CONTEXT STORAGE USING CLOUD-BASED MONGODB WITH RULE-BASED RETE ALGORITHM

Shivakumar. C[1], Dr. Siddhaling Urolagin [2], Dr.Poonam Ghuli [3]
[1]*Computer Science and Engineering,FET, Jain University,Ramanagar, INDIA*
k.shiva@jainuniversity.ac.in
[2]*Department of Computer Science, Birla Institute of Technology & Science, pilani, Dubai,*
siddhaling @ dubai.bits-pilani.ac.in
[3]*Computer Science and Engineering,RVCE,Bengaluru, INDIA*
poonamghuli@rvce.edu.in

Abstract: In this Context-aware computing era, everything is being automated and because of this, smart system's count been incrementing day by day. The smart system is all about context awareness, which is a synergy with the objects in the system. The result of the interaction between the users and the sensors is nothing but the repository of the vast amount of context data. Now the challenging task is to represent, store, and retrieve context data. So, in this research work, we have provided solutions to context storage. Since the data generated from the sensor network is dynamic, we have represented data using Context dimension tree, stored the data in cloud-based 'MongoDB', which is a NoSQL. It provides dynamic schema and reasoning data using If-Then rules with RETE algorithm. The Novel research work is the integration of NoSQL cloud-based MongoDB, rule-based RETE algorithm and CLIPS tool architecture. This integration helps us to represent, store, retrieve and derive inferences from the context data efficiently..

***Keywords*: NoSQL, CDT, ADLs, RETE, CLIPS**

1. Introduction

Structured data is being stored comfortably in Relational Database Management System (RDBMS) and retrieved whenever required efficiently [8]. The amount of data has rapidly increased from GB, TB, to the level of PB and EB [27]. So, when we deal with the unstructured and variety of data, need to use a different strategy. Different sensors like Passive infrared sensor (PIR), Magnetic, Flush, Pressure and Electric sensor generate their respective data. Global Positioning System (GPS) which provides geolocation and time, the latter generates the location and time data whenever the user changes his location, former generates the data whenever sensors sensing the user. So continuously sometimes and intermittently in others. The interesting thing is data generated in both the cases is unstructured most of the time and it poses a challenge to store it and retrieve efficiently. In such circumstances, it requires some special efforts to store and retrieve such context data. Extracting knowledge from unstructured data is time-consuming and expensive [1]. Even unstructured data affects the performance of query processing and increases the complexity to retrieve the data [2]. A computer requires annotations from the user or machine to manipulate unstructured data [3]. With high-performance NoSQL database stores unstructured data like email, documents, multimedia, etc., [4]. Carlo Strozzi conceived the NoSQL term in 1998 and allude to nonrelational databases, a term which was later reintroduced in 2009 by Eric Evans [5]. 'MongoDB', document-based the most popular NoSQL database it does not have pre-defined schema [6]. In MongoDB table called a collection, row as document and column as the field. It allows Join operations instead of reference. MongoDB execution time is slower than MySQL [7]. Many discern technology in document-oriented database as a natural descendent relational technology [8]. It creates the basic operations in MongoDB, Read, Update and Delete in short, we call it as CRUD operations. MongoDB supports master-slave replication. From the point of view of the CRUD operations, they are being not persuaded by the number of slaves' servers a master server has [9]. In connotation with the CAP theorem, MongoDB contemplated to be configurable, by convenience, as either AP (Availability, Partition) or CP (Consistency, Partition) because of Consistency, Availability and Partition tolerance all the three properties cannot be a document-based according to the CAP theorem [10]. NoSQL has the advantage of horizontal expansion, but for labyrinthine SQL requests. It cannot support them very well. For the Query based on KEY/VALUE and stupendous data storage requirements, NoSQL is a good choice [11]. When the enormous amount of data being generated because of the response of the sensors from the sensor network which is being stored in NoSQL database, it requires prodigious memory to store and retrieve. It is an arduous task and most of the time it becomes infeasible. So, solution for this is cloud storage [12]. Amazon cloud storage is one option which we can opt for. Data management perception is cloud computing give full availability where users can read and write data without ever being blocked [13].

The movement of a user from one place to another, people around users and devices are the different situations of a user which is termed as context. A context can be represented using the context dimension tree where each dimension is different roles of a user. The context consists of implicit information.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

