Global Scientific JOURNALS

GSJ: Volume 8, Issue 6, June 2020, Online: ISSN 2320-9186
www.globalscientificjournal.com

An Implementation of Enhanced Multi-tenancy Database Design in IoT System

Magy Ali ElBanhawy[1], Walaa Saber[2] and Fathy Amer[3]
[1]Electrical Engineering Dept., Faculty of Engineering, Port Said University, Port Said, Egypt.
[2]Electrical Engineering Dept., Faculty of Engineering, Port Said University, Port Said, Egypt.
[3]Computer Sciences Dept., Faculty of Computer and Information Science, 6 October University, Cairo, Egypt.

***Abstract* -- Since the Internet of Things (IoT) has become more and more important, additional designs should be proposed daily to adapt the specificities introduced by this systems of the physical world (Sensors and Actuators) and the public networks (The Internet). Some of these solutions use a Cloud Computing and Multi-tenancy structures as Multi-tenancy increases resource and utilization as well as sharing the same database instance to multiple tenants. Multi-tenancy, which lets multiple tenants share a single application instance securely, is a key enabler for building such a middleware. However, merging the multi-tenancy structure with IoT systems becomes more and more challenging due to the different data types of IoT and the multi-tenancy design should adapt to tenants workloads and fit their special requirements and the fact that the multi-tenant database is shared between multiple tenants. Therefore, this paper presents An Implementation of Enhanced Multi-tenancy Database Design in IoT System which should allow the IoT system to store its data in a multi-tenant environment with less cost and better performance based on the Enhanced Elastic Extension Tables (E3T) and Native XML Database (NXD). The presented implementation achieves high scalability and increases performance for tenants who need those requirements to intermediate with a simple IoT system.**

***Keywords* -- IoT, Multi-tenant Database, E^{3}T, Cloud Computing, SaaS, NXD-Schema.**

1. INTRODUCTION

The definition of the Internet of Things (IoT) has evolved due to the convergence of multiple technologies, machine learning and embedded systems in [1] and [2]. It is considered is a system of interrelated computing devices, mechanical and digital machines, objects, that are presented with specific identifiers and the possibility for transferring data over a network. Due to increase of user's requests to millions of online users application in a small period of time, organizations often spend large amounts of their time, resources and money managing and supporting information stored in their on- premises databases, to ensure that the correct information is available when it is required. Cloud computing is a paradigm for allowing suitable, on- demand network access to a shared pool of configurable computing resources (networks, servers, storage, applications, and services, e.g.) that can be rapidly provisioned and emitted with minimal management effort or service provider interaction [3]. SaaS offers a single configurable software and computing environment for multiple tenants [4]. The remainder of this paper is structured as follows. Section 2 reviews the related work for multi-tenancy main approaches, section 3 describes the An Implementation of Enhanced Multi-tenancy Database Design in IoT System, and section 4 concludes this paper and descries the future work.

2. Related Work

Many designs have been attempting to establish a robust multi-tenancy schemas as in [5] , [6] , [7] , [8] and [9] , however each one of the following technique has it's limitation, Therefore the need of stable schema is required. This section presents several of multi- tenant database algorithm techniques, including Private Tables, Extension Tables, Universal Table, Pivot Tables, Chunk Table, Chunk Folding, and XML Table.

Private Tables' technique as in [5] and [10] The Private Tables technique permits tenants to establish their own tables, which can be expanded and changed. Using this multi-tenant private tables technique can be transformed from one tenant to another by renaming tables each one a unique table name, and metadata without using extra

columns like **'tenant_id'** to distinguish and isolate the tenants' data.

The extension tables appeared after The Private Tables, These extended tables are separated tables joined with the base tables by adding tenants' metadata columns like **'tenant_id'** to construct source tables. This technique based on the Decomposed Storage Model described in [6] , [11], that splitting up n-columns table into n 2-column tables joined using surrogate values. All tenants can access the base tables and some extension tables. This method allow tenants to enlarge there number of tables with complexity cost.

