ABSTRACT
The management of legal domains is gaining great importance in the context of data management. In fact, the geographical distribution of data as implied – for example – by cloud-based services requires that the legal restrictions and obligations are to be taken into account whenever data circulates across different legal domains. In this paper, we start to investigate an approach for coping with the complex issues that arise when dealing with data spanning different legal domains. Our approach consists of a conceptual model that takes into account the notion of legal domain (to be paired with the corresponding data) and a reference architecture for implementing our approach in an actual relational DBMS.

Categories and Subject Descriptors
A.m [General]: Miscellaneous; H.2.m [Database Management]: Miscellaneous

Keywords
Data Management, Access Control, Legal Domains, Privacy, Legal Rules

1. INTRODUCTION
Consider a multinational corporation composed of several subsidiaries in different countries, in which possibly different laws are enforced. As collaboration ensues among various branches, particular care is needed in managing how data flow between branches.

In fact, there are several issues concerning how to deal with data physically residing in places enforcing different legislations (or legal domains, thereon), and how to cope with mismatches and inconsistencies that may arise when dealing with constraints imposed by different territories. Typical cases are when data has to move from one place to another (as a result of a query issued from a different legal domain, or as a migration imposed by a cloud-based storage management).

Consider the following example: Marco, an Italian employee of a multinational corporation, collaborates with colleagues from the US branch on a project and he is consequently going to work for a period of time in the United States. The required procedure states that Marco has to register to the acceptance office of the US branch in order to get the proper authorizations. The registration process requires the access to Marco’s sensitive information and since the data physically reside on a server positioned in Italy, they are subject to the Italian law concerning the protection of sensitive information. As such, it may be the case that some information may not be available or it has to be - once read - managed with care, according to the constraints prescribed by the Italian law.

The foreign branch’s information system connects to the information system of the Italian branch and it issues a query requesting the above mentioned information. As the systems operate in different countries, it may well be the case that the query issued in US is legally compliant while in Italy it is not. That is, the required information in Italy is considered as sensitive and its disclosure is protected by appropriate privacy-related laws. For example no information is available for disclosure or it is available only in some aggregated form, or further, the purposes for which information is requested, as well as obligations to be fulfilled after the information is released, have to be specified along with the original query.

Therefore, a way to express the relevant parts of the legal domains should be available, as well as - of course - mechanisms for checking possible inconsistencies and mismatches. Also performing adequate compensatory actions are required. In fact, suppose that the US branch collects - as part of the process to release authorizations for its workers - information about salaries, as it is allowed by the US legislation. Such kind of information is considered sensitive by Italian legislation, and thus its unconditional release is not allowed. Hence, the registration of Marco within the US branch should fail. In such case, multiple actions could be taken into account and the requester behavior may depend from case to case. For example, an error message could be returned or the forbidden action could be sent to a third party for further inspection.

Other scenarios may include: (i) the American branch acquires the Italian one, without having the power of completely managing the corresponding data. In this case, in order to use Italian branch-related data, the American branch has to state what are the purposes for which such data are required and, in the case they comply with Italian law, enforce the corresponding constraints; (ii) the American branch acquires sensitive data previously held by the Italian branch. In this case, data can reside in servers on American ground provided that the American branch complies with constraints specified by Italian law. The scenarios just described suggests that, in order to properly deal with data moving across different legal domains, one has to take into account several non-trivial issues: (i) represent the constraints derived from several legal domains in an effective and expressive yet usable way; (ii) integrate such constraints along with the data they operate upon; (iii) define suitable corresponding
enforcement mechanisms; (iv) devise suitable ways of confronting different legal domains and propose apt compensatory actions in the case of mismatches.

