

Ontology Construction from Relational Database

Abdul Jaleel¹, Shahid Islam¹, Asim Rehmat², Amjad Farooq² and Adnan Shafiq²

¹Rachna College of Engineering & Technology, Gujranwala

²University of Engineering & Technology, Lahore

Abstract– This research presents our work to construct Ontologies from relational databases. We have reviewed the existing rules for mapping database schema into ontology schema and consequently updated the mapping rules to accommodate the various missing cases. A real world relational model, the university course registration and fee collection, has been used to demonstrate the mapping of relational data model into ontology concepts. Our example relational model contains every possible construct of the database schema and we have demonstrated the conversion of each construct into ontology using our information system example.

Keywords– Semantic Web Services, Ontology, Relational Database and Construction

I. INTRODUCTION

World Wide Web (WWW) is a huge information space. Most of the information available over the www comes from the relational database system (RDBS). The data within relational database can be used to construct Ontologies which provides shared common meaning and reusable knowledge about a particular domain. Many approaches [6], [8]-[10] have been proposed to construct Ontologies from RDBS using mapping rules. However these mapping rules don't take into account all scenarios, found in a RDBS.

In this paper, we have reviewed and enhanced the existing rules for mapping the RDBS into Ontology schema. The rest of the paper is organized as follows. Section 2 summarizes the previously related research. Mapping rules are given and explained using an information system example, in section 3. Next section presents the discussion and results. Section 5 concludes our work and presents future directions.

II. RELATED WORK

Soares [2] reviewed the different approaches of ontology development to identify their suitability for the information systems development process. The study revealed that no standard approach exists for the ontological design and development of information systems. Almost 60% of the engineers don't use any particular methodology to build the Ontologies.

An approach was proposed by Stojanovic [3] to make the database-driven web information space visible and machine process-able. First, relational database schema was revealed using reverse engineering. Then relational data base to object-oriented database conversion rules were applied on the relational schema to generate the corresponding Ontologies under the supervision of expert.

Irina [8] presented an approach to migrate data-intensive web pages which are based on relational databases, into the ontology based Semantic Web. Their work uses information from web pages to construct the equivalent ontology. However this approach is not efficient because html forms do not fully represent the semantics of the backend database.

The work of [4], [6] and [7] explained the process of ontology creation from relation database using mapping rules. They presented a set of rules for mapping database schema into ontology. However, these works lack some specific database cases which we have dealt in our work.

III. CONSTRUCTION OF ONTOLOGY FROM RELATIONAL DATABASE


This section presents the revised mapping rules to construct the ontology from relational database. As an example, we take a database schema which stores the student related information. The database example taken here, stores the fee records and course enrollment of students at university level (having semester system). Figure: 1 presents the database schema of this scenario. The given schema contains all types of relationships and constraints that may exist in RDBS. We will use this relational schema for the explanation of mapping rules.

We have divided the process of constructing ontology from relational database into the following seven steps. Each step explains the mapping process considering different scenarios that may occur within RDBS.


Prerequisite: database schema must be in 3NF at minimum.

⇒ In our case the database schema is already in 3NF.

Step 1: Identify Primary Keys (P. Key) for all Relations

⇒ The attributes having symbol  before them (in figure: 1), are P. Keys of our relations.

Step 2: Identify Foreign Keys (F. Key) for all Relations

⇒ The attributes having symbol  before them (in figure: 1), are F. Keys of our relations.

⇒ The attributes having symbol  before them (in figure: 1), are F. Keys as well as P. Keys or part of P. Keys.

