A Contrivance to Encapsulate Virtual Scaffold with Comments and Notes

Nagarajan Balasubramanaian[1], Suguna Jayapal[2], and Satheeshkumar Janakiraman[3]
[1]Department of Computer Applications, Arunai Engineering College, India
[2]Department of Computer Science, Vellalar College for Women, India
[3]Department of Computer Science, Bharathiar University, India

Abstract: *CLOUD is an elision of Common Location-independent Online Utility available on-Demand and is based on Service Oriented Architecture (SOA). Today a chunk of researchers were working towards contrivance based on multi-tenant aware Software as a Service (SaaS) application development and still a precise pragmatic solution remains a challenge among the researchers. The first step towards resolving solution is to enhance the virtual scaffold and propose it as a System under Test (SuT). The entire work is proposed as a Model View Controller (MVC) where the tenant login through the View and write their snippet code for encapsulation. The proposed VirScaff schema acts as Controller and provides authentication and authorization by role/session assignment for tenant and thus helps to access data from the dashboard (Viz., Create, Read, Update and Delete (CRUD)). The SuT supports and accommodates both SQL and Not only Structured Query Language (NoSQL) dataset. Finally, this paper construed that SuT behaves well for both SQL and NoSQL dataset in terms of time and space complexities. To sum-up, the entire work addresses the challenges towards multitenant aware SaaS application development and highly commendable while using NoSQL dataset.*

Keywords: *Virtual scaffold, Multi-Tenant common gateway, pattern, model view controller, role-based access control, JavaScript object notation, not only structured query language, software as a service.*

Received July 14, 2017; accepted July 28, 2019
https://doi.org/10.34028/iajit/17/3/7

1. Introduction

Common Location-independent Online Utility available on-Demand (CLOUD) architecture is location independent on-line utility which is subscription based model. Multi-tenancy defines the single instance of software which is running on the service provider's infrastructure [1, 20]. Virtual scaffold provides a virtual platform, in which, the tenant utility available from the remote location(s) [1, 11]. The scaffold provides seamless interoperability through a model, view and controller. The model provides data and business logic. A view provides user interface. A controller acts as handler and issues business logic.

A pattern language provides re-usable component, and these components instruct the controller to handle the request between application and database. This contribution uses Model View Controller (MVC) pattern, which helps in tweaking the multi-tenant application development [3, 22].

Access Control is generally a policy or procedure that allows, denies or restrict access to a system [5, 7]. Various identity based access control model includes Mandatory Access Control (MAC), Discretionary Access Control (DAC) and Role-Based Access Control (RBAC) [6, 7].

National Institute of Standards and Technology (NIST) has laid standards for cloud characteristics, virtualization and RBAC [5]. Each tenant is isolated (remote) and seclude (segregated with private space for execution). Tenant can be from any corner of the globe [11].

A role can be thought of as the set of transactions that a tenant or group of the tenant can perform within the context of an organisation [7].

This paper addresses the issues related to a tenant logging into the system, access data from the dashboard (Viz., Create, Read, Update and Delete (CRUD)) and store the resultant solution in virtual container using SQL and NoSQL datasets. Enhanced virtual scaffold is proposed as SuT with Multi-Tenant Component Gateway (MTCG) pattern as pattern language [2, 19]. The VirScaff schema congregates the layers in the virtual scaffold and provided encapsulation between data and application. This contribution discusses a gamut and implementation of VirScaff schema in a pragmatic perspective.

The rest of this paper is organised as follows section 2, discussed on study on related literatures. Section 3 detailed about rudiments for contrivance. Section 4 elaborated on contrivance for encapsulation. Section 5 discussed about experiments conducted on virtual scaffold. The discussion at the end of section 5.4, provided comments on results obtained with a scope of future implementation. Section 6 concluded remarks on solution to multi-tenant SaaS application development.

2. Study on Related Literatures

Onset study for this orchestration begins with literature related to host multi-tenant SaaS applications.

Marino *et al*. [14] have proposed a middleware, by name, MIDAS, to provide interoperability between Software as a Service (SaaS) and Data as a Service (DaaS). They have claimed that, through MIDAS, an application will be able to get data for its operation through DaaS and return the expected result. Hui *et al*. [12] has proposed architecture by name M-Store. They argued that multi-tenant data management was a form of SaaS, whereby, third party service provider host DaaS and claimed scalability as the key feature for their architecture. Solomon *et al*. [18] has presented Zeros Framework. They claimed that, their Zeros Framework combines with fine-grained access control to allow existing application to migrate to cloud environment with very minimal software changes. Literature study on [12, 14, 18], provided sufficient information on existing architecture and its implementation on cloud multi-tenant data management.