The Universal Table (Spare Columns) as defined in [5] , [7] and [12] a sparse dataset consists of a large number of columns which allow tenants to store their columns directly without any other tables. The Universal Table is a table that contains extra columns of the base application schema columns. These columns name must be uniquely identified. The basic columns in the universal table are **'tenant_id'** and **'table_id'** with some general columns such as '**first_name'** and **'last_name'**. These metadata columns must have a VARCHAR data type to ensure that different data values can be stored in these columns.

In Pivot Tables technique in [13] , [7] , [6] and [8] this technique maps the schema into common structure in the database, in which each record is store with its meta data of as the following **'tenant_id'** , **'table_id'** , **'col_id'** and **'row_id'**. The rows in the Pivot Table contain four columns, including tenant, table, column, and row that identify which row in the logical source table refers to. Tenant column **'tenant_id'** refers to the particular tenant. Table column **'table_id'** refers to the particular table. Column **'col_id'** refers to the particular column. Tenant row **'row_id'** refers to the particular row. As well as a single data type column will contain all other rows data of the same data type. For example, the Pivot Tables can have two pivot tables, the first table **'pivot_int'** to store data which has INTEGER values, and the second table **'pivot_str'** to store data that has STRING values. The NULL values in this approach will be minimized.

The Chunk Table is another public structure technique that is comparable to Pivot Table. Except, it contains a group of data columns with a mix of data types that replacing the column 'col' in the Pivot Table with 'chunk' column in the Chunk Table.

Chunk Folding (CF) is derived from the Chunk Table technique and the Extension Tables as presented in [8] , [4] [14] and [9]. This schema vertically divided virtual tables into chunks. Each group has a **'chunk_id'** where a chunk of columns is partitioned into a group of column. This approach works by containing the most commonly used parts of the schema into base tables.

Chunk Table" technique and Chunk Folding" technique were having the capability to limit the number of tables, but on the other hand increase the query complexity, due to the joining operation.

The XML Table database extension technique is a mixture of relational database systems and Extensible Markup Language (XML) in [15],[2] and [16]. This is accomplished by either providing an XML data type or by adding the XML document into the database as Large Object (Character or Binary). That has a minimum time in the insertion and retrieval queries, but at the expense of the overall performance. XML database extension technique facilitating the creation of database tables, columns, views, variables and parameters, and isolating the application from relational data model. All the previous techniques preform all tenants' requirements as the original database relational schema cannot be changed, and XML table is performed by several relational database products techniques.

In summary, although all the previous technique had introduced an algorithm to establish a multi-tenant database design, each one of the previous techniques had its own limitation as explained in *Table 1*.

No.	Multi-tenancy Database Design	Limitation
1	Private Tables	This Technique would be suitable only when small number of tenants are using it.
2	The extension tables	The main inability of this technique lies in the way the actual dividing is performed as it leaves them in naturally occurring groups.
3	The Universal Table	This Technique low limitation is its performance due to the high number of NULL values it may contain as a result of merging all tenants columns in one table.

4	Pivot Tables	The first limitation is the overhead existing of the meta-data. The second one is the complexity of the recreation of the tables as n-column logical table requires (n x 1) aligning joins.
5	The Chunk Table	The only limitation of this technique is the complexity of the query.
6	Chunk Folding	This technique limitation is different from all other as its performance is acceptable but, the common schema must be well-known in advance and this consider a huge draw in databases systems.
7	The XML Table	This technique reduces the overall performance using XML files due to it consume large space of the RAM memory, as all XML file is loaded in the RAM.