The different contexts of a user are shown below [14,15,16]. If any system has to be context-aware it should be equipped with reasoning capabilities. Reasoning can be achieved by rule generation. In this regard, RETE algorithm is one of the prominent algorithms which computes condition and rule efficiently. This algorithm executes the action part only when it finds the matching rules [17]. It establishes data dependency between the different conditions of rules when the facts in the working memory are coordinated with all conditions of a rule. That will trigger the activities in the rule [18]. To make database active, we can use CLIPS tools that work on the RETE algorithm. CLIPS provide mechanisms for expert systems which comprises the rule-based language. It is a forward-chaining production system language where the reasoning proceeds from IF part of the rule to the THEN part. It bases the active component of the context database on clips [19]. In group decision making to select the best alternatives, we can use fuzzy preference relation [28].
The activity of daily living (ADLs) [25] dataset is been used throughout this research work in different instances. Firstly, the dataset is used to perform CRUD operations in the cloud-based MongoDB environment and finally used to generate rules using rule-based RETE algorithm CLIPS tool. ADLs dataset is a repository of different activities performed by the users in the smart environment to check the self-working capabilities of elderly people [25]. The rest of the paper comprises introducing different contexts through the context dimension tree in section 2. Context dimensions where been picturized. In section 3. The system architecture ADLs Rule-base NoSQL database cloud architecture which is an integration of MongoDB, ADLs dataset and cloud technology is been explained in section 4. The experimental results include CRUD operations, RETE algorithm and CLIPS rule-based tool. Results are been discussed in section 4 and followed by a conclusion
.

Context dimensions

The following tables describe the context dimensions and their descriptions. Table 1 depicts the various context dimensions from the ADLs dataset. It has infused the generic dimensions from the context dimension tree and their values in the table. Table 2 reveals generic context dimensions and their descriptions related to the dataset used in the current research work.

Table 1 Context Dimensions

SL. No	Context Dimensions	Context dimension values
1	Role	User1, User2
2	Interest Topic	Breakfast, Lunch, Dinner, Leaving, Toileting, Showering, Sleeping, Snack, Grooming.
3	Situation	Routine
4	Time	Range
5	Interface	Sensors (PIR, Magnetic, Flush, Pressure, Electric)
6	Place	Entrance, Living, Bedroom, Bathroom, Kitchen
7	Location	Bed, Cabinet, Basin, Toilet, shower, Fridge, Cupboard.

Table2 Context Dimensions Descriptions

SL. No	Context dimensions	Description
1	Role	In ADLs application, dataset user1 and user2 are the dominant entities they have captured whose activities for 35 total days to monitor their health status.
2	Interest topic	The different activities performed by the two users during their stay in the living room.
3	Situation	This means whether the activities are being held continuously, i.e. routine activities or occasionally taken part. In our application, it is a routine activity.
4	Time	Time taken for particular activities is absolute or relative. In this paper, we have considered absolute time. They are start time and end time of different activities.
5	Interface	Through which information being collected either human beings or machines. Case information collected with the help of sensors.
6	Place	This is the extra dimension that we have added to the CDT. There is a difference between place and location, Place comprises many locations.
7	Location:	The point at which various activities happened is being called location.

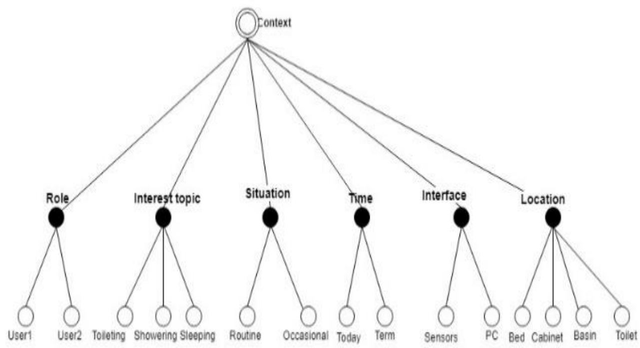

Fig.1 Context dimension tree

Context Dimension Tree (CDT) is a generic context data model as shown in Fig1. It is used to represent different contexts of the user. It comprises two types of nodes: one is black nodes and another is white nodes. Black nodes typify context dimensions and white nodes exemplify context values. CDT has one double circled node; we call it as a root node of the tree. Each leaf of the tree is a value node, and it facets many parameters. White squares represent parameters [20]. So, CDT helps to characterize the different instances of users under various circumstances.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

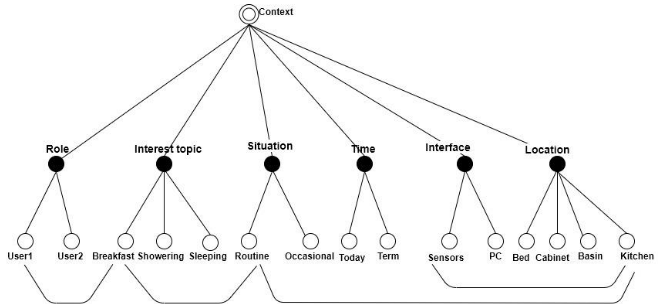

Fig.2 User1's context

The context dimension tree in Fig 2 depicts one context of User1. It shows that User1 is preparing breakfast in the kitchen routinely. Interface through which they captured it is Sensors. This context diagram covers the User's Role, Interest topic, Situation, Location, and Interface. If the location is fridge OR Cupboard AND place is kitchen, then Activity breakfast is possible at situation S as shown in equation 1.

