Ontology Driven Meta-Modeling for NoSQL Databases: A Conceptual Perspective

Shreya Banerjee¹ and Anirban Sarkar²

¹,²Department of Computer Applications, National Institute of Technology, Durgapur, India
¹shreya.banerjee85@gmail.com, ²sarkar.anirban@gmail.com

Abstract

In cloud environment, data intensive applications required to interact with various kinds of databases ranged from SQL (Structured Query Language) to NoSQL. NoSQL databases have wide-ranging physical level data models like key-value, document, column oriented, graph etc. Designing such databases are complex due to absence of well-accepted common meta-modeling concepts for varied physical level data model. Thus, research challenges are exist towards achieving common interpretation of data and their semantics at conceptual level over disparate databases. In this paper, an ontology driven meta-model, called Ontology Driven NoSQL Data Model (ODNSDM), is proposed to conceptualize data representation facets over heterogeneous kinds of databases. The novelty of the proposed approach is to provide common conceptual level abstraction based on semantically enriched formal vocabularies for both NoSQL and SQL databases. The formal vocabularies are further implemented using an ontology editorial tool Protégé based on OWL (Web Ontology Language). Several crucial properties of the proposed model are also prescribed in this paper. In addition, the proposed framework is illustrated using case studies in order to show its practical exhibition. Moreover, step wise algorithms are to map the proposed conceptualization towards SQL and NoSQL based databases.

Keywords: Ontology Driven Meta-Modeling, Schema-Based Data, Schema-Less Data, NoSQL Databases, Common Interpretation, Unified Conceptualization

1. Introduction

In recent days, due to the increasing demand of data intensive cloud based applications, the enormous amount of data set are generated. These data sets are characterized by great diversity of types (Variety); rapid synthesis and flow (Velocity); large size (Volume) and the biases, noise and abnormality in data (Veracity) [22]. Those can be ranged from schema-based to schema-less and from structured to unstructured type. Schema-based data have predefined type or context. It should be able to adhere strictly to some schema. Classical SQL databases are kinds of schema-based databases those manage schema-based data. On the other hand, schema-less data have no pre-defined type or context. Now-a-days, NoSQL databases are used by cloud application developers to harness huge schema-less data. NoSQL databases have flexible schema. These databases are referred as schema-less databases as these imply dynamic schema evolution. NoSQL databases neither require schema definition prior to inserting data nor demand a schema modification when data capture and management needs evolve. Multiple advantages are manifested in NoSQL databases over traditional SQL databases to store large data set. Such benefits are (1) high concurrent reading and writing of data with low latency in a cloud specific systems, (2) strong scalability and availability, (3) lower operational and management cost [10], (4) the ability to replicate and scatter data over many servers, (5) efficient use of distributed indexes and RAM for data storage, (6) the ability to add new
attributes towards data record dynamically, (7) flexible schema support (8) compatible with consistency model CAP [2] (Consistency, Availability and Partition Tolerance) theory and BASE (Basic Availability, Soft-state, Eventual Consistency) principles [24] to comply with distributed system characteristics [17].

Often, cloud based data intensive applications (multiple database applications) have to interact with distinct types of databases like NoSQL and SQL. Consequently, serious difficulties are created towards application developers to communicate with various kinds of databases. The key reason behind this is different physical level data models of heterogeneous databases such as key-store, document, graph, or column-store [11]. In a key-store database (for example, Riak [13]) every data item has an associated key. In Document store database (for example, MongoDB [21]) data are kept in a document, and similar documents are accumulated in a collection. Likewise, in a columnar database (for example, HBase [9]) data are stored within columns, and related columns are combined with a column family. Further, in Graph oriented database (for example, Neo4j [3]) data are represented as vertices and relationships are designated as edges between those vertices.

Data centric application developers thus have to deal with numbers of hindrances in a cloud environment to achieve efficient communication of applications with disparate databases. Such obstructions are, (1) lack of a common standardization among NoSQL databases have made harder to choose the right database by the organization [1, 28], (2) lack of commonly accepted conceptual model for both of NoSQL and SQL have made it difficult to choose the right physical data model for application developers [1, 28], (3) handling poor semantics in knowledge exchange between separate cloud based applications and databases, (4) working with ambiguous and incomplete interpretation of similar data over heterogeneous kinds of databases. Due to these barriers, application developers are forced to adapt the source code for every kind of database interaction and sometimes lead them to start from the scratch [25].

To address these aforementioned issues it is highly required to devise a meta-model description that can be able to conceptualize the discrete facets of various types of databases ranging from SQL to NoSQL (hence from schema-based to schema-less databases). This meta-model should facilitate enriched semantics to interpret heterogeneous data set at conceptual level and should provide a common abstraction of varied kinds of databases despite their different physical level representations. This unifying conceptualization is primarily capable to exploit commonalities between disparate databases. Further, it will facilitate the balance in the differences and variations at physical level implementation of diverse databases. Thus, using this proposed conceptualization application developers will be able to interpret data unambiguously over dissimilar databases. In this way, semantic interoperability is also improved in knowledge exchange between divergent data intensive applications and wide-ranging databases.

In recent years, growing interest has been flourished to use ontology in meta-modeling. Ontology is defined as an explicit specification of shared conceptualization in terms of concepts, relationships among concept and axioms (Nicola, 1991). Ontology driven meta-model will be helpful in general conceptualization of heterogeneous kinds of databases for numbers of reasons. First of all, semantically enriched and interoperable formal representation is provided relating to divergent databases. Secondly, consensus on the unifying conceptualization, provided by conventional meta-modeling for disparate databases, is facilitated [27]. Next, generic, shared, and precise insights from data distributed over wide-ranging physical level data models are acquired [5]. Further, automated reasoning can be conducted over unified conceptualization. Besides these, knowledge obtained through ontology is able to view the skeleton of the studied domain and its conceptual model [8]. Consequently, a consistent generalization for discrete
databases is synthesized. In addition, various kinds of databases can meaningfully communicate and inter-operate independently of their internal constructions.

In this paper, an ontology driven meta-model called Ontology Driven NoSQL Data Model (ODNSDM) is proposed to achieve common abstraction over varied types of databases. This will also lead semantic interoperability among distinct databases. The novelty of the proposed approach lies within the ability for the formal conceptualization of both schema-based and schema-less databases with the support of rich semantics. This unifying formal conceptualization is capable to (1) handle different kinds of databases beneath a universal perception at conceptual level, (2) preserve strong semantics in knowledge exchange amid discrete cloud-based applications and databases, (3) interpret similar data over dissimilar databases unambiguously, (4) add new attributes towards data records dynamically, (5) support availability, replication of data, reusability and flexibility, (6) represent hierarchical, non-hierarchical, symmetric and n-array relationships, (7) deal with disparate database management systems efficiently and (8) conform to CAP consistency model for NoSQL databases and ACID (Atomicity, Consistency, Isolation and Durability) consistency model for SQL databases.

Proposed logical axioms in meta-model are used to express semantics about distinct database organizations and data sets. The proposed logical axioms for ODNSDM are based on mathematical logic. Mathematical logic is a suitable foundation for computational logic [20] as using those logic one may be able to quantify over instances and predicates also. Computational logic is the description of problem which is specified without worrying about the strategy to carry out useful computations on that description [7]. Consequently, the proposed logical axioms in this paper are also capable to quantify over predicates and instances and provide foundation for computational logic. Regarding this, the proposed conceptualization is initially validated using an ontology editorial tool Protégé based on Web Ontology Language (OWL) [14]. OWL has aided in computationally interpretation of those logical axioms as it is a knowledge representation and computational logic-based language. Thus, OWL is capable to provide common semantic meaning of data by binding different diverse systems together [15]. Further, step wise algorithms have been proposed to map the proposed conceptualization towards schema-based or schema-less databases. Moreover, the proposed meta-model is illustrated using suitable case studies.

