Annals of Emerging Technologies in Computing (AETiC)
Vol. 5, No. 5, 2021

Review Article

A Unified Conceptual Model for Data Warehouses

Shreya Banerjee[1,*], Sourabh Bhaskar[2], Anirban Sarkar[3] and Narayan C. Debnath[1]

[1]Eastern International University, Binh Duong, Vietnam
shreya.banerjee@eiu.edu.vn; narayan.debnath@eiu.edu.vn
[2]Sardar Vallabhbhai National Institute of Technology, India
sourabhb440@gmail.com
[3]National Institute of Technology, Durgapur, India
sarkar.anirban@gmail.com
*Correspondence: shreya.banerjee@eiu.edu.vn

Received: 17th October 2020; Accepted: 17th November 2020; Published: 20th March 2021

Abstract: These days, NoSQL (Not only SQL) databases are being used as a deployment tool for Data Warehouses (DW) due to its support for dynamic and scalable data modeling capabilities. Yet, decision-makers have faced several challenges to accept it as a major choice for implementation of their DW. The most significant one among those challenges is a lack of common conceptual model and a systematic design methodology for different NoSQL databases. The objective of this paper is to resolve these challenges by proposing an ontology based formal conceptual model for NoSQL based DWs. These proposed concepts are capable of realizing the cube concepts for visualization of multi-dimensional data in NoSQL based DW solutions. In this context, two strategies are specified, implemented and illustrated using a case study for devising of the proposed conceptual model.

Keywords: ***Conceptual Model; MongoDB based Implementation; NoSQL Data Warehouse; Ontology-driven Model***

1. Introduction

Over the last few years, NoSQL databases have achieved strong popularity. These new generation databases are different from traditional relational databases for possessing several significant features such as persistent and non-relational data, flexible schemas, high availability, dynamic insertion of different kinds of data, replication, massive horizontal scaling and distribution. Modern Data Warehouses (DWs) are competent to cope with and excel with emerging data analysis trends such as fast query expectations from users, data generated from cloud, unstructured or non-relational data, and rapid synthesis of data [1]. Thus, DWs solutions nowadays demand to act more in internet-style than to enforce the user to act within predefined structures [2]. Usually, classical DW and On Line Analytical Processing (OLAP) are comprised of a set of concepts like, facts, dimensions, measures and dimension hierarchies, those are used for structured schema representations [3]. The concept of cube is used for multi-dimensional data visualization. However, in case of web-scale applications, many of the dimensional information may not be available in regular structure. Consequently, decision makers are increasingly using NoSQL databases to implement their business solutions [4].

NoSQL databases are classified based on different physical level data models. Those are Document Store, Graph databases, Key-Value stores, and Column-Family store [5]. This heterogeneity brings several dimensions of challenges in systematic design methodology for NoSQL based DW solutions. Firstly, lack of common conceptual model for different NoSQL

Shreya Banerjee, Sourabh Bhaskar, Anirban Sarkar and Narayan C. Debnath, "A Unified Conceptual Model for Data Warehouses", *Annals of Emerging Technologies in Computing (AETiC)*, Print ISSN: 2516-0281, Online ISSN: 2516-029X, pp. 162-169, Vol. 5, No. 5, 20th March 2021, Published by International Association of Educators and Researchers (IAER), DOI: 10.33166/AETiC.2021.05.020, Available: http://aetic.theiaer.org/archive/v5/v5n5/p20.html.

databases poses significant research challenges in design of DWs. Secondly, NoSQL based implementation of DWs requires a systematic design methodology, comprises of different levels of abstraction in DW design including, conceptual level, logical level and physical level [6]. A conceptual DW model will isolate the purpose of designer from its execution. Thirdly, representation of agreeable numerical data (DW concepts like, facts and measures) and contextual data (dimensions and its hierarchies) are needed in order to illustrate the effective associations among Fact, Measure and Dimension [1]. Fourthly, De-normalization of both contextual and numerical data is also required to achieve flexible characteristics of NoSQL databases. Fifthly, realizations of data cubes are important for visualizing and executing analytical queries effectively.

