Undefined 1 (2009) 1–5
IOS Press

RODI: Benchmarking Relational-to-Ontology Mapping Generation Quality

Editor(s): Name Surname, University, Country
Solicited review(s): Name Surname, University, Country
Open review(s): Name Surname, University, Country

Christoph Pinkel [a,*], Carsten Binnig [b], Ernesto Jiménez-Ruiz [c], Evgeny Kharlamov [c], Wolfgang May [d], Andriy Nikolov [a], Martin G. Skjæveland [e], Alessandro Solimando [f,g], Mohsen Taheriyan [h], Christian Heupel [a] and Ian Horrocks [c]

[a] *fluid Operations AG, Walldorf, Germany*
[b] *Brown University, Providence, RI, USA*
[c] *University of Oxford, United Kingdom*
[d] *Göttingen University, Germany*
[e] *University of Oslo, Norway*
[f] *Università di Genova, Genoa, Italy*
[g] *Inria Saclay & Université Paris-Sud, Orsay, France*
[h] *University of Southern California, Los Angeles, CA, USA*

Abstract Accessing and utilizing enterprise or Web data that is scattered across multiple data sources is an important task for both applications and users. Ontology-based data integration, where an ontology mediates between the raw data and its consumers, is a promising approach to facilitate such scenarios. This approach crucially relies on high-quality mappings to relate the ontology and the data, the latter being typically stored in relational databases. A number of systems to help mapping construction have recently been developed. A generic and effective benchmark for reliable and comparable evaluation of mapping quality would make an important contribution to the development of ontology-based integration systems and their application in practice. We propose such a benchmark, called *RODI*, and evaluate various systems with it. It offers test scenarios from conference, geographical, and oil and gas domains. Scenarios are constituted of databases, ontologies, mappings, and queries to test expected results. Systems that compute relational-to-ontology mappings can be evaluated using *RODI* by checking how well they can handle various features of relational schemas and ontologies, and how well computed mappings work for query answering. Using *RODI* we conducted a comprehensive evaluation of six systems.

Keywords: Mappings, Relational databases, RDB2RDF, R2RML, Benchmarking, Bootstrapping

1. Introduction

1.1. Motivation

Accessing and utilizing enterprise or Web data that is scattered across multiple databases is an important task for both applications and users in many scenarios [32,4]. Ontology-based data integration is a promising approach to this task, and recently it has been successfully applied in academia as well as in industry [17,11,16,9,12]. The main idea behind this approach is to employ an ontology—a vocabulary of classes and properties and a set of formal axioms capturing their semantics—to mediate between data consumers and (possibly multiple) databases. The ontology describes the application domain using terms that are familiar to data consumers, and it provides a conceptual schema over which consumers can formulate queries; it is related to the databases via mappings that associate each ontological term with underlying data. Mappings can be used either to export data to consumers by transforming it from the schema(s) of the underlying database(s) into the schema defined by the ontology via a suitable ETL (extract, transform, load) process, or to trans-

*Corresponding author. E-mail: christoph.pinkel@fluidops.com.

late (or *rewrite*) consumer queries into queries over the database(s). The latter approach (query rewriting) is often referred to as Ontology Based Data Access (OBDA).

Ontology-based data integration crucially depends on the quality of ontologies and mappings. Ontology development has attracted a lot of attention in the last decade, and ontologies have been developed for various domains including life sciences (e.g., [19]), medicine (e.g., [26]), the energy sector (e.g., [16]), and others. Many of these ontologies are of good quality, generic, and can be used in ontology-based integration scenarios.

Mapping development has, however, received much less attention. Moreover, existing mappings are typically tailored to relate generic ontologies to specific database schemata. As a result, in contrast to ontologies, mappings typically cannot be reused across integration scenarios. Thus, each new integration scenario essentially requires the development of mappings from scratch. This is a complex and time consuming process that calls for automatic or semi-automatic support, i.e., systems that (semi-) automatically construct mappings of good quality, and in order to address this challenge, a number of systems that generate relational-to-ontology mappings have recently been developed [8,40,14,53,3,46,28].

The quality of such generated relational-to-ontology mappings is usually evaluated using self-designed and therefore potentially biased benchmarks, which makes it difficult to compare results across systems, and does not provide enough evidence to select an adequate mapping generation system in ontology-based data integration projects. This limitation is evident in large scale industrial projects where support from (semi-)automatic systems is vital [17,16]. Thus, in order to ensure that ontology-based data integration can find its way into mainstream practice, there is a need for a generic and effective benchmark that can be used for the reliable evaluation of the *quality* of computed mappings w.r.t. their utility under actual query workloads. *RODI*, our mapping-quality benchmark for *R*elational-to-*O*ntology *D*ata *I*ntegration scenarios, addresses this challenge.

1.2. RODI *Benchmark Approach*

The *RODI* benchmark is composed of *(i)* a *framework* to test systems that generate mappings between relational schemata [27] and OWL 2 ontologies [5], *(ii)* a *scoring function* to measure the quality of system-generated mappings, *(iii)* different datasets and queries for benchmarking, which we call *benchmark scenarios*, and *(iv)* a *mechanism* to extend the benchmark with additional scenarios. Using *RODI* one can evaluate the *quality* of relational-to-ontology mappings produced by systems for ontology-based data integration from two perspectives: how good the mappings can translate between various particularities of relational schemata and ontologies, and how good they are from the query answering perspective.

To make this possible, *RODI* is designed as an end-to-end benchmark. That is, we consider systems that can produce mappings directly between relational databases and ontologies. Also, we evaluate mappings according to their utility for an actual query workload.

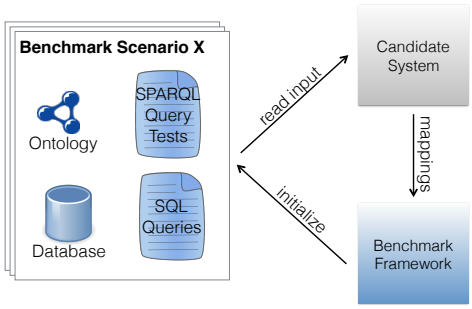


Figure 1. *RODI* benchmark overview

Figure 1 depicts the schematics of the resulting architecture: the benchmark comes with a number of benchmark scenarios. Scenarios are initialized and setup for use by the framework. Candidate systems then read their input from the active scenario and produce mappings, which are evaluated again by our framework.

1.3. *Contributions*

The main contributions of this paper are as follows:

- *Systematic analyses of challenges and existing approaches in relational-to-ontology mapping generation*: This supports us in drawing up a list of requirements for a generic and flexible benchmark that produces meaningful scores.
- *Evaluation scenarios*: *RODI* consists of 18 ontology-based data integration test scenarios from conference, geographical, and oil and gas domains. Scenarios are constituted of databases, ontologies, mappings, and queries to check expected results. Components of the scenarios are developed in such a way that they capture the key challenges of relational-to-ontology mapping generation.

– *The* RODI *framework*: the *RODI* software package, including all scenarios, has been implemented and made available for public download under an open source license.[1]
– *System Evaluation*: we used *RODI* to evaluate six relational-to-ontology systems: BootOX [14], COMA++ [10], IncMap [40], MIRROR [3], the -ontop- bootstrapper [23], and Karma [8]. The systems are chosen in a way that they cover the breadth of recent and traditional approaches in (semi-)automatic schema-to-ontology mapping generation. The insights gained from the evaluation allow us to point to specific strengths and weaknesses of individual systems and to propose how they can be improved.

We have originally introduced *RODI* in [7]. Results presented in this paper significantly extend our previous and preliminary results from [7] in several important ways: *(i) extended evaluation scenarios*: we provide 9 new evaluation scenarios that are important for testing mapping quality under real-world challenges such as high semantic heterogeneity or complex query workloads in different application domains. *(ii) extended scope of the benchmark*: now we can compare semi-automatic and fully automatic mapping generation, and support several modes of evaluation. *(iii) extended evaluation*: two more important systems, COMA++ and Karma, are evaluated and the discussion of evaluation results is significantly extended. Besides, we have modified the benchmark scenarios to produce more specific individual scores rather than aggregated values for relevant categories of tests. We also extended the benchmark framework to allow detailed debugging of the results for each individual test. On that basis we can point to individual issues and bugs in several systems, some of which have already been addressed by the authors of the evaluated systems.

1.4. Outline

First, we present our analysis of the different types of mapping challenges for relational-to-ontology mapping generation in Section 2. Then, in Section 3 we discuss differences in mapping generation approaches that impact mapping generation, and thus also need to be considered for designing appropriate evaluation approaches. Section 4 presents our benchmark suite and the evaluation procedure. Afterwards, Section 5 discusses some implementation details that should help researchers and practitioners to understand how their systems could be evaluated in our benchmarking suite. Section 6 then presents our evaluation, including a detailed discussion of results. Finally, Section 7 summarizes related work and Section 8 concludes the paper and provides an outlook on future work.

[1] https://github.com/chrpin/rodi

2. Integration Challenges

In the following we discuss our classification of different types of mapping challenges in relational-to-ontology data integration scenarios. As a high-level classification, we use the standard classification for data integration described by Batini et al. [1]: naming conflicts, structural heterogeneity, and semantic heterogeneity. For each challenge, we describe the central issue of the problem and the main task faced by the mapping generation tools.

2.1. Naming Conflicts

Typically, relational database schemata and ontologies use different conventions to name their artifacts even when they model the same domain and thus should use a similar terminology. While database schemata tend to use short identifiers for tables and attributes that often include technical artifacts (e.g. for tagging primary keys and for foreign keys), ontologies typically use long "speaking" names. Thus, the main challenge is to be able to find similar names despite the different naming patterns.

Other traditional differences include the use of plural vs. singular for class types, typically different tokenization schemes, etc. Those differences are not present in other cases of data integration (e.g., relational-to-relational or ontology alignment).

2.2. Structural Heterogeneity

The most important differences in relational-to-ontology integration scenarios compared to other integration scenarios are structural heterogeneities. We discuss the different types of structural heterogeneity covered by *RODI*.