2. Related Work

Several approaches are available in literature regarding meta-modeling of NoSQL databases. In [25], a unifying data model is described to interact with relational and a variety of NoSQL databases. Authors have signified distinctive elements and database hierarchy of NoSQL and relational databases with common formal representation. But, less suggestions are prescribed about how to model dynamically inserted data in a NoSQL database that have no predefined schema or type. In [17], the data model for NoSQL databases is a class diagram. This data model is built based on data query requirement and stored data structure. But, in this proposal description of schema-less data is absent. In [4], a common data model of NoSQL databases has been described. This data model is used to specify a system independent realization of the application data. The methodology is focusing on providing scalability, performance and consistency. Yet, in this proposition dynamically inserted data into NoSQL databases, their availability and reusability are not exploited. Further, major numbers of approaches are postulating conventional meta-modeling of both NoSQL and SQL databases. But, those are not in the positions to provide formalism that can deal with the semantics of data that is the key requirement for interoperable knowledge exchange between distinct databases.

A number of approaches are present in literature for transformation from SQL to NoSQL databases and vice-versa. But these proposals are barely about conversations
between particular types of SQL and NoSQL. These are not about common conceptualization of both. In [16], a SQL database MySQL and a NoSQL database MongoDB has been compared and a method is proposed to integrate these two brands of databases by adding a middleware between application layer and data layer. This middleware provides metadata that is included with conversion rules to transform from one format to another. In [30], a NoSQL database, MongoDB is modeled using relational algebra. In this framework, some of relational algebra is demonstrated on MongoDB’s collection. In [18], a relational database is transformed towards HBase based on the data model of Hbase. Guiding rules of transformation and different relationships between these two schemas are proposed in a heuristic manner that is further employed as queries for automatically transformation from relational database to Hbase. In [23], a data flow model is described that ranges over a class of higher order relations. A web service implementation of the described model is illustrated using JSONMatch [23]. In this methodology, distinct data stores are modeled as a relation using a set of key-value pairs. However, this model focuses more on formalization of several tasks like filter, group, split, reduce and map other than providing of common abstraction of disparate NoSQL databases.

In some approaches, ontology has been used for meta-modeling of databases. In [19], Virtual Data Space (VDS) is described for Big Data management in the science domain. In this methodology, local ontologies are created from different physical level data stores, and then they are mapped and formed a global ontology. Nevertheless, in this approach, only relational and XML (Extensible Markup Language) data store are used and both are schema-based databases. This framework is not included meta-modeling about schema-less databases. In [26], a graph method has been used to transfer ontology specification in OWL language towards ER (Entity-Relation) diagram of relational data bases. Yet, both of these methodologies are not capable to provide rich formal semantics for universal notion of heterogeneous databases.

Majority of existing approaches are not able to facilitate common abstractions for NoSQL and SQL databases in a rigorous form and using a common set of semantics, so that both kinds of databases can efficiently communicate with each other. To overcome this limitation, an ontology driven meta-model is proposed in this paper. The novelty of the proposed methodology is the representation of formal, and semantically inter-operable meta-model for both NoSQL and SQL databases using distinctive construct types, relationships and some significant properties.

3. ODNSDM: Ontology Driven NoSQL Data Model

The proposed Ontology Driven NoSQL Data Model (ODNSDM) is devised to formalize a common set of constructs and relationships those are necessary to conceptualize both schema-based and schema-less databases. Further, the proposed model has facilitated in common interpretation of heterogeneous schema-less kind of databases such as Key Value Store, Column Oriented database, graph database, document database etc. The proposed conceptualization is composed of set of constructs, relationships and a number of crucial properties of relationships. Discrete relationships and their properties enable ODNSDM to differentiate towards a variety of physical level realization of schema-based and schema-less databases. In addition, ODNSDM is consisting of three inter-related layers. Every layer in the proposed model has their recognizable construct types that make each layer distinct from each other.

3.1 Constructs of ODNSDM

ODNSDM can be realized as a layered organization composed of three main layers – Collection, Family and Attribute. Three layers have their respective construct type - Collection (Col), Family (FA) and Attribute (AT). Attribute is the bottom-most layer of the
meta-model. Attribute (AT) construct types are atomic that cannot be decomposable further. Attributes are the group of all possible same kinds of instances. One or several AT can uniquely identify individual instances. This particular AT can be known as Primary Attribute (PAT). Attribute has some built in types like string or number. Family is the intermediate layer of the meta-model. This layer can be further decomposed into multiple levels. Several semantically related Attributes are gathered together to form the lowest level Family Layer. Number of levels in Family layer can be increased or decreased based on the grouping of semantically related Families according to the semantic requirements of the database. Collection is the top-most layer of the data model. Semantically related upper most level Families are collected together to form a Collection. Constructs of discrete layers and levels are connected with each other using different kinds of relationships. From the top level, the entire database can be viewed as a set of Collections. Figure 1 has provided the description of the layered organization of ODNSDM [1].

3.2 Vocabularies of Relationships Types

Distinct relationships of ODNSDM can be of two types. One is Inter-layer kind relationships, and another is Intra-layer kind relationships.

Inter-layer kind relationships: These kind of relationships can be between dissimilar construct types of two different layers.

Intra-layer kind relationships: These kind of relationships can be between similar construct types of identical layer.

Various relationships such as Containment (Cnt), Inverse Containment (Icnt), Association (AS), Reference (ref), and Inheritance (IH) can be present between disparate or similar layer/levels. Among these, Containment, Inverse Containment and Reference can be both intra-layer and inter-layer kind relationships. Beside, Inheritance and Association can be only intra-layer kind relationship.

Formal semantics of these aforementioned relationships have been specified below. Let, \(sl()\) is a predicate. Arguments of \(sl()\) are construct types (Cmp) and express the fact that those are in identical layers and of similar type. In contrast, \(dl()\) is a predicate that
takes Cmp (construct types) as arguments and articulates the fact that those arguments are in different layers and of dissimilar types.

(i) Containment (Cnt): Containment relations are exhibited between two construct types when one is capable of including or comprising similar or different types of constructs. This relationship can be of both inter-layer and intra-layer kind. The axiom of Cnt is

\[
\forall x (\text{Cnt}(x) \leftrightarrow \exists y \exists^n z (\text{Cmp}(y) \land \text{Cmp}(z) \land ((\text{sl}(y,z) \land \text{lev}(y) \land \text{lev}_{\text{Next}}(z)) \lor (\text{dl}(y,z) \land \text{layer}(y) \land \text{layer}_{\text{Next}}(z)) \land x(y,z) \land \neg(y = z) \land (k(\text{range}(x) = z)) \land (m(\text{domain}(x) = y)) \land ((p(\text{value}(n) = 1) \lor \text{greaterthan}(\text{value}(n), 1)))))
\]

**Explanation:** This axiom has specified that one component (Cmp(x)) may contain another different component (Cmp(z)) of same (sl(y,z)) or different layer (dl(y,z)) using relationship Containment (Cnt(x)). In this axiom, k and m are predicates over functions range(x) and domain(x) those are returning range and domain value of the Containment relationship respectively; p is a predicate over the function value() that has returned number of instances n of component z and greaterthan() is a predicate implying whether the number of instances of z are greater than 1; lev() and levNext() are predicates implying whether arguments of these are belongs to a level and its next lower level of similar layer respectively; layer() and layerNext() are predicates implying whether arguments of these are belongs to a layer or its next lower layer (two different layer) respectively.

