Information Technology and Management Science

doi: 10.1515/itms-2014-0006

2014 / 17

Building Ontology from Relational Database

Andrejs Kaulins[1], Arkady Borisov [2], [1, 2] *Riga Technical University*

***Abstract* – This paper proposes an approach of building OWL2 ontology from data in a relational database. Compared with existing methods, the approach can acquire ontology from a relational database automatically by using a group of learning rules. Approach is independent from database implementation; it is designed based on standards, used to manage database systems. In this paper, we describe object mapping from a relational database and OWL2 ontology including classes, data properties, object properties, instances, axioms and annotations. Axioms are important part of OWL2 specification, and the approach is also suitable for the axiom building process.**

***Keywords* – Ontology, ontology building, relational database relational model.**

I. Introduction

Many approaches are developed to extract knowledge from existing relational databases. Tools are implemented using OWL language. However ontology models are not complete, still a lot of things are out of scope. This paper aims to solve the problem of building OWL ontology using brand new notation for describing ontologies. This notation OWL2 is recommended by W3C consortium, and it is extension of OWL [9]. It includes well-defined semantics for ontology reusability, reasoning and contains many profiles, thus enabling one to use OWL2 for describing knowledge in specific domains.

Relational database consists of data, data implemented as some relational model. Also operations and constraints are developed in the database to implement business rules. It is not an easy task to extract all these things and build ontology on them.

We provide a new approach, based on OWL2 language, and relation database is not normalized; this approach works with the database “as it is”.

The paper is organized as follows. Section I analyzes current related studies; Section II gives an overview of standards used in relational database management systems. Section III gives some preliminary definitions used in the mapping algorithm between objects in the relational database and OWL2 ontology. In Section IV we describe relational database sample schema. Section V defines approach learning rules, and section VI draws conclusions and gives an insight into the future research.

II. Related Work

Much attention has been devoted to the issue of building an ontology from a relational database. Ontology is used to support the sharing and reuse of formally represented knowledge among AI systems [2]. To solve the problem, researchers use different approaches. One of them is using concept hierarchy [4]. Concept hierarchy is a type of background knowledge, which expresses the structure of concept from a low-level concept to a more general concept. Advantages of this approach – it uses information entropy as a metric for the correlation between any two features. As a result, this approach allows extracting knowledge selectively and merging it to the existing ontology. Weakness of this approach: it does not allow building axioms used in reasoning. Ontology is built using OWL recommendation without formally defined meaning. For defining reasoning predicates some semantic web rule language should be used. Another approach [5], [10], [11] can acquire ontology from relational database automatically by using a group of learning rules. It is also based on OWL recommendation. The method [6] designed to build a conceptual model from a relational model. It uses data analysis in tables to build generalization abstraction. These conceptual models can be used in ontology building. This approach also lacks a reasoning process, and could not be used in decision support systems. Our proposed approach is aimed at building an ontology model with strong defined semantics and knowledge extracted from a relational database. OWL2 new recommendation owns these features. It could use other ontologies as well. To build ontology, we used learning rules, even comments and remarks in the source database were mapped in ontology as annotations and axioms.

III. Database Management Standards

There are more than 50 implementations of relational database management systems. Most popular is DB/2 from IBM, SQL server from Microsoft, Oracle Database from Oracle, MySQL and Sybase. To enable one to share information between databases, the structured query language (SQL) is used. To enable the portability of SQL applications across conforming implementations, standards are used. The latest standard is SQL99 (the so-called SQL/3) [8], SQL/4 is now undergoing the development phase.

