The Entity Name System: Enabling the Web of Entities

Heiko Stoermer, Themis Palpanas, George Giannakopoulos
University of Trento
{stoermer, themis, ggianna}@disi.unitn.eu

***Abstract*— We are currently witnessing an increasing interest in the use of the web as an information and knowledge source. Much of the information sought after in the web is in this case relevant to named entities (i.e., persons, locations, organizations, etc.). An important observation is that the entity identification problem lies at the core of many applications in this context. In order to deal with this problem, we propose the Entity Name System (*ENS*), a large scale, distributed infrastructure for assigning and managing unique identifiers for entities in the web.**

In this paper, we examine the special requirements for storage and management of entities, in the context of the *ENS*. We present a conceptual model for the representation of entities, and discuss problems related to data quality, as well as the management of the entity lifecycle. Finally, we describe the architecture of the current prototype of the system.

I. Introduction

One of the major problems that have emerged through the Semantic Web (SW) effort [20] is the problem of uniquely identifying entities. The problem derives from the fact that different users, or systems, assign and use different identifiers for the same real-world entity. As a result, we cannot effectively reason about this entity, exactly because it is not consistently being assigned the same identifier.

The same problem is also relevant to information and knowledge management in the enterprise environment, where its successful solution could help in two directions. First, by enabling the efficient integration and management of entity-centric information within an enterprise that is scattered over several data stores, and second, by allowing the effective correlation of the owned information with relevant information from external sources.

Even though in this work we do not formally define the notion of entity, we will use the term entity to refer to individuals, particulars, and instances. (We include a rigorous discussion and definition in our previous work [5].) Our notion of entity is quite liberal, and includes things like products, organizations, associations, countries, events, publications, hotels, people, etc. It may also include fictional objects (e.g., Pegasus), objects from the past (e.g., Plato), or abstract objects (e.g., Gödel's Theorem).

Consider the following two motivating scenarios, which demonstrate the added value that a web-based system for entity identification can bring.

Enhancing Business Intelligence (BI): Often times in BI applications the goal is to collect all relevant information about a specific entity. For example, SAP is interested in gathering information about each one of their products (differentiating among versions) from their internal databases, knowledge bases, and forums, as well as from external sources, such as weblogs. It turns out that this is a difficult task (also when considering only the internal sources), since the products are not always referred to with their official, full names. Tagging these references with global identifiers would alleviate the above problem, even when integrating information from external sources.

Connecting Physical Objects to the Web: Physical objects, tagged with an identifier that is easily readable by a machine (e.g., through RFID or 2-dimensional visual codes, such as QR) can be scanned by a mobile device, and lead the user to entity-centric information about this object gathered from the Web (e.g., detailed specifications for a product, nutrition information for a food package, etc.). Evidently, this information can be richer when these identifiers are global, enabling easy access to entity-centric information from a huge variety of data sources[1]. The city of Manor (in Texas, USA) has already a pilot program that uses these technologies (which are also available in a simpler form in Japan and South Korea).

Our thesis is that entity identification is at the core of several applications that are becoming increasingly relevant in a world of interconnected information. Along with the problem of assigning global identifiers to entities also come the problems of managing these identifiers throughout the entire lifetime of the entities. Giving efficient solutions to the above issues is the goal of our Entity Name System (ENS) [7], [3], [5]. The *ENS* aims to improve the linkage of data about individual entities in the World Wide Web (WWW), by handling the process of assigning and managing identifiers for entities. These identifiers are global, with the purpose of consistently identifying a specific entity across system boundaries, regardless of the place in which references to this entity may appear.

In this work, we present a high level overview of the *ENS* system (which is being developed within the OKKAM project), while focusing on the management of the entity lifecycle, one of the crucial components of the system. We describe our model for entity representation, and the data quality issues that arise in this environment. We discuss the need for evolution of the entity representations, and propose specific directions for enabling and assisting in this evolution

[1] An example of a system that provides such functionality is SIGMA (`http://www.sig.ma`)

process. Finally, we give an overview of our current prototype implementation of *ENS*.

II. Related Work

This is the first time that a system like the *ENS* is being developed. However, systems exist which have been developed for other domains and are used to solve slightly different problems that exhibit similar (to some extent) functionality to the *ENS*.

