An Approach for Automatically Generating R2RML-Based Direct Mapping from Relational Databases

Mohamed A.G. Hazber[1], Ruixuan Li[1(✉)], Guandong Xu[2], and Khaled M. Alalayah[3,4]

[1] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China
moh_hazbar@yahoo.co.uk, rxli@hust.edu.cn
[2] Faculty of Engineering and IT, University of Technology Sydney, Sydney, Australia
guandong.xu@uts.edu.au
[3] Computer Science, IBB University, Ibb, Yemen
kh101ed2005@yahoo.com
[4] Computer Science, Najran University-Sharurah, Najran, Saudi Arabia

Abstract. For integrating relational databases (RDBs) into semantic web applications, the W3C RDB2RDF Working Group recommended two approaches, Direct Mapping (DM) and R2RML. The DM provides a set of mapping rules according to RDB schema, while the R2RML allows users to manually define mappings according to existing target ontology. The major problem to use R2RML is the effort for creating R2RML mapping documents manually. This may lead to appearance of many mistakes in the R2RML documents and requires domain experts. In this paper, we propose and implement an approach to generate an R2RML mapping documents automatically from RDB schema. The R2RML mapping reflects the behavior of the DM specification and allows any R2RML parser to generate a set of RDF triples from relational data. The input of generating approach is DBsInfo class that automatically generated from relational schema. An experimental prototype is developed and shows the effectiveness of our approach algorithms.

Keywords: Relational Database to Resource Description Framework (RDB2RDF) · Direct Mapping · R2RML · Relational database · Resource Description Framework (RDF)

1 Introduction

The continuous explosion of resource description framework (RDF) data opens door for new innovations in big data, social network analysis, and semantic web initiatives, which can be shared and reused through the application, enterprise and community boundaries. The semantic web [1] is one of the most important research fields that aim to construct a web of data based on the RDF [2]

W. Che et al. (Eds.): ICYCSEE 2016, Part I, CCIS 623, pp. 151–169, 2016.
DOI: 10.1007/978-981-10-2053-7_15

data model. It allows data to be shared and reused through applications, enterprise and community boundaries. Relational databases (RDBs) are the primary sources of web data, "deep web" [3]. The main reason is one of the studies [3] showed that internet accessible databases contained up to 500 times more data compared to the static web, and roughly 70 % of websites are backed by RDBs. The W3C RDB2RDF Working Group recently recommended a specification for languages to map RDB (data and schemas) to RDF and OWL, tentatively called Direct Mapping (DM) [4] and R2RML (Relational Database to RDF Mapping Language) [5]. However, the W3C Working Group does not recommend any implementation for DM and R2RML. The DM provides a set of automatic mapping rules to construct an ontology schema (RDF(S) and OWL) from RDB schema and convert relational data to RDF graphs according to that schema [6]. The ontology constructed reflects the structure and content of the relational database. Nevertheless, the DM method may not be constantly sufficient or optimum, especially when mapping a relational database to an existing ontology. R2RML is a customized mapping language, which allows users to define mappings manually. In this approach, the expert user expresses the RDB schema using an existing target ontology in order to convert the relational data into RDF datasets.

The R2RML specification is accompanied by the DM specification [4], representing a standard approach for converting an RDB into RDF without the use of a customized mapping definition. Thus, the RDF generated using DM can be represented in R2RML. R2RML provides more flexibility than DM specification. Meanwhile, creating R2RML rules by domain experts manually is complex, time consuming process, cumbersome, mistakable, high cost process, and requires the supports of domain experts in knowledge acquisition. Moreover, the users who are interested to apply the R2RML for RDF generating from RDB are requested to learn how to create an R2RML mapping document, in addition to a significant gap between the structure of RDB and the R2RML mappings specifications. One of the ways to solve those problems and ease-of-loading for creating an R2RML document from users efficiently is to generate an initial R2RML mapping document automatically from RDB schema that reflects the conduct of the DM specification. Afterward users will be able to modify that document into a text editor or user interface (display screen). Thus, making the process engine of generating RDF triples (such as morph-RDB [7], nknos [8], RDF-RDB2RDF[1], etc.) takes the R2RML mapping document and RDB data as an input, and then provides an output corresponding RDF dataset (triples). This is done by automatically mapping RDB concepts to an ontology vocabulary, which could be used as a base to support generating RDF triples from RDB data. Recently, the two reports presented by a survey report [9] and W3C's RDB2RDF Implementation [10] are discussed and listed a few existing tools or ongoing projects that have been made available to support the task of mapping generation. However, some of those tools either create mappings in RDB2RDF languages such as ODEMapster GUI (creates R2O mappings) or only give

[1] https://metacpan.org/release/RDF-RDB2RDF.

syntactic sugar (form-based tools) to users, who still require a good knowledge of R2RML, which makes them not usable enough.

In this paper, we design and implement algorithms to automatically generate R2RML mapping documents that reflect the behavior of the Direct Mapping specification, which will be applicable as a base support generating the RDF triples from RDB data. Firstly, we design and implement an algorithm that takes an RDB schema as an input and extracts a DBsInfo class (has all the information about RDB schema) as an output. Secondly, we present an algorithm design approach to automatically generate an R2RML mapping document based on a DBsInfo class. Subsequently, generating an RDF dataset by integrating our work with the R2RML processor, which takes an R2RML mapping document and RDB data as inputs and generates the RDF triples as an output. The experimental results show important factors for the building R2RML mappings and their influence on the mapping generation time and size of R2RML and RDF file. These results together reflect the effectiveness of our algorithm and its implementation in Java with Jena API.

The rest of the paper is organized as follows. Section 2 provides an overview of the related works. Basic concepts which give a brief overview of the R2RML and DM with the relationship between them are described in Sect. 3. Section 4 proposes the approach and the algorithm. A prototype implementation of the architecture of our processor prototype and experimental results with discussion regarding the effectiveness and the run-time efficiency test on the proposed algorithms are presented in Sect. 5. Finally, Sect. 6 concludes this paper with the future work.