The objective of this paper is to address these abovementioned challenges. The research methodology followed in this paper is described next. An ontology driven common conceptual model for NoSQL based DW system is proposed to resolve the mentioned challenges. Ontology is defined as an explicit specification of shared conceptualization of the elements of DW domain in terms of concepts and related axioms [7]. Axioms enable ontology to provide enriched and formal semantics towards different concepts. The proposed conceptual model is capable to represent a generalized and rigorous formal set of concepts at the conceptual level design phase of DW using NoSQL database features. In the proposed conceptual model, several generic concepts of the model described in [8], are extended for DW domain. Further, the proposed conceptual model is implemented in a document-oriented database MongoDB. However, it can be transformed towards other NoSQL based DWs, such as Columnar, Key-Valued and Graph oriented. Figure 1 describes the proposed design methodology of NoSQL based DW system.

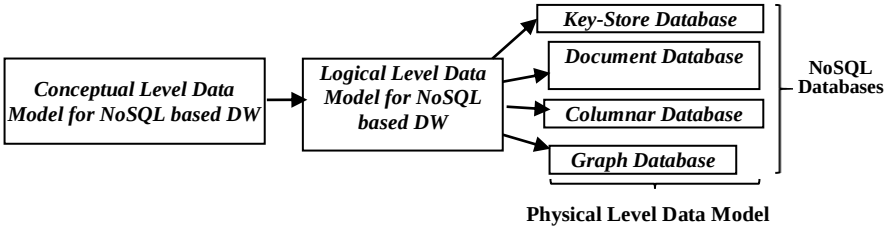

Figure 1. Proposed systematic design methodology for NoSQL based data warehouses

2. Related Work

Several research works exist in representation of formal conceptual model for NoSQL based DW. In [1], [9-11] authors have described a conceptual model for NoSQL based OLAP systems that can be mapped towards either Column or Document oriented DW using a set of rules. In [6], an existing benchmark for relational database based DWs is improved towards a generalized benchmark for distinct NoSQL based DWs. This approach is based on Star schema for DW. In [12], physical DW design is investigated over column-oriented databases through Map-Reduce framework using Hbase. In [4] and [13], authors described rules for implementing DW in document-oriented database systems and Hive respectively. In [14], authors have described a method that has used ontology to generate the multidimensional schema from a conceptual formalization of a domain. However, NoSQL databases are not considered in this approach. In [15], a new aggregation operator, known as CN-CUBE (Columnar NoSQL CUBE) is described. Further, it is implemented in column-oriented NoSQL database. Majority of existing works described models for NoSQL based DW system specific to its physical level implementations. These models are transformed towards either columnar or document oriented NoSQL databases using a set of rules. Further, semantics of distinct concepts are not well explored in these approaches. Moreover, illustrations of data cubes are represented by few approaches and are confined towards specific NoSQL solutions.

3. Proposed Conceptual Model for NoSQL Based DWs

Proposed conceptual model is consisting of group of constructs, relationships and a number of significant properties to unify conceptual level representations of different NoSQL based DW solutions. Ontology is applied for the proposed conceptualization to provide rigorous and formal vocabularies towards distinct facets. The proposed conceptualization is consisting of all details those are necessary for representation the concepts of facts, dimensions and measures in DW. Further, it provides the concepts of data cubes and dimension hierarchies when multi-dimensional data are heterogeneous types, and ranged from structured to semi-structured. The proposed conceptual model is equally useful for traditional DW modelling using relational databases when related dimensional data, fact data and their relationships are strictly structured and homogeneous in nature. All concepts in the proposed model are represented through axioms expressed using mathematical logic. Figure 2 has illustrated the proposed conceptual model.

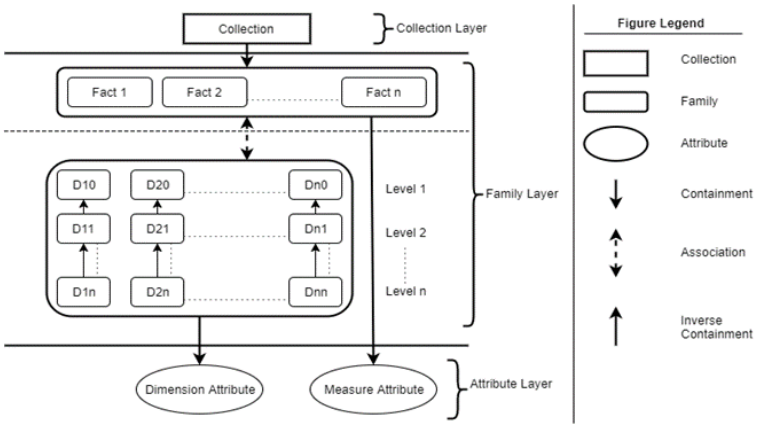

Figure 2. Proposed conceptual level data model for NoSQL based data warehouses

3.1. Constructs and Layers in proposed conceptual model

Proposed conceptual model has a layered organization. This model is consisting of three main layers namely- *Collection*, *Family* and *Attribute*. All these three layers have their respective construct types- *Collection* (*col*), *Family* (*FA*), and *Attribute* (*AT*). Fact and dimension hierarchies in DW map towards Family layer of the proposed conceptualization. The measure and members of dimensions are mapped towards *Attribute* layer. Further, *Collection* layer realizes the data cubes based on facts.