Jacobs and Aulbach [13] have implemented multi-tenant SQL Data model using Shared Table and Shared Instance (STSI) and suggest that it was essential for hosted services to manage high traffic volumes at low cost. They suggested those multi-tenant data models were useful to handle multiple branches that have the same schema. Hammes *et al.* [9] has examined the design, execution and subsequent performance between traditional RDBMS and NoSQL data model. The authors implemented them in cloud server using MongoDB document data model and results obtained were useful. Tudorica and Bucur [21] have listed and compared various NoSQL systems with multiple comparisons. Both authors have concluded their results by implementing Yahoo! Cloud Serving Benchmark (YCSB) dataset. Okman *et al*. [17] has discussed security issues in NoSQL. Literatures [9, 13, 21] gave clear idea on implementation on SQL and NoSQL data model.

Ferraiolo *et al.* [6] have proposed a NIST model for role-based access control towards an unified standard whereas Ferraiolo and Richard [7] have studied various access control model and proposed rules required for implementing role-based access control. Tang *et al*. [20] have proposed multi-tenancy authorization models for collaborative cloud services. They have remarked that most cloud service providers isolate user activities and data within a single tenant boundary with no or minimum cross tenant interaction. With this remarks, they have proposed a model for Authorization as a Services (AaaS). For Multi-Tenant Role-based Access Control (MT-RBAC) model family which aimed to provide fine-grained authorization in collaborative cloud environments by building trusted relations among tenants.

Yu *et al*. [23] enunciated, many new challenges when the user access confidential data in the untrusted environment. They have identified and addressed open challenges such as fine-graininess, scalability, and data confidentiality of access control in their literature by exploiting and uniquely combining technique of attribute-based encryption, proxy re-encryption and lazy re-encryption and claimed that they have developed the provably secure system under existing security models. Mehar *et al*. [15] has presented a modified fine-grained data access control algorithm for file storage cloud. They have presented the algorithm for fine-grained data access.

The study on above existing literatures, helped to understand that only limited authors have implemented the system with an architecture which supported huge dataset and thus help to develop multi-tenant SaaS applications. The problem identified were - time taken for execution of query and space requirement for storage of data. Limited authors have discussed solution for query execution.

2.1. Problem Statement

The proposed system congregates (or assembles) various layers into an enhanced virtual scaffold and implement a System under Test (SuT) with a re-usable component called MTCG pattern which receives RBAC and stores results using SQL and NoSQL document data model(s).

3. Architecture of Enhanced Virtual Scaffold

3.1. System under Test (SuT)

The virtual scaffold is enhanced and proposed as SuT in Figure 1. It is a composite assemblage of SaaS, Infrastructure as a Service (IaaS) and Platform as a Service (PaaS). The proposed SuT helped to specify, how application and database can be inter-operable. The pattern formulation is included into a scaffold to Prune the data available and provide RBAC to the seclude tenant entering into the system [3, 5].

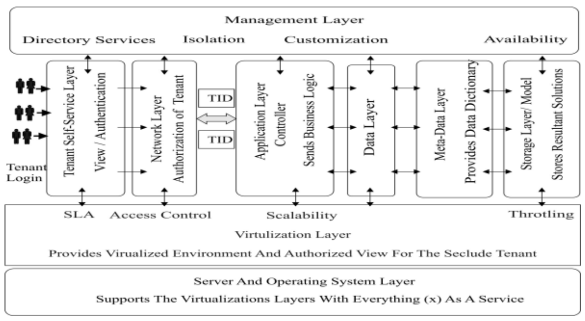

Figure 1. Enhanced Virtual Scaffold (SuT) for SaaS Application Development.

3.2. Pattern Formalisation

The Multi-Tenant Component Gateway (MTCG) Pattern [2, 21] is a single re-usable component which used several patterns [2, 3]. Thus, MTCG helped in tweaking multi-tenant SaaS application development. Model View Controller (MVC) pattern helped in separation of Model from View components and makes it possible to implement several user interfaces that reuse the common core business logic. Duplication of low-level model code is totally eliminated across multiple User Interface (UI) implementations. Hence, decoupling of model and view code results in an improved ability to write a unit test for the core business logic code. Modularity of components allowed core logic developers and UI developers to work simultaneously without affecting the other. The ability of unit test can be improved using implementing core reusable component such as authorization service pattern, which helped in identification of tenant [5]. Federated identity pattern provides the external identity for a tenant by assigning Tenant ID (TID) [5]. Gatekeeper pattern protects applications and services by using a dedicated host instance and prune pseudonymous and anonymous persons using this pattern [3, 5]. Valet key pattern will restrict direct access to a client for a specific resource or service [5]. Command and Query Responsibility Segregation (CQRS) Pattern can be applied in a scaffold to perform CRUD operation [21]. This pattern segregates the operation that read data (Queries) from the operation that update data (command) by using separate interfaces [1, 21]. The controller handles user request by executing appropriate business logic for the request send by the tenant from the dashboard.

4. Encapsulating SuT

4.1. Main Idea

In tenant self-Service layer, unique login authentication for the new tenant is done using an authentication pattern [3, 5]. The existing tenant logging into the scaffold with tenant ID and password issued by the authentication process.

