Ontological Representation of Relational Databases

Camila Zacché de Aguiar, Ricardo de Almeida Falbo, Vítor E. Silva Souza

Ontology & Conceptual Modeling Research Group (NEMO)
Federal University of Espírito Santo, Brazil
Av. Fernando Ferrari, 514 – Goiabeiras – Vitória, ES – 29075-910

camila.zacche.aguiar@gmail.com, {falbo, vitorsouza}@inf.ufes.br

Abstract. *Relational database systems is a well-known domain and an essential component of life in modern society. However, a well-founded ontological conceptualization of this domain is not yet defined and shared by the community, nor applied as a solution to problems such as semantic interoperability, database migration, etc. In this paper, we present RDBS-O, a well-founded reference ontology on the relational database domain, rigorously constructed, verified and validated according to an ontology engineering method. In addition to a reference ontology, we also implement and test its operational version, using it to automatic instantiate the ontology from a real database sample.*

1. Introduction

Recently, there has been growing interest in ontologies in the sense of computational artifacts, i.e., explicit and formal specifications of shared conceptualizations [Studer et al. 1998], as solutions for integration, interoperability, migration and publishing of relational databases. In this context, an ontology that clearly represents the database structure becomes essential to develop an interoperability solution, such as semantic interoperability between published data in the Semantic Web [de Laborda and Conrad 2005], semantic interoperability between relational database systems (RDBMSs) [Trinh et al. 2007], or semantic mapping among database ontologies for RDBMS interoperability [Guido and Paiano 2010].

Here, we refer to the database structure (i.e., its tables, columns, rows, constraints, etc.), not to any specific domain (e.g., government, life sciences, etc.) whose data happens to be stored in relational databases. In this sense, a database migration task, for instance, would consist on migrating any database schema, regardless of the data it contains, from one RDBMS to another (e.g., from Oracle to MySQL). Although most RDBMSs support the SQL standard, they also have their own proprietary SQL extensions and their own way of representing schemas internally. Freely available tools to perform this type of migration are usually tailored for a few specific RDBMSs and are not extensible. An ontology-based solution would allow us to add support to any RDBMS by mapping its particular concepts to those of the ontology.

In this paper, we present the Relational Database System Ontology (RDBS-O), a reference ontology for the relational database domain that represents the structure of a database model. Again, it is important to say that the domain of the information described in the data is not part of the scope of RDBS-O. The reference ontology is based on UFO [Guizzardi and Wagner 2004] and was developed according to the SABiO

method [Falbo 2014], in a modular way to foster its reuse. Validation and verification were performed in the reference ontology, which had its operational version, called RDBS-OWL, implemented and tested. Although such an ontology is useful in itself (as illustrated in the database migration scenario above), RDBS-O is being built in the context of a larger effort of building a network of ontologies on software development frameworks, in particular for object/relational frameworks [Bauer and King 2004]. Such ontologies will allow us to automate tasks such as migrating code from one framework to another or defining and detecting architectural smells in software projects.

The remainder of this paper is organized as follows. Section 2 discusses the ontological foundations used for developing RDBS-O. Section 3 presents RDBS-O. Section 4 addresses ontology verification, validation and testing. Section 5 discusses related works. Finally, Section 6 concludes the paper.

2. Baseline

Database systems (DBSs) are an essential component of life in modern society, since many of our activities involve some computer program accessing a database (e.g., buying something in a store, using a bank account, etc.). A database is a collection of logically related data that has some meaning, accessed through a set of programs that constitute a database management system (DBMS). A DBMS is a general-purpose software system that facilitates the processes of definition, specifying the data types, structures, and constraints of the data to be stored; manipulation, executing queries to retrieve and change stored data; and sharing, allowing simultaneous access to databases [Elmasri and Navathe 2011].

This representation provides data independence through data abstraction such that changes in the physical level are not propagated to the conceptual level and vice-versa. Futhermore, DBSs allows data to be perceived at different levels of detail from data models. Such a model describes the structure of the database as data types, relationships, and constraints that apply to the data [Elmasri and Navathe 2011].

The most widely used representational data model in commercial DBMSs is the relational model, the basis of relational database technology [Date 2004]. The relational model uses the concept of mathematical relation as basic structure and has its theoretical basis in set theory and first-order predicate logic [Elmasri and Navathe 2011]. In the model, relations are used to represent both data and the relationships among data, tuples represents facts that typically correspond to real-world entities or relationships, and attributes specify how to interpret the data values in each tuple adopting a unique type. In a relational database, relations are perceived as tables, tuples as rows, and attributes as columns. The collection of data stored in the database at a particular moment is called an instance [Elmasri and Navathe 2011].

As discussed in the previous section, in this work we propose a reference ontology on relational database systems. Our proposed ontology was built using the SABiO method [Falbo 2014], representing its reference version using OntoUML [Guizzardi 2005], and its operational version in OWL. We anchored our ontology in more general concepts reused from a Software Engineering Ontology Network [Ruy et al. 2016] which, in its turn, is based on the foundational ontology UFO [Guizzardi and Wagner 2004]. The use of a foundational ontology mainly re-

Table 1. OntoUML stereotypes and their ontological distinctions.

