Chapter 4: Integration of neurological data and knowledge:

A systematic approach to extract knowledge from Parkinson's Disease data sources

4.1 Introduction

The abundance of data in different data sources can be extracted for knowledge. The holy grail of biomedical data is to integrate the heterogeneous data sources to enable it to extract knowledge that would give both practitioners and researchers a better insight into the subjects. The exponential growth of such data containing research, clinical, histological, patient and laboratory data in varied formats can be a hindrance to providing significant biomedical correlations that can aid in the discovery of further or new processes. Sometimes these data contained in legacy databases are annotated for easy retrieval of knowledge which, although human readable, is incompatible for machine interpretation [98].

It is therefore essential to understand the interplay between the various disciplines that make up the workings of Parkinson's Disease (PD). The different strands of neurological research like clinical, physiotherapeutic, speech and language therapy, and dietary store disparate and heterogeneous data that needs to be marshaled. Such conceptually different, yet correlated, data should be integrated to provide better information to the researcher. This chapter focuses on the extraction of knowledge from the conceptual layer in the data model and the new concepts suggested by reverse engineering the database. Finally, the ontologies created were enhanced by aligning with a foundational ontology.

4.2 PD Knowledge base System

Large amount of information regarding PD is available from disparate sources like booklets, Internet resources, patient information and bibliography. Medical terms like

Shy Drager Syndrome can differ across neurological communities depending upon their perspective as a neurologist or a therapist. As such, the inherent meaning of the diversity of the terminologies can get lost unless they are collated to form a commonly accepted meaning. Like any other scientific development, the fluidity of the technical terms cannot be avoided with the advancement of knowledge.

While domain experts in collaboration with knowledge engineers help to build the knowledge base, it cannot replace the knowledge of the individual experts. The tacit knowledge that is prevalent in the domain experts is a culmination of experience, skill sets and knowledge that forms a part of human reasoning. It is used in the decision making process when explicit knowledge fails to deliver. As such, neurologists can prescribe medicines to patients based on their personal experience, shared knowledge from their colleagues, patient history and the kind of drugs that may be available locally. Ontologies can help to provide the means by which concepts can be arranged systematically in a semantic repository that traverses globally and across cultural barriers.

The primary focus of this section is to illustrate the ontology for the drug concordance for PD. The subsequent sections of the chapter then illustrate the following problems:

- generating the DDL for a database from ontology
- Extracting concepts and attributes by reverse engineering a database
- Aligning the ontology with a foundational ontology (FOL)

The knowledge based system for PD has been developed using OntoEdit♦ [99]. The PD knowledge management system consists of ontological-based modular knowledge-bases consisting of Symptoms and drugs (PDSymptomsdrugs), Physiotherapy (PDPhysiotherapy), Diet (PDDiet) and Speech and Language therapy (PDSpeechLanguagetherapy). These knowledge bases are described in detail in chapter 6.

♦ www.ontoprise.com; version 2.64 provided by Ontoprise GmbH

The ontology used in this demonstration is that of the symptoms and the corresponding treatments of the different types of PD developed in consultation with the domain experts. In order to add to the flexibility of the development of the ontology, primitive concepts like `Symptoms`, `Treatments` and `ParkinsonTypes` were created so that any new concepts extracted from the discovery of new knowledge can be seamlessly added. Concepts in ontology are referred to real-world entities that encompass the conceptual description of a domain. The subsumption relation of concepts is similar to 'is-a' or inheritance relationship of classes in the object-oriented programming paradigm [62]. The concepts have been classified according to the kinds of patients, the different types of symptoms, the existing healthcare givers, the prevalent types of drugs and the medications, the kinds of hospitals and other people employed in the health sector. The generalization/specialization relationship developed assumes that the criteria developed in a parent concept are automatically inherited by the child concept along with its own unique criteria. Similarly, it is also inferred that instances of a child will also be instances of the parent. Those concepts that are not interlinked are classified as disjoint. Such constraints, in addition to the axioms [95] are used to check the consistency of the ontology. This is exemplified in the case of the two concepts `AlternativeMedicine` and `Pharmacologic`. ***Alternative Medicine*** also known as 'complementary' medicine is medicine that is not associated with conventional medicine [2] while `Pharmacologic` in our context refers to the conventional medicine.