There exists lots of work, and a (renewed) recent interest, in Master Data Management (MDM) systems [10]. The purpose of MDM systems is to store and manage in a centralized manner all the information relevant to the core objects of some business (e.g., customers, suppliers, and products for a retail store). A crucial difference is that in the *ENS* we are not interested in handling all the information (knowledge) about an entity. The aim is rather to have a minimal amount of information based on which we are able to uniquely identify each entity in the system.

Yahoo! GeoPlanet [2] is an open infrastructure for managing unique identifiers for geographically-permanent named entities (i.e., geographic locations, such as cities and landmarks). The ENS has the ambitious goal of extending such a functionality to any named entity (that is, to persons, organizations, etc.). A similar vision was recently sketched by Yahoo! as well [8]. Though, it is not clear whether this approach will lead to an open system like the *ENS*, where users (and applications) can add new entities.

OpenCalais [1] is a framework of tools that allows users to automatically create semantic metadata for unstructured documents. When given as input a document, OpenCalais will identify the named entities mentioned in the document, and will additionally generate tags with known facts about these entities. Nevertheless, this framework does not allow users to freely create and annotate new entities in the system. Furthermore, for integration tasks we still have to do reasoning in order to identify entities that are the same across processed documents (since they may not be tagged with the same identifiers).

The Digital Object Identifier (DOI) system, which is based on the Handle system, has been developed for identifying content objects (mainly documents) on the internet [14], [19]. The DOI system is specifically focused on digital content objects, and given an identifier of such an object, it provides resources relevant to that object. The DOI system does not meet the requirements we have set for *ENS*, since it is not possible to search for the identifier of an object given any of its attributes (or any relevant keywords).

The Consistent Reference Service has been proposed for the problem of “URI synonymity” [13]. This service helps detect the set of identifiers that have been issued by various data sources for the same entity. This problem is orthogonal to the one tackled by *ENS*. In fact, such a service could provide information for the entity representations in the *ENS* (see Section IV).

Even though the *ENS* is not a knowledge base, with the help of (the identifiers assigned by) the *ENS*, information on specific entities can be easily and promptly accumulated. For example, in the context of the WWW such a service is already being provided by Sindice[2] and SIGMA[3] [9].

III. The Entity Name System

A. Overview

In this section we give a brief overview of the *ENS* (a more detailed presentation can be found elsewhere [7], [18], [3]), which we will use as the basis for our discussion. Note however, that our discussion is relevant to any system for entity identification management. The overall goal of the *ENS* is to handle the process of assigning and managing unique identifiers for entities in the WWW. These identifiers are global, with the purpose of consistently identifying a specific entity across system boundaries, regardless of the place in which references to this entity may appear (see Figure 1).

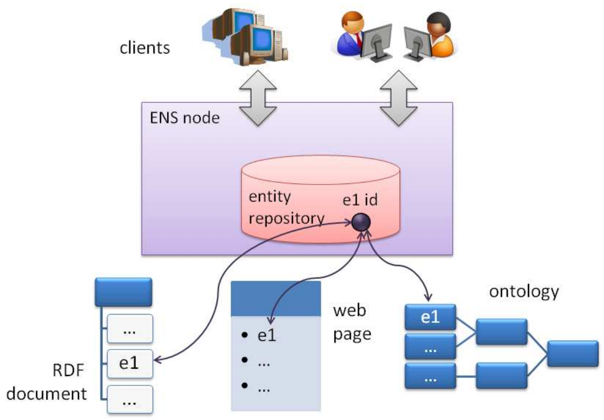

Fig. 1: Schematic of the *ENS* and its interactions.

The *ENS* has a repository for storing entity identifiers (note that this repository will be distributed and replicated) along with some small amount of descriptive information for each entity. The purpose of storing this information is to use it only for discriminating among entities, not exhaustively describing them. Entities are described by a number of attribute-value pairs, where the attribute names and the potential values are user-defined (arbitrary) strings, as we will discuss in more detail in Section IV-A.

Clients can be both human users and applications, and may inquire about the identifier of an entity by providing a set of attributes that describes this entity. If the entity exists in the repository, the system returns its identifier. Clients may also modify the state of the repository, either by inserting a new entity in the system, in which case the *ENS* returns the newly assigned identifier, or by changing some of the attributes of an existing entity.