Table 1.Limitations of Multi-tenancy Databases Techniques

3. Introducing The Enhanced Elastic Extension Tables for Multi-tenancy Databases to IoT System

The Previous proposed of the Enhanced Elastic Extension Tables (E^3T) database schema is a more advanced way of designing and creating a multi-tenant database that is used in order to minimize the workload in cloud data storage architecture. This paper proposes is based on a multi-tenant database architecture of the (E^3T) to integrate, create, and preform tenants' queries. This contribution consists of applying the (E^3T) database design for enabling tenants' applications to manipulate data efficiently by using the Native XML Database (NXD) as multi-tenant storage. And, applying this database structure to the internet of things (IOT) as presented in [17], [18] ,[19] and [21]. These objectives motivate us to propose An Implementation of Enhanced Multi-tenancy Database Design in IoT System.

3. Introducing The Enhanced Elastic Extension Tables for Multi-tenancy Databases to IoT System

The Previous proposed of the Enhanced Elastic Extension Tables (E^3T) database schema is a more advanced way of designing and creating a multi-tenant database that is used in order to minimize the workload in cloud data storage architecture. This paper proposes is based on a multi-tenant database architecture of the (E^3T) to integrate, create, and preform tenants' queries. This contribution consists of applying the (E^3T) database design for enabling tenants' applications to manipulate data efficiently by using the Native XML Database (NXD) as multi-tenant storage. And, applying this database structure to the internet of things (IOT) as presented in [17], [18] ,[19] and [21]. These objectives motivate us to propose An Implementation of Enhanced Multi-tenancy Database Design in IoT System.

3.1. Enhanced Elastic Extension Tables (E^3T)

The (E^3T) is the most recent way of designing and structuring a multi-tenant database as shown in [4], [7], [9], [20] and [21]. This design consists of three partitions of tables. The first class is Enhanced Common Tenant Tables (ECTT), the second partition is Enhanced Virtual Extension Tables (EVET) and the third partition is E^3T, which consists of eleven tables that are used to construct EVETs. Any requirements could be extended tenants' existing business database. Which will allow tenants to establish their virtual database structures from scratch by creating: (I) Virtual database tables' structures with virtual columns include virtual rows. (II) Virtual database relationship structure between the virtual tables. (III) Other database facilities such as adding primary key, foreign key, triggers, index column and routines. The E^3T tables are the sensible tables that grant tenant the virtual database structure and its facilities which consist of eleven tables that are used to establish tenant's EVETs shown in *Figure 1 and Table 2.*

No.	**Table Name**	**Table Features**

1	***"db_table"***	The first E^{3}T table that permits tenants to create logical tables
2	***"table_column"***	This E^{3}T table is the second table that permits tenants to create logical columns.
3	***"table_row"***	This E^{3}T table used for storing values like INT, FLOAT, DATE, VARCHAR, BOOLEAN, and other data types and large data values in other two tables.
4	***"table_row_blob"***	This E^{3}T table for storing the BLOB values for virtual columns with BLOB data type.
5	***"table_row_clob"***	This E^{3}T table is for storing all CLOB values for virtual columns with CLOB data type.
6	***"table_relationship"***	This E^{3}T table permits tenants to create virtual relationship with the virtual columns.
7	***"table_index"***	This E^{3}T table is used to add and create indexes for the virtual columns.
8	***"table_primary_key_column"***	This E^{3}T table permits tenants to create virtual primary key for the virtual columns.
9	***"table_trigger"***	This E^{3}T table permits tenants to grant triggers.
10	***"table_procedure"***	This E^{3}T table permits tenants to use procedures when it is essential to have a repetitive task.
11	***"table_routine"***	This E^{3}T table permits tenants to preform routines.