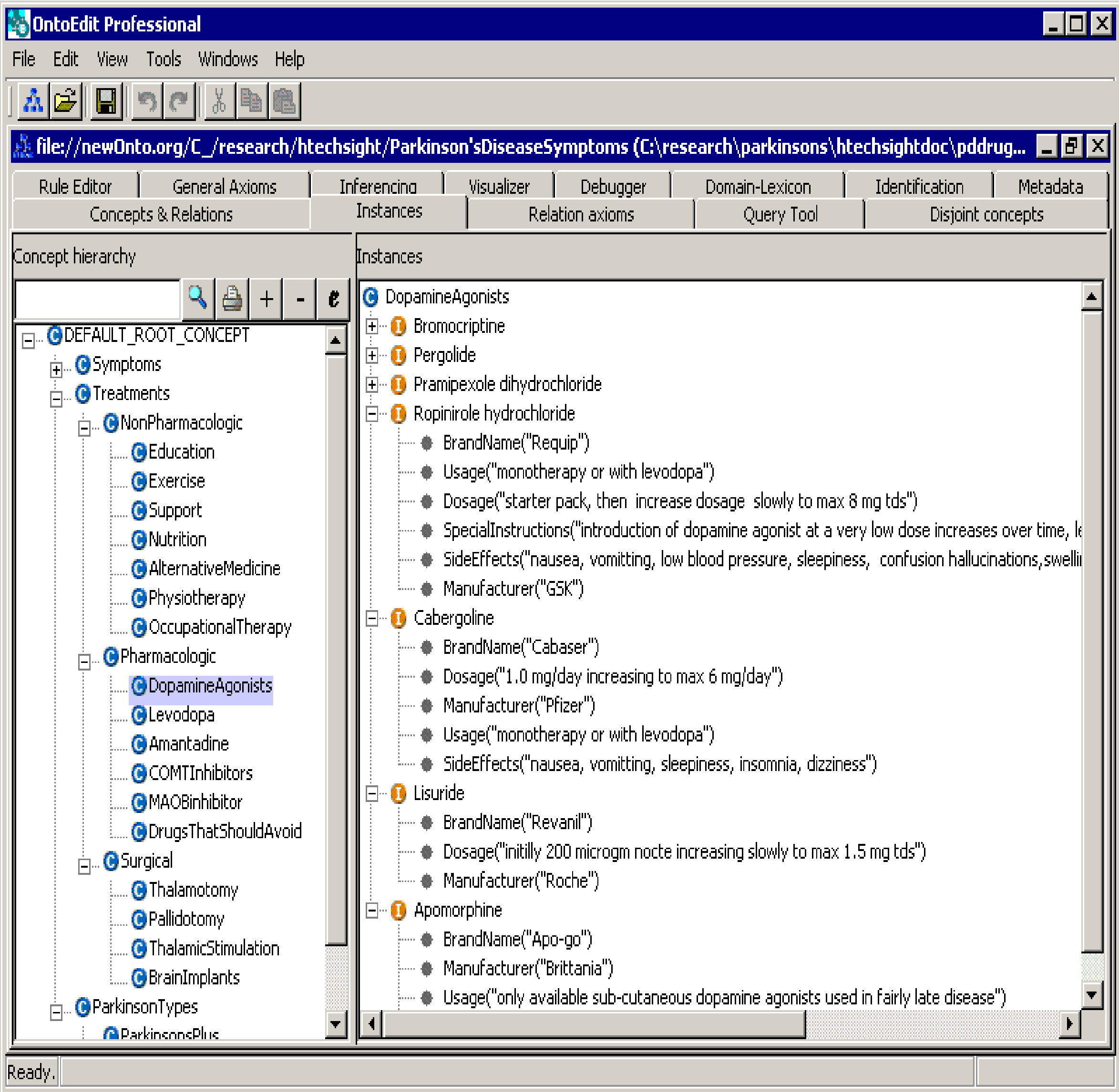

Figure 4.1 The knowledge base of PDSymptomsdrugs

Alternative Medicine involves such complementary medicines as ***Homeopathy*** and ***Naturopathy*** which are sub-classes of `AlternativeMedicine`. As such, there cannot be instances of `AlternativeMedicine` and its sub-classes `Homeopathy` and `Naturopathy` that exists in `Pharmacologic`. This reduces the semantic heterogeneity and makes the knowledge base more precise and compact. Concepts like Consultants and Nurses are also disjoint concepts.

Adding similar terms to the lexicon of the domain can expand the ontology further. The addition of synonyms give added meaning to the established concepts and where necessary we also filter out erroneous synonyms [99]. This would reduce the ambiguity of terms and eliminate any problems when integrating with external databases where such redundancy of data is abundant. Terms like `Atypical Parkinsons` and `Shy Drager Syndrome` are regarded as analogous to `ParkinsonsPlus`.

4.3 Integration of Ontology to Database

The ontology developed for the symptoms and drugs can now be integrated into a new database. This is the knowledge, and hence the vocabulary of the data, that is implemented into the database and not just the data. As such, by acting as the central repository which interconnects different data sources and rules it expedites the integration of the semantics of data and reduces the redundancy of the data.

Peterson **et al** [120] used ontology in the construction of an information system that facilitates the semantic interoperability of different databases. By focusing on the subset of a large ontology and using logic programming they were able to generate an XSB enabled deductive database. The disadvantage of such a method is that currently there is no commercially available deductive database which allows the benefits of concurrent query evaluation and transaction. In our case, we therefore used ontologies from a globally accepted view to generate the database that can be used in the relational format. Subsequently, using the expressive power of F-logic [95], we then show the integration of ontologies to database by querying instances from the semantically enabled database.

Since today's best practices can end up as in-built system rigidities for corporations in the future, ontologies can help remove the future problems of legacy databases by exporting database schema specific to the RDBMS since they are independent of database vendors. While databases like SQLServer♦ and Oracle are built on the basic model of relational tables, each of them has their own extensions of SQL (T-SQL in case of SQLServer, PL/SQL in case of Oracle♣) and particular functions. The

♦ www.microsoft.com/sql/default.mspx

♣ www.oracle.com

automatic generation of the SQL code by OntoEdit specific to the RDBMS allows for the rapid integration of the knowledge to the database. The use of ontology as a formal approach to model such business processes helps both the managers and the IT staff to effectively communicate and hence share the business logic between themselves [100]. Added to this, the semantic heterogeneity that is prevalent in biological [91] databases can be removed by building the ontology first although the syntactic heterogeneity [52] for each of the RDBMS remains. The consecutive E-R diagram generated from the knowledge base for Symptoms and drugs is shown in Figure 4.2.

In this case, the concepts are interpreted as tables while the attributes or relations are regarded as the columns of the table. Unlike an E-R model, which is not an object model, ontologies can read taxonomies and like the E-R model [101] ontologies have the capability to distinguish schema from facts. This enables the facts to be stored in the database. In addition, the ontology does not change should the database need to be altered thereby saving time and reducing maintenance costs. As such, since the data in the database is concurrently available it is automatically accessible to the ontology. This facilitates the seamless integration of data in the database to the instances of knowledgebase.

The concepts, as in Figure 4.1, have instances which spanned under the same URI. The concept `NonPharmacologic` is a superconcept of the 7 concepts of `Education, Exercise, Support, Nutrition, AlternativeMedicine, Physiotherapy` and `OccupationalTherapy.` The concept `AlternativeMedicine` has 4 relations: *holisticModalities, Ginger,GinkgoBiloba* and `coenzymeQ10`. The concept `NonPharmacologic` mapped as the table NonPharmacologic, while the other sub-concepts mapped to the corresponding table names. The 4 relations of `AlternativeMedicine` mapped to the corresponding