(a) ***Attribute Layer***: It is the base layer of the proposed conceptual model. Key construct type of this layer is *Attribute* (*AT*) that is the group of all possible instances of a data item. *AT* is elementary in nature. This can be of two types namely- *Measure Attribute* (M_{AT}) and *Dimension Attribute* (D_{AT}). A M_{AT} represents single measure of a fact in a DW. On the other hand, a D_{AT} represents single attribute belonging to a dimension in a DW. Formalization of *AT* is,

F1:$\forall x(AT(x) \rightarrow (M_{AT}(x) \oplus D_{AT}(x)))$

***Explanation*:** *F1* specifies that an *AT* instance *x* can be either M_{AT} type or D_{AT} type.

(b) ***Family Layer***: It is the middle layer of conceptual model. Main Construct type of this layer is *Family* (*FA*). An *FA* is created from a group of semantically related *AT*. It can be of two types namely- *Fact Family* (*FF*) and *Dimension Family* (*DF*). *DF* can be decomposed into multiple levels as per the designer's choice. However, an *FF* has single level. Multiple levels in *DF* represent hierarchies in dimensions. The lowest level *DF* will demonstrate the high level of granularity in NoSQL based DWs. This kind of *DF* will be composed from the set of D_{AT} only. The upper layer *DF* in the dimension hierarchy is the assembling of one or more D_{AT} and associated *DFs* of adjacent inner layer. *FF* comprises of related topmost layer *DFs* and a group of M_{AT} defined on measures. Formalization of Family is,

F2: $\forall x \exists r \exists v(FA(x) \leftrightarrow (AT(v) \wedge Cnt_{FA}(r) \wedge r(x,v) \wedge (FF(x) \vee DF(x))))$

***Explanation*:** If there exists an *FA* then that *FA* may encapsulate several *AT*.

***F3*:**$\forall x \exists r1 \exists y \exists v \exists r2(FF(x) \leftrightarrow (Cnt_{FF}(r1) \wedge M_{AT}(v) \wedge r1(x,v) \wedge AS(r2) \wedge DF(y) \wedge r2(x,y)))$

***Explanation*:** If there exists an *FF* then that *FF* should encapsulates several M_{AT} and associated *DF*. Here, x, v and y are instances of *FF*, M_{AT} and *DF* respectively.

F4:$\forall x1 \exists x2 \exists r1 \exists r2 \exists r3 \exists v1 \exists v2((DF(x1) \wedge DF(x2)) \rightarrow (Cnt_{DF}(r1) \wedge Cnt_{DF}(r2) \wedge Icnt_{DF}(r3) \wedge D_{AT}(v1) \wedge D_{AT}(v2) \wedge r1(x1, v1) \wedge r2(x2, v2) \wedge r3(x1, x2) \wedge notEqual(x1, x2)))$

***Explanation*:** If there exists a *DF* then it should encapsulate several D_{AT}. Further, those *DF* can be encapsulated in another *DF* dynamically.

(**c**) ***Collection Layer***: This is the top most layer of the conceptual model. Key construct type of this layer is *Collection* (*col*). A *col* is created from a combination of semantically related *FF*. Thus, from the top level the whole DW can be seen as set of *Collections*. Formalization of *Collection* is,

F5: $\forall x \exists r \exists v(col(x) \leftrightarrow (FA(v) \wedge Cnt_{col}(r) \wedge r(x, v)))$

***Explanation*:** If there exists a *col* then it should encapsulate several *FF*. Here, x and v are instances of *col* and *FF* respectively.