2.2.1. Type Conflicts

Relational schemata and ontologies represent the same artifacts by using different modeling constructs. While relational schemata use tables, attributes, and constraints, ontologies use modeling elements such as classes, data properties and object properties, restrictions, etc. Clearly there exist direct (i.e., naive) mappings from relational schemata to ontologies for some

of the elements (e.g., some classes immediately map to tables). However, most real-world relational schemata and corresponding ontologies cannot be related by any such naive mapping. This is because big differences exist in the way how the same concepts are modeled (i.e., type conflicts). Consequently, mapping rules need to be much more complex. One reason why these differences are so big is that relational schemata often are optimized towards a given workload (e.g., they are normalized for update-intensive workloads or denormalized for read-intensive workloads). Ontologies, on the other side, model a domain on the conceptual level. Another reason is that some modeling elements have no single direct translation (e.g., class hierarchies in ontologies can be mapped to relational schemata in different ways). In the following, we list the different type conflicts covered by *RODI*:

1. *Normalization artifacts:* Often properties that belong to a class in an ontology are spread over different tables in the relational schema as a consequence of normalization.
2. *Denormalization artifacts:* For read-intensive workloads, tables are often denormalized. Thus, properties of different classes in the ontology might map to attributes in the same table.
3. *Class hierarchies:* Ontologies typically make use of explicit class hierarchies. Relational models implement class hierarchies implicitly, typically using one of three different common modeling patterns (c.f., [27, Chap. 3]). Figure 2 illustrates those patterns: (1) In one common variant the relational schema materializes several subclasses in the same table and uses additional attributes to indicate the subclass of each individual. Those additional attributes can take the shape of a numeric type column for disjoint subclasses and/or a combination of several type or role flags for non-disjoint subclasses. In this case, several classes need to be mapped to the same table and can be told apart only by secondary features in the data, such as the value in a type column. With this variant, mapping systems have to resolve n:1 matches, i.e., they need to filter from one single table to extract information about different classes. (2) Another common way is to use one table per most specific class in the class hierarchy and to materialize the inherited attributes in each table separately. Thus, the same property of the ontology must be mapped to several tables. In this variant, mapping systems need to resolve 1:n matches, i.e., build a union of information from several tables to retrieve entities for a single class. (3) A third variant uses one table for each class in the hierarchy, including the possibly abstract superclasses. Tables then use primary key-foreign key references to indicate the subclass relationship. This variant has a closer resemblance to ontology design patterns. However, it is also rarely used in practice, as it is more difficult to design, harder to query, impractical to update, and usually considered unnecessarily complex.

2.2.2. Key Conflicts

In ontologies and relational schemata, keys and references are represented differently. In the following, we list the different key conflicts covered by *RODI*:

1. *Keys:* Keys in databases are usually implemented using primary keys and unique constraints. Keys may be composite and in some cases partial keys of a table identify different related entities (e.g., denormalized tables on the relational side). Ontologies use IRIs as identifiers for individuals. Technically, OWL 2 also supports a notion of keys, but this feature is very rarely used.
   Thus, the challenge is that integration tools must be able to generate mapping rules for creating IRIs for individuals from the correct choice of keys.
2. *References:* A similar observation holds for references. While references are typically modeled as foreign keys in relational schemata, ontologies use object properties. Moreover, sometimes relational databases do not model foreign key constraints at all. In that case an integration tool must be able to derive references from the relational schema (e.g., based on the naming scheme and types or individuals).

2.2.3. Dependency Conflicts

These conflicts arise when a group of concepts are related among each other with different dependencies (i.e., 1:1, 1:n, n:m) in the relational schema and the ontology. Relational schemata may use foreign keys over attributes as constraints to explicitly model 1:1 and 1:n relationships between different tables. They often model n:m relationships using an additional connecting table, which describes a *relationship relation*. Ontologies may model functionalities (i.e., functional properties or inverse functional properties) or they define cardinalities explicitly using cardinality restrictions. However, many ontologies do not make use of these restrictions and thus are often underspecified in this respect [6].

Table 1 lists all specific testable relational-to-ontology structural challenges that we have identified.

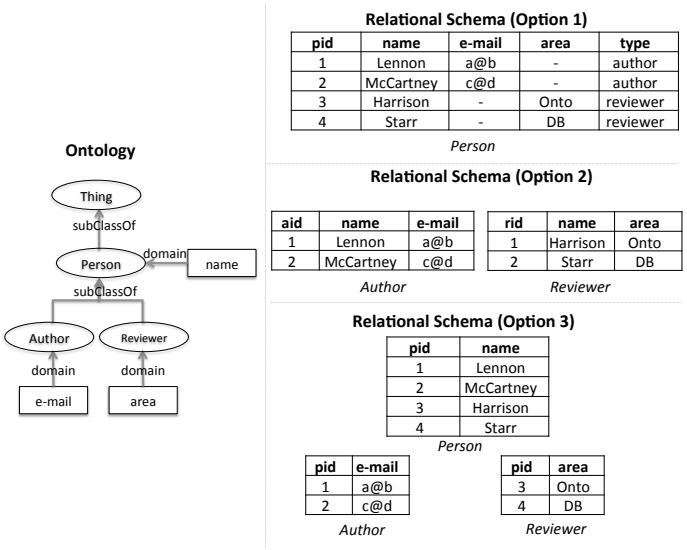


Figure 2. Class hierarchies – ontology vs. relational schema

Table 1

Detailed list of specific structural mapping challenges. RDB patterns may correspond to some of the "guiding" ontology axioms. Specific difficulties explain particular hurdles in constructing mappings.

#	Challenge type	RDB pattern	Examples of relevant guiding OWL axioms	Specific difficulty
(1)	**Normalization**	Weak entity table (depends on other table, e.g., in a part-of relationship)	owl:Class	JOIN to extract full IDs
(2)		1:n attribute	owl:DatatypeProperty	JOIN to relate attribute with entity ID
(3)		1:n relation	owl:ObjectProperty, owl:InverseFunctionalProperty	JOIN to relate entity IDs
(4)		n:m relation	owl:ObjectProperty	3-way JOIN to relate entity IDs
(5)		Indirect n:m relation (using additional intermediary tables)	owl:ObjectProperty	k-way JOIN to relate entity IDs
(6)	**Denormalization**	Correlated entities (in shared table)	owl:Class	Filter condition
(7)		Multi-value	owl:DatatypeProperty, owl:maxCardinality $[>1]$	Handling of duplicate IDs
(8)	**Class hierarchies**	1:n property match ("Option 2" in Figure 2)	rdfs:subClassOf, owl:unionOf, owl:disjointWith	UNION to assemble redundant properties
(9)		n:1 class match with type column ("Option 1" in Figure 2)	rdfs:subClassOf, owl:unionOf	Filter condition
(10)		n:1 class match without type column ("Option 1" in Figure 2)	rdfs:subClassOf, owl:unionOf	JOIN condition as implicit filter
(11)	**Key conflicts**	Plain composite key	owl:Class, owl:hasKey	Technical handling
(12)		Composite key, n:1 class matching to partial keys	owl:Class, owl:hasKey, rdfs:subClassOf	Choice of correct partial keys
(13)		Missing key (e.g., no UNIQUE constraint on secondary key)	owl:Class, owl:hasKey	Choice of correct non-key attribute as ID
(14)		Missing reference (no foreign key where relevant relation exists)	owl:ObjectProperty, owl:DatatypeProperty	Unconstrained attributes as references
(15)	**Dependency conflicts**	1:n attribute	owl:FunctionalProperty, owl:minCardinality $[>1]$, owl:maxCardinality $[>1]$, owl:cardinality $[>1]$	Misleading guiding axioms; possible restriction violations
(16)		1:n relation	owl:FunctionalProperty, owl:minCardinality $[>1]$, owl:maxCardinality $[>1]$, owl:cardinality $[>1]$	Misleading guiding axioms; possible restriction violations
(17)		n:m relation	owl:FunctionalProperty, owl:InverseFunctionalProperty, owl:minCardinality $[>1]$, owl:maxCardinality $[>1]$, owl:cardinality $[>1]$	Misleading guiding axioms; possible restriction violations

2.3. *Semantic Heterogeneity*

Semantic heterogeneity plays a highly important role for data integration in general. Therefore, we extensively test scenarios that bring significant semantic heterogeneity.

Besides the usual semantic differences between any two conceptual models of the same domain, three ad-

ditional factors apply in relational-to-ontology data integration: (1) the *impedance mismatch* caused by the object-relational gap, i.e., ontologies group information around entities (objects) while relational databases encode them in a series of values that are structured in relations; (2) the impedance mismatch between the closed-world assumption (CWA) in databases and the open-world assumption (OWA) in ontologies; and (3) the difference in semantic expressiveness, i.e., databases may model some concepts or data explicitly where they are derived logically in ontologies. All of them are inherent to all relational-to-ontology mapping problems.

3. Analysis of Mapping Approaches

Different mapping generation systems make different assumptions and implement different approaches. Thus, a benchmark needs to consider each approach appropriately. In the following, we first discuss the major differences regarding the availability of input. For instance, do we only have access to the ontology's T-Box axioms or is also to some additional A-Box facts that could be used as data examples? Afterwards, we discuss the different approaches of implementing mapping processes and discuss the effects for a benchmark, e.g., automatic vs. different forms of semi-automatic processes.

3.1. Differences in Availability and Relevance of Input

Different input may be available to an automatic mapping generator. In relational-to-ontology data integration, the main difference on available input concerns the target ontology. The ontology could be specified entirely and in detail, or it could still be incomplete (or even missing) when mapping construction starts. Moreover, other differences are also related to available input. For instance, data or a query workload could be available in addition to mere schema information on either side.

The case where both the relational database schema and the ontology are completely available could be motivated by different situations. For example, a company may wish to integrate a relational data source into an existing, mature, Semantic Web application. In this case, the target ontology would already be well defined and would also be populated with some A-Box data. In addition, a SPARQL query workload could be known and could be available as additional input to a mapping generator.