Stereotype	Ontological distinction	Example
<<*category*>>	A concept whose instances share common properties but obey different principles of identity.	DBMS Item
<<*collective*>>	A functional complex formed by equal parts, with owner identity principle that is kept both for its instances and in all possible worlds.	Database
<<*kind*>>	A functional complex formed by distinct parts, with owner identity principle that is kept both for its instances and in all possible worlds.	Schema
<<*mode*>>	A concept whose instances represent intrinsic properties of an individual.	Loaded DBMS Copy
<<*role*>>	A concept whose instances do not hold the same principle of identity in all possible worlds, becoming relationally dependent of a rigid identity principle.	Primary Key Column
<<*subkind*>>	A concept whose instances inherit an identity principle from a kind.	Informational Schema

duces semantic interoperability problems shown in open and dynamic scenarios, such as databases; allows integrating different ontological portions to compose an extensive domain and facilitates reuse since new ontologies can appropriate well-founded concepts defined in these ontologies. We briefly summarize this baseline here.

SABiO [Falbo 2014] is a systematic approach for building operational and reference ontologies comprised of five main phases. In **Purpose Identification and Requirements Elicitation** we identify the purpose and intended uses of the ontology, define its functional (Competency Questions) and non-functional requirements (NFRs), and decompose the ontology into independent and interconnected parts to facilitate maintenance and development. **Ontology Capture and Formalization** aims to objectively record the domain conceptualization based on an ontological analysis using a foundation ontology, suggesting the adoption of a graphic model to represent the reference ontology. In **Design**, we define the implementation environment and technological NFRs for the codification of the reference ontology in a machine-readable language. **Implementation** follows, with the codification of the ontology in the chosen operational language. Finally, **Testing** verifies the operational ontology using queries in the implementation environment and validates the ontology on software applications according to its intended uses.

SABiO suggests the use of OntoUML [Guizzardi 2005], an ontology modeling language based on a version of UML 2.0 class diagrams that incorporates important foundational distinctions made by the Unified Foundational Ontology (UFO) [Guizzardi and Wagner 2004], both adopted in this research. Such distinctions are made explicit in the model by means of UML class stereotypes, summarized in Table 1.

Since a database system is a software, we also reuse two ontologies from the Software Engineering Ontology Network (SEON) [Ruy et al. 2016]: the Software Process Ontology (SPO) [de Oliveira Bringuente et al. 2011] and the Software Ontology (SwO) [Duarte et al. 2018]. Figure 1 shows fragments from these ontologies used in this paper.

SPO aims at establishing a common conceptualization on the software process domain, including processes, activities, resources, people, artifacts and procedures. In this paper we are interested in the *Artifacts* and *Resources* sub-ontologies. In the first, we reuse the concept of ***Software Artifacts***, objects consumed or produced during the software process, which are described by a ***Language***, a set of symbols used for encoding and decoding information. *Software Artifacts* can be, among other things, a ***Software Item***,

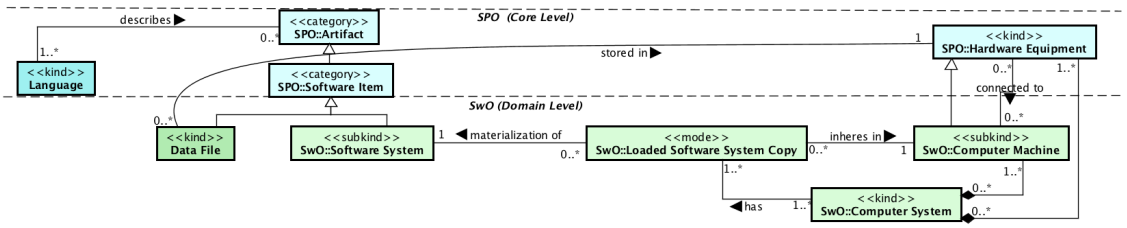

Figure 1. Fragment of SPO and SwO ontologies.

a piece of software, considered an intermediary result, such as the front page of a Web application. In the second sub-ontology, we reuse the concept of ***Hardware Equipment***, a physical object used for running software programs or to support some related action, such as a computer on which Web applications are deployed.

SwO captures that software products are constituted of software artifacts of different natures, including software systems, data file, programs and code. ***Software System*** is a *Software Item* that aims at satisfying a specification, concerning a desired change in a data structure inside a computer, abstracting away from the behavior. A ***Loaded Software System Copy*** is the materialization of a ***Software System***, inhering in a ***Computer Machine*** and associated to a ***Computer System***, which is a system composed of one or more *Computer Machines*, and other *Hardware Equipments*. ***Data File*** is a computer file which stores data in a *Hardware Equipment*.

3. The Relational Database System Ontology (RDBS-O)

The Relational Database System Ontology (RDBS-O) aims to identify and represent key concepts of the relational database domain in the architectural scope, not covering control and execution details. The non-functional requirements of RDBS-O are: be comprehensible and extensible, mainly because the ontology will be used for different purposes; have an operational version of the reference ontology that can be used in applications; be modular or embedded in a modular framework to facilitate reuse of other ontologies and subsequent reuse of this ontology; and be based on well-known sources from the literature. Its functional requirements, i.e., the knowledge the ontology is supposed to represent, are presented in the following subsections.