IV. Preliminary Definitions

The underlying model of relational database is the relational model [1], in which each relation R_1 corresponds to a table. The columns of a relation are called attributes denoted as A. Each attribute has type and range of values of this type. Let us define some predicates, which acquire the table primary key, table foreign key, attribute name, type and range of values:

1. pkey(R_1) – returns the name of primary key; the key can be composed of one or many table columns;
2. |pkey(R_1)|– a number of attributes participated in key composition;
3. pkeyAttr(pkey(R_1)) – returns attributes contained in primary key, when |pkey(R_1)| > 1;

4. fkey(R_1) – returns the name of foreign key; key values are values from the primary key in another table. Through a foreign key a referential constraint is implemented in a relational model;
5. fkeyAttr(fkey(R_1)) – returns attributes contained in the foreign key, when |fkey(R_1)| > 1;
6. attr(R_1) – returns attributes contained in a specific relation R_1, output is an array of all attributes, contained in table A_{11}, …. , A_{1n}, where n – is an attribute count;
7. attrType(R_1 , A_1) – returns the attribute type of attribute A_1, contained in table R_1 ;
8. attrDom(R_1 , A_1) – returns a value range of values of attribute A_1, contained in table R_1 ;
9. attrName(R_1 , A_1) – returns the name of attribute A_1 contained in table R_1 ;
10. tuple(R_1 ,T_1) – returns values of record 1 (tuple 1) of table R_1 . Number of rows in table R_1 can change, but for reference let us assume m is the current maximum number of tuples;
11. attrValue(R_1 , A_1 , T_1) – returns the value of attribute A_1 in table R_1 of tuple 1 - T_1;
12. tableName(R_1) – returns the name of table R_1 .

Relational database contains not only tables, but also constraints and triggers, which help encapsulate business rules into the database [3]. Constraints can define ranges on attribute values, limit value duplication in attributes (unique constraint), or define reference to other tables (referential constraint).

Relationships between relations are important. Two relations can have one-to-one, one-to-many or many-to-many relationship types. The data duplication degree in a database depends on the database normalization type [12]. In practice, it is not always possible to normalize a data model. These factors are taken into account designing the proposed approach.

V. Relational Database Schema

Let us consider a relational database schema example to demonstrate mapping for tables, constraints and data to ontology (Fig. 1). This sample schema is taken from oracle predefined examples and could be installed with oracle database installation [7].

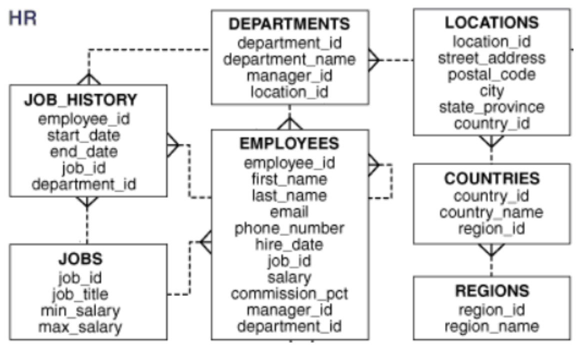


Fig. 1. A relational database schema example.

VI. Proposed Approach and Learning Rules

An approach for building ontology in OWL2 consists of 6 major steps:

Step I. Relation (table) analysis. At this step using predicates defined in Section IV, we find out the name of tables, names of all columns, comments on table and columns. Using a database dictionary we find out all constraints, defined in the table, including a primary key, foreign key(s). Table I provides details of database schema structure and table constraints.

Based on this information, we could build OWL2 class as a knowledge concept in ontology.

Rules used at this step:

<u>Rule 1</u>: For relations R_1 , R_2 ,…, R_k in database. Let us suppose that P_1 = pkey(R_1), P_2 = pkey(R_2),…, P_k = pkey(R_k). If between relations R_1 , R_2 ,…, R_k there is equivalence, i.e., $R_1(P_1) = R_2(P_2) = \ldots = R_k(P_k)$, then the information spread across R_1 , R_2 ,…, R_k should be integrated into an ontological class.

Rule 2: An ontological class can be created based on relation R_i, if the following conditions can be satisfied:

|pkey(R_i)| = 1 or |pkey(R_i)| > 1 and there is A_i, where $A_i \in$ pkeyAttr(pkey(R_i))

and $A_i \notin$ fkeyAttr(fkey(R_i)).

