LIQUID Description

1 Background

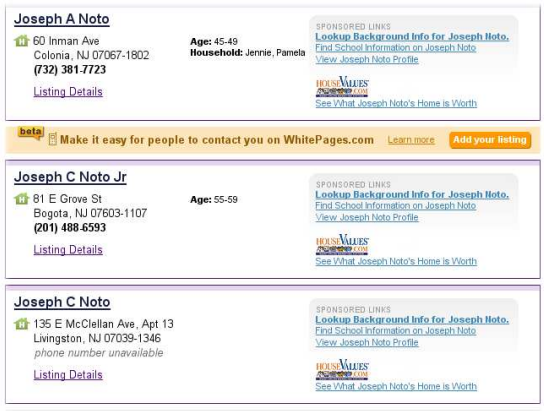
The term *Deep Web* (sometimes also called *Hidden Web*) [23, 8, 12] refers to the data content that is created dynamically as the result of a specific search on the web. For example, when we query a Yellow Pages website, the generated result is the result of a query posed on an underlying database, and cannot be indexed by a search engine. In this respect, the Deep Web content resides outside web pages, and is only accessible through interaction with the web site – typically via HTML forms. As an example of Deep Web source, consider the Whitepages website. Suppose we are looking for a person named Joseph Noto, and living in New Jersey; the form would be filled as in Figure 1, where a corresponding SQL query is also shown. The result in Whitepages.com is shown in Figure 2, and it is clearly representable as a relational table (shown in the bottom part of the figure).

Therefore, a Deep Web source can be naturally modelled as a relational table (or a set of relational tables) where not all queries are allowed; in fact, *access limitations* exist on such relations. In particular, they can be queried only according to so-called *access patterns*, each of which enforces the selection on some of the attributes (which corresponds to filling the corresponding field in the form with a value).

It is believed that the size of the Deep Web is several orders of magnitude larger [19] than that of the so-called *Surface Web*, i.e., the web that is accessible and indexable by search engines. The information stored in the Deep Web is invaluable, but at the same time hard to collect automatically on a reasonably large scale.


```
SELECT *
FROM whitepages
WHERE firstname='joseph'
AND lastname='noto'
AND stateprov='NJ'
```

Figure 1: Query posed to whitepages.com.



Firstname	Lastname	Address	City	...
Joseph A.	Noto	60 Inman Ave.	Colonia	...
Joseph A.	Noto	81 E Grove St.	Bogota	...
Joseph C. Jr.	Noto	135 E McCellan Ave.	Livingston	...
...	...	...	...	...

Figure 2: Query result in whitepages.com.

Two main approaches to accessing the Deep Web exist [23]. In the *virtual data integration* approach, each Deep Web site is a data source, and sources are integrated and unified by a *global schema* representing the domain of interest. Normally, in virtual data integration there is a relatively small number of sources, storing data related to a single domain. In the *Surfacing* approach, instead, answers are pre-computed from Deep Web sources by posing suitable queries, and then the results are indexed as normal static pages in a search engine.

In this project, we focus on the virtual integration approach, with the aim of providing techniques for seamless integration of a possibly large number of sources. We assume the schema (in relational form) of each source is known, and we do *not* address the problem of automatic source discovery and schema inference, which goes beyond the aims of this project. Several works in the literature address Deep Web wrapper induction via machine learning [29, 28], web form understanding [16, 3], constraint induction [31] and mapping generation [14].

Modelling and reasoning. In virtual data integration, as above mentioned, the domain of interest is represented by means a *global schema* (a.k.a. *mediated schema*) [21], which is constructed and designed, at least in principle, independently of the sources in question. A *mapping*, in general consisting of views, expresses the semantic relationship be-

