International Journal of Electrical and Computer Engineering (IJECE)
Vol. 10, No. 6, December 2020, pp. 6618~6628
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i6.pp6618-6628

A native enhanced elastic extension tables multi-tenant database

Magy El Banhawy[1], Walaa Saber[2], Fathy Amer[3]
[1,2]Department of Electrical Engineering, Faculty of Engineering, Port Said University, Egypt
[3]Department of Computer Sciences, Faculty of Computer and Information Science, 6 October University, Egypt

Article Info

Article history:

Received Apr 10, 2020
Revised May 29, 2020
Accepted Jun 6, 2020

Keywords:

Cloud computing
EET
IoT
Multi-tenant database
NXD-schema

ABSTRACT

Multi-tenancy is a cloud computing service that enables multiple customers to share the same software. The internet of things (IoT) is based on distributing devices and objects through the internet that needs an effective multi-tenant schema. Merging the multi-tenant structure with IoT systems becomes more challenging due to the different data types of IoT and workloads. There is a need to handle a massive amount of metadata by multi-tenant applications. This paper has two main aims to solve this problem. First, it proposes an enhanced elastic extension tables (E^3T) schema to query the IoT database efficiently. Second, it proposes a Hybrid multi-tenant database (NXD-E^3T) as a combination of the E^3T and Native XML Database to reduce the response time and support a common query language. The proposed schema was implemented on an IoT benchmark. The simulation results show that the proposed NXD-E^3T outperforms the other multi-tenant databases in terms of the query speed.

Corresponding Author:

Magy El Banhawy,
Department of Electrical Engineering,
Port Said University,
Port Said, Egypt.
Email: mbanhawy2004@gmail.com

1. INTRODUCTION

Due to increasing the user's requests to millions of online users, organizations often spend large amounts of their time, resources and cost in their database stored, to ensure that the correct information is available when it is required. Cloud computing relies on sharing of various resources such as networks, servers, storage, applications, and services to acquire coherence and economies of scale [1]. These services can accomplish in many forms such as outsourcing, on-demand resource provision, the economics of scale, and pay-as-you-go. This cloud software approached is called software as a service (SaaS). SaaS is capable of presenting a single configurable software and computing environment for multiple tenants [2, 3]. SaaS applications are deployed as hosted services and accessed from everywhere over the web. The idea of outsourcing software through several experiences as in [4].

Many companies desire to outsource their data to a third party which hosts a multi-tenant database system to provide the data management service with much fewer resources, through the same server instance. The main benefit of multi-tenancy is to minimize the operating costs of running software from the provider's perspective [5]. Many problems appeared while achieving this target such as lacking the tenant privileges and the long time required to perform a single query.

Therefore, this paper proposed a native enhanced elastic extension tables database (NXD-E^3T) as a combination of the proposed E^3T schema and the Native XML data model to increase the performance by computing the response times of the preformed query. Appling this Schema to the IoT System is considered an additional contribution in this paper. The remainder of this paper is structured as follows. Section 2 reviews the background for multi-tenancy main approaches and the related work of multi-tenancy database techniques, section 3 describes the proposed Schema for NXD-E^3T Multi-Tenant Database, section 4 proposes the results for the performance of our proposed method. Finally, section 5 concludes this paper and describes future work.

2. BACKGROUND AND RELATED WORKS

2.1. Background and basics concept

Since IoT has become more significant, additional designs must be presented each day to adapt the specificities introduced by these systems of the physical world and the public networks [6-8]. Some of these solutions based on SaaS and Multi-tenancy [9]. Paying the multi-tenancy models based on the way how data is isolated for individual tenants, there are three main approaches of multi-tenancy databases as shown in Figure and in [10]. Multi-tenancy is based on Shared DB–Shared Schema [11]. Multi-tenancy is capable of serving a large number of tenants with a minimum general cost [12].

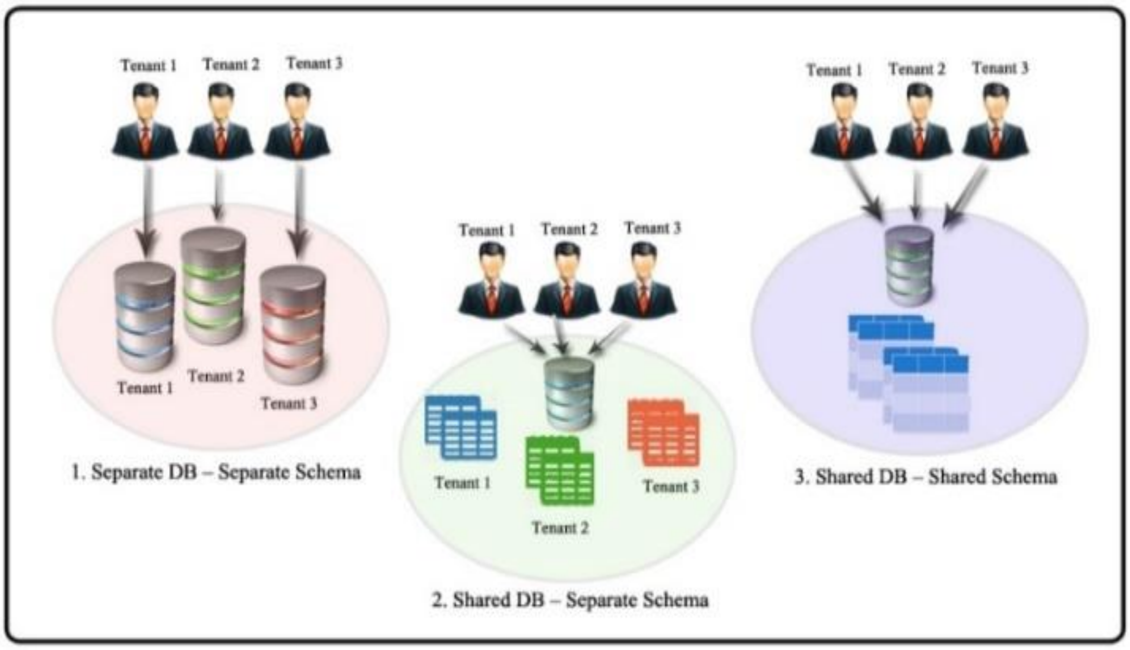


Figure 1. Approaches to managing multi-tenancy

2.2. Related works

There are many techniques submitted for demonstrating data in multi-tenant databases. These techniques accomplish some features, at the expense of other features. These techniques are still not enough, and not exceeding multi-tenant database challenges. Some of these techniques are examined in the following paragraphs.