On the other side, relational-to-ontology data integration might be motivated by a large-scale industry data integration scenario (e.g., [15,18]). In this scenario, the task at hand is to make complex and confusing schemata easier to understand for experts who write specialized queries. In this case, at the beginning no real ontology is given. At best there might be an initial, incomplete vocabulary. Mappings and ontology are basically being developed simultaneously over time. That is, no complete target ontology is available as input to a mapping generator.

Essentially, the different scenarios can all be distinguished by the following question: which information is available as input, besides the relational database? This can be a mix of an ontology's T-Box (or even just incomplete T-Box), A-Box data and an existing query workload in either SQL or SPARQL. Note, that we always assume that the relational source database is completely accessible (both schema and data), as this is a fundamental requirement without which relational-to-ontology data integration applications cannot reasonably be motivated. Besides the *availability* of input for mapping generation, there could be additional knowledge about which parts of the input are even *relevant*. For instance, it may be clear that only parts of the ontology that are being used by a certain query workload need to be mapped. If so, this information could also be leveraged by the mapping generation system (e.g., by analyzing the query workload).

3.2. Differences in Mapping Process

Other differences can arise from the process in which mapping generation is approached. These can be either fully-automatic approaches or semi-automatic approaches. Truly semi-automatic approaches are usually iterative [25], as they consist of a sequence of mapping generation steps that get interrupted to allow human feedback, corrections, or other input. Their process is driven by the human perspective rather than by an automatic component. Since we want to better adjust our benchmark to the semi-automatic approaches, we first discuss different ways that are known for the semi-automatic case.

Heyvaert et al. [21] have recently identified four different ways for manual relational-to-ontology mapping creation. Each of those directions inflicts a different interaction paradigm between the system and the user and thus solicits different forms of human input: users can edit mappings based on either the source or target definitions, they can drive the process by providing result examples or could theoretically even edit mappings irrespective of either the source or target in an abstract fashion. Moreover, while some approaches consider man-

ual corrections only at the end of the mapping process, more thoroughly semi-automatic approaches allow or even require such input during the process. Some of us have also earlier identified two fundamentally different user perspectives on mapping generation [39] that drive the process in a different order depending on whether the user feels more at home with the source database or with the target ontology.

In terms of their potential evaluation, iterative approaches of this kind must be considered according to two additional characteristics: First, whether iterative human input is mandatory or generally optional. Second, whether input is only used to improve the mapping as such, or if the systems also exploit it as feedback for their next automated iteration. Systems that solicit input only optionally and do not use it as feedback can be evaluated like non-iterative systems on a fully automatic baseline without limitations. Systems with only optional input that *do* learn from the feedback (if provided), can still be evaluated on the same baseline but may not demonstrate their full potential. Where input is mandatory, systems need to be either steered by an actual human user or at least require simulated human input produced by an oracle.

Next, the kind of human input that a system can process makes a difference for evaluation settings. Most semi-automatic systems either provide suggestions that users can confirm or delete, or they allow users to manually adjust the mapping. An alternative approach is *mapping by example*, where users provide expected results. In addition, however, some systems may require complex or indirect interactions, or simply resort to more unusual forms of input that cannot easily be foreseen.

All the differences discussed before have an impact on how mapping generation systems need to be evaluated. Each mapping generation system is usually tied to one specific approach and does not allow for much freedom. We therefore decided that an end-to-end evaluation that allows the use of different types of input is best. Since semi-automatic approaches are becoming more and more relevant, we decided to support them using an automated oracle that simulates user input where possible.

4. *RODI* Benchmark Suite

In the following, we present the details of our *RODI* benchmark: we first give an overview, then we discuss the data sets (relational schemata and ontologies) that can be used, as well as the queries. Finally, we present our scoring function to evaluate the benchmark results.

4.1. Overview

Figure 3 gives an overview of the scenarios used in our benchmark. The benchmark ships with data sets from three different application domains: conferences, geodata and the oil & gas exploration domain. In its basic mode of operation, the benchmark provides one or more target ontologies for each of those domains (T-Box only) together with relational source databases for each ontology (schema and data). For some of the ontologies there are different variants of accompanying relational schemata that systematically vary the types of targeted mapping challenges.

The benchmark asks systems to create mapping rules from the different source databases to their corresponding target ontologies. We call each such combination of a database and an ontology a *benchmark scenario*. For evaluation, we provide query pairs for each scenario to test a range of mapping challenges. Query pairs are evaluated against the instantiated ontology and the provided databases, respectively. Results are compared for each query pair and aggregated in the light of different mapping challenges using our scoring function.

While challenges that result from different naming or semantic heterogeneity are mostly covered by complete scenarios, we target structural challenges on a more fine-granular level of individual query tests with a dedicated score. Table 2 again lists individual structural challenges and our coverage by dedicated tests.

Multi-source integration can be tested as a sequence of different scenarios that share the same target ontology. We include specialized scenarios for such testing with the conference domain.

In order to be open for other data sets and different domains, our benchmark can be easily extended to include scenarios with real-world ontologies and databases. In our initial version, we already provide one such extension from a real-world application of the oil and gas domain.

4.2. Data Sources and Scenarios

In the following, we present the data sources (i.e., ontologies and relational schemata) as well as the combinations used as integration scenarios for the benchmark in more details. *RODI* ships with scenarios based on data sources from three different application domains.

4.3. Conference Scenarios

As our primary domain for testing, we chose the conference domain: it is well understood, comprehensible

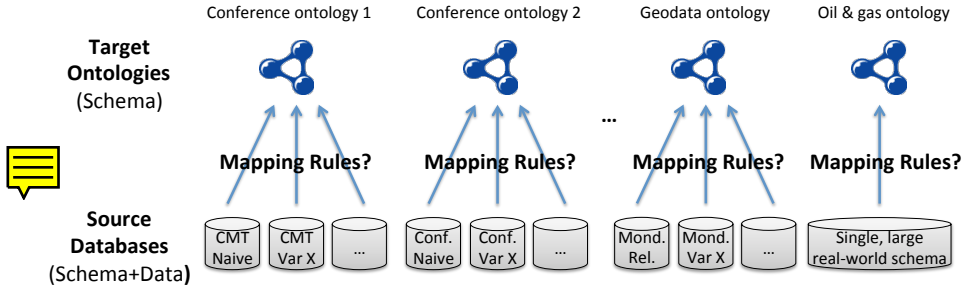


Figure 3. Overview of *RODI* benchmark scenarios

Table 2

Coverage of structural challenges in default benchmark scenarios. Challenges marked with a check are tested throughout majority of scenarios. 'Single scenario' marks challenges that could only be tested in a dedicated scenario. For dependency conflicts, we test only part of the challenge (misleading axioms), but no restriction violations.

#	Challenge type	RDB pattern	Examples of relevant guiding OWL axioms	Covered
(1)	**Normalization**	Weak entity	owl:Class	✓
(2)		1:n attribute	owl:DatatypeProperty	✓
(3)		1:n relation	owl:ObjectProperty, owl:InverseFunctionalProperty	✓
(4)		n:m relation	owl:ObjectProperty	✓
(5)		Indirect n:m relation	owl:ObjectProperty	✓
(6)	**Denormalization**	Correlated entities	owl:Class	✓
(7)		Multi-value	owl:DatatypeProperty, owl:maxCardinality [>1]	✓
(8)	**Class hierarchies**	1:n property match	rdfs:subClassOf, owl:unionOf, owl:disjointWith	✓
(9)		n:1 class match with type column	rdfs:subClassOf, owl:unionOf	✓
(10)		n:1 class match without type column	rdfs:subClassOf, owl:unionOf	✓
(11)	**Key conflicts**	Plain composite key	owl:Class, owl:hasKey	✓
(12)		Composite key, partial matching	owl:Class, owl:hasKey, rdfs:subClassOf	✓
(13)		Missing key	owl:Class, owl:hasKey	Single scenario
(14)		Missing reference	owl:ObjectProperty, owl:DatatypeProperty	Single scenario
(15)	**Dependency conflicts**	1:n attribute	owl:FunctionalProperty, owl:minCardinality [>1], owl:maxCardinality [>1], owl:cardinality [>1]	Only misleading axioms
(16)		1:n relation	owl:FunctionalProperty, owl:minCardinality [>1], owl:maxCardinality [>1], owl:cardinality [>1]	Only misleading axioms
(17)		n:m relation	owl:FunctionalProperty, owl:InverseFunctionalProperty, owl:minCardinality [>1], owl:maxCardinality [>1], owl:cardinality [>1]	Only misleading axioms

even for non-domain experts but still complex enough for realistic testing and it has been successfully used as the domain of choice in other benchmarks before (e.g., [52,24])

4.3.1. Ontologies

The conference ontologies in this benchmark are provided by the Ontology Alignment Evaluation Initiative (OAEI) [52,24] and were originally developed by the OntoFarm project [20]. We selected three particular ontologies (CMT, SIGKDD, CONFERENCE), based on a number of criteria: variation in size, the presence of functional coherences, the coverage of the domain, variations in modeling style, and the expressive power of the ontology language used. Different modeling styles result from the fact that each ontology was modeled by different people based on various views on the domain, e.g., they modeled it according to an existing conference management tool, expert insider knowledge, or according to a conference website. To cover our mapping challenges (Section 2), we selectively modified the ontologies (e.g., we added labels to add interesting lexical matching challenges). In SIGKDD, we have fixed a total of seven inconsistencies that we discovered in this ontology as follows: (1) we selectively added annotations like labels and comments, as these can help to identify correspondences lexically; (2) we added a few additional datatype properties where they were scarce, as they test other mapping challenges than just classes and object properties; and (3), we fixed a total of seven

inconsistencies that we discovered in SIGKDD when adding A-Box facts (e.g., each place with a zip code automatically became a sponsor, who were modeled as a sub class of person).