tween the sourche schemata and the global schema. A crucial aspect in modelling is the use of *constraints* (also called *dependencies* in database parlance; they are also said to constitute an *ontology*) are expressed on the global schema so as to better model the domain; "classic" constraints are key and foreign key dependencies. Global constraints are a fundamental tool for accurately representing the properties of the domain of interest, and in general the data may not satisfy them; instead, constraints are considered as logical assertions and employed to infer useful information. Starting from Johnson and Klug's seminal work in the context of query containment [18], the problem of reasoning on constraints has been addressed in several works in the literature. In particular, [6] consider the case where the ontology is represented as an Entity-Relationship schema. Other approaches consider Description Logic constraints [10, 17], possibly integrated with rules [25]; "traditional" constraints in database theory such as key and inclusion dependencies [11]; constraints defining ontologies derived from F-logic [20]; extensions of Datalog [5]. While reasoning techniques on constraints and rules are abundant in the literature, these are rarely considered in conjunction with access limitations, which is a natural setting for the Deep Web; moreover, to the best of our knowledge, the issue of constraints *on the sources*, that are locally enforced, and that could cause more constraints to hold globally, has never been tackled. Notice that the presence of access limitations restricts the set of data that are accessible to queries; since the accessible data are the only "visible" from outside, we are to consider such data as the only relevant data. The data visible in the system enjoy specific properties, as they are "shaped" by the same access limitations. From the formal point of view, therefore, access limitations are in fact constraints, and as such they interact with other semantic global constraints.

We believe, in the light of the above observations, that in order to build the next generation of Deep Web integration systems, as we aim to do, the following research questions are to be answered.

1. What constraint languages are suitable for Deep Web integration systems? Which reasoning services do they offer, and what is the computational complexity of such services?
2. How do constraints in a Deep Web data integration system, both at source and global level, interact with access limitations on the sources? How are new global constraints entailed from given ones (both at source and global level), mapping, and access limitations?

Query processing Answering queries is the ultimate goal of a data integration system [21]. Several work deal with query processing under constraints [6, 25, 5]. Notably, the work on *DL-Lite* [1, 9, 27], a family of tractable description logics, has focussed on low *data complexity* (complexity with respect to the data size only), normally in the low complexity class AC_0, of conjunctive query answering and of knowledge base satisfiability.

The presence of access limitations alone complicates query processing significantly [22, 30, 2]. To illustrate this, we show a simple example. Consider the following relational Deep Web sources: $r_1(Title, City, Artist)$, representing information about concerts, with song title, city of performance, and artist name, and requiring the second attribute to be selected; $r_2(Artist, Nation, City)$, representing name, nationality and city of birth of artists, and requiring the first attribute to be selected. In this case, given the conjunctive query $q(A) \leftarrow r_2(A, italian, modena)$ asking for names of Italian artists born in Modena, we notice that q cannot be immediately evaluated, since r_2 requires the first attribute to be bound to a constant (selected). However, the two attributes named $City$ in r_1 and r_2 both represent city names, and similarly the attributes named $Artist$ represent artist names. In such a case, we can use names of artists extracted from r_1 to access r_2 and thus extract tuples that may contribute to the answer. More precisely, we start from the constant $modena$, present in the query, and access r_1; this will return tuples with new artist names; such constants (artist names) can be used to access r_2. In turn, new tuples from r_2 may provide new constants representing city names, that can be used to access r_1, and so on. Once this recursive process has terminated, we have retrieved all obtainable tuples that contribute to the answer. It is easily seen that query answering under access limitations is inherently recursive, and therefore costly [22, 8]. Reducing the computational cost of query processing is a key issue addressed only by a few studies in the literature [8, 4]. The presence of constraints further complicates query processing [13]. So far, no study

has addressed the problem of query processing in the presence of expressive constraints, such as those of [5], derived from Datalog.

In our endeavour towards the creation of a new kind of Deep Web integration systems, we identify the following major research questions.

1. How are the answers to queries formally defined in a Deep Web integration system, with constraints of high expressive power as well as access limitations on the sources?
2. What algorithms are suitable for query answering in this context, and what is their computational cost?
3. How can the cost of query processing be reduced, by making use of the information carried by constraints and access limitations?

2 Objectives

The goal of the LIQUID (*Logic-based Integration and Querying of Unindexed Internet Data*) project is to develop techniques for processing queries over integrated Deep Web sources, giving raise to a new kind of intelligent, scalable integration systems for seamless integration of large sets of Deep Web data. To achieve such an ambitious goal, we identify the following objectives.

OBJ1 We will investigate formalisms to model Deep Web sources and their integration, including expressive logic constraints to model both integrity constraints enforced on the sources and global constraints to represent the domain of interest.

OBJ2 We will study logic-based reasoning for answering queries in the simultaneous presence of access limitations and constraints, as well as mappings between source and global schemata. Efficiency of query processing is crucial, therefore we shall devise techniques for reducing the time resources used in this crucial task.