2 Related Work

Several approaches (auto or manual) have been proposed in the integrating RDB and semantic web, mainly concerning the creation and maintenance of mappings between RDF and RDB. Mapping RDB to RDF is a domain where quite a few works have been proposed over the last years [11]. Generally, the objective is to express the RDB contents using ontology (RDF graph) in a way that allows queries submitted to the RDF schema to be answered with data stored in the RDB. Also, for bringing data residing in RDB into the semantic web, several automated or semi-automated methods for ontology schemes representation have been created [12–14].

Currently, there are two main approaches recommended by W3C RDB2RDF Working Group for mapping RDB into RDF that we have mentioned previously: DM [4] and R2RML [5]. In the DM approach the ontology model is constructed from RDB model, and the contents of the RDB are transformed to generate ontology instances [6,12,15,16]. The approach [6,12] proposed (automatic-direct mapping rules) by investigating several cases of RDB schema to be directly mapped into ontology represented in RDF(S)-OWL and transformed RDB data to ontological instances (represented in RDF triples) based on the structure of the database schema. While in approach [16] a tool RDB2OWL language for

mapping a database into an ontology in a compact notation within the ontology class and property annotations was presented. This tool was implemented by converting the RDB2OWL mappings into executable D2RQ mappings to produce the RDF dump of the source RDB, or to turn it into an SPARQL endpoint.

On the other side, the customized mapping approach such as ODEMapster [17], Triplify [18], D2R Server [19], and OpenLink Virtuoso [20] lets a domain expert to create a mapping between the relational schema and an existing target ontology, which is used to convert RDB content to RDF. However, early surveys of RDB-to-RDF tools [21] revealed that the tools typically adopt proprietary mapping languages. Triplify [18] offers a Linked Data publishing interface and provides a simplistic approach to publish RDF from RDB. D2R Server [19] is an engine that directly maps the RDB into RDF and uses D2RQ mappings to translate requests from external applications to SQL queries on the RDB. This implementation was first available for the D2R language and later for R2RML. Moreover, there are some tools for implantation DM and R2RML such as r2rml4net[2] and db2triples[3]. The r2rml4net is a library for processing the R2RML mapping documents, which provides functions to load R2RML mapping document and functions to convert relational data to the RDF dump. The db2triples-software tool is an RDB2RDF Antidot[4] Java implementation of the DM specifications and the R2RML for extracting data from RDBs and loading data into an RDF triple store. Recent efforts offered MIRROR system [22] for produce mappings in the R2RML language and an RML mapping language [23], an extension of R2RML, for non-relational sources and the integration of heterogeneous data formats to support XML and JSON data sources expressions in the mappings. In this work, we focus on the RDB schema. Meanwhile, other researches introduced a semi-automatic mapping approach for generating R2RML mappings based on a set of correspondence assertions (mapping between relational metadata and the vocabulary of a domain ontology) defined by domain experts [24,25]. Therefore, the user still needs to draw correspondence assertions (CAs) from the input system (source RDB schema and target ontology/RDF schema) to specify the mapping between them.

Based on the previous literature, mapping generation remains far from well understood and need to be further explored. Therefore, generation of R2RML mapping documents automatically from RDBs becomes an important challenge to avoid appearance of mistakes in R2RML mappings in addition, it reduces the generation time and no need for domain experts.

3 Basic Concepts

This section gives a brief overview of the R2RML and DM with the relationship between them. The W3C has recently standardized the RDB-to-RDF

[2] https://bitbucket.org/r2rml4net/core/wiki/Home.
[3] https://github.com/antidot/db2triples.
[4] http://www.antidot.net/.

(RDB2RDF) mapping mechanism and language to bridge the gap between RDBs and the semantic web. These standardized namely Direct Mapping (DM) of relational data to RDF [4] and R2RML: RDB to RDF mapping language [5]. The mapping engine of approaches/tools generates RDF dataset from RDB schema and its instances. The main step in this engine is to decide how to represent RDB schema concepts in terms of RDF classes and properties from tables and columns. This is done by mapping RDB concepts to an ontology vocabulary, to be used as the base to generate a set of RDF triples from relational data.

3.1 R2RML Standard

R2RML is a language for describing customized mappings from a relational database to RDF dataset. The input of an R2RML mapping is an RDB schema and its instance. The output is an RDF graph. This mapping definition is represented as an RDF graph using the R2RML vocabulary and serialized in the RDF Turtle syntax (RDF triple Language) [26] which is the recommended syntax to write R2RML mapping documents. The structure of an R2RML mapping document consists of one or more triples maps, which contains a logical table, a subject map, and a number of predicate-object maps. The logical table can either be an SQL table, an SQL view, or an SQL query statement. The triples map specifies a rule for mapping each row of a logical table to a set of RDF triples. The subject map contains the rules for generating the subject for each row, often represented as an IRI. While the predicate-object map contains the rules for generating a predicate maps and object maps (or referencing object maps) from the values in the table row. The referencing object map allows using the subjects of another triples map as an object. Since both triples maps may be based on different logical tables, it may require a correlation between the logical tables.

Furthermore, a triples map specifies RDF triples corresponding to a logical table while the subject map and the number of predicate-object maps used to specify how the triples should be. So, RDF triples are created by combining the subject map with a predicate map and a (referencing) object map, and applying these three to each logical table row.

3.2 Direct Mapping (DM)

The DM is a notable one as the W3C candidate recommendation [4]. It is default method to translate a relational database (schema and data) to an ontology (OWL/RDF(S) and RDF triples) automatically through directly mapping without user interaction. The ontology represented in OWL/RDF(S) format. The RDF will reflect the exact data model of the relational data, rather than the domain of the data. A direct mapping is typically working by transform each table to a class, column to property, and relationship to an object property. Each row in the table will be transformed to an individual that will be a member of

the table's class. The foreign key transformed with a property that links one individual to another. The range of other properties will be literals.

Furthermore, generating IRI (prefix-name space) for the triples of the RDB schema and data (tables, columns in a table, and each row in a table) during the mapping process produced by combining base IRI and table name for each table, and base IRI, table name and column(s) name for each column in table. While the IRI for each row in table produced by combining base IRI and primary key column(s) of the table.