Location (A, S) Λ Place (B, S) V Place (B1, S) → Poss (Activity ((A, B, C), S)
(1)

Whereas,
A =Fridge (Location)
B = Kitchen (Place)
B1 = Kitchen (Cupboard)
C = Breakfast (Activity)
S = Situation

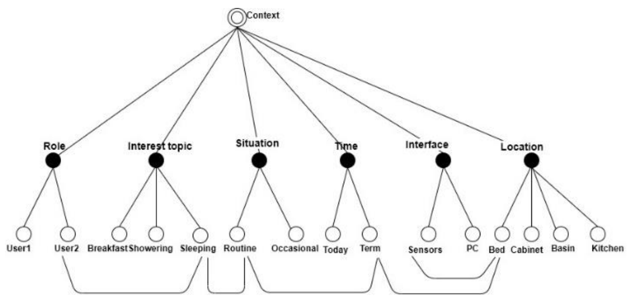

Fig.3 User2's context

The context dimension tree in **Fig 3** reveals one context of User2. The diagram represents that the User2 is sleeping in the bedroom occasionally, and they captured this through the interface sensors. It uses almost all dimensions of context dimension tree, which are being described above. If the location is bed AND place is bedroom, then Activity 'sleeping' is possible in some situation, S as shown in equation 2.

Location (X, S) Λ Place (Y, S) → Poss (Activity ((X, Y, Z), S
(2)
X=bed
Y= bedroom
Z= sleeping
S = Situation

System Architecture

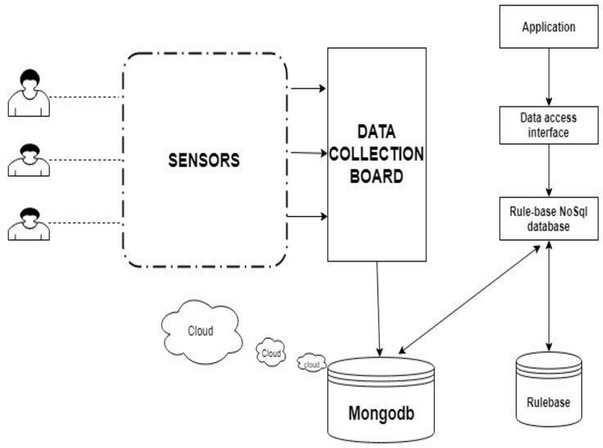

Fig.4 ADLs Rule-base NoSQL database cloud architecture

When the sensor observes any object or its activities at any instance, we will store it in the data collection board. The data was as binary numbers 0's and 1's. Here 0 is absent and 1 is present. We have confirmed the presence or absence of the user at a particular place. A dataset is being constructed by the authors which is ADLs [25] i.e., Activities of a daily living dataset as shown in Fig 4. This dataset comprises User_id, Start_time, and End_time of activity, LocationPlace, and type of sensors used. Since it is unstructured data, we store it in different dimensions and schema is being changed every time when the sensor senses the different objects. In such a context, NoSQL database is appropriate because it has a dynamic schema which changes according to the senses data. So, we store this unstructured data in MongoDB, which the data is being stored on the google cloud. By integrating the MongoDB and rule-based tool CLIPS [19] Context database can realize the situations. Using the data, we can generate rules.

MongoDB

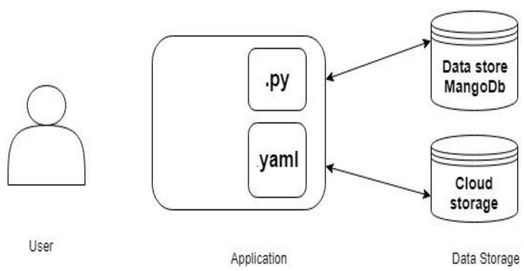

Fig.5.MongoDB cloud storage framework

Users can interact through an application, which is the interface between the user and the database. The application uses Cloud Storage to store binary data, pictures in this case, while continuously using NoSQL database (MongoDB), as shown in Fig 5.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

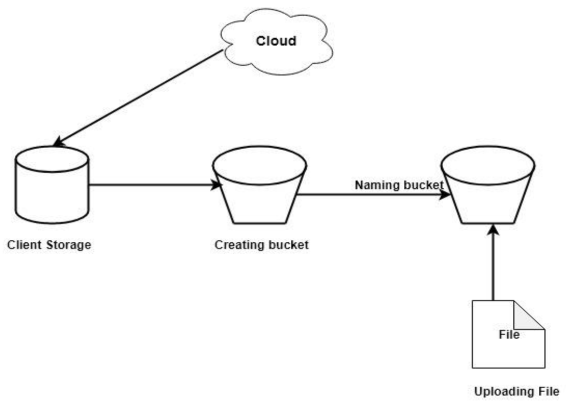

Fig 6: creating cloud buckets

From Google cloud storage, need to get client storage and then the bucket is being created and named consequently thereafter files are being uploaded to the bucket. It is as shown in Fig.6.