OBJ3 We will investigate methodologies and algorithms to design and build a Deep Web virtual integration system, scalable to a potentially large number of sources. We will validate our investigation by building and testing a prototype system.

The project has therefore the goal of devising systems capable of providing high-quality information by means of answers to structured queries posed on the global schema, freeing users from tedious keyword searches and browsing of results. LIQUID will constitute a significant leap forward with respect to the state of the art.

3 National importance

We believe that the LIQUID project will contribute to several related research disciplines. In particular, LIQUID places itself in the Information and Communication Technology Theme, with a focus on the areas of Information Systems, Artificial Intelligence Technologies and Databases.

LIQUID is aimed at providing formal and algorithmic tools for building Deep Web integration systems; such tools, if the project is successful, will enable firms to provide high-quality information that is currently accessible only manually and at a high cost. The ability of integrating structured information on the web can impact several aspects of the industry; in general, this would make markets and production processes more efficient, with potential reduction of cost in providing services. This, from a more general and social point of view, would also indirectly improve sustainability.

From a more foundational point of view, the research outcomes of LIQUID will benefit researchers in the Semantic Web, Databases (especially logics applied to databases), and more generally Knowledge Representation and reasoning. These are the areas in which the UK has traditionally been world leader in scientific research.

4 Methodology

In LIQUID, we envisage two main parts, which we further divide into workpackages (WPs).

Part 1: Source modelling and constraint inference (WP 1-2). In this phase we study suitable languages to model sources with access limitations, global schemata with expressive constraints, and mappings. A particular emphasis is given to automated reasoning tasks that infer new constraints and discover inconsistencies.

Part 2: Query processing and optimisation (WP 3-4). In this part we develop query processing

techniques that take into account the access patterns and all constraints in a virtual integration system for Deep Web data. In order to obtain scalability and efficiency, we make use of the semantic information carried by the constraints, the mapping and the access patterns in order to *prune* the query plan that is to compute the answers to a query. This amounts to avoiding sources accesses that are known not to return any new information that is useful to the query. This can be done at plan generation time (*static* optimisation) or at run-time (*dynamic* optimisation); in the latter case, also the data extracted previously are involved in the reasoning.

4.1 Source modelling and constraint inference

Workpackage 1: Deep Web data modelling. In this workpackage we shall devise a novel logic-based model for properly representing the different parts of the LIQUID framework. In particular, we will study formalisms and methodologies for a *knowledge base*, that will serve for modelling all parts of the system. The knowledge base will be based on the relational model, and it will consist of expressive logic assertions to specify the source and global constraints as well as the mappings. In LIQUID, the Datalog$^{\pm}$ family of ontology languages [5], which is based on Datalog, will be our starting point for the data and ontology modelling. While Datalog is well understood and established, it has some important limitations [26]. However, Datalog$^{\pm}$ is inspired by *Datalog with value invention*, and is therefore capable of expressing most of the constructs used in the Semantic Web and conceptual modelling of data. The LIQUID system will be highly innovative in this respect, for reasons that we summarise here. *(1)* We will use Datalog$^{\pm}$ to specify not only the mappings between sources and global schema, but also the main global ontology. The system will therefore perform inference on the basis of an expressive knowledge base, offering the schema designer a high expressive power. *(2)* The Datalog$^{\pm}$ formalism will be used in LIQUID also to specify inter-source dependencies. One of the goal of the project is in fact to investigate the mutual influence of source constraints (both inter-source and intra-source) and the global ontology. This is a novel aspect and we believe it needs to be investigated especially having in mind possible optimizations. *(3)* LIQUID will possibly manage *uncertainty* of data by means of assigning suitable probability spaces to mappings. Therefore, the system will answer probabilistic queries, following an approach inspired by that of [7].

Workpackage 2: Reasoning on constraints in the Deep Web. As mentioned above, we are going to investigate *constraint inference* in the knowledge base representing a Deep Web virtual integration system. Interestingly, to the best of our knowledge, the interaction between source and global constraints in information integration has hardly been addressed in the literature. Constraints and access patterns on the sources might entail constraints on the global schema, and vice-versa. It is therefore of utmost importance to develop techniques for automated reasoning on access patterns and constraints so as to be able to detect inconsistencies (i.e., unsatisfiaility of the knowledge base) and to infer new constraints, whose discovery can help schema design at the global level. This task in LIQUID is rather ambitious, especially considering the high expressive power of the chosen modelling formalisms.