Therefore, a DM is the default and automatic way to translate RDBs into RDF without any input from the user, while R2RML is a mapping language, which allows users to manually define mappings. Thus, the DM can be represented in R2RML, which is accompanied with the DM specification, describing a standard method for generating RDF from RDB without using a customized mapping definition.

4 Approach and Algorithm

In this section, we introduced an approach that provides (RML-BDM) R2RML based on direct mapping rules from RDB to RDF(S)-OWL for automatically generating an R2RML mapping document from an RDB schema. Then any R2RML engine (e.g. nknos-r2rml parser) can be used to create the RDF dataset following the DM specification.

4.1 RDB Metadata Generation (DBsInfo Class)

In this section, we introduce a DBsInfo class, as a representation of an RDB's metadata, to be used as a source of information for RML-BDM generation. Basic information needed to proceed includes table names, view names and columns' properties that include column names, data types, size, and whether the column is nullable, index, primary key (PK), unique key (UK), and/or foreign key (FK). Moreover, the most important information needed when the attribute is FK are Ref_to_Table (reference to table) and Ref_to_Column (reference to column). The processor for producing a DBsInfo class, which contains all the information about the database, is shown in the Fig. 1. This processor has three levels, which contain forth algorithms (classes) to extract all the information about database tables, views, columns, datatype, columns properties, PKs, FKs, UKs, columns index, and relationships between tables through the foreign keys, etc. These algorithms are FillDBsInfo, FillTablesInfo, FillColumnsInfo, and FillTableRelationships.

Briefly, FillDBsInfo is the main algorithm that extracts the general important information about the database and invoking the FillTablesInfo algorithm to represent the functionality of tables and views. Algorithm FillTablesInfo extracts all the information about table and view, in addition to the information of table columns and table relationships with other tables by invoking algorithm FillColumnsInfo and algorithm FillTableRelationships, respectively.

A DBsInfo provides an image of metadata obtained from an existing RDB. The main purpose behind constructing a DBsIno class is to read essential metadata into memory outside the database's secondary storage. In this study, the DBsInfo class is designed to upgrade the semantic level of RDB and to play the role of an intermediate stage for database migration from RDB to RDF acting on both levels: schema translation and data conversion.

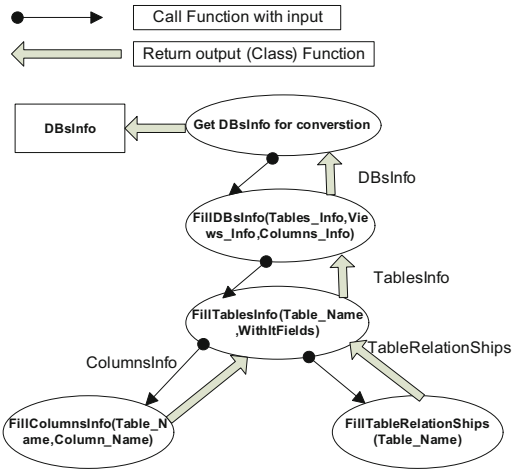

Fig. 1. Processor of algorithms for extracting RDB metadata (DBsInfo).

4.2 Rules of Approach: R2RML-Based Direct Mapping

This section defines the algorithms, which the mapping RDB schema to R2RML file is based on DM approach (RML-BDM). The RDB2RDF algorithm is the core of an R2RML engine. According to the algorithm ideas proposed in W3C recommendation R2RML [5], DM [4] and our previous works [6,12], we have designed a group of mapping algorithms, **R2RML Generator** (Algorithm 1), **GreateMapClass** (Algorithm 2), **GenerateLogicalTable** (Algorithm 3), **GenerateSubjectMap** (Algorithm 4), **GenerateTempate** (Algorithm 5), **GeneratepredicateObject** (Algorithm 6), and **GenerateRefObjMap** (Algorithm 7), to achieve the R2RML-BDC mapping file. Concisely, **R2RML Generator** is the main algorithm invoking the other algorithms to implement the functionality of R2RML triples map.

- **R2RML Generator:** This algorithm is the main algorithm to generate the R2RML mapping file based on the direct approach mapping.
- **GreateMapClass:** This algorithm creates map class name from the table/view name. The map class name is a triples map that used to translate each row in the logical table to number of RDF triples and link-connect between triples map classes (classes corresponding to tables that have the

relationships with each other). The output is a map class name corresponding table/view name.

- **GenerateLogicalTable:** This algorithm maps the table/view into logical table using the r2rml format depending on DM method, where the table name of RDB is rr:tableName in the logical table. This reflects which table/view is mapped to generate triples from its rows by using an R2RML parser. Then the return of this algorithm is the RDF triples.
- **GenerateSubjectMap:** This is one of the important algorithms. It generates the unique IRI used as a subject for all the RDF triples generated from the row of the table in rr:tableName. This algorithm invokes the GenerateTemplate algorithm to the identification form of the primary key of each triple generated from the row of table/view. The algorithm's input is a table/view name of DBsInfo.TablesInfo class. The DBsInfo.TablesInfo class used to store all the columns of the table/view, the properties of the table and properties of its columns, and its relationships to other tables.
- **GenerateTemplate:** This algorithm of generating IRI format for all triples from base IRI, table name, and table columns especially from primary keys (PKs), unique keys (UKs), or collect some columns that are not-null (when the table does not have any PK or UK defined). Therefore, this algorithm is characterized by the formation of a basic IRI key that is unrepeatable for all the triples generated from the table rows, where each row maps to a set of triples refer to the same subject (IRI key row) and all the triples of the table rows refer to the table name in rr:Class. Then the return of this algorithm is the RDF triples format in r2rml.
- **GeneratepredicateObject:** This algorithm maps the table/view column to PredicateObjectMap that includes a pair of predicate and object map. It generates the RDF terms for the predicate and object of a triple respectively. The value of rr:predicate is IRI consists of base IRI, table name and the columns name which are the algorithm inputs. The rr:object is a column name. Then the return of this algorithm is the RDF triples format in r2rml that will be associated with a subject (generated by the GenerateSubjectMap algorithm).
- **GenerateRefObjMap:** This algorithm maps all the table relationships to the reference triples, which are generated for *referencing object maps*, through a rr:joinCondition to another table similar to local triples. The *referencing object map* allows using the subjects of another triples map as an object (produced by a predicate-object map). Since both triples maps may base on different logical tables, it may require a link between the logical tables. All relationships of the table are stored in the DBsInfo.TableRelationShips, including FK_Table_Name, FK_Column_Name, Ref_To_TableName, and Ref_To_Column_Name, which are the inputs of algorithm. Then the return of this algorithm is the RDF triples format in r2rml corresponding to relationship of table with other tables. The output of the algorithm is associated with the objectMap generated by a PredicateObjectMap.