As shown in Figure 1, the end result is that all instances of the same entity (i.e., mentioned in different systems, ontologies, web pages, etc.) are assigned the same identifier. Therefore, joining these data sources and merging their information becomes a much simpler and more effective process than before.

[2]`http://sindice.com`
[3]`http://www.sig.ma`

B. Identity and Identifiers

The landscape of the SW contains systems that provide identifiers for semantic entities, in the form of RDF URIs. Among them are identifiers for geographic entities (e.g. in the Geonames ontology[4]), for encyclopedic entities (in the DBPedia system[5] or the Yago ontology[6]) or even Genes (in the Uniprot Gene Ontology[7]). The current practice is to use RDF URIs for several different goals: (i) redirecting to a set of assertions about a non-web resource; (ii) linking from one set of assertions to another; (iii) providing a surrogate/substitute/proxy for non-web resources. It is evident that there is a certain ambiguity in this approach, which has been partly resolved by an architectural mechanism which helps distinguishing which of the three roles an RDF URI is playing.

We note the following important distinction between the RDF URI identifiers, and the identifiers produced and used by the *ENS* (for a more detailed, theoretical discussion refer to [5]): the *ENS* identifiers are *rigid designators*[8] for any kind of concrete and particular entity. In simplified terms, the above statement says that while the interpretation of an RDF URI is a set of RDF triples (about an entity), the interpretation of an *ENS* identifier is the entity itself. The important point being that even though the *ENS* identifier can be used to access information about the entity it denotes (in the *ENS*), the sole purpose of this information is to support the process of reaching consensus about the identity of the entity, i.e., to fix the referent. The use of the RDF URI should thus be limited to the first two goals mentioned above, while the *ENS* identifier covers the third one.

IV. Entity Management

We now discuss issues relevant to the lifecycle management of the entities, which represents an important component of the system with several new challenges. More specifically, we present our approach to entity representation, data quality, repository evolution, and online monitoring of the use of the repository.

A. Entity Representation

In the *ENS*, we represent an entity E as a tuple $E =< eid, Aid, prid, D, M >$ [3]. An entity is identified by an entity identifier, *eid*, assigned by the system. Along with this identifier, we store a set of alternative ids, *Aid* that other systems have assigned to the same entity, which can also be used to access the entity or produce *same-as* statements. The preferred id, *prid*, is one of the above identifiers that we use when displaying the entity[9]. The descriptive part of the entity representation, D, contains the attribute name value pairs that describe the entity, and a set of references (external to the system) that are about this entity. Finally, we store a set of metadata for each entity, M, that include usage statistics, and provenance and access control metadata. All the above information is individually indexed to allow fast access to it, and then stored as a single XML file on disk.

Note that our goal in *ENS* is to keep the entity representation as simple and general as possible, without imposing a fixed schema of entity types or attributes [18]. This approach is similar to the dataspace paradigm [12], and can benefit from the techniques developed in that area.

However, we acknowledge the benefit that such a schema would bring in. In order to reach a compromise between the generality of the description and the precision in the functionality of the system, we propose the (optional) classification of all entities in seven broad, top-level categories, namely, person, organization, location, event, artifact, and other. For each one of these entity types, we have identified a set of default attributes that are most commonly used to describe them [3]. These default attributes are suggested to users during entity creation, but are of course optional. Nevertheless, we expect that most of the created entities will use (at least some of) the default attributes, which in turn will have positive influence on the performance of the system.

When describing entities using attribute name value pairs, we face the following two problems. First, the same attribute name may appear in several different variations that are all semantically equivalent (e.g., “last name” and “surname”). Second, the same attribute name may appear in several different languages (e.g., “last name” in english and “cognome” in italian). In order to solve these problems, we use a canonical representation for attribute names, which is one of the forms that a particular attribute name may take. The name entered by the user at creation time is stored in the entity representation, and if its mapping to one of the canonical names that the system knows can be inferred, then this canonical name is the one used internally by the system in order to improve the quality of results of query answering. These mappings are managed separately from the entity descriptions, and can be enriched and extended over time, by adding new vocabularies or standardized ontologies, in order to better match the way that the system is being used.

B. Data Quality

Data quality issues are a major concern in the *ENS*. In the following paragraphs, we briefly describe our efforts in assuring data quality at creation time, as well as over the lifetime of the entity representation in the repository.