(**d**) ***Cube***: Cube is the de-facto logical representation for data visualization. Cube can be created from *FF* and realized as a *col* in the proposed conceptual model. If there are multiple *FF*, then a cube can be devised for each *FF* or combinations of *FF*. In the latter case, *FF* can share *DF* and related *MA*. Formalization of cube as,

F6: $\forall x \exists r1 \exists e \exists v \exists r2 \exists k \exists r3(cube(x) \leftrightarrow (FF(v) \wedge DF(e) \wedge M_{AT}(k) \wedge Cnt_{col}(r1) \wedge Cnt_{FF}(r2) \wedge AS(r3) \wedge r1(x, v) \wedge r3(v, e) \wedge r2(v, k)))$

***Explanation*:** Several instances *Cube (v)* can be realized from each instance of *FF* (x) including base and apex level cubes.

3.2. Relationships in the Proposed Conceptual Model

In the proposed conceptual model, distinct construct types are connected with each another using different relationships. Proposed relationships can be classified in two types. One is *inter-layer kind* relationships and another is *intra-layer kind* of relationships. *Inter-layer kind* relationships exist between disparate construct types of two distinct layers. Whereas, *Intra-layer kind* relationships exit between analogous construct types of a similar layer.

(a) ***Containment* (*Cnt*):** These relationships exist when one construct type encapsulates another construct type. Thus, *Cnt* are present between three pairs of concepts in the proposed conceptual model – (i) one *col* can contain several *FF*, (ii) an *FF* can contain several M_{AT} and (iii) a *DF* can contain several D_{AT}. Therefore, both inter-layer and intra-layer kind relationships can include *Cnt* relationships. Formal axioms of *Cnt* are

***F7*(i)**:$\forall r \exists y \exists^n z(Cnt_{col}(r) \leftrightarrow (Col(y) \wedge FF(z) \wedge r(y, z) \wedge (greaterthanEqual(value(n), 1))))$

***F7*(ii)**:$\forall r \exists y \exists^n z(Cnt_{FF}(r) \leftrightarrow (FF(y) \wedge M_{AT}(z) \wedge r(y, z) \wedge (greaterthanEqual(value(n), 1))))$

***F7*(iii)**:$\forall r \exists y \exists^n z(Cnt_{DF}(r) \leftrightarrow (DF(y) \wedge D_{AT}(z) \wedge x(y, z) \wedge (greaterthanEqual(value(n), 1))))$

(b) ***Inverse Containment* (*Icnt*)**: This relationship is intra-layer kind and connects two construct types when one is encapsulated towards another construct type dynamically. Direction of this relationship is opposite to the *Cnt* relationship. In the proposed conceptual model, lower level *DFs* are encapsulated towards higher-level *DFs* using *Icnt* relationships. This relationship is helpful to represent distinct levels of granularity in dimension hierarchies. It is capable to add different dimensions in distinct granular level on the fly and useful to change granularity level dynamically.

F8: $\forall x \exists y \exists^n z(Icnt_{DF}(r) \leftrightarrow (DF(y) \wedge DF(z) \wedge DF\_level(y) \wedge DF\_level\_next(z) \wedge r(z, y) \wedge (greaterthanEqual(value(n), 1))))$

(c) ***Association* (*AS*):** These relationships are intra-layer kind and connect constructs types anticipated to achieve several goals together. An *AS* may exist between *FF* and *DF*. Further, *AS* can be present between two different *cols*.

***F9*(i)**: $\forall x \exists^n y, z \; \exists l(AS(r) \leftrightarrow (FF(y) \wedge DF(z) \wedge r(y, z) \wedge (greaterthanEqual(value(n), 1))))$

***F9*(ii)**:):$\forall x \exists^n y, z \; \exists l(AS(r) \leftrightarrow (Col(y) \wedge Col(z) \wedge notEqual(y, z) \wedge r(y, z) \wedge (greaterthanEqual(value(n), 1))))$

3.3. Properties of various relationships

The proposed set of relationships support several properties like, *Cardinality*, *Ordering*, and *Modality* to handle both structured and flexible nature in the model.

(*a*) *Cardinality* (*Crd*) and *Modality* (*Mdl*): Numbers of participate instances in *Cnt, Icnt* and *AS* Relationships are represented through *Crd*. *Mdl* defines optional and/or mandatory participation of constructs in a relationship. Optional participation is formally represented through possibility operator- ◊ and mandatory participation is represented through the necessity operator - □. *Crd* and *Mdl* is shown in the proposed conceptual model graphically using *P*. There can be different values for *P*. Those are

(i) 1:1 – This represents *AT* and *FA* relationship with mandatory total participation

(ii) 0:1 – This represents *AT* and *FA* relationship with optional one participation.

(iii) 1:M – This represents *AT* and *FA* relationship with mandatory multiple participation.