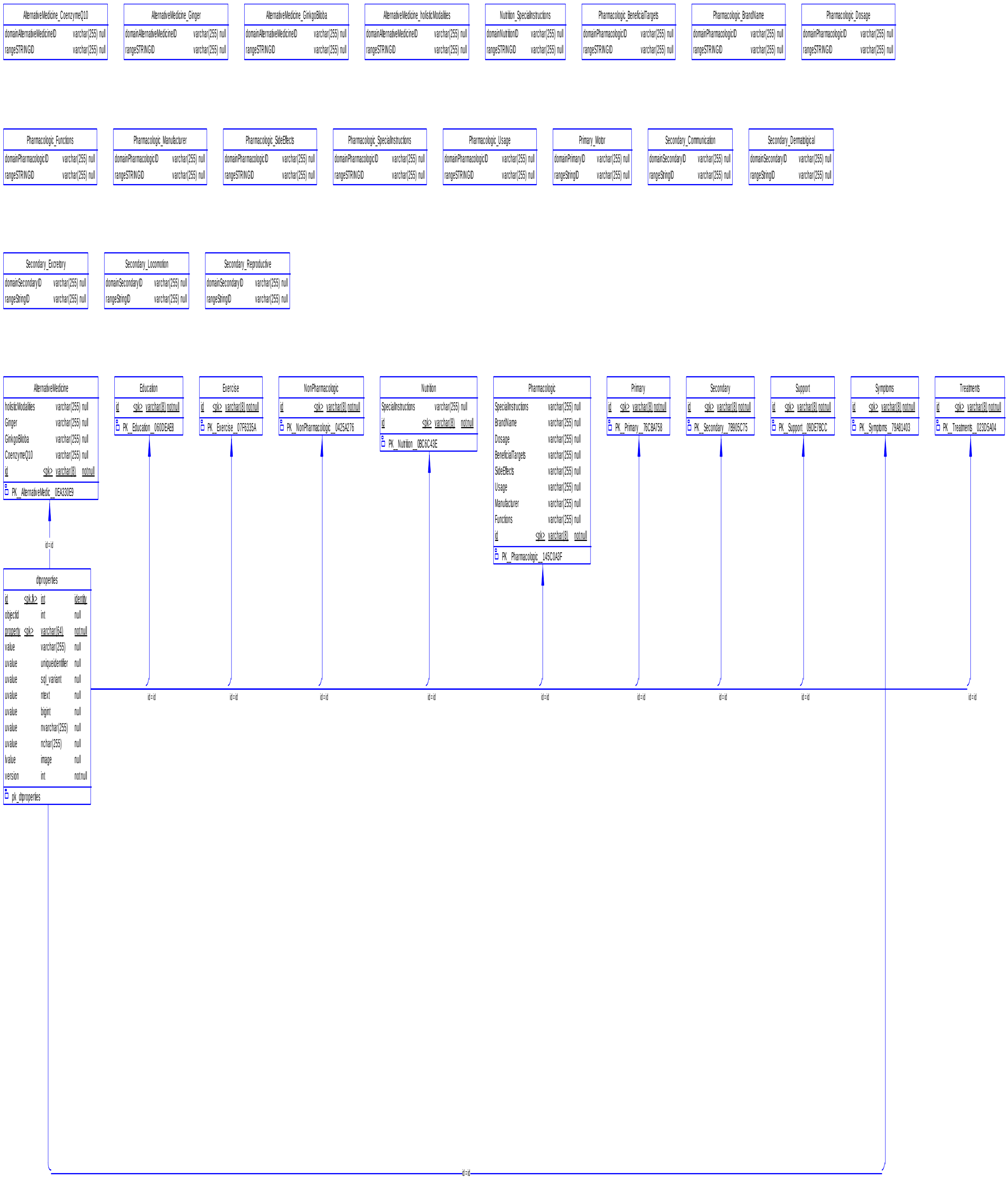

Figure 4.2 E-R diagram generated from PDSymptomsdrugs knowledge base *(large copy in Appendix A)*

column names. Each of the relations had a constraint from 1 to n, a range of type String and the domain was of type `AlternativeMedicine`. The relations are mapped to the database as separate tables with the domain and the range as the 2 columns for each of the tables. The *holisticModalities* attribute is mapped as

AlternativeMedicine_holisticModalites with 2 columns of domainAlternativeMedicineID and rangeStringID. It was also noticed that each of the concepts generated a separate table. This causes limitations in the database schema. As concepts are added or deleted during the lifetime of an ontology development, transactional processing that is the bedrock of a relational database schema is significantly impeded.

Using F-logic [95], it was possible to retrieve the instances from the database. If we want to find out the kind of holistic modalities that are present in the database from the knowledge base containing the ontology for the Symptoms and drugs then we can execute the following query:

FORALL X <-sql("select holisticModalities from AlternativeMedicine",[X],"mssqlserver2000","pddrug","ontohost","sa","pwd").

Output:

Evaluating the query: FORALL X <-sql("select holisticModalities from AlternativeMedicine",[X],"mssqlserver2000","pddrug","localhost","sa","pwd").
"holisticModalities"

Such seamless integration of ontology to the database facilitates the presentation of the content to the user. Ontologies can then not only use the results to be ranked and filtered according to the semantics but also to inter-link different data sources [102].

4.4 Extraction of Ontology from Database

In order to integrate the heterogeneous data sources into a valid knowledge base that is shareable, ontology provides the capability to define the formal semantics that allows for the easy retrieval from different data sources. The need to integrate the data sources into ontology can be due to several reasons: new software systems need to be integrated with legacy systems and mergers of companies can require integration of software infrastructure [121]. The integration of information by semantic means can exist at various levels. They can be at the hardware (computer platforms, networks, routers), software (different Operating Systems, different application versions), syntactic (SQL languages like T-SQL or PL/SQL) and at the semantic levels (ontology). The focus of this thesis is on the integration at the semantic level.

Presently, when data needs to be migrated from one database vendor to another, migration scripts are written which are specific to the format so that data migration is possible. However, such scripts need to be constantly updated with corresponding changes in database version. As such, with this type of integration it is not possible to scale up since a script needs to be constantly maintained and upgraded.

In case of semantic integration, an explicit mapping between different concepts is possible as shown in chapter 6. This mapping is possible because a formal specification of the concepts is shared across the same domain. The concepts are expressed in a formal language that can also facilitate the corresponding semantics to be inherited by concepts expressed in less formal language. Thus by extracting concepts from the conceptual layer of the database schema, it is possible to give them a formal specification by aligning them with the concepts of an ontology. This is shown in section 4.5.

Both the conceptual layer of the database schema and ontology share some common characteristics. They both deal at the same abstraction level and the domain that they address is independent of any implementation. Since a semantic approach to integration is used, this allows the meaning of the data to be captured. As such, the entities of the different data sources can be extracted as concepts that can then be linked to each other by mapping to a central ontology.

In this section the overall design process of constructing a domain specific ontology by extracting the schemas from the logical data model is demonstrated. Ontologies are constructed by reverse engineering the E-R (Entity Relationship) model from the underlying schema containing tables, columns and relationships. The schema has been used in designing the annotated physiotherapy database (PDPHDB) for PWPD (People With Parkinson’s Disease). Parkinson’s Disease Clinical Database (PDCDB) and

PDPHDB were abstracted to generate the ontology. The E-R diagrams for these databases have earlier been shown in Figures 3.1 and 3.2 respectively in chapter 3.

The E-R model of PDCDB is a data model that describes the drug concordance, caregiver quality, the neurological examination, the symptoms, side effects, medication, drug exposure and the toxicity of the drugs on patients. It is used to seek data from patients. The PDPHDB is a conceptual model that is annotated for services, both external and internal, the techniques involved in assessing the patient by physiotherapists and the outcomes of the services rendered by the physiotherapists. E-R models, like ontologies, distinguish schema from facts. This makes it possible to reverse engineer the semantics of the schema of the E-R models into ontology. Although the essential components of reverse engineering tools comprise an information extractor, a repository and an analyzer/visualiser [103], each of the tools represent differently depending on the knowledge manifestation of the software held by the tool[104].

Using OntoEdit, the database schema was imported from MS SQLServer 2000. The database tables were transformed into concepts while the tables holding the columns were translated into attributes for the corresponding concepts. The Services table is linked to the table IndividualService by the referential integrity constraint and the import of the table into ontological concepts and attributes of `Services` is shown in figure 4.3. Columns like organization and acute_problems were mapped as attributes for the concept `Services.`

Integration of the neurological data sources into ontology is an iterative process that reduces the time taken to interact with domain experts in building the ontology. It reduces the time for the interaction between the knowledge engineer and the subject-matter experts (SMEs) who although knowledgeable in their respective fields may not, or need not, be proficient in the technical know how of ontology development. A conceptual

