I J C T A, 9(7), 2016, pp. 3037-3052
© International Science Press

ISSN: 0974-5572

An R2RI approach updating Relational data through SPARQL/Update

V. Sitharamulu* and B. Raveendra Babu**

ABSTRACT

Most of the enterprises today tend to use the Relational. DB for storing and managing the data. The data semantics are not directly embedded in the relational Schema, but they are indirectly encoded at the application levels. The semantic web technologies with other ontologies pave path for semantics directly to use the data across enterprises, large number of applications and for many other application specific domains. Relational.DB's stores large volumes of data in the web after converting to RDF. But, the conversion of relational schema to the respective RDF is very impractical. This is the reason why read only access to the database is adopted by many of the ontologies. Presently, all approaches rely only on the read only data accessibility and there is no explicit support for write access. Hence, the approach of R2RI which overcomes the limitation of read access is introduced. The R2RI approach richly supports the read/write access to the data. The update-aware mapping language (R2RI) comprises of algorithms and examples to define the translations of SPARQL/Update to SQL. So, R2RI can also be considered as RDB-to-RDF mapping language. In contrast, the current paper's primary contribution is to present and explain the translating process using algorithms, documenting and brief explanation of examples with special focus on prototype implementation.

Keywords: Relational databasee; Resource Description Framework: Semantic Web.

1. INTRODUCTION

Relational DB's are most widely used in day to day enterprise environments to manage and store the data. Relational databases are good to store and handle enormous data, despite not designated to handle and preserve data semantics. The data available in the databases is obviously implicit during the application process at the application domain level and is not directly encoded explicitly in relational schema. The semantic webresources associating other technologies specifically facilitate semantics for data reusability and among different enterprises, community boundaries and applications [1].

Semantic web technologies when applied at the enterprises environment levels, facilitate data exchanging and processing at the semantic level. A semantic layer residing at the top of the database schema featured to push the data units to semantic level from its native syntax stages. Shared ontology and RDF are used for exchanging of data even though a relational schema does not match. The introduction and background knowledge over ontology is also a valuable as set.

Conversion of data from RDB-to-RDF is not possible enough since most of the applications in the databases do really depend on the relational schema portrays. The standards of the triples and their implementations always remain below the RDB's as benchmarks [2]. Therefore, an alternative approach to preserve the compatibility of the existing relational applications is the usage of mediation technique which transforms the translation of semantic web that requests to the corresponding SQL. This enables the access for the ontology-based software to synchronize and work with the same and existing data. In addition to this, the mediation allows further in exploiting the benefits of the advanced database schemas in scalability, query throughput, security and endorsement for transaction.

* Assoc. Professor, Dept. of Computer Sc. & Engineering, SBIT, Khammam, India, *Email: vsitaramu.1234@gmail.com*

** Professor & HOD, Dept. of CSE, Dean-Administration & Finance, VNRVJIET, Hyderabad, India, *Email: raveendrababu.b@vnrvjiet.in*

All the current approaches for RDB-to-RDF aim in projecting the existing data in the relational schema to the corresponding semantic web. The approaches also enhance the SPARQL endpoints to query the relational data. But, neither addresses the data updates nor exhibit applications in the enterprise environment. Our esteemed contribution in the current work is based on the ontology with respect to write access to the relational data through [2] SPARQL/Update, the upcoming DML of semantic web. Update-aware mapping language RDB-to-RDF Ingression (R2RI) with algorithms and examples for translating SPARQL/Update in to SQL DML are also presented.

2. RELATED WORK

Relational.OWL [3] which is a web ontology language states the ontology in representing relational schema and the data in Resource Description Framework (RDF). The task of the Relational.OWL is to map the attributes and tables in to terms and simultaneously maintain track information pertaining to the foreign/ primary keys being used and also specifies the exact data type of the attributes involved in the tables. The Relational.OWL approach entails the structure and semantics during conversion of schema to the RDF. The RDquery[4], supplements the SPARQL interface for converting SPARQL queries in to the SQL on the top layer of the Relational.OWL.

RDB's are supposed to be published on the semantic web, and the approach used was D2R. It enhances the browsing facility of the relational data as RDF through URI availability. D2R's main initiative is to provide inter linkage of datasets in the web and expressed in the form of RDF.

The open link software Virtuoso features views in RDF [5] in the relational data and specifically uses the metadata schema mapping language for converting all the teams in ontology to respective considerations in relational schema. This approach finalizes the SPARQL as complimentary language. The views in the RDF are not updatable since the views are of read only tagged queries.

Triplify[6] which is another approach to publish web applications information in RDF. Triplify is a source to convert RDB-to-RDF by constituting the SQL queries. These queries are manually performed and the RDF generation is executed automatically.

MASTRO-I[7] is one among many data integration approaches on the basis of GAV (Global As View) mappings. The independent schema is integrated with the help of data federation tool. MASTRO-I is also lacking with the write access and accessible only for read only data.

The World Wide Web Consortium (W3C) has identified the significance of mapping languages and especially RDB-to-RDF. In this connection, it has enforced RDB2RDF Incubator Group (XG) to study the process for recognition. XG recommends [8] to define a base for mapping language i.e.., RDB-to-RDF.