In network layer, using federated identity pattern, a tenant is authorized and provided role identification. The session identification is provided to the tenant and Virtual Table Authorization (VTA) is done.

In the data layer, once role selection is done successfully, tenant operation can be done as Data Provider (DP), Data Consumer (DC) or create/delete a table.

The business layer acts as controller in the server side, which does tenant operation (such as, execution of CRUD operation on tenant's model (or table) available in server side) using CQRS pattern. The resultant table generated by the tenant after Create Read Update and Delete (CRUD) operations are stored in virtual storage container based on Event Sourcing Pattern [18].

Table 1. Rule for Tenant Access Right (TAR) Code.

Rule 1	**Rule 2**	**Rule 3**
The Active Role (AR) for the tenant (subject) can be defined as follows AR(S: subject) = {the active role for subject S} Each tenant (subject S) may be authorized to perform one or more roles: Role Authorization (RA)(s: subject) = {authorized role of subject s} Each role may be authorized by RA to perform one or more transaction. Transaction Authorization (TA) (r: role) = {Transaction authorized for role r} Tenant (Subject S) may execute the transaction. The predicate exec(s, t) will be true if subjects can execute transactions t at the current time, otherwise, it will be false. exec (s: subject, t: trans) = true iff subject 's'can execute transaction t.	In Role Assignment (RA), a tenant (subject) can execute a transaction only if the subject has selected or been assigned a role. Based on definition 1, the equation 1 will be as follows: **∀s: subject, t: trans,(exec(s,t)⇒ AR(s) ≠ ɷ → 1** The identification and authentication process is not considered as a transaction. All other tenant activities on the system are conducted through transactions. Thus all active tenants are required to have a same active role. In RA, a subject's active role must be authorised for the subject: **∀s: subject, (AR(s) ⊆ RA(s)) → 2** With equation (1), this rule ensures that users can take only the roles for which they are authorised**.** In Transaction authorization (TA), a tenant (subject) can execute a transaction only if the transaction is authorised for the subject's active role. **∀s: subject, t: trans, (exec(s,t) ⇒t Є TA(AR (S)) → 3** With equation 1 and 2, this rule ensures that users can execute only transactions for which they are authorised. With reference to equation 1, 2 and 3, to enforce control to access resources (objects) the following equation (4) can be used **∀s: subject, t: trans, o: object, (exec(s, t) access (AR(s), r, t, o, x)) → 4** The equation 4 could be defined using a transaction access (r, t, o, x) which indicates - a subject (tenant) in role r to access object (resources) o in mode x using transaction t, where x is taken from some set of modes such as DP or DC.	USERS, ROLES, OPS, OBS (users, roles, operations, and objects, respectively) UA ⊆ USER X ROLES, a many to many mapping users to role assignment relation. Assigned users: (r: ROLES) → 2^{USERS}, the mapping of role r onto a set of user. Formally: assigned _user(r) = {u Є USERS | (u, r) Є UA}. PRMS = $2^{(\text{OPS-X OBS})}$, the set of permissions PA ⊆ PRMS X ROLES, a many to many mapping permission –to-role assignment relation. Assigned – permission (r: ROLES) → 2PRMS, the mapping of role r onto a set of permission. Formally : assigned –permissions (r) = { p Є PRMS | (p,r) Є PA} Ob (p: PRMS) → {OP ⊆ OPS}, the permission-to-operation mapping, which gives the set of objects associated with permission p. Ob (p: PRMS) → {OP ⊆ OBS}, the permission-to-operation mapping, which gives the set of objects associated with permission p. SESSIONS, the set of sessions. User sessions (u: USERS) → 2^{SESSIONS}, the mapping of user u onto a set of sessions. Session roles (s: SESSIONS) → 2^{ROLES}, the mapping of user u onto a set of roles. Formally session roles ($s_{i)}$⊆{r ROLES} (session users (si), r)Є UA) USER represents the subject (tenant), ROLES represents the role assigned to the tenant according to their option. OPS represent the operation to be performed by the tenant and OBS represents the object (resource), generally available through ISVs in the data layer. PRMS represents the permission granted during the user session.

4.2. Mathematical Foundation for RBAC

The user request for the pattern language is based on the tenant (subjects) and resources (objects). In the cloud environment, the Independent Software Vendors (ISVs) will provide resources (objects) to the tenant. ISVs can share data using STSI data model. According to tenant's choice, role will be assigned to seclude tenant before tenant login into the scaffold. NIST has formulated definitions for RBAC [6, 8]. Table 1 provides the basic rules for TAR code. To sum up, rule 1 provides AR, RA and TA. The rule 2 gives the rule to implement the RBAC. The rule 3 provides ways and mathematical definition for implementing RBAC [2, 3].

4.3. Schema Representation

For clarity, the schema is presented in two levels, namely system level and algorithm level. At the system level, high-level operations were explained which will be implemented as the algorithms in the next level.