4.2 Query processing and optimisation

Workpackage 3: Query processing under constraints and access patterns. Query processing is the ultimate goal in LIQUID. This makes this workpackage a crucial one. In our approach, queries will be structured and expressed on the global ontology. Notice that in LIQUID queries are processed on-the-fly, without resorting to pre-computed data pages; this ensures that the answers are *fresh*. One problem that a system like LIQUID faces is the presence of access limitations on the sources [24]. Even to answer simple select-project-join (a.k.a. conjunctive) queries, a *recursive* plan is in general required [22, 8]. Being inherently recursive, query processing can be slow, due to the usually slow response time of web sources. Reducing Deep Web source accesses is therefore extremely important. The combination of expressive constraints (both global and local), mappings and access patterns complicates query answering. For instance, the access limitations restrict the part of the source data that can be accessible by queries, but constraints might extend the accessible part, thus impacting query answering.

We will possibly also deal with processing of *probabilistic queries*, devise a sound semantic for them, and query processing techniques. Finally, in LIQUID, we shall build a *prototype* to test our query answering algorithms and the optimisations. Given the limited time and resources, we do not plan to build a full-fledged Deep Web integration system; instead, we will produce a light-weight prototype where Deep Web sources will be most likely simulated locally. This will be enough to validate our findings.

Workpackage 4: Query optimisation. Several works address the problem of optimizing query processing in this case [8, 15, 22]. However, in LIQUID we have an ambitious goal: to exploit the information about the intra-source and inter-source constraints to reduce the query processing time, by discovering accesses that are unnecessary to query answering, but are possible according to the generated query plan. In the case of Datalog$^{\pm}$ assertions, or even simple inclusion or functional dependencies, the problem is still open. The problem is theoretically extremely interesting and challenging, and we intend to tackle it in LIQUID.

Static optimisation. One measure of the cost of the execution of a plan is the number of accesses to Deep Web sources; in LIQUID we are likely to consider other ones as well. The idea is to generate a query plan that avoids accesses to sources by determining, according to suitable criteria, whether they provide useful information. This reasoning is to take into account the access patterns as well as the constraints, and it presents significant challenges. In static optimisation then, once the plan is generated and optimised, it is esecuted as it is on the sources. However, there is still room for more computational cost reduction. *Dynamic optimisation.* Once we have a (recursive, in general) query plan, there is still the chance that we are able to save accesses on the basis of: *(1)* the constraints on the sources; *(2)* the data extracted at a certain point in query evaluation. This has been done, for example, by considering functional dependencies only (see [8]). A work that addresses the problem is [4]. Let us see a simple example. Consider a source $s(A_1, A_2, A_3)$, where both the first and the second attribute have to be selected to access the relation; suppose the functional dependency $s : A_1 \rightarrow A_2, A_3$ holds on s. Let D be a database and t be a tuple previously extracted from s, with $t[A_1] = a_1$ ($t[A_1]$ denotes the projection of t on the single attribute A_1). Suppose that we have some value a_2 with which to access s by a selection on A_2. If we try to access s using the pair a_1, a_2, the access cannot provide any new tuple: because of the functional dependency that has to be satisfied by D, it can either provide t alone (in case $a_2 = t[A_2]$), or no tuple (in case $a_2 \neq t[A_2]$). Therefore, the information on t, which can be known *only at runtime* and never at query plan generation time, allows us to avoid an unnecessary access. In LIQUID we will deal with much more expressive constraints, thus facing hard theoretical and practical problems.

References

[1] Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The DL-Lite family and relations. *J. Artif. Intell. Res.*, 36:1–69, 2009.

[2] Michael Benedikt, Pierre Bourhis, and Clemens Ley. Querying schemas with access restrictions. *PVLDB*, 5(7):634–645, 2012.

[3] Michael Benedikt, Tim Furche, Andreas Savvides, and Pierre Senellart. ProFoUnd: program-analysis-based form understanding. In *Proc. of WWW*, pages 313–316, 2012. Demonstration.

[4] Michael Benedikt, Georg Gottlob, and Pierre Senellart. Determining relevance of accesses at runtime. In *Proc. of PODS*, pages 211–222, 2011.

[5] Andrea Calì, Georg Gottlob, Thomas Lukasiewicz, Bruno Marnette, and Andreas Pieris. Datalog+/-: A family of logical knowledge representation and query languages for new applications. In *Proc. of LICS*, pages 228–242, 2010.