In ontology OWL2, a class entity for table HR.EMPLOYEES can be declared as:

Ontology(<http://www.my.example.com/example>
Declaration(Class(a:Employees))
Declaration(Class(a:Departments))
Declaration(Class(a:Jobs))
Declaration(Class(a:Locations))
Declaration(Class(a:Regions))
Declaration(Class(a:Countries))
Declaration(Class(a:Job_History))

Step II. In the selected table we analyze columns and extract all constraints with predicates attr(R_i),attrType(R_i , A_j), attrDom(R_i , A_j), attrName(R_i , A_j), pkey(R_i), pkeyAttr(pkey(R_i)), fkey(R_i),fkeyAttr(fkey(R_i))

Rules used at this step:

<u>Rule 3:</u> For relations R_i and R_j, if there is a dependency relationship, i.e., $R_i (A_i) \subseteq R_j (A_j)$ and $A_i \notin$ pkeyAttr(pkey(R_i)) are satisfied, then object property P_i can be created on A_i. On ontology OWL2 these can be declared as follows (for relation "HR"."EMPLOYEES"):

ObjectPropertyAssertion(*a:worksInDepartment a:Employees a:Departments*)

ObjectPropertyAssertion(*a:haveJobt a:Employees a:Jobs*)

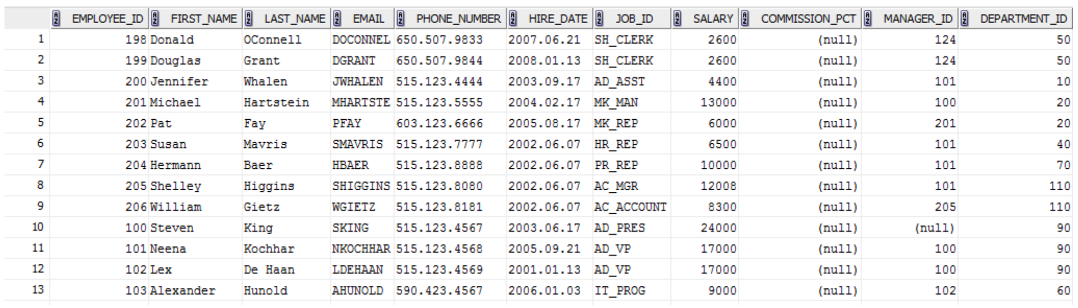
ObjectPropertyAssertion(*a:haveManager a:Employees a:Employees*)

The third object property describes a relation of table to itself, i. e., managers are also employees, but each employee has a manager.