4.3.1. System Level Schema

The high-level operations involved in this schema include login/assign a role for the tenant, a creation of the table and performing CRUD operations, torage of the file in the virtual container and log-out of tenant from the scaffold. To sum-up, the operations involved include new tenant authentication, session allocation and role management, CRUD operations, storage of files in the virtual container and tenant revocation.

4.3.2. Algorithm Level Operations

The tenant login into this enhanced virtual scaffold is from any corner of the globe. The Figure 2 outlines main idea and flow chart for writing contrivance to encapsulate the virtual scaffold. The steps for algorithm level operations were summarized as follows

The tenant self-service layer helps the existing/ new tenant to login into the system using proper authentication (View) (Algorithm 1.1 and Algorithm 1.2).

The Network layer authorize the tenant and provide a seclude User Interface (or View) to the seclude tenant with unique Tenant ID (TID) (Algorithm 2). The tenant is assigned with unique Job role by network layer and then tenant invocates the resource R by writing suitable snippet code in the business layer (Algorithm 2).

The business layer (Controller) encapsulates with data layer and search for the information in the metadata layer (Algorithm 3).

The Model fetches the data invocated by the tenant, if available and a discrete view of the data is available in the tenant's View (Algorithm 3).

The master database and resultant solutions either in JSON document format or XML format is stored in the storage layer (Model) (Algorithm 4).

The tenant logout from the scaffold once they complete their process (Algorithm 5).

4.3.3. Definition and Notation

The role based seclude tenant invoke for the resource R through their dashboard and controller receives the request and encapsulate the tenant with data layer to fetch the data invocated. To restrict curious snooper into the scaffold, Tenant Access Right (TAR) code (Table 2) is generated as Boolean values based on rule 2 and 3 in Table 1. Notation table (Table 3) for VirScaff algorithm include a Request (Req) sends the request to the CLOUD and invokes the action on the service.

Table 2. Tenant Access Right (TAR) code.

TAR Code	**TAR Generation Code**		
	COMMIT		**ROLLBACK**
	DP	**DC**	
1	T	T	F
2	F	T	T
3	F	F	T
4	F	F	F

Table 3. Notion table used in VirScaff algorithm.

Notation	**Description**
Req ()	Tenant's request with essential parameters.
Serv()	Tenant's service with a network-based interface and pre-defined operations
Res ()	A resource that is acted upon by one or more cloud services
Env()	An environment which contains critical information and not related with any entity
D	Details regarding the request including Tenant mail ID, First Name, Last Name etc.,
T_{id}	Tenant Id for the valid request.
SecPass	Security Password for the tenant
SA	Security Authentication
SAP	Security Authentication Plan
R_{id}	Role id for the tenant login into the system
TAR	Tenant Access Rights (TAR) (Refer Table 1)
TOP	Tenant OPeration based on TAR
C_{id}	Command issue id
Q_{id}	Query issued id
S_{id}	Session id
VTA	Virtual Table Authentication
TU_{id}	Table unique id
TUL	Tenant User List
VS_{id}	Virtual Storage specific to T_{id}
FU_{id}	File User Identification in the Virtual Table
FUP_{id}	File Update id in the Virtual Table

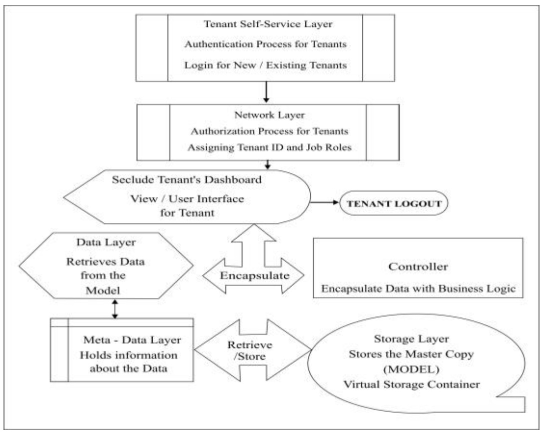
Tenant Self-Service Layer
Authentication Process for Tenants
Login for New / Existing Tenants
Network Layer
Authorization Process for Tenants
Assigning Tenant ID and Job Roles
Seclude Tenant's Dashboard
View / User Interface for Tenant
TENANT LOGOUT
Data Layer
Retrieves Data from the Model
Encapsulate
Controller
Encapsulate Data with Business Logic
Meta - Data Layer
Holds information about the Data
Retrieve /Store
Storage Layer
Stores the Master Copy
(MODEL)
Virtual Storage Container

Figure 2. Flowchart to write the VirScaff Schema.

A Service (Serv) software and hardware with a network-based interface and per-defined operations.

A Resource (Res) that is acted upon by one or more CLOUD services, with a specified set of state data which is either XML documents or JSON documents. An environment (Env) contains information useful in taking the access decision.