We argue that a legal domain is a fundamental feature of the data that is associated with, and its management has to be tightly coupled to such data, representing and integrating the legal domain at the same conceptual level of the data themselves. In this paper we present a framework for representing the main components of a legal domain and we show how to integrate them with data stored in a relational database. We define a simple yet expressive rule-based language for representing the legal statements that compose a legal domain and we show how a legal domain regulates the operations on the data. Furthermore, we define guidelines to translate LegalRuleML [1] documents into our rule-based language. The paper is organized as follows: in Section 2 an overview of the related work; Section 3 introduces the Legal Domain Model; Section 4 presents the architecture we define for integrating such model into a relational DBMS; Section 5 sketches a translation methodology from standard LegalRuleML to our model; Section 6 outlines possible future work and draws the conclusions.

2. RELATED WORKS

The issue of representing legal rules or, more in general, legal knowledge and how to integrate them into a proper framework, in order to interact with a given application domain, is a long-standing open problem. However, to the best of our knowledge, our work is the first which proposes a tight integration of data and the legal domains, that may influence the process of querying such data. In a series of papers ([2], [3], [4]) the authors present an approach for the specification and the implementation of business and contracts rules from an human-oriented form to a readable and executable language form based on RuleML [5]. This line of work partially share some objectives with ours. Namely, they propose a model and a formalized language for representing artifacts that have features similar to legal rules (such as obligations, for example). However, the cited works do not propose a modelization of full-fledged legal domains neither propose methods for (a) integrating the information deriving from a legal domain with or (b) comparing different legal domains. A very relevant work that has helped draw attention on the problems one encounters when managing data physically dislocated in very large – possibly spanning different continents – geographical areas is [6]. Such work describes mechanisms for selectively replicating a large, cloud-based repository. The replication policy is expressed by means of an ad-hoc, constraint-based language that allows to express which portion of data to replicate, according to its geographical position and other workload-related information. However, the presented language cannot manage the constraints deriving from the legal domains associated with the data and with the query source. A large body of work on themes relevant for our work has been carried on by the EU-funded project called Optimis [7]. Such project offers a clean-slate approach in defining and implementing the full stack of frameworks that are necessary for supporting IaaS cloud services, in such a way that service providers may effectively orchestrate cloud services. In this way the providers may operate (possibly legacy) apps on the cloud and perform sensible decisions about the deployment that take into account a wide variety of parameters like trust, risk, efficiency, legal constraints and cost. Furthermore, Optimis requires to completely rewrite the entire enterprise architecture and completely redesigning its data repositories. On the other side, our approach aims at conservatively extend the already existing repositories with information about the corresponding legal domains. Optimis defines also an helpful guidance [8] about legal issues of cloud computing. In addition they provides a series of recommendations to the EU Commission concerning Data Protection and Data Security. They follow a high level approach by assessing legal problems at European level in order to ensure compliance across the various jurisdictions of the Member States. Moreover, most of the works associated with Optimis [9, 10] are focused on data migration based on performance and reliability purposes while our approach could be use to migrate data to accomplish and respect legal domain laws. The same reasoning applies to [6]. The problem of how to model norms and to obtain a sound representation of legal reasoning is the subject of an intense and vast research effort, see for example [11, 12].