The aim of the data quality enforcement at entity creation time is to ensure that the new entities satisfy a minimal set of quality requirements. This assessment takes place not only every time a new entity is about to be stored in the system, but also when an existing entity is being modified. The results of the assessment are of two kinds: a list of corrective actions that the user has to follow before the entity can be stored in the system; or a set of warnings that aim at focusing the attention of the user at some particular areas of the entity description that the system thinks may be problematic from the data quality perspective.

[4] `http://www.geonames.org/ontology/`

[5] `http://www.dbpedia.org`

[6] `http://www.mpi-inf.mpg.de/yago-naga/yago/`

[7] `http://www.uniprot.org/manual/gene_ontology`

[8] An identifier is called a *rigid designator* if in all possible worlds it denotes the same object [15].

[9] The only guaranteed, unique identifier is still the *eid*, which is always used internally in the *ENS*.

The data quality assessment process at creation time that is currently in place within ENS, consists of the following three types of checks.

1) Attribute value quality, where we want to ensure that the values entered for the various attributes in the entity description are correct and valid.
2) Intra-entity description quality, where we check the quality of the entity description as a whole.
3) Inter-entity description quality, where we make sure that the changes about to happen in the repository will not degrade the overall quality performance of the system.

Examples for the first type of checks include the existence of attributes with no value specified, and the conformance to some standard formats for special attributes like date and time.

In the second type of checks, we look for attribute name-value pairs that are duplicated in the description of the entity, attributes with the same name but different values (this issues a warning), and for entity descriptions with too few or too many attributes.

The third type of checks is particularly important, because it gives us the unique opportunity to identify and remedy problems that will affect the performance of the entire system in a streaming fashion, right at the time when such problems will first appear (i.e., when creating or modifying an entity description). In this case, we perform a duplication check, which looks for other entities in the repository that are very similar to the entity under consideration. If such entities exist, it means that either the same real world entity is already stored in the *ENS* (therefore, it should not be stored a second time), or the description provided for that entity cannot sufficiently differentiate it from other entities in the repository (therefore, its description has to be enhanced). Note that in a dynamic and large scale system like the *ENS*, the alternative of identifying and correcting this type of problems *after* they appear is extremely cumbersome and prohibitively expensive, since it has to be an offline process, operating on the entire repository.

Apart from the above techniques for increasing the quality of the data that enters the system, we are also working on mechanisms that will ensure a high data quality over time. In particular, we are interested in maintaining the freshness of the data stored in *ENS*. In our case, freshness translates to how accurate the description of the entity is over time, or in other words, how closely it follows the real world entity as this changes over time. Evidently, we can only approximate this process, since we do not have a complete knowledge of the real world. The algorithms we propose can efficiently monitor the change patterns of individual entities (based on the history of *observed* changes), and provide a prediction for the timeframe of the next change. Therefore, we can take proactive steps for ensuring the accuracy of the entity representation in the repository (e.g., perform a targeted crawl of authoritative sources for the particular entity).

C. Evolution of Identifiers

Even though the identifier of an entity should never change, in some special circumstances this may happen. For example, this happens when we realize that two different entity representations refer to the same real world entity, or when the same entity representation is already being used to refer to two distinct real world entities. In these cases, we would like to take corrective action, by performing a *merge* and a *split*, respectively. Note that these operations may also become necessary as a result of the update of an existing entity. As such, they are closely related to the data quality problem discussed earlier.

The *ENS* supports the merge and split operations as follows. In the case of a merge operation, we merge the distinct attribute name-value pairs of the two entities, and select one of the existing identifiers (i.e., the one that according to the system statistics belongs to the most popular entity of the two) to be the identifier of the merged entity. The other identifier will still exist, so that users can refer to it, but only the selected identifier will be returned as a result. When splitting an entity representation into two new ones, we have no option but creating two new identifiers, since the old identifier has been used to refer to a non-existing (wrong) entity. In both cases, a human user has to refine the final entity descriptions, and the system keeps enough information to be able to undo these operations. We also provide a service that makes the information about these changes available for users (and/or machines) to read. This service publishes the merge and split operations along with their attached timestamps on a server that *ENS* users can access.