(ii) Inverse Containment (Icnt): This relationship binds two construct types when one is dynamically inserted into different or same type of construct. This can also be of intra-layer and inter-layer relationships. Direction of this relationship is opposite to the containment relationship. The axiom of Icnt is

\[
\forall x (\text{Icnt}(x) \leftrightarrow \exists y \exists^m z (\text{Cmp}(y) \land \text{Cmp}(z) \land ((\text{sl}(y,z) \land \text{lev}(y) \land \text{lev}_{\text{Next}}(z)) \lor (\text{dl}(y,z) \land \text{layer}(y) \land \text{layer}_{\text{Next}}(z)) \land x(z,y) \land \neg(y = z) \land (k(\text{range}(x) = y)) \land (m(\text{domain}(x) = z)) \land ((p(\text{value}(k) = 1) \lor \text{greaterthan}(\text{value}(k), 1)))))
\]

**Explanation:** This axiom has specified that one component (Cmp(z)) may dynamically inserted towards another component (Cmp(y)) of same (sl(y,z)) or different layer (dl(y,z)) using relationship Inverse Containment (Icnt(x)). The direction of this relationship is in opposite of Containment relationship. Other notations of predicates and functions of this axiom are similar as Containment relationship.

(iii) Association (AS): Association relationships attach similar types of constructs, which are connected with each other in the purpose of accomplishment of some objectives jointly. Association can be intra-layer relationships. Besides, there is an Associative_Connector (Acon) for connecting n (≥ = 2) numbers of associative constructs. The axiom of AS is

\[
\forall x (\text{AS}(x) \leftrightarrow \exists^y z (\exists (\text{Cmp}(y) \land \text{Cmp}(z) \land \text{sl}(y,z) \land \text{Acon}(l) \land \text{AE}(y, Acon, z) \land x(y,z) \land \neg(y = z) \land (p(\text{value}(n) = 1) \lor (\text{greaterthan}(\text{value}(n), 1)))))
\]

**Explanation:** This axiom has specified that Association relationship between component y and z (Cmp(y) and Cmp(z)) has been accomplished when those have participated in performing a common objective (AE). This relationship should be happened in same layer. Further, in this axiom Acon is an Associative_Connector via which n numbers of components can be joined and take participation in an Association relationship. Beside, notation p, value(), greaterthan() are used for similar purpose as in previous two relationships.

(iv) Inheritance (IH): Inheritance relationship is the linking between two similar types of constructs, when one inherits some properties of another construct. Inheritance can be intra-layer relationship. The axiom of this relationship is
R4: \( \forall i h \exists x \exists y (IH(ih) \leftrightarrow ((Cmp_{DP}(x) \rightarrow \exists y \exists i ((y \in PCM_{DP}) \land i(x, y))) \land (Cmp_{DC}(a) \rightarrow \exists b \exists i ((b \in PCM_{DC}) \land i(a, b))) \land (\text{Subset}(PCM_{DP}, PCM_{DC}) \land sl(x, a) \land ih(x, a))) \)

**Explanation:** This axiom has specified the Inheritance (IH) relationship between parent construct elements (Cmp_{DP}) and child construct elements (Cmp_{DC}). Cmp_{DP} have a set of properties (PCM_{DP}) and Cmp_{DC} have a set of properties (PCM_{DC}). Subset() is a predicate that has represented whether properties of parent element (PCM_{DP}) is subset of properties of child element (PCM_{DC}). Further, \( i \) is connecting parent or child element with their respective properties. Beside, \( sl() \) is representing that components are in same layer.

(v) **Reference (ref):** Reference relationship symbolizes the connection when one construct type can refer other construct types that have the same instances. Reference can be only intra-layer relationship. This is symmetric relationship. In this relationship, the instances of referred construct and referring construct will be same. The axiom of ref is

R5: \( \forall x \forall a((\text{ref}(x) \land \text{ref}(a)) \leftrightarrow \exists y \exists z (\text{Cmp}(y) \land \text{Cmp}(z) \land a(z, y) \land x(y, z) \land sl(y, z) \land ((p(\text{range}(a) = y) \land n(\text{domain}(a) = z))) \leftrightarrow (q(\text{range}(x) = z) \land m(\text{domain}(x) = y))) \land (k(\text{In}(y)) = l(\text{In}(z)))) \)

**Explanation:** This axiom has specified the Reference (ref) relationship between component \( y (\text{Cmp}(y)) \) and \( z (\text{Cmp}(z)). \) The part of the axiom

\[ ((p(\text{range}(a) = y) \land n(\text{domain}(a) = z))) \leftrightarrow (q(\text{range}(x) = z) \land m(\text{domain}(x) = y))) \]

has specified that direction of reference relationship between \( y \) and \( z \) is opposite of the reference relationship between \( z \) and \( y. \) Further, \( k \) and \( l \) are predicates over the function \( \text{In}() \) which is returning the instances of components \( y \) or \( z. \)

(vi) **Has Time (HT):** This relationship represents the connection between construct types and time-duration of their survival. Every construct type is persisting within some time-intervals, which is its existence time. The axiom of HT is

R6: \( \exists i \exists j (HT(i) \leftrightarrow \exists y \exists x ((t \in Tm) \land \text{Cmp}(x) \land i(x, t))) \)

**Explanation:** This axiom has specified HasTime relationship between component \( x (\text{Cmp}(x)) \) and \( time. \) In this axiom, \( Tm \) is the set of times \( t1, t2, \ldots . \)

3.3 Vocabularies of Layers and Construct Types

In this section semantics of individual layer of ODNSDM is formally specified along with their relevant construct types and relationships attached to them.

(i) **Collection:** Collection layer may consist of several Collection (Col) constructs and various relationships between them and lower layer constructs. Relationships in Collection layer can be CollectionContainment (Cnt_{col}), CollectionAssociation (AS_{col}), CollectionReference (ref_{col}), CollectionInheritance (IH_{col}) and CollectionHasTime (HT_{col}). CollectionContainment (Cnt_{col}) relation can be present from a Collection (Col) to a top-most level Family (FA) construct type. CollectionAssociation (AS_{col}) and CollectionInheritance (IH_{col}) can take place between different Collections (Col) constructs. CollectionReference (ref_{col}) can be exist between Collection (Col) and Family (FA) or between Collection (Col) and Collection (Col) constructs. CollectionHasTime (HT_{col}) is persisting between Collection (Col) and its survival duration. Based on the above the axiom set for Collection are as follows.

C1: \( \forall x \forall y \exists \exists \exists \exists (\text{Col}(x) \leftrightarrow (\text{Cnt}_{col}(l) \lor \text{AS}_{col}(e) \lor \text{IH}_{col}(s) \lor \text{ref}_{col}(f) \lor \text{FA}(v)) \)

**Explanation:** This axiom has specified the semantics of Collection. It has implied that if there are several Collections (Col) then there should be relationships such as
Containment (Cnt) [Axiom C2], Association (AS) [Axiom C4], Inheritance (IH) [Axiom C3], or Reference (ref) [Axiom C5] and also construct types Family (FA).

\[ \forall x \exists y \exists z((\text{Col}(x) \land FA(y) \land Cnt(l)) \leftrightarrow l(x, y)) \]

**Explanation:** This axiom has indicated that Containment relationship [Axiom R1] related to Collection should be between Collection and Family.

\[ \forall x \exists y \exists z((\text{Col}(x) \land \text{Col}(z) \land IH(r)) \leftrightarrow r(x, z)) \]

**Explanation:** This axiom has indicated that Inheritance relationship [Axiom R4] related to Collection should be between two instances of Collection.

\[ \forall x \exists y \exists z((\text{Col}(x) \land \text{Col}(y) \land AS(t)) \leftrightarrow e(x, y)) \]

**Explanation:** This axiom has indicated that Association relationship [Axiom R3] related to Collection should be between two instances of Collection.

\[ \forall x \exists y \exists z((\text{Col}(x) \land \text{Col}(y) \land ref(t)) \leftrightarrow t(x, y)) \]

**Explanation:** This axiom has indicated that Reference relationship [Axiom R5] related to Collection should be between two instances of Collection.

\[ \forall x \forall t \exists \text{Stp} \exists \text{Etp} \exists i((\text{Col}(x) \land HT(i) \land (t \in \text{Tm}) \land (\text{Stp} \in \text{Tm}) \land (\text{Etp} \in \text{Tm}) \land i(x, \text{Stp}) \land i(x, \text{Etp}) \leftrightarrow (\text{duration}(\text{Stp}, \text{Etp}) \leftrightarrow \text{existence}(x)) \land (((t < \text{Stp}) \lor (t > \text{Etp})) \leftrightarrow \neg \text{existence}(x))) \]

**Explanation:** This axiom has specified HasTime relationship between Collection and Time (Tm). In this axiom, \( l \) is a predicate over the function duration() returning time intervals between its two arguments which are time boundaries [12] known as starting point (Stp) and ending point (Etp); both Stp and Etp are members of the set, Tm; existence() is a predicate specifying whether the Collection survives in time duration between Stp and Etp or not.