Private Tables [13] technique permits each tenant to possess his private tables which can be different and expanded. The query is processed by database without using extra columns like “tenant_id” to distinguish and isolate tenant’s data, as submitted in this paper for the rest of the multi-tenant database schema mapping techniques [14].

The extension tables are simulated due to the development of the decomposed storage model that had been required described in [15]. This technique consists of creating segments of a table of n- columns into n 2-column tables. This technique of using Extension Tables is considered outstanding to the Private Tables approach described above. Multiple tenants can use the base tables also the extension tables.

The Universal Table as presented in [16] a sparse dataset consists of a huge number of columns. The Universal Table is a table that contains extra columns of the basics application schema columns, which permits tenants to store their needed columns side by side. Storing such a dataset can potentially lower its performance due to the high number of NULL values it may contain.

In Pivot Tables, each column of each row in a logical source table is converted to its record in the Pivot Table [3, 16]. The rows in the Pivot Table represented by four columns: ‘tenant_id’, ‘table_id’, ‘col’, and ‘row’ which are used to identify each row in the logical source table. The selected data types are elastic data types such as VARCHAR to increase the possibility of other different data types that can be cast to. For example, the "pivot_str" table can be used to store non-numerical data types. And "pivot_int" table which is used to store the numerical data type values.

The Chunk Table is another structure technique that is closest to the Pivot Table technique [4]. Except, it contains a group of data columns with a mixture of data types that are placed the column 'col' in the Pivot Table with the 'chunk' column in the Chunk Table. This technique splits the logical source table into groups of columns. Each group appointed to a unique Chunk-ID and converted into a suitable Chunk Table [17].

The Chunk Folding technique is derived from the Chunk Table technique and the Extension Tables as presented in [1, 13]. This schema vertically segmented virtual tables into chunks. Each segment has a 'chunk_id' where a chunk of columns is partitioned into segments of columns. The remaining parts are mapped into the extensions. But it can be implemented in open-source relational database products such as PostgreSQL.

The XML Table database extension technique is constructed by extensible markup language (XML) [18, 19]. This is accomplished by providing an XML data type or by adding the XML document into the database as a Large Object. Nerveless, this technique reduces the overall performance using XML files due to it consumes large space of the RAM.

The Elastic Extension Tables database technique which is the most recent technique of designing and structuring a multitenant database [13, 14]. EET is a database technique that uses the Shared DB–Shared Schema model. EET is consists of three main classes: common tenant tables (CTT), virtual extension tables (VET), and elastic extension tables (EET). The first class is the CTT that can be performed on any business domain database. The second class is the VET that grants tenants the ability to extend the existing business database. The third class is elastic extension tables (EET), which consists of eight tables that are used to construct VETs tables as presented in Table 1. EET exhibits a flexible method for constructing the tenants' database schema. With all the EET features and advantages, it has some drawbacks such as the access time to large numbers of records there will be a noticeable delay due to the usage only of the relational data system. The EET is lacking the tenant privileges to cooperating with all database programming features required nowadays such as triggers, routines, and procedures.

Table 1. Elastic extension tables

No.	Table Name	Table Features
1	"db_table"	Enable tenants to create virtual tables, give them unique names and determine the number of columns of each tenant table.
2	"table_column"	Allow tenants to create virtual columns of the previously created "db_table" tables.
3 4 5	"table_row" "table_row_blob" "table_row_clob"	The Row Elastic Extension Tables store records of the Virtual Extension Columns which tenant created in three separate distinguish tables. These tables are separated based on the data type of values. - "table_row" table is utilized for storing values like INT, FLOAT, DATE, TIMESTAMP, TIME, CHAR, VARCHAR and BOOLEAN. - "table_row_blob" table is utilized for storing large data values such as the BLOB values for virtual columns with the BLOB data type. - "table_row_clob" table is utilized for storing all CLOB values for virtual columns.
6	"table_relationship"	Utilizes tenants to create virtual relationships for their virtual tables.
7	"table_index"	Applied to add and create virtual indexes to virtual columns to improve the speed and the performance.
8	"table_primary_key_column"	Enable tenants to create virtual primary keys for the virtual extension columns which are stored in the "table_column" table.

All the previous techniques did not consider the heterogeneity of data. One of the major challenges is how to handle an increasing amount of heterogeneous IoT data with a variety of data types and data sources. Despite the popularity of relational databases, the scalability of the NoSQL database model and the document-centric data structure of NXD databases appear to be promising features for an efficient IoT system [20]. The results in [21] show that the NoSQL database is the best choice for query speed, whereas NXD is advantageous in terms of flexibility and extensibility, which are essential to cooperate with the characteristics of IoT data. The characteristics and limitations of the previously described techniques are explained in Table 2. In this paper, a new multi-tenant database schema NXD-E^{3}T is proposed that combines a new proposed E^{3}T schema and the Native XML Database to implement the database query more efficiently and reduce the query response time [22].