Table 2. The E^{3}T eleven tables

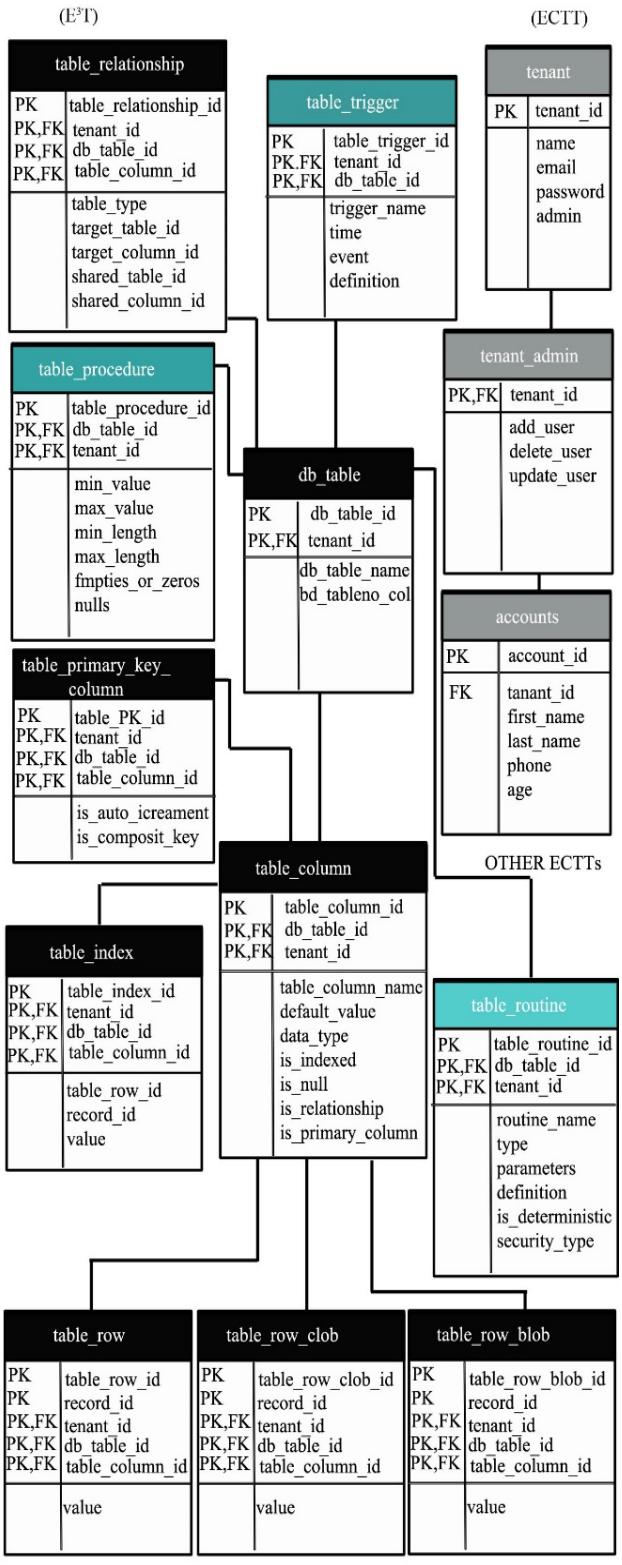

Figure 1. Enhanced Elastic Extension Tables

3.2. Using Native XML with the E^{3}T Database

The XML database design limitation was consuming large space of the RAM memory, as all XML file is loaded in the RAM. The Native XML databases (NXD) provide the support multi-tenancy architecture reduces this consummation of RAM because it divided the database into partitions as in [15] , [9] ,[17] and [22] which are generally classified as native and relational. When NXD data is preformed into relational databases, NXD execute queries need to be transformed to SQL queries in the relational data. Using NXD will provide the multi-tenancy databases to be relational with the virtual tables to each tenant as well as will increase the overall performance.