View updates have become a typical shortcoming in the research of the database systems. Many research activities in this area have stated that the requirements about the updates can be specified with VDL (View Definition Language). If VDL is built on the basis ofbijective mappings (i.e.., the views pertaining to the updates can be leveled on to the ontology). The problem of view update can be resolved to some extent and can be eradicated [9].

ORM (Object Relational Mapping) is another approach used for connecting the gaps existing during the mappings from RDB-to-RDF. ORM specifically uses Hibernate to persist the data objects existing in the object oriented applications. This benefits the abstraction of object-orientation in the relational model. The ORM then generates a new RDB schema facilitating storing and retrieving of objects.

3. RDB-TO-RDF INGRESSION APPROACH

The R2R Ingression based approach to access RDB's provides read/write access to relational schema. It primarily hostsupdate-aware mapping technique. R2RI connects the variations existing between the ontology and the RDB. It also connects the access interface to SPARQL/UPDATE which is an upcoming data manipulation tool.

Relational schema is updated to semantic web. It faces various challenges and issues for the mediation tool with the mapping languages. The gap which arises between the RDF and the Relational model (Triples Vs. Tuples) produces constraints from the RDB and are systematically transferred to the layer of Symantec web. The consequences arising after transformation helps the non-validation of few of the update requirements to the respective applications in the native Triple store. Since, the relational model is tuple oriented, it acquires data about each and every entity (for example – attributes in the entity are declared mandatory). The other existing requirements are indulged with the databases and their schema, influencing with various integrity constraints which may or may not be usually speculate in the RDF and ontologies. This happens more wisely with the vocabularies already reused and existed. However, to authorize the write access to the RDF through a specified ontology, various constraints and the resulted errors are respected and handled carefully. If the information is seated in the mapping based on these constraints, the requests relating to the non-validation updates are flexible enough to detect and enhance semantically good feedback to client.

We consider the schema of the RDB in a publication database relation, as one of the use case in this presentation. The database consists of information relating to authors and their esteemed publications. Considering Figure.1, it describes about the database schema considering an illustration with a table comprising of data types and their attributes. Each and every table has unique primary key known as id of data type integer. The tables, author and publications clearly represent the usages of main concepts in the use cases. The table comprises of publication title, publication type, publication year and publisher.

The year and title are the data attributes. The publisher and type are considered as the foreign keys associated to the illustrated two tables-publisher and the pubtype respectively. The pubtype and publisher

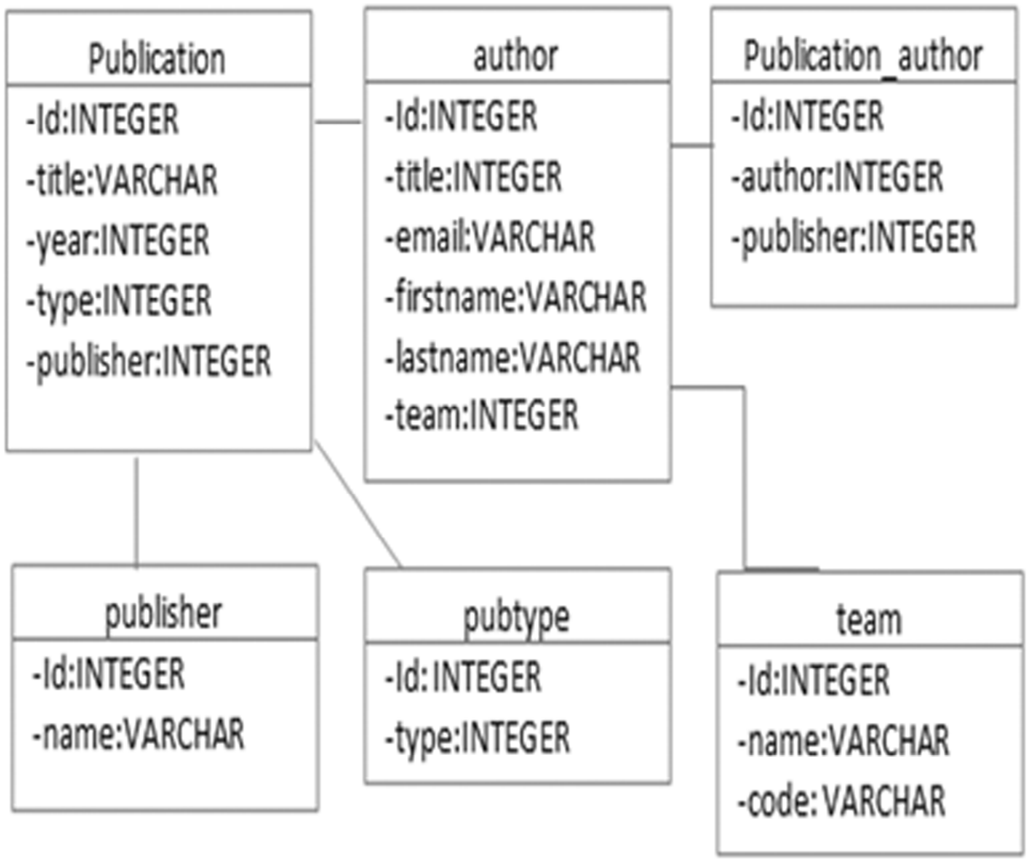

Figure 1: RDB Schema for publication

table consisting of one attribute as textual to the publication or to the publisher. All the sustainable publications must own a publication year and a title. Therefore, the illustrated table with given two attributes should not valid with NOT NULL values.