3. LEGAL DOMAIN MODEL

The conceptual model we present is an abstraction of the relevant building blocks of a legal domain. As a starting point, we are interested in modelling the legal rules that state what kind of actions can be performed on a given class of objects, and what are the various kinds of involved subjects, which we refer to as legal roles (consider as an example, the data owner), and what are their relationships. Typically, the conditions under which actions are allowed comprise purposes and obligations, that specify respectively what a legal entity has to perform (i) before it gains the right to perform the actions themselves and (ii) after it has acted on the objects, as stated in a legal rule. In some circumstance, conditions are prerequisites that the entity must posses to perform the desired tasks and no further acts can vary the rule behavior. For example, consider the following statement (belonging to the Italian law that regulates the management of sensitive information and privacy-related issues): "the – possibly temporary – transfer outside Italian territory of personal information, is possible in the case the information owner has explicitly expressed her/his consent. The consensus has to be in written form in the case the transferred data are sensitive" (translation of Article 43.1 point (a), in Section 7 of Privacy Law 196/2003). In this case, (i) the legal role explicitly mentioned in the statement is the owner of the personal information to be transferred, (ii) the object is – quite obviously – the mentioned personal information, (iii) the action is the physical transfer of such information outside Italy, (iv) the conditions are related to the explicit consent (possibly in written form) of the information owner. Another example is given in point (g) of the same Article 43.1 in which the transfer of sensible data is allowed, given that such data are accessed for scientific purposes (what are "scientific purposes" is defined elsewhere in the text of the above mentioned law). The building blocks composing our model are: (i) a set $U$ of users, that specify the legal entities; (ii) a set $A$ of actions, that specify what can be performed by legal entities or what can be performed on objects owned by legal entities; (iii) a set $O$ of objects, upon which the previously mentioned actions operate; (iv) a set $C$ of conditions, that express the circumstances under which the actions may be executed or prevented; (v) a set $I$ of intents, that express the motivations for a set of actions to be performed; (vi) a set $OB$ of obligations, that specify what subsequent actions are to be performed (by whom) after the action is accomplished. Hence, we define a legal rule as an element of the Cartesian product of actions, objects, conditions, intents and obligations: a legal rule $LR$ is a tuple from $A \times O \times C \times I \times OB$. We define a legal domain $LD$ as a set of legal rules. Finally, a Legal Domain Model (or LDM for short) is therefore defined as a couple $(U, LD)$, where $LD$ is the legal domain users in $U$ are subjected to. In most legal domains, users are modelled through roles. More precisely, by role we mean a set of users identified by some well-specified criteria (i.e., the possession of credentials or the behaviours they
are supposed to exhibit, in a given context). For example, the roles DPTitular and DPResponsible denote respectively the sets of users that are the titular of the processing of sensitive data (that is, the users that are in charge of deciding what are the purposes for collecting and storing sensitive data and under what circumstances they may be accessed) and the users that are liable for the actual management of the data, on behalf of the holders. We now define in a more precise way how conditions and obligations are expressed in our model. Conditions are expressed using boolean formula (built up from predicates involving users, roles - which are set of users - objects and environment variables), while obligations are expressed as tuples of the form \((\text{action, object})\). Such basic entities are combined in order to express legal rules, from simple ones containing conditions and actions, to more complex ones that take into account obligations, intents or complex actions as well. The basic building blocks of such conditions are variables that record relevant information about legal statements that is to be taken into account when comparing two or more legal domains. Even if the condition language \(C\) has limited expressive power, it is able to model the most relevant conditions found in legal statements. The conditions that can be stated by \(C\) are defined in the following paragraphs. We fix a finite set \(V\) of variables. Each variable \(v \in V\) is associated to a finite domain \(D_v\). An atomic condition is written as \((x op y)\) or \((x op y)\), where \(x, y\) are variables from \(V\), \(\text{op}\) is an operator from the set \(\{=, \neq, \leq, \geq, <, >\}\), \(v\) is a constant from the domain \(D_v\). Each variable \(c\) is defined on objects and roles yielding \(true\) or \(false\) and are atomic conditions of the language. Then, the set \(C\) of conditions expressed by \(C\) (defined over \(V\)) is defined as: (a) an atomic condition is a condition of \(C\); or (b) if \(c_i, c_j\) are conditions of \(C\), then (i) \(c_i \land c_j\), (ii) \(c_i \lor c_j\), (iii) \(\neg c\), are conditions of \(C\). In order to describe a real-world example, we define several variables and predicates that are useful when interacting with data that is under the jurisdiction of a given legal domain: (i) OwnerAccord\((\text{data})\) OA: a predicate that represents the owner consent for elaboration on the specified data. The predicate yields \(true\) if and only if the owner gave his consent about the utilization of its data; (ii) intentOfCollect\(IOC\): a variable that records the motivation for which the data are originally collected. Its values depend on the specific context domain, in our example possible values are \{research, commercial, billing, authority\}; (iii) intentOfRequest\(IOR\): a variable that represents the reason for which a request is done. As in the previous case, its values depend of the specific context domain, in our case, they are \{research, commercial, billing, authority\}; (iv) requestLegalDomain\(RELD\): a variable that represents the legal domain of the requester; (v) ownerLegalDomain\(OWLD\): a variable that represents the legal domain of the data’s owner; (vi) objectLegalDomain\(OBLD\): a variable that represents the legal domain on which the data is physically located; It also important to define the domains over actions, objects, intents and legal domain areas take their corresponding values (the terms between parentheses are their abbreviations): (i) actions are taken from the set \{read, create, update, delete\}; (ii) objects are taken from the set \{name, surname (SNA), email address (EM), education (EDU), address (AD), age (A), salary (SA), PersonalData (PerD), SexData (sexSD), ReligionData (RelD), FinancialData (FinD), HealthData (HeaD)\}; (iii) intents range over the set \{research, commercial, billing, authority\}; (iv) legal domains are named using the names of the countries (or super-national organizations) in which they hold, \{Italy(IT), Germany(GER), United State of America(USA), China(CN), Europe(EU)\}. We now present some translations of actual laws (or parts of them) into legal rules defined in our model. We start by representing the Article 5, Point (b) of the European Privacy Law: *Principles relating to personal data processing – personal data must be: (b) collected for specified, explicit and legitimate purposes and not further processed in a way incompatible with those purposes*. Thus, a first, possible translation according to our model is the following legal rule, that contains the symbol \(\perp\) as a place holder for the components of the legal rules that are not specified in the statement. Furthermore, we use the symbol \(+\) to denote that any value taken from the proper domain can be used in the rule. Finally, we assume that the variable \(iOC\) matches with purpose that has been defined as “legitimate”. In this case, we assume that “research” is such a purpose.