Document oriented model

We relate document-oriented databases to NoSQL databases, which are schema-less. Collections in this database comprising unique documents which are being used to store heterogeneous, unstructured and complex structures documents [21]. The document-oriented data model for the database is as shown in the Fig.7

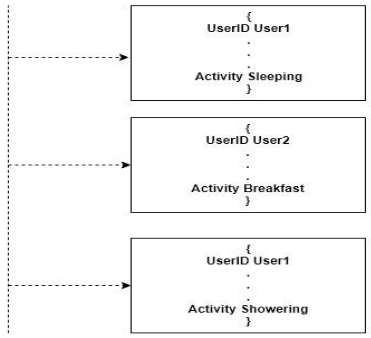

Fig.7 Document oriented model

RETE Algorithm design

Working memory is a short-term memory. Working memory elements are facts (data) and rules. Rules are long-term memory. Conflict set is just a collection of rule matching data combos. RETE algorithm takes changes in the working memory and it feeds them into what we call a RETE net where the compilation of rules takes place. RETE net is just a representation of rules in a network format. It gives changes in the conflict set [22].

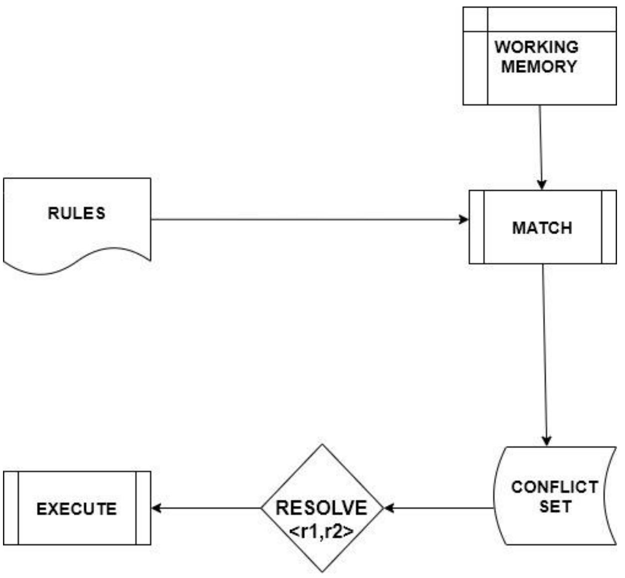

Fig.8 Rete algorithm framework

2. EXPERIMENTAL Setup And RESULTS

The below-mentioned dataset captured using a distinct set of sensors at different locations inside the house environment for 35 days of two users to determine their self-working capabilities independently [25]. The dataset reveals the different activities performed by the users. **Table.3** Activity of daily living (named as ADLs) comprises User_id, start_time, end_time, and activity. **Table.4** named as Activity of daily living Sensor (ADLsSensor) comprise User_id start_time, end_time, location, place, and type.

Table.3 Adls Activity Labels For User

UserID	**Start_time**	**End _time**	**Activity**
User1	2011-11-28 02:27:59	2011-11-28 10:18:11	Sleeping
User1	2011-11-28 10:21:24	2011-11-28 10:23:36	Toileting
User2	2011-11-28 10:34:23	2011-11-28 10:34:41	Breakfast
User2	2011-11-28 10:34:44	2011-11-28 10:37:17	Breakfast

Table.4 Adlssensor Events For User

UserID	Start_time	End_time	Location	Place	Type
User1	2011-11-28 02:27:59	2011-11-28 10:18:11	Bed	Bedroom	Pressure
User1	2011-11-28 10:21:24	2011-11-28 10:23:36	Cabinet	Bathroom	Magnetic
User2	2011-11-28 10:34:23	2011-11-28 10:34:41	Fridge	Magnetic	Kitchen

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

User2	2011-11-28 10:34:44	2011-11-28 10:37:17	Cupboard	Magnetic	Kitchen

A. NOSQL CRUD operations

MongoDB uses BSON for the binary encoding of JSON like documents. It uses to store documents in the collection. BSON format keeps the elements in the order field name, data type, and value. The CRUD operations are Create, Read, Update and Delete. Create operation creates the documents in the collection equal to Insert in SQL. Read operation is used to retrieve the data here we used to find () or findOne () operation this equal to count and aggregate operations in SQL. An update is used to change the document if data is inserted previously. To remove the document from the collection, we use delete operation such as remove () this is like delete operation in SQL [23].