4.3.4. Schema for Encapsulation

```
Algorithm VirScaff
{
    Algorithm 1: Tenant Authentication
    Algorithm 1.1: New user login into Scaffold
    Public Virscaff-NewLogin (D)
      {
            If Req(D) == Valid   Then  Send mail
            regarding pass to the Tenant Email ID
            Tenant start login into his mail and
             get his SecPass;
             Endif;
             If Tenant (SecPass) == valid
            Tenant login into Tenant Dashboard
             Endif;
      }

   Algorithm 1.2: For the existing user
   Public VirScaff-ExistLogin (Tid, Rid, TOP, TUid)
     {
            If Tid (SecPass) == Valid
           Then SA is done in the network layer
           Else intimate Tenant as invalid password
           Endif;
          If Tid (Sec Pass) == invalid Then
             Submit forget password. Network layer
             sends alternative SecPass to his mail ID.
          Endif;
    } // End – Algorithm 1
  Algorithm 2: Tenant authorization and session allocation
     Public VirScaff-aur (Tid, TOP, Rid, TUid, Sid)
         {
          If Tid == Valid Then SA grants permission
             To perform TOP- Tenant enter into
             Dashboard. Tid is assigned with a Rid
              according to TA. Rid is assigned with
            TUid.  Tenant Sid starts VSid is assigned
        to Tid   at the end of every Sid.
       Endif;
      }// End – Algorithm 2

Algorithm 3: Encapsulating the scaffold
Public class virscaff-TenantCQRS (Rid, Qid,
                                VTA, VSid, TUL)
  { // Query Method –Virtual Table Authentication
          // create a Seclude tenant specific table
  Public void createTable (Tid, VTA, Sid, Qid,
                                      TOP, TUid,)
     {// query method creates a virtual table with
                Table User id (TUid)
     }
      // Command Method
Public void InsertTable (Tid, VTA, Sid, Cid,
                               TUid, TOP, VSid)
     {  //insert Tenant virtual table by issued a
         // Command id and unique table id (TUid)
        // store the data into Virtual Storage
        // specific to tenant (VSid)
     } // End – InsertTable
Public void Update table (Tid, VTA, Sid, Cid,
                                     TUid, TOP, VSid)
       {  // find tenant's virtual table (VTA),
          // by issuing Command id and
         // unique table id (TUid)
        //update the tenant data through
       // the seclude tenant dashboard
     // store the data into Virtual Storage
    // container specific to Tenant (VSid)
   } // End - Update Table
Public void DeleteTable (Tid, VTA, Sid, Cid, TUid, TOP, VSid)
        {     // find tenant in the data storage by TID
             // Delete the tenant data stored in
            //  the virtual container
        }    // End - DeleteTable
  } // End – Algorithm 3

Algorithm 4: Storage of Tenant's file
Public VirScaff-VitStorageReq (Tid, SAP, TOP, Sid, Rid, FUid,
FUPid, UL)
[Storage of Tenant's specific data]
  {
      Tid is authenticated and authorized
      before storage.  Rid of the Tid is ascertained.
      SAP extracts the file attributes whether
     Tid possesses authority to upload the file or not.
     If system == Tid   upload file   Then
        Extract the owner's information relevant
         to the FUid. Store the file in the seclude
        tenant's virtual storage container
      else return to submit the valid new request
  Endif;
      If  FUid ==  exists and
        Tid wants to insert/delete data into file FUid
        Then  Check tenant information Tid and
        file information FUid are valid,  Update data
        into the file requested by the tenant
     else exception request send by tenant are sent
       back requesting to send valid file request.
       Update the virtual storage container with list of
       Tenant Request List (TUL) against FUid
    Endif;
   } // End – Algorithm 4

Algorithm 5 - Tenant Revocation
Public VirScaff-Logout (Tid, SA, SAP, Sid, Rid, FUid, FUPid,
TUL)
  {  [Tenant Log-out from the Virtual Scaffold]
       If Tid == FUid
          close all virtual files
        Endif;
       If Tid == FUPid
           Close all updated virtual files
        Endif;
       If SAP == Tid
      Close the tenant's session Sid and role Rid
       Endif;
       If SA== TUL valid logout Tid
      from the dashboard and close the session
       Endif;
       Exit from Virtual Scaffold.
  } End – Algorithm 5
} // End of Algorithm VirScaff
```

4.3.5. Performance Analysis for VirScaff Schema

Consider |L| as the number of user grant login with authentication. |N| as the number of tenants logged into the system currently with valid authorization and using the resources according to their job role. The computation complexity is summarised as follows Based on the complexity tabulated in Table 4 and through priori analysis provided on every algorithm considering unambiguous, input, output, finiteness, feasibility and independent it is concluded that the efficiency of the algorithm is commendable for implementation and experimentation using posterior analysis.

Table 4. Complexity for virScaff schema.

Operation	Complexity
User grants	O(|L|)
File Access	O(max| L|,N)
Tenant Revocation	O(N)

In posterior analysis, outcome of priori analysis is taken into consideration and pragmatic implementation is done to prove the effectiveness and efficiency of the algorithms using a suitable experimental setup.