Table 2. Multi-tenancy techniques characteristics

No.	Multi-tenancy Database Design	Extra Columns	Large No. of Tenants	Divide Tables	NULL values	Overhead	Complexity Query	Schema Known in Advance	Large Space of the RAM	Noticeable delay	Programming Features	IoT Benchmark
1	Private Tables	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗
2	The extension tables	✓	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗
3	The Universal Table	✓	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗
4	Pivot Tables	✓	✓	✓	✗	✓	✓	✗	✗	✓	✗	✗
5	The Chunk Table	✓	✗	✓	✗	✓	✓	✗	✗	✓	✗	✗
6	Chunk Folding	✓	✗	✓	✗	✓	✓	✓	✗	✓	✗	✗
7	The XML Table	✓	✓	✗	✗	✗	✗	✗	✓	✓	✗	✗
8	The EET Technique	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✗

3. THE PROPOSED NATIVE ENHANCED ELASTIC EXTENSION TABLES DATABASE (NXD-E^{3}T)

This paper proposes a multi-tenant database NXD-E^{3}T schema to reduce the query response time included programming features with a simple and fixed number of basic tables. The contribution of this paper consists of two main partitions: First, allowing tenants to establish their full databases using the proposed enhanced elastic extension tables (E^{3}T) and even providing all programming features required. These features started from insert, update, and delete data to add triggers, routines, and procedures. Performing the E^{3}T schema provides availability as this design serves multiple tenants with their sub-users. Then, applying this new structure of database on IoT benchmark [23]. Second, it proposes the NXD-E^{3}T schema that structures data for sharing and support in relational databases, demonstrates the idea of executing as multi-tenant storage. These objectives are the motivation to propose an improved database design E^{3}T schema to support multiple levels of data isolation and performance for all tenants in the multi-tenant database.

3.1. The proposed enhanced elastic extension tables (E^{3}T) schema

The E^{3}T database schema proposes an enhanced design of multi-tenant database schema to eliminate the miss selecting of a large scale of heterogeneous IoT data from the database and permits any tenant to handle the database queries (insert, update, delete) correctly [24]. The E^{3}T schema handles the data as follows:

3.1.1. Storing the data based on tenant action

The proposed design adds three tables to the Elastic Extension Tables to ensure that the tenant has full programing control to their database. It provides three tables to solve three different cases for IoT systems [14].

a. "table_trigger" table, this table allows tenants to add triggers, it is associated with a specific table that activates when a specific event occurs for that table. This is needed if a tenant requires to trigger an action that activates when a specific event occurs [25].
b. "table_procedure" table, this table is used when it is required to perform in a single call to a procedure a repetitive task that requires checking, looping, multiple statements on the server with a return value that can be stored to start many another actions. This is needed if a tenant intends to perform a procedure as a result of specific data.
c. "table_routine" table, this table permits tenants to create routines for preventing the need to keep reissuing the individual statements but can refer to the stored routine instead while accessing the VET or CTT [26]. This is needed when a tenant attends to create a routine which essentially used to gather parameters and compare values following by single return value.

3.1.2. Storing the data based on data size and format

Data in IoT systems can be in several different formats, ranging from text and numbers to audio, pictures, and videos. The proposed E^{3}T database allows the user to store different data types in three different separated tables [27]`.

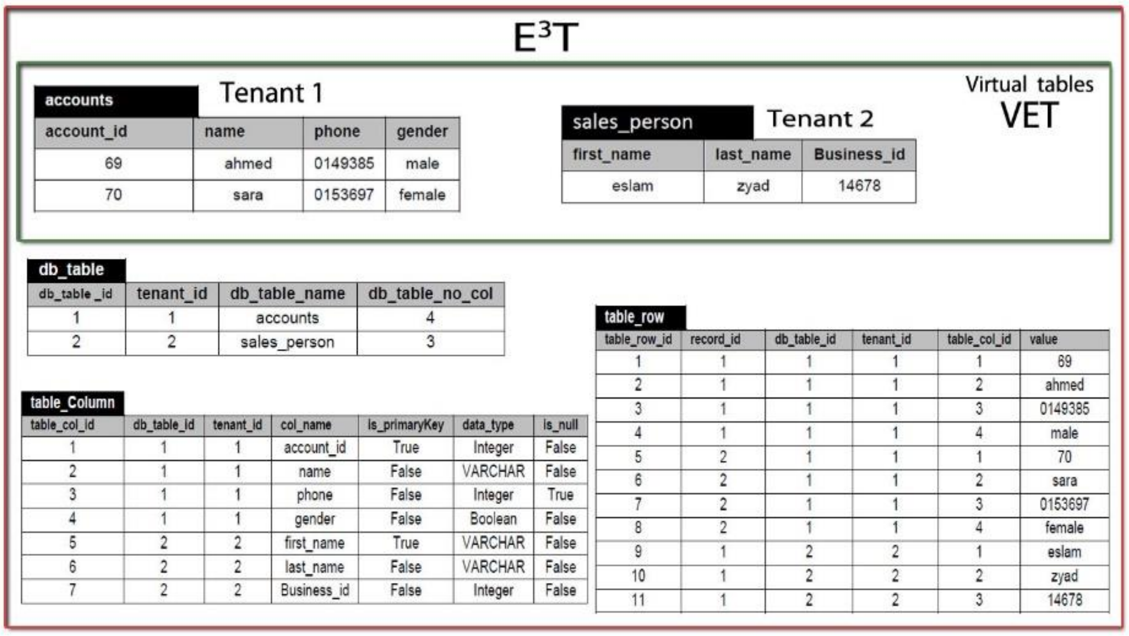
a. "table_row" table, for storing small data types values such as (CHAR, VARCHAR, INT, FLOAT, DATE, TIME).

b. "table_row_blob" table, for storing multimedia files. It stores the data in the form of a binary large object (BLOB) data type.
c. "table_row_clob" table, for storing image files. It stores the data in the form of a character large object (CLOB) data type.