Ontology capture was supported by a process of knowledge acquisition that used consolidated sources of knowledge, including books [Abbey et al. 2002, Date 2004, Elmasri and Navathe 2011, Melton and Simon 2001] and standards [ISO/IEC9075-1 2008, ISO/IEC9075-2 2003]. Therefore, concepts and relationships were identified and defined in a dictionary of terms [Aguiar 2018] to define and ensure consensual understanding of the domain. The process of formalizing the ontology happened iteratively, in order to address different aspects/refinements at each iteration, and interactively, so domain experts and ontology engineers could discuss the conceptualization of the domain modeled in OntoUML.

Given its scope, we decided to integrate RDBS-O into SEON, in order to reuse relevant concepts, as well as SEON's grounding in UFO. As mentioned in Section 2, two ontologies from SEON were reused: the Software Process Ontology (SPO) and the Software Ontology (SwO). Moreover, we decided to develop a more general ontology

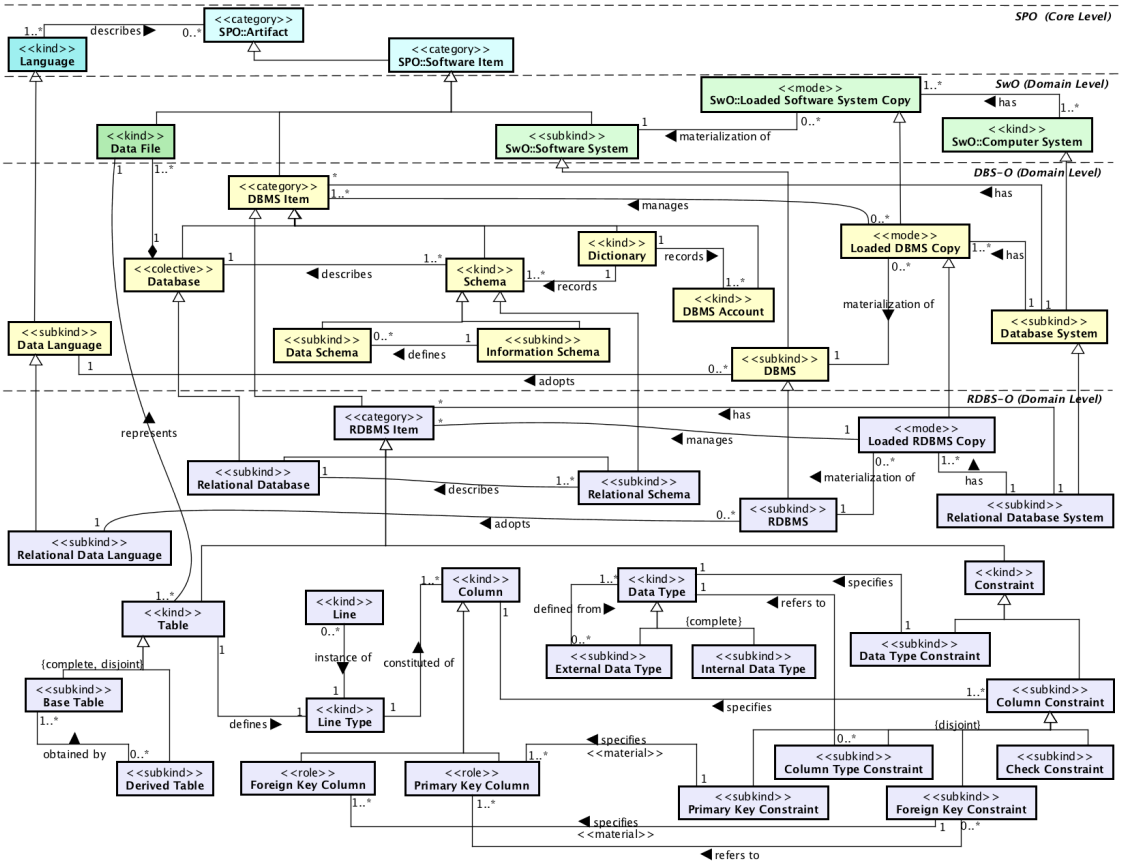

Figure 2. Database System and Relational Database System ontologies.

representing database systems in general, called DBS-O. DBS-O is not limited to relational databases, and thus it can be easily reused as basis for defining ontologies for database systems of other types. RDBS-O, in turn, extends DBS-O, focusing on relational databases. Figure 2 shows the conceptual model of both DBS-O and RDBS-O. In this figure, concepts reused from the Software Processes Ontology and from the Software Ontology are preceded by the corresponding acronyms (*SPO::* and *SwO::*, respectivelly). Next, we describe these ontologies in details and a more detailed specification is available in a technical report [Aguiar 2018]

3.1. The Database System ontology (DBS-O)

The purpose of DBS-O is to represent the concepts relevant to the database system domain in general. However, given how extensive this domain is, this ontology requires further study to properly assess its generalizability. The functional requirements are defined in the following Competency Questions (CQs): **CQ1:** What is the DBMS of a database system? **CQ2:** What are the languages used by a database system? **CQ3:** What is the database of a database system? **CQ4:** What are the schemas of a database? **CQ5:** What is the default schema of a database? **CQ6:** What are the files of a database?

