A Novel Method to Transform Relational Data into Ontology in the Bio-medical Domain

V.Rajeswari* Dharmishtan K. Varughese** Aravind Selvan***
*Asst. Professor, Karpagam Collegge of Engineering, Coimbatore, India
**Professor, Karpagam College of Engineering Coimbatore, India
*** Software Engineer, CA Technologies, India Technology Center, Hyderabad, India
*rajeswari.vp21@gmail.com ** dr.dharmishtan@gmail.com ***aravind646@gmail.com

***Abstract*—Bio-Medical knowledge that has been gathered over decades, is unique in nature since large variety of data models are used in the medical field. Since this domain forms a very important aspect in the well being of the society, measures are afoot to organise the knowledge in a systematic way for interoperability and re-usability. We have discussed here an approach which makes it possible to make knowledge base available in medicinal drug industry shareable across platforms and data models. This is achieved through the application of new developments that have taken place in the computer engineering field. The work involves transforming relational data into ontology to help meet the future computing needs. With 'cloud' deemed to be the future computing environment for all, this work, we humbly feel will contribute towards that paradigm.**

Keywords- Bio-Medical Data, Drug Database, Data Management, Heterogeneous Data, OWL, RDF, Semantic Web, World Wide Web, Ontology

I. INTRODUCTION

Semantic Web is an extension of the World Wide Web in which information is associated with a well-defined meaning. Semantic Web intends to create a universal medium for information exchange by giving semantics to the content of documents on the web in a manner understandable by machines. The focus of this paper being the medicinal drug industry, where formulations of same chemical bear different trade names, semantics have a very vital role. Ontology, the tool that focuses on the semantic layer of knowledge base, is the back bone of our work. Application of Ontology makes it easier to find the required information with high relevance. The vision of our research is to address the issues of interoperability and retrieval of relevant information from the various heterogeneous information sources of a given domain.

II. RESEARCH ISSUE

Reputed and large drug manufacturing companies, normally have brands and formulations based on their popularity in the market and the patterns they own. Smaller companies using the same chemical formulations and base products are not allowed to have the same brand name. So we find a situation where same drug combination is available in different names. We consider MedGuide India, an organisation providing online data for medicines and drugs and other related medical information for our research work. MedGuide India (Fig. 1) provides a vast amount of data on a very large number of products of companies operating out of India. It also provides fairly good amount of data on basic medical conditions and illnesses with simple prescriptions. We have taken their data base model and applied our algorithm to get an Ontology model out of that.

Ontology formally defines different concepts of a domain and relationships between these concepts. Ontologies are an important resource to deal with semantic heterogeneity. They are used to achieve the semantic interoperability and retrieval of relevant documents. In a domain context, they reflect the relevant knowledge based on enterprise-specific concepts and their relations. We consider Ontologies as building blocks for demand-oriented information supply in networked organizations. With the advent of Semantic Web and recent standardization efforts, Ontology has quickly become the core enabling technology. It is the solution for the knowledge based systems of the future.

Fig.1. MedGuideIndia Home page

Domain heterogeneity comprises differences in naming, scope, encoding and attribute scope, as well as modeling paradigm, ontology and content heterogeneities when viewed in a wider context. The Semantic Web regulates tasks such as sharing, reuse and intelligent processing by computer agents. The key problem which is addressed by Ontology is the semantic interoperability. Ontology itself is an explicitly defined reference model of application domains with the purpose of improving information consistency and reusability, systems interoperability, and knowledge sharing. The employment of an ontology building method affects the quality of ontology. The use of ontologies is rapidly growing since the emergence of the Semantic Web. To date, the platform of web ontologies availability continues to increase at a phenomenal rate. We have discussed here an approach which makes it possible to make knowledge base available in medicinal drug industry shareable across platforms and data models.

III. PROPOSED METHODOLOGY

In order to achieve flexible mapping and high re-usability, we present our approach into four separate modules. The first module deals with understanding of the structure of the relational database and its meaning. The Second module deals with conversion of RDBMS into Ontology using the GIO Algorithm. The mapping process starts by detecting some particular cases for tables in the database schema. According to these cases, each database component (table, column, constraint) is then converted to a corresponding ontology component (class, property, relation). The set of correspondences between database components and ontology components is conserved as the mapping result to be used later. In the Third module we describe the different building blocks for creating the Ontology along with the properties. The Fourth module deals with mapping of Graphs, Relational Database and Ontology.