3.1.3. Storing the data based on the query type

In the E^{3}T schema, the database tables are partitioned into two main parts: CTT, and E^{3}T tables.

a. The CTT tables that store metadata (i.e. data about database structure) are the physical tables that existed and shared between all tenants. These tables are created by each tenant administrators and the number of these tables is unlimited. The query that is based on data definition language (DDL) deals with the CTT tables.
b. The E^{3}T tables that store the tenant's data. These tables are responsible for the creation of the VET tables to establish the virtual data structure that each tenant has, which can be created to be convenient for each tenant requirements as shown in Figure 2. These tables are added to the previously EET tables to ensure that this schema is compatible with any tenant needs. The query that is based on data manipulation language (DML) deals with the E^{3}T tables.


E³T
Tenant 1
Tenant 2
Virtual tables
VET

accounts

account_id	name	phone	gender
69	ahmed	0149385	male
70	sara	0153697	female

sales_person

first_name	last_name	Business_id
eslam	zyad	14678

db_table

db_table _id	tenant_id	db_table_name	db_table_no_col
1	1	accounts	4
2	2	sales_person	3

table_Column

table_col_id	db_table_id	tenant_id	col_name	is_primaryKey	data_type	is_null
1	1	1	account_id	True	Integer	False
2	1	1	name	False	VARCHAR	False
3	1	1	phone	False	Integer	True
4	1	1	gender	False	Boolean	False
5	2	2	first_name	True	VARCHAR	False
6	2	2	last_name	False	VARCHAR	False
7	2	2	Business_id	False	Integer	False

table_row

table_row_id	record_id	db_table_id	tenant_id	table_col_id	value
1	1	1	1	1	69
2	1	1	1	2	ahmed
3	1	1	1	3	0149385
4	1	1	1	4	male
5	2	1	1	1	70
6	2	1	1	2	sara
7	2	1	1	3	0153697
8	2	1	1	4	female
9	1	2	2	1	eslam
10	1	2	2	2	zyad
11	1	2	2	3	14678

Figure 2. Enhanced elastic extension tables

The E^{3}T tables provide high scalability, low space overhead, and flexibility for the following reasons: The total number of tables from multiple tenants is a fixed number and not too large. Then the number of tables does not grow linearly with the number of tenants. Eliminate the existence of NULL values comparing other multitenancy techniques such as universal table technique as discussed before. Low storage needed as all the tenants are put into the same tables considering importance. It is more flexible because the cost of adding new or deleting old attributes is much smaller than other approaches.

3.2. The NXD-E^{3}T Database

In this section, the NXD-E^{3}T schema is proposed as a combination of the E^{3}T and Native XML databases. The Native XML database is used to minimize the amount of data placed in the memory and improve the performance of the E^{3}T by reducing the access time. In the beginning, the application program interface (API) gather the information data from the different interfaces and devices. The query access control checks the privilege of the tenant's query to access either CTT or VET. Then the NXD targets the selected tables to load it. and then, the queries are pushed to the Query optimizer which carries out the required function. Finally, the queries are passed to be access data and processed as shown in Figure 3.

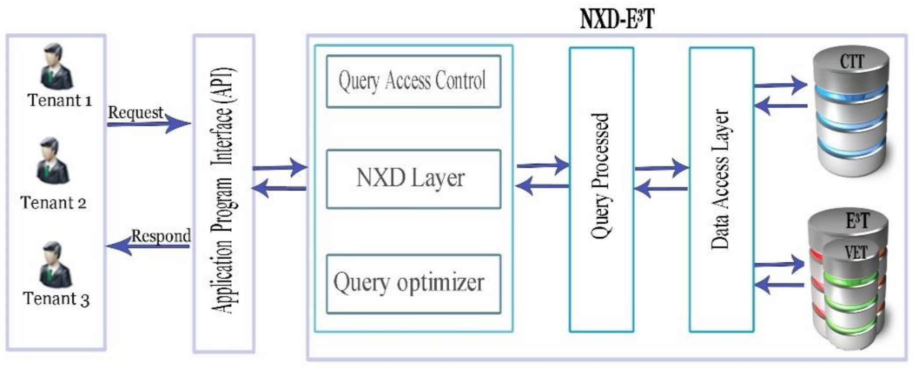


Figure 3. NXD-E^{3}T schema

When NXD data is loaded into relational databases, NXD recognizes and understands the queries need to be transformed into SQL queries in the relational data. The advantage of this technique is that it does not demand many modifications of the existing database engine. Merging the E^{3}T database with the NXD will provide the multi-tenancy databases to be relational with the virtual tables to each tenant as well as will increase the speed and decrease the response execution time of the multi-tenant database. Considering the NXD is a middle layer between the tenant requests and the query result, and the actual tables either CTT or VET, this layer consumes less memory. Then all query processing is passed into a relational query optimizer without any extra processing work. Using NXD-E^{3}T has two main objectives. First, query the database using a common query language. Second, support multiple tenants and sub-users roles.

4. RESULTS AND DISCUSSIONS

In this section, the performance of the proposed NXD-E^{3}T schema is evaluated. The proposed schema is compared against EET and E^{3}T on a heterogeneous IoT database [28]. Moreover, the proposed schema is compared against variant multi-tenant techniques; Private tables, Extension tables, Universal tables, Pivot tables, Chunk tables, Chunk folding, XML tables, EET, and E^{3}T, where this evaluation is implemented on a traditionally structured database [13, 14]. The proposed schema simulates a real multi-tenant scenario by passing query requests from many tenants concurrently and by affecting many numbers of rows and then evaluate the solutions by analyzing the execution time data captured during executing those experiments. The experiments were completed on four different query types which are select, insert, update, and delete to verify the E^{3}T performance.

4.1. Experimental data set

The experiment was implemented with Apache Version 2.4.18, PHP Version 7.4, and MySQL Version 8.0.4. This paper database structure is deployed in a server platform HP ProLiant ML350 Gen10 Server Tower. Table 3 shows the specifications of the used platform.