(ii) **Family (FA):** Family layer may be made of numerous Family (FA) constructs and a number of relationships between them and adjoining layer constructs. Relationships in this layer can be Family_Containment (CntFA), Family_Inverse_Containment (IcntFA), Family_Reference (refFA), Family_Association (ASFA), Family_Inheritance (IHFA) and Family_HasTime (HTFA). Former five can be intra-layer/level relationships connecting only Family (FA) construct types. In addition, Family_Containment (CntFA) from Family (FA) to Attribute (AT) constructs, Family_Inverse_Containment (IcntFA) from Family (FA) to Collection (Col) constructs and Family_Reference (refFA) from Family (FA) to Collection (Col) constructs can be inter-layer relationships linking dissimilar construct types. Besides, Family_HasTime (HTFA) is occurring between Family (FA) and its survival time-duration. Axiom set for Family layer and construct type is as follows.

\[ \forall x \exists y \exists z \exists u \exists a \exists b \exists d((FA(x) \leftrightarrow (\text{CntFA(l)} \lor \text{IcntFA}(r) \lor \text{IHFA}(b) \lor \text{refFA}(t) \lor \text{ASFA} \lor \text{AT}(v) \lor \text{Col}(d))) \]

**Explanation:** This axiom has specified the semantics of Family. It has implied that if there are several Families (FA) then there should be relationships such as Containment (CntFA) [Axiom F2 and F3], Inverse Containment (IcntFA) [Axiom F4 and F5], Association (ASFA) [Axiom F7], Inheritance (IHFA) [Axiom F8], or Reference (refFA) [Axiom F6] and also construct types Collection (Col) and Attribute (AT).

\[ \forall x \exists y \exists z \exists t \exists u \exists a \exists b \exists c \exists d((\text{FA}_{ulev}(x) \land \text{FA}_{alev}(c) \land \text{AT}(d) \land \text{CntFA}(l)) \leftrightarrow (l(x, c) \land \neg l(x, d) \land \neg l(c, x))) \]

**Explanation:** This axiom has indicated that Containment [Axiom R1] relationship related to Families of upper–most level (FA_{ulev}) should contain Families of adjacent lower level (FA_{alev}) and should not contain Attribute.
F3: \(\forall \exists c \exists x(((FA(x) \lor FA_{\text{lev}}(x)) \land AT(y) \land Cnt_{FA}(l)) \leftrightarrow (l(x,y)))\)

**Explanation:** This axiom has indicated that **Containment** [Axiom R1] relationship related to **Families** should contain **Attribute** if there are no level hierarchy in **FA**. It has also specified that if the **Families** are of lower most level \((FA_{\text{lev}})\) then those should also contain **Attribute**.

F4: \(\forall \exists d \exists x(((FA_{\text{lev}}(x) \lor FA(x)) \land Icnt_{FA}(s) \land Col(d)) \leftrightarrow (s(x,d)))\)

**Explanation:** This axiom has specified that **Inverse Containment** relationship [Axiom R2] related with **FA** in upper-most \((FA_{\text{lev}})\) level can be dynamically connected towards **Collection** (\(Col\)). It has also specified that if the **Families** have no level hierarchy then all **FA** should be dynamically connected towards **Collection**.

F5: \(\forall \exists c \exists e \exists x((FA_{\text{lev}}(x) \land FA_{\text{alu}}(c) \land Col(e) \land Icnt_{FA}(s)) \leftrightarrow (s(x,c) \land \neg s(e,c) \land \neg s(c,e)))\)

**Explanation:** This axiom has indicated that **Inverse Containment** relationship [Axiom R2] related to **Families** of lower-most level \((FA_{\text{alu}})\) should contain **Families** of adjacent upper level \((FA_{\text{alu}})\) and should not contain **Collection**. In this case, the relationship is intra kind relationship.

F6: \(\forall \exists y \exists w \exists x((FA(x) \land FA(y) \land Col(w) \land ref_{FA}(t)) \leftrightarrow (t(x,y) \land t(x,w)))\)

**Explanation:** This axiom has indicated that **Reference** relationship [Axiom R5] related to **Family** should be between two instances of **Family**.

F7: \(\forall \exists y \exists x((FA(x) \land FA(y) \land AS_{FA}(t)) \leftrightarrow t(x,y))\)

**Explanation:** This axiom has indicated that **Association** relationship [Axiom R3] related to **Family** should be between two instances of **Family**.

F8: \(\forall \exists x \exists y((FA(x) \land FA(y) \land IH_{FA}(r)) \leftrightarrow r(x,y))\)

**Explanation:** This axiom has indicated that **Inheritance** relationship [Axiom R4] related to **Family** should be between two instances of **Family**.

F9: \(\forall x \forall t \exists Stp \exists Etp \exists i(((FA(x) \land HT_{FA}(i) \land t \in Tm) \land (Stp \in Tm) \land (Etp \in Tm) \land i(x,Stp) \land i(x,Etp)) \leftrightarrow ((\text{duration}(Stp,Etp)) \leftrightarrow \text{existence}(x)) \land (((t < Stp) \lor (t > Etp)) \leftrightarrow \neg \text{existence}(x)))\)

**Explanation:** This axiom has specified **HasTime** relationship between **Family** and **Time**. Notations are similar as in axiom C6.

(iii) **Attribute (AT):** Attribute layer may comprise of certain numbers of **Attribute** (\(AT\)) constructs and diverse relationships between them and upper most layer constructs. This layer can have relationships like **Attribute_Inverse_Containment** \((\text{lcnt}_{AT})\) (from **AT** to **FA** construct type), **Attribute_Reference** \((\text{ref}_{AT})\) (between **AT** constructs) and **Attribute_HasTime** \((HT_{AT})\) (between **AT** and its existence time). Axiom set for **Attribute** layer and construct type is as follows.

A1: \(\forall x \exists \exists r \exists d(\text{AT}(x) \leftrightarrow (\text{lcnt}_{AT}(r) \lor \text{ref}_{AT}(t) \lor \text{FA}(d)))\)

**Explanation:** This axiom has specified the semantics of **Attribute**. It has implied that if there are several **Attributes** (\(AT\)) then there should be relationships such as **Inverse Containment** \((\text{lcnt}_{AT})\) [Axiom A2], or **Reference** \((\text{ref}_{AT})\) [Axiom A3] and also construct types **Collection** (\(Col\)) and **Family** (\(FA\)).

A2: \(\forall x \exists \exists d((\text{AT}(x) \land \text{lcnt}_{AT}(s) \land \text{FA}(d)) \leftrightarrow s(x,d))\)

**Explanation:** This axiom has specified the **Inverse Containment** [Axiom R2] relationship related with **AT** will be from **AT** to **Family** or **Collection**.
A3: $\forall x \exists y \exists t ((AT(x) \land AT(y) \land ref_{AT}(t)) \leftrightarrow (t(x, y))$

**Explanation:** This axiom has specified the Reference [Axiom R5] relationship related with AT will be from AT to Family or Collection.