An Author table comprisesan email address, a title, an affiliation to research team, a last name and a first name. In author table, a valid and genuine author must constitute last name So that the NOT NULL attribute can be allocated to the attribute of last name. The ‘team attribute’ defines itself as a foreign key. All the leftover attributes existing in the author table comprises of data values.

In the above example, the table ‘team’ comprises of the information related to research groups.

The N:M relationship is duly represented by the newly formed link table by the publication author table that defines relations of authors and publications..

The domain ontology is employed to illustrate the given example with Figure.2. Hence, we have reused the vocabulary from Dublin Core Meta standard and Friend of a Friend (FOAF) project to create this ontology. We have also implemented our own elements (ONT) in the ontology, if there are similar terms existing in the two vocabularies. The figure also defines our domain ontology with respect to five classes with their properties that were being used in the respective range.

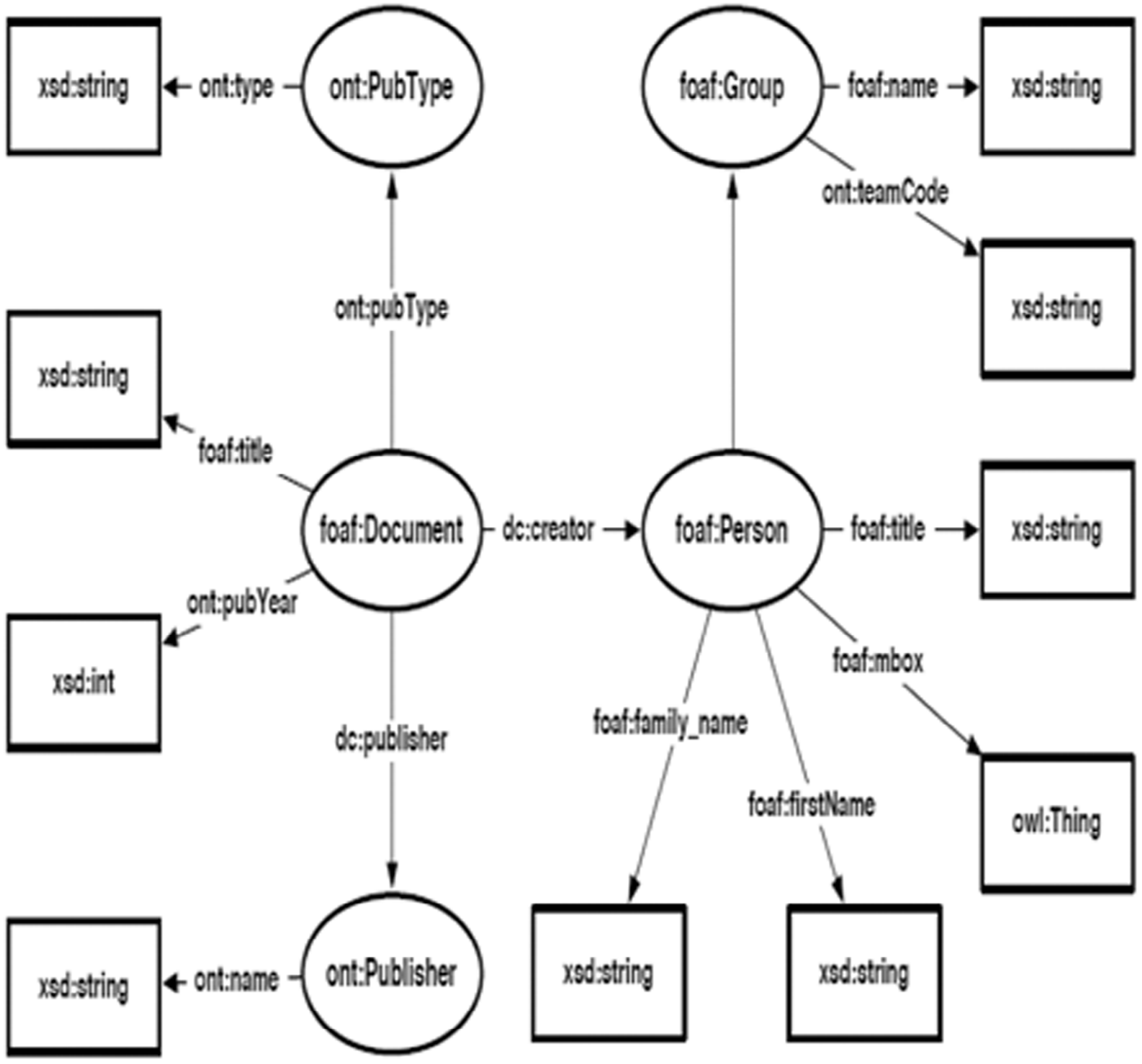

Figure 2: Domain Ontology

4. THE RDB-TO-RDF, R2RI MAPPING LANGUAGE

R2R Ingression which is also an RDB-to-RDF mapping language, maintains a catalogue of additional and required information to support various data manipulations incurred and also identifies the invalid update requests takes place during the process of translation. The simplicity and updatability were the two important goals of this kind of mapping language.