Step 3: Formation of Ontology classes

Rule 1: this rule is composed of following three cases

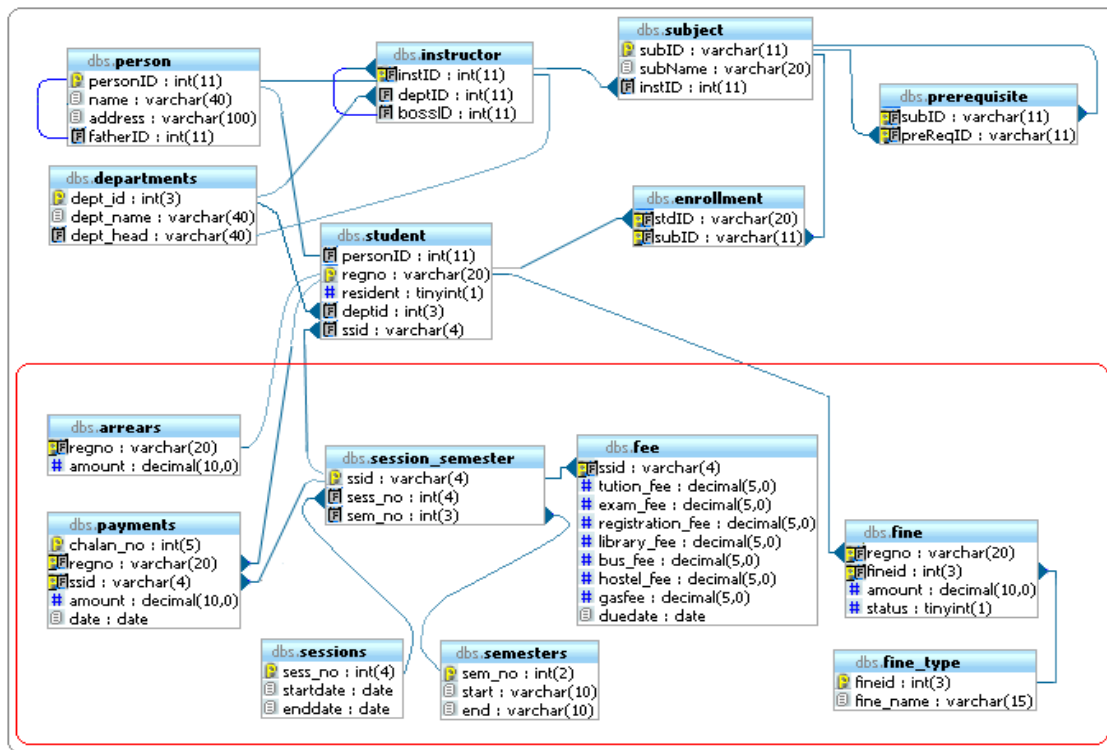


Figure 1 : Database Schema of Example Information System

Case (a) If a relation is self dependent, i.e. there is no F. key

➤ Map the relation as ontology class

⇒ *The ontology classes created from this rule are Session, Semester, and Fine_type.*

Case (b) If a relation contains a F. Key and F. Key is an alternate key.

(F. Key is neither P. Key nor part of P. Key, but may become P. Key)

➤ Map the relation as subclass of foreign key's parent class

⇒ *The ontology class created from this rule is Student. It will be subclass of Person class.*

Case (c) If a relation contains a F. Key and F. Key is not P. Key, not part of P. Key and also not an alternate key

➤ Map the relation as Ontology class

⇒ *The ontology classes created from this rule are Person, Subject, Department and Session_Semester.*

Case (a) If F. Key is being used as the only P. Key in the child relation.

(P. Keys are same in hierarchically linked relations)

➤ Then merge the child relation into parent relation to make single ontology class.

⇒ *In our example case, Arrears relation will be merged into student relation. The attributes of arrear will be added into student class.*

⇒ *The fee relation will be merged into session_semester relation. The ontology class created from this rule is Session_Semester_Fee.*

Case (b) If condition of (a) is satisfied but the child relation has some dignity and can't be merged into parent relation

➤ Then make the child relation as subclass of parent relation's ontology-class.

⇒ *In our example case, instructor relation has personID (F.Key) as its P.key. So according to case (a) of this rule, instructor should be merged into Person class.*

Rule 2: In case of hierarchical association between relations

But as we see, it has other attributes which dignify it, so it will become subclass of Person class.

Rule 3: If the P. Key is composite and in addition to F. key/keys, some local attributes are also part of the P. key

➤ The relation may be mapped as ontology class

⇒ *The ontology class created from this rule is Payments.*

Rule 4: If the P. Key is composite and consists of only F. keys

➤ The relation is a by-product of many-to-many relationship (associative entity) and may not be mapped as an ontology class. Instead they should be used for learning object properties.