IV. ADOPTED FRAMEWORK FOR MAPPING THE RDBMS TO ONTOLOGY

Databases include conceptual models and information resources that together can be taken as the conceptualisation repository of ontology [4], based on the analyses of the formal corresponding relationships between relational databases and OWL Ontologies. A relational database contains several tables, a table contains several fields and records are the collection of a field's value, whereas OWL ontology contains several classes. A class contains several properties and instances are the collection of property values. The formal corresponding relationships between tables, fields and records in relational databases and classes, properties and instances in OWL ontologies make it possible to convert one schema to another [4]. The corresponding relationships between relational database components and ontology components are shown in Fig.2 and Fig. 3.

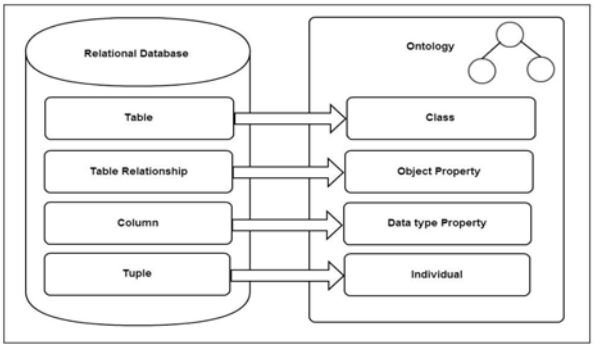

Fig. 2: Relational database and corresponding ontology components

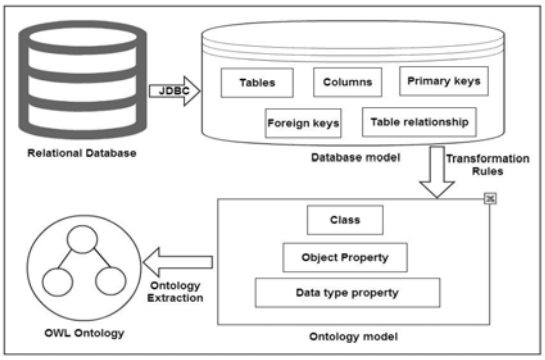

Fig. 3: Construction of ontology from a relational database.

The use of existing relational databases to generate ontology automatically is the main objective of the proposed approach, in order to reduce the manual tedious work, save developing time and improve the efficiency of ontology. The building of local ontology architecture from a relational database is shown in Fig.2.

Construction of ontology from a relational database includes the following steps:

- Extraction of metadata from the relational database by Java database connectivity components
- Analysis of the metadata from Step 1 and transfer of the database model to the ontology model by transformation rules
- Transfer of the ontology model to the OWL ontology.

V. BUILDING BLOCKS OF ONTOLOGY

The representation model should be developed before creating the real-time ontology. The representation model should be general, expressive and compatible. Our proposed representation model is general and more efficient way of building. Ontology formally represents knowledge as a set of concepts within a domain, and the relationships between pairs of concepts. It can be used to model a domain and support reasoning about entities. Ontology provides a shared Vocabulary, which can be used to model a domain, that is, the type of objects and/or concepts that exist, and their properties and relations [6].

Ontologies are the structural frameworks for organizing information and are used in artificial intelligence, the Semantic Web, systems engineering, software engineering, biomedical informatics, library science, enterprise bookmaking, and information architecture as a form of knowledge representation about the world or some part of it. The creation of domain Ontologies is also fundamental to the definition and use of an enterprise architecture framework. The various phases of building Ontology are detailed below.

A. Phase 1: Proposed Algorithm for the Ontology Mapping

We propose a GRO (Graph_Realtion_Ontology), an algorithm which depicts the following functionalities (Fig. 4).

- The Graph is created having www.medguideindia.com as a root node. Drugs, Health_Insurance and Immunization are the children of root node. The identified Node is Brand. The Drug is the parent node for the node Brand.
- If Brand is the identified node, then the function *databaseToOwl()* is called, else it check for the next node. Then it will check for the leaf of Brand in the Graph. Every leaf will become the attribute of the table Brand
- The identified node has converted to a relation, then it is transferred as rdf:resource in the OWL file. Each value of the attribute becomes the Annotation of the OWL File

```
procedure File
createOntology(Graph,
nodeName)
create a queue Queue
root ← Graph.getroot()
enqueue root onto Queue
mark root

while Queue is not empty:
   t ← Queue.dequeue()
   if (t == nodeName):
      return databaseToOwl(t)
for all edges edge in
Graph.adjacentEdges(t) do
   nextVertex ←
adjacentVertex(t,edge)
```

```
if nextVertex is not marked:
      mark nextVertex
      enqueue nextVertex onto Queue
return null
end
procedure File databaseToOwl(String
tableName)
table ←
database.getTable(tableName)
String owlstr ← ""
owlfile ← new owlfile
for each column c in table.columns do
   owlstr.append("<owl:Class
rdf:ID=")
   owlstr.append(c.toString)

owlstr.append(">\n<rdfs:subClassOf
   rdf:resource=\"#" + table.name)

owlstr.append("\"/>\n</owl:Class>")
end
```

```
for each row r in table do
   for each column c in
table.columns do
      owlstr.append("<fo:")
      owlstr.append(c +
"rdf:resource=\"" + r.value
            + "\"/>")
      r.next
end
end

if owlstr is not empty
   owlfile.write(owlstr)
   return owlfile
return null
end
```

