Entity Data Management in OKKAM

Themis Palpanas[1], Junaid Chaudhry[2], Periklis Andritsos[1], Yannis Velegrakis[1]

[1]University of Trento

[2]Ajou University

Abstract

In the recent years, we are witnessing an increasing interest in the Semantic Web and the relevant technologies, which can have a significant impact in the enterprise environment of information and knowledge management. An important observation is that the entity identification problem lies at the core of many semantic web applications. In this paper, we examine the special requirements of storage and management for entities, in the context of an entity management system for the semantic web. We study the requirements with respect to creating and modifying these entities, as well as to managing their evolution over time. Finally, we propose a conceptual model for the representation of entities, and discuss related research directions.

1. Introduction

The Semantic Web (SW) is an evolving extension of the WWW, in which the meaning of data and services is defined by attaching semantic concepts to them, making it possible for applications and machines to make sense of the web content [2].

One of the major problems that have emerged through the SW effort is the problem of uniquely identifying entities[1] [3]. The entities play a major role for the SW since they represent the atomic objects of reference and reasoning. Nevertheless, we currently face the problem of identifying and referencing these entities, which prohibits us from moving to the next step towards the goal of the SW, that of reasoning about 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 entity identification problem is also relevant to information and knowledge management in the enterprise environment. Its successful solution can help in two directions. First, it will enable the efficient management of information within an enterprise, overcoming the present difficulties in consolidating and integrating all the data about a single entity that are scattered across several repositories. Second, it will allow the enterprise to effectively correlate the information it owns about an entity, with relevant information that lies outside the boundaries of the enterprise, thus, delivering significantly richer knowledge management opportunities.

We claim that the entity identification problem is at the core of the semantic web effort. Along with the problem of assigning global identifiers to entities in the semantic web 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 OKKAM [3], a web-scale system for assigning and managing unique, global identifiers to entities in the WWW.

In this study, we discuss requirements for the storage and management of entities, and propose a conceptual model for entity representation.

[1] In the rest of this paper, we will use the term *entity* to refer to individuals, particulars, and instances. This 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).

1.1 Background

In this section, we give a brief overview of the OKKAM system (a more detailed presentation can be found elsewhere [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 OKKAM 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).

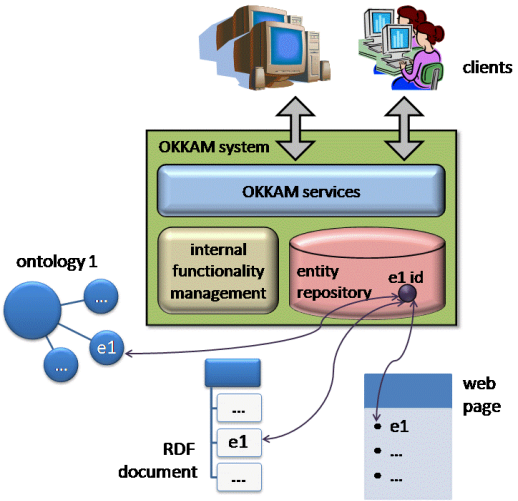

Figure 1: Schematic of OKKAM system and interactions.

The OKKAM system has a repository for storing the entity identifiers along with some small amount of descriptive information for each entity. The purpose of storing this information is to use it for discriminating among entities. Entities are described by a set of attribute-value pairs, where the attribute names and the potential values are user-defined (arbitrary) strings.

Clients interact with the system through the OKKAM Services layer. Clients can be both human users and applications. There are two types of interaction. First, clients may inquire about the identifier of an entity by querying the system for results that satisfy a set of attribute values describing this entity. If the entity exists in the repository, the system returns its identifier. Second, clients may insert a new entity in the system, by providing a set of *ad hoc* attribute-value pairs that characterize this entity. The system returns the newly assigned identifier.

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 OKKAM identifier. Therefore, entity identity resolution becomes trivial, and is done without any further interactions with OKKAM.

1.2 Related Work

There has been lots of work on rule-based reasoning for data partitioning and placement techniques [6][7]. Fred et al. [1] propose a storage infrastructure that effectively takes into account not only disk read and writes, but also data creation and deletion. Various techniques that employ different strategies have been proposed for efficiently storing different versions of data objects [24][25]. Versioning has also been studied in the context of semi-structured documents [26], and efficient query answering algorithms have been proposed [27].

When entities are created and modified, we are interested in keeping track of information related to the provenance of the entity data stored in the repository [22]. An important issue in data providence is its characterization. That is, to find the answers of questions like "*why is a piece of data in the output?*" and "*where is this piece of data copied from?*". Buneman et al. [23] target these issues and propose a framework for addressing them. Chapman et al. [29] propose efficient strategies for reducing the provenance storage size.

Several works have focused on the important problems of *record linkage* and *record matching* [8][11][19]. Other studies have focused on the problem of how to efficiently support the above operations in the context of relational database systems [14][16].