⇒ *Prerequisite, enrollment and fine are associative relations. They will be used for learning object properties, latter on.*

Step 4: Learning of object properties from Relations

The F. Keys are used to define object properties. Following different cases may occur in RDBMS which will define different object properties.

Case 1: The F. key in a relation is not part of the P. key

➤ F. key will become object property of the ontology class.

The domain of Property will be the class in which it is F. Key and range will be the class where it is P. Key.

⇒ *fatherID in Person class (recursive O-t-O Relation) will create hasFather object property for Person class. Domain and Range will be same i.e. Person class.*

⇒ *bossID in Instructor class (recursive O-t-M Relation) will create hasBoss object property for Instructor class. Domain and Range will be same i.e. Instructor class.*

⇒ *deptID in Instructor (O-t-M) will create belong_to_dept object property with domain instructor and range Department.*

⇒ *deptID in Student (O-t-M) will create belong_to_dept object property with domain Student and range Department.*

⇒ *headID in Department (O-t-O) relation will create headed_by object property with domain department and range instructor.*

⇒ *instID in subject (O-t-M) will create taught_by object property with domain Subject and range Instructor.*

⇒ *sess_id & sem_id in session_semester_fee class (created after merging session_semester & Fee classes in Rule 2 case (a)) will become its object properties.*

The domain will be session_semester_fee and range will be session and semester respectively.

Case 2: If P. key of a relation is composite and F. key is part of P. key

➤ Then “has-part” and “is-part-of” object properties will be created. These will be inverse properties of each other.

➤ If there exist some other F. keys which are not part of P. key, they will be mapped as object properties.

⇒ *Regno in Payments will create is_paid_by and has_paid object properties between payment and student. (represented through blue & orange lines)*

⇒ *ssid in Payments relation will create is_paid_for and has_payment object properties between payments and session_semester_fee. (represented through blue & orange lines)*

Case 3: If a relation is an associative entity, i.e. by product of many-to-many relationship

➤ The relation will be resolved and two object properties are added, one for each class of the many-to-many relationship.

▪ These object properties will be functional properties (cardinality 0 or 1)

⇒ *pre-requisite & enrollment relations will be resolved and converted into object properties between their parent tables.*

Case 4: If a relation is an associative entity and also have some other attributes in addition to foreign keys

➤ It will be resolved such that the attributes other than foreign keys are moved to respective relation.

▪ Then two object properties are added, one for each class of the many-to-many relationship.

⇒ *In fine relation, amount and status will be moved to fine_type (as shown in figure). The fine relation will be resolved to create two object properties*

has-fine and *fine-of* between *student* and *fine_type*.

Figure 2 represents the extracted classes and their object properties extracted using above rules. The Red lines show hierarchical 'is-a' relationship between classes. Black lines show foreign key based object properties. Pink and Green combination is used to

represent object properties extracted after resolving many-to-many relationship. Brown and Blue combination is used to represent the object properties extracted from the cases where foreign key is also part of primary key.

The ontology classes for the fee subsystem from the above database schema are shown below in Figure 3.