D. Monitoring of Repository Usage

The way that users access the system and interact with it may provide useful insight on what actions to take in order to improve the performance of the *ENS*. Consider the following example. Assume that many users search for an entity with attributes A_1 and A_2, and always select entity E_1, which is the only entity in the repository that contains attribute A_1 in its profile. If E_1 does not contain A_2 as well, we may choose to add it to the profile of E_1, because many users refer to E_1 using A_2. Alternatively, assume that the query for entities with attributes A_1 and A_2 returns n entities, $E_1, E_2, \ldots, E_n$, that satisfy the search conditions, but the interested users always select entity E_k, $1 \leq k \leq n$. In this case, we may choose to increase the importance of entity E_k, so that it ranks first for the particular query.

In both the above situations, we are interested in monitoring the data streams relevant to the usage patterns of the system. We have extended and adapted algorithms that can operate in an online fashion, and are flexible enough to allow effective and efficient data analysis of the incoming data streams [21], [17]. By monitoring and analyzing the way users interact with the *ENS* we can, for example, determine which entities or profile attributes are relevant to specific queries or to certain contexts, and modify the rankings of the entities in the query results, or update the representation of the entities, in order to produce more relevant search results.

E. Exploiting User Feedback

The volume and volatility of the data contained within the *ENS* offer a challenge concerning both the maintenance and accessibility of the information. The maintenance of the entity data would be a rather expensive and cumbersome task, was it not designed as a joined effort with the *ENS* user-community. By relying on the collective wisdom and collaboration of users, we aim to induce and use their feedback in order to achieve high data quality. The premise is that users will be willing to "adopt" some entities, and make sure that their descriptions in the *ENS* are always accurate.

In support of this functionality, we are developing the *Adaptive Entity Subscription System (AESS)* [11]. The AESS allows a user to *subscribe* to entities of interest (e.g., one's own entity, or entity of one's home town) and get informed about *changes* on these entities. The subscription system helps clients follow the changes of entities in the *ENS* at the time they occur through asynchronous messages (e.g., via e-mail). The dissemination of change information allows users to follow the evolution of the descriptions for the entities of interest, and enables them to take corrective actions, without having to periodically check for problems.

The personalized aspect of AESS takes into account what kinds of change are interesting for a given subscription client. The system ranks the information on changes in every sent message, according to user preferences on what is important. It is further able to adapt to the individual use-patterns of each client. The ranking is based on a user model, which is created through machine learning and a feedback-based user modeling methodology [11]. The system further adapts to user interest shifts, based on the information of the feedback, so that the information flow to the user remains aligned to user interest. This kind of personalization provides ease of use and enhances the user's experience and interaction with the system, thus facilitating the update of and access to *ENS* data.

V. System Design

In this section we give an overview of our efforts in implementing a publicly available prototype of the *ENS*.

A. System Architecture

The *ENS* consists of the following four main components:

1) **Entity Store**. This component is responsible for the low-level tasks of storing and managing entity data, as well as relevant metadata.
2) **Index**. Search operations for an identifier via an entity description in the *ENS* are accelerated via index structures over the data persisted in the entity store.
3) ***ENS* Core**. This component is implementing the services that are exposed by the *ENS*, namely, entity creation and update, matching, as well as processes relevant to the entity lifecycle management.
4) **Offline Processing**. The *ENS* is a system that is designed and optimized to achieve good performance in request-response style online processing. Additionally, there is the overall requirement to accommodate for components that are not directly connected to answering online identifier search requests, such as aspects of data quality assurance or the AESS, mentioned in Section IV. For this reason, the overall system architecture includes a component for offline processing that is separate from the online system, so as not to negatively influence its performance.