Table 3. Specifications of the used platform

Operating system	Windows Server 2019
Processor	Intel® Xeon® Scalable 4210 (10 core, 2.2 GHz, 13.75 MB, 85W)
Memory	32 GB RDIMM DR
Storage capacity per disk	4 x 8 TB

4.2. The NXD-E^{3}T implementation on unstructured IoT database

In this comparison, the proposed schema is compared against EET and E^{3}T on a heterogeneous IoT database in terms of the execution time. The results of this experiment are executed and calculated on 1, 10, 50,100, and 1000 rows [24]. Figure 4 shows the effect of increasing the number of records on the insertion operation. It shows that the average execution time of the NXD-E^{3}T is approximately 45% faster than the average execution time of the E^{3}T. Figure 5 shows the effect of this increase on the update operation. The result shows that the average of the execution time of the NXD-E^{3}T is 30% faster than the average execution time of E^{3}T. Another noticeable feature that this performance is taking a line function with the increase of the rows.

The Delete operation is shown in Figure 6. The result shows that the average of the execution time of the NXD-E^{3}T is almost the same as the average of the execution time of E^{3}T with small numbers of rows or even a little faster. But as the number of rows increases the execution time of NXD-E^{3}T becomes faster. The Selection operation is shown in Figure 7. The result shows that the execution time of the of NXD-E^{3}T is approximately the same as the average of the execution time E^{3}T even after the number of records increases the same time is taken in each method.

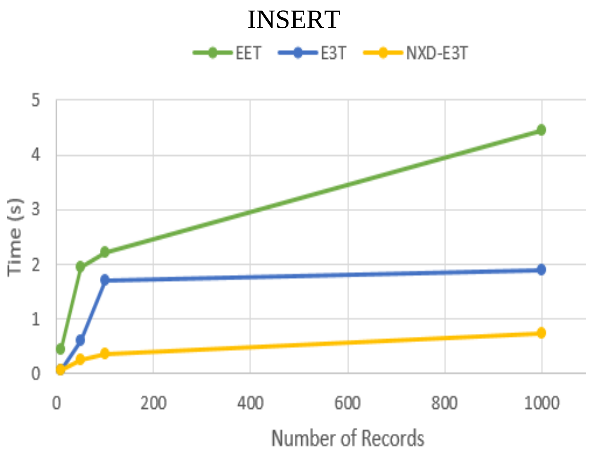


Figure 4. The execution time of INSERT query

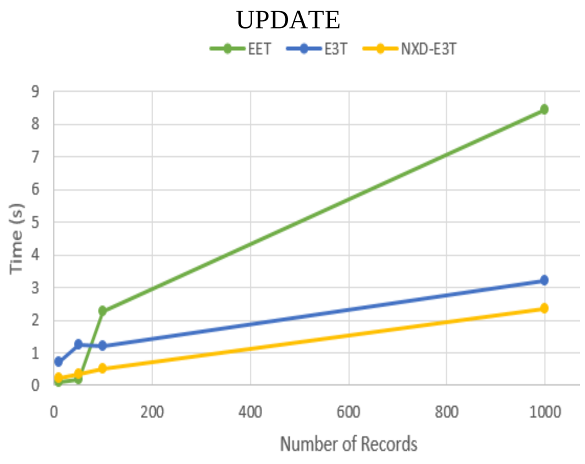


Figure 5. The execution time of UPDATE query

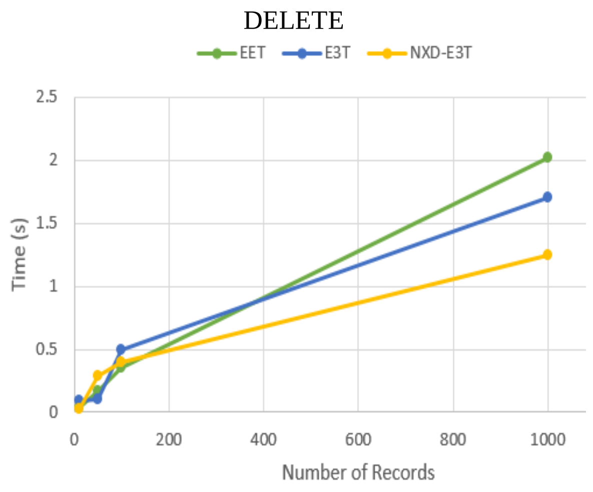


Figure 6. The execution time of DELETE query

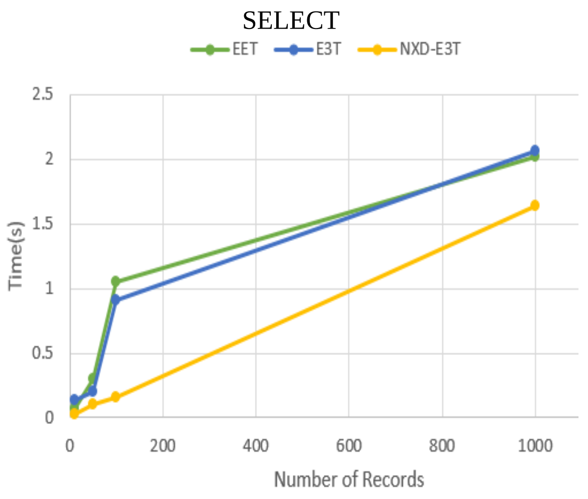


Figure 7. The execution time of SELECT query

4.3. The NXD-E^{3}T implementation on structured database

In this sub-section the comparison is implemented on a traditionally structured database, the proposed schema is compared against different multi-tenant techniques such as Private Tables, Extension Tables, Universal Table, PivotTables, Chunk Table, Chunk Folding, XML, and EET. The results of this experiment are executed and calculated when operating 1, 50, or 100 rows.