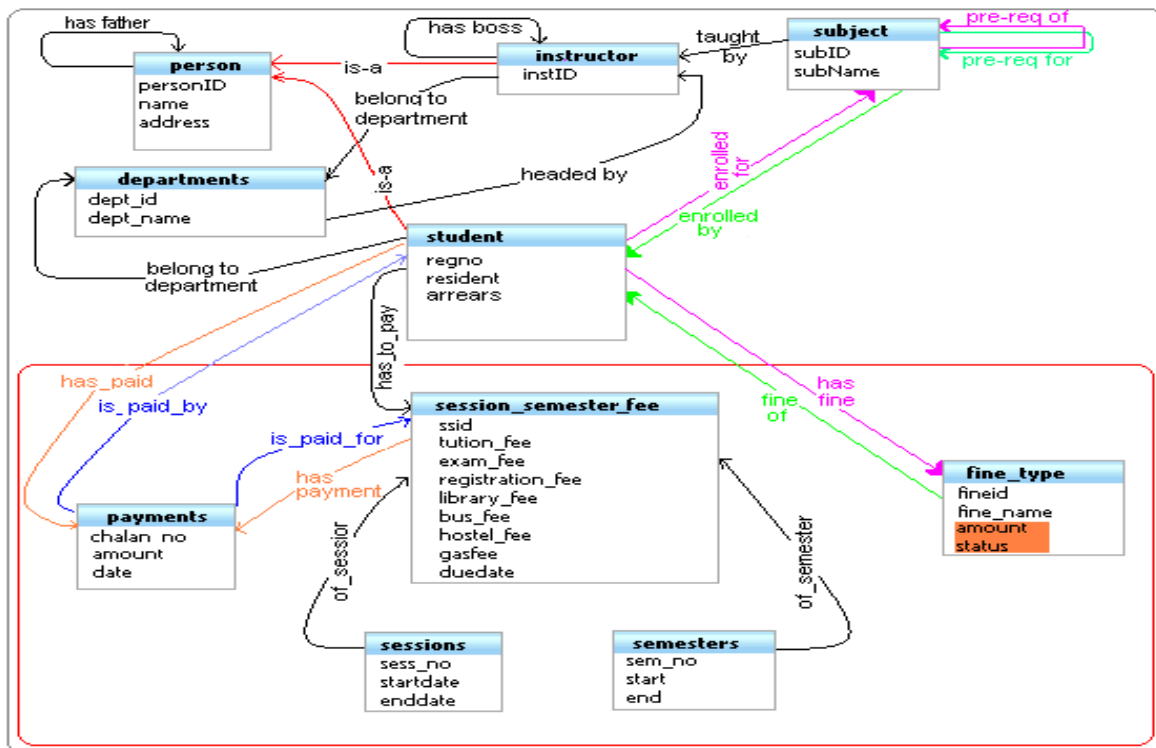


Figure 2 : Extracted Classes and Object Properties

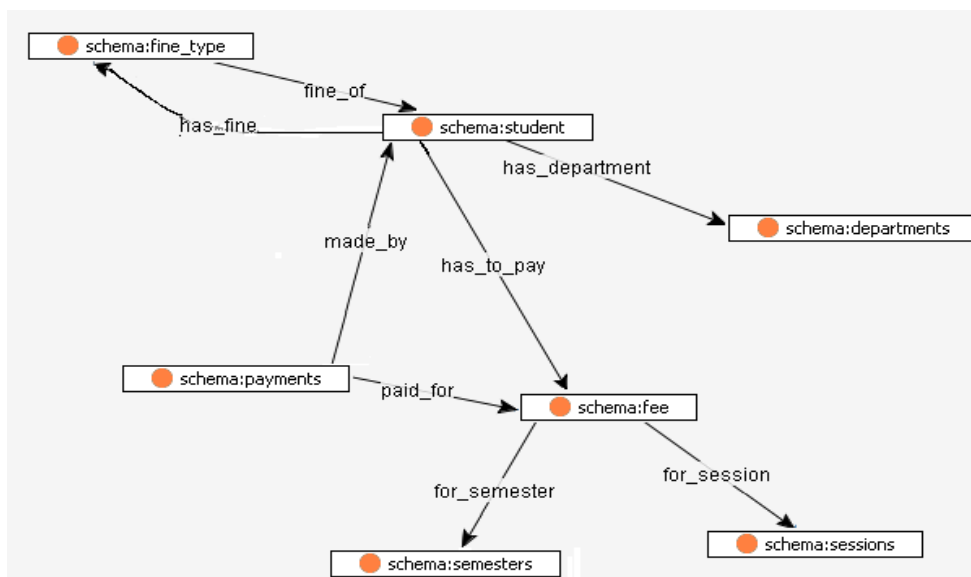


Figure 3: Fee System Ontology classes and relations

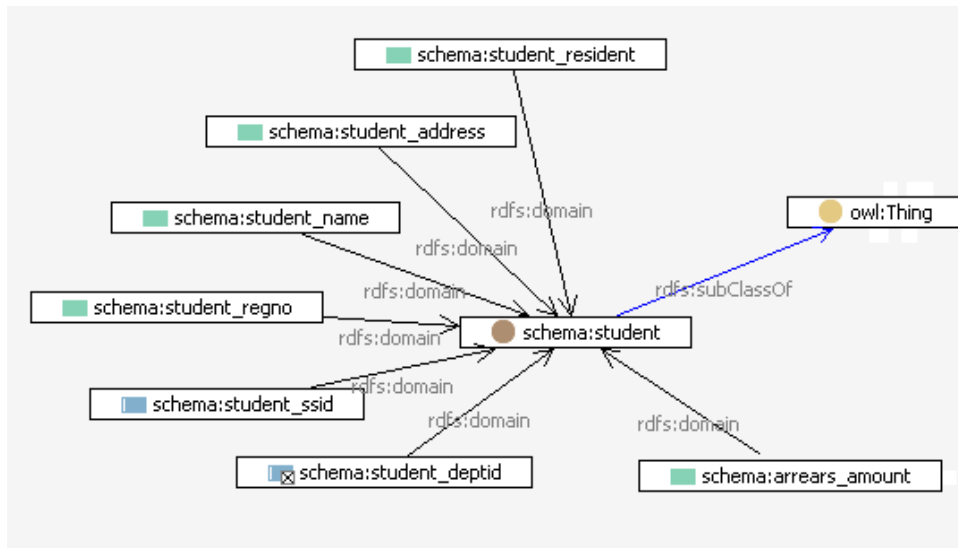


Figure 4: Student class with Data Properties

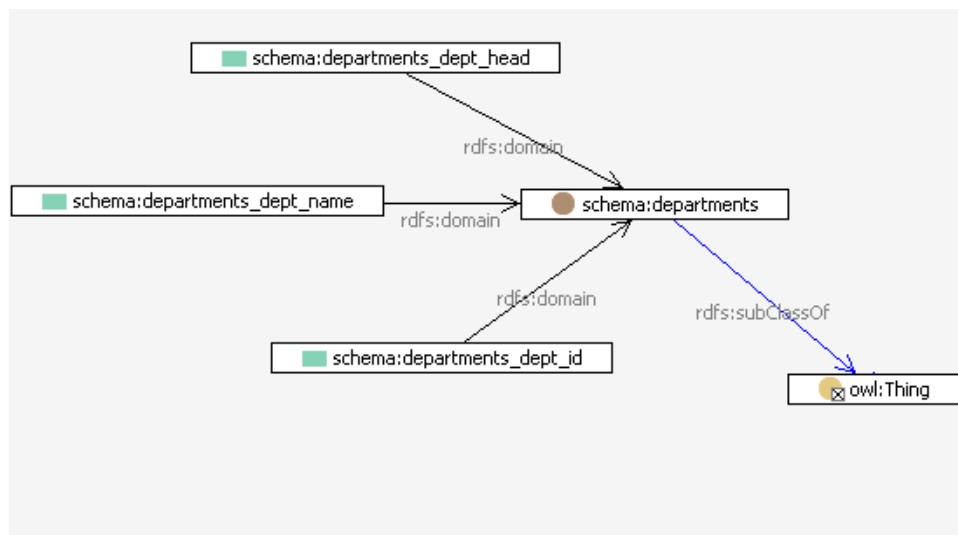


Figure 5 : Department class with Data properties

Step 5: Finding out data properties

The attributes of a relation other than foreign keys are mapped as data properties for the corresponding ontology class. The range of a data property is the XML schema data type which is equivalent to the data type of original column. Figure: 4 represents the Student class with data properties.

Figure: 5 represents the Department class with data properties

Step 6: Cardinalities Handling

❖ In case of Primary Key constraint, minimum and maximum cardinality will be one.

- ❖ If NOT NULL constraint is specified for an attribute, the minimum cardinality of the corresponding property will be one.
- ❖ If UNIQUE constraint is specified for an attribute, the maximum cardinality of the corresponding property will be one.
- ❖ The CHECK constraint will affect the range of corresponding property.