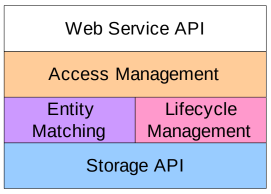

Fig. 2: The *ENS* Core Architecture

The *ENS* Core itself, as illustrated in Figure 2, consists of several parts that we describe bottom-up. The **Storage API** abstracts from the complexities of accessing the Entity Store and Index components mentioned above. In particular, the underlying strategies for addressing the individual, load-balanced clusters and retrieving all relevant data are hidden from upper layers. The **Matching** component is responsible for ensuring a high top-k precision of entity identifier search results. It has two main tasks: (i) parsing, analyzing and – if necessary rewriting or expanding – the search request coming from a client, and (ii) applying sophisticated ranking mechanisms with the aim to establish the best match between the search request and the candidate entities. The **Lifecycle Manager** takes care of the entity creation- and update-related tasks, and the important aspects of data quality and data lifecycle management. Most of the functionalities of this component have been outlined in Section IV. The **Access Manager** ensures that the requirements of access control and privacy are fulfilled. It performs, among other things, query and result set filtering to avoid undesired use of the system. All services that the *ENS* offers as its public API are exposed through the **Web Services** component, which constitutes the uppermost layer of the *ENS* component stack.

The *ENS* is conceived to be used as a background component in a service-oriented architecture. The only way to access the system is via a SOAP web service API. The *ENS* also provides end-user frontends that are accessible via the web, for identifier search, creation of identifiers, as well as administrative tasks. Access control is realized via certificate-based authentication which secures all web services.

B. Prototype Implementation

In the following, we give a brief description of the *ENS* prototype implementation (see Figure 3 for the concrete architecture of the system).

In order to fulfill the scalability requirements, the main components of the system support distributed processing. The *ENS* Core has been developed from scratch, implementing the architectural elements depicted in Figure 2 as a single compilation unit. The *ENS* Core is a stateless in-memory component which implements no persistence by itself and requires no shared memory, and can thus be distributed in a simple fashion behind a request-based load balancer. Both the entity store and the index follow the concept of horizontal partitioning (or "sharding") to deal with large data sizes plus replication to

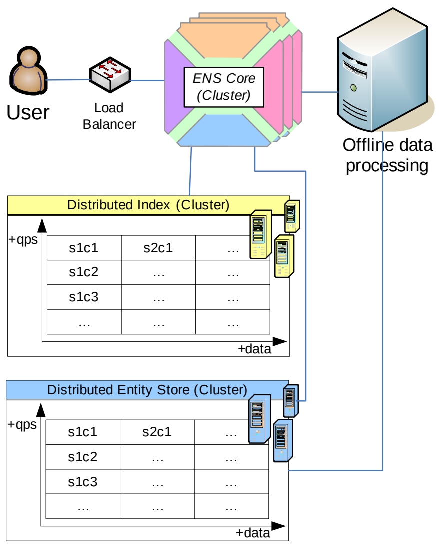

Fig. 3: Architecture of the *ENS* prototype implementation.

	v1.1 (02/2009)	**v2.0** (11/2009)	v3.0 (06/2010)
Repository capacity	1Mio	50Mio	500Mio
Repository population	1.03Mio	7.4Mio	50Mio
Avg. response time	800msec	750msec	400msec
Queries per second	5	7	50
Number of CPU Cores	4	32	32
Clustered components	index	all	all

TABLE I: Quantitative and qualitative indicators describing the *ENS* (data for v3.0 are estimates).

deal with large numbers of requests. After evaluating several available technologies , Project Voldemort[10] has been chosen for implementing scalable and highly performing storage of entity data, while Apache Solr[11] has been the only viable open-source solution of choice for maintaining very large retrieval indexes.

All components and services performing background tasks such as data cleansing etc. as mentioned before, are hosted in a separate environment alongside the *ENS* Core, labeled “offline data processing” in Figure 3.

Table I presents some qualitative and quantitative data on the evolution of the system. In the current release (i.e., v2.0), the repository contains around $7,500,000$ entities, and can support up to $600,000$ queries per day. The next version of the *ENS* (to be released in June of 2010) will contain more than $50,000,000$ entities, and will be able to handle more than $4,000,000$ queries per day.

In addition to the *ENS* itself, several tools and plugins have been developed for enabling the tagging of named entities with *ENS* identifiers during the time of creation of the data content. Such tools are available for use with knowledge engineering tools (e.g., NeOn Toolkit and Protégé) [16], social networking applications (e.g., foaf) [4], and word processors (e.g., Microsoft Word) [6].