4.3.2. Relational Schemata

We synthetically derived different relational schemata for each of the ontologies, focusing on different mapping challenges. We provide benchmark scenarios as combinations of those derived schemata with either their ontologies of origin, or, for more advanced testing, paired with any of the other ontologies. First, for each ontology we derived a relational schema that can be mapped to the ontology using a naive mapping as described in [29]. The algorithm works by deriving an entity-relationship (ER) model from an OWL DL ontology. It then translates this ER model into a relational schema according to text book rules (e.g., [27]). For this paper, we extended this algorithm to cover the full range of expected relational design patterns. In particular, the previous version did cover only one out of the above-mentioned three design patterns to translate class hierarchies into relational tables. Additionally, we extended this algorithm to consider ontology instance data to derive more proper functionalities (rather than just looking at the T-Box as the previous algorithms do). Otherwise, the generated naive relational schemata would have contained an unrealistically high number of n:m-relationship tables. The naively translated schemata of the algorithm are guaranteed to be in fourth normal form (4NF), fulfilling normalization requirements of standard design practices. Thus, the naive schemata already include various normalization artifacts as mapping challenges.

From each naively translated schema, we systematically created different variants by introducing different aspects on how a real-world schema may differ from a naive translation and thus to test different mapping challenges:

1. *Adjusted Naming:* As described in Section 2.1, ontology designers typically consider other naming schemes than database architects do, even when implementing the same (verbal) specification. Those differences include longer vs. shorter names, “speaking” prefixes, human-readable property IRIs vs. technical abbreviations (e.g., “hasRole” vs. "RID"), camel case vs. underscore tokenization, preferred use of singular vs. plural, and others. For each naively translated schema we automatically generate a variant with identifier names changed accordingly.
2. *Restructured Hierarchies:* The most critical structural challenge in terms of difficulty comes with different relational design patterns to model class hierarchies more or less implicitly. As we have discussed in Section 2.2, these changes introduce significant structural dissimilarities between source and target. We automatically derive variants of all naively translated schemata where different hierarchy design patterns are presented. The choice of design pattern in each case is algorithmically determined on a “best fit” approach considering the number of specific and shared (inherited) attributes for each of the classes.
3. *Combined Case:* In the real world, both of the previous cases (i.e., adjusted naming and hierarchies) would usually apply at the same time. To find out how tools cope with such a situation, we also built scenarios where both are combined.
4. *Removing Foreign Keys:* Although it is considered as bad style, databases without foreign keys are not uncommon in real-world applications. This can be a result from either lazy design or come with legacy applications (e.g., one popular open source DBMS introduced plugin-free support for foreign keys less than five years ago). The mapping challenge is that mapping tools must find the join paths to connect tables of different entities. Additionally, they sometimes even need to guess a join path for reading attributes of the same entity if its data is split over several tables as a consequence of normalization. Therefore, we have created one dedicated scenario to test this challenge with the CONFERENCE ontology and based it on the schema variant with restructured hierarchies.
5. *Partial Denormalization:* In many cases, schemata get partially denormalized to optimize for a certain read-mostly workload. Denormalization essentially means that correlated (yet separated) information is jointly stored in the same table and partially redundant. We provide one such scenario for the CMT ontology. As denormalization requires conscious design choices, this schema is the only one that we had to hand-craft. It is based on the variant with restructured hierarchies.

In some cases, data transformations may also be required for a mapping to work fully as expected. A significant number of fundamentally different transformation types needs to be considered, each adding complexity in a different way. These comprise translations between different representations of date and time (e.g., a dedicated date type versus Epoch time stamps), simple numeric unit transformations (e.g., MB vs. GB),

Table 3
Basic scenario variants

	CMT	CONFERENCE	SIGKDD
Naive	(✓)	(✓)	(✓)
Adjusted Naming	✓	✓	✓
Restructured Hierarchies	✓	✓	✓
Combined Case	(✓)	(✓)	✓
Missing FKs	-	✓	-
Denormalized	✓	-	-

unit transformations requiring more complex formulae (e.g., degrees Celsius vs. Fahrenheit), string based data cleansing (e.g., removing trailing white space), string compositions (e.g., concatenating a first and last name), more complex string modifications (e.g., breaking up a string based on a learned regular expression), table based name translations (e.g., replacing names using a thesaurus), noise removal (e.g., ignoring erroneous tuples), etc.

While our extension mechanism (see Section 4.6) is suited to even add dedicated scenarios for testing such conversions we excluded them from our default benchmark for mere practical reasons: (1) To the best of our knowledge no current relational-to-ontology mapping generation system implements such transformation functionality to date, so there is little practical use for benchmarking it. And (2), not all of the different transformation types typically co-occur in the same application domain and it would be hard to incorporate them into our conference domain scenario in appropriate variety without making the scenario less realistic.

4.3.3. Integration Scenarios

For each of our three main ontologies, CMT, CONFERENCE, and SIGKDD, the benchmark includes five scenarios, each with a different variant of the database schema (discussed before). Table 3 lists the different versions.

As discussed before, *Naive* closely mimics the structure of the original ontology, but the schemata are normalized and thus the scenario contains the challenge of normalization artifacts. *Adjusted Naming* adds the naming conflicts as discussed before. *Restructured hierarchies* tests the critical structural challenge of different relational patterns to model class hierarchies, which, among others, subsumes the challenge to correctly build n:1 mappings between classes and tables. In the *Combined Case*, renamed, restructured hierarchies are employed and their effects are tested in combination. This is a more advanced test case. A special challenge arises from databases with no (or few) foreign key constraints (*Missing FKs*). In such a scenario, mapping tools must guess the join paths to connect tables that correspond to different entity types. The technical mapping challenge arising from *Denormalized* schemata consists in identifying the correct *partial* key for each of those correlated entities, and to identify which attributes and relations belong to which of the types.

To keep the number of scenarios small for the default setup, we differentiate between default scenarios and non-default scenarios. We excluded scenarios with the most trivial schema versions. In addition, we did limit the number of combinations for the most complex schema versions by including only one of each type as a default scenario. While the default scenarios are mandatory to cover all mapping challenges, the non-default scenarios are optional (i.e., users could decide to run them in order to gain additional insights). Non-default scenarios are put in parentheses in Table 3. However, they are not supposed to be executed in a default run of the benchmark.

Similarly, we include scenarios that require mappings of schemata to one of the other ontologies (e.g., mapping a CMT database schema variant to the SIGKDD ontology). They represent more advanced data integration scenarios and are part of default scenarios.

4.3.4. Data

We provide data to fill both the databases and ontologies. Conference ontologies are origially provided as T-Boxes, only, i.e., no A-Box. We first generate data as A-Box facts for the different ontologies, and then translate them into the corresponding relational data. Transformation of data follows the same process as translating the T-Box. For evaluation, data is only needed in the relational databases, so generating ontology A-Boxes would not even be necessary. However, this procedure simplifies data generation since all databases can be automatically derived from the given ontologies as described before. Our conference data generator deterministically produces a scalable amount of synthetic facts around key concepts in the ontologies, such as conferences, papers, authors, reviewers, and others. In total, we generate data for 23 classes, 66 object properties (including inverse properties) and 11 datatype properties (some of which apply to several classes). However, not all of those concepts and properties are supported by every ontology. For each ontology, we only generate facts for the subset of classes and properties that have an equivalent in the relational schema in question.

4.3.5. Queries

We test each integration scenario with a series of *query pairs*, consisting of semantically equivalent

queries against the instantiated ontology and the provided databases, respectively.

Query pairs are manually curated and designed to test different mapping challenges. To this end, all query pairs are tagged with categories, relating them to different mapping challenges. All scenarios draw on the same pool of 56 query pairs, accordingly translated for each ontology and schema. However, the same query may face different challenges in different scenarios, e.g., a simple 1:1 mapping between a class and table in a naive scenario can turn into a complicated n:1 mapping problem in a scenario with restructured hierarchies. Also, not all query pairs are applicable on all ontologies (and thus, on their derived schemata).

Query pairs are grouped into three basic categories to test the correct mapping of *class instances*, instantiations of *datatype properties* and *object properties*, respectively. Additional categories relate queries to n:1 and n:m mapping problems or prolonged property join paths resulting from normalization artifacts. A specific category exists for the de-normalization challenge.

4.4. Geodata Domain – Mondial Scenarios

As a second application domain, *RODI* ships scenarios in the domain of geographical data.

The Mondial database is a manually curated database containing information about countries, cities, organizations, as well as about geographic features such as waters (with subclasses lakes, rivers, and seas), mountains, and islands. It has been designed as a medium-sized case study for several scientific aspects and data models [35].

Based on Mondial, we have developed a number of benchmark scenarios. First, there is a scenario based on the original relational database, which features a wide range of relational modeling patterns, and the Mondial OWL ontology. In addition, we have added a series of further scenarios with synthetically modified variants of the database to focus on the effect of specific different relational modeling patterns. This is similar to the different variants produced in the conference domain. To keep the number of tested scenarios at bay, we do not consider those additional synthetic variants as part of the default benchmark. Instead, we recommend to only test the main Mondial scenario with others being available as optional tests to dig deeper into specific behavioral patterns in this domain.

In all scenarios, we use a query workload that mainly approximates real-world explorative queries on the data, although limited to queries of low or medium complexity. Still, those queries typically co-relate more than one concept or require several attributes to be correctly mapped at the same time in order to return any correct results. The degree of difficulty in Mondial scenarios is therefore generally higher than the one of our scenarios in the conference domain.

4.5. Oil & Gas Domain – NPD FactPages Scenarios

Finally, we include an example of an actual real-world database and ontology, in the oil and gas domain: The Norwegian Petroleum Directorate (NPD) FactPages [48]. Our test set contains a small relational database (≈40 MB), but with a relatively complex structure (70 tables, ≈1000 columns and ≈100 foreign keys), and an ontology covering the domain of the database. The database is constructed from a publicly available dataset containing reference data about past and ongoing activities in the Norwegian petroleum industry, such as oil and gas production and exploration. The corresponding ontology contains ≈300 classes and ≈350 properties.