TABLE I

RELATIONAL DATABASE SCHEMA DETAILS

No.	**Relation**	**Constraints**	**Primary Key**	**Foreign Key**
1	"HR"."EMPLOYEES" ("EMPLOYEE_ID" NUMBER(6,0), "FIRST_NAME" VARCHAR2(20 BYTE), "LAST_NAME" VARCHAR2(25 BYTE), "EMAIL" VARCHAR2(25 BYTE), "PHONE_NUMBER" VARCHAR2(20 BYTE), "HIRE_DATE" DATE, "JOB_ID" VARCHAR2(10 BYTE), "SALARY" NUMBER(8,2), "COMMISSION_PCT" NUMBER(2,2), "MANAGER_ID" NUMBER(6,0), "DEPARTMENT_ID" NUMBER(4,0))	1. "LAST_NAME" IS NOT NULL 2. "EMAIL" IS NOT NULL 3. "HIRE_DATE" IS NOT NULL 4. "JOB_ID" IS NOT NULL 5.CONSTRAINT "EMP_SALARY_MIN" CHECK (salary > 0) 6.CONSTRAINT "EMP_EMAIL_UK" UNIQUE ("EMAIL")	"EMPLOYEE_ID"	1. CONSTRAINT "EMP_DEPT_FK" FOREIGN KEY ("DEPARTMENT_ID") REFERENCES "HR"."DEPARTMENTS" ("DEPARTMENT_ID") 2. CONSTRAINT "EMP_JOB_FK" FOREIGN KEY ("JOB_ID") REFERENCES HR"."JOBS" ("JOB_ID") 3. CONSTRAINT "EMP_MANAGER_FK" FOREIGN KEY ("MANAGER_ID") REFERENCES HR"."EMPLOYEES" ("EMPLOYEE_ID")
2	"HR"."DEPARTMENTS"("DEPARTMENT_ID" NUMBER(4,0), "DEPARTMENT_NAME" VARCHAR2(30 BYTE), "MANAGER_ID" NUMBER(6,0), "LOCATION_ID" NUMBER(4,0))	"DEPARTMENT_NAME" IS NOT NULL	"DEPARTMENT_ID"	1."DEPT_LOC_FK" FOREIGN KEY ("LOCATION_ID") REFERENCES "HR"."LOCATIONS" ("LOCATION_ID") 2."DEPT_MGR_FK" FOREIGN KEY ("MANAGER_ID") REFERENCES "HR"."EMPLOYEES" ("EMPLOYEE_ID")
3	"HR"."JOBS"("JOB_ID" VARCHAR2(10 BYTE), "JOB_TITLE" VARCHAR2(35 BYTE), "MIN_SALARY" NUMBER(6,0), "MAX_SALARY" NUMBER(6,0))	"JOB_TITLE" IS NOT NULL	"JOB_ID"	-
4	"HR"."LOCATIONS"("LOCATION_ID" NUMBER(4,0), "STREET_ADDRESS" VARCHAR2(40 BYTE), "POSTAL_CODE" VARCHAR2(12 BYTE), "CITY" VARCHAR2(30 BYTE), "STATE_PROVINCE" VARCHAR2(25 BYTE), "COUNTRY_ID" CHAR(2 BYTE))	"CITY" IS NOT NULL	"LOCATION_ID"	CONSTRAINT "LOC_C_ID_FK" FOREIGN KEY ("COUNTRY_ID") REFERENCES "HR"."COUNTRIES" ("COUNTRY_ID")
5	"HR"."REGIONS"("REGION_ID" NUMBER "REGION_NAME" VARCHAR2(25 BYTE))	"REGION_ID" IS NOT NULL	"REGION_ID"	-
6	"HR"."COUNTRIES"("COUNTRY_ID" CHAR(2 BYTE), "COUNTRY_NAME" VARCHAR2(40 BYTE), "REGION_ID" NUMBER)	"COUNTRY_ID" IS NOT NULL	"COUNTRY_ID"	CONSTRAINT "COUNTR_REG_FK" FOREIGN KEY ("REGION_ID") REFERENCES "HR"."REGIONS" ("REGION_ID")
7	"HR"."JOB_HISTORY"("EMPLOYEE_ID" NUMBER(6,0), "START_DATE" DATE, "END_DATE" DATE, "JOB_ID" VARCHAR2(10 BYTE), "DEPARTMENT_ID" NUMBER(4,0))	1. "EMPLOYEE_ID" IS NOT NULL 2. "START_DATE" IS NOT NULL 3. "END_DATE" IS NOT NULL 4. "JOB_ID" IS NOT NULL	PRIMARY KEY ("EMPLOYEE_ID", "START_DATE")	1. CONSTRAINT "JHIST_JOB_FK" FOREIGN KEY ("JOB_ID") REFERENCES "HR"."JOBS" ("JOB_ID") 2. CONSTRAINT "JHIST_EMP_FK" FOREIGN KEY ("EMPLOYEE_ID") REFERENCES "HR"."EMPLOYEES" ("EMPLOYEE_ID") 3. CONSTRAINT "JHIST_DEPT_FK" FOREIGN KEY ("DEPARTMENT_ID") REFERENCES "HR"."DEPARTMENTS" ("DEPARTMENT_ID")