3.3. Applying E^{3}T database to the internet of things (IoT)

The Internet of Things (IoT) is posing new challenges and perspectives for data management techniques. For storing small data types values such as (INT, FLOAT, DATE, TIMESTAMP, TIME, CHAR, VARCHAR, BOOLEAN and so on) the E^{3}T table ***"table_row"*** table that is used for storing theses values. Since to store multimedia files is required of large size so, the blob (Binary Large Object) data type could be used in [6], [10], [23], [25]. Then the E^{3}T table *"table_**row_blob"*** is used. For other data types such as images that required medium data size values the E^{3}T table ***"table_row_clob"*** can be the suitable data type for it [25]. If a user intends to trigger an action that activates when a specific event occurs for the specific event, according to E^{3}T table ***"table_trigger"*** can be used to accomplish this case [17] , [2] and [19]. Case a user intends to perform a procedure as a result of specific data, in this case the ***"table_procedure"*** which used when it is essential to have a repetitive task that requires checking, looping and multiple statements do it with a single call to a procedure that's stored on the server with a return output values that can be stored to start many another actions [24] , [23] and [1] . The last case scenario is that when user attend to create routine which mainly used to gather parameters and compute values following by single return value which preventing the need to keep reissuing the individual statements. For this finally case the E^{3}T table ***"table_routine"*** may be used as presented in [24] and [25]**.**

4. Conclusion

In this paper, we introduced An Implementation of Enhanced Multi-tenancy Database Design in IoT System. The E^{3}T allows tenants to perform their own elastic relational database schema. The E^{3}T can be implemented in the IoT systems as it provide the tenants with the essential needs for a simple IoT.

5. Future work

In the future publications will focus on reduce data to fulfill multi-tenant business needs and cost in the storage section as shown in the previous section. The E^{3}T will be focused on how to ensure that proper security of these tenants' data. By applying layers of security with the multi-tenant security methods, tenant data will be secured from any unauthorized access, miss data retrieval and SQL injection attacks. And Also in the future work The E^{3}T will introduce the concept of NoSQL as One of the biggest challenges is the amount of data; more specifically, the system's difficulties in accepting and processing a large amount of data in as little time as possible and this NoSQL data storage aims to be very fast and always available.

References

[1] M. U. Kalay, "Database System Suggestions for the Internet of Things (Iot) Systems," *Mugla J. Sci. Technol.*, no. October, pp. 46–52, 2018.

[2] M. G. Kibria, S. Ali, M. A. Jarwar, and I. Chong, "A framework to support data interoperability in web objects based IoT environments," *Int. Conf. Inf. Commun. Technol. Converg. ICT Converg. Technol. Lead. Fourth Ind. Revolution, ICTC 2017*, vol. 2017-Decem, pp. 29–31, 2017.

[3] R. Hope and B. Ca, "DATABASE HIGH AVAILABILITY AS A SERVICE FOR CLOUD COMPUTING," pp. 1–10, 2016.

[4] S. Singhal, S. Maheshwari, and M. Meena, *Mapping Database Layer for the SaaS-Based Multi-tenant Application*, vol. 707. 2019.

[5] H. Yaish, M. Goyal, and G. Feuerlicht, "A Proxy Service for Multi-tenant Elastic Extension Tables," *Springer-Verlag Berlin Heidelb.*, vol. 2, pp. 1–33, 2015.

[6] S. Fang and Q. Tong, "A Comparison of Multi-Tenant Data Storage Solutions for Software-as-a-Service," *ICCSE 2011 - 6th Int. Conf. Comput. Sci. Educ. Final Progr. Proc.*, pp. 95–98, 2011.

[7] H. Yaish and M. Goyal, "Multi-tenant database access control," *Proc. - 16th IEEE Int. Conf. Comput. Sci. Eng. CSE 2013*, pp. 870–877, 2013.

[8] Y. D. and Z. X. Li heng, "Survey on Multi-Tenant Data Architecture for SaaS," *IJCSI Int. J. Comput. Sci. Issues*, vol. 9, no. 6, pp. 198–204, 2012.

[9] H. Yaish, M. Goyal, and G. Feuerlicht, "Evaluating

the performance of multi-tenant Elastic Extension Tables," *Procedia Comput. Sci.*, vol. 29, pp. 614–626, 2014.