Fig. 4: GRO Algorithm

B. Phase 2: Implementation using Relational Algebra

1) Data Structure for Ontology

Ontology can be defined using graph (Fig.5) formalism. In the following work we define an ontology *O* as a directed labeled graph *GO = (N, E)* where *N* is a finite set of labeled nodes and E is a finite set of labeled edges.

An edge e is written as a triplet *(n1, α, n2)* where *n1* and *n2* are members of *N* and *α* is the label of the edge. The structure of graph consisting from [7]:

- A set of concepts. They are referred vertices in the graph);
- A set of relationships connecting concepts. They are the directed edges of a graph;
- A set of instances assigned to a particular concepts. The data records are assigned to concepts or relation.
- The domains and data types elements are included
- ER model can be defined as
  - *ER = (E, A, D, R, DT)* where *E* - set of entities, *A* – set of attributes, *D* - set of domains, *R* – set of relationships, *DT* – set of data types.

We have adopted graph transformation language for our research work. Node addition, edge addition, node deletion, edge deletion and abstraction are the basic operations of the language. The Node addition and Edge addition are implemented in our work.

i) Node Addition : [7] Given the graph G, a node N and its adjacent edges *{(N, αi, mj)}* to add, the node addition results in a graph *G'=(M', E')* where *M'=M U N and E' = E U {(N, αi, mj)}*.

ii) Edge Addition: [7] Given a graph G and a set of edges *SE = {(mi, αj, mk)}* to add the edge addition operation *EA [G, SE]* results in a graph *G'= (M, E')* where *E'=E U SE*.

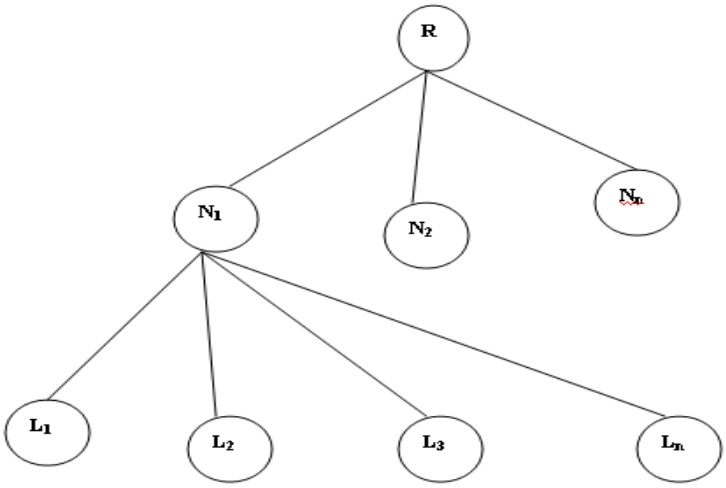

Fig. 5: Graph for Ontology

Let,

a) R = Root of the Graph,

b) $N_1, N_2 \ldots N_n$ = Nodes of the *R.*,

c) $L_1, L_2, L_3 \ldots L_n$ = Nodes of the $N_1, N_2 \ldots N_n$

The first node is the union of the leaves corresponding to the Node One:

$$N_1 = L_1 \cup L_2 \cup L_3 \cup \ldots \cup L_n$$

The root is the combination of the Nodes:-

$$R = N_1 \cup N_2 \cup \ldots \cup N_n$$

C. Phase 3: Implementation of RDBMS Schema for Ontologies

It is a well known fact that there are large quantities of existing data on the web stored using relational database technology. This information is often referred to as the Deep Web as opposed to the surface web comprising all static web pages. The RDBMS is implemented using the Relational Algebra Calculus. The relational algebra is a procedural query language. It consists of a set of operations that take one or two relations as input and produce a new relation as their result.