\[ (EU, (\text{read, }*)_{\perp}, \text{IOC= research, } \perp) \]

Going back to Marco’s example, in which there is a legal statement prescribing that sensitive data of Italian public administration workers cannot be physically stored in servers outside Italian territory, such legal statement, if applied to health information, is expressed as:

\[ (\text{IT}, (\text{write, HealD), *}, \text{OBLD! } \text{IF, L}) \]

### 4. ARCHITECTURE

The model we presented in the previous section is a conceptual representation of the components of a legal domain. In summary, our model accounts for data objects, their owners, the users that aim at accessing them and the conditions and actions that have to be satisfied by the requesters in order to gain such access. We now proceed to illustrate how to translate such high-level, rule-based model into a corresponding set of tables and constraints defined in the relational model. That is, assuming that the data are stored in a set of relational tables, we provide a set of precise guidelines for representing all the components of our model, along with a set of constraints over such tables. We identify three different legal domains that play a fundamental role in the relational representation of our model: the owner legal domain \(OWLD\): the legal domain of the owner of the object or the person who takes his place; the requester legal domain \(RELD\): the legal domain of the entity that perform the request on a object; the object legal domain \(OBLD\): the legal domain which regulates the territory where the object is physically located when the request is performed. Henceforth, the relational representation of such legal domains (along with their constraints) define what actions may be performed on the data objects (represented in relations as well). In order to introduce this additional information to an already existing database, new tables and attributes have to be added to it and we start by defining attributes for model components previously described. The first attribute to represent is the one that permits to identify to which legal domain (and respective rules) an information belong to. We allow a fairly high level of granularity in the binding among the information and the legal domains by assuming that the minimal amount of data stored in the database that can be associated to a legal domain is a tuple. In this way every tuple is an object which belong to a legal domain as a single indivisible block. Hence to every tuple is added a new attribute. This may be the \(OWLD\) or \(RELD\) depending on the specific implementation and the legal domain that want to be considered as legally pertinent. This associated information may changes during the time. In fact the \(OWLD\) usually never change while the \(OBLD\) may change while moving the data on the cloud. Again, the same tuple could have different \(OBLD\) if replicas are stored in different locations which belong to legal domains. In our prototype, we choose \(OWLD\). Furthermore, we have to consider another information related to the legal domains identification. Each database user (or system process) has to be associated tp a legal domain which indicates the domain regulating where they reside or live (\(RELD\)). This information is needed while querying the database to ascertain the ability to access data
To better understand how the system works is useful to follow the path of a simple query in the above described process. We assume that data are stored in a relational DBMS, as shown in Fig. 1.