(iv) 0:M – This represents optional multiple participation of *AT* and *FA* in a relationship.

(v) 0:X – This represents optional exclisive participation of *AT* and *FA* in a relationship.

(vi) 1:X – This represents *AT* and *FA* relationship with mandatory exclusive participation.

Formally, *Crd* and *Mdl* for *Icnt* relationship can be expressed as,

$\boldsymbol{F10}$: $\forall r \exists y \exists z \exists P((Icnt_{DF}(r) Crd(r) \wedge Mdl(r)) \leftrightarrow (DF(y) \wedge DF(z) \wedge r(y,z) \wedge q(value(P))))$

In the similar way, *Crd* and *Mdl* for other relationships can be expressed.

(*b*) *Ordering* (*Ord*): This property realizes whether the constructs participating in a relationship are in order or not. *Ord* is shown in the proposed conceptual model graphically using θ. If value of θ is *1*, then participants are in order. On the other hand, if value of θ is 0, then participants are not in order. Formally, *Ord* for *Icnt* relationship can be expressed as,

$\boldsymbol{F11(v)}$: $\forall r1 \exists r2 \exists y \exists z \exists a((Icnt_{DF}(r1) \wedge Icnt_{DF}(r2) \wedge DF(y) \wedge DF(z) \wedge DF(a) \wedge r1(z,y) \wedge x2(a,y) \wedge notEqual(z,a) \wedge Ord(r1) \wedge Ord(r2)) \rightarrow Ordered\_Set(r1,r2))$

In the similar way, *Ord* for other relationships can be represented.

4. Illustration of the proposed conceptual model using a case study

Let, a case study related to a DW system based on sales and shipping. Sales of different products can be done in sale branches. Branches can be located in multiple locations. Shipping can have multiple shippers who will ship the product from one location to another.

Collections (*Cubes* created from *Fact Families*)
FACT FAMILY 1 (*SALES*)
FACT FAMILY 2 (*SHIPPING*)
SALES(*Location*, *Branch*, *Product*, *Time*, *units sold*, *dollars sold*)
SHIPPING (*Location*, *Shipper*, *Product*, *Time*, *units shipped*, *dollars shipped*)
Location (location_Id, pin code, {street}, city_Id)
City (city_id, city, state_Id)
State (state_Id, state, country_Id)
Country (country_Id, country)
Branch (branch_Id, branchName)
Product (product_Id, product_Name, productType_Id)
ProductType (productType_Id, productType_Name)
Time (time_Id, time, day_Id)
Day (day_Id, day, month_Id)
Month (month_Id, month, year_Id)
Year (year_Id, year)
Shipper (shipper_Id, shipperName, locaton_Id)

Nomenclature

Collections: In Capitalize and **bold**;
Fact Families: in UPPERCASE and *italic*
Dimension Families: in Capitalize and *italic*
Measure Attributes: in lowercase and *italic*
Dimension Attributes: in lowercase
Optional Construct Type: within {}

Figure 3. Key elements of the specified case study

This case study has two facts – *Sales* and *Shipping*. These two facts may have multiple dimensions with hierarchy. Several dimensions can be shared by both facts. *Sales* is associated with four dimensions - *Location*, *Branch*, *Product*, and *Time*. Further, *Shipping* is associated with four dimensions - *Location*, *Shipper*, *Product*, and *Time*. Thus, two facts share three dimensions. Several dimensions have hierarchy and specific attributes. For example, dimension Time has hierarchy – *Time→Day→Month→Year*. Time has several attributes for example *Time* Id, and *Time*. Beside this, each fact are associated with two measures. *Sales* is associated with *Units Sold* and *Dollars Sold*. *Shipping* is associated with *Units Shipped* and *Dollars Cost*. In some cases, *Location* dimension has attributes either *Pin Code* or *Street* and information related to *Branch* dimension is missing.

Last two statements in the previous paragraph specify that the described data set is irregular. This requires flexible representation. Consequently, this data set need to be demonstrated using NoSQL databases. According to the case study, *Sales* and *Shipping* are *FF* in proposed conceptual

model. Further, all dimensions and its related hierarchy are mapped towards *DFs*. *Attributes* contained in dimensions are mapped towards D_{AT} and measures are mapped towards M_{AT}. Figure 3 represents the key elements of the case study. Data cubes related to the case study can be realized through distinct *col*s based on different *FFs*. Figure 4 has illustrated Shipping *FF* along with related DFs and M_{AT} with corresponding cardinality and dimension hierarchy.