5. Schema Implementation

5.1. Experimental Setup

The existing system earlier resulted with OLAP using either TCP or YCSB based benchmarks. A datasets based on SQL were used to perform OLAP. Further, the related literatures revealed that only limited provision were available for implementing NoSQL document data model [4, 10]. Hence, a modified existing system by name VirScaffSQL have to be devised along with a proposed system by name VirScaff NoSQL both executes CRUD operations.

Considering the implementation of SQL and NoSQL data model for processing in a single SuT, VirScaffSQL using .NET framework with SQL SERVER 2008 R2 (where SQL CRUD queries were executed) and VirScaffNoSQL using Mongo DB (a NoSQL document data model, where JSON document were executed) has been setup as an experimental setup. The business logic (which acts as a Controller) is using C# (C Sharp) and ASP. NET to access the dataset for both VirScaffSQL and proposed new system [16, 19].

5.2. Data Set Description

The experiment is conducted with servers and set of nodes as the clients. The dataset uses two tables namely, Tbl_login and Tbl_tenant. The student dataset is used for processing.

For the purpose of tenant authentication Tbl_login is used and consists of the attributes email-id, password, first name, last name and type of user (admin / user). Every role-based tenant can take the role of DP or DC. The DP will insert or update the table Tbl_tenant (as user) whereas DC will select and read values in the table Tbl_tenant (view the data-as admin).

The other table used is Tbl_tenant. The purpose of this table is to input/modify student data as DP. The administrator can view the data as DC. Tbl_tenant consists of attributes necessary for admission of student into a course viz., student name, name of father /guardian, gender, nationality, religion, stream (vocational/science/others), medium of instruction, address of student, pin code, mobile Number and community. Student has to enter their details as DP and admission will be decided by the management after login into the system and view the data as DC.

5.3. Metrics for Evaluating VirScaff Schema

The existing system used YCSB as benchmark with meager dataset for processing. Further, only limited information were available on usage of tools and metrics for evaluation. Since, cloud environment handles CRUD operations with big data set. A schema should be devised to process the same with minimum time and less storage space. Hence, to check the effectiveness of the VirScaff schema the metrics namely time complexity and space complexity were used.

The time complexity helps to understand the time taken to execute CRUD operations on the student database. Here, time complexity is tested with 10,000 to 1, 00,000 CRUD operations on table Tbl_tenant. The space complexity helps to understand the storage space required for a tenant to store their data into table Tbl_tenant. The space complexity, is tested with 10,000 to 1, 00,000 records stored on the table Tbl_tenant after CRUD operations.

5.4. Result and Discussion

The results were generated with a VirScaffSQL system using. NET framework with SQL SERVER 2008 R2 (where SQL CRUD queries were executed) and VirScaffNoSQL system using MongoDB (a NoSQL document data model, where JSON document were executed).

In order to test the time complexity based on Table 5, initially 10,000 CRUD operations on Tbl_tenant were considered. The VirScaffSQL takes 500 MS to execute the CRUD operations whereas the proposed system takes only 120 MS. When as many as 1,00, 000 CRUD operations are executed, in the case of VirScaffSQL, time required is 2500 MS and on the flip side, the VirScaffNoSQL executes in 900 MS. Hence, this research work, based on Figure 3, concludes that, irrespective of number of CRUD operations performed in the table Tbl_tenant, the proposed VirScaff Schema behaved effectively for VirScaffNoSQL than the VirScaffSQL.

In order to test the space complexity based on Table 6, a seclude tenant has to login into the scaffold successfully and start storing resultant solution after the CRUD operations. The experiment initially stored 10,000 records into Tbl_tenant table. The space occupied by the VirScaffSQL is 12,000 KB.

Table 5. Pragmatic Results for time complexity.

TIME COMPLEXITY		
Time Taken (in Milli-Seconds)		
Number of CRUD Operations	**VirScaffSQL**	**VirScaffNoSQL**
10000	500	120
25000	1000	200
50000	1500	350
75000	2000	500
100000	2500	900

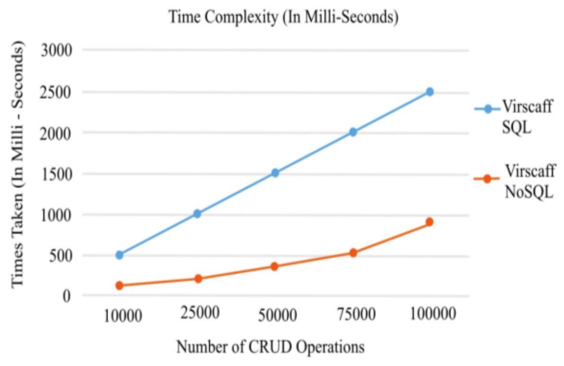

Figure 3. Time Complexity.

Table 6. Pragmatic results for space complexity.