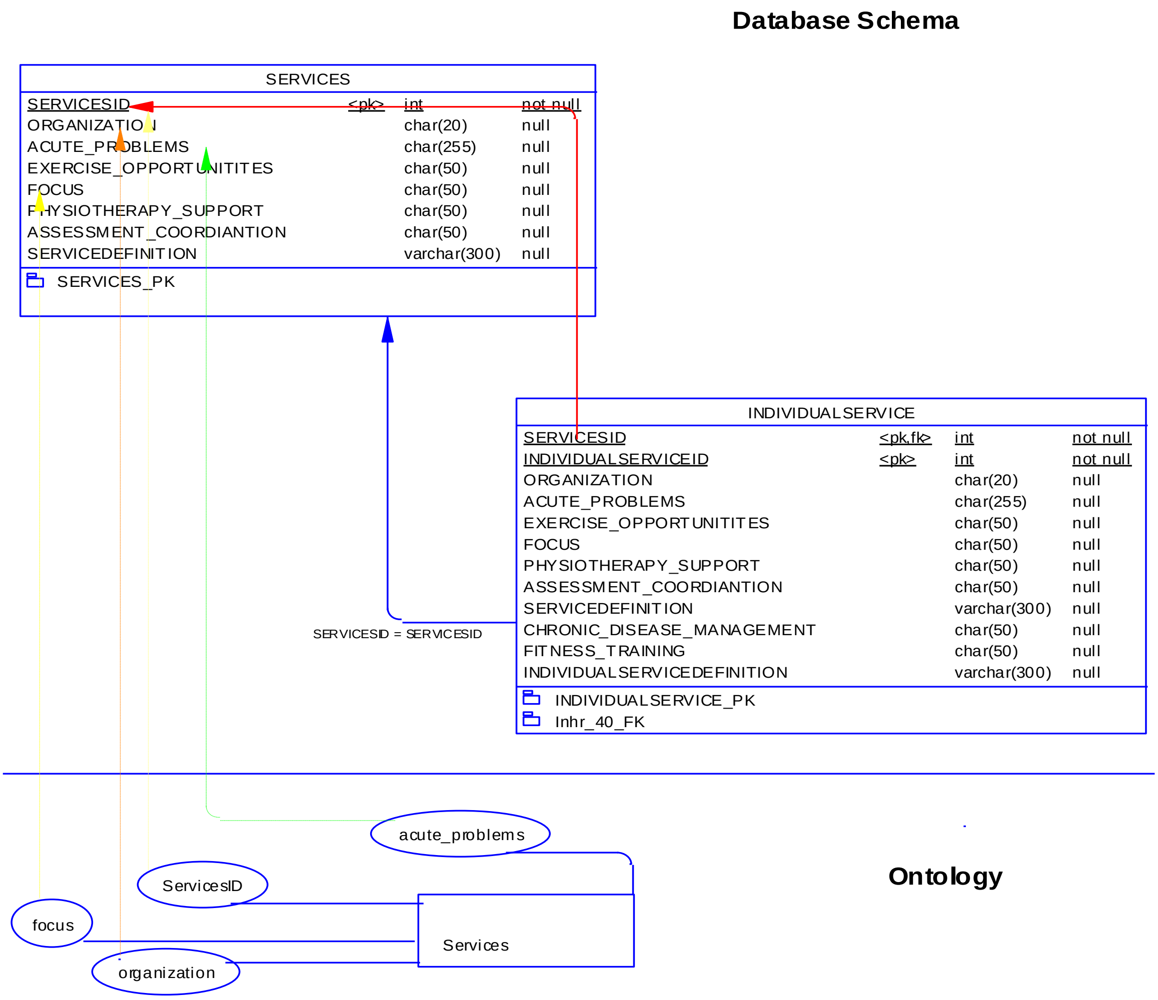

Figure 4.3 Ontology mapping of Services concept to Services table in the database schema

or logical data model is a manifestation of the domain representation that is specific to the application domain and can be transferred into ontology as demonstrated. As such, while ontology development is generic and independent of the application, the relative importance of the domain experts at the initial stages is reduced since it involves the inclusion of entities, attributes and relationships of the conceptual layer of the database into ontology. This is possible since the conceptual layer of the data model holds the knowledge that is abstracted in the ontology. This does not preclude their involvement since their expertise can be marshaled at the later stages to refine the generated ontology to fulfill the knowledge for the user community it is targeted at. Although many domain specific databases containing equivalent database fields may exist, the database field labels and the format varies from vendor to vendor, as in the case of T-SQL in

MSSQL Server 2000 and PL/SQL in Oracle. For speedy retrieval of data, it is imperative to define the semantics of the database fields although this risks associating the ontology close to the data fields. Annotated databases like Swiss-Prot and that in our research project help in the understanding of the data but no knowledge is assumed *a priori*.

Conceptual data models represent the universe of discourse (UoD) for the given application but the domain constructors available for the information system do not always get translated in the ontology. This is exemplified in the referential integrity constraints which check for the validity of the database for the specific application but do not hold the conceptualization of the UoD. This can lead to over-simplification in the extraction of ontology from logical schema by the addition of extra entities during the generation process.

This becomes indecipherable when users need to perform operations involving queries that transact across different databases since it involves joining different sets of data and, if replication is involved, data formats. Hence, the need for the development of an ontology that will mitigate the semantic irregularity of a federated database.

4.4.1 Reverse engineering Schema to Ontology

While different database models exist to store data, in our research project the relational model has been used to capture the information and store the data for both the PDPHDB and PDCDB.

The primary keys for the entities are generated as an attribute for the concepts with integer as the range. The foreign key constraints are generated as an attribute to the concept of the child table with the range corresponding to the concept of the parent table. This can be demonstrated in the entity IndividualService, a child table of Services, where the composite key of *ServicesID* is generated as a concrete concept of `IndividualService` with the primary key of *ServicesID* generated as an attribute with the range of integer. The foreign key constraint of *ServicesID* is generated as an attribute with the range corresponding to the concept of `Services`. Some of the attributes of the ontology corresponding to the schema entity for PDPhysiotherapy are shown in Figure 4.4.

The abstraction of some of the attributes is inconsequential and redundant. The referential integrity constraints maintain the consistency of the database but its generation as attributes does not provide any information in the conceptualization of the domain. As such, attributes (e.g. *SERVICESID_FK_INDIVIDU_INHR_402_SERVICES*) can be abstracted out of the ontology since it creates clutter and confusion to the domain experts