Algorithm 1. R2RML Generator (DBsInfo, NS Prefix)

```
Input: DBsInfo Has all information of database schema, NS Prefix;
Output: R2RML Mapping File: RDF Dataset;
Var
TBs: TableInfo[] //Class as List to save list information of tables in DBsInfo;
Col: ColumnsInfo[] //Class as List of table columns information;
TBR: TableRelationShipInfo[]//Class as List of table relationships;
Triples:String //to save triple generated;
Begin
Triples ← θ;
TBs ← DBsInfo.Schema_TableInfo;
for each table T in TBs do
    Triples ← CreateMapClass (T.Table_Name) ;
    Triples ← Triples U GenerateLogicalTable (T.Table_Name, T.Table_Type) ;
    Triples ← Triples U GenerateSubjectMap (T.Table_Name, T) ;
    Col ← T.Table_ColumnInfo;
    for each Column C in Col do
        Triples ← Triples U GeneratepredicateObjectMap (T.Table_Name, C.Column_Name) ;
    end for
    TBR ← T.Table_RelationShipInfo;
    for each TableRealtionShip TR in TBR do
        Triples ← Triples U GenerateRefObjectMap(TR.getForeignKeyTable_Name () ,
                TR.getForeignKeyColumn_Name () , TR.getReferenceToPKTable_Name () ,
                                                TR.getReferenceToPKColumn_Name ());
    end for
end for
Triples ← Triples U "." + "\n\n";
R2RMLMapping File ← Triples;
Return R2RML Mapping File // to Save it in RDF File Format Or TTL
End
```

Algorithm 2. CreateMapClass(Table_Name)

```
Input: Tabl_Name: Name of table or view that will be mapped;
Output: MapClassName : Triples //ex map:Students;
Begin
MapClassName ← "map : " + Table_Name + "s" + "\n";
return MapClassName;
End
```

5 Prototype Implementation

5.1 Architecture

Figure 2 shows the architecture of our RML-BDM processor prototype. Depending on the proposed algorithms, we have implemented an R2RML-BDB processor prototype and have been integrated with nknos-r2rml parser[5] [8]. The processor takes system configuration, a DB connection to the relational database and a base IRI as inputs and produces automatically the R2RML mapping document and resulting RDF dataset as outputs shown by screen display.

The architecture and process flow of the R2RML-BDB processor prototype is illustrated in Fig. 2, where the functional modules are briefly described as follows.

[5] https://github.com/nkons/r2rml-parser.

Algorithm 3. GenerateLogicalTable(Table_Name,Table_Type)

1: **Input**: Tabl_Name: Name of table or view that will be mapped;
2: Table_Type: table or view;
3: **Output**: Triples :Map table or view to Triple of LogicalTable;
4: **Begin**
5: $Triples \leftarrow \theta$;
6: $Triples \leftarrow Triples\ U$ "$rr : logicalTable[rr : tableName'$" + "\"" + $Table\_Name$ + "\"" + "';];";
7: $Triples \leftarrow Triples\ U$ "\n";
8: **return** Triples;
9: **End**

Algorithm 4. GenerateSubjectMap(Table_Name,T)

1: **Input**: Tabl_Name: Name of table or view;
2: T : has all table Information (DBsInfo.Schema_TableInfo);
3: **Output**: Triples :Map Row ID in table to subjectMap;
4: **Begin**
5: $Triples \leftarrow \theta$;
6: $Triples \leftarrow Triples\ U$ "$rr : subjectMap[$" + "\n";
7: $Triples \leftarrow Triples\ U\ GenerateTemplate(Table\_Name, T)$;
8: $Triples \leftarrow Triples\ U$ "$rr : class\ \ NS :$" + $Table\_Name$ + ";\n";
9: $Triples \leftarrow Triples\ U$ "];";
10: $Triples \leftarrow Triples\ U$ "\n";
11: **return** Triples;
12: **End**

- *System config module:* This module configures the execution environment for the R2RML-BDB processor according to the user-specified settings, including:
  1. DB config: This is used to specify all parameters for connection to any database.
  2. R2RML file input type: It is used to specify the file name and type of R2RML document that will be used for storing an R2RML schema generated from RDB schema and then used it with any R2RML Parser to produce RDF triples from RDB data.
  3. RDF triples output type: It is used to specify the name file and type (format) of RDF graph to be used for storing RDB data as RDF graph format.
  4. Base IRI (NS Prefix): This NS-IRI is used to specify the namespace prefix of IRI for all the RDF triples.
- *DB connection:* This module uses to connect with the database (using JDBC driver engine in Java) and make it ready for reading. The input is DB config parameters and the output is DB-connection class.
- *DB analysis processor:* This module implements the algorithms of extracting RDB metadata (DBsInfo) (Fig. 1) from RDB. Metadata is extracted from RDB using JDBC driver engine in Java. The output is DBsInfo class that has many classes to store all the information about RDB schema such as