Inserting Query Experimental Result. This experiment is showing that the average execution time of the NXD-E^{3}T is approximately 29% faster than the average execution time of the EET Schema as presented in Figure 8. NXD-E^{3}T Schema is considered the fastest average execution time from all multi-tenant techniques in the insert query. Updating query experimental result. This experiment is showing that the average execution time of the NXD-E^{3}T 9% faster than the average execution time of the EET Schema in Figure 9. NXD-E^{3}T Schema is considered the fastest average execution time from all multi-tenant techniques in the update query.

Deleting query experimental result. This experiment is showing that the average execution time of the NXD-E^{3}T, is approximately 24% faster than the average execution time of the EET as presented in Figure 10. The delete and the insert query approximately have the same average as both having almost the same processing methods. The number executing the query is in ascending relationship with the number of rows affected.

Selecting Query Experimental Result. This experiment is showing that the average execution time of the NXD-E^{3}T is nearly 8% faster than the average execution time of the EET as in Figure 11. The SELECT query is the more complicated query in any multi-tenant technique as it requires more processing in JOIN operation and more understanding of the technique. The time of the query executing is in a linear relationship with the number of rows affected. With all these challenges the NXD-E^{3}T Schema has managed to perform this query with minimum response execution time in comparison to the multi-tenant techniques.