1) Fundamental Operations used in the Relational Algebra (Fig. 6)

A basic expression in the relational algebra consists of either one of the following:

- A relation in the database
- A constant relation

Let *E1* and *E2* be relational-algebra expressions; the following are all relational-algebra expressions:

E1 U E2, E1 – E2 ,E1 x E2
$\sigma_p(E1)$, *P* is a predicate on attribute in *E1*
$\prod_s(E1)$, *S* is a list consisting of some of the attribute in *E1*
$\rho_s(E1)$, *X is the new name for the result of E1*

Fig. 6: Relational Algebra Expressions

- The select, project, and rename operations are called *unary* operations, because they operate on one relation.
- The other three operations (union, set difference, Cartesian product) operate on pairs of relations and are, therefore, called *binary* operations.

i) Select Operation (σ): The select operation selects tuples that satisfy a given predicate. Symbol σ is used to denote the select operator. Predicate appears as a subscript to σ and argument relation in parenthesis.

ii) Project Operation (Π): The project operation selects certain columns from a table while discarding others. It removes any duplicate tuples from the result relation. The symbol *Π* (pi) is used to denote the project operation. Attribute list to be projected is specified as subscript of *Π* and *R* denotes the relation.

iii) Rename Operation (ρ): The relation or the attributes or both can be renamed in the relational algebra. The general rename operation can take any of the following forms: The symbol ρ (rho) is used to denote the RENAME operator.

iv) Set Difference Operation (-): To find tuples that is in one relation but is not in another. The two condition of union operation also apply for set difference.

- Relation1 - Relation 2

v) Cartesian-Product Operation(X): Cartesian product is also known as CROSS PRODUCT or CROSS JOINS. Cartesian product allows us to combine information from any 2 relation.

- Relation1 ***X*** Relation 2

vi) Set Intersection Operation:

The result of intersection operation is a relation that includes all tuples that are in both Relation1 and Relation2. The general form of the intersection operation will be as in Fig. 7, where relations are indicated as R.

ρ_S (new attribute names) (R) → renames both the relation and its attributes
ρ_S (R) →renames only the relation
ρ (new attribute names) (R)→renames only the attribute name

Fig. 7: Set Intersection Operations

2) Applying Relational Algebra to the RDBMS

The drug is the subclass of www.medguideindia.com. The Project operation is applied to the drugs. It is the union of Brand, Manufacturer and Generic.

Case 1: Drugs =

∏Brand_brand,Category,Constituents_per_unit,Manufacturer_brand,Package_per_unit,Price_per_unit,PriceInRS_brand,Sno,Type_brand,Unit_brand (Brand) **U**

∏Indication_ContraIndication_Precaution_Sideeffects,MatchedBrandswith_CombinationOfGenerics,MatchedBrandswith_SingleGeneric,Sno_generic (Generic) **U**

∏Address,BrandList(),Email,Fax,ManufacturerName,PhoneNo,S_No,SubDivision,URL (Manufacturer)

Case 2: Manufacturer =

∏Address,Email,Fax,ManufacturerName,PhoneNo,S_No,SubDivision,URL(Manufacturer_br) **U**

∏Brand_Name,Constituents_per_Unit,PackageUnit,Price_per_Unit,PriceInRS,SNo,Type,Unit (BrandList)

3) RDBMS Schema for Ontologies

- **Relation1:MGI**={Drugs, Health_Insurance, Immunization}
  - MGI={Drugs(Brand(Brand_brand,Category,Constituents_per_unit,Manufacturer_brand,Package_per_unit,Price_per_unit,PriceInRS_brand,Sno,Type_brand,Unit_brand),GenericIndication_ContraIndication_Precaution_Sideeffects,MatchedBrandswith_CombinationOfGenerics,MatchedBrandswith_SingleGeneric,Sno_generic),Manufacturer(Address,BrandList(Brand_Name,Constituents_per_Unit,PackageUnit,Price_per_Unit,PriceInRS,SNo,Type,Unit),Email,Fax,ManufacturerName,PhoneNo,S_No,SubDivision,URL),Health_Insurance,Immunization}
- **Relation2:Drugs** = {Brand U Manufacturer **U** Generics}
  - Drugs={(Brand_brand,Category,Constituents_per_unit,Manufacturer_brand,Package_per_unit,Price_per_unit,PriceInRS_brand,Sno,Type_brand,Unit_brand),GenericIndication_ContraIndication_Precaution_Sideeffects,MatchedBrandswith_CombinationOfGenerics,MatchedBrandswith_SingleGeneric,Sno_generic),Manufacturer(Address,BrandList(Brand_Name,Constituents_per_Unit,PackageUnit,Price_per_Unit,PriceInRS,SNo,Type,Unit),Email,Fax,ManufacturerName,PhoneNo,S_No,SubDivision,URL)
- **Relation3:Brand**={Brand_brand,Category,Constituents_per_unit,Manufacturer_brand,Package_per_unit,Price_per_unit,PriceInRS_brand,Sno,Type_brand,Unit_brand}
- **Relation4:Generic=**{Indication_ContraIndication_Precaution_Sideeffects,MatchedBrandswith_CombinationOfGenerics,MatchedBrandswith_SingleGeneric,Sno_generic}
- **Relation5:Manufacturer**= Manufacture_br **U** BrandList
  - Relation5a:Manufacture_br={Address,BrandList,Email,Fax,ManufacturerName,PhoneNo,S_No,SubDivision,URL}