A4: $\forall x \forall t \exists Stp \exists Etp \exists i(((AT(x) \land HT_{AT}(i) \land (t \in Tm) \land (Stp \in Tm) \land i(x, Stp) \land i(x, Etp)) \leftrightarrow ((\text{duration}(Stp, Etp)) \leftrightarrow \text{existence}(x)) \land (((t < Stp) \lor (t > Etp)) \leftrightarrow \neg \text{existence}(x)))$

**Explanation:** This axiom has specified HasTime relationship between Attribute and Time.

A5: $\forall x \exists y \exists d ((PAT(z) \land AT(z) \land IN(c, z) \land IN(d, z)) \leftrightarrow (\neg (c = d)))$

**Explanation:** The axiom has represented that one or several Attributes may be acted as Primary Attribute which may identify unique Attribute contained in a Family or a Collection. Thus, with this characteristics of contained Attribute, Collections and Families can be represented and identified uniquely. In the axiom A5, IN() is representing whether the first argument is the instance of the second argument (Attribute) or not.

### 3.4 Properties of Various Relationships in ODNSDM

Relationships, existing in this data-model, can have several properties to characterize ODNSDM and its corresponding construct types and relationships. Such properties are Multiplicity, Ordering, Modality, Availability and Conditional-Participation. In the next section, these properties are formally described in mathematical logic. Let, $RL$ is a predicate, indicating relationships.

(i) **Multiplicity (Mlp):** Multiplicity defines how many instances of the construct types are taking involvement in the relationship.

**P1:** $\forall x \exists y \exists z ((RL(x) \land Mlp(x)) \leftrightarrow (Cmp(y) \land Cmp(z) \land x(y, z) \land (q(\text{value}(k) \land \text{value}(l))))$

**Explanation:** This axiom has indicated the multiplicity (Mlp) of the component $y$ ($Cmp(y)$) and $z$ ($Cmp(z)$) in the relationship $x$ through the value() of $k$ and $l$. In this axiom, $q$ is a predicate over a function value() that has returned the number of instances ($k$ and $l$) of participating components $y$ and $z$ involved in the relationship.

(ii) **Modality (Mdl):** Modality has defined whether the relationship is optional or mandatory. Two modal logical operators are used to represent modality. One is the possibility operator $\diamond$ and another is the necessity operator $\square$ [6]. Possibility operators manifest that the relationship is optional. In contrast, necessity operators demonstrate that the relationship is mandatory.

**P2:** $\forall x \exists y \exists z ((RL(x) \land Mdl(x)) \leftrightarrow (Cmp(y) \land Cmp(z) \land x(y, z) \land ((\square x) \lor (\diamond x))))$

**Explanation:** This axiom has specified modality (Mdl) of the relationship $x$ between component $y$ ($Cmp(y)$) and $z$ ($Cmp(z)$) through possibility operator $\diamond$ and optional operator $\square$.

(iii) **Ordering (Ord):** Ordering of relationship states that the constructs participating in a relationship are in order or not in order.

**P3:** $\forall x \exists y \exists z ((RL(l) \land order(l)) \leftrightarrow ((\neg Cmp(a) \land Cmp(b) \land Cmp(c) \land (a = b = c) \land ((a, b, c) \land ((\neg (\neg (or(a, b) \land or(b, c)) \rightarrow or(a, c))) \land ((\neg (or(a, b)) \rightarrow \neg (or(b, a))))))$

**Explanation:** This axiom has specified the strict ordering (order) of components ($Cmp(a)$, $Cmp(b)$ and $Cmp(c)$) in a relationship $l$. In this axiom, $or()$ is a predicate that specifies its arguments ($a, b, c$) are in order.
\( P4: \forall l \exists a \exists b \exists c ((RL(l) \land \text{not in order}(l)) \iff (l(a, b, c) \land Cmp(a) \land Cmp(b) \land Cmp(c) \land \neg \text{order}(l))) \)

**Explanation:** This axiom has specified that not ordering (not in order) of components \((Cmp(a), Cmp(b) \text{ and } Cmp(c))\) in a relationship \(l\). In this axiom, order() [Axiom P3] is a predicate that specifies its arguments \((a, b, c)\) are in order.

\( P5: \forall l \exists^k Cmp1 \exists^m Cmp2 ((RL(l) \land \text{order}(l)) \iff (((\text{order}(l) \iff \forall a \forall b(Cmp1(a) \land Cmp1(b) \land l(a, b)) \land (\neg \text{not in order}(l) \iff \forall c \forall d(Cmp2(c) \land Cmp2(d) \land l(c, d)))) \land (\text{not in order}(l) \iff \forall a \forall b(Cmp1(a) \land Cmp1(b) \land l(a, b)) \land (\text{order}(l) \iff \forall c \forall d(Cmp2(c) \land Cmp2(d) \land l(c, d)))))) \)

**Explanation:** This axiom has specified the partial order (order) of two types of components \((Cmp1 \text{ and } Cmp2)\) in a relationship \(l\). In partial ordering several components can be in order and rest may not be in order. The part of predicate order() [Axiom P3] in the axiom has indicated the components those are in order. Likewise, the part of predicate not in order() [Axiom P4] in the axiom has indicated the components those are not in order. In this axiom, order() is a predicate that specifies its arguments \((a, b, c)\) are in order.

\( (iv) \text{ Availability (Avl):} \) Availability of a relationship specifies the time-duration or the time stamps of existence of the relationship.

\( P6: \forall x \exists y \exists z \exists t \exists t_2 \exists t_3 ((RL(x) \land Avl(x)) \iff (Cmp(y) \land Cmp(z) \land x(y, z) \land (t_1 \in Tm) \land (t_2 \in Tm) \land (t_3 \in Tm) \land (\text{existence}(x) \iff (t_1 \lor t_2 \lor (l(\text{duration}(t_1, t_2)))) \land (\neg \text{existence}(x) \iff ((t < t_1) \lor (t > t_2)))))) \)

**Explanation:** This axiom has specified Availability (Avl) of relationships \((RL(x))\). In this axiom, \(l\) is a predicate over the function duration() returning time intervals between its two arguments These two arguments are time boundaries and members of the set, Time \((Tm)\). Further, existence() is a predicate specifying whether the relationship \((x)\) has survived in time duration between \(t_1\) and \(t_2\) (both inclusive) or not.

\( (v) \text{ Conditional Participation (Cnd):} \) This property identifies how many instances of constructs take participation in the relationship based on some conditions. Predicate allof() stipulates participation of all instances of a specific construct in a relationship. Similarly, onlyoneof() and anyof() are predicates demanding only one and alternative options for participation of instances correspondingly.

\( P7: \forall r((RL(r) \land allof(r)) \iff \forall x \forall y (Cmp(x) \land Cmp(y) \land r(x, y) \land q(\text{range}(r) = y) \land p(\text{domain}(r) = x) \land (\neg y))) \)

**Explanation:** This axiom has specified that predicate allof() has implied mandatory participation \((\in)\) of all instances of component \(y\) \((Cmp(y))\) in a relationship between components \(x\) and \(y\).

\( P8: \forall r((RL(r) \land onlyoneof(r)) \iff \forall x \exists ! y (Cmp(x) \land Cmp(y) \land r(x, y) \land q(\text{range}(r) = y) \land p(\text{domain}(r) = x) \land (\neg y))) \)

**Explanation:** This axiom has specified that predicate onlyoneof() has implied mandatory participation \((\in)\) of exactly one instance \((\exists ! y)\) of component \(y\) \((Cmp(y))\) in a relationship between components \(x\) and \(y\).