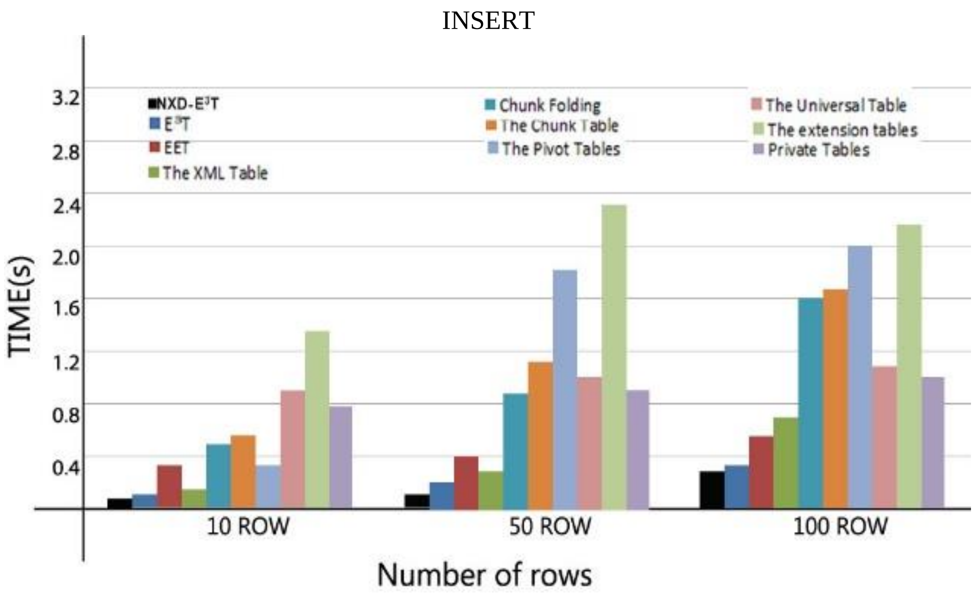


Figure 8. The execution time of INSERT query

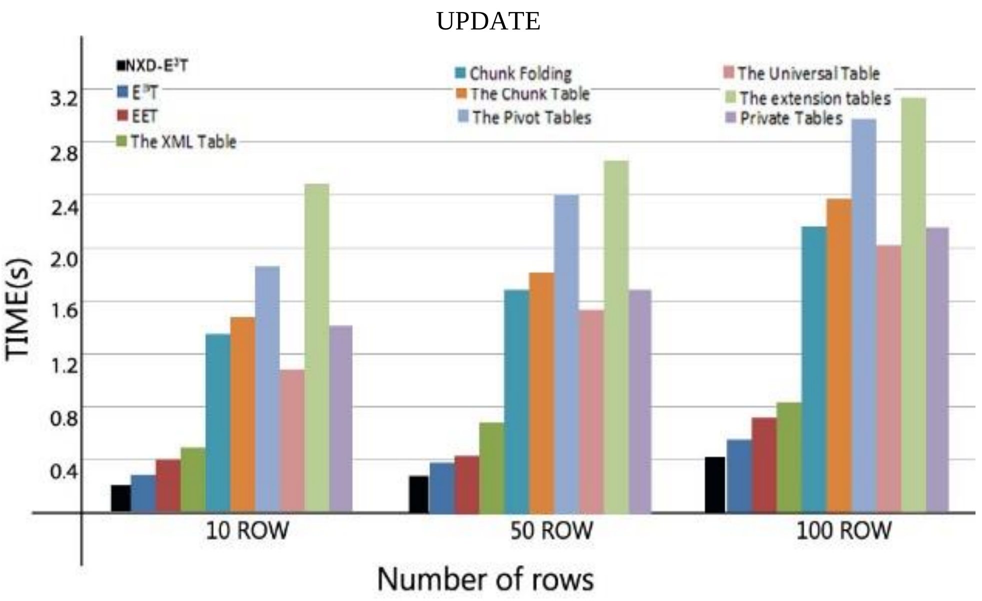


Figure 9. The execution time of UPDATE query

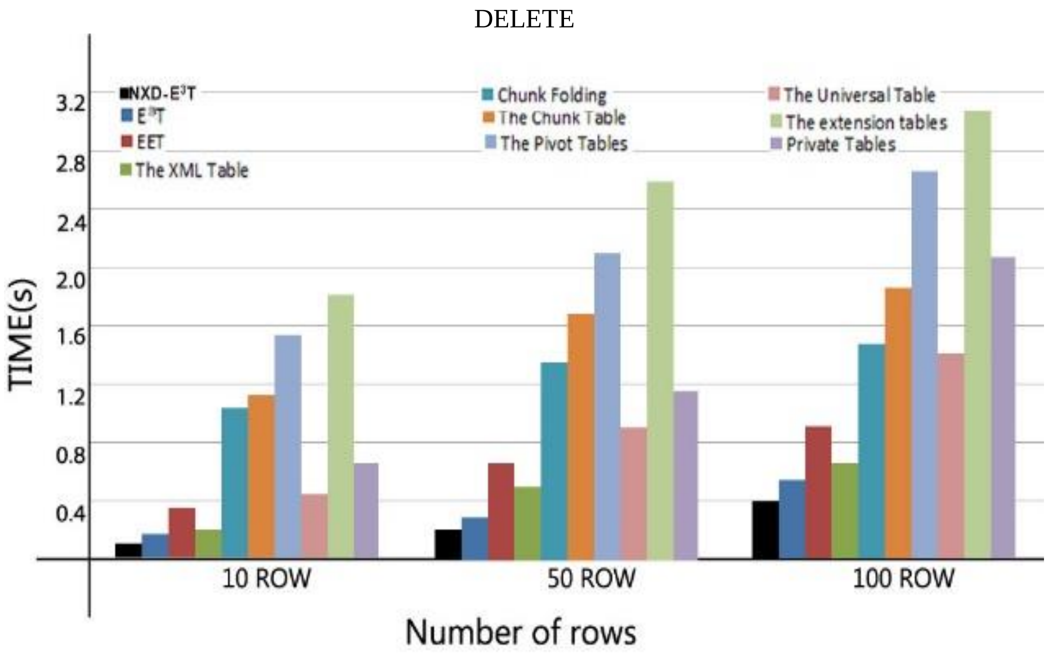


Figure 10. The execution time of DELETE query

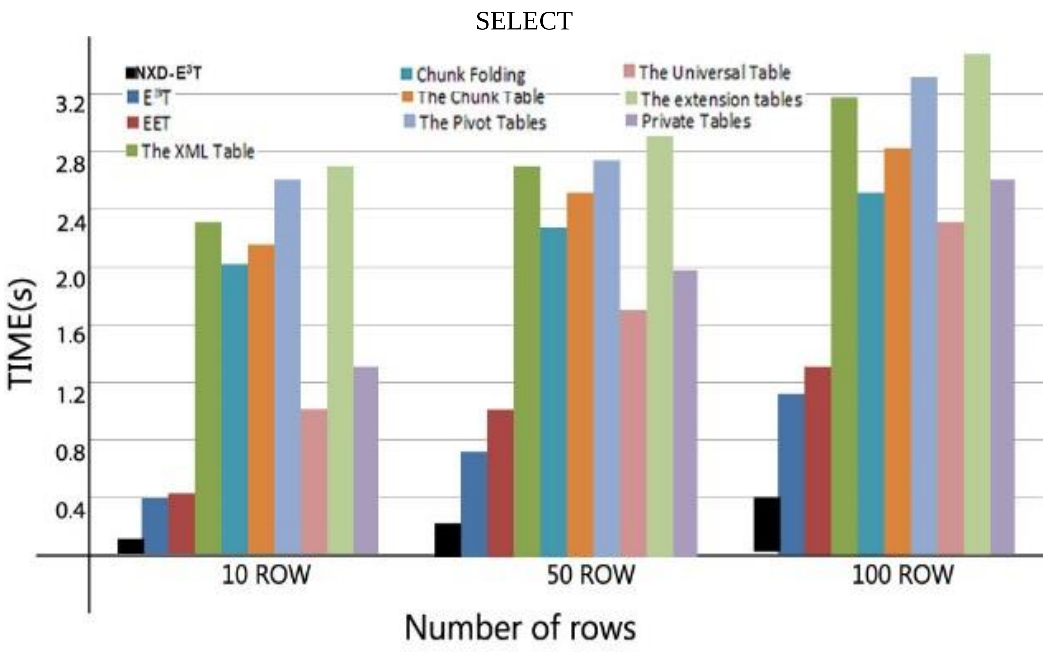


Figure 11. The execution time of SELECT query

5. CONCLUSION

In this paper, a native enhanced elastic extension tables (NXD-E3T) database is introduced as a multi-tenancy database system. The NXD-E^{3}T Schema permits tenants to create databases with different requirements needed for any business which achieve high scalability and all needed programming features with a high average response time. The NXD-E3T Schema enables tenants to own their elastic relational database schema. It can be implemented in the IoT systems efficiently. Future publications will focus on merging the NoSQL to the E3T as the NoSQL data storage aims to handle data in the IoT systems more accurately.

REFERENCES

[1] A. Jumagaliyev, “A Modeling Language for Multi-tenant Data Architecture Evolution in Cloud Applications,” PhD Thesis, Lancaster University, 2019.

[2] J. K. R. Sastry and M. T. Basu, “Securing multi-tenancy systems through multi DB instances and multiple databases on different physical servers,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 2, pp. 1385-1392, 2019.

[3] L. Li, et al., “Multi-tenant data authentication model for SaaS,” *Open Cybernetics and Systemics Journal*, vol. 8, no. 1, pp. 322-329, 2014.

[4] A. Rico, et al., “Extending multi-tenant architectures: a database model for a multi-target support in SaaS applications,” *Enterprise Information Systems*, vol. 10, no. 4, pp. 400-421, 2016.

[5] G. B. Pallavi and P. Jayarekha, “An efficient resource sharing technique for multi-tenant databases,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 3, pp. 3216-3226, 2020.

[6] T. A. M. Phan, et al., “Cloud databases for internet-of-things data,” *2014 IEEE International Conference on Internet of Things, (iThings 2014), 2014 IEEE Green Computing and Communications (GreenCom 2014), and IEEE Cyber, Physical and Social Computing*, pp. 117-124, 2014.

[7] K. K. Patel, et al., “Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & amp; Future Challenges,” *International Journal of Engineering Science and Computing*, vol. 6, no. 5, pp. 6122-6131, 2016.

[8] J. A. Stankovic, “Research directions for the internet of things,” *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 3-9, 2014.