Algorithm 5. GenerateTemplate(Table_Name,T)

```
1: Input: Tabl_Name, T;
2: Output: Triples: Map Row ID in table to subjectMap;
3: Var
4: Cols_Template ← θ; //Array to save List of columns to generate Row Id as SubjectMap;
5: NotNulls ← θ; // Array to save List of columns are not null;
6: PKs_Count ← T.Table_ColumnPKs.length;
7: UNQs_Count ← T.Table_ColumnUniques.length;
8: Begin
9: Triples ← θ;
10: if (PKs_Count == 0 && UNQs_Count == 0) then
11:     NotNulls=getNotNullColumns(idx_TBs);
12: end if
13: if (PKs_Count > 0) then
14:     Cols_Template=DBs_Info.Schema_TableInfo[idx_TBs].Table_ColumnPKs;
15: else if (UNQs_Count > 0) then
16:     Cols_Template=DBs_Info.Schema_TableInfo[idx_TBs].Table_ColumnUniques;
17: else if (NotNulls.length > 0) then
18:     Cols_Template=NotNulls;
19:     if (Cols_Template.length > 2) then
20:         Cols_Template=getPartoFColumnsLeftPictureFileds(idx_TBs,Cols_Template,3);
21:     end if
22: else
23:     Cols_Template=ConvertArrayListToStringArray(DBs_Info.Schema_TableInfo[idx_TBs].
          Columns_Name);
24:     if (Cols_Template.length > 2) then
25:         Cols_Template=getPartoFColumnsLeftPictureFileds(idx_TBs,Cols_Template,3);
26:     end if
27: end if
28: Triples ← Triples U "rr : template'" + pfx.NS + Table_Name + "_";
29: for each column Col in Cols_Template do
30:     Triples ← Triples U "{\"" + Col + "\"}";
31: end for
32: Triples ← Triples U "';";
33: Triples ← Triples U "\n";
34: return Triples;
35: End
```

Algorithm 6. GeneratepredicateObjectMap(Table_Name,Column_Name)

```
1: Input: Tabl_Name, Column_Name;
2: Output: Triples :Map Column to Triple of PredicateObjectMap;
3: Begin
4: Triples ← θ;
5: Triples U""rr : predicateObjectMap[" + "\n";
6: Triples ← Triples U "rr : predicate NS : " + Table_Name + "." +
     Column_Name + ";\n";
7: Triples ← Triples U "rr : objectMap[rr : column\"" + Column_Name + "\"]" +
     "\n";
8: Triples ← Triples U "];";
9: Triples ← Triples U "\n";
10: return Triples;
11: End
```

Algorithm 7. GenerateRefObjectMap(FK_Table_Name, FK_Column_Name, Ref_To_TableName, Ref_To_Column_Name)

Input: FK_Table_Name: Name of table that has FK;
FK_Column_Name : Name of column that is FK;
Ref_To_TableName: Name of table referred to it;
Ref_To_Column_Name: Name of column referred to it;
Output: Triples :Map relationships to Triples of RefObjectMap;
Begin
$Triples \leftarrow \theta$;
$Triples \leftarrow Triples\ U\ "rr : predicateObjectMap[" + "\texttt{\textbackslash n}";$
$Triples \leftarrow Triples\ U\ "rr : predicate\ \ NS : " + FK\_Table\_Name + "." + FK\_Column\_Name + ";" + "\texttt{\textbackslash n}";$
$Triples \leftarrow Triples\ U\ "rr : objectMap[" + "\texttt{\textbackslash n}";$
$Triples \leftarrow Triples\ U\ "a\ \ rr : RefObjectMap;" + "\texttt{\textbackslash n}";$
$Triples \leftarrow Triples\ U\ "rr : parentTriplesMap" + CreateMapClass(Ref\_To\_TableName) + ";" + "\texttt{\textbackslash n}";$
$Triples \leftarrow Triples\ U\ "rr : joinCondition[" + "\texttt{\textbackslash n}";$
$Triples \leftarrow Triples\ U\ "rr : child\backslash"" + FK\_Column\_Name + "\";" + "\texttt{\textbackslash n}";$
$Triples \leftarrow Triples\ U\ "rr : parent\backslash"" + Ref\_To\_Column\_Name + "\";]" + "\texttt{\textbackslash n}";$
$Triples \leftarrow Triples\ U\ "];" + "\texttt{\textbackslash n}";$
$Triples \leftarrow Triples\ U\ "];";$
$Triples \leftarrow Triples\ U\ "\texttt{\textbackslash n}";$
return Triples;
End

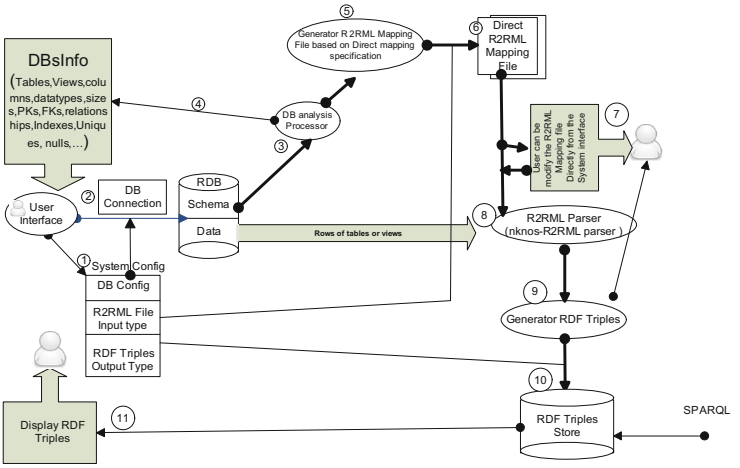

Fig. 2. A general overview of the R2RML mapping generation process in RML-BDM system.

tables, views, columns, data types, sizes, constraints, PKs, FKs, relationships, indexes, unique, and nulls, etc.
- *Generator R2RML Mapping File:* This module is used to automatically generate an R2RML mapping file based on the behavior of the DM specification from the DBsInfo class, according to our approach algorithms. The input of this processor is DBsInfo (contains all information about RDB schema) the output is R2RML mapping file formatted as rdf format (or TTL). The output file can encapsulate all mapping results into a standard input for any R2RML processor later to produce a set of RDF triples that is similar to those resulting from DM.
- *R2RML Parser (RDF processor):* It is used to generate a real RDF triples file from RDB data depending on R2RML mapping file to make it accessible to RDF store. Moreover, this stage is to satisfy our approach for generating R2RML mapping file. We used the open source tool nkons-r2rml-parser (or use any other R2RML processor) which is integrated with our RML-BDM system to generate a set of triples that correspond to the ones generated by DM approach.
- *Screen display (user interface):* The user can specify the configuration setting for execution environments (system config module), display the database information-schema (DBsInfo class), R2RML mapping file, and the resulting RDF triples on the tool screen through the user interface.

