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

**Context Dimension Tree (CDT)** is a generic context data model as shown in Fig.1. 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.
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.

\[
\text{Location (A, S) \land Place (B, S) \lor Place (B1, S) \Rightarrow Poss (Activity ((A, B, C), S)}
\]

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

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.

\[
\text{Location (X, S) \land Place (Y, S) \Rightarrow Poss (Activity ((X, Y, Z), S)}
\]

Whereas,
X=bed
Y= bedroom
Z= sleeping
S = Situation

System 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 sensed 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

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.
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

**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].

---

**Table 3: Adls Activity Labels For User**

<table>
<thead>
<tr>
<th>UserID</th>
<th>Start_time</th>
<th>End_time</th>
<th>Activity</th>
</tr>
</thead>
</table>

**Table 4: Adlssensor Events For User**

<table>
<thead>
<tr>
<th>UserID</th>
<th>Start_time</th>
<th>End_time</th>
<th>Location</th>
<th>Place</th>
<th>Type</th>
</tr>
</thead>
</table>
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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6. Adls Fact Generation

<table>
<thead>
<tr>
<th>ADSL Collection</th>
<th>ADLS Sensor Collection</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create Operation</strong></td>
<td><strong>Create Operation</strong></td>
</tr>
<tr>
<td>db. ADLS.insertOne({'UserID': 'User2', 'Start_time': '2012-11-12 01:54:00', 'End_time': '2012-11-12 09:31:59', 'Activity': 'Toileting'})</td>
<td>db. ADLSSensor.insertOne({'UserID': 'User2', 'Start_time': '2012-11-12 01:54:00', 'End_time': '2012-11-12 09:31:59', 'Activity': 'Toileting', 'Place': 'Bedroom', 'Location': 'Cabinet', 'Type': 'Magnetic'})</td>
</tr>
</tbody>
</table>

**Read operation**


**Update operation**

| db. ADLS.update({'Activity': 'Toileting'}, {'Set': {'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 7. Defining Rules

<table>
<thead>
<tr>
<th>ADSL</th>
<th>ADSLS Sensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>(deftemplate ADSL</td>
<td>(def template ADSLSensor</td>
</tr>
<tr>
<td>(slot UserID)</td>
<td>(slot UserID)</td>
</tr>
<tr>
<td>(slot Start_time)</td>
<td>(slot Start_time)</td>
</tr>
<tr>
<td>(slot End_time)</td>
<td>(slot End_time)</td>
</tr>
<tr>
<td>(slot Activity)</td>
<td>(slot Location)</td>
</tr>
<tr>
<td>(slot Type)</td>
<td>(slot Location)</td>
</tr>
<tr>
<td>(assert (ADLS1 (UserID User1) (Start_time 2011-11-28 01:54:00, (End_time 2011-11-28 10:18:11, (Activity toileting))) (Fact-1) (Type Pressure) (Place Bedroom) &lt;Fact-1&gt;</td>
<td>(assert (ADLS Sensor (UserID User1) (Start_time 2011-11-28 02:27:59, (End_time 2011-11-28 10:18:11, (Activity sleeping))) (Place Bed) (Fact-1) (Type Pressure) (Place Bedroom) &lt;Fact-1&gt;</td>
</tr>
</tbody>
</table>

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

| (deffrule ADSL12Sensor (User1 is in Bedroom) => (assert (User1 is in toilet)) (watch facts) (watch activations) (assert (User1 is in Bathroom)) == f-1 (User1 is in Bedroom) => Activation 0 ADSL1 Sensor: f-1 <Fact-1> | (deffrule ADSLSensor (User1 is in Bathroom) => (assert (User1 is in toilet)) (watch facts) (watch activations) (assert (User1 is in Bathroom)) == f-1 (User1 is in Bathroom) => => Activation 0 ADSL Sensor: f-1 <Fact-1> |

(agenda) ADSL Sensor: f-1 For a total of 1 activation (clear) (agenda) ADSL Sensor: f-1 For a total of 1 activation (clear)
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

![Image 9: Alpha Nodes](image)

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

![Image 10: Rete Network](image)

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.

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...
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:

```
```

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**
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


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


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