Table 5. Adls Collection And Adlssensor Collection

ADLS Collection	**ADLsSensor collection**
Create Operation	**Create Operation**
db. ADLS.insertOne({UserID: “User1”, Start_time: “2011-11-28 02:27:59”, End_time: “2011-11-28 10:18:11”, Activity: “Sleeping”})	db. ADLsSensor.insertOne ({UserID: “User1”, Start_time: “2011-11-28 02:27:59”, End_time: “2011-11-28 10:18:11”, Location: “Bed”, Place: “Bedroom”, Type: “Pressure”});
db. ADLS.insertOne ({UserID: “User1”, Start_time: “2011-11-28 10:21:24”, End_time: “2011-11-28 10:23:36”, Activity: “Toileting”});	db. ADLsSensor.insertOne ({UserID: “User1”, Start_time: “2011-11-28 10:21:24”, End_time: “2011-11-28 10:23:36”, Location: “Cabinet”, Place: “Bathroom”, Type: “Magnetic”});
db. ADLS.insertOne ({UserID: “User2”, Start_time: “2012-11-12 01:54:00”, End_time: “2012-11-12 10:53:59”, Activity: “Sleeping”});	db. ADLsSensor.insertOne ({UserID: “User2”, Start_time: “2012-11-12 01:54:00”, End_time: “2012-11-12 09:31:59”, Location: “Bed”, Place: “Bedroom”, Type: “Pressure”});
db. ADLS. insertOne ({UserID: “User2”, Start_time: “2012-11-12 10:53:00”, End_time: “2012-11-12 10:53:59”, Activity: “Toileting”});	db. ADLsSensor.insertOne ({UserID: “User2”, Start_time: “2012-11-12 10:53:00”, End_time: “2012-11-12 10:53:59”, Location: “Cabinet”, Place: “Bathroom”, Type: “Magnetic”});
Read operation	**Update operation**
db. ADLsSensor.find({UserID: “User1”, Start_time: “2011-11-28 10:21:24”, End_time: “2011-11-28 10:23:36”, Location: “Cabinet”, Place: “Bathroom”, Type: “Magnetic”});	db. ADLS.update({‘Activity’: ‘Toileting’}, {$set: {‘Activity’: ‘Sleeping’}});
Delete operation	
db. ADLS.remove ({‘Activity’: ’Sleeping’});	

B. CLIPS rule-based

To make data manipulation more effective and in an organized manner, we can rely on CLIPS rule-based [24]. Where facts with two relation names ADLS and ADLsSensor being created, the different slots are UserId, Start_time, End_time, Activity, Place, Location, type. A cozy def template construct is used to group the facts with the same relation name which contains common information. To insert fact, assert we use command, to display its facts we use command and to check the changing state of the fact watch fact.

Table.6 Adls Fact Generation

ADLS	**ADLS1 Sensor**
(deftemplate ADLS (slot UserID) (slot Start_time) (slot End_time) (slot Activity)	(def template ADLsSensor (slot UserID) (slot Start_time) (slot End_time) (slot Location) (slot Type) (slot place)
(assert (ADLs1 (UserID User1) (Start_time 2011-11-28 02:27:59) (End_time 2011-11-28 10:18:11) (Activity sleeping))) **(Fact-1)**	**(assert (ADLs1Sensor (UserID User1)** (Start_time 2011-11-28 02:27:59) (End_time 2011-11-28 10:18:11) (Location Bed) (Type Pressure) (Place Bedroom) **<Fact-1>**

CLIPS bestir to bout the patterns of rules against facts from the facts list. If it matches, the rule is being activated. Then it will put on the agenda, Agenda is the congregation of activations.

Table.7 Defining Rules

(defrule AdLS_Sensor (User1 is in Bedroom) => (assert (User1 is sleeping)) (watch facts) (watch activations) (assert (User1 is in Bedroom)) = => f-1 (User1 is in Bedroom) = => Activation 0 AdLs1_Sensor: f-1 <Fact-1>	(defrule ADLS_sensor (User1 is in Bathroom) => (assert (User1 is toileting))) (watch facts) (watch activations) (assert (User1 is in Bathroom)) = => f-1 (User1 is in Bathroom) = => Activation) 0 ADLS_Sensor: f-1 <Fact-1>
(agenda) ADLS_Sensor: f-1 For a total of 1 activation (clear)	**(agenda)** ADLS_Sensor: f-1 For a total of 1 activation (clear)

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

When rules are been judged against facts or data in application scenario the process of evaluation and ordering the statements will be costly. This inference approach saves both.

C. Creating the rete network

RETE network is the heart of the RETE algorithm. It has nodes that consist of many objects which satisfy the specific or associated conditions. This algorithm works on facts. The first phase of the RETE network is discrimination tree where it starts with alpha nodes connected to classes [26]. All instances of a given class will be listed in alpha node. The network can be constructed as below.