5.2 Implementation

The algorithms described in this paper have been implemented in RML-BDM processor prototype and integrated with nkons-r2rml-parser [8]. This prototype has been implemented on Netbeans IDE 7.3.1 (J2SE, JDK 1.7) platform. Thus, the inputs of processor are user-specific configuration system, a SQL connection to the relational database, and a base IRI. Meanwhile, the outputs are produced automatically, including the DBsInfo class, an R2RML mapping document and resulting RDF dataset. These outputs are shown in screen display, and the R2RML mapping document and RDF triples can be saved in an RDF file in different syntax formats (RDF/XML, N-TRIPLES, TURTLE (or TTL), and N3). Moreover, the RDF mapping file can be used with any R2RML parser for converting relational data to RDF triples. Current system prototype supports SQL connections to MySQL Server and already included drivers for major commercial and open source databases, including Postgres, SQL Server and Oracle.

5.3 Experimental Results and Discussion

We carried out RDB metadata and R2RML mapping extraction experiments with our R2RML-BDC tool on a Laptop with configurations as Windows 7 (32-bit), CPU Intel(R) Core i5-2410M 2.30 GHz, RAM 6 GB. A prototype for this experiment is implemented using MYSQL, Java programming language, Netbeans IDE 7.3.1 and Apache Jena tools. Experimental tests on the effectiveness and validity of our RDB2RDF mapping algorithms were conducted with the

Table 1. A list of RDB schema and data sizes

RDB	SizeDB (kb)	SizeDB Schema(kb)	View	Table	Column	FKs	PKS	Rows
rdblab	100035.2	160	0	6	16	5	7	100200
Iswc	20176	256	0	9	46	11	13	90000
Tracker	20000	1056	0	25	162	38	25	95003
Sakila	9132.26	625	7	16	131	22	18	92227
Norhwind	14048	576	16	13	191	13	16	120943

Table 2. A list of results for our approach

RDB	Time of extract metadata + GenerateR2RMLFile(ms)	SizeR2RML (kb)	R2RML Tuples	SizeRDF (mb)	RDF triples
rdblab	295	6.45	181	58.9	752916
Iswc	685	14.8	387	74.7	756094
Tracker	3551	50	1276	103	1083491
Sakila	3697	37.8	956	88	953581
Norhwind	4069	50.1	1141	133	1278842

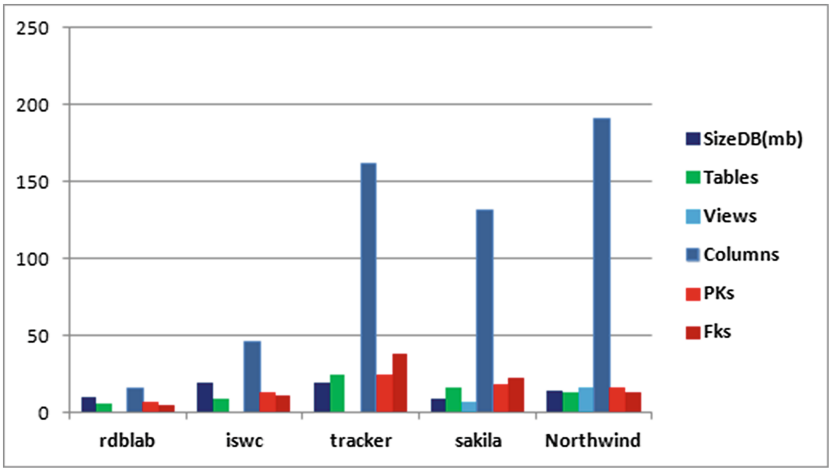

Fig. 3. Important factors of RDBs to build R2RML mapping file. (Color figure online)

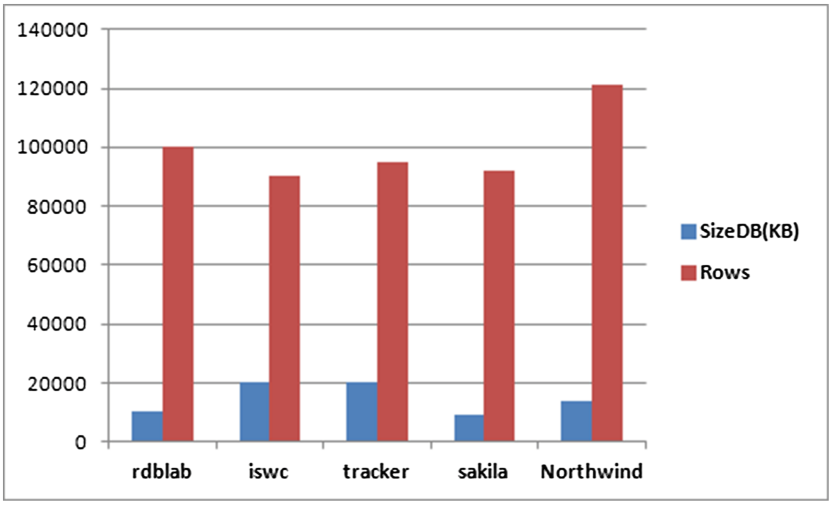

Fig. 4. Dataset sizes in RDBs (schema and data). (Color figure online)