[10]`http://project-voldemort.com/`
[11]`http://lucene.apache.org/solr/`

VI. Conclusions

Several modern applications are moving towards the direction of adding semantics to the information, and using these semantics for enabling a vastly richer range of data analytics. In this paper, we argue for an entity naming system, where unique identifiers for entities are assigned and managed, as the basis for enabling the vision mentioned above. We describe the design principles and architecture of the *ENS* system, which is under development. We examine the special requirements of representing entities within this context, and present relevant research directions. Finally, we present preliminary results on the performance of our implementation of the *ENS*.

Acknowledgment

This work was partially supported by the FP7 EU Large-scale Integrating Project OKKAM - Enabling a Web of Entities (contract no. ICT-215032). For more details, visit http://www.okkam.org.

References

[1] Open Calais. http://www.opencalais.com/, Nov. 2009.
[2] Yahoo! GeoPlanet. http://developer.yahoo.com/geo/geoplanet/, Nov. 2009.
[3] B. Bazzanella, J. A. Chaudhry, T. Palpanas, and H. Stoermer. Towards a General Entity Representation Model. In *SWAP*, 2008.
[4] S. Bortoli, H. Stoermer, and P. Bouquet. Foaf-O-Matic - Solving the Identity Problem in the FOAF Network. In *SWAP*, December 2007.
[5] P. Bouquet, T. Palpanas, H. Stoermer, and M. Vignolo. A conceptual model for a web-scale entity name system. In *Asian Semantic Web Conference (ASWC)*, Shanghai, China, 2009.
[6] P. Bouquet, H. Stoermer, W. Barczynski, and S. Bocconi. Entity-centric semantic interoperability. In Y. Kalfoglou, editor, *Cases on Semantic Interoperability for Information Systems Integration: Practices and Applications*, chapter 1. IGI Global, October 2009.
[7] P. Bouquet, H. Stoermer, and B. Bazzanella. An entity name system (ens) for the semantic web. In *ESWC*, pages 258–272, 2008.
[8] N. N. Dalvi, R. Kumar, B. Pang, R. Ramakrishnan, A. Tomkins, P. Bohannon, S. Keerthi, and S. Merugu. A web of concepts. In *PODS*, pages 1–12, 2009.
[9] R. Delbru, A. Polleres, G. Tummarello, and S. Decker. Context dependent reasoning for semantic documents in sindice. In *Scalable Semantic Web Knowledge Base Systems (SSWS)*, October 2008.
[10] A. Dreibelbis, E. Hechler, I. Milman, M. Oberhofer, P. van Run, and D. Wolfson. *Enterprise Master Data Management: An SOA Approach to Managing Core Information*. Pearson/IBM Press, 2008.
[11] G. Giannakopoulos and T. Palpanas. Adaptivity in entity subscription services. In *Proceedings of ADAPTIVE2009*, Athens, Greece, 2009.
[12] A. Y. Halevy, M. J. Franklin, and D. Maier. Principles of dataspace systems. In *PODS*, pages 1–9, 2006.
[13] A. Jaffri, H. Glaser, and I. Millard. Uri identity management for semantic web data integration and linkage. In *3rd International Workshop On Scalable Semantic Web Knowledge Base Systems*. Springer, 2007.
[14] R. Kahn and R. Wilensky. A framework for distributed digital object services. *Int. J. on Digital Libraries*, 6(2):115–123, 2006.
[15] S. Kripke. *Naming and Necessity*. Basil Blackwell, Boston, 1980.
[16] X. Liu, H. Stoermer, P. Bouquet, and S. Wang. Supporting the Reuse of Global Unique Identifiers for Individuals in OWL/RDF Knowledge Bases. In *The Semantic Web: Research and Applications, ESWC*, 2009.
[17] N. Manerikar and T. Palpanas. Frequent Items in Streaming Data An Experimental Evaluation of the State-of-the-Art. *Data Knowl. Eng. (DKE)*, 68(4):415–430, 2009.
[18] T. Palpanas, J. A. Chaudhry, P. Andritsos, and Y. Velegrakis. Entity data management in okkam. In *DEXA Workshops*, pages 729–733, 2008.
[19] N. Paskin. Digital object identifiers for scientific data. *Data Science Journal*, 4:12–20, 2005.
[20] N. Shadbolt, T. Berners-Lee, and W. Hall. The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101, 2006.
[21] F. I. Tantono, N. Manerikar, and T. Palpanas. Efficiently discovering recent frequent items in data streams. In *SSDBM*, pages 222–239, 2008.