<table>
<thead>
<tr>
<th>employee</th>
<th>occupation</th>
<th>project</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK ID</td>
<td>name</td>
<td>surname</td>
</tr>
<tr>
<td>FK1</td>
<td>PK ID</td>
<td>FK2</td>
</tr>
</tbody>
</table>

**The Legal Domain Management System.**

To handle the new database structure and correctly shunt requests and legal domains we need to introduce a new layer, the Legal Domain Management System (LDMS). This framework silently intercepts any request/traffic directed to the DBMS As already discussed, the database has ad-hoc tables representing legal domains and also new attributes are added to the current tuples.

The LDMS is composed of different modules:

- **Query Analyzer**: Intercept the user’s query and user’s metadata before they reach the DBMS. It’s purposes are to extract all the available data (RELD, IOR, A and all the present attributes) and pass them to the Elaboration Unit.
  - input: the original query and requester metadata.
  - output: variables instantiate with any information available in the SQL query.

- **Information Retrieval Module**: it interfaces directly with the DBMS and with a series of interrogations retrieves all the information needed.
  - input: any request from LDMS’ units and the original query from the Query Analyzer.
  - output: the Legal Domain tuples and any request object metadata available to the Elaboration Unit.

- **Elaboration Unit**: it performs the comparison algorithm on the available data.
  - input: all the needed information from the other Units.
  - output: the permitted information (which respects all the constraints) to the Answer Composer.

- **Answer Composer**: it builds the final answer with the permitted information and sends it back to the requester.
  - input: data from the Elaboration Unit.
  - output: an answer to the requester less any preserved information.

To understand how the system works is useful to follow the path of a simple query in the above described process. We assume that data are stored in a relational DBMS, as shown in Fig. 1.
For INSERT and UPDATE commands the procedure is basically the same. Before making a change to the DB which may be prevented by one of the rule, the layer verifies the target objects with a series of READ queries and then, only if the operation is allowed, it forward the original query to the DBMS. As such, the LDMS output could be only one of the following: (i) an information if the query involves only data reading. The Information Retrieval Module may be filtered; (ii) a successful attestation if the query requires data manipulation and the process is successful. This happens normally on insert and update requests; (iii) an error message if the action could not be completed. Normally this occur if a rule prevents the operation.

Complexity. It is useful to analyze and determinate the number of resources, as time and storage, that are necessary to correctly handle our proposed architecture. We will analyze the worst case. On a traditional database, the introduction of legal domains involves the addition of new fields to every available tuple (the attribute which represent the LD which could be OWLD, OBLD or both of them). Following the presented example and suppose to have a database composed of \( N_I \) tuples then \( N_I \) attributes are added. The second addition are Legal Domains tables. These are collection of rules and then their dimension depends on the number of permission statements that are inserted. If no rules are inserted their dimension is equal to zero. Any inserted rules need seven attributes (ID, text, object, action, intent, condition, obligation). If \( N_R \) rules are inserted than \( 7N_R \) more space is needed. In the end the total space required is \( N_I + 7N_R \) new attributes.

