Novel Technique for Storing Knowledge using Hypergraphs and NoSQL

Rahul Mishra
Master of Computer Applications,
Veermata Jijabai Technological Institute

Anala A. Pandit
Department of Computer Applications
Veermata Jijabai Technological Institute,

ABSTRACT
Traditionally, major focus in knowledge management has been on various techniques of representation of knowledge. In the current scenario the information is available is more often, in unstructured formats. Representation and storage of this information is posing new challenges. This paper introduces the initial approach to knowledge management and the problems faced during knowledge collection. It also describes the use of hypergraphs and NoSQL for knowledge representation and the way data can be stored as information. This paper describes a novel technique for representation of knowledge and also proposes storage solution for the unstructured information in an effective and flexible manner as compared to traditional way of knowledge representation.

Keywords
Knowledge management, Hypergraphs, NoSQL

1. INTRODUCTION
Knowledge management is a term, used to refer to stored information that is useful for decision making and to define long term plans. [2] It aids learning from past experiences and refrain from making the same mistakes. The important part in knowledge management is to convert data into information which can be used to obtain conclusions for a particular scenario. This is an extremely important component in development of Knowledge management systems or Expert

Systems.

The traditional techniques of knowledge management focused mainly on processing whereas there was a need to focus on storage with increase in quantum of data. Hypergraph is a way of knowledge representation that consists of hyperedges which can connect to multiple vertices representing relationship between them. [1] Also using NoSQL for storage of knowledge gives the requisite flexibility to handle the unstructured information formats.

The rest of the paper is structured in the following manner. Section 2 presents the current methods and approach which are used for knowledge management and the collection of information along with problems faced during the process. Section 3 provides the proposed technique for knowledge representation and storage of unstructured data using hypergraphs and NoSQL. Section 4 explains the advantages of the proposed technique over current methods which are followed. Section 5 descsribes the conclusion and future directioins for the work.

2. CURRENT METHODS
Knowledge management consists of different steps from collection of information to applying different techniques such as *rule-based approach* or *case-based reasoning approach* in order to derive appropriate conclusions. However, each method of knowledge representation does have some limitations. Humans feel comfortable taking decisions based on rule-based approach, where 'if' part represents the 'condition' and the 'then' part specifies the action to be performed. In case of computational operation case-based reasoning is much preferred where, for any specific situations the actions are taken based on similar cases from the past. [2] [3]

To avoid problems of using a single technique, a combination of both the techniques were used for knowledge management which would support rule based and case based reasoning to arrive at conclusions. [2] But the issue with this solution is that it requires a lot of processing power, and in such a case then,

efficiency is a major concern. These are traditional ways of knowledge representation. [2] .The information required for knowledge representation is not directly available. It requires collection of data from different sources that can be people, system reports, past records etc.

The information can be obtained as '*Tacit Knowledge*' or '*Formalized Knowledge'*. [2] [3]. When the information is provided by people based on their experience, it is known as '*Tacit Knowledge*'. The problems faced during the collection of information is that people's data/opinions differ since the information is subjective. A person providing information should have proper domain knowledge which is required to design a knowledge representation. Also not everybody remembers all the information to guarantee that the information is complete and correct. '*Formalized knowledge*' is one which is available to all the employees of the organization. The records obtained from past reports might have different format because of which it would be difficult to integrate them and generate a proper document containing required information. This happens when information is collected from different sources that may be in different formats.

In traditional systems of the past, there was no means to import a data file already containing information because of technology constraints. One such example is of MYCIN in which user has to enter all the information manually through the use of keyboard and mouse which was time consuming. Modern information from internet, apps etc. are unstructured which requires a lot of time, effort and understanding to classify information. [2] [3]

3. PROPOSED TECHNIQUE

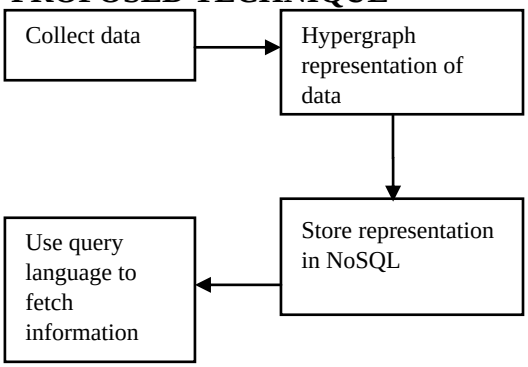

Fig 3.1 Flow of proposed technique

According to the techniques mentioned in Section 2 which are used for knowledge management there is a need to focus on storage of information so that it supports user specific queries.