1. First, alpha nodes are created for each class as shown in Fig.9

Fig 9: Alpha Nodes

2. Conditions are then appended as shown in Fig.10

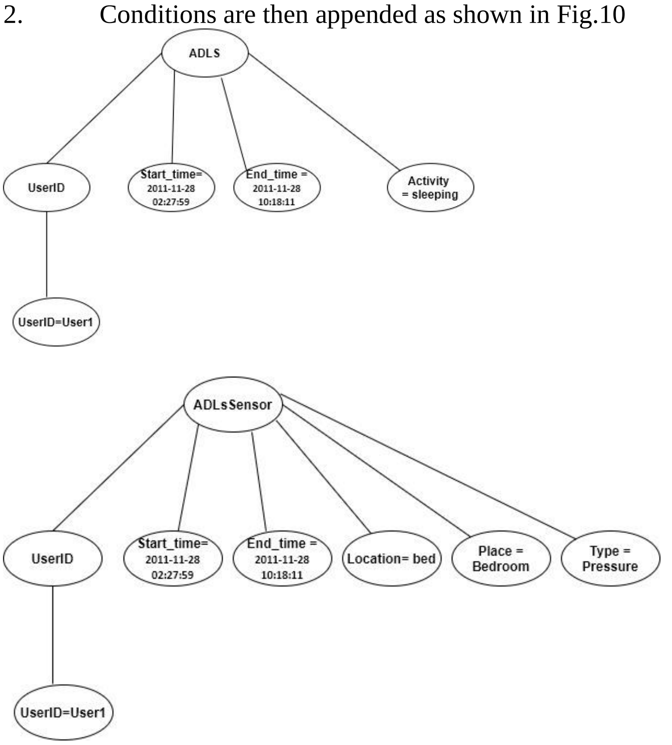

Fig 10: Rete Network

3. Finally, the nodes are connected across classes.
4. The path eventually ends with the action part of the rules.

D. RETE Cycle: Evaluate

The evaluation phase comprises running the data through the RETE network to identify the applicable rules. It satisfies if conditions with some rules, then they are active on the agenda. The agenda comprises a list of rules and objects which are being executed together that are responsible for the conditions to be true.

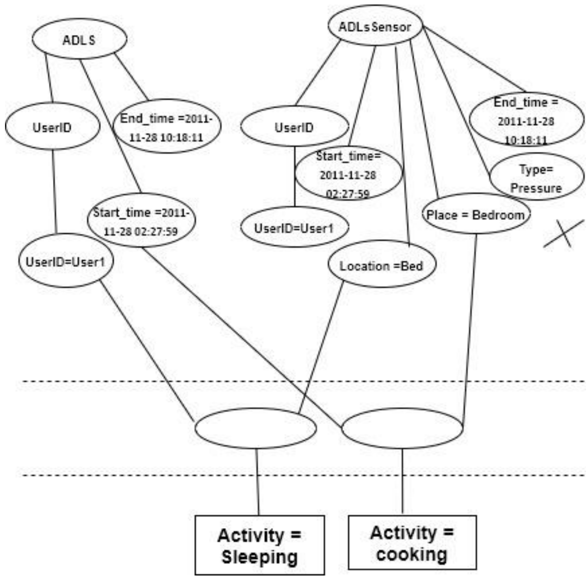

Fig.11 RETE evaluation

3. Results

A. Input

UserID	Start_time	End _time	Activity
User1	2011-11-28 02:27:59	2011-11-28 10:18:11	Sleeping
User1	2011-11-28 10:21:24	2011-11-28 10:23:36	Toileting
User2	2011-11-28 10:34:23	2011-11-28 10:34:41	Breakfast
User2	2011-11-28 10:34:44	2011-11-28 10:37:17	Breakfast

Assured	Start_time	End_time	Location	Place	Type
User1	2011-11-28 02:27:59	2011-11-28 10:18:11	Bed	Bedroom	Pressure
User1	2011-11-28 10:21:24	2011-11-28 10:23:36	Cabinet	Bathroom	Magnetic
User2	2011-11-28 10:34:23	2011-11-28 10:34:41	Fridge	Magnetic	Kitchen
User2	2011-11-28 10:34:44	2011-11-28 10:37:17	Cupboard	Magnetic	Kitchen

The ADLs data runs throughout and from the RETE evaluation, we can conclude that the rules satisfied being accepted by the algorithm and rest are being rejected as shown above. The active and inactive rules on the agenda are as shown below in the Fig.12(a) and Fig.12(b). User can sleep on the bed will be true and the user can cook in the place

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

bedroom will be false. So, according to the system architecture presented after rule generation data and details are being stored in the cloud

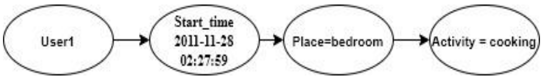

Fig.12(a) Inactive rule.

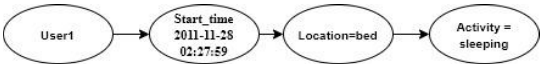

Fig.12(b) active rule

B. *Output*

we can generate the rules using RETE algorithm as shown below:

(Rule user activities

<user ^ name (n) >

<Activity1^2011-11-28 10:34:23 ^user2 ^Kitchen ^ (breakfast) >

<Activity2^2011-11-28 02:27:59 ^user1^ bedroom^ (sleeping) >

→ (Activities<users^ Activities<breakfast + sleeping>))

The rule has three patterns one says that of User name who we will call as n and then for breakfast n should have some activities let's call it a1. For sleeping that student should have some activity a2. Then we want to produce a record which says that the total activities of users n are a1 plus a2. So, it is the job of the beta nodes to gather these three working memory elements so somewhere just visualize that we have this Rete network which has other rules of other kinds for example for grading there might be rules and all kinds of things might be there. And there we yield these three pieces of data, that there is a user called user1, then user1 accomplishes, let's say, breakfast. That's one data record. Then user2 accomplishes a second activity that is a second data record, let's say. The instant we get these three pieces of data they will flow down the network and then this rule called user activities will fire so we will have once we get this data, one element added to the conflict set which says that the rule user activities is ready to fire and it has got these three pieces of data it is identical. And like that, if there are 50 other activities, there may be 50 other rules and then based on the problem-solving tactic we will take one using resolve and so on and so forth.