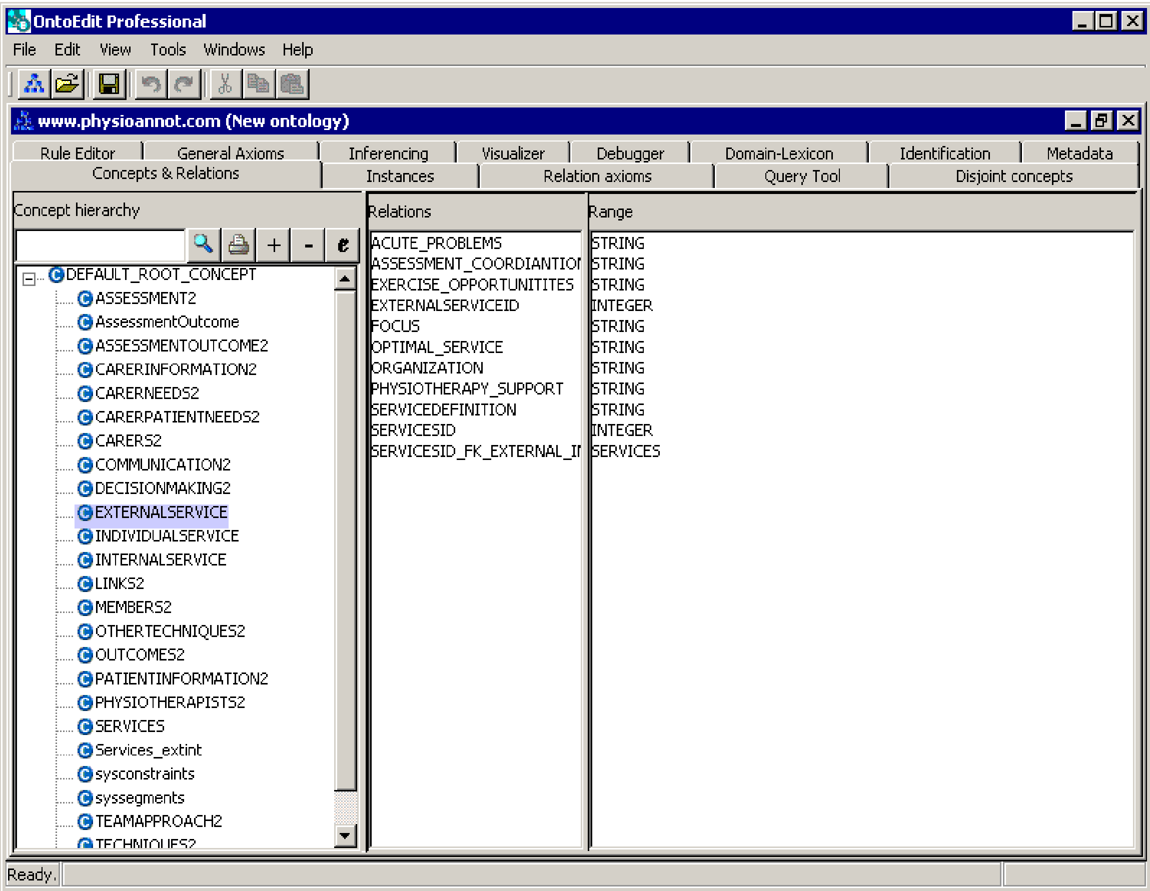

Figure 4.4 Ontology generation from PDPHDB data model

in the refinement of the ontology. But attributes like *ServicesID* are preserved since this can be used as an object identifier that can be mapped to the query processor as an underlying unique identification key. It is also noted that child tables are not transferred as sub-concepts of their corresponding parent entities.

The ontology generated has 26 concepts, 171 relations (or slots) and 50 axioms for PDPHDB and 27 concepts, 312 relations and 50 axioms in case of PDCDB data model. The database had 23 entities and these were translated into concepts with the addition of system tables of MS SQLServer 2000 for PDPHDB and the same behaviour was noticed in the case of PDCDB database. There are 190 attributes in the schema and this has generated into 171 relations and is illustrated in Figure 4.5.

The ratio of classes to attributes generated is 1:7. This is expected since relational data models contain a high number of redundant relationships (in this case 24). The concepts were generated as concrete in nature while the data types for the attributes were translated as string or integer. This is expected since data modelers use concrete data types to represent entities for the domain of discourse and explicit specification of shared conceptualization is irrelevant for the problem at hand.

4.4.1.1 Identifying ontology in multiple entities

In RDBMS, information stored in one particular entity can be distributed across multiple entities for better query processing and hence performance, management of both data manipulation language (DML) statements and restricting user access to specific tables. The interconnection of these multiple tables is facilitated by referential integrity constraints. The presence of a foreign key in a table implies that this table is related to another table and information contained in this table can be found in the parent entity. In figure 4.6, the tables Services, InternalService, ExternalService and IndividualService are connected by the presence of *ServicesID* as the foreign key in the

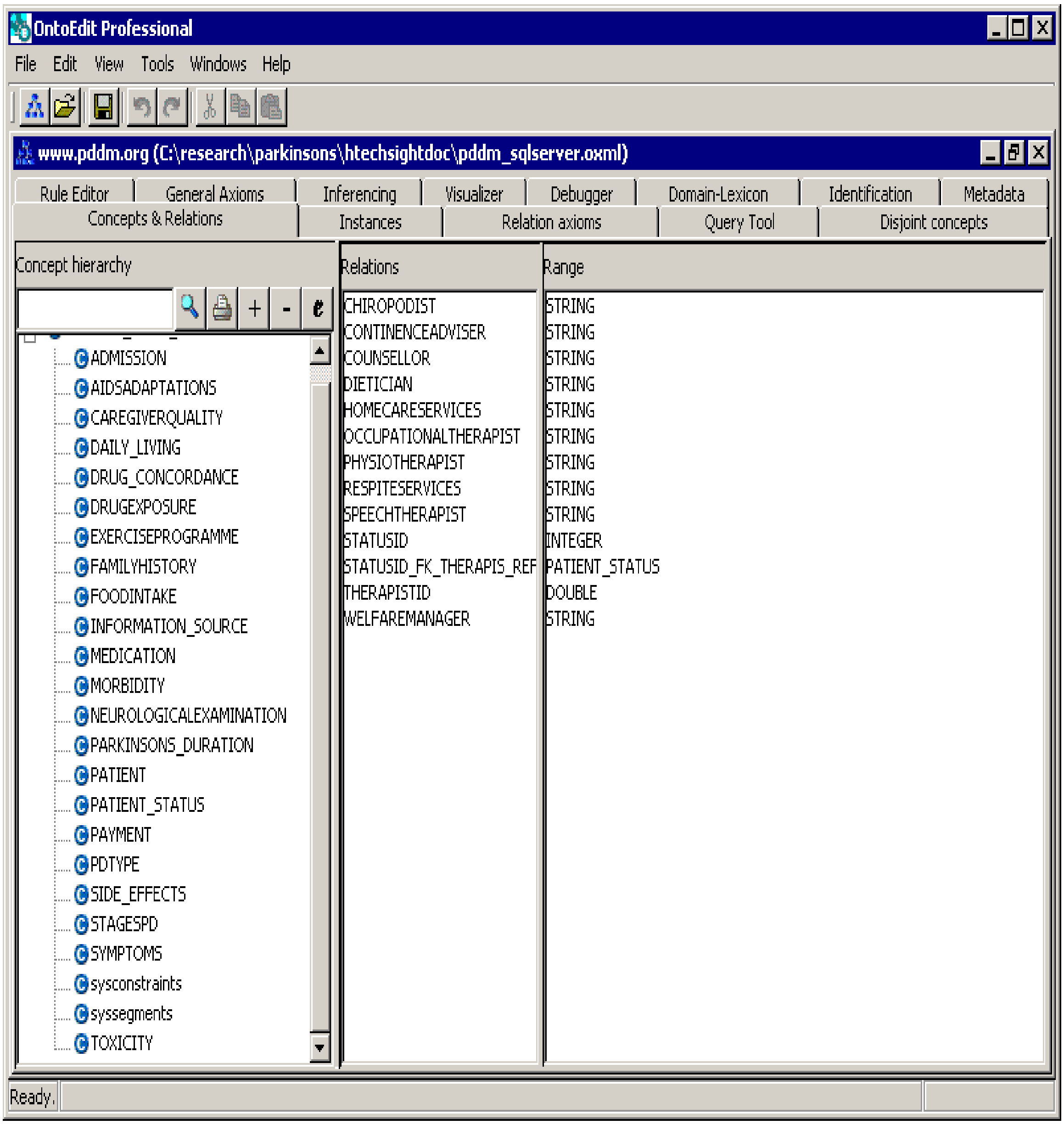

Figure 4.5 The ontology generated from the PDCDB database

last 3 tables corresponding to the primary key *ServicesID* in the Services table. This shows that these entities are related to each other and that information present in InternalService, ExternalService and IndividualService are related to the Services in the ontology. Although each of the 3 entities were generated as individual concepts they had the same attributes of `Services` in addition to their own attributes. Making them subsumption to the Services concept eliminated the redundancy of the attributes.

