-tenant application. These are physical tables which can construct a base schema of a particular business domain application. For example, a multi-tenant application for sales business domain may have a sales schema tables like: sales person, customer, product, sales fact, and any other sales tables. These tables have columns which are used by most of tenants, therefore these tables considered as CTT (Yaish et al. 2011). Also, CTT contain two sets of tables: (1) user interface tables to store different tenant's user interface settings, preferences, field's validations, and business rules. (2) Access control tables to store different tenant's access control settings, preferences, for the data tier, and the presentation tier.

- **Virtual Extension Tables (VET):**
  These are virtual tables that can be created, updated, or deleted by storing records in, or removing records from, the EET to extend tenants business domain CTT, or create non-business domain related tables which satisfy tenants business requirements.

The aim of these virtual tables is to be used like any other physical tables by having virtual primary keys, virtual relationships with VET or CTT, and virtual indexes (Yaish et al. 2011).

4.1.2 Database Services

This step describes the database service designs that will be used to allow tenants configuring and managing their database. These services include: Proxy Service, Query Optimizer Service, and Virtual Schema Handler Service.

- **Proxy Service:**
  After designing the novel EET multi-tenant database mapping technique that enables tenants to create their own elastic database schemas or VET, we are proposing this service to generate and execute tenants' queries by using codebase, CTT, and VET. This service has two objectives: (1) enables multi-tenant application to retrieve records from VET, or retrieves combined records from two or more tables of CTT and/or VET, by using fields' values which are common between these tables. (2) Enables the rest of proposed services to invoke this service and access database tables (CTT, EET, and VET) through it. This service consumes different services and database tables including: Query Optimizer Service, Database Access Control Service, CTT, EET, and VET.

Also, it is exposed to be used by services including: Query Optimizer Service, Virtual Schema Handler Service, User Interfaces Configuration Service, User Interface Render Service, Authentication Service, User Interfaces Access Control Service, and Database Access Control Service.

- **Query Optimizer Service:**
  A query optimizer service proposed in the architecture design to optimize performance, speed up query retrievals, and use the most efficient way to execute a query by using virtual primary keys, virtual relationships between CTT and/or VET, virtual indexes, efficient executing plans, efficient logic, and join algorithms. This service consumes Proxy Service to access the

functionalities that it provides, and access through these functionalities CTT, EET and VET.

- **Virtual Schema Handler Service**:
  This service configures tenant database tables, by updating CTT, and on the other hand by creating, updating, or deleting VET. It is a service that exposed to be used by Database Configuration User Interface screens, these screens are used for configuring tenant database CTT and VET. This service consumes other services including: (1) User Interface Access Control Service, to use user interfaces access control settings and preferences to manage accessing Database Configuration User Interface screens, (2) Database Access Control Service, to use database access control settings and preferences to manage accessing Database Configuration User Interface screens, and (3) Proxy Service, to retrieve and modify tables structures and fields details from one or more tables of CTT and/or VET.

4.2 Configuring Multi-tenant User Interface

Since there are complexities in developing an easy to use multi-tenant user interface, a configurable user interface proposed in our architecture design to create an elastic user interface that can be associated with CTT, EET, and VET. This user interface can be configured by tenants' administrator and rendered for tenants' users dynamically and instantly without writing any code during runtime execution. This user interface design comprises of two user interface services: User Interface Configuration Service, and User Interface Render Service.

- **User Interface Configuration Service**:
  Several configuration user interface screens are consuming this service to configure multi-tenant application user interface by configuring its business domain pages, menus, and fields. This service is exposed to be used by Configuration User Interface screens, and User Interface Render Service. It is also consumes two other services including: (1) User Interface Access Control Service, to use user interface access control settings and preferences for Configuration User Interface screens, and (2) Proxy Service, to retrieve tables structures and fields details from one or more tables of CTT and/or VET to use them in configuring user interface screens and fields.
- **User Interface Render Service**:
  This run-time execution service generates Enterprise Portal User Interface screens by using codebase and configuration data, these screens will be rendered for a specific business domain application which can be dynamically created by tenants in order to be used for their businesses. This service consumes other services including: (1) User Interface Configuration Service, to use settings and preferences of configured user interface screens, and (2) Proxy Service, to render combined records from two or more tables of CTT and/or VET into Enterprise Portal User Interface screens.