Duplicate detection through record linkage has also been studied [10][18]. These approaches are based on different flavors of clustering algorithms. Benjelloun et al. [9] propose three algorithms for solving the entity resolution problem, namely, G-Swoosh, R-Swoosh, and F-Swoosh. These algorithms take into account the characteristics of the match and merge functions, and can also provide approximate results.

2. Entity Data Management Requirements

We now discuss the requirements for entity data storage and management, in the context of a large, distributed repository of entities for the semantic web.

2.1 Representing Entity Content

The OKKAM system is designed to store arbitrary entities, referring to very diverse domains, including (but not limited to) persons, buildings, documents, and products. As such, the representation of the entities in the system has to be flexible in order to accommodate the requirements of all the different domains. Note that in OKKAM we are merely interested in assigning and managing unique ids to entities, which means that we do not need to represent all the known information about an entity, but rather only a small amount of data that can help us discriminate this entity from all the rest. Nevertheless, the set of data that need to be stored can vary drastically among entities.

2.2 Creating and Modifying Entities

The creation of new entities can be initiated through one of the following two ways: an automatic OKKAM-ization process, or a manual interface-based method. When a document is parsed, entities are identified using an automatic *entity identification* process that is part of the OKKAM-ization process. When new entities are created, we have to check if these entities already exist in the system. If the entity is unique, it is assigned a unique identifier, and stored in the system.

Once the entity has been added to the system, it is subject to updates. New attributes may be added to the description of the entity, or the values of existing attributes may change (e.g., when the information is outdated). If the description of an entity changes, we once again have to check if this entity is a duplicate of another entity stored in the repository of the system.

2.3 Data Provenance

When entities are created and modified, we are interested in keeping track of information related to the provenance of the entity data stored in the repository [22]. This includes information related to the source of the corresponding data, the owner, and creation and modification times of an entity's attributes. The above information can potentially be very useful for other algorithms operating on the entities in the repository, such as matching and merging.

The information on data lineage can refer to each entity as a whole, or be more fine-grained, and refer to each individual attribute of every entity in the repository. The latter alternative results in a much more detailed view of how all the entity data was inserted in the repository, but also leads to higher space requirements and management cost. Efficient techniques for reducing the storage cost have been proposed in the literature [29], and similar approaches could also be used in OKKAM.

2.4 Versioning of Entities

Regular updates of the attributes describing an entity lead to the creation of different versions of the same entity. The user should be able to query across different versions of the entity or in some cases, search the changes that have been performed on an entity over a certain period of time. The possibility of having different versions of the same entity raises some efficiency questions, along the dimensions of storage space, query answering and matching mechanisms, and indexing structures.

2.5 Merging Entities

As more and more entities are added in the system, it may be the case that the same entity is represented by multiple instances in the repository. We would like to employ techniques able to detect these situations, and merge the duplicate entries. Ideally, we would like to use online algorithms to identify duplicates at the time when new entities are inserted in the system (or when an old entity is updated and becomes a duplicate of another existing entity), rather than delegating this responsibility to an offline algorithm that would have to scan the entire entity repository to discover the duplicates. Evidently, the latter option is computationally more expensive than the former, and cannot deliver results equally fast.

3. Proposed Approach

In what follows, we briefly outline the directions that we will pursue related to entity representation.

3.1 Entity Representation Conceptual Model

In OKKAM, we represent an entity E as a tuple $<P, M>$, where P is the *profile* of the entity (i.e., description), and M are the *metadata* for the entity.

In the current version of the system, we store in M information on the owner of the entity E, its creation and last modification times, the number of times E was matched and selected as a result to a client query, and the last time E was selected by a client. The above metadata are used to support complex algorithms for the other functionalities offered by OKKAM, such as entity matching.

The profile P = <*eid*, *t*, ***A***, ***R***> contains all the information that describes the entity. This information is as follows.

- *eid*: Entity identifier assigned by OKKAM.
- *t*: Semantic type of entity (e.g., one of the high-levels classifications in *Wordnet*).
- ***A***: Set of attributes describing characteristics of the entity.
- ***R***: Set of external references that refer to this entity.

In the next paragraphs, we describe in more detail the sets ***A*** and ***R***.

The set ***A*** is composed of a set of arbitrary, used-defined attributes that describe the entity. For example, if the entity is a person, possible attributes are *name*, *date of birth*, and *nationality*. Note that this set of attributes can be different for every entity, even for entities in the same domain. An attribute A from the set ***A*** is a tuple of the form A = <n, v, *veid*, M_A>, containing the following information.

- n: Name of attribute.
- v: Value of attribute.
- *veid*: Entity identifier assigned by OKKAM for the entity described by v (e.g., if v = "Trento" then veid = $\text{eid}_{\text{Trento}}$).
- M_A: Metadata for attribute A.