This is shown in the figure 4.6 and shown as ‘is-a’ in the edges relating the concept between Services and InternalService, ExternalService and IndividualService.

4.4.1.2 Identifying relationships in tables

In this case, the relationships that exist between different tables are transformed into one or more concepts. The tables Services, InternalService, ExternalService and Services_extint contain the following columns for their respective tables:

Services – ServicesID (pk)
InternalService—ServicesID (fk to Services), InternalServiceID (pk)
ExternalService-- ServicesID (fk to Services), ExternalServiceID (pk)
Services_extint-- ServicesID (fk to Services), InternalServiceID(fk to InternalService), **ExternalServiceID**(fk to ExternalService)

The presence of ServicesID, InternalServiceID and ExternalServiceID columns in the Services_extint table suggests that there exists information that can be captured due to the relationship that exists in the form of foreign key columns. This can be translated in the form of a concept in the domain ontology as follows:

ServiceProvided(<servicetype>,<association>,<managingdisease>) where
<servicetype> is obtained from the Services table due to the presence of ServicesID as the foreign key in the table
<association> is obtained from the tables ExternalService and InternalService due to the presence of ExternalServiceID and InternalServiceID as foreign keys
<managingdisease> is obtained from the table InternalService due to the presence of InternalServiceID as foreign key.

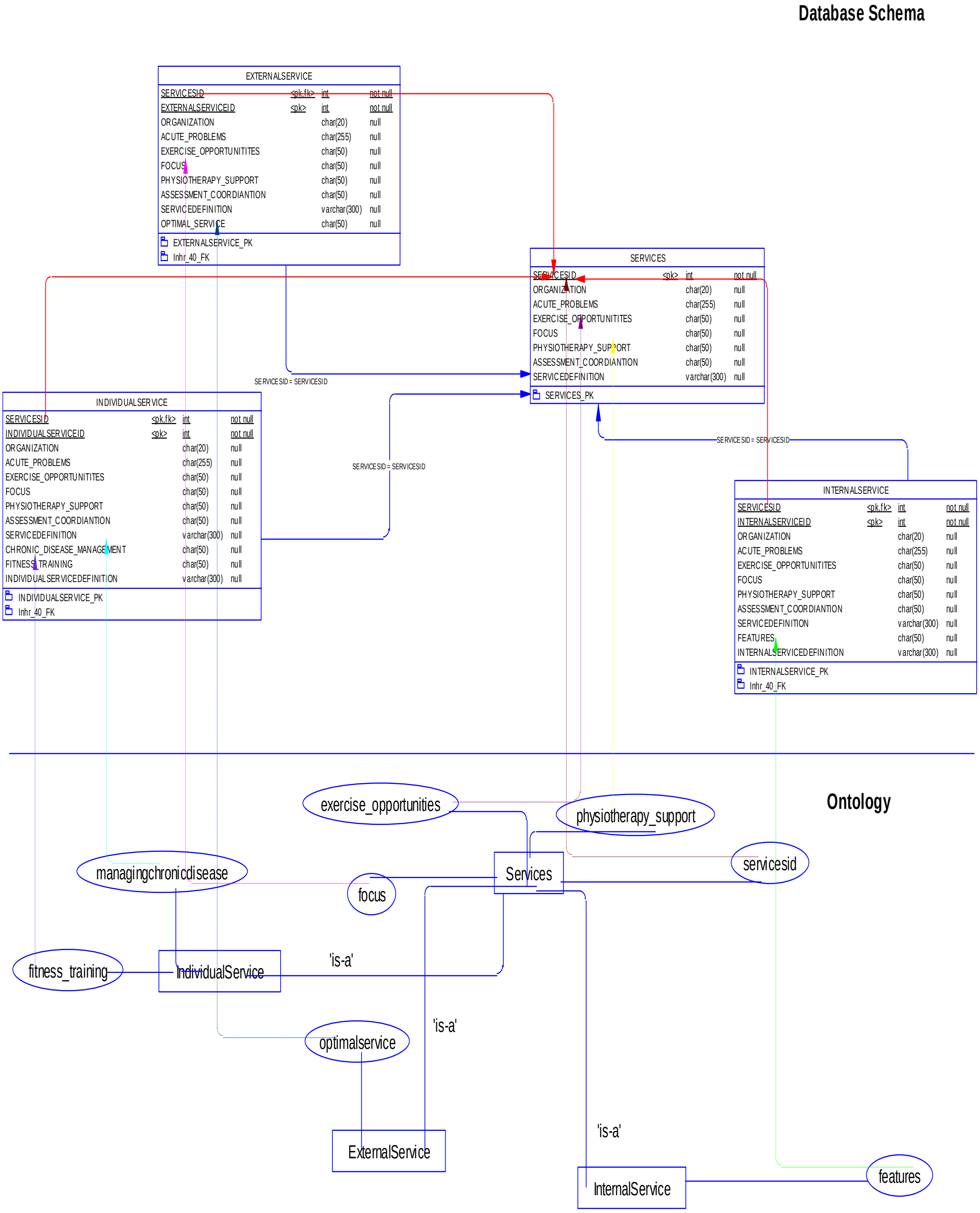

Figure 4.6 Ontology mapping of Services, InternalServices and ExternalServices concept to Services table in the database schema

The relationship between the `ServiceProvided` concept and the concepts Services, ExternalService and InternalService is regarded as a 'has-a' relationship as shown in the accompanying figure 4.7.