Speaking about computation complexity, the Comparison algorithm is the only component which introduces substantial computation work in the LDMS. It is invoked once time for every action-attribute pair requested. This means that if the requester queries for \( N_a \) attributes it is executed exactly \( N_a \) times. Each execution check all the rules matching to the action-attribute pair to verify if one of the rule found prevents the requested action. The number of permission rules for action-attribute pair is \( N_{pa} \) (rules for attribute). For every rule of \( N_{ra} \) the algorithm control the Intent and then initiate a dual loop to control one to one every condition present. If the number of conditions in the rule is \( N_{rc} \) (rule conditions) and the number of information that the requester bring with him is \( N_{rC} \) (requester conditions) the total number of comparison in the worst case (the element we look for is always the last in the list) is \( N_{rc} \times N_{rC} \). For every condition correspondence three confronts are done so totally there are \( 3N_{rc} \) comparisons. If full match is found than the obligations in the rule are added to the obligation_list array. This operation requires to \( 2N_{rc} \) This process is repeated for every condition and then the system check if all the conditions rule have been satisfied to know if the rule has been fulfilled. This step introduces \( N_{rc} \) checks. Again at the end it check if all the rules passed the control and then fulfill all the obligations in obligation_list array. Finally, we may summarize the algorithm complexity by the number of operations it needs to return an ALLOW or PROHIBITED string to the Answer Composer. The total process is \( N_a \times N_{rc} \times (N_{rc} \times N_{rC} \times (3N_{rc} + 2N_a) + N_{rc}) \).

5. TRANSLATING INTO LEGALRULEML

We now provide guidelines to translate a generic LegalRuleML rule into our model. LegalRuleML (LRML) is an extension of RuleML with features a specific syntax to handle the different aspects and problems that can rise in legal contexts, through the formalization of rules, laws, regulations, guidelines and legal reasoning. We consider again the example previously introduced and the legal statement prescribing that sensitive data of Italian workers can be physically stored in servers outside Italian ground. To obtain a LegalRuleML representation we need to determinate which atoms are involved and combine them in a graph to model conditions and consequences. A condition is formed by the following components: (a) a request access on some data; (b) the data is a sensitive information; (c) the requester is positioned outside the Italian ground, while a consequence contains only the statement affirming that request access on data is prohibited. First, each sentence has to be analyzed to identify the main predicate and variables.

\[
\begin{align*}
\text{input : } & \text{rules_list = an array of the rules targeting the same action-object pair;} \\
\text{output: } & \text{allow = an array of the requester information;} \\
\text{foreach (rule of rules_list) do} & \\
\text{ } & \text{rules_count++;} \\
\text{ } & \text{if (rule.Intent == request.Intent) then} \\
\text{ } & \text{foreach (condition cv of rule) do} \\
\text{ } & \text{cv_count++;} \\
\text{ } & \text{foreach (condition cv of request) do} \\
\text{ } & \text{if (rule.cv.name == request.cv.name AND} \\
\text{ } & \text{rule.cv.value == request.cv.value) then} \\
\text{ } & \text{if (rule.mode == ‘!’) then return} \\
\text{ } & \text{PROHIBITED;} \\
\text{ } & \text{foreach (obligation ob of rule) do} \\
\text{ } & \text{add ob to obligation_list;} \\
\text{ } & \text{else return PROHIBITED} \\
\text{ } & \text{end} \\
\text{ } & \text{fulfilled_cv++;} \\
\text{ } & \text{end} \\
\text{ } & \text{if (fulfilled_cv == cv_count) then fulfilled_rules++;} \\
\text{ } & \text{end} \\
\text{ } & \text{if (fulfilled_rules == rules_count) then} \\
\text{ } & \text{foreach (obligation ob of obligation_list) do} \\
\text{ } & \text{fulfill ob;} \\
\text{ } & \text{end} \\
\text{ } & \text{return ALLOW;} \\
\text{end} \\
\text{else return PROHIBITED;} \\
\end{align*}
\]

Figure 2: Confrontation Algorithm

\[
<\text{rule id="rule43.1(a).7"}>
<\text{Atom id="condition01">} \\
<\text{Rel>sensible/Rel}> \\
<\text{Var} data/Var> \\
<\text{Atom}> \\
<\text{neg}> \\
<\text{Atom id="condition02">} \\
<\text{Rel>italian/Rel}> \\
<\text{Var} requester/Var> \\
<\text{Atom}> \\
</\text{neg}> \\
<\text{Atom id="condition03">} \\
<\text{Rel>action/Rel}> \\
<\text{Var} request/Var> \\
<\text{Atom}> \\
\]
The above LRML rule is a short, simplified translation for example purpose, but it is still possible to notice how general is the representation itself. While LRML is perfect for formalization and distribution among heterogeneous systems, it is not specific enough to deploy it directly in a relational database. We need specific targets and actions chosen from a finite set of possibilities in order to apply restrictions to a relational representation of real-world items.

