CSCI 5408: Assignment 5

Multi-tenant Database Access Control [1].

1. Summary:

The *Multi-tenant Database Access Control* [1] paper proposes an alternative and much more efficient and effective multi-tenant access control model that provides access control for multiple tenants and multiple users per tenant. This alternative is a self-created database schema called the Elastic Extension Tables (EET). The proposed model also defines access control data architecture along with the EET access grants. Lastly, they also propose an algorithm that allows the users to access data granted to them based on various groups or roles assigned to them.

Moving on to the growth of multi-tenant cloud computing services – The users do not prefer switching up their services just for the sake of reducing their Total Cost of Ownership (TCO) even though the data is being outsourced to the cloud. So, three data isolation approaches can be applied: The Separate Database or the Shared Database – Separate Schema or the Shared Database – Shared Schema. For this paper, the author has chosen the *Shared Database – Shared Schema* approach, where data isolation among tenants is done with the help of a Tenant ID column which differentiates and isolates the tenant's data. So, this multi-tenant approach consists of two data types: shared tenants' data and tenants' isolated data, which becomes complete on being combined.

The above approach has its fair share of challenges like;

- How to isolate tenant's data so that only that tenant can access his data and no one else can.
- Making sure that the tenant's data is robust as well as secure.
- Need to optimize the performance of the database.
- A database structure needs to be designed that works well with different business domain applications.
- Consider the fulfilment of various business requirements of different tenants by utilizing a tenant-aware data management based on Shared Database – Shared Schema approach.

The Elastic Extension Tables (EET) are the proposed novel multi-tenant database schema design consisting of various sub-tables that constructs the entire structure of the model:

- Common Tenant Tables (CTT) – They are shared between all the tenants using a single instance of the multi-tenant database, consisting of physical, relational tables.
- Virtual Extension Tables (VET) – They are mainly used to expand on the existing business domain providing relationships. It creates virtual database tables, virtual database relationships, and other database constraints.
- Extension Tables (ET) – They are used to create the data architecture of the system as it consists of eight tables used to construct the VETs.

The author ran two experiments to verify the effectiveness and efficiency of granting a tenant user accessibility on his rows and columns. The Elastic Extension Tables Access Control (EETAC) and Elastic Extension Tables Proxy Service (EETPS) are used. The results showed that the cost of executing any query for a user being granted access to fewer table columns or rows is less than the cost of a user being granted access to more table columns and rows.

A proxy service is created that helps fetch the data from the tables that perform similar functionalities of SELECT query, joins, and relationship outputs. These proxy services retrieve outputs in a specific pre-defined self-created format where the tenant does not have to write any queries. Various access tables decide what data can be accessed by what part of the tenant's users. A variety of tables represent a single entity like users and tenants, whereas some represent multiple entities such as a group table, role table, and many more. Various join access control tables also include information about which user belongs to which group and their respective roles and other information.

Hierarchical access to data has been used where the user is assigned to a particular group, and its respective roles are checked. One more thing that is checked is which all columns and rows can the user access, and hence accordingly, a particular block of data is provided as per the user's query. Various algorithms have been demonstrated with their sudo codes showcasing the working functionalities on obtaining the user roles and number of columns that the user has the right to access. The output generated by using these algorithms mentioned above then becomes the SELECT query's input having the WHERE clause. So, finally, after performing both the experiments proposed by the author in the paper. The results showed that low cost is incurred if a user with a lower role and access tries to query the database rather than a user with the higher role and access rights on the database.

2. Scope of Improvements:

There are a few areas in the paper where some improvements could be made. As the paper focuses on various approaches used to implement multi-tenant database access control, several existing and novel methods and solutions have been suggested. It covers the challenges faced by various solutions and workarounds to achieve security and efficiency in using multi-tenant database access control models.

The first improvement can be made in the experimental setup, where the Java, Spring, Hibernate, PostgresSQL, and Jboss versions are obsolete. The latest versions have better support and improved efficiency, in turn affecting the experiment results. Secondly, the experiments are performed only for one tenant, thus limiting the various possibilities of testing multiple tenants with multiple users. As per fig. 6 [1], testing was done for only 1/200th of the data. Instead, if it would have been for 1/100th, the retrieval time could have been appropriately assessed. The experiments performed covered accessing data from table columns and table rows. One more experiment that could be added is accessing data from different databases.

The experiments can also be assessed on the OLTP benchmark, such as TPC-C and Yahoo! Cloud Servicing Benchmark (YCSB) [2]. These experiments would reveal how the model performs in today's Big Data and Cloud dominated world, focusing mainly on latency and throughput.

References:

[1] H. Yaish and M. Goyal, "Multi-tenant Database Access Control," in 2013 IEEE 16th International Conference on Computational Science and Engineering, Sydney, Australia, 2013.

[2] G. B. P. and P. J. , "Secure and efficient multi-tenant database management system for cloud computing environment," in International Journal of Information Technology (2020), Switzerland , 2020.