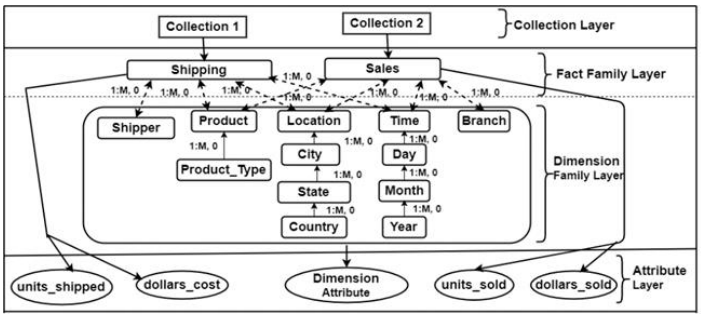

Figure 4. Shipping and Sales Fact Family with related Dimension Families and Measure Attributes

5. Implementation Strategy

In this section, two kinds of strategies are proposed for implementation of data cubes in NoSQL based DW systems. Further, the proposed conceptual model is transformed towards a Document Oriented database MongoDB. In addition, two implementations strategies are illustrated using MongoDB based on the case study specified in section ***4***. Proposed implementation strategies are useful for visualization of multi-dimensional nature of NoSQL based DW systems. However, there is no binding to use other kinds of NoSQL databases for implementation inline of the proposed strategies.

Table 1. Comparison Table between *Multiple Collection* based and *Single Collection* based Implementation

Multiple Collection based Implementation	**Single Collection based Implementation**
This strategy has less redundancy, because, a fact or shared dimensions are defined once	This strategy has high-level redundancy, since a fact or shared dimensions are defined multiple times
After defining once, insertion of new data definitions are propagated to other places. Hence, addition of data definitions can be handled easily	addition of data definitions is costlier, as newly added dimensions and measures have to be defined multiple times
maintenance is inexpensive than single one	maintenance is expensive than multiple one
due to more data integration policy, query execution time will be higher	due to less data integration policy, query execution time will be lower

Single Collection based Implementation Strategy: In this strategy, data cubes will be realized as a single *col* of a *FF*. Thus, numbers of data cubes in DW system depend on numbers of *FFs*. Hence, if there are *n* numbers of *FFs*, then there should be *n* numbers of data cubes. These *FFs* have nested related *DFs*, *DF Hierarchies*, D_{AT} and M_{AT}.

Multiple Collection based Implementation Strategy: In this strategy, a data cube can be realized based on multiple *cols* of *FFs* and related *DFs*. These multiple *cols* include *cols* of each *DFs* related with a *FF* and a *col* of the *FF* itself. These *DFs* nest related dimension hierarchies and D_{AT}. Further, the M_{AT} are nested in the *FF*. In this strategy, data cubes will be devised dynamically (on the fly) by associating multiple *cols* of *FF* and *DF*. This strategy is capable of creation of flexible schema for NoSQL based DWs by adding of measure and dimension definitions using *Icnt* and *AS* relationships. Table 1 has described the differences between these two different strategies.

5.1. Mapping towards MongoDB

Table 2 specifies the transformation between constructs of proposed conceptual model and MongoDB. In single collection based Implementation Strategy, data cubes are realized through single"Collection" element of MongoDB that is comprised of "Documents" elements corresponding to a *FF*. Based on the case study specified in section *4*, data cubes are created from *Sales* fact and realized as a single "Collection" element that nests "Document" element corresponding to *Sales* fact. Further, *Sales* fact encapsulates "Documents" elements for related dimension hierarchies, namely,

Location, *Branch, Product* and *Time*. "Document" elements for *Sales* fact also encapsulate measures *Units Sold* and *Dollars Sold*. Similarly, another data cube can be created from *Shipping* fact separately. Figure 5 has illustrated the corresponding implementation in MongoDB.

Table 2. Summarization of Mapping from proposed conceptual model towards MongoDB

Facets of proposed conceptual model	**Equivalent MongoDB representation**
Collection construct type	Collection
Fact family construct type	Document
Dimension family construct type	Document
Dimension attributes	Field
Measure attributes	Field
Association	Represented using Nested document
Containment	Represented using Nested document
Inverse Containment	Dynamic insertion of document towards another document without specifying its schema
Cardinality	***1: M***:- The construct type with participation *M* will be parent document or member field, and the construct type with participation *1* will be the nested document or member field. ***1: 1***:- Any one of two construct types can become nested document or nested member field of another construct type.
Optional modality	Flexible modality of all relationships.
Ordering	Ordered set is mapped towards "Array" and unordered set mapped towards "document".