\( P9: \forall r((RL(r) \land anyof(r)) \iff \forall x \exists y \exists z (Cmp(x) \land Cmp(y) \land Cmp(z) \land r(x, y) \land r(x, z) \land p(\text{domain}(r) = x) \land q(('\text{range}(r) = y) \lor ('\text{range}(r) = z)) \land (\neg y) \lor (\neg z))) \)

**Explanation:** This axiom has specified that predicate anyof() has implied alternative participation of instances of component \(y\) and \(z\) \((Cmp(y) \text{ and } Cmp(z))\) in a relationship between components \(x\) and \(y\) or \(x\) and \(z\).
P10: \( \forall r ((RL(r) \land Cnd(r)) \leftrightarrow (allof(r) \lor oneof(r) \lor anyof(r)) \) 

**Explanation:** This axiom has specified that predicate \( Cnd() \) has implied Conditional-Participation of components in a relationship. These Conditional-Participations can be \( allof() \) [Axiom P7], \( onlyoneof() \) [Axiom P8] and \( anyof() \) [Axiom P9].

In proposed ODNSDM, \( Containment, Inverse \ Containment \) and \( Association \) relationship have all the five properties. However, \( Inheritance \) and \( HasTime \) have two properties – \( Modality \) and \( Availability \); and \( Reference \) has three properties – \( Modality, Availability \) and \( Conditional \ Participation \).

4. Schema-based and Schema-less Database Realization based on ODNSDM

Both schema-based and schema-less databases may have hierarchy of one, two or all of three distinct layers of the proposed ODNSDM and their corresponding construct types. Nevertheless, they are different from each other for possessing some distinguished relationships and their properties.

Schema-based databases have not supported dynamically insertion of data without modifying the existing schema. Further, these kind of databases have not explicitly used time-versioning of data. On the other hand, schema-less databases have supported dynamically insertion of data and also time-versioning. Through, \( Has \ Time \) relationship proposed ODNSDM is able to represent explicit attachment between time and construct types [Axiom R6, C6, F9 and A4]. Beside, with the aid of \( Inverse \ Containment \) relationship proposed ODNSDM is able to realize dynamic insertion of data towards databases [Axiom R2, F4, F5 and A2]. Hence, these two relationships of proposed ODNSDM have not been transformed towards schema-based databases. However, these relationships may be transformed towards schema-less databases. Moreover, data set of schema-less databases are flexible where data set of schema-based databases are strict.

Flexibility of schema-less databases can be represented using \( anyof() \) [Axiom P9] \( Conditional \ Participation \) and \( optional \ modality \) [Axiom P2] of proposed ODNSDM. On the other hand, for interpreting strict regular data set of schema-based databases \( allof() \) \( Conditional \ Participation \) [Axiom P7] and \( mandatory \ modality \) [Axiom P2] of proposed ODNSDM have been used. Further, schema-less databases have not supported joining of distinct databases. Therefore, \( Association \) relationship [Axiom R3] has not been applied towards schema-less databases. Although, this relationship has been transformed towards schema-based databases, as schema-based databases have supported joining.

In this section, with the purpose of realizing the aforementioned differences and also commonalities among schema-based and schema-less databases a couple of algorithm has been proposed. These algorithms have transformed proposed ODNSDM towards schema-based and schema-less databases.

4.1 Algorithm for Transformation from Proposed ODNSDM Towards Schema-based Databases

Schema-based databases possess several characteristics specifically regular data set, whole-part relations, generalization, specialization, referential integrity, n-array relationships, symmetric relationships and joining. An algorithm is proposed here in order to transform proposed ODNSDM towards those aforementioned characteristics of schema-based databases. In this case the algorithm is applied on a relational database.

*Step 1: Collection* of proposed ODNSDM has been mapped towards “tables” of a relational database.
Step 2: Family of proposed ODNSDM has been mapped towards “rows” of that relational database.

Step 3: Attribute of proposed ODNSDM has been mapped towards “columns” of that relational database.

Step 4: Containment relationship has been mapped towards relationship between “table” and “row”; towards relationship between “row” and “column”.

Step 5: Containment relationship has been mapped towards whole part relationship between concepts of a relational database.

Step 6: Primary Attributes of proposed ODNSDM has been mapped towards “Primary Keys” of relational databases.

Step 7: Reference relationship of proposed ODNSDM has been mapped towards relationship realization between foreign and primary key of relational databases.

Step 8: Association relationship can be mapped towards joining between two different tables using reference relationship in foreign and primary key level.

Step 9: Inheritance relationship can be mapped towards generalization and specialization relationship between “Tables”.

Step 10: Mandatory options be mapped towards compulsory modality of all relationships.

Step 11: Multiplicity and allof() options be mapped towards cardinality of Containment and Association relationship.

Step 12: Strict Ordering be mapped towards sets of tuples.

The proposed ODNSDM can be applied towards Schema-less databases through an algorithm. Here, the algorithm is applied on a MongoDB type of NoSQL database.

4.2 Algorithm for Transformation from Proposed ODNSDM towards Schema-less Databases

Schema-less databases approve distinct traits those are irregular data set, dynamically inserted data, time versioning of data, generalization, specialization, whole-part relations, replication of data, symmetric relationships, both hierarchical and non-hierarchical structures. An algorithm has been prescribed here for the purpose of transformation from proposed ODNSDM towards schema-less databases. In this case the algorithm is applied on a NoSQL database MongoDB [21]. MongoDB is a document database. A record in MongoDB is a document which is composed of field and value pairs. The values of fields may include other documents, arrays and arrays of documents. Further, MongoDB stores documents in collection. This kind of database support high availability, dynamic schema, and replication of data. Beside, MongoDB has stored documents as BSON documents in collection. BSON (Binary JSON) is binary encoding of JSON (Java Script Object Notation) like documents. BSON contain extensions that allows representation of data types that are not part of JSON specification.

Step 1: Collection of proposed ODNSDM has been mapped towards “Collection” of MongoDB database.

Step 2: Family of proposed ODNSDM has been mapped towards “Documents” of MongoDB database.

Step 3: Attribute of proposed ODNSDM has been mapped towards “fields” of MongoDB database.
Step 4: Containment relationship has been mapped towards relationship between “Collection” and “Document”; towards relationship between “Document” and “Field”.

Step 5: Containment relationship has been mapped towards whole part relationship between concepts of a relational database.

Step 6: Inverse Containment relationship has been mapped towards dynamic insertion of (i) “Fields” towards “Documents” and (ii) “Documents” towards “Collections” without specifying its schema.

Step 7: Primary Attributes of proposed ODNSDM has been mapped towards BSON object type in MongoDB database.

Step 8: Reference relationship of proposed ODNSDM has been mapped towards reference relationship between referral “Collection” and nested BSON object type field of referred “Collection” or “Document”.

Step 9: Inheritance relationship can be mapped towards tree structures like parent and child reference, array of ancestor, materialized path or nested sets in MongoDB.

Step 10: Optional modality be mapped towards flexible modality of all relationships in MongoDB.

Step 11: Multiplicity and allof(), anyof() and onlyoneof() options be mapped towards cardinality of Containment relationship.

Step 12: Has Time relationship should be mapped towards relationship between MongoDB document and nested BSON time stamp.

Step 13: Ordered set is transferred towards “Array” and unordered set is transferred towards “Document” in MongoDB.

Further, proposed ODNSDM conforms to consistency models for both kinds of databases. It acts accordance with ACID (Atomicity, Consistency, Isolation and Durability) model for schema-based databases and CAP (Consistency, Availability and Partition tolerance) model for schema-less databases. Atomicity is achieved through Family and Attributes construct types as distinct transaction updates and rollbacks may be happened on a single Family or Attribute construct types. Isolation is fulfilled as there may be individual paths from Collection to specific Family and Attribute. Consistency is accomplished by implying referential integrity using Reference relationships and constraints likely Ordering, Multiplicity and Conditional-Participation. Durability is undertaken as any transaction update may be sustained on Family or Attribute construct types despite any types of database failure. Similarly, Consistency of CAP model is reinforced in ODNSDM by different constraints imposed on distinct relationships and properties of relationships like Modality, Ordering, Conditional-Participation and Multiplicity. Availability of CAP model is supported by Has Time relationships and Availability property. In addition, an ODNSDM schema follows Partition tolerance by being distributed on two or more dissimilar nodes and supporting either Consistency or Availability property of CAP model.