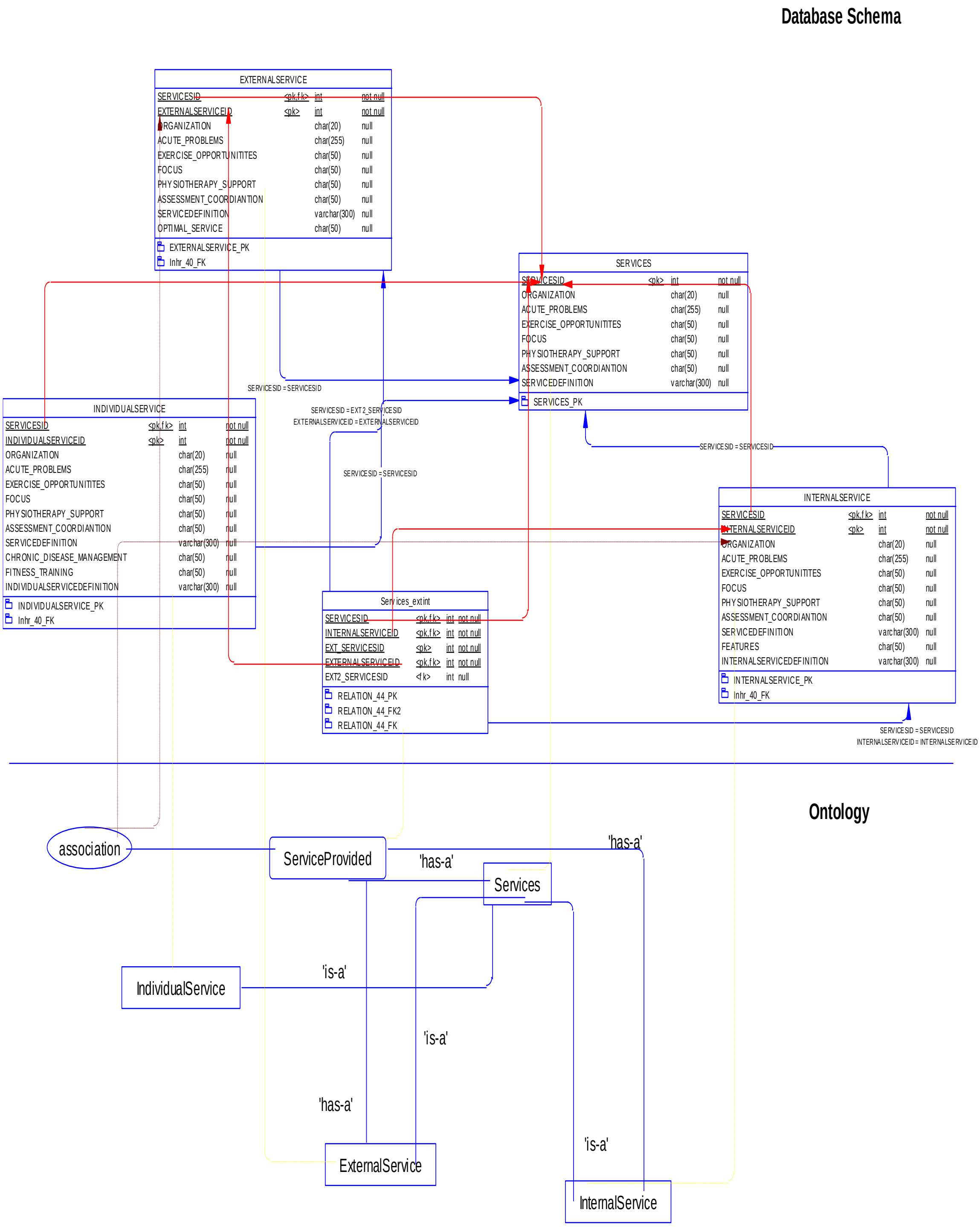

Figure 4.7 Ontology mapping of Services, InternalService, ExternalService concepts and the creation of new concept ServiceProvided to the corresponding tables in the database schema

4.4.2 New concepts suggested from database schema

In this section some of the concepts that were evolved from the extraction of the database for PDPHDB is shown. The domain ontology extracted from the database is shown in figure 4.8.

ServiceProvided: This has already been illustrated in the previous section
AssessmentMeasure (<measuredoutcome>,<outcometype>)
where <measuredoutcome> is obtained from the table Timing2 due to the presence of timingid2 foreignkey
<outcometype> is obtained from the table Outcomes2
PatientStatus(<carername>,<rehabilitationservices>,<symptoms>)
Where <carername> is obtained from the table CarerInformation2
<rehabilitationservices> is obtained from the tables CarerInformation2 and PatientInformation2 and Physiotherapists2 due to the presence of rehabilitation

Polymorphic entities or attributes are either renamed or removed to reduce ontological redundancy as in the case of rehabilitatation. Ontologies include the following elements: Taxonomic relations between classes, data type properties, descriptions of attributes of elements of classes, object properties, descriptions of relations between elements of classes.

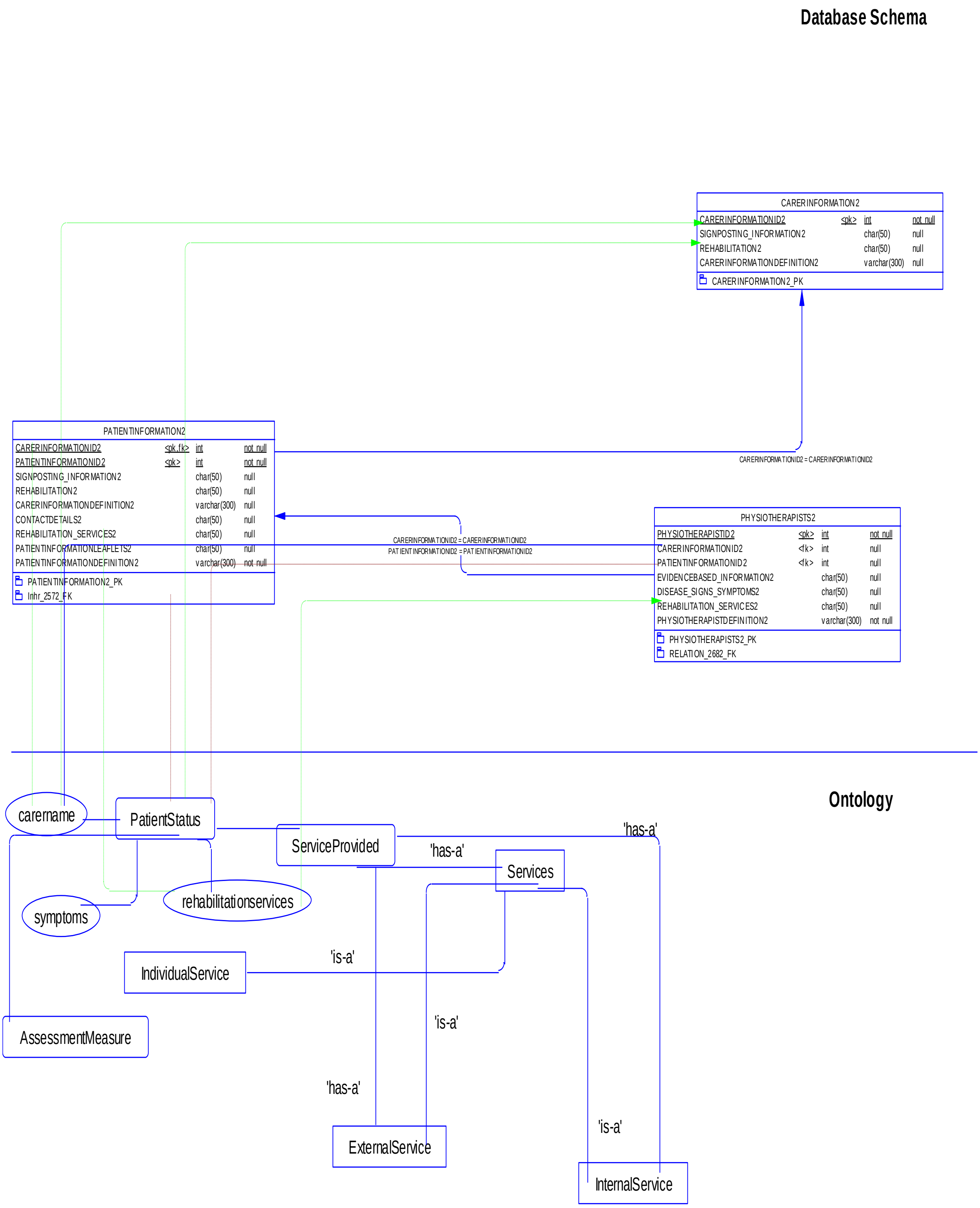

Figure 4.8 Ontological mapping of PDPHDB database

4.5 Ontology Enhancement

In this section, the incorporation of the foundational ontology (FOL), DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) in the domain ontology is discussed in order to enrich and enhance the domain ontology extracted from legacy database.