The mapping approach is implemented to map the table to convert attributes in to the properties and database tables to the respective ontology classes. This clearly enumerates every table with relational database with similar concept and is being mapped to specific ontology class. Similarly, each attribute in the data base constitutes a link between data value and an entity. So far relationship between these data values or entities is mapped to the ontology property that bonds instances of the classes to the literal values. From this, the table rows in the considered example are exactly mapped to the RDF triple sets. One among the set of triples identifies an entity which acts as particular instance to the class being mapped. Therefore, one triple in each data table attribute relates to another instance (for example-foreign keys) or to a data value. Link tables are generally used in the RDB's to define and describe the N:M relationships among several relations. The auxiliary constructs are not required and not in use in the RDF. That is why the reason R2R Ingression richly supports to map the tables to the object properties instead of the classes.

R2RI mapping functions with respect to root element is described in Listing 1.

a. The map apparently represents the accessibility information for R2RI mediator in lines 2 to 5 in the data base.

```
1  map:database a r2ri:DatabaseMap;
2  r2ri: jdbcDriver “com.mysql.jdbc.Driver”;
3  r2ri: jdbcUrl “jdbc:mysql://localhost/db”;
4  r2ri: username “user”;
5  r2ri: password “pw”;
6  r2ri: uriPrefix “http://example.org/db/”;
7  r2ri: hasTablemap:author, map:publication, map:publisher
8  map:publication_author, map:team, map:pubtype
```

Listing 1 Example for DatabaseMap

b. URI prefix in line 6 is specified for generating URI instances to all classes which we defined in the mapping.

   The instances of the URI primarily comprises of two parts out of which the foremost defines the URI prefix in the mapping and secondarily URI pattern specified and defined in each of the TableMap. The major importance of the URI prefix is to simplify the mappings which are alike to the mechanisms in the XML Namespaces.

c. All the tables which relate to this data base schema are briefly listed as TableMaps illustrated inlines 7 and 8.

```
1  map:author a r2ri:TableMap;
2  r2ri:hasTableName “author”;
3  r2ri:mapsToClass foaf:Person;
```

```
4 r2ri:uriPattern “author %%id%%”;
5 r2ri:hasAttribute map:author_id,
6 map:author_title, map:author_email,
7 map:author_firstname, map:author_lastname,
8 map:author_team.
```

Listing 2. Example For Table Map

A TableMap briefly describes the mapping pertaining to single data base table in the listing 2.

a. The TableMap at line2 specifies the name of the database table.
b. The considered and deployed ontology is mapped to line3.
c. The (URI) pattern is used to generate specific instance URI’s in line 4 for the class and the generate dinstance URI’s over rides the formed pattern if it is a valid URI. The values of the attributes are added by indicating the specific name of given attributes between the double percentage special symbols signs in the data base tables. Typically, the attributes with primary key are also associated in the (Universal Resource Identifier) URI pattern (for example-%%id%%, and id-name to primary key attribute).
d. A Table Map also consists of list Attribute Maps at lines 5 to 8 that maps the attributes of the given table to the properties on the ontology

Each and every attribute of given data base table is recognized by an Attribute Map in Listing 3

```
1 map:author-_team a r2ri:AttributeMap;
2 r2ri:hasAttributeName “team”;
3 r2ri:mapsToObjectProperty ont:team;
4 r2ri:hasConstraint [ a r2ri:ForeignKey;
5 r2ri:referencesmap:team.].
```

Listing 3 Example for Attribute Map

a. The *Attribute Map* represents the name in line 2 of the attribute in the data base schema.
b. The map also represents the name for the ontology property that is being mapped in line 3.Depending upon the value or type of the property attribute, the property can be assumed as a Data Property or an Object Property. The obtained Data Property reflects the mapping vocabulary either as r2ri: maps To DataPropertyor r2rimapsToObjectProperty.
c. The Attribute Map also includes some information regarding the constraints in lines 4 and 5 which are described on the attributes (for example-the foreign key to the table it refers). During the current implementation process, the following key constraints also supported: r2ri: Primary Key, r2riForeign Key, r2ri Not Null, andr2ri Default.

```
1 map:publication_author a r2ri:LinkTableMap;
2 r2ri:hasTableName publication_author”;
3 r2ri:mapsToObjectProperty dc:creator;
4 r2ri:hasSubjectAttribute map:pa_publication;
5 r2ri:hasObjectAttributemap:pa_author.
```

Listing 4 Example for LinkeTableMap

A LinkTableMap is usually used to map the LinkTables to the properties listed in 1.4 in the ontology.

a. A LinkTableMap indicates the name of the LinkTables being used in the data base at line 2.

b. LinkTableMap also indicates the name of the objective property being mapped at line 3.The LinkTable comprises of two foreign key attributes relating to N:M relationship. Hence, a triple represents this link with subject and object.

c. At line 4, the attribute of a subject (subject attribute) is created from the table.

d. Object attribute is constituted by the table at line 5.