A ***Database System*** (DBS) is a ***Computer System*** whose main purpose is to store information and allow users to seek and update such information when requested [Date 2004]. The advantage of a DBS is data independence, i.e., the file structure

is stored in the ***DBMS*** (Database Management System), a ***Software System*** which ensures that any change in the data physical representation does not drastically affect the programs that use them [Elmasri and Navathe 2011]. The *DBMS* adopts a ***Data Language***, a declarative ***Language*** that describes the problem rather than the solution and specifies what should be done (but not how), to manipulate the data in a *Database*.

The *Database System* has one or more ***Loaded DBMS Copies*** inhering in a computer machine as part of the *Database System*. Each *Loaded DBMS Copy* manages ***DBMS Items***, such as *Database* and *Dictionary*. A ***Database*** is a collection of organized ***Data Files*** so that its content can be easily accessed and managed [Date 2004]. A ***Dictionary*** describes items of system interest [Date 2004], i.e., it records the primary structure of the *Database* [Elmasri and Navathe 2011] such as *Schemas* and *DBMS Accounts*. Each *Dictionary* must record at least one ***DBMS Account***, physical or computational user making requests to the database, and any number of ***Schemas***, a persistent named collection of descriptors [ISO/IEC9075-1 2008]. There is exactly one ***Informational Schema*** which effectively and accurately defines all the settings of all other *Schemas* (said ***Data Schemas***) in the *Dictionary* [Date 2004]. Collectively, the *Schemas* describe the *Database*.

3.2. The Relational Database System ontology (RDBS-O)

RDBS-O specializes the concepts of DBS-O for the relational databases domain. The functional requirements are defined the following Competency Questions (CQs): **CQ1:** What are the tables of a database system? **CQ2:** What data does a table hold? **CQ3:** What are the columns of a table? **CQ4:** What is the data type of a column? **CQ5:** What is the primary key of a table? **CQ6:** Which columns refer to a primary key? **CQ7:** What is the foreign key of a table? **CQ8:** Which columns refer to a foreign key? **CQ9:** Which tables are related by means of a foreign key? **CQ10:** Which constraint specify a data type? **CQ11:** Which constraints specifies a column? **CQ12:** What are the base tables of a database system? **CQ13:** What are the derived tables of a database system?

Relational Database Management System (***RDBMS***) is a specialization of a Database Management System (*DBMS*) whose data abstraction is based on the relational model [Date 2004] and, therefore, its main ***RDBMS Item*** is the ***Table***. In an *RDBMS*, a *Data File* is represented as *Tables*, defined by a ***Line Type*** that is instantiated as ***Lines***, which are true propositions. A *Line Type* is the most specific type of a line [ISO/IEC9075-2 2003], such that all lines in a table have an unique line type [ISO/IEC9075-1 2008]. Each *Line Type* is constituted of a set of ***Columns*** that represent a table field [ISO/IEC9075-1 2008]. In an *RDBMS* the *Line* is the smallest data unit that can be inserted and deleted from a *Table* [ISO/IEC9075-2 2003]. A *Table* is either a ***Base Table***, which represents data stored in the database in an autonomous and independent way, such as persistent, global temporary, or local temporary tables [Melton and Simon 2001]; or a ***Derived Table***, non-base table which can be obtained by means of relational expression on one or more *Base Tables* in a non-autonomous and dependent way [Date 2004] (e.g., a view).

A *Column* is specified by ***Column Constraints***, being exactly one ***Column Type Constraint*** in order to restrict the values that a *Column* can take [Date 2004] with respect to a ***Data Type*** [Abbey et al. 2002, ISO/IEC9075-2 2003, Melton and Simon 2001], which in turn is a set of representable values specifying the information type maintained

in a *Column* [ISO/IEC9075-2 2003, Abbey et al. 2002]. The *Data Type* can be either an ***Internal Data Type***, defined by the *RDBMS*, or an ***External Data Type***, defined by the user based on existing *Data Types* [Date 2004]. In addition, a *Data Type* T is specified by a ***Data Type Constraint*** that defines the set of valid values for T. However, a *Column Type Constraint* may never be violated if the *Data Type Constraints* are checked [Date 2004].

Furthermore, a *Column Constraint* can be a ***Check Constraint***, a ***Primary Key Constraint*** or a ***Foreign Key Constraint***. A *Check Constraint* specifies a condition that must be satisfied for any *Line* of the *Table* [ISO/IEC9075-1 2008, ISO/IEC9075-2 2003], such as a condition of values, a null value, a special value used to indicate the absence of any data value in a column, a unique value (i.e, a *Column* should not have two equal non-null values), a default value, etc. A *Primary Key Constraint* is an integrity constraint that satisfies both uniqueness and irreducibility properties, i.e., it defines what makes a data line exclusive within a table [Date 2004, Abbey et al. 2002]. It is the combination of the unique and the null value *Check Constraints*, however, a *Table* can have at most one *Primary Key Constraint* and any number of *Check Constraints* [Date 2004]. A *Foreign Key Constraint* is a referential constraint that specifies one or more *Columns* of a reference table that correspond to *Columns* in a referenced table [ISO/IEC9075-2 2003].