4.3 Configuring Multi-tenant Access Control

This section describes the Access Control design that can be implemented by vendors and used by tenants to set up accounts for their users who interact with multi-tenant application. These accounts can be authenticated by using authentication functionality. Typically, multi-tenant applications accessing resources and business functions by using roles that map to specific functions or system features. Each role can have one or more permission which may be assigned to one or more tenant's users to enable them perform actions according to tenant's business rules. The access control design comprises of three services: Authentication Service, User Interface Access Control Service, and Database Access Control Service.

- **Authentication Service**:
  This service is exposed to be used by user interface screens to authenticate tenants' administrators and tenants' users when they login to multi-tenant application. These user interfaces screens include: Enterprise Portal User Interface, Configuration User Interface, Access Control User Interface, and Database Configuration User Interface. This service only consumes Proxy Service to access through it the authentication CTT.
- **User Interface Access Control Service**:
  This service is exposed to be used by Access Control User Interface screens to allows tenant's administrator to create roles and permissions and map them to tenant's users. Also, it is exposed to be used by two other services including: User Interface Customization Service, and Virtual Schema

Handler Service. This service only consumes Proxy Service to access through it the user interface access control CTT.

- **Database Access Control Service**:
  This service is exposed to be used by Access Control User Interface screens, these screens are used to create roles and permissions and map them to tenant's users. Also, there are two services consume this service including: Proxy Service, and Virtual Schema Handler Service. This service only consumes Proxy Service to access through it the database access control CTT.

4.4 User Interfaces

This section is divided into four sub sections: Configuration User Interface, Enterprise Portal User Interface, Database Configuration User Interface, and Access Control User Interface.

- **Configuration User Interface**:
  This is an administration user interface which allows a tenant administrator to configure a multi-tenant application user interface, by configuring a business domain pages, menus, and fields. Then, this user interface will be rendered as an enterprise portal user interface. This user interface consumes two services, the first one is Authentication Service, to authenticate tenant administrator when he/she tries to login to this user interface. The second one is User Interface Configuration Service that discussed in section 4.2.

- **Enterprise Portal User Interface**:
  This User Interface is a run-time execution interface which is flexible and dynamic, and can be rendered as per tenant configuration discussed in section 4.2. This user interface consumes two services, the first one is Authentication Service, to authenticate tenant administrator when he/she tries to login to this user interface. The second one is User Interface Render Service that discussed in section 4.2.

- **Database Configuration User Interface:**
  This administration user interface allows a tenant administrator to configure tenant database tables, by updating CTT, and on the other hand by creating, updating, or deleting VET. This user interface consumes two services, the first one is Authentication Service, to authenticate tenant administrator when he/she tries to login to this user interface. The second one is Virtual Schema Handler Service that discussed in section 4.1.2.

- **Access Control User Interface**:
  This administration user interface allows a tenant administrator to configure tenant's users and other administrators roles and permissions which enable them accessing their database and user interfaces. This user interface consumes three services, the first one is Authentication Service, to authenticate tenant administrator when he/she tries to login to this user interface. The second one is User Interface Access Control Service that discussed in section 4.3. The last one is Database Access Control Service that discussed in section 4.3.

5 CONCLUSIONS AND FUTURE WORK

Designing and developing a configurable multi-tenant application is hard, complex, and it needs extra work and time to be developed. The current studies of SaaS configurations are not sufficient in allowing SaaS vendors to create a multi-tenant application, which facilitates for their tenants configuring their application. Based on Level 3 of SaaS Maturity Model, and based on EET and it's customizable database design technique, we introduced in this paper a novel multi-tenant SaaS architecture design that in turn will provide a programming model to simplify and speed up the development of multi-tenant SaaS application, which can deal with challenges from both technical and business perspectives by solving the issues of configuring SaaS multi-tenant application. Our proposed multi-tenant architecture design permits SaaS vendors to implement a single instance application which supports multiple tenants on the same hardware and software infrastructure. This single instance application, have flexibility to configure three multi-tenant application aspects: database model, user interface, and access control. This architecture design will be beneficial for SaaS vendors to implement any business domain in multi-tenant application, in a short time and cost-effective manner.

Our future work will focus on elaborating, developing, and evaluating services and user interfaces that proposed in our multi-tenant architecture design in greater details.