	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
1	198	Donald	OConnell	DOCONNEL	650.507.9833	2007.06.21	SH_CLERK	2600	(null)	124	50
2	199	Douglas	Grant	DGRANT	650.507.9844	2008.01.13	SH_CLERK	2600	(null)	124	50
3	200	Jennifer	Whalen	JWHALEN	515.123.4444	2003.09.17	AD_ASST	4400	(null)	101	10
4	201	Michael	Hartstein	MHARTSTE	515.123.5555	2004.02.17	MK_MAN	13000	(null)	100	20
5	202	Pat	Fay	PFAY	603.123.6666	2005.08.17	MK_REP	6000	(null)	201	20
6	203	Susan	Mavris	SMAVRIS	515.123.7777	2002.06.07	HR_REP	6500	(null)	101	40
7	204	Hermann	Baer	HBAER	515.123.8888	2002.06.07	PR_REP	10000	(null)	101	70
8	205	Shelley	Higgins	SHIGGINS	515.123.8080	2002.06.07	AC_MGR	12008	(null)	101	110
9	206	William	Gietz	WGIETZ	515.123.8181	2002.06.07	AC_ACCOUNT	8300	(null)	205	110
10	100	Steven	King	SKING	515.123.4567	2003.06.17	AD_PRES	24000	(null)	(null)	90
11	101	Neena	Kochhar	NKOCHHAR	515.123.4568	2005.09.21	AD_VP	17000	(null)	100	90
12	102	Lex	De Haan	LDEHAAN	515.123.4569	2001.01.13	AD_VP	17000	(null)	100	90
13	103	Alexander	Hunold	AHUNOLD	590.423.4567	2006.01.03	IT_PROG	9000	(null)	102	60

Fig. 2. Table "HR"."EMPLOYEES" data example.

Rule 4: For relations R_i and R_j, two ontological object properties “has-part” and “is-part-of” can be created, if the two conditions are satisfied:

|pkey(R_i)| > 1 and

fkeyAttr(fkey(R_j)) $\subset$ pkeyAttr(pkey(R_i))

Rule 5: For relations R_i , R_j and R_k ,if A_i = pkeyAttr(pkey(R_i)), A_j = pkeyAttr(pkey(R_j)), $A_i \cup A_j$ = fkeyAttr(fkey(R_k)) and $A_i \cap A_j = \emptyset$, then two object properties P_i , P_j can be created based on the semantics of R_k

Rule 6: For relations R_1 , R_2 ,…, R_i and R_k in the database, if A_1 = pkeyAttr(pkey(R_1)), A_2 = pkeyAttr(pkey(R_2)), … A_i = pkeyAttr(pkey(R_i)), $A_1 \cup A_2 \cup \ldots \cup A_i$ = fkeyAttr(fkey(R_k)) and $A_1 \cap A_2 \cap \ldots \cap A_i = \emptyset$, then object properties P_1 , P_2 , …, P_i can be created. Rule 6 extends Rule 5 for n-ary relationship provided by R_k .

This rule introduces mapping datatype properties (basic oracle database and OWL2 datatypes are presented in Table II) to ontology, presented in OWL2:

Rule 7: For an ontological class C_i and datatype properties set of C_i denoted as DP(C_i), if C_i is corresponding to relations $R_1, R_2, \ldots, R_i$ in database, then for every attribute in $R_1, R_2, \ldots, R_i$, if it cannot be used to create an object property by using Rule 3, then it can be used to create datatype property of C_i . The domain and range of each property are extracted using predicates for relational database attrDom(R_i, A_j) and attrValue(R_i, A_j, T_k).