Thus, a *Column* can assume the role of ***Primary Key Column*** — which identifies a unique non-null value for any table instance when associated with a *Primary Key Constraint* — or the role of ***Foreign Key Column*** — which belongs to a referencing table and whose values correspond to those of a *Primary Key Column* of some referenced table associated with a *Foreign Key Constraint* [Date 2004].

4. Evaluation

The RDBS-O ontology, presented in Section 3, was implemented in OWL, giving rise to its operational version RDBS-OWL.[1] This process was done manually in order to balance expressiveness and computational properties, through adaptations in RDBS-OWL. Verification, validation and testing techniques, as suggested by SABiO, followed.

For **ontology verification**, SABiO suggests we demonstrate that the ontology elements are able to answer the Competency Questions (CQs) that were raised. Table 2 describes a part of the RDBS-O verification presenting the results for some of its CQs.

Ontology validation and testing activities were supported by a semi-automatic approach for constructing and publishing ontologies from relational databases, illustrated in Figure 3. The activities were performed using a sample created by Oracle, called **HR database**,[2] a schema of Human Resources application whose main purpose is to store the records of employees of an organization.

For **ontology validation**, the reference ontology should be instantiated to check if it is able to represent real-world situations, i.e., the structure of a relational database. For this, we elaborated a **mapping file** for the Oracle RDBMS using the RDBS-OWL operational ontology, which indirectly instantiates RBDS-O with real entities from the

[1]The ontology and all other resources mentioned in this section are available at `http://nemo.inf.ufes.br/projects/sfwon/` so the interested readers can perform this evaluation for themselves.

[2]HR data model available at `http://www.oracle.com/technetwork/developer-tools/datamodeler/sample-models-scripts-224531.html`.

Table 2. Results for RDBS-O verification.

CQ	Concepts and *Relations*
CQ1	Loaded RDBMS Copy *materialization of* RDBMS and Loaded RDBMS Copy *manages* RDBMS Item and RDBMS Item *specialized in* Table
CQ2	Table *defines* Line Type and Line *instance of* Line Type
CQ3	Table *defines* Line Type and Line Type *constituted of* Column
CQ4	Column Constraint *specifies* Column; Column Constraint *specialized in* Column Type Constraint and Column Type Constraint *refers to* Data Type.
CQ5	Primary Key Constraint *specializes* Column Constraint; Column Constraint *specifies* Column; Table *defines* Line Type and Line Type *constituted of* Column
CQ6	Primary Key Constraint *specifies* Primary Key Column and Primary Key Column *specialized of* Column
CQ7	Foreign Key Constraint *specializes* Column Constraint; Column Constraint *specifies* Column; Table *defines* Line Type and Line Type *constituted of* Column
CQ8	Foreign Key Constraint *specifies* Foreign Key Column and Foreign Key Column *specialized of* Column
CQ9	Foreign Key Constraint *specializes* Foreign Key Column; Foreign Key Column *specialized of* Column; Table *defines* Line Type and Line Type *constituted of* Column Foreign Key Constraint *refers to* Primary Key Column; Primary Key Column *specialized of* Column; Table *defines* Line Type and Line Type *constituted of* Column
CQ10	Data Type Constraint *specifies* Data Type Data Type Constraint *specifies* Data Type and Data Type *specialized in* Internal Data Type Data Type Constraint *specifies* Data Type and Data Type *specialized in* External Data Type
CQ11	Column Constraint *specifies* Column and Column Constraint *specializes* Column Type Constraint, Primary Key Constraint, Foreign Key Constraint and Check Constraint
CQ12	Loaded RDBMS Copy *materialization of* RDBMS and Loaded RDBMS Copy *manages* RDBMS Item and RDBMS Item *specialized in* Table and Table *specializes* Base Table
CQ13	Loaded RDBMS Copy *materialization of* RDBMS and Loaded RDBMS Copy *manages* RDBMS Item and RDBMS Item *specialized in* Table and Table *specializes* Derived Table

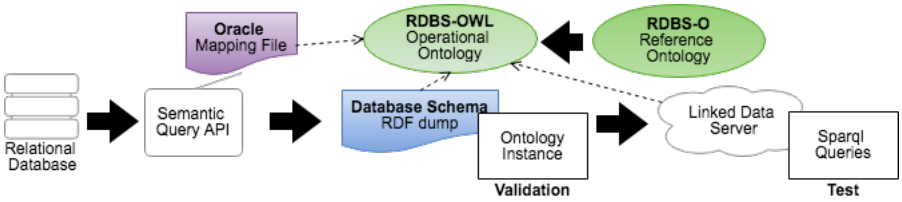

Figure 3. Approach to assist validation and testing of the ontology.

domain, hence validating it. The mapping file is used as input to a **Semantic Query API**, namely D2RQ,[3] which performs SQL queries to extract information from the database based on the mapping. Then, an **RDF dump** describing such extracted information using concepts defined in an ontology is generated, as an instance of RDBS-OWL, with real information from a relational database. Table 3 describes part of the RDBS-O validation, presenting some of the concepts instantiated by the ontology from the HR database.