With this pair of a database and ontology, we have constructed two scenarios that feature a different series of tests on the data: first, there are queries that are built from information needs collected from real users of the FactPages and cover large parts of the dataset. Those queries are highly complex compared to the ones in other scenarios and require a significant number of schema elements to be correctly mapped at the same time to bear any results. We have collected 17 such queries in scenario *npd_user_tests*. In addition, we have generated a large number of small, atomic query tests for baseline testing. These are similar to the ones used with the conference domain, i.e., they test for individual classes or properties to be correctly mapped. A total of 439 such queries have been compiled in scenario *npd_atomic_tests* to cover all of the non-empty fields in our sample database.

A specific feature resulting from the structure of the FactPages database and ontology are a high number of 1:n matches, i.e., concepts or properties in the ontology that require a UNION over several relations to return complete results. 1:n matches as a structural feature can therefore best be tested in the *npd_atomic_tests* scenario.

4.6. Extension Scenarios

Our benchmark suite is designed to be extensible, i.e., additional scenarios can be easily added. The primary aim of supporting such extensions is to allow domain-specific, real-world mapping challenges to be tested alongside our more default scenarios. Extension

scenarios can be added by users of our benchmark without any programming efforts and creating and adding scenarios is described in the user documentation of the *RODI* benchmark suite.

4.7. Evaluation Criteria – Scoring Function

It is our aim to measure the practical usefulness of mappings. We are therefore interested in the utility of query results, rather than comparing mappings directly to a reference mapping set or than measuring precision and recall on all elements of the schemata. This is important because a number of different mappings might effectively produce the same data w.r.t. a specific input database. Also, the mere number of facts is no indicator of their semantic importance for answering queries (e.g., the overall number of conferences is much smaller than the number of paper submission dates, yet are at least as important in a query about the same papers). In addition, in many cases only a subset of the information is relevant in practice and we define our queries on a meaningful subset of information needs.

We therefore observe a score that reflects utility of the mappings with relation to our query tests as our main measure. Intuitively, this score reports the percentage of successful queries for each scenario.

However, in a number of cases, queries may return correct but incomplete results, or could return a mix of correct and incorrect results. In these cases, we consider per-query accuracy by means of a local per-query F-measure. Technically, our reported overall score for each scenario is the average of F-measures for each query test, rather than a simple percentile of successful queries. To calculate these per-query F-measures, we also need to consider query results that contain IRIs.

Apparently, different mapping generators will generate different IRIs for the same entities, e.g., by choosing different prefixes. F-measures for query results containing IRIs are therefore w.r.t. the degree to which they satisfy *structural equivalence* with a reference result. For practical reasons, we use query results on the original, underlying SQL databases as technical reference during evaluation. Structural equivalence effectively means that if same-as links were established appropriately, then both results would be semantically identical. For a formal definition of structural result equivalence, please refer to our initial *RODI* paper [7].

Table 4 shows an example with a query test that asks for the names of all authors. Result set A is structurally equivalent to the reference result set, i.e., it has found all authors and did not return anything else, so both precision and recall are 1.0. Result set B is equivalent

Table 4
Example results from a query pair asking for author names (e.g., SQL: SELECT name FROM persons WHERE person_type=2; SPARQL: SELECT ?name WHERE ?p a :Author; foaf:name ?name)

(a) Result A (equals reference result)

Jane
John

(b) Result B

John

(c) Result C

Jane
John
James

with only a subset of the reference result (e.g., it did not include those authors who are also reviewers). Here, precision is still 1.0, but recall is only 0.5. In case of result set C, all expected authors are included, but also another person, James. Here, precision is 0.66 but recall is 1.0.

To aggregate results of individual query pairs, a scoring function calculates the averages of per query numbers for each scenario and for each challenge category. For instance, we calculate averages of all queries testing $1{:}n$ mappings. Thus, for each scenario there is a number of scores that rate performance on different technical challenges. Also, the benchmark can log detailed per-query output for debugging purposes.

4.8. System Requirements

With *RODI*, we can test mapping generators that work in either one or two stages: that is, they either directly map data from the relational source database to the target ontology in one stage (e.g., [40,10]). Or, they bootstrap their own ontology, which they use as an intermediate mapping target. In this case, to get to the full end-to-end mappings that we can test, the intermediate ontology and the actual target ontology should be integrated via ontology alignment in a second stage. Two-stage systems may either include a dedicated ontology alignment stage (e.g., [14]) or they deliver the first (intermediary) stage only ([23,3]). In the latter case, *RODI* can step in to fill the missing second stage with a standard ontology alignment setup [47].

Our tests check the accuracy of SPARQL query results. Queries ask for individuals of a certain type (or their aggregates), properties correlating them, associated values and combinations thereof, sometimes also using additional SPARQL language features such as filters to narrow down the result set. This means that mapped data will be deemed correct if it contains correct RDF triples for all tested cases. For entities, this means that systems need to construct one correctly typed IRI for each entity of a certain type. For object

properties, they need to construct triples to correctly relate those typed IRIs, and for datatype properties, they need to assign the correct literal values to each of the entity IRIs using the right predicates. Systems do therefore not strictly need to understand or to produce any OWL axioms in the target ontology. However, our target ontologies are in OWL 2, using different degrees of expressiveness. Axioms in the target ontology can be important as *guidance* to identify suitable correspondences for one-stage systems. Similarly, if two-stage systems *construct* expressive axioms in their intermediate ontology, this may guide the second stage of ontology alignment. For instance, if a predicate is known to be an object property in the target ontology, results will suffer if a mapping generation tool assigns literal values using this property. Also, if a property is known to be functional it might be a better match for a n:1 relation than a non-functional property would be.

5. Framework Implementation

In this section, we discuss some implementation details in order to guide researchers and practitioners to include their system in our benchmarking suite.

5.1. Architecture of the Benchmarking Suite

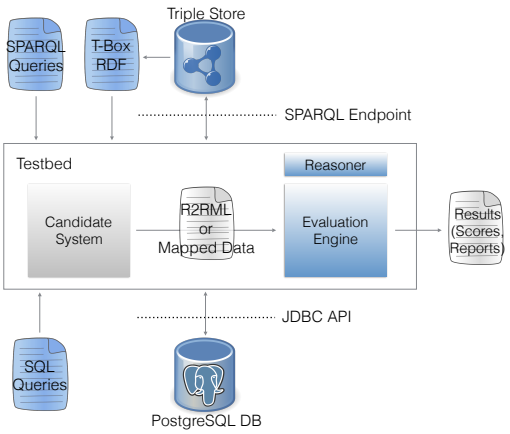


Figure 4. *RODI* framework architecture

Figure 4 depicts the overall architecture of our benchmarking suite. The framework requires upfront initialization *per scenario*. Artifacts generated or provided during initialization are depicted blue in the figure. After initialization, a mapping tool can access the database (directly or via the framework's API) and target ontology (via the Sesame API or using SPARQL, or serialized as an RDF file). Finally, it submits generated R2RML[2] mappings in a special folder on the file system, so evaluation can be triggered. As an alternative, mapping tools could also execute mappings themselves and submit final mapped data instead of R2RML. This would be the preferred procedure for tools that do not support R2RML but other mapping languages. More generally, mapping tools that cannot comply with the assisted benchmark workflow can always trigger individual aspects of initialization of evaluation separately.

5.2. Details on the Evaluation Phase

Unless a mapping system under evaluation decides to skip individual steps, i.e., to implement them independently, in the evaluation phase, the benchmark suite will: (1) read submitted R2RML mappings and execute them on the database, (2) materialize the resulting A-Box facts in a Sesame repository together with the target ontology (T-Box), (3) optionally apply reasoning through an external OWL API [30] compatible reasoner to infer additional facts that may be requested for evaluation, (4) evaluate all query pairs of the scenario on the repository and relational database, and (5) produce a detailed evaluation report.

We evaluate query results as described in Section 4.7 by attempting to construct an isomorphism ϕ to transform query result sets into reference results. Technically, we use the results of the SQL queries from query pairs to calculate the reference result set. For each SQL query in a query pair, we flag attributes that together serve as keys, so keys can be matched with IRIs rather than with literal values. Obviously, keys and IRIs need to match only on the count of being the same unique value wherever they appear, while literal values need to be exact matches.

For constructing ϕ, we first index all individual IRIs (i.e., IRIs that identify instances of some class) in the query result. Next, we build a corresponding index for keys in the reference set. For both sets we determine binding dependencies across tuples (i.e., re-occurrences of the same IRI or key in different tuples). As a next step, we narrow down match candidates to tuples where all corresponding literal values are exact matches. Finally, we match complete result tuples with reference tuples, i.e., we also check for viable correspondences between keys and IRIs. As discussed, the criterion for a viable match between a key and an IRI is that for each occurence of this particular key and of this particular

[2] http://www.w3.org/TR/r2rml/

IRI in any of the tuples, both need to be matched with the same partner. This last step corresponds to identifying a maximal common subgraph (MCS) between the dependency graphs of tuples on both sides, i.e., it corresponds to the MCS-isomorphism problem. For efficiency reasons, we approximate the MCS if dependency graphs contain transitive dependencies, breaking them down to fully connected subgraphs. However, it is usually possible to formulate query results to not contain any such transitive dependencies by avoiding inter-dependent IRIs in SPARQL SELECT results in favor of a set of significant literals describing them. All queries shipped with this benchmark are free of transitive dependencies, hence the algorithm is accurate for all delivered scenarios.

Finally, we count tuples that could not be matched in the result and reference set, respectively. Precision is then calculated as $\frac{|res|-|unmatched(res)|}{|res|}$ and recall as $\frac{|ref|-|unmatched(ref)|}{|ref|}$. Aggregated numbers are calculated per query pair category as the averages of precision and recall of all queries in each category.

6. Benchmark Results

6.1. Evaluated Systems

We have performed an in-depth analysis using *RODI* on a wide range of systems. Those include current contenders in the automatic segment (*BootOX* [14,18,17] and *IncMap* [40,41]), more general-purpose mapping generators that we combine with ontology alignment to measure in the benchmark (*-ontop-* [23] and *MIRROR* [3]), as well as a much earlier, yet state-of-the-art system in inter-model matching (*COMA++* [10]). In a specialized semi-automatic series of experiments, we are also evaluating Karma [8,51], which does not support a fully automatic mapping generation mode and works with a much sophisticated model of human intervention. As a consequence, it requires a specific experimental setup.