Fig 3.1 shows the steps of the proposed technique. The first step is similar and common which indicates the data should be collected from sources required. This data can be of any format.

The second step is to convert the data into hypergraph representation according to requirements of an organization. This is followed by storing this information into NoSQL as it supports flexibility and is widely used for storing unstructured data. The fourth step is to query the stored information using appropriate query language for required tasks.

3.1 Hypergraphs for representation:

Hypergraphs consist of non-empty subsets of vertices and hyperedges. As compared to directed graphs the hyperedges can connect to multiple vertices. Also hyperedges can be a part of another hyperedge too. Hypergraphs are significant because they can be used to represent relationships among different entities considering the vertices as entities. The entities contain their information and hyperedges represent relationship among the vertices. [1]

Hypergraphs inspite of storing a lot of information represents a graph structure which is simple to understand and can be used to explain an structure of an organization or other aspects. Also hypergraphs are capable of storing any relational information. [1]

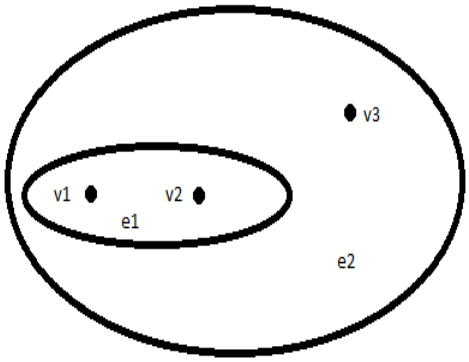

Fig 3.1.1: Hypergraph

Figure 3.1.1 shows a hypergraph that contains vertices or nodes represented by *v* and hyperedges represented by *e*. In above diagram *e2* is connected to *e1* which represents a relationship among them. In order to represent knowledge using hypergraph consider an example where '*employee*' and '*department*' can be considered as vertices. These two vertices are connected by a relation called '*belongs to'* that represents an hyperedge. This hyperedge indicates that there is a relation between *employee* and *department* represented by an hyperedge *belongs to.* However, there is one more point to be considered which is the roles defined to employee and department. The role of *employee* is one who *belongs* and that of *department* is where he belongs. [1]

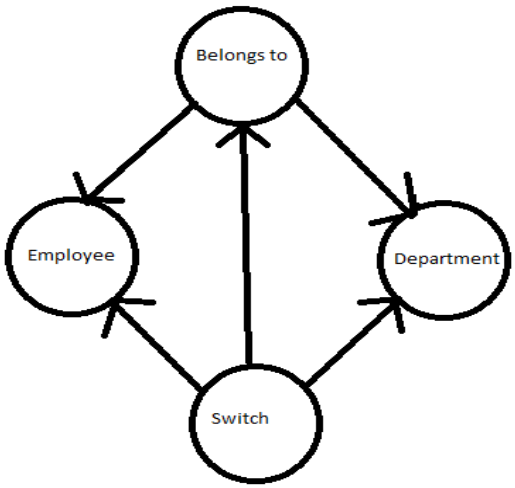

Fig 3.1.2 :Employee hypergraph

Consider figure 3.1.2 which is other relation switch which is a hyperedge. This one obviously connects employee and department, it provides the roles *one who belongs* and *where one belongs to* but it also contains the relation *belongs to* because employee should be in department to be *switch*. This is beauty of representation of knowledge using hypergraph.

Table 1. Department Table

Id	**Department name**
Dep1	Sales

Table 2. Employee Table

Id	**Name**	**Department**
1	Abc	Dep1

Table 3. Department switch table

SwitchId	**BelongsId**	**DepId**	**EmpId**
Sw1	B1	Dep1	1

If one compares this to a database format as shown in table 1, table 2 and table 3 then it can be seen that relations *belongs to* and *switch* preserve values of relationship and *employee* and *department* represent entities. There are no primary or foreign key attributes here. The strict nature of database tables does not allow us to make changes easily in the database for example keeping an attribute blank or adding more values. But in hypergraphs we can add additional attribute values representing an entity.