For the **test cases**, we performed SPARQL queries using the endpoint provided by D2RQ, which works as a Linked Data Server (i.e., a triplestore) over the RDF dump generated during validation. The queries correspond to the CQs presented in Section 3. Table 4 describes a part of the RDBS-O test cases presenting some of these queries from the HR database and their test results.

Starting from this validation approach, and provided other RDBMSs (e.g., MySQL) are mapped to the ontology, one could perform database migration, publish the database structure (or even its contents) as linked data, etc. Due to space constraints, only part of the results of RDBS-O evaluation is presented here, but the interested reader can refer to the aforementioned website for the complete results.

[3] `http://d2rq.org`

Table 3. Results of RDBS-O instantiation using the HR database.

Concept	Instances
Table	**Instance:** COUNTRIES
	Dump: <rdf:Description rdf:about="#TABLE/COUNTRIES"> <rdbs-owl:HASCOLUMN rdf:resource="#COLUMN/COUNTRIES/COUNTRIES_ID"/> <rdbs-owl:HASCOLUMN rdf:resource="#COLUMN/COUNTRIES/COUNTRY_NAME"/> <rdbs-owl:HASCOLUMN rdf:resource="#COLUMN/COUNTRIES/REGION_ID"/> <rdfs:label>#COUNTRIES</rdfs:label> <rdf:type rdf:resource="http://rdbsowl/TABLE"/> </rdf:Description>
Column	**Instance:** COUNTRY_ID
	Dump:<rdf:Description rdf:about="#COLUMN/COUNTRIES/COUNTRY_ID"> <rdbs-owl:BELONGSTOTABLE rdf:resource="#TABLE/COUNTRIES"/> <rdbs-owl:DEFINEDBYDATATYPE>#DATATYPE/CHAR</rdbo:DEFINEDBYDATATYPE> <rdfs:label>COLUMN #CONTRY_ID</rdfs:label> <rdf:type rdf:resource="http://rdbs-owl/COLUMN"/> </rdf:Description>
Primary key Column	**Instance:** COUNTRY_ID
	Dump:<rdf:Description rdf:about="#PRIMARYKEYCOLUMN/COUNTRIES/COUNTRY_ID"> <rdbs-owl:BELONGSTOTABLE rdf:resource="#TABLE/COUNTRIES"/> <rdfs:label>PRIMARYKEYCOLUMN COUNTRIES COUNTRY_ID</rdfs:label> <rdf:type rdf:resource="http://rdbs-owl/PRIMARYKEYCOLUMN"/> <rdbs-owl:REFERESTOCOLUMN rdf:resource="#COLUMN/COUNTRIES/COUNTRY_ID"/> </rdf:Description>
Foreign key Column	**Instance:** REGION_ID
	Dump:<rdf:Description rdf:about="#FOREIGNKEYCOLUMN/COUNTRIES/REGION_ID"> <rdbs-owl:REFERSTOCOLUMN rdf:resource="#COLUMN/COUNTRIES/REGION_ID"/> <rdbs-owl:REFERSTOPRIMARYKEYCOLUMN rdf:resource="#PRIMARYKEYCOLUMN/REGIONS/ REGION_ID"/> <rdbs-owl:BELONGSTOREFERENCINGTABLE rdf:resource="#REFERENCING-TABLE/COUNTRIES"/> <rdbs-owl:REFERSTOREFERENCEDTABLE rdf:resource="#REFEREN-CEDTABLE/REGIONS"/> <rdfs:label>FOREIGNKEYCOLUMN COUNTRIES REGION_ID</rdfs:label> <rdf:type rdf:resource="http://rdbs-owl/FOREIGNKEY-COLUMN"/> </rdf:Description>
Primary key Constraint	**Instance:** PK_COUNTRY
	Dump:<rdf:Description rdf:about="#PRIMARYKEYCONSTRAINT/PK_COUNTRY"> <rdbs-owl:DEFINESPRIMARYKEYCOLUMN rdf:resource="#PRIMARYKEYCOLUMN/COUNTRIES/ COUNTRY_ID"/> <rdbs-owl:SPECIFIESTABLE rdf:resource="#TABLE/COUNTRIES"/> <rdfs:label>PRIMARYKEYCONSTRAINT PK_COUNTRY</rdfs:label> <rdf:type rdf:resource="http://rdbs-owl/PRIMARYKEYCONSTRAINT"/> </rdf:Description>
Foreign key Constraint	**Instance:** FK_COUNTRY_REGION
	Dump:<rdf:Description rdf:about="#FOREIGNKEYCONSTRAINT/FK_COUNTRY_REGION"> <rdbs-owl:DEFINESFOREIGNKEYCOLUMN rdf:resource="#FOREIGNKEYCOLUMN/COUNTRIES/ REGION_ID"/> <rdbs-owl:REFERESTOPRIMARYKEYCONSTRAINT rdf:resource="#PRIMARY-KEYCONSTRAINT/PK_REGION"/> <rdbs-owl:SPECIFIESTABLE rdf:resource="#TABLE/ COUNTRIES"/> <rdfs:label>FOREIGNKEYCONSTRAINT FK_COUNTRY_REGION</rdfs:label> <rdf:type rdf:resource="http://rdbs-owl/FOREIGNKEYCONSTRAINT"/> </rdf:Description>

5. Related Works

In this section, we compare our proposal to some ontologies designed to describe the relational database structure, which are part of approaches to build ontologies about the schema and the data from relational databases, similar to the approach presented in Section 4. Although such ontologies represent the same domain, none of them present a reference ontology built on an foundation ontology such as RDBS-O.