The subject attribute and object attributes are linked to the AttributeMap, that they were not linked to any other property earlier. This maintains the track record of all the names of tables and the attributes they refer. It is discussed in the listing 5

```
1  map: pa_author a r2ri:AttributeMap;
2  r2ri: hasAttributeName “author_id”;
3  r2ri: hasConstraint [ a r2ri:ForeignKey;
               r2ri: references map:author.].
```

‘*Listing 5 Example for Attribute Map (not mapped)*

A basic R2R Ingression mapping can be easily generated from the data base schema if it exclusively renders the information presenting to the foreign key relationships. The Domain ontology terms cannot be easily assigned to individual concepts as per the mapping definitions in the data bases. However, the usage of Graphical tool assistance decreases the user’s efforts in defining the mapping scenarios.

5. THE SEMANTIC QUERY LANGUAGE-SPARQL/UPDATE TO SQL DATA MANIPULATION LANGUAGE

The World Wide Web Consortium (W3C) has recommended a language called SPARQL [PS08] for semantic web. This query language is exclusively meant for read-only data and does not provide any write access to inserting, deleting or modifying data to the RDF. In the due course of time, the community of semantic web has created a solution for the problem leading to SPARQL/Update [SMB08] which is also called data manipulation language. SPARQL/Update also serves and facilitates the updating feature through the W3C standards SPARQL working group (WG). The introduced update functionality of SPARQL Update comes with the different update operations.

1. DATA INSERTION: INSERT data in listing 6 is used for inserting new triples in RDF graph.
2. DATA DELETION: DELETE data in listing 7 is used for removing known triples in the graph.
3. MODIFY: The MODIFY operation work primarily for the two variants of SPARQL CONSTRUCT queries, and the resulting RDF triples are added or removed from the data. MODIFY in listing 8 is used for inserting or deleting the data from the triples. These are further coagulated to the triple pattern over a shared WHERE clause.

```
Listing 6   INSERT DATA: INSERT DATA {triples}
Listing 7   DELETE DATA: DELETE DATA {triples}
Listing 8   MODIFY: MODIFY DELETE {template}
INSERT {template} WHERE {pattern}
```

In [10], Angles and Gutierrez has manifested that the SPARQL too possess the similar expressiveness as like as relational calculus and subsequently the SPARQL can be freely construed to the SQL. From the above sentence, it is very clear that the SPARQL/Update is completely dependent on SPARQL and SPARQL/ Update is rendered to SQL DML.

5.1. INSERT/DELETE DATA Operations

The sets of triples comprising of data Insertion and data Deletion operations are either removed or added from the existing data. The translation process is very similar to SQL and slightly differs in the type of SQL statements being used. It is very vital to understand the process of translating for an algorithm and how exactly a relational schema is perfectly mapped in to the ontology. Such as tables consisting of native concepts are to be mapped to the classes, while the link tables and attributes are represented to as the ontology properties. Hence, we consider INSERT data operation as shown in Listing 9 which acts like an example, explaining the translation Algorithm1.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX ont: <http://example.org/ontology#>
PREFIX ex: <http://example.org/db/>

INSERT DATA {
ex: author6 foaf: title “Mr.”;
foaf: first name “sunny”;
family_name “Vedula”;
foaf: mbox<mailto :sunny@gmail.com>;
ont: team ex: team5
}
```

Listing 9 Example for INSERTDATA operation

```
INSERT INTO author (id, title, firstname, lastname, email, team,)
VALUES (6, ‘Mr.’, ‘sunny’, ‘vedula’, ‘sunny@gmail.com’, 5);
```

Listing 10 Translations, SQL Insert Statement

```
……………………………………………………………
Algorithm 1 R2RI Translation of the RDF triples in to SQL
…………………………………………………………….
1: subGroups: = group.Triples(t)
2: for all subGroup in subGroupsdo
3: table: = identifyTable(subGroup.getSub())
4: if (check(subjGroup, table)) then
5: sql: = generateSQL(subGroup, t)
6: statements.add(sql)
```