C. *Query Outputs*

Using context data, we have executed CRUD operations from cloud-based MongoDB which belongs to NoSQL Database, and we have shown results as below:

db.ADLS.insertOne({ UserID: "User1", start_time: "2011-11-2802:27:59", End_time:"2011-11-2810:18:11", Activity: "Sleeping"})	{"acknowledged": true, "insertedId": ObjectId("5f704db86fbfbc92fe225a9d") }

db.ADLS.insertOne({UserID: "User1", Start_time: "2011-11-28 10:21:24", End_time: "2011-11-28 10:23:36", Activity: "Toileying"}) }	{"acknowledged" : true, "insertedId" : ObjectId("5f704f656fbfbc92fe225a9e") }

db.ADLsSensor.insertOne({UserID: "User1", Start_time:"2011-11-28 02:27:59", End_time: "2011-11-28 10:18:11", Location: "Bed", Place: "Bedroom", Type: "Pressure"})	{"acknowledged": true, "insertedId": ObjectId("5f70509a6fbfbc92fe225a9f") }

db.ADLsSensor.insertOne({UserID: "User1",Start_time: "2011-11-28 10:21:24", End_time: "2011-11-28 10:23:36", Location:"Cabinet", Place:"Bathroom", Type:"Magnetic"})	{ "acknowledged" : true, "insertedId" ObjectId("5f7051e96fbfbc92fe225aa0") }

The graph in Fig.13(a), Fig.13(b) outlines active rule on agenda location by type where sensors used at different location and active rule on the agenda of activity where different activities are being performed by the users.

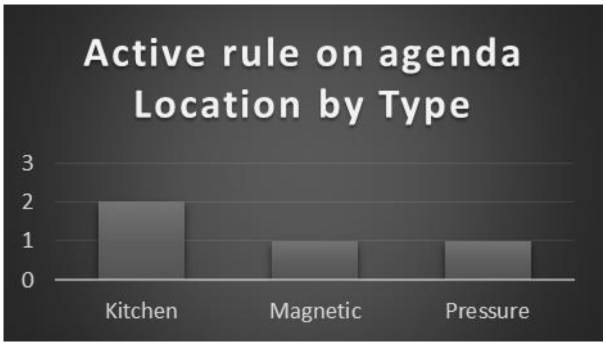

Fig.13(a) Active rule on agenda location by type

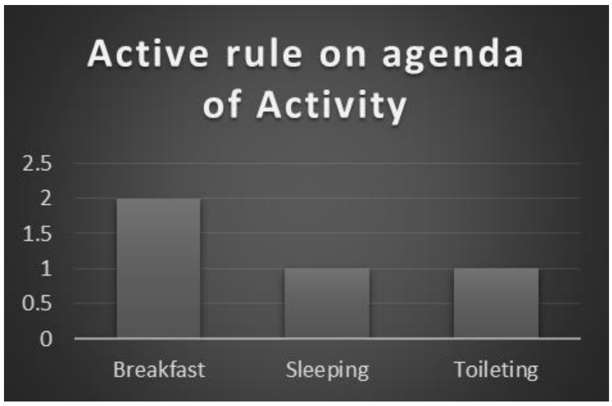

Fig.13(b) Active rule on agenda location by type

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

Different contexts of the user are nothing but rules as an input for RETE algorithm. It will retrieve it from cloud-based MongoDB and predict the particular context accurately. We consider the user actions and the locations to generate the different rules using the CLIP tool, which based on the RETE algorithm.

4. Conclusion

When a user moves from one place to another place and so on, his or her activities are being noted down manually and can create a database. But when the same activity is being observed by the different sensors, it is difficult to represent, store and to retrieve using normal database concepts which is a static database and doesn't have reasoning capabilities. So, we have represented sensor sensed data using context dimension tree in our research work. Context data storage and retrieval achieved using CRUD operations from cloud-based MongoDB which belong to NoSQL Database. Since data sensed is dynamic with the help of rule-based RETE, algorithm instances are being reasoned to make the system context-aware. This research work has achieved an effective integration of NoSQL, Cloud technology and rule-based concepts. This helps us to draw inferences from the running dataset.

References

[1] Rusu, O., Halcu, I., Grigoriu, O., Neculoiu, G., Sandulescu, V., Marinescu, M., & Marinescu, V. (2013, January). Converting unstructured and semi-structured data into knowledge. In 2013 11th RoEduNet International Conference (pp. 1-4). IEEE.

[2] Yafooz, W. M., Abidin, S. Z., Omar, N., & Idrus, Z. (2013, December). Managing unstructured data in relational databases. In 2013 IEEE Conference on Systems, Process & Control (ICSPC) (pp. 198-203). IEEE.

[3] Li, W., & Lang, B. (2010). A tetrahedral data model for unstructured data management. Science China Information Sciences, 53(8), 1497-1510.

[4] Kumar, J., & Garg, V. (2017, October). Security analysis of unstructured data in NOSQL MongoDB database. In 2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN) (pp. 300-305). IEEE.