[6] Andrea Calì, Georg Gottlob, and Andreas Pieris. Ontological query answering under expressive entity-relationship schemata. *Inf. Syst.*, 37(4):320–335, 2012.

[7] Andrea Calì, Thomas Lukasiewicz, Livia Predoiu, and Heiner Stuckenschmidt. Tightly coupled probabilistic description logic programs for the semantic web. *J. Data Semantics*, 12:95–130, 2009.

[8] Andrea Calì and Davide Martinenghi. Query-

ing the deep web. In *Proc. of EDBT*, pages 724–727, 2010.

[9] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The DL-lite family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

[10] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. On the decidability of query containment under constraints. In *Proc. of PODS*, pages 149–158, 1998.

[11] Diego Calvanese and Riccardo Rosati. Anwering recursive queries under keys and foreign keys is undecidable. In *Proc. of KRDB*, 2003.

[12] Kevin Chen-Chuan Chang, Bin He, and Zhen Zhang. Toward large scale integration: Building a metaquerier over databases on the web. In *Proc. of CIDR*, pages 44–55, 2005.

[13] Alin Deutsch, Bertram Ludäscher, and Alan Nash. Rewriting queries using views with access patterns under integrity constraints. *Theor. Comput. Sci.*, 371(3):200–226, 2007.

[14] Ronald Fagin, Laura Haas, Mauricio Hernández, Renée Miller, Lucian Popa, and Yannis Velegrakis. Clio: Schema mapping creation and data exchange. *Conceptual Modeling: Foundations and Applications*, pages 198–236, 2009.

[15] Daniela Florescu, Alon Y. Levy, Ioana Manolescu, and Dan Suciu. Query optimization in the presence of limited access patterns. In *Proc. of SIGMOD*, pages 311–322, 1999.

[16] Tim Furche, Georg Gottlob, Giovanni Grasso, Xiaonan Guo, Giorgio Orsi, and Christian Schallhart. OPAL: Automated form understanding for the deep web. In *Proc. of WWW*, pages 829–838, 2012.

[17] Birte Glimm, Carsten Lutz, Ian Horrocks, and Ulrike Sattler. Conjunctive query answering for the description logic shiq. *J. Artif. Intell. Res. (JAIR)*, 31:157–204, 2008.

[18] David S. Johnson and Anthony C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *Journal of Computer and System Sciences*, 28(1):167–189, 1984.

[19] Govind Kabra, Zhen Zhang, and Kevin Chen-Chuan Chang. Dewex: An exploration facility for enabling the deep web integration. In *Proc. of ICDE*, pages 1511–1512, 2007.

[20] Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42:741–843, July 1995.

[21] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS*, pages 233–246, 2002.

[22] Chen Li and Edward Chang. Computing complete answers to queries in the presence of limited access patterns. *VLDB Journal*, 12(3):211–227, 2003.

[23] Jayant Madhavan, Loredana Afanasiev, Lyublena Antova, and Alon Y. Halevy. Harnessing the deep web: Present and future. In *Proc. of CIDR*, 2009.

[24] Todd D. Millstein, Alon Y. Levy, and Marc Friedman. Query containment for data integration systems. In *Proc. of PODS*, pages 67–75, 2000.

[25] Boris Motik and Riccardo Rosati. Reconciling description logics and rules. *J. ACM*, 57(5), 2010.

[26] Peter F. Patel-Schneider and Ian Horrocks. A comparison of two modelling paradigms in the semantic web. *Journal of Web Semantics*, 5(4):240–250, 2007.

[27] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008.

[28] Sriram Raghavan and Hector Garcia-Molina. Crawling the hidden web. In *Proc. of VLDB*, pages 129–138, 2001.

[29] Pierre Senellart and et al. Automatic wrapper induction from hidden-web sources with domain knowledge. In *Proc. of WIDM*, 2008.

[30] Guizhen Yang, Michael Kifer, and Vinay K. Chaudhri. Efficiently ordering subgoals with access constraints. In *Proc. of PODS*, pages 183–192, 2006.

[31] Daisy Zhe Wang, Xin Luna Dong, Anish Das Sarma, Michael J. Franklin, and Alon Y. Halevy. Functional dependency generation and applications in pay-as-you-go data integration systems. In *Proc. of WebDB*, 2009.