```
7: else
8: error()
9: end if
10: end for
11: sortedSql: = sortSQL(statements)
12: executeSQL(sortedSql)
```

STEP 1: STEP 1 describes that the triples are needed to be clubbed equally to the subjects and these clubbed triples affirms the data equally to its entity. The same is being targeted for the table. The triples, as we have seen in our illustration operation, all tend to utilize the similar and alike subjects. Accordingly, step 1 recur one group consisting of all native triples. The recurred groups are carefully handled individually at line 2.STEP 2: Under STEP 2 at line 3, the pretentiousset oftriples in the table are identified specially by the URI. The URI subject in our concerned illustration can be traced fromhttp://example.org/db/author 1. The URI's in the mappings as specified in the listings 1 and 2 clearly matches with the pattern http:// example.org/db/author%%id%% and considers this as the author table. From this point of view we derive the value 1 which can be used as the primary key attribute id.

STEP 3: In STEP 3 at the line 4, the validity and duration of the concerned request is obviously checked and examined if the existing data defined in the request meet the constraints defined in the schema. In INSERT data operation, the existence of a triple is mandatory that possess a property for each and every corresponding data base attribute with NOTNULL constraints and specifically there should be no Default value. This requirement is generally met with our INSERT data operation which contains triples and the properties of triples duly matching with each and every attribute in the *author* table.

STEP 4: The SQL statement is generated in step 4 at line 5 from the properties corresponding to the *TableMap* with the current subject, by adding the value resulted from triples object and also supplementing the attribute names. In the concerned example in the table-the property ont:team is searched and exactly projected to the corresponding attribute *team* in the listing 3. The name of the attribute and the value which is extracted from the object together are added to the SQL statement, namely 5. The leftover triples are also exercised in similar way. Steps 2 and 4 are often continued for several times to each and every set of triples and the resulting SQL assertions are grouped at line 6.

STEP 5: Once all SQL statements are collected by grouping of triples, these are now sorted depending upon the foreign key relationships with in the tables. This is not mandatory and specific, if all the SQL statements are executed in a single transaction. During the transaction process, the RDB system checks for constraint-Referential Integrity. As a matter of fact, execution of the already generated SQL statements may definitely hit the transactions for failures. In spite of the transaction with respect to the arbitrary order, it is better to perform transaction of SQL statements in the sorted order that produces better aspects. In our example, sorting is not precise because, we have only one SQL statement generated after sorting.

STEP 6: After generating and sorting, all the SQL statements are executed in step 6 at line12. The SQL statements with one SPARQL/Update operation is also executed in single data base transaction, so as to acknowledge the atomicity for committed operation. The listing 10 clearly depicts all the rendered SQL INSERT assertions duly generated from the example of our consideration SPARQL/Update, data insertion operation.

5.2. DATA INSERTION Operation

The data insertion operation of the SPARQL/Update can also be rendered to the SQL Data Manipulation Language as per the algorithm discussed in the above section. Relying upon the situation of the database, the translation process effects and yields either an UPDATE or INSERT INTO SQL statement. The triples in the RDF allows only minimal data to insert about an entity with data insertion operation Primarily (for example-the last name of the author) and secondarily adds further information (for example-first name and the e-mail address of the author).From the above it is very clear that the SQL INSERT operation inserts a new record in relational table (from the RDB perspective) including NULL values for the missing attributes. Secondly, the INSERT data operation (additional information) works in translating the SQL Update statement with replacing the NULL's with actual values. This means that it checks the entity if it already exists in databases and identifies the type of SQL generated.

5.3. DELETE DATA Operation

The DELETE data operation pertaining to the SPARQL/Update is translated according to the algorithm 1. The translation process of this algorithm 1 results in two different kinds of SQL affirmations based on the operation. The data which is undergoing for operation comprises of subset of data in the databases. The said operation activity is immediately translated to SQL UPDATE affirmative with all attributes being declared as NULL (this is permissible with all given constraints). The complete row of the relational table is removed if the data request matches with the data in the database table consisting of NULL. Therefore, the affected tuple is studied and retrieved back.

```
MODIFY
DELETE { ?x foaf: mbox ?mbox.}
INSERT {
?x  foaf:  mbox<mailto:sunny@example.com>.
}
WHERE {
?x rdf: type foaf:Person;
foaf:  firstname       “sunny”;
foaf:  family_name  “vedula”;
foaf:  mbox ?mbox.
}
```

Listing 11 Example for Modify Operation

```
DELETE DATA {
ex:  author6  foaf:  mbox<mailto:sunny@gmail.com>.
}

INSERT DATA {
ex:  author6  foaf:  mbox<sunny@gmail.com>.
}
```

Listing 12 Yielded DELETE DATA operation and INSERT DATA Operation

5.4. The MODIFY Operation

The Modify operation is not intended to translate directly to the SQL since SQL DML lacks its matching statement. The operation is a unique combination of both INSERT and DELETE that enables the featuring the replacement of the triples. So that adding /removing of arbitrary triples can be performed with this operation. In contrast to this, the update operation is very limited in modifying the existing data. However, the SPARQL grammar can be reused in many waysas SPARQL/Update can also be used in the multiple steps as possible.

Algorithm 2 clearly pictures the ways of MODIFY operation is used for translating to the SQL. The listing 11 clearly pictures the usage and implementation of MODIFY operation as an example in the Algorithm. It allocates the e-mail address to “sunny” as (sunny@example.com). The MODIFY operation is categorized in to individual parts, WHERE, INSERT and DELETE clauses in lines 1 to 3.