For table "HR"."EMPLOYEES" we could define datatypes as follows:

DataPropertyRange(a:hasFirstName xsd:string)
DataPropertyRange(a:hasLastName xsd:string)
DataPropertyRange(a:hasEmail xsd:string)
DataPropertyRange(a:hasPhoneNumber xsd:string)
DataPropertyRange(a:hasHireDate xsd:dateTime)
DataPropertyRange(a:hasSalary xsd:decimal)
DataPropertyRange(a:hasCommision% xsd:decimal)

Rule 8: For relations R_i and R_j , supposed that P_i = pkeyAttr(pkey(R_i)) , P_j = pkeyAttr(pkey(R_j)), if P_i (R_i) $\subseteq$ P_j (R_j) is satisfied, then class/property corresponding to R_i is subclass/subproperty of the class/property corresponding to R_j .

Table "HR"."EMPLOYEES" contains employees and also managers, so in the ontology this fact could be expressed as follows:

Ontology(http://www.my.example.com/example
SubClassOf(a:Managers a:Employees))

Rules for learning cardinality

Rule 9: For relation R_i and $A_i \in$ attr(R_i), if A_i = pkeyAttr(pkey(R_i)) or A_i = fkeyAttr(pkey(R_i)), then the minCardinality and maxCardinality of the property P_i corresponding to A_i is 1.

In our example, at each location there may be only one warehouse, in the ontology model this can be expressed as:

ObjectExactCardinality(1 *a:hasWarehouse a:Location*)

Rule 10: For relation R_i , and $A_i \in$ attr(R_i), if A_i is declared as IS NOT NULL, the maxCardinality of the property P_i corresponding to A_i is 1.

Rule 11: For relation R_i , and $A_i \in$ attr(R_i), if A_i is declared as UNIQUE, the maxCardinality of the property P_i corresponding to A_i is 1

TABLE II
BASIC ORACLE DATABASE DATATYPE MAPPING TO OWL2 ONTOLOGY

Data type	RDBMS data types	OWL2
number	number	xsd:int
number	number(precision, scale)	xsd:decimal
number	binary_float	xsd:float
number	binary_double	xsd:double
text	char, nchar, varchar, varchar2, nvarchar2, long, clob, nclob	xsd:string
date time	date	xsd:dateTime
date time	timestamp, timestamp with time zone	xsd:time
binary	blob, dfile	xsd:base64Binary

Step III. The third step is the extraction of rows (tuples) from the relational database and ontology addition with individuals.

Rule 12: Every tuple T_i in the relational database corresponds to an individual of class in the ontology model.

In table "HR"."EMPLOYEES", each record can be represented as an individual of class a:Employees:

DataPropertyAssertion(a:hasFirstName a:Employees "Douglas" ^^ xsd:string *)*
DataPropertyAssertion(a: a:hasLastName *a:Employees "Grant"* ^^ xsd:string *)*
DataPropertyAssertion(a: hasEmail *a:Employees "DGRA"* ^^ xsd:string *)*
DataPropertyAssertion(a: hasPhoneNumber *a:Employees "6505079844"*^^ xsd:int *)*
DataPropertyAssertion(a: hasHireDate *a:Employees "2008.01.13"*^^ xsd:dateTime*)*
DataPropertyAssertion(a: hasSalary *a:Employees "2600"* ^^ xsd:int *)*
DataPropertyAssertion(a: hasCommision% *a:Employees "2"* ^^ xsd:int *)*

Mappings between objects in the oracle database and OWL2 ontology is provided in Table III.

Step IV. Next we analyze table values and refine some relationship between OWL entities. That can be done using predicate attrValue(R_i, A_j, T_k).

Step V. Finally we can add to ontology some information from the relational database, which will not participate in a reasoning process, i.e., commentaries, remarks, etc.:

AnnotationAssertion (rdfs:comment a:hasFirstName "First name of the employee. A not null column.")

Step VI. Next step is ontology validation, for validation process our approach uses domain specific rules to build axioms and derive new facts in domain.