- Relation5b:BrandList={Brand_Name,Constituents_per_Unit,PackageUnit,Price_per_Unit,PriceInRS,SNo,Type,Unit}

D. Phase 4: Global approach to database-to-ontology mapping[7]:

A declarative mapping is a set of explicit correspondences between components of two models. A mapping can be defined at different levels. In our case, it will be defined at the implementation level between a database's SQL description and ontology's implementation (Fig. 8). Furthermore, the intended direction of the mappings is from the database to the ontology, which means that we perform a process of data extraction from the database and we populate the ontology with the extracted information. That is why these correspondences will actually have the following form and not the other way round.

Adopted Property: Concepts *C: = (name, syn-set, A, key-A, key-R),* • *name* : Name of the Concept • *syn-set* : Set of its synonyms, • *A* : Attributes, • *Key* - A :key attributes, • *key-R*: key relationships with other concepts.	***Characteristics of "Object Property"*** • Functional Properties • Inverse Functional Properties • Transitive Properties • Symmetric Properties and • Asymmetric properties • Reflexive properties and Irreflexive properties

Fig. 8: Mapping Constructs

E. Phase 5: Classes and Subclasses (Hierarchy) Creation [2]

If a concept A is a super-class of concept B, then every instance of B is also an instance of A In other words, the concept B represents a concept that is a "kind of" A. The hierarchy depends on the possible uses of the ontology, the level of the detail that is necessary for the application, personal preferences, and sometimes requirements for compatibility with other models [3]. The Classes, Subclasses and the Disjoint Classes are defined in the Ontology

1) Adopted Property: Acyclic Graph

G:=(N,E), where *N*=<*C*> and *E*=<*is-a*>

- *G* is acyclic directed rooted graph. It consists of nodes and edges. Each node is a concept (or instance of a concept). Each edge has "is-a" relation

2) Hierarchy

- OWL classes are interpreted as sets that contain individuals [2]. They are described using formal descriptions that state precisely the requirements for membership of the class. The Classes may be organized into a superclass-subclass hierarchy, which is also known as taxonomy [2]. Subclasses specialize (`are subsumed by') their superclasses.
- One of the key features of OWL-DL is that these superclass-subclass relationships (subsumption relationships) can be computed automatically by a reasoner.The word concept is sometimes used in place of class. Classes are a concrete representation of concepts. In OWL classes are built up of descriptions that specify the conditions that must be satisfied by an individual for it to be a member of the class.

3) Set of Classes and Subclasses in the Ontology

The Drugs, Immunization and the Health Insurance are the three major categories of the MedguideIndia.com

The classes and subclasses of MedGuideIndia (Fig. 9) are:

Drugs,Brand,Brand_brand,Category,Constituents_per_unit,Manufacturer_brand,Package_per_unit,Price_per_unit,PriceInRS_brand,Sno,Type_brand,Unit_brand,GenericIndication_ContraIndication_Precaution_Sideeffects,MatchedBrandswith_CombinationOfGenerics,MatchedBrandswith_SingleGeneric,Sno_generic,Manufacturer,Address,BrandList,Brand_Name,Constituents_per_Unit,PackageUnit,Price_per_Unit,PriceInRS,SNo,Type,Unit,Email,Fax,ManufacturerName,PhoneNo,S_No,SubDivision,URL,Health_Insurance,Immunization

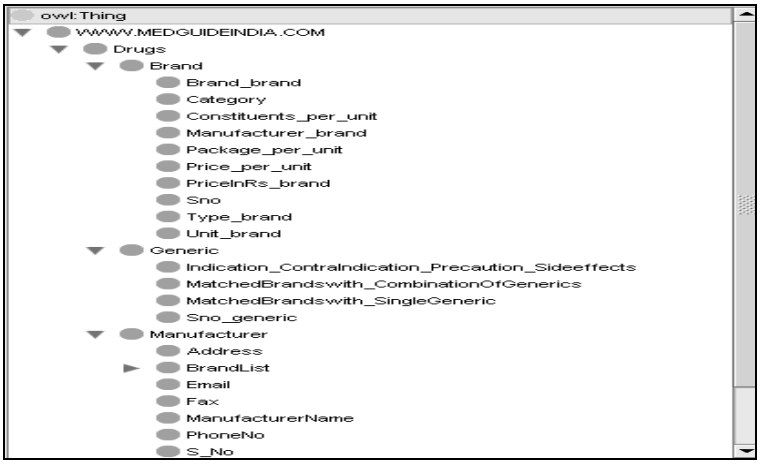

Fig. 9: Taxonomy of MedGuideIndia

F. Phase 6: Domain and Range

Properties may have a domain and the specified range [2] [5]. Properties link individuals from the domain to individuals from the range. A domain ontology (or domain-specific ontology) [1] models a specific domain, which represents part of the world. Particular meanings of terms [1] applied to that domain are provided by domain ontology. Allowed classes for slots of type Instance are often called a **range** of a slot.