In multiple collections based implementation strategy, data cubes from multiple "Collection" elements in MongoDB are created for all FF and DF elements and further aggregated towards creation of required data cube. Based on the case study specified in section 4, a data cube created for *Shipping* fact is based on an aggregated "Collection" element. This aggregated "Collection" is implemented by associating "Collection" elements of the fact Shipping and each related dimension hierarchies Location, Shipper, Time, and Product. "Document" element representing Shipping fact also encapsulates measures Units Shipped and Dollars Shipped. In MongoDB, a data cube can be built for Multiple Collections based Implementation Strategy using "aggregate()" function. Figure 6 has specified multiple "Collection" elements. Figure 7 has illustrated a data cube that is created from multiple "Collection" elements (figure 6) using "aggregate" operator.

```
{ "_id" : ObjectId("5a1806f29933a9a339ae590a"), "units_sold" : 10.0, "dollars_sold" : 18.0, "Location" : {"location_Id" : 101.0, -----}
```

Figure 5. Single Collection based Implementation Strategy in MongoDB based on the specified case study

```
Collection of Shipper Dimension (collection1){"_id" : ObjectId("5a140bd51b94b042474b8fd7"),"Shipper" : {"shipper_Id" : ----}
Collection of Location Dimension (collection2){ "_id" : ObjectId("5a155ec53b820e506813c9f9"),"Location" : {"location_Id" :---}
Collection of Product Dimension (collection3){"_id" : ObjectId("5a1560733b820e506813c9fc"),"Product" : {"product_Id" :----}
Collection of Time Dimension (collection4){ "_id" :ObjectId("5a1561643b820e506813c9fe"), "Time" : {"time_Id" : 401.0, "----}
Collection of Shipping Fact { "_id" :ObjectId("5a17b7728d4ca0e7112c7931"), "location_Id" : 101.0, "shipper_Id" : 501.0, ---}
```

Figure 6. MongoDB based implementation for Collections of Shipping fact and related dimensions

```
db.shipping.aggregate([{"$lookup":{"from":"shipper","localField":"shipper_Id","foreignField":"Shipper.shipper_Id","as":"collection1_
doc"}},{"$unwind":"$collection1_doc"},
{"$lookup":{"from":"location","localField":"location_Id","foreignField":"Location.location_Id","as":"collection2_doc"}},
{"$unwind":"$collection2_doc"}, ---------"Location":"$collection2_doc.Location","dollars_cost":1,"units_shipped":1}}]).pretty()
```

OUTPUT:

```
{ "_id" : ObjectId("5a17b7728d4ca0e7112c7931"), "dollars_cost" : 64.0, "units_shipped" : 4.0,"Shipper" : {"shipper_Id" : 501.0,
"shipper_Name" : "Ankur", "loction_Id" : 132.0}, ----------
```

Figure 7. Multiple Collection based Implementation Strategy in MongoDB for the specified case study

6. Conclusion

This paper has proposed an ontology driven conceptual model for NoSQL based DW solutions, which is independent of physical level implementation. The proposed conceptual model defines the formal semantics of the related DW concepts in NoSQL based solutions. The novelties of the proposed work are manifolds. Besides proposing a systematic methodology for implementation of NoSQL based DWs, it facilitates, (i) a generalized and rigorous formal conceptual model that can be transformed towards different kinds of NoSQL databases; (ii) handling dimension hierarchies at different granular levels; (iii) realization of flexible characteristics of NoSQL based DWs by de-normalizing both contextual and numerical data; (iv) multiple implementation strategies of data

cubes and efficient visualization techniques over NoSQL based databases; and (v) realization of traditional DWs when ordering and modality of distinct relationships are strictly set to *1* and *Inverse Containment* relationships do not exist. Validation and performance evaluation of the proposed conceptual model will be an important future work. Further, automated transformation mechanism from proposed conceptual model into specific physical databases will also be a crucial future task.

References

[1] Chevalier M., Malki M. El, Kopliku A., Teste O. and Tournier R. (2015). Benchmark for OLAP on NoSQL technologies comparing NoSQL multidimensional data warehousing solution. In 9th Int. Conf. on Research Challenges in Information Science (RCIS), IEEE, pp. 480-485, Athens.