TABLE III

RDBMS OBJECT AND OWL2 ENTITY MAPPING

No.	RDBMS object	OWL2 model object
1	Table	OWL2 Class
2	Table Column	Functional property
3	Table Row	OWL2 individual
4	Column metadata: Data Type Not Null Null	OWL2 restriction: All values from restriction Cardinality() restriction maxCardinality() restriction
5	Constraint: Not Null Unique Foreign Key Check	OWL2 property: ObjectExactCardinality(1 *OPE* CE) InverseFunctionalProperty objectProperty hasValue

After all these steps are done, we can continue knowledge extraction from the relational database, switching to a next table or can stop the building process.

VII. CONCLUSION AND FUTURE RESEARCH

In this paper, we proposed a new approach for building ontology, using new OWL2 language for describing ontologies. Approach uses mapping rules to define different entities in ontology – classes, functional properties, object data properties, class cardinality, restrictions, individuals and annotations. To demonstrate OWL2 semantics and language notation, we used a plain example, implemented in the oracle database. Approach is not restricted to specific database implementation, because it is based on international standards, used to manage data – SQL99.

In future, we plan to implement this approach on Java and to build a system prototype for an expert advising system in health treatment domain.

REFERENCES

[1] E. F. Codd, “A Relational Model of Data for Large Shared Data Banks”, *Communications of ACM*, vol. 13, no. 6, June 1970, pp. 377–387.

[2] T. R. Gruber, “A translation approach to portable ontology specifications”, technical report KSL 92–71, 1993.

[3] C. J. Date, *An introduction to Database Systems*, 8th ed., Addison-Wesley, 2004, p. 1024.

[4] H. A. Santoso, S. C. Haw, Z. N. Abdul-Mehdi, “Ontology extraction from relational database: Concept hierarchy as background knowledge,” *Knowledge-Based Systems*, vol. 24, issue 3, Apr. 2011, pp. 457–464, 2010. http://dx.doi.org/10.1016/j.knosys.2010.11.003

[5] M. Li, X.Y. Du, S. Wang, “Learning ontology from relational database”, *4th International Conference on Machine Learning and Cybernetics,* 2005, pp. 3410–3415.

[6] N. Lammari, I. Comyn-Wattiau, J. Akoka, “Extracting generalization hierarhies from relational database: A reverse engineering approach”, *Data & Knowledge Engineering* vol. 63, issue 2, 2007, pp. 568–589. http://dx.doi.org/10.1016/j.datak.2007.04.002

[7] R. Leyderman, *Oracle Database Sample Schemas*, 11g Release 2 (11.2), E10831-02, Oracle, 2010.

[8] ISO/IEC SQL/Framework Standard, ISO/IEC 9075-1:1999, ISO, 1999.

[9] B. Motik, P. F. Patel-Schneider, B. Parsia, OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax. Available: http://www.w3.org/TR/owl2-syntax/, [Accessed Oct. 12, 2014].

[10] M. Reda Chbihi Louhdi, H. Behja and S. Ouatik El Alaoui, “Transformation rules for building OWL Ontologies from relational databases”, *Second International Conference on Advanced Information Technologies and Applications,* ICAITA-2013, 2013, pp. 271–283.

[11] I. Astrova , N. Korda and A. Kalja, “Rule-Based Transformation of SQL Relational Databases to OWL Ontologies”, *Proceedings of the 2nd International Conference on Metadata & Semantics Research*, 2007, pp. 415–424.

[12] H. V. Jagadish, T. Nadeau, S. S. Lightstone, T. J. Teorey, *Database Modeling and Design,* 5th ed., Morgan Kaufmann, 2011.

Andrejs Kaulins received the B. Sc. and M. Sc. degrees in 1993 and 2002 from Riga Technical University and the University of Latvia, respectively. Since 2013 he has been studying at Riga Technical University to obtain a Doctoral Degree in Computer Science. Currently major field of study is complex IT system design based on ontologies.
E-mail: andrejs.kaulins@rtu.lv ; phone +37126737122.