1) Adopted Property: Relationship between Concepts

R:= (name, syn-set, domain, range), where

- *name* : Name of the Relationship ;*syn-set* : Set of its synonyms ; *domain* : Domain of the Relationship ;
- *range* : Range of Relationship

G. Phase 7: Attribute and Relationship Extraction

Based on the defined scope we extract relevant attributes for each concept and existing relationships between concepts from the information sources in the context.

1) Adopted Property - 1: Value of the Relationship

V:= (value),

- This feature is used for representation range of one relationship that is a value.

2) Adopted Property - 3: Acyclic Graph

G:=(N,E), where *N*=<*C*> and *E*=<*is-a*>

- *G* is acyclic directed rooted graph. It consists of nodes and edges. Each node is a concept (or instance of a concept). Each edge has "*is-a*" relation

3) Relationship (Properties)

Properties are binary relations on individuals - i.e. properties link two individuals together. For example, the inverse of ***hasOwner*** is ***isOwnedBy***. Properties can be limited to having a single value. They can also be either transitive or symmetric. They are also known as roles in description logics and relations in UML and other object oriented notions. A binary relation is a relation between two things.

4) Named Sub/Super classes of the "Drugs"

- Brand is a **SubClassOf** Drugs–**exactly** 1 **Thing;**Manufacturer is a **SubClassOf** Drugs –**exactly** 1 **Thing**
- Generics is a **SubClassOf Drugs** –**exactly** 1 **Thing**

H. Phase 8: Describing and Defining Classes

Some of the properties are already created. The following properties are used to describe and define our University Ontology classes.

1) Adopted Property: Terms of Ontology

The Ontology term is a combination of concept, [2] [3] attribute, relationship and range of value in the relationship.

T:= (C, A, R, V), where C: Concept or Instance of one concept ;*A*: attribute of a concept;*R*: relationship between concepts ;*V*: value range of one relationship

2) Necessary and Sufficient Conditions

The following are the necessary and sufficient conditions to determine the Ontology

- Primitive and Defined Classes;Automated Classification;Universal Restrictions and Value Partitions
- Creating Individuals and Enumerated Classes;Annotation Properties;Multiple Sets Of Necessary & Sufficient Conditions and Quantifier Restrictions;Combining Existential And Universal Restrictions in Class Descriptions ;Intersection Class, Union Classes and Data Type Properties

I. Phase 9: Axiom -property determination [3]

We determine the key properties for each concept chosen from its attributes and relationships. Key properties define role and main semantic of concept.

1) Adopted Property 5: Value of the Relationship

V:=(value)

- This feature is used for representation range of one relationship that is a value.

Following are the types of axioms, restrictions and cardinality functions to create Ontology.

- Closure Axioms and Covering Axioms ;Cardinality Restrictions ;Qualified Cardinality Restrictions
- *hasValue* Restrictions and *someValuesFrom allValuesFrom* and Universal Restrictions;*hasValue* Restrictions and Cardinality Restrictions;Minimum and Maximum Cardinality Restrictions

J. Phase 10: Synonym determination:

A direct mapping scheme maps terms directly to other terms. Each term can have one or more *synonyms.* so it can map directly to one or more terms and it can be mapped to one or more terms. Formally, a direct mapping scheme has the form: *uris*:*termi* → *urit*:*termj* , where *uris* and *urit* are source and target namespaces.

We specify synonyms of each concept, attribute and relationship. Synonym of each term is other name that is used by other communities for representing the same term. Semantic equivalence, a sample as shown in Table I., can be obtained from synonyms or any vocabulary like WordNet.

1) Adopted Property 3: Attribute of the Concept

A:= (name, syn-set), attribute is defined with a name and a set of synonyms.

- *name* : Name of the attribute;*syn-set* : Set of its synonyms,

Node:= (semantic_value$_1$,semantic_ value $_2$, semantic_value$_3$……semantic_value$_n$),

TABLE I
Semantic Equivalence for nodes

Term	*Semantic Equivalence*
Drug	*{medicine, treatment, preparation, remedy}*
Generic	*{general, broad, common, basic, nonspecific, standard}*
Brand	*{make, product, brandname, variety, kind, sort, tradename, trademark}*
Manufacturer	*{producer, maker, company, firm}*

K. Phase 11: Modeling:

The extracted terms from the above phases are combined to form a Medical Ontology.