```
……………………………………………………………
Algorithm 2 R2RI Modify operation in to Sqldml translation
…………………………………………………………….
1:  delete: = extract Delete (modify)
2:  insert: = extract Insert (modify)
3:  where: = extract Where (modify)
4:  select: = create Select (where)
5:  select SQL: = translate Select (select)
6:  results: = execute SQL (select SQL)
7:  for all binding in results do
8:  delete Data: = create Delete Data (delete, biding)
9:  insert Data: = create Insert Data (insert, binding)
10: delete SQL: = translate Delete (delete Data)
11: insert SQL: = translate Insert (insert Data)
12: execute SQL (delete SQL, insert SQL)
13: end for
```

The WHERE clause is used forcreating the SPARQL SELECT query in line 4 in retrieving the data that is wanted by the INSERT and DELETE templates. The WHERE part is translated to SQL in line5 and is duly evaluated to the relational data in the line 6. Based on the results of translated query, ONE INSERT in line 9 and one DELETE in line 8 operations are built and combined with each binding in line 7, as per the INSERT and DELETE templates of MODIFY operation. The SELECT query in our example returns only one binding (ex:author6 for x variable and mailto:sunny@gmail.com for mbox). Therefore, one data insertion and one data deletion are built on the binding as given in listing 12. Now the operations are translated in lines 10 and 11 as per the specifications described in Algorithm 1 in the previous sections.

In most of the cases, the MODIFY represent replacement of triples or modification of the data. In such cases, optimizing is enhanced by removing DELETE DATA operations corresponding to INSERT DATA (triples may differ in their objects). Under such circumstances the insert sets same and similar attribute to the new value and delete sets NULL attribute value. Hence, the delete is redundant and is liable for omission and further omitted.

6. THE PROTOTYPE IMPLEMENTATION

The mapping languages and their algorithms (SPARQL/Update) and (R2RI to SQL DML) are described during our presentations. We have developed a prototype, which mediates between the RDB and SPARQL/ Update requests. The prototype which is implemented as an HTTP endpoint, paves path for all clients to login through remote server to alter the relational schema. All the SPARQL/Update affirmations are analyzed syntactically from the requests of HTTP and are redirected to rendition module. The algorithm prescribed is substantially used to generate the equivalent SQL affirmatives that depend upon R2RI mapping definition. The translation operation further is executed and examined by the database engine. Thus, results an error message or confirmation message. The error message is retraced back to translation module for further rectification. The error message after confirmation is transformed to RDF representations and then redirected to the respective client.

At present, the prototype implementation has petty limitations to DELETE and INSERT data operations. But, also facilitates a great support for MODIFY, such that many SQL queries are implemented which are under development. Besides this, a feedback protocol is wisely implemented and planned to facilitate parsing error information to client. The future work in the implementation would involve better enhancements and features to the users

7. THE FEASIBILITY STUDY

In our primary evaluation, we present an approach of the feasibility study based on Domain ontology and the RDF schema. The Table 1 clearly summates the mappings from attributes and tables of the database schema to their properties and classes of Domain ontology. The first and foremost column in the table pinpoints the table’s corresponding class. column 2 lists out the properties of each table in the schema and demonstrates their attributes for which they are mapped:

The publication table mentioned in the Table 1 is exactly mapped to its corresponding class foaf. Document. The publisher attribute and title attribute is mapped to properties, while attributes type and year use the properties existing in our ontology ONT. The table pubtype and publisher with all their attributes are plotted to our ontology. The table ‘author’ is treated as foaf:person. All its attributes are mapped corresponding to its equivalent concepts though the FOAF vocabulary. The FOAF vocabulary except the table class that uses the property from the ontology ONT. The table ‘team’ is represented with the foaf:Group and its code attribute is being mapped to ont:teamcode and name to foaf:name. The table publication_author forms as a link table representing N:M relationship existing between the author and publication. Therefore, publication author is not mapped to class instead to property dc:creator.

All the mapping definitions used in the earlier sections states that the mediation-prototype is used for processing and executing the operating existing in SPARQL/Update. In the last but not the least, we have clearly presented the examples using SPARQL/Update and the generated SQL statements after generation.

In listing13, it explains about SPARQL/Update operation data insertion technique, that specifically inserts some data about team.

In Listing 14 the data insertion operation affects the database table and then translated to SQL INSERT statement.

Listing 15 clearly describes about the complex handling of INSERT DATA Request. It comprises of a complete set of data i.e.. The INSERT DATA request ensures in inserting new elements of data in to each and every database table and thus generating group of SQL statements. The SQL statements duly generated with the foreign key and tables are irrelevant with respect order of triples.

Listing 16 clearly depicts the order of the SQL statements being generated based on the order of their execution

Table 1
Use Case Mapping Overview

table → class	*attribute → property*
publication → foaf: Document	title → dc:title
	year → ont:pubyear
	type → ont:pubtype
	publisher → dc:publisher
publisher → ont:publisher	name → ont:name
pubtype → ont:pubtype	type → ont:type
author → foaf:person	title → foaf:title
	email → foaf:mbox
	firstname → foaf:firstname
	lastname → foaf:family_name
	team → ont:team
team → foaf: Group	Name → foaf:name
	code → ont: teamcode
Publication_author → –	– → dc:creator

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX ont: <http://example.org/ontology#>
PREFIX ex: <http://example.org/db/>

INSERT DATA {
ex:team4 foaf:name “Software engineering”;
ont: teamcode “SE”.
}
```