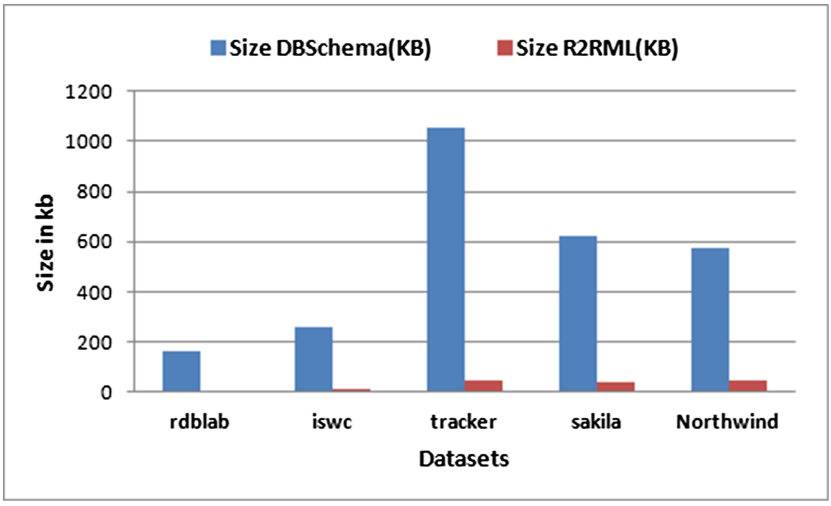

Fig. 5. Schema sizes in RDBs and R2RML mapping files. (Color figure online)

schema sizes of the five RDBs created with MYSQL Server and tested in our experiments. These RDBs are rdblab [15], iswc[6], tracker[7], sakila[8], and Northwind[9], which covering various of important RDB concepts such as tables, views,

[6] http://d2rq.org/example/iswc-mysql.sql.
[7] http://www.artfulsoftware.com/mysqlbook/sampler/mysqled1_appe.html#5-1.
[8] http://mysql-tools.com/en/downloads/mysql-databases/4-sakila-db.html.
[9] https://code.google.com/p/northwindextended/downloads/detail?name=Northwind.MySQL5.sql.

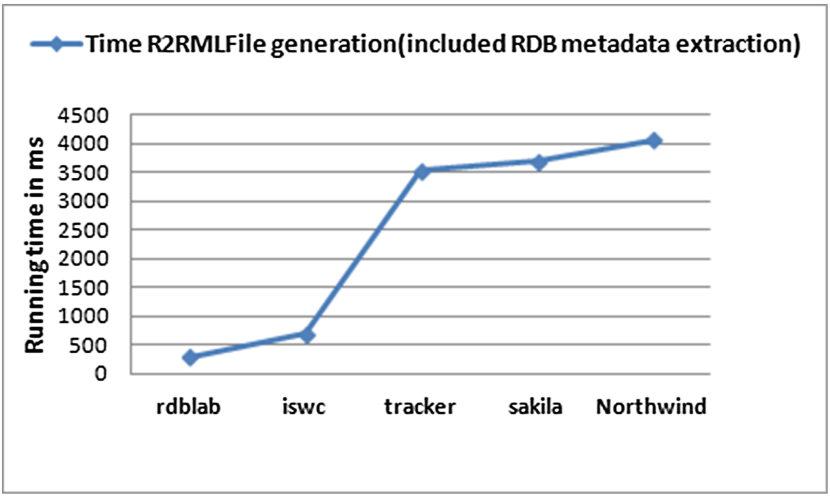

Fig. 6. The running time of algorithmic routines in RML-BDM.

columns, constraints, single or composite primary key, one or many foreign key in one table, and all types of relationships. Moreover, some databases have encoded a parent-child relationship which one table as parent related to many tables as child.

First, the databases information concepts that automatically extracted by our system are shown in Table 1. These concepts are the important factors for building R2RML mapping files and affecting the algorithm performance for extracting RDB metadata, generating R2RML mapping file and converting data of RDB to RDF datasets. Second, the results of R2RML schema tuples and RDF triples are shown in Table 2. The two fields namely SizeDBschema (in Table 1) and SizeR2RML (in Table 2) show the sizes of the RDBs schema compared to R2RML mapping files that were extracted. From these two fields, we can observe that the ontology is the best to store the schema and infer the knowledge from the ontology. Figures 3 and 4 are the analysis bar-chart of the Table 1. Figure 4 shows schema sizes and rows of the RDBs that tested in our experiments. Figure 5 reveals the best case to store the schema in ontology compared to RDB. The X-axis used to show the different domain datasets, whilst Y-axis shows the size of the database in kilobytes (kb). Moreover, the performance analysis of the different databases is shown in the Fig. 6. The execution time of creating R2RML mapping file included the running time for extracting RDB metadata-DBsInfo (RDB have data) and generating R2RML mapping file in algorithm **R2RML Generator**. Therefore, from the tables and figures analyses, we can conclude what are the most important factors for extracting R2RML mapping document from RDB and affecting the time of extraction. Although the size of tracker database greater than the size of sakila and northwind database, but the execution time of **R2RML Generator** algorithm is smaller, because there are other

factors have influenced the execution time an R2RML mapping file generator with the size of the database.

6 Conclusion and Future Work

We introduced an approach tool for the automatically generating an R2RML mapping document from an RDB schema, and any R2RML engine can be use this mapping document to create a set of RDF dataset that following the DM specification. RML-BDM is a tool that enables domain experts and non-experts to automatically create an R2RML mapping files from RDBs by using the R2RML format. RML-BDM has been integrated with nknos-r2rml parser that can export RDB contents as RDF graphs, based on an R2RML mapping document.

The process was tested using five RDBs having different sizes of schema and covering several RDB concepts and types of relationships between tables. In future works, we will add other tools to address an expressing customized mappings from various types of data sources such as XML, NoSQL, and object-oriented to an RDF triples.

Acknowledgments. This work is supported by National Natural Science Foundation of China under grants 61572221, 61173170, 61300222, 61370230, 61433006 and U1401258, Innovation Fund of Huazhong University of Science and Technology under grants 2015TS069 and 2015TS071, Science and Technology Support Program of Hubei Province under grant 2014BCH270 and 2015AAA013, and Science and Technology Program of Guangdong Province under grant 2014B010111007.