The quality of the ontologies extracted, as seen in the above sections, needs to be enhanced in order to make a meaningful relationship between the several concepts and the corresponding attributes. The quality of such ontologies can be enhanced for the purposes of reusability and utility by aligning to a FOL like DOLCE. It has a rich set of axiomatised foundational ontology. This allows the facilitation of *meaning negotiation* among agents and *establishing consensus* by lieu of its ontological commitments [105]. Both the ontologies extracted from the database schema are linked to the DOLCE ontology by adding subsumption relationships to the top level concepts of PDPHDB and PDCDB ontologies.

Foundational ontologies are domain-independent and act as complementary to light-weight ontology by acting as the bridge between human and machine understanding. While light-weight ontology can be semi-automatic since the semantic meaning has already been developed, FOL describes the notions that are used in a general manner be it in the field of economics, insurance, agriculture or law. Core ontologies (COL), on the other hand, are specific to the domain and are characterized by the main conceptual schema that members of the neurological research fraternity would use. The combination of FOL and COL gives us the infrastructure that is necessary to integrate diverse ontologies of the same domain. DOLCE ontology has been used in the alignment of fishery and legal ontologies. In this case, we use it for the alignment of ontologies that have been used to provide ontological commitment as envisaged by Bechhofer et al [106] for the neurological domain for PWPD.

The 4 top categories of DOLCE (version 2.0) are classified as: (a) endurant which include entities like object and substance, (b) perdurant that contain event and state like entities, (c) quality that includes individual attributes and (d) abstracts (mainly conceptual “regions’ for structuring attributes). The version used contains 80 classes and 80 attributes. The attributes in the database schema are reverse-engineered in two categories. Database attributes containing referential integrity constraints are translated as ObjectProperties while the other column attribtutes are generated as DatatypeProperties in DAML+OIL format. This is shown in figure 4.9 for the PDPHDB ontology

```
<daml:DatatypeProperty
rdf:about="www.physioannot.com#ASSESSMENTOUTCOMEID2">
    <rdfs:subPropertyOf rdf:resource="www.physioannot.com#PDPhysiotherapy"/>
  </daml:DatatypeProperty>
  <daml:DatatypeProperty rdf:about="www.physioannot.com#ASSESSMENTID2">
    <rdfs:subPropertyOf rdf:resource="www.physioannot.com#PDPhysiotherapy"/>
  </daml:DatatypeProperty>

<daml:ObjectProperty
rdf:about="www.physioannot.com#ASSESSMENTOUTCOMEID2_FK_ASSESS
ME_INHR_1632_ASSESSME">
    <rdfs:subPropertyOf rdf:resource="www.physioannot.com#PDPhysiotherapy"/>
  </daml:ObjectProperty>
  <daml:ObjectProperty
rdf:about="www.physioannot.com#CAR_ASSESSMENTOUTCOMEID2_FK_ASS
ESSME_RELATION__CARERNEE">
    <rdfs:subPropertyOf rdf:resource="www.physioannot.com#PDPhysiotherapy"/>
  </daml:ObjectProperty>
```

Figure 4.9 DAML+OIL code showing the DatatypeProperty and the ObjectProperty of PDPHDB attributes

In this case, the alignment was done by subsuming the relationships of the ontology for PDCDB and PDPHDB to that of DOLCE. The PDPHDB:Services, PDPHDB:InternalServices, PDPHDB:ExternalServices, PDPHDB:IndividualService and PDPHDB:ServiceProvided are sub-concepts to **dolce:Activity** while PDPHDB:Physiotherapists, PDPHDB:Members and PDPHDB:Carers subsumed by **dolce:Social-Role.** The PDPHDB:ServiceProvided and PDPHDB:Services are then associated along with their cardinality constraints as illustrated in the UML diagram showing the alignment of PDCDB and PDPHDB to DOLCE ontology (figure 4.10). This shows that all these concepts are *endurants* which are essentially entities or objects. It was also found that PDPHDB:TeamApproach, another *endurant*, was subsumed by **dolce:Organisation**. The other techniques like pulmonary rehabilitation, kinesiotherapy, cueing strategies, manual handling and mechanical treatment in physiotherapeutic assessments are processes used to treat patients with PD. As such, concepts like PDPHDB:Techniques and PDPHDB:OtherTechniques are subsumed by **dolce:Biological-Process**. Similarly, PDCDB:NeurologicalExamination, PDCDB:Symptoms, PDCDB:FoodIntake and PDCDB:ExerciseProgramme are all biological processes and are therefore subclasses of **dolce:Biological-Process**. These are then regarded as *perdurants*, processes or events that affect change. The PDCDB:CaregiverQuality is a sub-class of the **dolce:PhysicalQuality**, while PDCDB:Patient is a subclass of **dolce:Natural-Person** since a human being is a subsumption of **dolce:Natural-Person** [105]. The different kinds of information like

`Weblinks` and `Patient-Information` are sources of information and these are child classes of **dolce:Information-Object**. The ontological commitment mentioned earlier is thereby fulfilled by giving added meaning to the concepts and the relations by explicating from the database schema.

Figure 4.10 Alignment of PDCDB and PDPHDB with DOLCE ontology

4.6 Conclusion

It is shown that ontology can be used to generate a semantically enabled database. The PDSymptomsdrugs ontology was used to generate the database. It was also noticed that each of the concepts generated a separate table. The subsumption relationship was not followed as a child table in the corresponding databases. This can cause limitations in the generation of the DDL. As concepts are added or deleted during the lifetime of an ontology development, transactional processing that is the bedrock of a relational database schema is significantly impeded.

The two heterogeneous databases PDPHDB and PDCDB were reverse engineered to generate ontology. The ontologies extracted from the two legacy databases contained concepts that mapped to the tables and attributes of the corresponding columns of the respective databases. It was found that the attributes generated were of two types, notably DatatypeProperties corresponding to columns while columns containing referential integrity constraints generated ObjectProperties. The ontologies generated were then enhanced by aligning with the foundational ontology of DOLCE to provide ontological commitment.

The following chapters describe the knowledge layer of the architecture that shows the different aspects of modeling the knowledge base and finally the development of the knowledge bases.