[10] B. Alam, M. N. Doja, M. Alam, and S. Mongia, "5-Layered Architecture of Cloud Database Management System," *AASRI Procedia*, vol. 5, pp. 194–199, 2013.

[11] H. Yaish, M. Goyal, and G. Feuerlicht, "An Elastic Multi-tenant Database Schema For Software as a Service," *Proc. - IEEE 9th Int. Conf. Dependable, Auton. Secur. Comput. DASC 2011*, pp. 737–743, 2011.

[12] S. Aulbach, T. Grust, D. Jacobs, A. Kemper, and J. Rittinger, "Multi-tenant Databases for Software as a Service: Schema-Mapping Techniques," *Proc. ACM SIGMOD Int. Conf. Manag. Data*, pp. 1195–1206, 2008.

[13] H. Yaish and M. Goyal, "A Multi-tenant Database Architecture Design for Software Applications," *2013 IEEE 16th Int. Conf. Comput. Sci. Eng.*, pp. 933–940, 2013.

[14] O. Schiller, B. Schiller, A. Brodt, and B. Mitschang, "Native support of multi-tenancy in RDBMS for software as a service," *Proc. 14th Int. Conf. Extending Database Technol. - EDBT/ICDT '11*, p. 117, 2011.

[15] S. Balamurugan and A. Ayyasamy, "Performance Evaluation of Native XML Database and XML Enabled Database," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 7, no. 5, pp. 182–191, 2017.

[16] M. N. A. Khan, A. Shahid, and S. Shafqat, "Implementing a storage pattern in the OR mapping framework," *Int. J. Grid Distrib. Comput.*, vol. 6, no. 5, pp. 29–38, 2013.

[17] S. Wu, "A Method for Building Shared Massive Heterogeneous IoT Data Environment," *Proc. - 2018 5th Int. Conf. Inf. Sci. Control Eng. ICISCE 2018*, pp. 40–45, 2019.

[18] N. Dalčeković, S. Vukmirović, S. Stoja, and N. Milošević, "Enabling the IoT paradigm through multi-tenancy supported by scalable data acquisition layer," *Ann. des Telecommun. Telecommun.*, vol. 72, no. 1–2, pp. 71–78, 2017.

[19] P. T. A. Mai, J. K. Nurminen, and M. Di Francesco, "Cloud databases for internet-of-things data," *Proc. - 2014 IEEE Int. Conf. Internet Things, iThings 2014, 2014 IEEE Int. Conf. Green Comput. Commun. GreenCom 2014 2014 IEEE Int. Conf. Cyber-Physical-Social Comput. CPS 20*, no. iThings, pp. 117–124, 2014.

[20] H. Yaish, M. Goyal, and G. Feuerlicht, "Proxy service for multi-tenant database access," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8127 LNCS, pp. 100–117, 2013.

[21] H. Yaish, M. Goyal, and G. Feuerlicht, "Elastic Extension Tables for Multi-tenant Cloud Applications," 2016.

[22] J. Ni, G. Li, L. Wang, J. Feng, J. Zhang, and L. Li, "Adaptive Database Schema Design for Multi-Tenant Data Management," *IEEE Trans. Knowl. Data Eng.*, pp. 1–1, 2013.

[23] I. Fosic and K. Šolic, "Graph database approach for data storing, presentation and manipulation," *2019 42nd Int. Conv. Inf. Commun. Technol. Electron. Microelectron. MIPRO 2019 - Proc.*, pp. 1548–1552, 2019.

[24] J. A. Stankovic, "Research directions for the internet of things," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 3–9, 2014.

[25] S. Cherrier, Z. Movahedi, and Y. M. Ghamri-Doudane, "Multi-tenancy in decentralised IoT," *IEEE World Forum Internet Things, WF-IoT 2015 - Proc.*, no. October, pp. 256–261, 2015.