1) Adopted Property - 2: Graph

G:=(N,E), where *N=<C>* and *E=<is-a>*

- *G* is acyclic directed rooted graph. It consists of nodes and edges. Each node is a concept (or instance of a concept). Each edge has “is-a” relation.

The completed Ontology of the MedGuideIndia data model is given in Fig.10 for reference.

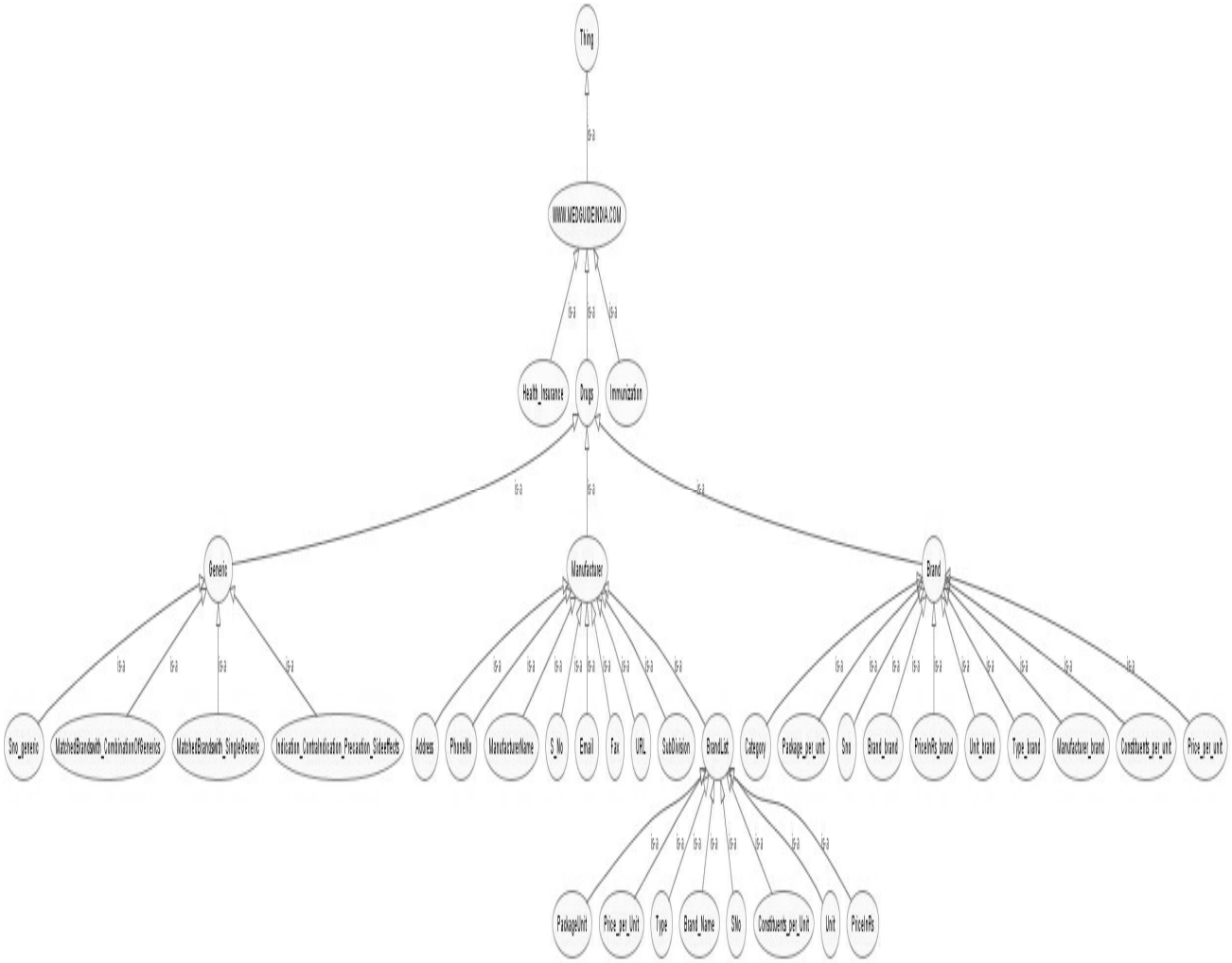

Fig. 10: MedGuideIndia Ontology Model

VI. MAPPING RELATIONAL DATABASE WITH ONTOLOGY

In order to achieve flexible mapping and high usability, we presented our approach into several separate phases. The first phase is to understand the structure of the MedGuideIndia. After that Metadata of the relational schema is extracted into the record-set of the database. Finally, we describe the mapping process for generating the structureand data of OWL document. The node from the GRAPH is mapped to attributes in the RDBMS and it is mapped with Ontology.