SPACE COMPLEXITY		
Space Occupied (in Kilo-Bytes)		
Number of RecordStored	**VirScaffSQL**	**VirScaffNoSQL**
10000	12000	6000
25000	18000	8500
50000	21000	10000
75000	24000	11000
100000	29000	14000

The VirScaffNoSQL required only 6000 KB of storage space. When 1,00,000 CRUD operations were considered for testing, the VirScaffSQL takes 29000 KB of storage space while the VirScaffNoSQL required only 15,000 KB of storage space to store the student records in Tbl_tenant table. Hence, it is concluded, based on Figure 4, irrespective of records stored in the table Tbl_tenant after CRUD operations, the proposed VirScaff schema behaved efficiently for VirScaffNoSQL than the VirScaffSQL.

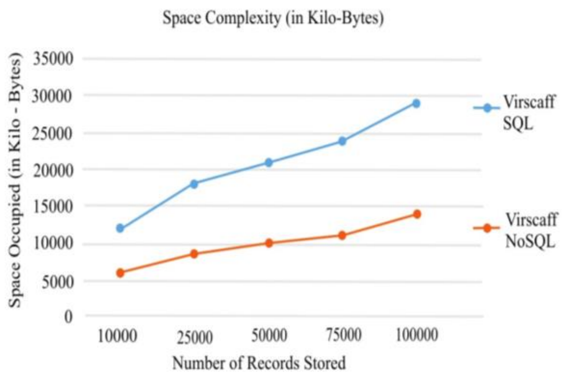

Figure 4. Space complexity.

6. Conclusions

This contribution assumed an enhanced virtual scaffold as system model and used a MTCG pattern language to provide a model view controller to the system. RBAC was used to provide the role to the tenant, thus, restricting the snooper intruding into the system. The VirScaff schema, implemented, acts as a controller, while developing multi-tenant SaaS application. The experiment is conducted on performance analysis in terms of time and space complexity concluded that the VirScaff schema is efficient to implement and effective to work with CRUD operations with minimum time and occupies less storage space. To sum up, this contribution will be a compendium for the skeptics and researchers, who want to develop multi-tenant SaaS application using NoSQL document data model. The future research work includes enhancing MTCG pattern and work with the Dockers servers and provide container with Multi-Cloud environment as a mobile application with alternative document database model as a back-end which will be useful for people in all walks of life anywhere any time.

References

[1] Balasubramanian N. and Suguna J., "A Virtual Scaffold for Storage Multi-Tenant SaaS Data Models," *International Journal of Applied Engineering Research*, vol. 10, no. 20, pp. 40775-40780, 2015.

[2] Balasubramanian N. and Jayapal S., "Rumination on Scaffold Pattern Language for Multi-Tenant SaaS Application Development," *International Journal of Control Theory and Application*, vol. 9, no. 16, pp. 8257-8265, 2016.

[3] Balasubramanian N. and Jayapal S., "Enhanced Scaffold Design Pattern for Seclude Multi-tenant SaaS Application," *in Proceedings of 1st International Conference on Computational Intelligence and Informatics*, Hyderabad, pp. 671-680, 2016.

[4] Dede E., Govindaraju M., Gunter D., Canon R., and Ramakrishnan L., "Performance Evaluation of a mongo DB and Hadoop Platform for Scientific Data Analysis," *in Proceedings of 4th ACM Workshop on Scientific Cloud Computing*, Tucson, pp. 13-20, 2013.

[5] Fehling C., Leymann F., Retter R., Schumm D., and Schupeck W., "An Architectural Pattern Language Of Cloud-Based Applications," *in Proceedings of 18th Conference on Pattern Languages of Programs*, Portland Oregon USA, pp. 13-20, 2011.

[6] Ferraiolo D., Sandhu R., Gavrila S., Kuhn D., and Chandramouli R., "Proposed NIST Standard for Role-Based Access Control," *ACM*

Transactions on Information and System Security, vol. 4, no. 3, pp. 224-274, 2001.

[7] Ferralolo D. and Richard K., "Role-Based Access Control," *in Proceedings of 15th National Computer Security Conference*, Baltimore, pp. 554-563, 1992.

[8] Habiba M., Islam M., and Ali A., "Access Control Management for Cloud," *in Proceedings of 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, Melbourne, pp. 485-492, 2013.

[9] Hammes D., Medero H., and Mitchell H., "Comparison of NoSQL and SQL Databases in the Cloud," *in Proceedings of the Southern Association for Information Systems Conference*, Macon, pp. 1-8, 2014.

[10] Han J., Song M., and Song J., "A Novel Solution of Distributed Memory NoSQL Database for Cloud Computing," *in Proceedings of 10th IEEE/ACIS International Conference on Computer and Information Science*, Sanya, pp. 351-355, 2011.

[11] Hou D., Zhang S., and Kong L., "Placement of SaaS Cloud Data and Dynamically Access Scheduling Strategy," *in Proceedings of 8th International Conference on Computer Science and Education*, Colombo, pp. 834-838, 2013.

[12] Hui M., Jiang D., Li G., and Zhou Y., "Supporting Database Applications As A Service," *in Proceedings of IEEE 25th International Conference on Data Engineering*, Shanghai, pp. 832-843, 2009.