REFERENCES

Bezemer, C. P., Zaidman, A., Platzbeecker, B., Hurkmans, T. & t' Hart, A. 2010, 'Enabling multi-tenancy: An industrial experience report', *Software Maintenance (ICSM), 2010 IEEE International Conference on*, Timisoara, Romania, pp. 1-8.

Chang, J. G., Wei, S., Ying, H., Zhi, H. W., Bo, G. 2007, 'A Framework for Native Multi-Tenancy Application Development and Management', *E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services, 2007. CEC/EEE 2007. The 9th IEEE International Conference on*, Beijing, China, pp. 551-8.

Chong, F. 2006, Multi-tenancy and Virtualization, Msdn, viewed 10 February 2012, http://blogs.msdn.com/b/fred_chong/archive/2006/10/23/multi-tenancy-and-virtualization.aspx.

Frederick, C., Carraro, G. 2006, 'Architecture Strategies for Catching the Long Tail', Msdn, viewed 10 February 2012,http://msdn.microsoft.com/en-us/library/aa479069.aspx.

Hongbo, L., Yuliang, S., Qingzhong, L. 2009,' A Multi-granularity Customization Relationship Model for SaaS', *Web Information Systems and Mining, 2009. WISM 2009. International Conference on*, Jinan, China, pp. 611-5.

Hudli, A. V., Shivaradhya, B., Hudli, R. V. 2009, 'Level-4 SaaS Applications for Healthcare Industry', Proceedings of the 2nd Bangalore Annual Compute, Bangalore, India, p. 4.

Kuo, Z., Xin, Z., Wei, S., Haiqi, L., Ying, H., Liangzhao, Z., Xuanzhe, L. 2007, 'A Policy-Driven Approach for Software-as-Services Customization', *E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services, 2007. CEC/EEE 2007. The 9th IEEE International Conference on*, Tokyo, Japan, pp. 123-8.

Kwok, T., Thao, N., Linh, L. 2008, 'A Software as a Service with Multi-tenancy Support for an Electronic Contract Management Application. Services Computing', *2008. SCC '08. IEEE International Conference on*, Hawaii, USA, pp. 179-8.

Menken, I., Blokdijk, G. 2009, Saas and Web Applications Specialist Level Complete Certification Kit- Software as a Service Study Guide Book and Online Course, 1st edn, Emereo Pty Ltd, Australia.

Samuel, J., Lopes, N. 2009, 'SaaS (software as a service) – models and infra-structures', thesis, Universidade Tecnológica de Lisboa, Lisboa, Portugal.

Mietzner, R., Metzger, A., Leymann, F., Pohl, K. 2009, 'Variability modeling to support customization and deployment of multi-tenant-aware Software as a Service applications', *Principles of Engineering Service Oriented Systems, 2009. PESOS 2009*, Vancouver, Canada, pp. 18-8.

Shao, Q. 2011, Towards Effective and Intelligent Multi-tenancy SaaS, thesis, Arizona state university, Arizona, USA.

Wei, C., Beijun, S., Zhengwei, Q. 2010, 'Template-based business logic customization for SaaS applications', *Progress in Informatics and Computing (PIC), 2010 IEEE International Conference on*, Shanghai, China, pp. 584-5.

Wei, S., Xin, Z., Chang, J. G., Pei, S., Hui, S. 2008,' Software as a Service: Configuration and Customization Perspectives', *Congress on Services Part II, 2008. SERVICES-2. IEEE*, Shanghai, China, pp. 18-7.

Yaish, H., Goyal, M., Feuerlicht, G. 2011, 'An Elastic Multi-tenant Database Schema for Software as a Service', *Cloud and Green Computing CGC 2011*, Sydney, Australia, pp. 737-7.

Ying, L., Bin, Z., Guoqi, L., Deshuai, W., Yan, G. 2010, 'Personalized Modeling for SaaS Based on Extended WSCL', *2010 the 2nd International Conference on Advanced Computer Control (ICACC 2010)*, Liaoning, China, pp. 298 -5.

Jansen, S., Houben, G., Brinkkemper, S. 2010, 'Customization Realization in Multi-tenant Web Applications: Case Studies from the Library Sector', *10th International Conference on Web Engineering*, Vienna, Austria, pp. 445-5.