References

1. Shadbolt, N., Hall, W., Berners-Lee, T.: The semantic web revisited. IEEE Intell. Syst. **21**(3), 96–101 (2006)
2. Manola, F., Miller, E., McBride, B.: RDF 1.1 Primer. W3C Working Group Note, 24 June 2014 (2014)
3. He, B., Patel, M., Zhang, Z., Chang, K.C.-C.: Accessing the deep web. Commun. ACM **50**(5), 94–101 (2007)
4. Arenas, M., Bertails, A., Prud, E., Sequeda, J.: A direct mapping of relational data to RDF. W3C Recommendation, 27 September 2012. http://www.w3.org/TR/rdb-direct-mapping/
5. Souripriya Das, O., Seema Sundara, O., Cyganiak, R.: R2RML: RDB to RDF mapping language. W3C. http://www.w3.org/TR/r2rml/
6. Hazber, M.A.G., Li, R., Zhang, Y., Xu, G.: An approach for mapping relational database into ontology. In: Proceedings of the 12th Web Information System and Application Conference (WISA 2015), Jinan, Shangdong, China, pp. 120–125 (2015)
7. Priyatna, F., Corcho, O., Sequeda, J.: Formalisation and experiences of R2RML-based SPARQL to SQL query translation using morph. In: Proceedings of the 23rd International Conference on World Wide Web (WWW 2014), New York, USA, pp. 479–490 (2014)

8. Konstantinou, N., Kouis, D., Mitrou, N.: Incremental export of relational database contents into RDF graphs. In: Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS 2014), Thessaloniki, Greece, p. 33 (2014)
9. Michel, F., Montagnat, J., Faron-Zucker, C.: A survey of RDB to RDF translation approaches and tools. Research Report. I3S (2014). https://hal.inria.fr/hal-00903568
10. Villazn-Terrazas, B., Hausenblas, M.: RDB2RDF Implementation Report. W3C Working Group Note, 14 August 2012. http://www.w3.org/TR/rdb2rdf-implementations/
11. Sahoo, S.S., Halb, W., Hellmann, S., Idehen, K., Thibodeau Jr., T., Auer, S., Sequeda, J., Ezzat, A.: A survey of current approaches for mapping of relational databases to rdf. W3C RDB2RDF XG Incubator Report W3C (2009)
12. Mohamed, H., Jincai, Y., Qian, J.: Towards integration rules of mapping from relational databases to semantic web ontology. In: 2010 International Conference on Web Information Systems and Mining (WISM), Sanya, China, pp. 335–339 (2010)
13. Vavliakis, K.N., Grollios, T.K., Mitkas, P.A.: RDOTE publishing relational databases into the semantic web. J. Syst. Softw. **86**(1), 89–99 (2013)
14. Sequeda, J.F., Arenas, M., Miranker, D.P.: On directly mapping relational databases to RDF and OWL. In: Proceedings of the 21st international conference on World Wide Web (WWW 2012), Lyon, France, pp. 649–658 (2012)
15. Hazber, M.A.G., Li, R., Gu, X., Xu, G., Li, Y.: Semantic SPARQL query in a relational database based on ontology construction. In: Proceedings of the 11th International Conference on Semantics, Knowledge and Grids (SKG 2015). Beijing, China (2015)
16. Čerāns, K., Būmans, G.: RDB2OWL: a language and tool for database to ontology mapping. In: Proceedings of the CAiSE 2015 Forum at the 27th International Conference on Advanced Information Systems Engineering (CAiSE 2015), Kista, Sweden, pp. 81–88 (2015)
17. Barrasa, J., Corcho, Ó., Gómez-Pérez, A.: R2O, an extensible and semantically based database-to-ontology mapping language. In: Second Workshop on Semantic Web and Databases(SWDB 2004), pp. 1069–1070 (2004)
18. Auer, S., Dietzold, S., Lehmann, J., Hellmann, S., Aumueller, D.: Triplify: light-weight linked data publication from relational databases. In: : Proceedings of the 18th International Conference on World Wide Web (WWW 2009), pp. 621–630 (2009)
19. Bizer, C., Cyganiak, R.: D2R server-publishing relational databases on the semantic web. In: Presented at the 5th International Semantic Web Conference (ISWC 2006), Athens, GA, USA (2006)
20. OpenLink Virtuoso Universal Server. http://virtuoso.openlinksw.com
21. Sequeda, J.F., Tirmizi, S.H., Corcho, O., Miranker, D.P.: Survey of directly mapping sql databases to the semantic web. Knowl. Eng. Rev. **26**(4), 445–486 (2011)
22. de Medeiros, L.F., Priyatna, F., Corcho, O.: MIRROR: automatic R2RML mapping generation from relational databases. In: Cimiano, P., Frasincar, F., Houben, G.-J., Schwabe, D. (eds.) ICWE 2015. LNCS, vol. 9114, pp. 326–343. Springer, Heidelberg (2015)
23. Dimou, A., Vander Sande, M., Colpaert, P., Verborgh, R., Mannens, E., Van de Walle, R.: RML: a generic language for integrated RDF mappings of heterogeneous data. In: Proceedings of the 7th Workshop on Linked Data on the Web (LDOW 2014), Seoul, Korea (2014)

24. Neto, L.E.T., Vidal, V.M.P., Casanova, M.A., Monteiro, J.M.: *R2RML by assertion*: a semi-automatic tool for generating customised R2RML mappings. In: Cimiano, P., Fernández, M., Lopez, V., Schlobach, S., Völker, J. (eds.) ESWC 2013. LNCS, vol. 7955, pp. 248–252. Springer, Heidelberg (2013)
25. Pequeno, V.M., Vidal, V.M., Casanova, M.A., Neto, L.E.T., Galhardas, H.: Specifying complex correspondences between relational schemas and RDF models for generating customized R2RML mappings. In: Proceedings of the 18th International Database Engineering & Applications Symposium, Porto, Portugal, pp. 96–104 (2014)
26. Prud'hommeaux, E., Carothers, G., Beckett, D., Berners-Lee, T.: RDF 1.1 Turtle: terse RDF triple language. World Wide Web Consortium. http://www.w3.org/TR/turtle/. Accessed 24 Dec 2014