Step 7: Instance creation

The instances of an ontological class consist of the tuples in the corresponding relation/relations of database.

IV. RESULTS & DISCUSSIONS

Several authors [4], [6], and [7] have presented rules for the migration of relational database schema into corresponding Ontology schema. However these rules map only the most essential properties of relational data model into Ontology model and hence the resulting Ontology schema fail to capture every nuance of relational schema. These rules are inadequate for the conversion of complex real-world relational schema into Ontology schema.

Table 1 contrasts our work with the work of other researchers [4], [6], [7]. It is evident from the table that previous rules are silent that how Ontology classes will be created from a relation when primary key is composite (consists of only foreign keys) and when the primary key is composite and consists of foreign key attributes along with other local attributes. Existing mapping rules also fail to answer the object properties when an associative relation also has some other attributes along with the foreign keys. Reader is referred to Table 1 for a detailed comparison of our approach with the existing ones.

The main contribution of this paper is the review of the existing mapping rules and hereafter the extension of these rules while taking into account every possible design aspect of relational database schema. We have also explained the mapping process with the help of a case study that converts a relational database schema, to manage the record of students' enrollment and fee collection in a university, into Ontology schema.

V. CONCLUSION AND FUTURE DIRECTIONS

The mapping rules for the construction of ontology from relational database are presented in this work. Moreover, these rules are explained with the help of a case study; relational data model of an information system to manage the fee and subject enrollment of students in a university.

In future we are planning to design and develop a system for automated conversion of relational database schema into ontology schema using these given rules and without the need for human experts.

Table 1: Comparison Table

Database case	Man Li (2005)	Nadine Cullot (2007)	Zdenka Telanrove (2010)	Our Work
Formation of Ontology classes considering different cases of a Relational Database				
Relation is self dependent (no foreign key)	✓	✓	✓	✓
F. Key is an alternate key	–	–	✓	✓
F. Key is not P. Key, not part of P. Key, also not an alternate key	partially	partially	Partially	✓
F. Key is the only P. Key	✓	✓	✓	✓
P. Key is composite & consists of F. key/keys and local attributes	–	–	–	✓
P. Key is composite and consists of only F. keys	–	–	–	✓
Learning of object properties considering different cases of a Relational Database				
F. key is not part of the P. key	✓	✓	✓	✓
P. key is composite and F. key is part of P. key	✓	–	✓	✓
associative relation	✓	✓	✓	✓
Associative Relation also having attrib in addition to F. keys	–	–	–	✓
Cardinalities Handling				
Primary Key constraint	✓	✓	✓	✓
NULL constraint	✓	✓	✓	✓
UNIQUE constraint	✓	✓	✓	✓
CHECK constraint	–	–	–	✓

REFERENCES

- [1]. Kupfer A., Eckstein S., Coevolution of Database Schemas and Associated Ontologies in Biological Context, Proceedings of 22nd British National Conference on Databases Vol. 2, 2005, pp 45-50
- [2]. Soares A., Fonseca F., Building Ontologies for Information Systems: What We Have, What We Need, ACM 2009
- [3]. Stojanovic L., Stojanovic N., Volz R., Migrating Data-intensive Web Sites into the Semantic Web, ACM Spain 2002
- [4]. Man Li, Xiao-Yong Du, Shan Wang, Learning Ontology from Relational Database, Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 2005
- [5]. Olivier Cure, Mapping Databases to Ontologies to Design and Maintain Data in a Semantic Web Environment, CYBERNETICS, 2006, pp 52-57.
- [6]. Nadine Cullot, Raji Ghawi, and Kokou Yetongnon, DB2OWL: A Tool for Automatic Database-to-Ontology Mapping, Proceedings of 15th Italian Symposium on advanced database systems, SEBD, 2007, pp 491-494.
- [7]. Telnarova Z., Relational Database as a Source of Ontology Creation, Proceedings of the International Multiconference on Computer Science and Information Technology, 2010, pp. 135-139.
- [8]. Irina Astrova, Bela Stantic, Reverse Engineering of Relational Databases to Ontologies: An Approach based on an Analysis of HTML Forms, 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, Italy, 2004, pp 73-78