[2] Bicevska Z. and Oditis I. (2017). Towards NoSQL-based Data Warehouse Solution, In ICTE 2016, Procedia Computer Science, Riga Technical University, pp. 104-111, Latvia, DOI: 10.1016/j.procs.2017.01.080.

[3] Sarkar A., Choudhury S. and Debnath N. C. (2012). Graph semantic based design of XML Data Warehouse: A conceptual perspective. In 10th Int. Conf. on Industrial Informatics (IEEE INDIN), IEEE, pp. 992-997, Beijing, DOI: 10.1109/INDIN.2012.6300839.

[4] Yangui R., Nabli A. and Gargouri F. (2017). ETL Based Framework for NoSQL Warehousing. In Information Systems, European, Mediterranean, and Middle Eastern Conference on Information Systems (EMCIS 2017), Lecture Notes in Business Information Processing, Springer, Cham, pp. 40-53, Coimbra, Portugal, DOI: 10.1007/978-3-319-65930-5_4.

[5] Hecht R. and Jablonski S. (2011). NoSQL evaluation: A use case oriented survey. In Cloud and Service Computing (CSC '11), IEEE Computer Society, pp. 336-341, Hong Kong, China, DOI: 10.1109/CSC.2011.6138544.

[6] Chevalier M., Malki M. E., Kopliku A., Teste O. and Tournier R. (2015). Implementing Multidimensional Data Warehouses into NoSQL. In 17th Int. Conf. on Enterprise Information Systems (ICEIS 2015) SCITEPRESS - Science and Technology Publications, pp. 172-183, Lda, Portugal, DOI: 10.1007/978-3-319-29133-8_6.

[7] Guarino N., Oberle D. and Staab S.(2009) . What is an Ontology?. In Handbook on Ontologies, 2nd ed. S. Staab, R. Studer, Eds. Verlag, Berlin Heidelberg, Germany: Springer, pp.1-17, DOI: 10.1007/978-3-540-92673-3.

[8] Banerjee S. and Sarkar A. (2016). Ontology Driven Meta-Modeling for NoSQL Databases: A Conceptual Perspective. International Journal of Software Engineering and Its Applications, Science & Engineering Research Support Society (SERSC), vol. 10, no.12, pp.41 – 64, DOI: 10.14257/IJSEIA.2016.10.12.05.

[9] Max C., El Malki M., Kopliku A., Teste O. and Tournier R.(2016). Document-oriented data warehouses: Models and extended cuboids, extended cuboids in oriented document. In 10th Int. Conf. on Research Challenges in Information Science (RCIS), IEEE, pp. 1-11, Grenoble, DOI: 10.1109/RCIS.2016.7549351.

[10] Yangui R., Nabli A. and Gargouri F. (2016). Automatic Transformation of Data Warehouse Schema to NoSQL Data Base. In 20th Int. Conf. on Knowledge-Based and Intelligent Information & Engineering Systems (KES-2016), Procedia Comput. Sci., pp.255-264, York, UK, DOI: 10.1016/j.procs.2016.08.138.

[11] Khaled D., Bentayeb F., Boussaid O. and Kabachi N. (2015). Using the column oriented NoSQL model for implementing big data warehouses. In Int. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA), CSREA Press, pp. 469-475, Las Vegas, Nevada, USA.

[12] Scabora L. C., Brito J. J., Ciferri R. R. and Ciferri C. D. A. (2016). Physical Data Warehouse Design on NoSQL Databases. In 18th Int. Conf. on Enterprise Information Systems, Lecture Notes in Business Information Processing, Springer, pp. 111-118, Poland, DOI: 10.5220/0005815901110118.

[13] Santos M. Y., Martinho B. and Costa C. (2017). Modelling and implementing big data warehouses for decision support. Journal of Management Analytics, vol. 4, no. 2, pp.111-129, doi:10.1080/23270012.2017.1304292.

[14] Romero O. and Abelló A. (2010). A framework for multidimensional design of data warehouses from ontologies, Data and Knowledge Engineering, vol. 69, no. 11, pp.1138-1157, DOI: 10.1016/j.datak.2010.07.007.

[15] Dehdouh K., Bentayeb F., Boussaid O. and Kabachi N. (2014), Columnar NoSQL CUBE: Aggregation operator for columnar NoSQL data warehouse. In Int. Conf. on Systems, Man, and Cybernetics (SMC), IEEE, pp. 3828-3833, San Diego, CA, USA, DOI: 10.1109/SMC.2014.6974527.