The Relational.OWL [de Laborda and Conrad 2005] ontology describes the schema of a relational database in the abstract form and generates a representation format of the database itself. The ontology is written in Java with JDBC and Jena. Thus, from the platform-independent representation, a database extracted from a platform A can be imported on a platform B, currently supporting MySQL and DB2. The ontology consists of four classes (dbs:Database, dbs:Table, dbs:Column, dbs:PrimaryKey) and seven properties (dbs:has, dbs:hasTable, dbs:hasColumn, dbs:isIndentifiedBy, dbs:references, dbs:scale, dbs:length). An ontological analysis on Relational.OWL has resulted in some questionable points: (i) the model is represented only in OWL, which is not very expres-

Table 4. Test cases using SPARQL queries.

CQ	SPARQL Query	Test Results
CQ1	`SELECT DISTINCT ?instance WHERE ?instance a rdbs-o:TABLE ORDER BY ?instance`	COUNTRIES, DEPARTMENTS, EMPLOYEES, JOBS, JOB_HISTORY, LOCATIONS, REGIONS
CQ3	`SELECT DISTINCT ?instance WHERE ?instance a rdbs-o:COLUMN ; rdbs-o:BELONGSTOTABLE ?table. FILTER regex(?table, "COUNTRIES") ORDER BY ?instance`	COUNTRIES_ID, COUNTRY_NAME, REGION_ID
CQ5	`SELECT DISTINCT ?instance WHERE ?instance a rdbs-o:PRIMARYKEYCONSTRAINT; rdbs-o:DEFINESPRIMARYKEYCOLUMN ?column. FILTER regex(?column , "COUNTRIES_ID") ORDER BY ?instance`	COUNTRY_C_ID_PK
CQ6	`SELECT DISTINCT ?instance WHERE ?instance a rdbs-o:PRIMARYKEYCOLUMN; rdbs-o:BELONGSTOTABLE ?table. FILTER regex(?table , "COUNTRIES") ORDER BY ?instance`	COUNTRY_ID
CQ7	`SELECT DISTINCT ?instance WHERE ?instance a rdbs-o:FOREIGNKEYCONSTRAINT; rdbs-o:DEFINESPRIMARYKEYCOLUMN ?column. FILTER regex(?column , "COUNTRIES_ID") ORDER BY ?instance`	COUNTRY_C_ID_PK
CQ8	`SELECT DISTINCT ?instance WHERE ?instance a rdbs-o:FOREIGNKEYCOLUMN; rdbs-o:BELONGSTOTABLE ?table. FILTER regex(?table , "COUNTRIES") ORDER BY ?instance`	REGION_ID

sive; (ii) cardinality associations are not presented; and (iii) it presents limited coverage of the domain, not considering, for instance, data type, foreign keys and constraints.

The OWL-RDBO [Trinh et al. 2006] ontology describes an OWL vocabulary, its semantic relationships, and constraints of the relational database system. From this vocabulary, a tool generates and publishes ontology instances dynamically. OWL-RDBO is written in Java with JDBC in order to extract metadata and structural constraints from the database, currently supporting MySQL, PostegreSQL and DB2, and generate an ontology in OWL as an instance of OWL-RDBO. The paper describes that the ontology is formed by some classes (rdbo:DatabaseName, rdbo:RelationList, rdbo:Relation, rdbo:AttributeList, rdbo:Attribute) and properties (rdbo:hasRelations, rdbo:hasType, rdbo:referenceAttribute, rdo:referenceRelation), but the full ontology is not available for access. An ontological analysis on OWL-RDBO has resulted in some questionable points: (i) the model is represented only in OWL, which is not very expressive; (ii) what OWL-RDBO defines as a Relation, we decided to use the term Table in order to decrease semantic conflicts; and (iii) it includes concepts external to the domain, such as RelationList to group a set of Relation and AttributeList to group a set of attributes.

Other approaches, such as [Barrett et al. 2002, Bizer 2003] convert data stored in relational database to RDF objects from query into domain-specific ontologies. These and other approaches aim to represent the real world relations from a domain ontology and not an ontology representing the database structure, which is more accurate but unable to reconstruct the data to the original format of the database.

In order to analyze the related ontologies according to the coverage on the relational database domain, we verified if they answer the Competency Questions presented in Section 3. The results of this verification are shown in Figure 4. As we can observe, although Relational.OWL and OWL-RDBO answer some questions, they do not consider some important aspects, especially those that allow reverse engineering of the model. Note that CQ2 is answered by RDBS-O, however to be included by RDBS-OWL it is necessary to convert RDBS-O instantiated from the database structure into a second

ontology and instance it with the data from that database.