1. *BootOX* (**B.OX**) is based on the approach called *direct mapping* by the W3C:[3] every table in the database (except for those representing n:m relationships) is mapped to one class in the ontology; every data attribute is mapped to one data property; and every foreign key to one object property. Explicit and implicit database constraints from the schema are also used to enrich the bootstrapped ontology with axioms about the classes and properties from these direct mappings. Afterwards, BootOX performs an alignment with the target ontology using the LogMap system [33,13,49].
2. *IncMap* (**IncM.**) maps an available ontology directly to the relational schema. IncMap represents both the ontology and schema uniformly, using a structure-preserving meta-graph for both. It runs in two phases, using lexical and structural matching. We evaluate a current (and yet unpublished) work-in-progress version of IncMap, as opposed to the initial version previously evaluated in [7]. The main difference between the two versions of IncMap are improvements in lexical matching and mapping selection, as well as engineering improvements that add to mapping quality.
3. *MIRROR* (**MIRR.**) is a tool for generating an ontology and R2RML direct mappings automatically from an RDB schema. MIRROR has been implemented as a module of the RDB2RDF engine morph-RDB [43]. Their output is oblivious of the required target ontology, though, so we perform post-processing with the ontology alignment tool LogMap [33].
4. The *-ontop- Protege Plugin* (**ontop**) is a mapping generator developed for -ontop- [23]. -ontop- is a full-fledged query rewriting system [44] with limited ontology and mapping bootstrapping capabilities. Just as with MIRROR, we need to post-process results with ontology alignment.
5. *COMA++* (**COMA**) has been a contender in the field of schema matching for several years already; it is still widely considered state of the art. In contrast to other systems from the same era, COMA++ is built explicitly also for inter-model matching. To evaluate the system, we had to perform a translation of its output into modern R2RML.
6. *Karma* is one of the most prominent modern relational-to-ontology mapping generation systems. It is strictly semi-automatic, i.e., there is no fully automatic baseline that we could use for non-interactive evaluation. In addition, Karma's mode of iterations is designed to take advantage mostly from integrating a series of data sources to the same target ontology. Karma is therefore not well suited for single-scenario evaluations. We therefore only evaluate Karma in a dedicated line of experiments that suit its specifications.

6.2. Experimental Setup

We conduct benchmark default experiments as described in Section 4 for all systems except Karma. This

[3] http://www.w3.org/TR/rdb-direct-mapping/

Table 5
Overall scores in default scenarios (scores based on average of per-test F-measure). Best numbers per scenario in bold print.

Scenario	B.OX	IncM.	ontop	MIRR.	COMA
	Conference domain, adjusted naming				
CMT	**0.76**	0.45	0.28	0.28	0.48
Conference	0.51	**0.53**	0.26	0.27	0.36
SIGKDD	**0.86**	0.76	0.38	0.30	0.66
	Conference domain, restructured				
CMT	0.41	**0.44**	0.14	0.17	0.38
Conference	**0.41**	**0.41**	0.13	0.23	0.31
SIGKDD	**0.52**	0.38	0.21	0.11	0.41
	Conference domain, combined case				
SIGKDD	**0.48**	0.38	0.21	0.11	0.28
	Conference domain, missing FKs				
Conference	0.33	**0.41**	-	0.17	0.21
	Conference domain, denormalized				
CMT	**0.44**	0.40	0.20	0.22	-
	Geodata				
Classic Rel.	**0.13**	0.08	-	-	-
	Oil & gas domain				
User Queries	0.00	0.00	0.00	0.00	-
Atomic	**0.14**	0.12	0.10	0.00	0.02

includes a selection of nine prototypical scenarios from the conference domain, one from the geodata domain and two from the oil & gas domain, as well as six different cross-matching scenarios. For all of these main experiments, we observe and report overall *RODI* scores as well as specific selected scores in individual categories.

In addition, we perform two different semi-automatic experiments on selected scenarios for Karma and IncMap, respectively. For Karma, we had to conduct experiments with an actual human in the loop to perform steps that Karma could not automate. With IncMap, we could simulate human feedback by responding to suggestions by taking a response from the benchmark that indicates changes in mapping quality. In both semi-automatic cases, we chiefly observe the number of interactions.

6.3. Default Scenarios: Overall Results

Table 5 shows scores for all systems on all basic default scenarios. At first impression we can observe that all tested systems manage to solve some parts of the scenarios, but with declining success as scenario complexity increases.

For instance, relational schemata in the conference adjusted naming scenarios follow modeling patterns from their corresponding ontologies most closely, and all systems without exception perform best in this part of the experiments. Quality drops for all other types of scenarios, i.e., whenever we introduce additional challenges that are specific to the relational-to-ontology modeling gap. The drop in accuracy between *Adjusted Names* and *Restructured hierarchies* settings is mostly due to the n:1 mapping challenge introduced by one of the relational patterns to represent class hierarchies which groups data for several subclasses in a single table. In the most advanced conference cases, systems lose further due to the additional challenges, although to different degrees. Good news is that some of the most actively developed current systems, BootOX and IncMap, could improve their scores compared to previous numbers recorded in January 2015 [7]. A somewhat disappointing general observation, however, is that measured quality is overall still modest compared to results that are known from ontology alignment tasks involving some of the same ontologies (c.f. [52,24]). This is disappointing, especially while state-of-the-art ontology alignment software is employed in some of the systems. It could indicate that the specific challenges in relational-to-ontology mapping generation can not convincingly be solved with the same technology that is successful in ontology alignment, but may call for more specialized approaches.

While all of the conference scenarios test a wide range of specific relational-to-ontology mapping challenges, they do so in a highly controlled fashion, on schemata with at best medium size and complexity, and using a largely simplified query workload. For instance, queries in the conference domain scenarios would separately check for mappings of authors, person names, and papers. They would not, however, pose any queries like asking for the names of authors who did participate in at least five different papers. The huge difference here is that, if two out of three of these elements were mapped correctly, the simple, atomic queries would report an average score of 0.66, while the single, more application-like query that correlates the same elements would not retrieve anything, thus resulting in a score of 0.00. None of the systems managed to solve even a single test on this challenge. This kind of real-world queries that mimick an actual application query workload, are precisely what we focus on in the remaining three default scenarios, which are set in the geodata and oil & gas exploration domains. Consequently, scores are lower again in those scenarios. In the geodata scenario, only a minority of query tests could be solved. Detailed

Table 6
Overall scores in cross-matching scenarios (scores based on average of per-test F-measure). Best numbers per scenario in bold print.

Source	B.OX	IncM.	ontop	MIRR.	COMA
	Target ontology: CMT				
Conference	0.20	**0.35**	0.10	0.00	0.00
SIGKDD	**0.33**	**0.33**	0.19	0.00	0.14
	Target ontology: Conference				
CMT	0.20	**0.34**	0.05	0.00	0.05
SIGKDD	0.13	**0.30**	0.09	0.00	0.04
	Target ontology: SIGKDD				
CMT	0.46	**0.51**	0.19	0.00	0.24
Conference	0.22	**0.44**	0.13	0.00	0.09

debugging did show the reason for this to be exactly in the nature of queries, most of which go beyond returning simple results of just a single mapped element. In the oil & gas case, the situation becomes even more problematic. Here, the schema and ontology are again a bit more complex than in the geodata scenario, and so is the explorative query workload ("user queries"). None of the systems was able to answer any of these queries correctly after a round of automatic mapping. To retrieve meaningful results, we added a second scenario on the same data, but with a synthetic query workload of atomic queries ("atomic"). On this scenario, results could be computed but overall scores remain low due to the size and complexity of the schema and ontology with a large search space as well as many 1:n matches.

Table 6 showcases results from the most advanced scenarios in the conference domain. All of them are built on the "combined case" scenarios, i.e., they contain a mix of all of the standard relational-to-ontology mapping challenges except for denormalization and lazy modeling of constraints. In addition, they increase the level of semantic heterogeneity by asking for mappings between a schema derived from one ontology to a completely different and independent ontology in the same domain. Scores are generally lower than in the basic conference cases discussed above. Reasonable scores can still be achieved by some systems. Also, the overall trend of performance between the systems mostly remains the same as in the basic scenarios, with a few exceptions. Somewhat surprisingly, COMA loses out more than other contenders. Even more surprising, the performance of BootOX is noticeably low compared to the baseline results from basic scenarios in Table 5. This is unexpected as BootOX essentially applies ontology alignment technology that has proven itself in tasks with high semantic heterogeneity [33]. It could, again, be an indicator that out-of-the-box ontology alignment techniques could not take the same leverage that they do when aligning original ontologies.

The big picture shows that the two most specialized and actively developed systems, BootOX and IncMap, are leading the field. Among those two, BootOX is at a clear advantage in scenarios where the inter-model gap between relational schema and ontology is small (e.g., "adjusted naming"). IncMap is gaining ground when more specific inter-model mapping challenges are added. MIRROR and -ontop- generally show weaker results. It has to be noted, though, that both of these systems have been originally designed and optimized for a somewhat different task than the full end-to-end mapping generation setup tested with *RODI*. Both of these systems also fail to execute some of the scenarios due to technical difficulties. For MIRROR in particular, we have encountered a number of so far unresolved difficulties that may also have a detrimental effect on MIRROR scores. COMA keeps up well, given that it is no longer actively developed and improved. Also, while COMA has been constructed to support inter-model matching in general, it has not been explicitly optimized for the specific case of relational-to-ontology matching.

As part of our detailed analysis of the results we could also identify, and partially even fix, a number of technical shortcomings in tested systems. For instance, we encountered issues with MIRROR in certain multi-schema matching cases on PostgreSQL and did implement a solution in exchange with the authors of the system. In another example, IncMap's poor performance in the geodata scenario could in part be explained by its failure to understand the specification of property domains and ranges as a union of several concrete classes. This pattern did lead IncMap to skipping such properties altogether. While not yet fixed, the observation points to concrete technical improvements in IncMap. In BootOX, incomplete and unfavorable reasoning settings were detected and fixed.