[9] S. Cherrier, et al., “Multi-tenancy in decentralised IoT,” *IEEE World Forum on Internet of Things, (WF-IoT 2015)*, pp. 256-261, 2015.

[10] C. Li, et al., “Dynamic multi-objective optimized replica placement and migration strategies for SaaS applications in edge cloud,” *Future Generation Computer Systems*, vol. 100, pp. 921-937, 2019.

[11] E. A. Boytsov, “Designing and development of an imitation model of a multitenant database cluster,” *Automatic Control and Computer Sciences*, vol. 48, no. 7, pp. 437-444, 2014.

[12] S. K. Pippal and D. S. Kushwaha, “A simple, adaptable and efficient heterogeneous multi-tenant database architecture for ad hoc cloud,” *Journal of Cloud Computing: Advances, Systems and Application*, pp. 1-14, 2013.

[13] H. Yaish, et al., “A Proxy Service for Multi-tenant Elastic Extension Tables,” *Transactions on Large-Scale Data amd Knowledge Centered Systems XX*, pp. 1-33, 2015.

[14] H. Yaish, et al., “Evaluating the performance of multi-tenant Elastic Extension Tables,” *Procedia Computer Science*, vol. 29, pp. 614-626, 2014.

[15] A. I. Saleh, et al., “A Hybrid Multi-Tenant Database Schema for Multi- Level Quality of Service,” *International Journal of Advanced Computer Science and Applications*, vol. 5, no. 11, pp. 132-139, 2014.

[16] J. Kabbedijk, et al., “Multi-tenant Architecture Comparison,” *European Conference on Software Architecture*, pp. 202-209, 2014.

[17] R. P. Mahapatra and S. Khan, “A Survey Of Sql Injection Countermeasures,” *International Journal of Computer Science and Engineering Survey*, vol. 3, no. 3, pp. 55-74, 2014.

[18] A. Qtaish and K. Ahmad, “XAncestor: An efficient mapping approach for storing and querying XML documents in relational database using path-based technique,” *Knowledge-Based Systems*, vol. 114, pp. 167-192, 2016.

[19] Y. Cao, et al., “Support mechanisms for cloud configuration using XML filtering techniques: A case study in SaaS,” *Future Generation Computer Systems*, vol. 95, pp. 52-67, 2019.

[20] O. Osemwegie, et al., “Performance benchmarking of key-value store NoSQL databases,” *International Journal of Electrical and Computer Engineering*, vol. 8, no. 6, pp. 5333-5341, 2018.

[21] I. Fosic and K. Šolic, “Graph database approach for data storing, presentation and manipulation,” *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectron. (MIPRO 2019)*, pp. 1548-1552, 2019.

[22] S. Balamurugan and A. Ayyasamy, “Performance Evaluation of Native XML Database and XML Enabled Database,” *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 7, no. 5, pp. 182-191, 2017.

[23] M. G. Kibria, et al., “A framework to support data interoperability in web objects based IoT environments,” *International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 29-31, 2017.

[24] S. Wu, “A Method for Building Shared Massive Heterogeneous IoT Data Environment,” *2018 5th International Conference on Information Science and Control Engineering (ICISCE 2018)*, pp. 40-45, 2018.

[25] M. Winckler and K. Kuusinen, “Absolute Indirect Touch Interaction : Impact of Haptic Marks and Animated Visual,” *Int. Fed. Inf. Process. 2019 Publ. by Springer Nat. Switz. AG 2019 C. Bogdan al. HCSE 2018, LNCS 11262*, vol. 1, pp. 231-247, 2019.

[26] N. Dalčeković, et al., “Enabling the IoT paradigm through multi-tenancy supported by scalable data acquisition layer,” *Annals of Telecommunication*, vol. 72, no. 1-2, pp. 71-78, 2017.

[27] M. U. Kalay, “Database System Suggestions for the Internet of Things (IoT) Systems,” *Mugla Journal of Science and Technology*, pp. 46-52, 2018.

[28] M. Samaniego and R. Deters, “Supporting IoT Multi-Tenancy on Edge Devices,” *2016 IEEE International Conference on Internet of Thingsand IEEE Green Computing and Communications and IEEE Cyber, Physical and Social Computing and IEEE Smart Data*, pp. 66-73, 2017.

BIOGRAPHIES OF AUTHORS

Magy ElBanhawy is currently a Software Engineer at Damietta University From 2014. Received the B.Sc. degree in Computers and Control Systems Department, Faculty of Engineering, Mansoura in 2012, Pre-master in Electrical Engineering, Faculty of Engineering, Port Said University. She was Teacher Assistant at Nile Academy for Engineering in Mansoura from 2012 to 2014. Web Developer at Milestone in Mansoura Company From 2009 to 2012. Her research covers Multi-tenancy Database, Networking, Web developing and focuses on Cloud Computing.

Walaa Saber is an assistant professor in the Electrical Engineering Department, Port Said University, Egypt. She received her B.Sc. and M.Sc. degrees in Computer and Control Engineering, Suez Canal University in 2001 and 2008, respectively. Her Ph.D. degree in computer and control engineering is taken in 2014. Her research interests are in the area of computer networking, including cloud computing, clustering, and Internet of Things.

Fathy Amer is currently a Professor at 6th of October University. He received the B.Sc. in military science and the B.Sc. in Electrical and communication engineering from Military Technical College (MTC), Cairo, Egypt, in 1970. He received the M.Sc. in Electrical, mechatronics and communication Engineering (major area: electronics and communication(from Azhar University, Cairo, Egypt, in 1985. He received the Ph.D. in Philosophy in Computer Science from computer science department, Azhar University, Cairo, Egypt. Vice Dean for community Affairs and the Environment, Misr University for Science and Technology. His research interests are in the area of Local, expanding and international networking, Databases, multimedia, Internet of Things and Information technology.