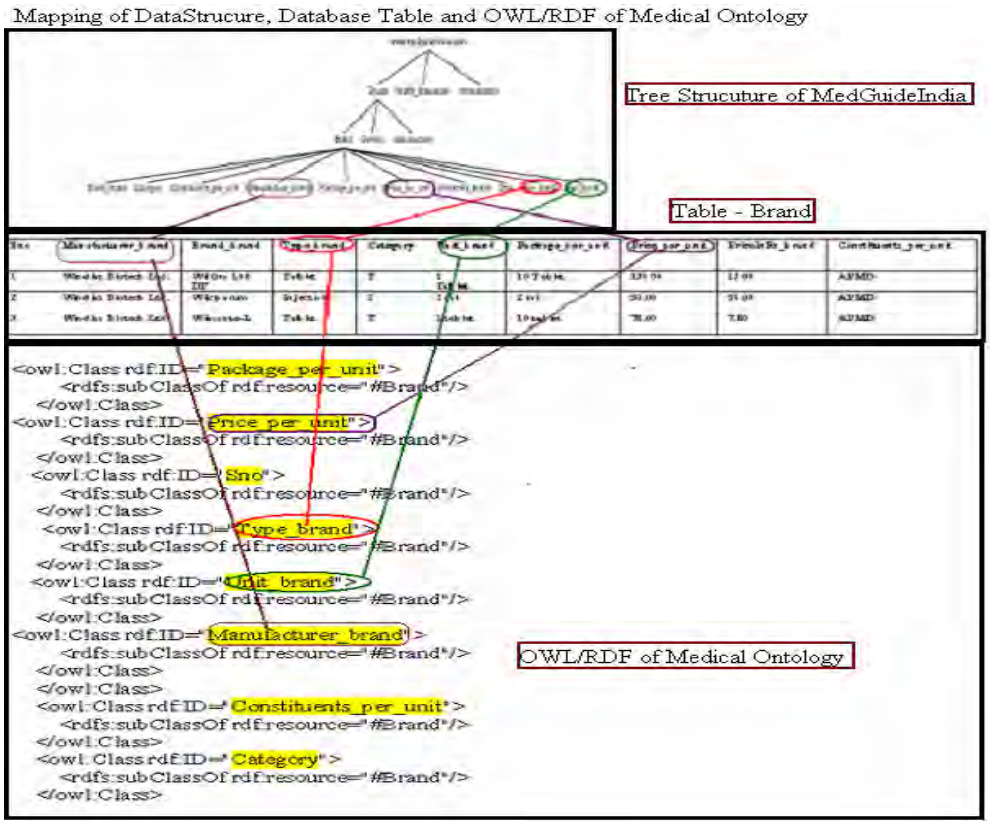

Fig. 11: Mapping of Graph, Relational Data and Ontology

VII. Conclusion And Future Enhancements

The work undertaken in this research focused on building ontology out of a relational data model. The outcome is meant to be used to access knowledge available in the bio-medical domain which has its data repositories with varied data format, using a common platform through Ontology. It will be our endeavor to extend this work to semantically merge different ontology models of the same domain, to obtain a comprehensive global ontology. This will enhance the chances of knowledge systems being more robust providing information with high degree of contextual relevance.

References

[1] Annika Öhgren, Kurt Sandkuhl, *Towards A Methodology for Ontology Development in Small and Medium-Sized Enterprises*, IADIS International Conference on Applied Computing, 2005.

[2] Bruno Bachimont, Antoine Isaac and Raphae Troncy, *Semantic Commitment for Designing Ontologies: A Proposal*, Springer-Verlag Berlin, Heidelberg, 2002.

[3] Fernández López, M., *Overview Of Methodologies For Building Ontologies*, Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5) Stockholm, Sweden, 1999.

[4] Justas Trinkunas, Olegas Vasilecas, *Building Ontologies from Relational Databases Using Reverse Engineering Methods*, International Conference on Computer Systems and Technologies - *CompSysTech'07*

[5] Mike Uschold, *Building Ontologies: Towards a Unified Methodology*, Proceedings of Expert Systems of the 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems, Cambridge.

[6] Sari Hakkarainen, Darijus Strasunskas, Lillian Hella, Stine Tuxen, *Classification of Web-Based Ontology Building Method Guidelines: a Case Study*

[7] Oscar Corcho, Asunción Gómez-Pérez, *A Layered Model for Building Ontology Translation Systems*, Idea Group Publishing, http://www.idea-group.com.