3.2 NoSQL for storage

As hypergraphs are flexible it implies that it does not have a strict schema. The nodes and hyperedges can be added or removed based upon the needs of the problem or the user. In

addition, there is also a need to store these hypergraphs which contain relationship between nodes in the form of hyperedges. It stores knowledge to perform different operations on it.

Relational databases cannot be used for this because of their strict schema and inflexibility. The structure of the data is not the same. To store such wide variety of data, one would require a lot of time and effort for cleaning and converging to a standard format out of it. Also if there are any changes required then the same process will need to be repeated. Since NoSQL allows data in different formats and the data collected from different sources have different formats, NoSQL is suitable in order to store hypergraphs. [13]

In NoSQL the schema is flexible. Data can be stored in the

following forms in different types of NoSQL databases available:

- key value pairs,

- document,

- graph,

- columnar [13].

Cloud companies use NoSQL for handling large amounts of distributed data because it's structure is not rigid. [13] Any form of data can be stored in NoSQL. There is no need to change schema in order to make changes, just add it in a suitable format into NoSQL. All these properties makes NoSQL a perfect option in order to store hypergraphs. These databases can easily store hyperedges and information about nodes.

Consider an organization which is using hypergraphs to represent the data. In this case if the data is first collected from multiple sources, the data can be represented as hypergraphs to represent knowledge. These hypergraphs can then be stored into NoSQL. If any changes occur in future then a new node with new hyperedge representing a relation between them can be added, as hypergraphs and NoSQL both support flexibility.

3.3 Using GRAKN.AI with an example

GRAKN.AI is a hyper relational database which is used as backend for knowledge storage. It uses graql which is a query language. GRAKN uses concepts to define the schema of storing knowledge. Concepts are anything which represents the domain. In simple words it consists of all components of the specific domain which should be a part of the knowledge base.

Consider a scenario of creating a schema for storing the employee information belonging to a particular department in an organization. In this scheme of things, everything is represented as a concept which are part of the domain which in this case can be *employee*, *department*. The next step is to identify entities, relationships, attributes from these concepts.

GRAKN follows a hierarchical structure therefore each concept should have exactly one parent. One type can be a child of another type. For example we can define an entity as animal and cat can be a child of that entity. The roles are defined for each entity so that there is no confusion. One cannot change meaning of a relationship by providing invalid values like an oil platform owns company.

The schema can be defined as follows:

define

"employee" sub entity

has name plays oneWhoBelongs;

"department" sub entity

has name

plays whereItBelongs

plays depItWants;

"belongsto" sub relationship relates oneWhoBelongs relates whereItBelongs;

"switch" sub relationship relates oneWhoBelongs relates depItWants;

"oneWhoBelongs" sub role;

"whereItBelongs" sub role;

"depItWants" sub role;

name sub attribute datatype string; [10] [11]

command to load graql schema file [12]:

/graql console -k yourkeyspace -f Path to/schema.gql

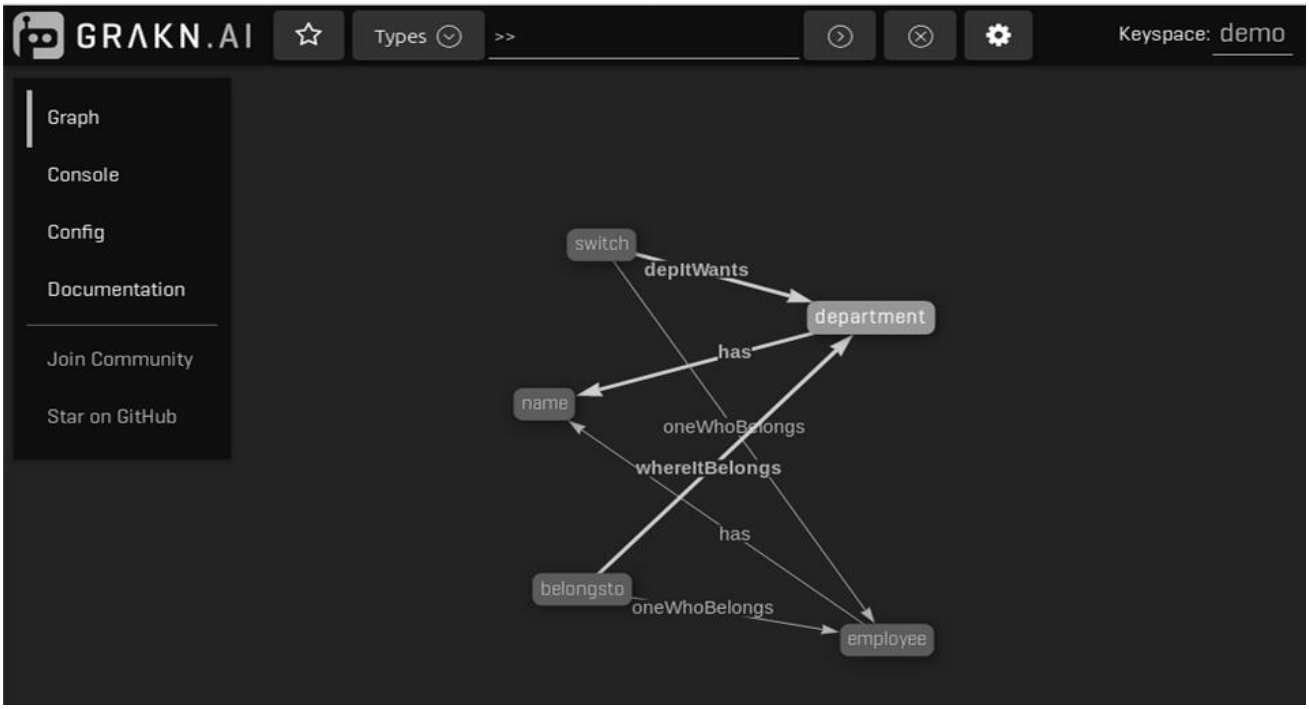

Fig 3.3.1: Schema for employee

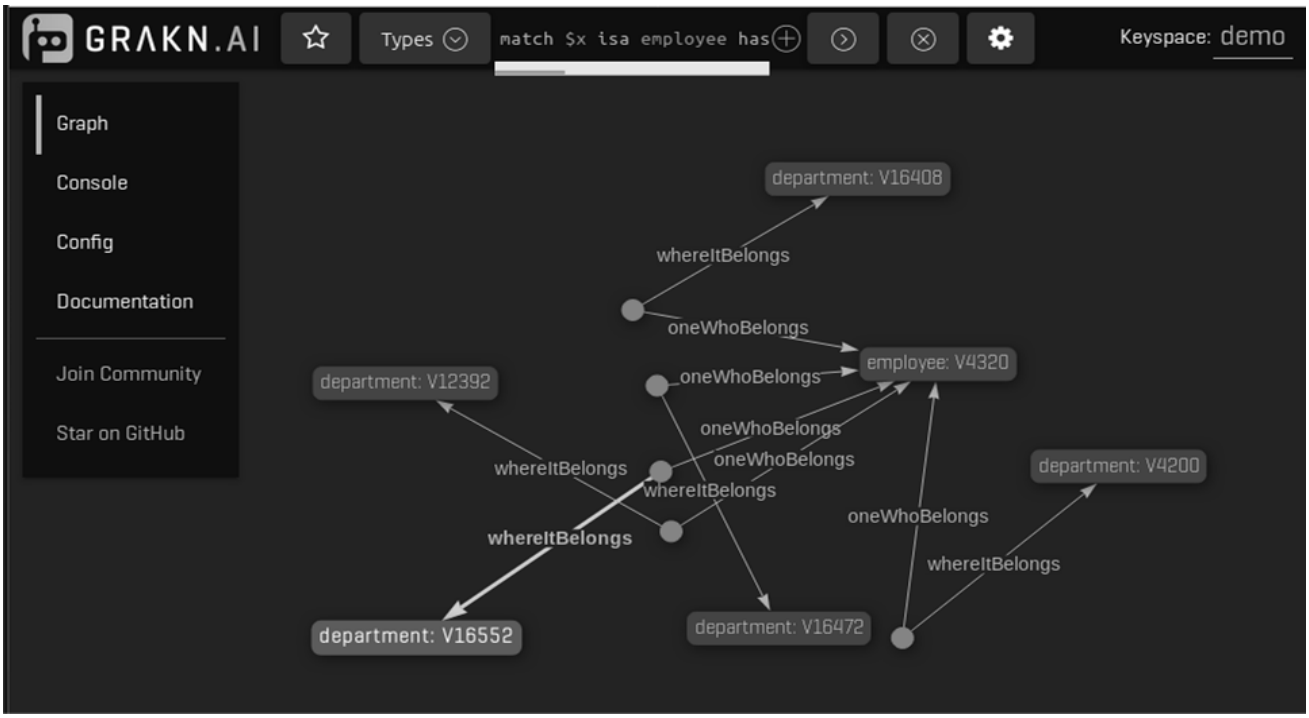

Fig 3.3.2: Employee belongsto Relation

Fig 3.3.1 is to check whether file is loaded properly or not. The diagram shows hierarchical relationship among all types. Loading data can be done using insert command

insert $x isa employee has name="abc";

insert $x isa department has name="dep1";

GRAQL query language is used to fire queries to find and add relationship as follows:

match $x isa employee has name="abc"; $y isa department has name="dep1"; insert (oneWhoBelongs:$x,whereItBelongs:$y) isa belongsto; [7]

After adding the above relation the schema in GRAKN.AI changes to figure 3.3.2 which can be viewed using query:

match $x isa employee has name="abc";($x,$y) isa belongsto; offset 0; limit 30; get; [6]

Fig 3.3.2 displays that a relation is created between an *employee* and *department*. There are many more operations which can be performed in GRAKN but the special feature of GRAKN is that it provides a hyper relational approach which is efficient and flexible.

4. ADVANTAGES

The proposed technique combines the benefits of both hypergraphs and NoSQL. The representation and storage of information is flexible which means that in future if there is a need to make changes it can be done easily as compared to strict structure of relational schemas. It saves time by reducing task of changing the schema to suit current need which affects all previous entities. This increases efficiency. It expresses higher-order information such as nesting of information, relations between relations etc. It does not impose any restriction on attributes belonging to an entity. Graph oriented computation techniques can be applied. The structure of the representation of information is similar to graph which is easy to understand. [1]

5. CONCLUSION