[5] Hoberman, S. (2014). Data Modeling for MongoDB: Building Well-Designed and Supportable MongoDB Databases. Technics Publications.

[6] T. tean, Bazele de date NoSQL – o analiza, comparativa, To Day Software Magazine, number 10 [Online]. Available: http://www.todaysoftmag.ro/article/304/bazele-de-date-nosql-o-analizacomparativa, acceseed oct. 2014.

[7] Győrödi, C., Győrödi, R., Pecherle, G., & Olah, A. (2015, June). A comparative study: MongoDB vs. MySQL. In 2015 13th International Conference on Engineering of Modern Electric Systems (EMES) (pp. 1-6). IEEE.

[8] Băzăr, C., & Iosif, C. S. (2014). The transition from rdbms to nosql. a comparative analysis of three popular non-relational solutions: Cassandra, mongodb and couchbase. Database Systems Journal, 5(2), 49-59.

[9] Truica, C. O., Boicea, A., & Trifan, I. (2013, August). CRUD operations in MongoDB. In 2013 International Conference on Advanced Computer Science and Electronics Information (ICACSEI 2013). Atlantis Press.

[10] Ameri, P., Grabowski, U., Meyer, J., & Streit, A. (2014, September). On the application and performance of MongoDB for climate satellite data. In 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications (pp. 652-659). IEEE.

[11] Khan, S., & Mane, V. (2013). SQL support over MongoDB using metadata. International Journal of Scientific and Research Publications, 3(10), 1-5.

[12] Han, J., Song, M., & Song, J. (2011, May). A novel solution of distributed memory nosql database for cloud computing. In 2011 10th IEEE/ACIS International Conference on Computer and Information Science (pp. 351-355). IEEE.

[13] Chandra, D. G., Prakash, R., & Lamdharia, S. (2012, November). A study on cloud database. In 2012 Fourth International Conference on Computational Intelligence and Communication Networks (pp. 513-519). IEEE.

[14] Shivakumar.C, Dr. Siddhaling Urolagin," Context Data Representation using Context-Aware. Model in Ubiquitous Computing Scenario", May 2017, International Journal of Innovations & Advancement in Computer Science, ISSN 2347 – 8616 Volume 6, Issue 5

[15] Bolchini, C., Curino, C. A., Quintarelli, E., Schreiber, F. A., & Tanca, L. (2009). Context information for knowledge reshaping. International Journal of Web Engineering and Technology, 5(1), 88-103.

[16] Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., & Steggles, P. (1999, September). Towards a better understanding of context and context-awareness. In International symposium on handheld and ubiquitous computing (pp. 304-307). Springer, Berlin, Heidelberg.

[17] Schmidt, K. U., Stühmer, R., & Stojanovic, L. (2008, September). Blending complex event

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

processing with the rete algorithm. In Proceedings of iCEP2008: 1st International Workshop on Complex Event Processing for the Future Internet (Vol. 412).

[18] Xiao, D., & Zhong, X. (2010, June). Improving Rete algorithm to enhance performance of rule engine systems. In 2010 International Conference On Computer Design and Applications (Vol. 3, pp. V3-572). IEEE.

[19] Dong, Y., Chee, C. F. Y., He, Y., & Goh, A. (1997). Active database support for STEP/EXPRESS models. Journal of Intelligent Manufacturing, 8(4), 251-261.

[20] Chang, S. K., Yung, D., Colace, F., Greco, L., Lemma, S., & Lombardi, M. (2015). An adaptive contextual recommender system: a slow intelligence perspective. 27th Software Engineering and Knowledge Engineering (SEKE).

[21] Chevalier, M., El Malki, M., Kopliku, A., Teste, O., & Tournier, R. (2015, September). Implementation of multidimensional databases with document-oriented NoSQL. In International Conference on Big Data Analytics and Knowledge Discovery (pp. 379-390). Springer, Cham.

[22] Prof Deepak Khemani, Department of computer science and engineering, IIT Madras, khemani@iitm.ac.in.

[23] Truica, C. O., Boicea, A., & Trifan, I. (2013, August). CRUD operations in MongoDB. In 2013 International Conference on Advanced Computer Science and Electronics Information (ICACSEI 2013). Atlantis Press.

[24] Lai, Sen-Hao. "Cri/Ib." PhD diss., Massachusetts institute of technology, 1991.

[25] Ordóñez, F., De Toledo, P., & Sanchis, A. (2013). Activity recognition using hybrid generative/discriminative models on home environments using binary sensors. Sensors, 13(5), 5460-5477.

[26] Carole-Ann Berlioz." How the Rete Algorithm Works" Mar 14, 2011.

[27] Yan, Y., Zhang, L., Feng, T., Xie, P. and Gao, X., Location Big Data Partition and Publishing Method based on Sampling and Adjustment.

[28] Li, T., Zhang, L. and Yang, Z., Multi-criteria Group Decision Making Based on the Multiplicative Consistency of Intuitionistic Fuzzy Preference Relation.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731