6.4. Default Scenarios: Drill-down

All systems struggle with correctly identifying properties as Table 7 shows. A further drill-down shows that this is in part due to the challenge of normalization artifacts, with systems struggling to detect any properties that map to multi-hop join paths in the tables. Mapping data to class types appears to be generally easier for all contenders. BootOX is performing best in most cases with all kinds of properties, with IncMap coming in second. This represents a change over the previous versions of both systems benchmarked earlier this year, where IncMap was clearly leading on properties [7].

Table 7
Score break-down for queries on different match types with adjusted naming conference scenarios. 'C' stands for queries on classes, 'D' for data properties, 'O' for object properties.

Scenario	B.OX			IncM.			ontop			MIRR.			COMA		
	C	D	O	C	D	O	C	D	O	C	D	O	C	D	O
CMT	**0.92**	**0.73**	**0.50**	0.58	0.46	0.17	0.67	0.00	0.00	0.56	0.00	0.00	0.75	0.46	0.00
Conference	**0.81**	0.27	**0.38**	**0.81**	**0.53**	0.13	0.63	0.00	0.00	0.53	0.00	0.00	0.50	0.40	0.00
SIGKDD	**1.00**	**0.90**	**0.25**	0.80	0.70	**0.25**	0.73	0.00	0.00	0.46	0.00	0.00	0.80	0.70	0.00

Table 8
Score break-down for queries that test n:1 matches in restructured conference domain scenarios. 1:1 and n:1 stands for queries involving 1:1 or n:1 mappings among classes and tables, respectively.

Scenario	B.OX		IncM.		ontop		MIRR.		COMA	
	1:1	n:1	1:1	n:1	1:1	n:1	1:1	n:1	1:1	n:1
CMT	**0.86**	0.00	0.79	0.00	0.57	0.00	0.00	0.00	0.58	0.00
Conference	0.78	0.00	**0.89**	0.00	0.56	0.00	0.00	0.00	0.56	0.00
SIGKDD	**1.00**	0.00	0.86	0.00	0.86	0.00	0.00	0.00	0.86	0.00

Table 9
Score break-down for queries that require 1:n class matches on the Oil & Gas atomic tests scenario.

Scenario	B.OX			IncM.			ontop			MIRR.			COMA		
	1:1	1:2	1:3	1:1	1:2	1:3	1:1	1:2	1:3	1:1	1:2	1:3	1:1	1:2	1:3
Oil & Gas Atomic	0.17	**0.11**	**0.07**	**0.20**	0.01	0.03	0.10	0.09	**0.07**	0.00	0.00	0.00	0.03	0.00	0.00

Tables 8 and 9 show the behavior of systems for finding n:1 and 1:n matches between ontology classes and table content, respectively. We highlight the n:1 case on restructured conference scenarios and 1:n matches on the oil & gas scenario as they include the highest number of tests in their respective categories. In both cases results are staggering with all systems failing the large majority of tests. For 1:n matches the situation is slightly better than it is with n:1 matches. This is not particularly surprising in general, as 1:n matches can be composed in mapping rules by adding up several correct 1:1 matches. A correct mapping of n:1 matches between classes and tables, on the other side, usually requires the much more challenging task of *filtering* from the table that holds entities of different types.

6.5. Semi-Automatic, Iterative Scenarios

We have also conducted semi-automatic, iterative experiments on *RODI* scenarios with two different systems, Karma and IncMap. While IncMap was also evaluated in the main line of experiments before on its fully automatic mode, Karma does not support such a baseline mode and always requires human intervention in different forms. This is mainly due to Karma's need for so called *Python transformations*, essentially tiny Python scripts, to skolemize entity IRIs. In contrast to class and property matches, Karma does not learn those transformations. Also, both systems work according to completely different semi-automatic processes. Karma is designed for multi-source integration and learns from human interactions in one scenario to provide suggestions in the *next ones*. IncMap, on the other side, adjusts its suggestions after simple yes/no feedback during *one single* scenario but has no memory between any two scenarios.

For these reasons, a direct experimental comparison between the two systems is not feasible. Instead, we run a separate dedicated experiment for each of them and identify similarities and differences in performance in the following discussion.

With Karma, we ran three experiments, each of which consists of a series of three related scenarios on the same target ontology. This translates to three different source schemata that Karma needs to integrate in a row. As Karma cannot produce any results completely

Table 10
Semi-automatic Karma mappings: generally very high scores thanks to human input.

Series	1st	2nd	3rd
To CMT	0.97	0.85	0.99
To Conference	0.90	1.00	1.00
To SIGKDD	1.00	0.99	1.00

Table 11
Impact of incremental mapping: scores for IncMap after k interactions in adjusted naming scenarios.

Scenario	@0	@6	@12	@24
CMT	0.45	0.73	0.92	0.96
Conference	0.53	0.61	0.68	0.77
SIGKDD	0.76	0.85	1.00	1.00

automatically, we conducted this experiment interactively and did record the number of human interactions needed to complete the mapping for each of the data sources. Figure 5 shows that in all cases the total number of required interactions drops for later data sources over previous ones. The drop in manual class matches and property matches is made possible by type learning. Python transformations remain approximately constant across subsequent data sources as no learning support and suggestions are available for these transformations.

Due to the manual input, mappings resulting from Karma's semi-automatic process are generally of high quality and did mostly reach scores close to 1.0 (cf. Table 10).

For IncMap, we ran a series of regular single-scenario tests, but in an incremental, semi-automatic setup [38]. That is, for each of the scenarios, we simulated human feedback in the form of choosing a suggestion from shortlists of three suggestions, each. To simulate this kind of feedback we simply used the benchmark as an oracle to identify the best pick. We observed how the score achieved by IncMap's mappings changes after a number of iterations, i.e., we are reporting a score at k human interactions [37].

Table 11 reports those numbers for three conference domain scenarios. We are reporting scores before feedback (@0), and after 6, 12, or 24 interactions, respectively. It is clearly visible that scores increase with ongoing feedback. From the first few rounds of feedback, the system profits most. After that, gains are moderate.

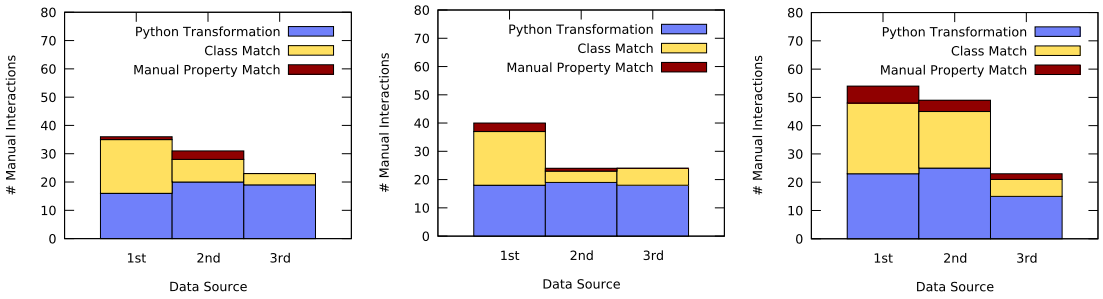
Note that these changes in score are based on feedback during several iterations on the same scenarios. It would be most interesting to see an evaluation of a system that combines the approaches of Karma and IncMap. From the results available from these two systems so far, it becomes clear that either approach has its own benefits. A direct comparison is not possible, though, as both follow a fairly different kind of process (multi-source vs. single-source) and also request different forms of human input (e.g., Python transformations in Karma).

7. Related Work

Mappings between ontologies are usually evaluated only on the basis of their underlying correspondences (usually referred to as ontology *alignments*). The Ontology Alignment Evaluation Initiative (OAEI) [52,24] provides tests and benchmarks of those alignments that can be considered a de-facto standard. Mappings between relational databases are typically not evaluated by a common benchmark. Instead, authors typically compare their tools to one or more of the industry standard systems (e.g., [22,10]) in a scenario of their own choice. A novel TPC benchmark [42] was recently created to close this gap. However, no results are reported so far on the TPC-DI website. To the best of our knowledge, no benchmark to measure specifically the quality of inter-model relational-to-ontology mappings was available before the original release of *RODI* [7].

Similarly, evaluations of relational-to-ontology mapping generating systems were based on one or several data sets deemed appropriate by the authors and are therefore not comparable. In one of the most comprehensive evaluations so far, QODI [53] was evaluated on several real-world data sets, though some of the reference mappings were rather simple. IncMap [40] was first evaluated on a choice of real-world mapping problems based on data from two different domains. Such domain-specific mapping problems could be easily integrated in our benchmark through our extension mechanism.

A number of papers discuss different quality aspects of relational-to-ontology mapping generation in a more general way. Console and Lenzerini have devised a series of theoretical OBDA data quality checks w.r.t. consistency [2]. As such, these could also be used to judge mapping quality to a certain degree. However, the focus of this work is clearly different. Also, the approach is agnostic of actual requirements and expectations and only considers consistency of data in itself. A more multi-dimensional approach has been proposed by Westphal et al. [54]. Their proposals do not include a comparable



(a) Target Ontology CMT (b) Target Ontology Conference (c) Target Ontology SIGKDD

Figure 5. Karma multi-source integration counting human interactions

scoring measure, though. In their benchmark, Impraliou et al. generate synthetic queries to measure the correctness and completeness of relational-to-ontology query rewriting [31]. The presence of complete and correct mappings is a prerequisite to their approach. Mora and Corcho discuss issues and possible solutions to benchmark the query rewriting step in OBDA systems [36]. Mappings are supposed to be given as immutable input. The NPD benchmark [34] measures performance of OBDA query evaluation. Neither of these papers, however, address the issue of systematically measuring mapping quality.

A comprehensive overview of relational-to-ontology efforts, including related approaches of automatic mapping generation, can be found in the following surveys [45,50].

8. Conclusion

We have presented a novel benchmark suite *RODI* that allows to test the quality of system-generated relational-to-ontology mappings. The prime application area of *RODI* is ontology-based data integration. *RODI* tests a wide range of data integration challenges that are specific to relational-to-ontology mappings, and which we identified in this paper.