In other words, a LRML rule is to be translated into one or more specific rules, each of them regarding a specific object and action.

For example, we already translated the above rule in a more specific rules, each of them regarding a specific object and action.

The above LRML rule is a short, simplified translation for example purpose, but it is still possible to notice how general is the representation itself. While LRML is perfect for formalization and distribution among heterogeneous systems, it is not specific enough to deploy it directly in a relational database. We need specific targets and actions chosen from a finite set of possibilities in order to apply restrictions to a relational representation of real-world items.

In other words, a LRML rule is to be translated into one or more specific rules, each of them regarding a specific object and action.

For example, we already translated the above rule in a more specific one using our presented model: \( (\text{IT,}(\text{write,HealD}),\text{oblD} \neq \text{IT,L})) \); in this case, the object under scrutiny is considered as sensitive information and that we want to protected accomplishing the current laws: t. In other environment the sensitive data registered could concern the symata data preferences (SynadacateD) and the rule modeled as: \( (\text{IT,}(\text{write,SyndacateD}),\text{oblD} \neq \text{IT,L})) \);

As a matter of fact, we assume the existence of some national/international authority or, again, a legitimate organization which provides to us with a series of laws expressed using LRML. In all these circumstances, we need to process and rewrite such rules to follow our model and adapt the generic concept to specific database objects over a set of possible actions. We thus sketch a set of informal guidelines to translate a LRML rule in one or more LDML rules: (i) identify the main structure of the rule, in most of the case the body is composed of two branches: if and then; (ii) identify in the if branch which data the law protects and which action may trigger the respective prohibitions, restrictions or impediments. Normally the law target is identified by the ‘Var’ tag while the actions, as predicates, are indicated by the ‘Rel’ tag. The target represents our model object. If the final context is known then the genetar data could be modeled to represent real data objects instead of class of elements (i.e. salary and tax code could be used instead of 'fiscal information'). The same reasoning can be done for the trigger action that is model in the LDML ‘action’ field. For example, if the rule has to be translated and protect information in a database context, a generic ‘access’ action could be translated in a read request. If a motivation is present it is indicated as a variable and is translate in the LDML intent. If a branch is enclosed in a negation tag then it’s meaning that it is to be considered as negated. Eventually the if may contain the conditions as the circumstances that must happen in order to apply the law or the characteristic that the law subject must posses. In this case the additional data are modelled by the LDML conditionx. If multiple conditions are present then they are added in a logical conjunction. (iii) identify in the then branch the consequence of the previous section. Only if the if branch is satisfied then this part is executed. If obligations are present (also as predicated and variables), they are modelled in the LDML ‘obligation’ field.

6. CONCLUSIONS

In this work, we have presented a conceptual model for representing the legal domain associated with (the legal entities that may manage) a given dataset and a corresponding architecture for integrating such model into a relational DBMS. In order to facilitate the exchange of information about legal constraints among different legal domains, we have introduced a set of simple guidelines for translating a LegalRuleML statement in our model. As further work, we plan to fully implement the presented architecture with a real-world test case. Also, we plan to study more effective procedures for testing the compatibility – given different definitions of such notion – among different legal domains and to better specify the translation process into LegalRuleML.

Acknowledgments: this work has been supported in part by the Italian Ministry of Education, Universities and Research PRIN project “Relazioni sociali ed identita’ in Rete: vissuti e narrazioni degli italiani nei siti di social network”, by FIRB project RBFR107725. We would like to thank Francesca Gaudino of Baker & McKenzie, Milan for her invaluable support concerning legal matters.

7. REFERENCES