Listing 13 Example for INSERT DATA Operation

```
INSERT INTO project (id, name, code)
VALUES (6, ’software engineering’, ’SE’);
```

Listing 14 Translated into SQL INSERT Statement

```
PREFIX     foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ont: <http://example.org/ontology#>
PREFIX ex: <http://example.org/db/>

INSERT DATA {
ex:pub12 dc:title “Relational ...”;
ont:pubyear “2015”;
ont:pubtype ex:pubtype4;
dc:publisher ex:publisher3;
dc:creator ex:author 6.

ex:author6 foaf:title “Mr.”
foaf:firstname “sunny”;
foaf:familyname “vedula”;
foaf:mbox <mailto :sunny@gmail.com>;
ont:team ex:team 5.

ex:team 5 foaf:name “Software Engineering”;
ont:teamCode “EXPERT GROUP” .

ex:pubtype4 ont:”inproceedings”

ex:publisher3 ont:name “springer”
}
```

Listing 15 Example for INSERT DATA Operation

```
INSERT INTO team (id, name, code)
VALUES (5, ’Software Engineering’, ’expertgroup’);

INSERT INTO pubtype (id, type)
VALUES (4, ’ inpreceedings’);

INSERT INTO publisher (empid, name)
VALUES (3, ’springer’);

INSERT INTO publication (id, title, year, type, publisher)
VALUES (12, ’Relationaldatabase’,’2014’, ’4’, ‘3’);

INSERT INTO author (id, title, firstname, lastname, email, team)
VALUES (6, ’Mr.’, ’sunny’, ’vedula’,’sunny@gmail.com);

INSERT INTO publication_author{publication, author}
VALUES (12, 6 );
```

Listing 16 Translated into SQL INSERT Statements

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX ont: <http://example.org/ontology#>
PREFIX ex: <http://example.org/db/>

DELETE DATA {
ex: author6 foaf: mbox<mailto:sunny@gmail.com> .
}
```

Listing 17 Example for DELETE DATA Operation

Listing17 presents an example pertaining to SPARQL/Update data deletion that substantially eradicates the email address of the respective author.

```
UPDATE employee
SET phone = NULL
WHERE id = 5 AND phone = 9834020094;
```

Listing 18 Translated into SQL UPDATE Statement

Listing 18 theentries in the author tables comprises of large information rather than single entry (e-mail address information), the request is further rendered to a SQL Update affirmativepertaining to the Algorithm 1.

All the requests relating to SPARQL/Update focused in this area of presentations are considered to be only examples, where the user is very free enough to express the capricious requests. The capricious requests are further rendered to SQL DML affirmations relating to the ontologyterms and honor the respective constraints of the database.

8. CESSATION

In this contribution, we presented the manipulations of the relational data through SPARQL/Update by using the Ingression approach. We have also introduced the RDB-to-RDF mapping language R2RI which apprehend necessary information (about integrity constraints) in the database schema. This particular data information confesses the observation of invalid update requests with respect to the RDB. Such a kind of invalid requests are not compiled and executed by database engine because they strictly violate the integrity constraints in the database schema. The information is exploited simply to facilitate the parsing feedback to client. Therefore, the probable and inducement causes for disapproving of these requests are highlighted and possible suggestions for improvements are also represented in suitable format.

REFERENCES

[1] T. Berners-Lee, J. Hendler, and O. Lassila. TheSemantic Web. Scienti_c American, 2001.

[2] A. Schultz and C. Bizer.Benchmark the Performance issues of Storage Systems. Proceedings of the IvthInternational Workshop on “Scalable SW Knowledge Base Systems” at the International SW Conference-2008.

[3] S. Conrad & C. P. Laborda Relational.OWL [A Data and Schema Representation on OWL. “Proceedings of the IIndAsia-Pacifc Conf.. on Concept Modelling”-2005.

[4] M. Zloch, C. P. Laborda and S. Conrad. R. D. Query [Querying Rdb on-the-y with RDF-QL. Proceedings of the XVth International Conf. on Knowledge Engineering and Management-2006.

[5] I. Mikhailov and O. Erling. RDF Support in Virtuosoo RDBMS. Proceedings of the SABREConf. on SW-2007.

[6] S. Auer, J. Lehmann, and D. Aumueller. Triplify [Light Weight Link Data Publication in rdb. Proceedings of the XVIIIth International World Wide WebConf..-2009.

[7] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, A. Poggi, and R. Rosati. Mastro-i:E_cient Integrating Relational.Data via DL Ontologies. Proceedings of the XXth InternationalWorkshop on Descript Logics-2007.

[8] A. Malhotra. W3C RDB-to-RDF Incubator expert Group Report. http://www.w3.org/2005/Incubator/rdb2rdf/XGR-rdb2rdf-20090126/,2009.

[9] A. Bohannon, B. C. Pierce, and J. A. Vaughan. Relational Lenses: Language for Updating Views. Proceedings of the XXVth ACM SIGACT-SIGMOD

[10] E. Prud'hommeaux and A. Seaborne. SPARQL for RDF. W3C Recommendation.http://www.w3.org/TR/rdf-sparql-query/, 2008.