5. Implementation of ODNSDM using Protégé

The proposed meta-model has been implemented in this section using OWL (Web Ontology Language) based ontology editorial tool Protégé [14]. Protégé has facilitated representation of formally expressed axiom set of ODNSDM towards OWL [14] logic. It is composed of a number of reasoners for automated reasoning or inference on ontological theory expressed in OWL logic.

OWL is based on Description Logic (DL) [15]. Using OWL, it is possible to computationally interpret the proposed logical axioms of ODNSDM. Computational logic
has enabled to describe and reason over problem that to be solved [7]. Hence, OWL is designed to exchange and reason with knowledge. It can enable computers to automatically understand structures and meanings of diverse information sources and conduct automatic knowledge inference or reasoning form existing data or documents.

Proposed axioms have described the unifying data model ODNSDM in mathematical logic those may quantify over predicates, functions and instances. Thus, proposed model is also capable to describe and reason over knowledge of different databases. Consequently, OWL will facilitate in implementation of proposed mathematical logic.

OWLVis and OntoGraf are two separate plug-ins of Protégé. They assist to build valid ontological graphs of ODNSDM employing OWL logic. These graphs are useful to visualize vocabulary of proposed model and validate the consistency and unambiguity in heterogeneous interpretation of data over divergent databases through the proposed ODNSDM. Hence, Protégé can be capable of providing initial validation for logical theory of the proposed model.

Three layers and their construct types – Collection (Col), Family (FA) and Attribute (AT) of ODNSDM have been mapped towards Protégé classes - “COLLECTION”, “FAMILY” and “ATTRIBUTE”. Time and Associative_Connector can be specified as “TIME” and “ASSOCIATIVE CONNECTOR”. Further, main six relationships are plotted as object properties in Protégé according to their intra-layer and inter-layer characteristics. Association relationship is signified as “INTRA ASSOCIATION” and Inheritance is denoted as “INTRA INHERITANCE”. Likewise, Reference is also considered towards “INTRA REFERENCE”. HasTime relationship has two parts in Protégé. “HAS START TIME” and “HAS END TIME”. Former is used for connection from every construct type towards their corresponding start time. In contrast, the later is used for the connection from construct types towards respective end time. Beside, Containment and Inverse containment relationships also have two correspondence in Protégé. Containment relationship is mapped towards “INTER CONTAINMENT” and “INTRA CONTAINMENT” relationship. Similarly, Inverse Containment relationships are realized using “INTER INVERSE CONTAINMENT” and “INTRA INVERSE CONTAINMENT” relationships. Level hierarchies of Family can be represented using Containment and Inverse Containment relationships between two Families.
In Protégé, relationships have further sub object properties. “INTER CONTAINMENT” has sub object properties “INTER CONTAINMENT COLLECTION”, “INTER CONTAINMENT FAMILY” and “INTER CONTAINMENT ATTRIBUTE”. Figure 2 is a picture of ODNSDM graph obtained through OntoGraf plug-in of Protégé. Classes are visualized as squares. Object properties or relationships between the classes are pictured as dotted edges between the squares. Besides, different properties of relationships like 

- **Multiplicity**
- **Conditional-Participation**
- **Availability**
- **Modality**
- **Ordering**

have been displayed on the graph.

**Figure 2. Ontology Graph of Proposed ODNSDM Obtained through OntoGraf Plug-in of Protégé**
Restrictions of ODNSDM towards schema-based and schema-less databases can also be implemented using Protégé. Distinct construct types and layers of ODNSDM have been specified as “SCHEMA-BASED-COLLECTION”, “SCHEMA-BASED-FAMILY” and “SCHEMA-BASED-ATTRIBUTE” for schema-based databases and “SCHEMA-LESS-COLLECTION”, “SCHEMA-LESS-FAMILY” and “SCHEMA-LESS-ATTRIBUTE” for schema-less databases in Protégé. Figure 3 is exhibiting the inferred graph of the proposed ODNSDM, including schema-based and schema-less database restrictions. The graph is generated by the aid of Protégé plugin OWLViz [14]. This graph realizes “SCHEMA-BASED-COLLECTION”, “SCHEMA-BASED-FAMILY”, and “SCHEMA-BASED-ATTRIBUTE” as subclasses of “SCHEMA-BASED-DATABASE-CONSTRUCTS”. Likewise, it recognizes “SCHEMA-LESS-COLLECTION”, “SCHEMA-LESS-FAMILY” and “SCHEMA-LESS-ATTRIBUTE” as subclasses of “SCHEMA-LESS-DATABASE-CONSTRUCTS”. The inbuilt reasoner HermiT [14] of Protégé infer knowledge that these six classes are also subclasses of “COLLECTION”, “FAMILY” and “ATTRIBUTE” respectively. The graph in Figure 3 is too demonstrating that fact. This graph is presenting the relationship “is-a” between different classes. Classes are represented in this graph as yellow circle.

Ontology graph of Figure 4 has been acquired through OntoGraf plugin [14]. In Figure 4, the separate set of relationships and construct types of ODNSDM for both schema-based and schema-less databases are pictured. This demonstrates individual features of schema-based and schema-less databases. From Figure 4, it is manifested that two distinct relationships of schema-less databases – Has Time (relationships 29-34) and Inverse Containment (relationships 23, 25, and 27) are not present in schema-based databases and Association relationships (relationships 1 and 4) of schema-based databases are not exist in schema-less databases.
6. Illustration of ODNSDM using Case Studies in Protégé

In this section the proposed data model has been elucidated using case studies for both of schema-based and schema-less databases separately.

6.1 Case Study Regarding Schema-based Database

Consider a simple application that wants to keep track of departments and visited hospitals of doctors. A doctor may visit multiple numbers of hospitals. A hospital may have many doctors. In addition, a hospital may have multiple numbers of departments.
and many doctors may be enlisted in a department. The application must keep track of data related to doctors, hospitals and departments. Such data is doctors’ name, address, registration number, specialization, hospital location, department name, etc.

This case study is purely in schema-based nature as it has a regular data set and it adheres to a strict schema. All doctors are only about to visit hospitals. All hospitals have one or more departments and enrolled doctors. There is nothing about irregularity or flexibility in this case study. So, several properties like mandatory Modality, Ordered data and all of Conditional-Participation are necessary along with distinct relationships namely Association, Reference and Containment to represent this case study.

According to this case study, “Doctor”, “Hospital”, and “Department” are detected as Collections in proposed ODNSDM. One more Collection “Doc_Hospital” is used for illustration of n to n association between “Hospital” and “Doctor”. Doctors have information like “doctor_name”, “doctor_registration_no.”, “doctor_specialization” and “doctor_address”. Former three are simple attributes, where the latter is a composite attribute, which is consisting of simple Attributes “doctor_city” and “doctor_state”. In proposed ODNSDM, former three are depicted as Attributes and the last is represented as a Family that contains Attributes “doctor_city” and “doctor_state”. Similarly, Other Collections have some related Families. Such as “Doctor” has Family “Doctor_1”. Inter Containsments relationship are used to represent the connection between Collection

**Figure 5. Ontology Graph of ODNSDM Schemata Representing the Case Study Related with Schema-based Database**
“Doctor” and Family “Doctor_I”. Beside, Association relationships are present between two Collections such as “Doctor” and “Department”. Further, Reference relationships may be present between different Attributes such as “doctor_Registration_No.” and “doctor_ID” as those Attributes represent similar instances. Key elements of this case study are listed below. Collections (Col) are specified in bold letters; Families (FA) are identified in italic letters, and Attributes (AT) are described in non-italic letters. Figure 5 is displaying the ontology graph of this case study implemented in Protégé.