Ontologies	Competence Questions													CVGE
	CQ1	CQ2	CQ3	CQ4	CQ5	CQ6	CQ7	CQ8	CQ9	CQ10	CQ11	CQ12	CQ13	
Relational.OWL	X		X		X			X						31%
OWL-RDBO	X		X	X	X		X							38%
RDBS-OWL	X		X	X	X	X	X	X	X	X	X	X	X	92%

Figure 4. Competency questions answered by related ontologies.

6. Final Considerations

In this paper we present an ontology on relational database systems in order to represent the domain according to ontological foundations. RDBS-O is built according to an ontology engineering method and based on well-known data sources. The ontology incorporates concepts based on software engineering and database system ontologies in order to clarify the relationship with the relational database domain. Moreover, the ontology allows us to semantically represent any relational database structure independent of vendor.

Verification, validation and testing activities were successfully completed with emphasis on the RDBS-O sub-ontology coverage. Furthermore, the RDBS-OWL operational version was applied in a semi-automatic approach that instantiates its concepts from a database. Thus, the ontology, along with this approach, provide a way to describe relational database systems using vocabulary defined in RDBS-O. Such approach can be extended to demonstrate other uses of the ontology.

Finally, future work intends to apply the ontology in the context of semantic interoperability among object/relational mapping frameworks and the definition of architectural smells in software projects.

Acknowledgments

NEMO (`http://nemo.inf.ufes.br`) is currently supported by Brazilian research funding agencies CNPq (process 407235/2017-5), CAPES (process 23038.028816/2016-41), and FAPES (process 69382549/2015).

References

Abbey, M., Corey, M., and Abramson, I. (2002). Oracle9i–guia introdutório–aprenda os fundamentos do oracle 9i. *Editora Campus, Rio de Janeiro–RJ*.

Aguiar, C. Z. (2018). Relational Database Ontology — Reference Ontology Specification Document (in Portuguese), available on: `http://nemo.inf.ufes.br/projects/sfwon/`. Technical report, Federal University of Espírito Santo (UFES).

Barrett, T., Jones, D., Yuan, J., Sawaya, J., Uschold, M., Adams, T., and Folger, D. (2002). Rdf representation of metadata for semantic integration of corporate information resources. In *International Workshop Real World and Semantic Web Applications*, volume 2002. Citeseer.

Bauer, C. and King, G. (2004). *Hibernate in Action*. Manning, 1 edition.

Bizer, C. (2003). D2R MAP - A Database to RDF Mapping Language. In *Proc. of the 12th International World Wide Web Conference - Posters*.

Date, C. J. (2004). *Introdução a sistemas de bancos de dados*. Elsevier Brasil.

de Laborda, C. P. and Conrad, S. (2005). Relational. owl: a data and schema representation format based on owl. In *Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling-Volume 43*, pages 89–96. Australian Computer Society, Inc.

de Oliveira Bringuente, A. C., de Almeida Falbo, R., and Guizzardi, G. (2011). Using a foundational ontology for reengineering a software process ontology. *Journal of Information and Data Management*, 2(3):511.

Duarte, B. B., Leal, A. L. d. C., Falbo, R. A., Guizzardi, G., Guizzardi, R. S. S., and Souza, V. E. S. (2018). Ontological Foundations for Software Requirements with a Focus on Requirements at Runtime. *Applied Ontology*, Preprint(Preprint):1–33.

Elmasri, R. and Navathe, S. (2011). *Database systems*, volume 9. Pearson Education Boston, MA.

Falbo, R. A. (2014). Sabio: Systematic approach for building ontologies. In *ONTO. COM/ODISE@ FOIS*.

Guido, A. L. and Paiano, R. (2010). Semantic integration of information systems. *International Journal of Computer Networks and Communications (IJCNC)*, 2.

Guizzardi, G. (2005). *Ontological Foundations for Structural Conceptual Models*. PhD Thesis, University of Twente, The Netherlands.

Guizzardi, G. and Wagner, G. (2004). A unified foundational ontology and some applications of it in business modeling. In *CAiSE Workshops (3)*, pages 129–143.

ISO/IEC9075-1 (2008). Information technology–database languages–sql–part 1: Framework (sql/framework).

ISO/IEC9075-2 (2003). Information technology–database languages–sql–part 2: Foundation (sql/foundation). *ISO/IEC*.

Melton, J. and Simon, A. R. (2001). *SQL1999 understanding relational language components*. Elsevier.

Ruy, F. B., de Almeida Falbo, R., Barcellos, M. P., Costa, S. D., and Guizzardi, G. (2016). Seon: A software engineering ontology network. In *European Knowledge Acquisition Workshop*, pages 527–542. Springer.

Studer, R., Benjamins, V. R., and Fensel, D. (1998). Knowledge engineering: principles and methods. *Data & knowledge engineering*, 25(1-2):161–197.

Trinh, Q., Barker, K., and Alhajj, R. (2006). Rdb2ont: A tool for generating owl ontologies from relational database systems. In *Telecommunications, 2006. AICT-ICIW'06. International Conference on Internet and Web Applications and Services/Advanced International Conference on*, pages 170–170. IEEE.

Trinh, Q., Barker, K., and Alhajj, R. (2007). Semantic interoperability between relational database systems. In *Database Engineering and Applications Symposium, 2007. IDEAS 2007. 11th International*, pages 208–215. IEEE.