[13] Jacobs D. and Aulbach S., "Ruminations on Multi-Tenant Databases," *in Proceedings of Datenbanksysteme in Business*, Technologie and Web, Aachen, pp. 514-521, 2007.

[14] Marinho T., Cidreira V., Claro D., and Mane B., "Midas: A middleware to Provide Interoperability between SaaS and DaaS," *in Proceedings of the XII Brazilian Symposium on Information Systems on Brazilian Symposium on Information Systems: Information Systems in the Cloud Computing*, Brazilian, pp. 401-408, 2016.

[15] Mehar D., Vishwakarma G., and Jain Y., "Modified Fine-Grained Data Access Control Algorithms for File Storage Cloud," *International Journal of Computer Applications*, vol. 116, no. 22, pp. 15-19, 2015.

[16] Ni J., Li G., Zhang J., Li L., and Feng J., Adapt: Adaptive Database Schema Design for Multi-Tenant Applications," *in Proceedings of 21st ACM International Conference on Information and Knowledge Management*, pp. 2199-2203, 2012.

[17] Okman L., Gal-Oz N., Gonen Y., Gudes E., and Abramov J., "Security Issues in Nosql Databases," *in Proceedings of 10th International Conference on Trust, Security and Privacy in Computing and Communications*, Changsha, pp. 541-547, 2011.

[18] Solomon M., Sunderam V., and Xiong L., "Towards Secure Cloud Database with Fine-Grained Access Control," *in Proceedings of IFIP Annual Conference on Data and Applications Security and Privacy*, Vienna, pp. 324-338, 2013.

[19] Tang B., Sandhu R., and Li Q., "Multi-Tenancy Authorization Models for Collaborative Cloud Services," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 11, pp. 2851-2868, 2015.

[20] Tang B., Li Q., and Sandhu R., "A Multi-Tenant RBAC Model for Collaborative Cloud Services," *in Proceedings of 11th Annual Conference on Privacy, Security and Trust*, Tarragona, pp. 229-238, 2013.

[21] Tudorica B. and Bucur C., "A Comparison between Several Nosql Databases with Comments and Notes," *in Proceedings of RoEduNet International Conference 10th Edition: Networking in Education and Research*, Iasi, pp. 1-5, 2011.

[22] Xiaorong C. and Tianqi L., "A Trusted Virtual Network Construction Method Based on Data Source Dependencies," *The International Arab Journal of Information Technology*, vol. 16, no. 5, pp. 889-893, 2019.

[23] Yu S., Wang C., Ren K., and Lou W., "Achieving Secure, Scalable, and Fine-Grained Data Access Control in Cloud Computing," *in Proceedings IEEE INFOCOM*, San Diego, pp. 1-9, 2010.

Nagarajan Balasubramanian completed his bachelor's of Science in Computer Science from Sri Sankara Arts and Sciece College, Kanchipuram, India in 1995. He has completed his Master of Computer Applications (MCA) from Arunai Engineering College, Tiruvannamalai, India in 1998 and his Master of Philosophy in Computer Science from Bharathiar university in 2006. He recently (on 09 December 2019) defended successfully his Phd., Thesis at R&D center, Department of Computer Science, Bharathiar University, Coimbatore, India. He has around 20 + years of teaching experience at Master degree. His research interest were Cloud storage and security.

Suguna Jayapal received the Master's degree in mathematics from Annamalai University, Chidambaram in 1988 and the Ph.D. degree in Computer Science from the Bharathiar University, Coimbatore in 2009. She is currently an Associate Professor with the Department of Computer Science, Vellalar College for Women (Autonomous), Erode, Tamil Nadu. Her research interests are AI, Data Mining, Text Mining and Image Processing. She is the author or co-author of over 30 publications in journals, conference proceedings and book chapters. She has presented a paper in an International Conference held at Cincinnati University, Cincinati, Ohio, USA. She has produced over 22 M.Phil. Scholars, 2 Ph.D. Scholars in Computer Science and guiding 6 Ph.D. Scholars at present. Completed one Mnor Research Project funded by UGC.

Satheeshkumar Janakiraman is with the Department of Computer Applications, School of Computer Science and Engineering, Bharathiar University, Coimbatore, Tamil Nadu, India. He received his Masters in Computer Applications (MCA) during 1999 and Doctor of Philosophy during 2010. He is having 18 plus years of research and teaching experience. He has published more than 140 research articles in reputed journals and conference proceedings. He is the member of IEEE, IETE and CSI. He completed two research projects under university grants commission and one collaborative project for the funding of 60 lakhs under the Department of Science and Technology (DST). He acted as an organizing secretary for various international conferences and chaired sessions in international and national conferences. His area of specialization includes soft computing, networks, Image processing and medical imaging. He is an eminent speaker who visited and delivered speech in more than 125 institutions. He is an identifiable researcher in the area of Indian Music and its influence on human brain.