**Doctors (Doctor_I)**

*Doctor_I* (doctor_name, doctor_address, doctor_registration_no., doctor_specialization)  
*doctor_address* (doctor_city, doctor_state)

**Hospitals (Hospital_I)**

*Hospital_I* (hospital_name, hospital_city, hospital_registration_no., Department_Id)

**Departments (Department_I)**

*Department_I* (department_name, department_id)

**Doc_Hospital_set (Doc_Hospital_set_I)**

*Doc_Hospital_set_I* (hospital_registration_no., doctor_registration_no.)

### 6.2 Case Study Regarding Schema-less Database

Consider an application that wants to keep track of the visit records of patients. Let, any patient visits doctor either in a hospital or at a clinic. One patient can visit more than one doctor in multiple clinics or hospitals for same disease or symptoms. Similarly, one doctor can visit many patients in multiple places. The application also keeps track of patient’s symptoms. Number and nature of symptoms can be frequently changed. Further, it must keep track of knowledge related to doctors, patients and visits. Such information are doctor’s name, doctor’s registration number, patient’s name, patient’s identification, patient symptoms, visit date, visit place etc.

In this case study, one patient can visit a doctor either at hospital or at clinic. Further, for same disease one patient can visit either to one doctor or to many doctors. On the other hand, one doctor may visit a patient either at hospital or at clinic or at both. In addition, nature of patient’s symptoms is not predefined and they can be dynamically changed. All of these features of the case study imply that the data set will be highly irregular and required flexible representation of such data set. Hence this case study cannot be illustrated by schema-based databases. Consequently, schema-less databases are required to demonstrate this data set.

To elucidate this case study in ODNSSDM, “Visit” is interpreted as a Collection. “Visit” contains data like “Patient”, and “Doctor”. “Doctor” and “Patient” include data such as “doctor_professional_info” and “patient_health_info”. Further, “doctor_professional_info” and “patient_health_info” is consisting of data elements “doctor_registration_no.”, “patient_symptoms” etc. Moreover, “patient_symptoms” can be consisting of “internal_symptoms” and “external_symptoms”. Therefore, “Doctor” and “Patient” are the top-most Level Families; “doctor_professional_info” and “patient_health_info” are adjacent lower level Families; and “patient_symptoms” are the bottom-most level Families. The attachment between Collection “Visit” and Families “Doctor” and “Patient” has been realized using Inter Containment relationship. Further, the level hierarchy between Families “Doctor” and “doctor_professional_info” or “Patient” and “patient_health_info” are represented by Intra Containment Relationship. Beside, “Visit_date”, “internal_symptoms” and “external_symptoms”, and “doctor_registration_no.”, are Attributes. Inverse Containment and HasTime relationships are required to represent data like “internal_symptoms” or “doctor_clinic” which are inserted in a database dynamically, and has no pre-defined schema. Moreover, the relationship between “doctor_clinic” and “doctor_professional_info” is of Intra Inverse Containment kind as both are Families (similar layer). However, the relationship between
“internal_symptoms” and “patient_symptoms” may be Inter Inverse Containment kind as the former is Attribute and the later is Family. In addition, optional Modality operator, unordered or partial-ordered data and anyof() Conditional-Participation properties are necessary for realizing of irregular data in this case such as “Visit”, which is including information about visit place that can be optional – either a “hospital” or a “clinic”.

In this segment, the main elements of this case study are itemized. Collections (Col) are specified in bold letters. Families (FA) are identified in italic letters. Attributes (AT) are denoted in non-italic letters. Elements within parenthesis are mandatory whether elements within braces are optional. Optional data can be included in a database at different times. In “doctor_professional_info” Family, anyof Conditional-Participation is used to represent alternative options for visit place - hospital or clinic. Figure 6 is displaying the ontology graph of this case study implemented in Protégé.

Visit (Patient, Doctor)
Patient (patient_health_info, patient_personal_info)
Doctor (doctor_personal_info, doctor_professional_info)
doctor_personal_Info (doctype_name, visit_date)
doctor_professional_info (doctor_registration_no., patient_name, anyof({doctor_hospital}, {doctor_clinic})))
doctor_hospital (hospital_name, department)
department (department_id, department_name)
doctor_clinic (clinic_name)
patient_health_info (food_habit, patient_symptoms, doctor_name)
patient_personal_info (patient_age, patient_name)
patient_symptoms ({internal_symptoms}, {external_symptoms})

7. Conclusion and Future Work

Ontology Driven NoSQL Data Model (ODNSDM), proposed in this paper, has provided a set of formal vocabularies for unified conceptualization of both schema-based and schema-less databases. It efficiently supports dynamically inserted data, availability and replication of data, reusability, flexibility, hierarchical and non-hierarchical structures, n-array relationships, symmetric relationships, unambiguous representation of data over disparate databases, precise semantics for knowledge exchange between databases, common abstraction of data over distinct databases, commonalities and differences between schema-based and schema-less databases and both CAP and ACID consistency model. The entire data modeling specification is based on mathematical logic. Initial validation of this data model is aided with an ontology editorial tool Protégé based on OWL which is a computational logic based language.

The proposed ODNSDM is consisting of three main construct types – Collection, Family and Attribute and these construct types can be transformed towards concepts of heterogeneous databases. The concepts of ODNSDM is enriched with different relationships like Containment, Association, Inheritance, Reference, HasTime and Inverse Containment. These relationships have several significant properties such as Multiplicity, Ordering, Availability, Conditional-Participation and Modality. Novelties of the proposed approach is manifolds. Firstly, proposed ODNSDM is capable to handle different kinds of databases ranging from schema-based to schema-less beneath a universal perception at conceptual level. Secondly, proposed methodology is able to preserve strong semantics in knowledge exchange amid discrete cloud-based applications and databases. Thirdly, proposed conceptualization supports dynamically inserted data using Inverse Containment relationship. It has also supported time versioning and availability of data using HasTime relationship; flexibility using properties like Modality, Ordering, Conditional-Participation and relationships such as Inverse Containment and HasTime; reusability of several construct types using either Inheritance relationship or Containment relationships (as Whole-Part) within the levels of Family layer and Collection layer. Fourthly, the approach is also capable to support hierarchical and non-hierarchical structure using Containment and Inheritance; symmetric relationships and replication using Reference relationship and n-array relationships using Association and Reference. Fifthly, through distinct facets, proposed ODNSDM can be able to conform to ACID and CAP model. In addition, diverse database organization hierarchies of both schema-based and schema-less databases can also be derived from ODNSDM. Moreover, a variety of relationships and properties make proposed ODNSDM to model a number of commonalities and variations between schema-based and schema-less databases.

In future, a suitable query language can be designed based on this ontological meta-model vocabularies that can be able to query both schema-based and schema-less databases. Further, mechanism and tools for automatic generation of logical schema for both schema-less and schema-based databases from proposed ODNSDM can be devised. Automated identifications of schema-less and schema-based databases based on ODNSDM, are also prime future objective.
References


Authors

Shreya Banerjee, is presently a research fellow in National Institute of Technology, Durgapur, India. She received her MTech in Software Engineering from same institute in 2013, and BTech in Computer Science and Engineering from West Bengal University of Technology, Kolkata, India in 2008. Her areas of research interests are Software Engineering, and Database Technologies.

Anirban Sarkar, is presently a faculty member in the Department of Computer Applications, National Institute of Technology, Durgapur, India. He received his PhD degree from National Institute of Technology, Durgapur, India in 2010. His areas of research interests are Database Systems, Software Engineering and Cloud Computing. His total numbers of publications in various international platforms are about 100. He has active research collaborative with several Institutions in India and USA. He has also served in the committees of several international conferences in the area of software engineering and computer applications.