The scope of information required for decision making has increased in recent times and will continue as time progresses. There will always be a need for integration of information from different sources to form a knowledge base. Hypergraphs and NoSQL provides a reasonable approach to store information and form relationship to enable to make decisions. It treats everything as a concept, not as a record to be stored. This makes it more efficient and flexible. It supports computational processing. To take this work further, more work is required in the implementation of storage and identifying the best type of NoSQL suitable for knowledge storage. There has been a lot of focus in the past for processing of information but due to increase in unstructured data there is a need to work in future on representation of information for analytical purposes.

6. REFERENCES

[1] Modelling Data with Hypergraphs, Szymon Klarman, 2017

[2] Artificial Intelligence Support of Knowledge Transformation in Knowledge Management Systems, Tatiana V. Avdeenko, Ekaterina S. Makarova, Irina L. Klavsuts, 2016

[3] The Artificial Intelligence in Personal Knowledge Management, Lixin Diao1, Mingzhang Zuo1, Qiang Liu1, 2009

[4] https://grakn.ai/, available on 06/06/2018

[5] https://dev.grakn.ai/academy/graql-intro.html, available on 06/06/2018

[6] https://dev.grakn.ai/academy/get-queries.html, available on 06/06/2018

[7] https://dev.grakn.ai/academy/insert-delete-queries.html, available on 06/06/2018

[8] https://dev.grakn.ai/academy/schema-elements.html, available on 06/06/2018

[9] https://dev.grakn.ai/academy/conceptual-modeling-intro.html, available on 06/06/2018

[10] https://dev.grakn.ai/academy/schema-building.html, available on 06/06/2018

[11]https://dev.grakn.ai/academy/schema-building-continued.html, available on 06/06/2018

[12] https://dev.grakn.ai/academy/loading-files.html, available on 06/06/2018

[13]https://searchdatamanagement.techtarget.com/definition/NoSQL-Not-Only-SQL, available on 06/06/2018