Using *RODI* we have conducted a thorough evaluation of six prominent relational-to-ontology mapping generation systems from different research groups. We have identified strengths and weaknesses for each of the systems and in some cases could even point to specific erroneous behavior. We have communicated our observations to the authors of BootOX, IncMap, MIRROR and -ontop- and they already used our feedback to improve their systems and the quality of computed mappings. Overall, systems demonstrate that they can cope well with relatively simple mapping challenges. However, all tested tools perform poorly on most of the more advanced challenges that come close to actual real-world problems. Thus, further research is needed to address these challenges.

Future work includes repeated evaluations of a growing number of relational-to-ontology mapping generation systems. It would be particularly interesting to evaluate semi-automatic tools in a more comprehensive way, and to directly compare different tools under identical settings. Additionally, we expect several of the tested systems to address issues pointed by our evaluation with *RODI*. Another avenue of future work includes the extension of the benchark suite, e.g., by adding scenarios from other application domains relevant for ontology-based data integration.

References

[1] C. Batini, M. Lenzerini, and S. B. Navathe. A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Comput. Surv.*, 18(4):323–364, 1986.

[2] Marco Console and Maurizio Lenzerini. Data Quality in Ontology-Based Data Access: The Case of Consistency. In *AAAI*, 2014.

[3] Luciano F. de Medeiros, Freddy Priyatna, and Oscar Corcho. MIRROR: Automatic R2RML Mapping Generation from Relational Databases. In *ICWE*, 2015.

[4] Xin Luna Dong and Divesh Srivastava. Big Data Integration. *PVLDB*, 6(11):1188–1189, 2013.

[5] Bernardo Cuenca Grau et al. OWL 2: The Next Step for OWL. *J. Web Sem.*, 6(4):309–322, 2008.

[6] Birte Glimm et al. OWL: Yet to arrive on the Web of Data? In *LDOW*, 2012.

[7] Christoph Pinkel et al. How to Best Find a Partner? An Evaluation of Editing Approaches to Construct R2RML Mappings. In *ESWC*, 2014.

[8] Christoph Pinkel et al. RODI: A Benchmark for Automatic Mapping Generation in Relational-to-Ontology Data Integration. In *ESWC*, 2015.

[9] Craig A. Knoblock et al. Semi-Automatically Mapping Structured Sources into the Semantic Web. In *ESWC*, 2012.

[10] Cristina Civili et al. MASTRO STUDIO: Managing Ontology-Based Data Access applications. *PVLDB*, 6(12), 2013.
[11] David Aumueller et al. Schema and Ontology Matching with COMA++. In *SIGMOD*, 2005.
[12] Diego Calvanese et al. A 'Historical Case' of Ontology-Based Data Access. In *Proceedings of Digital Heritage (DH)*, 2015.
[13] Domenico F. Savo et al. Mastro at work: Experiences on ontology-based data access. In *DL*, 2010.
[14] Ernesto Jiménez-Ruiz et al. Large-scale Interactive Ontology Matching: Algorithms and Implementation. In *ECAI*, 2012.
[15] Ernesto Jiménez-Ruiz et al. BOOTOX: Practical Mapping of RDBs to OWL 2. In *ISWC*, 2015.
[16] Evgeny Kharlamov et al. Optique 1.0: Semantic Access to Big Data – The Case of Norwegian Petroleum Directorate's FactPages. In *ISWC (Posters & Demos)*, 2013.
[17] Evgeny Kharlamov et al. How Semantic Technologies Can Enhance Data Access at Siemens Energy. In *ISWC*, 2014.
[18] Evgeny Kharlamov et al. Ontology Based Access to Exploration Data at Statoil. In *ISWC*, 2015.
[19] Martin Giese et al. Optique – Zooming In on Big Data Access. *IEEE Computer*, 48(3), 2015.
[20] Michael Bada et al. A Short Study on the Success of the Gene Ontology. *J. Web Sem.*, 1(2), 2004.
[21] Ondrej Svab et al. OntoFarm: Towards an Experimental Collection of Parallel Ontologies. In *ISWC (Posters & Demos)*, 2005.
[22] Pieter Heyvaert et al. Towards Approaches for Generating RDF Mapping Definitions. In *ISWC (Posters & Demos)*, 2015.
[23] Ronald Fagin et al. Clio: Schema Mapping Creation and Data Exchange. In *Conceptual Modeling: Foundations and Applications*. Springer, 2009.
[24] Timea Bagosi et al. The ontop Framework for Ontology Based Data Access. In *CSWS (Posters & Demos)*, 2014.
[25] Zlatan Dragisic et al. Results of the Ontology Alignment Evaluation Initiative 2014. In *OM*, 2014.
[26] Sean M. Falconer and Natalya Fridman Noy. Interactive Techniques to Support Ontology Matching. In *Schema Matching and Mapping*. Springer, 2011.
[27] Fred Freitas and Stefan Schulz. Survey of current terminologies and ontologies in biology and medicine. *RECIIS – Elect. J. Commun. Inf. Innov. Health*, 3:7–18, 2009.
[28] Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *Database Systems – The Complete Book*. Prentice Hall, 2nd edition, 2008.
[29] Peter Haase, Ian Horrocks, Dag Hovland, Thomas Hubauer, Ernesto Jimenez-Ruiz, Evgeny Kharlamov, Christoph Pinkel, Johan Klüwer, Riccardo Rosati, Valerio Santarelli, Ahmet Soylu, et al. Optique System: Towards Ontology and Mapping Management in OBDA Solutions. In *WoDOOM13*, 2013.
[30] Thomas Hornung and Wolfgang May. Experiences from a TBox Reasoning Application: Deriving a Relational Model by OWL Schema Analysis. In *OWLED*, 2013.
[31] Matthew Horridge and Sean Bechhofer. The OWL API: A Java API for OWL ontologies. *Semantic Web*, 2(1), 2011.
[32] Martha Impraliou, Giorgos Stoilos, and Bernardo Cuenca Grau. Benchmarking Ontology-based Query Rewriting Systems. In *AAAI*, 2013.
[33] Jennie Duggan et al. The BigDAWG Polystore System. *SIGMOD Record*, 44(2), 2015.
[34] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. LogMap: Logic-Based and Scalable Ontology Matching. In *ISWC*, 2011.
[35] Davide Lanti, Martin Rezk, Mindaugas Slusnys, Guohui Xiao, , and Diego Calvanese. The NPD Benchmark for OBDA Systems. In *SSWS*, 2014.
[36] Wolfgang May. Information Extraction and Integration with FLORID: The MONDIAL Case Study. Technical report, Universität Freiburg, Institut für Informatik, 1999.
[37] Jose Mora and Oscar Corcho. Towards a Systematic Benchmarking of Ontology-Based Query Rewriting Systems. In *ISWC*, 2014.
[38] Heiko Paulheim, Sven Hertling, and Dominique Ritze. Towards Evaluating Interactive Ontology Matching Tools. In *ESWC*, 2013.
[39] Christoph Pinkel. Interactive Pay as You Go Relational-to-Ontology Mapping. In *ISWC, Part II*, 2013.
[40] Christoph Pinkel, Carsten Binnig, Evgeny Kharlamov, and Peter Haase. IncMap: Pay-as-you-go Matching of Relational Schemata to OWL Ontologies. In *OM*, 2013.
[41] Christoph Pinkel, Carsten Binnig, Evgeny Kharlamov, and Peter Haase. Pay as you go Matching of Relational Schemata to OWL Ontologies with IncMap. In *ISWC (Posters & Demos)*, 2013.
[42] Meikel Poess, Tilmann Rabl, and Brian Caufield. TPC-DI: the first industry benchmark for data integration. *PVLDB*, 7(13):1367–1378, 2014.
[43] Freddy Priyatna, Oscar Corcho, and Juan Sequeda. Formalisation and Experiences of R2RML-based SPARQL to SQL Query Translation Using Morph. In *WWW*, 2014.
[44] Mariano Rodriguez-Muro and Martín Rezk. Efficient SPARQL-to-SQL with R2RML mappings. *J. Web Sem.*, 33, 2015.
[45] Juan Sequeda, Syed Hamid Tirmizi, Óscar Corcho, and Daniel P. Miranker. Survey of Directly Mapping SQL Databases to the Semantic Web. *Knowledge Eng. Review*, 26(4), 2011.
[46] Juan F. Sequeda and Daniel Miranker. Ultrawrap Mapper: A Semi-Automatic Relational-Database-to-RDF (RDB2RDF) Mapping Tool. In *ISWC (Posters & Demos)*, 2015.
[47] Pavel Shvaiko and Jérôme Euzenat. Ontology Matching: State of the Art and Future Challenges. *IEEE Trans. Knowl. Data Eng.*, 25(1), 2013.
[48] Martin G. Skjæveland, Espen H. Lian, and Ian Horrocks. Publishing the Norwegian Petroleum Directorate's FactPages as Semantic Web Data. In *ISWC*, 2013.
[49] A. Solimando, Ernesto Jiménez-Ruiz, and G. Guerrini. Detecting and Correcting Conservativity Principle Violations in Ontology-to-Ontology Mappings. In *ISWC*, 2014.
[50] Dimitrios-Emmanuel Spanos, Periklis Stavrou, and Nikolas Mitrou. Bringing Relational Databases into the Semantic Web: A Survey. *Semantic Web*, 3(2), 2012.
[51] Mohsen Taheriyan, Craig Knoblock, Pedro Szekely, and José Luis Ambite. Learning the Semantics of Structured Data Sources. *J. of Web Semantics*, 2015.
[52] The Ontology Alignment Evaluation Initiative (OAEI). http://oaei.ontologymatching.org.
[53] Aibo Tian, Juan F. Sequeda, and Daniel P. Miranker. QODI: Query as Context in Automatic Data Integration. In *ISWC*, 2013.
[54] Patrick Westphal, Claus Stadler, and Jens Lehmann. Quality Assurance of RDB2RDF Mappings. Technical report, University of Leipzig, 2014.