Arkady Borisov received the Doctoral degree in Technical Cybernetics from Riga Polytechnic Institute in 1970, and *Dr. habil. sc. comp*. degree from Technical Cybernetics from Taganrog State Radio Engineering University in 1986.

He is a Professor of Computer Science with the Faculty of Computer Science and Information Technology, Riga Technical University. His research interests include fuzzy sets, fuzzy logic, computational intelligence and bioinformatics. He has more than 235 publications in the field.

He is a member of IFSA European Fuzzy System Working Group, Russian Fuzzy System and Soft Computing Association, member of the Scientific Advisory Board of the Fuzzy Initiative Nordrhein.
E-mail: arkadijs.borisovs@cs.rtu.lv.

Andrejs Kauliņš, Arkādijs Borisovs. Ontoloģiju izveide no relāciju datubāzes
Ontoloģijas izveides process prasa daudz laika un dažādu speciālistu iesaistīšanos. Relāciju datubāzes satur informāciju, ko var izmantot, lai veidotu ontoloģijas. Šajā rakstā ir aprakstīta metode, kas ļauj automātiski uzbūvēt ontoloģiju. Būvēšanas procesā no relāciju datu bāzes izmanto kartēšanas noteikumus. Šie noteikumi nosaka, kādā veidā objekti no relāciju datu bāzes ir sasaistīti ar ontoloģijas elementiem. Metodei nav ierobežojumu relāciju datubāzei, jo tā ir balstīta uz pieņemtiem datu glabāšanas standartiem SQL99. Lai aprakstītu ontoloģiju, tika izmantota OWL2 valoda. Kartēšanas noteikumi ļauj veidot klases un to īpašības, datu īpašības un eksemplārus ontoloģijā. Kartēšanas noteikumu izveidošana ir sarežģīts uzdevums. Tomēr tas ir atrisināts vienkāršiem datu modeļiem. Reālos dzīves gadījumos, relāciju datu bāzē tiek izmantotas sarežģītas datu struktūras. Tās ietver patvaļīgus objektus, kurus definē pats lietotājs. Rakstā ir apskatīti kartēšanas noteikumi, kuri darbojas ar dažādu relāciju datu bāžu datu modeļiem. Aprakstīta atbalsta sistēma, kura nodrošina ontoloģijas konstruēšanu ar kartēšanas noteikumu palīdzību. Galvenais šīs sistēmas mērķis ir atklāt saikni starp objektiem relāciju datu bāzē un ontoloģijas elementiem. Nākotnē ir plānots realizēt aprakstīto metodi, izmantojot JAVA platformu un uzbūvējot sistēmu, balstītu uz ontoloģiju veselības aprūpes jomā.

Андрей Каулиньш, Аркадий Борисов. Построение онтологии по реляционной базе данных
Процесс построения онтологии требует много времени и участия широкого круга специалистов. Реляционные базы данных содержат информацию, которая может быть использована для построения онтологий. В данной работе рассмотрен метод построения онтологии из реляционной базы данных, использующий правила отображения. Метод позволяет построить начальную версию онтологии в автоматическом режиме. Описанный метод не имеет ограничений к реляционной базе данных, поскольку использует стандарты описания данных SQL99. Правила отображения определяют соответствия между объектами реляционной базы данных и элементами онтологии. Для описания онтологии использован язык OWL2. Правила соответствия позволяют строить классы, свойства классов и данных, и экземпляры онтологии. Построение правил отображения является трудной задачей, которая решена для простой модели базы данных. Во многих случаях реляционные базы данных используются для хранения сложных конструкций данных. К таким относятся произвольные объекты, определённые пользователем, коллекции данных. В статье систематизированы правила отображения для различных моделей реляционных баз данных. Рассмотрена система поддержки отображений между реляционными базами данных и онтологиями. Основной задачей такой системы является выявление объектов в реляционной базе данных и построение связей, позволяющих построить онтологию. В будущем планируется реализовать описанный метод на JAVA платформе и использовать систему в области здравоохранения.