The attribute metadata, M_A, refer to the metadata of a specific attribute A of a specific entity E. Similarly to the entity metadata, the metadata in M_A include information on the owner of the attribute A, its creation and modification times, the number of times, as well as the last time A was referenced in a client query, whether A can be displayed as a result of a query (e.g., it may be the case that A can only be displayed within the domain of an enterprise, but not to the general public), and the natural language that A is expressed in.

The set ***R*** is used to store information about the relationships of the entity with other entities in OKKAM (if we know that it is identical to another entity stored in the system), or outside OKKAM (if there exists another id assigned to the entity by another system). A reference R from the set ***R*** is a tuple of the form R = <c, p, M_R>, containing the following information.

- c: Category of reference (e.g., ontology).
- p: URL pointing to the external reference.
- M_R: Metadata for reference R.

The reference metadata, M_R, refer to the metadata of a specific reference R of a specific entity E, and store information on the time R was last checked (i.e., the latest time we know this reference was valid), and whether R can be displayed as a result of a query.

3.2 Processing of Usage Patterns

The way the users access the system and interact with it may determine various aspects of the representation of an entity. 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, … , 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. By monitoring and analyzing the way users interact with OKKAM, we can determine which entities, or profile attributes, are relevant to specific queries or to certain contexts, and update the profile or the metadata of these entities, in order to produce more relevant search results. The above kind of processing has to happen in an online fashion, be flexible enough to allow effective and efficient data analysis of the incoming data streams [5][28], and evolve over time by supporting time-decaying representations of the streaming data [4].

3.3 Repository Adaptation

The results of the usage patterns monitoring techniques are also relevant to the repository evolution process. One of the important aspects of this process is the entity merging operation, which takes place when we discover that two entities in the repository represent the same real-world entity.

As we already mentioned, when merging entities, it is important to consider the type of values at hand, i.e., numerical vs. categorical with their corresponding measures. Adopting techniques like BIRCH [20] for numerical data, and LIMBO for categorical data [21], we may perform the assessment of similarity among entities very efficiently as these algorithms promise linear complexity as the size of the input increases.

The greatest advantage of employing techniques as the ones mentioned above is that we get to summarize the input data set, i.e. the OKKAM entities, in summaries that retain as much of the initial information as possible. We plan to extend the construction of

summaries so that a) they can handle numerical and categorical data at the same time, and b) can be used effectively with streaming data. Once again, amnesic representations [4] will play a crucial role as entities and their properties evolve over time.

4. Conclusions

The web is quickly moving towards the direction of adding semantics to the online information, and using these semantics for enabling a vastly richer range of applications and user-experiences. In this paper, we argue for an entity naming system, where unique identifiers for entities are assigned and managed. We examine the special requirements of representing entities, and present relevant research directions.

Acknowledgements

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] Fred Douglis, John Palmer, Elizabeth S. Richards, David Tao, William H. Tetzlaff, John M. Tracey, and Jian Yin, "Position: Short Object Lifetimes Require a Delete-Optimized Storage System," 11th ACM SIGOPS European Workshop, September 2004.

[2] Nigel Shadbolt, Tim Berners-Lee, Wendy Hall: The Semantic Web Revisited. IEEE Intelligent Systems 21(3): 96-101 (2006).

[3] Paolo Bouquet, Heiko Stoermer, and Daniel Giacomuzzi. OKKAM:. In WWW2007 Workshop i3: Identity, Identifiers and Identification, May 2007.

[4] Themis Palpanas, Michail Vlachos, Eamonn J. Keogh, Dimitrios Gunopulos, Wagner Truppel: Online Amnesic Approximation of Streaming Time Series. ICDE 2004: 338-349.

[5] Themis Palpanas, Vana Kalogeraki, Dimitrios Gunopulos: Online Distribution Estimation for Streaming Data: Framework and Applications. SEBD 2007: 430-438.

[6] E. Pierre. Introduction to ILM: A tutorial. http://www.snia.org/, 2004.

[7] C. Johnson. ILM Case Study: Complete Data Lifecycle Management Solution. http://www.snia.org/, 2004.

[8] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, Vassilios S. Verykios: Duplicate Record Detection: A Survey. IEEE Trans. Knowl. Data Eng. 19(1): 1-16 (2007).

[9] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S.E. Whang, and J. Widom. Swoosh: A Generic Approach to Entity Resolution. To appear in VLDB Journal, 2008.

[10] P. Singla and P. Domingos, "Multi-Relational Record Linkage," Proc. KDD-2004 Workshop Multi-Relational Data Mining, pp. 31-48, 2004.

[11] S. Sarawagi and A. Bhamidipaty, "Interactive Deduplication Using Active Learning," Proc. Eighth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '02), pp. 269-278, 2002.

[12] S. Tejada, C.A. Knoblock, and S. Minton, "Learning Domain- Independent String Transformation Weights for High Accuracy Object Identification," Proc. Eighth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '02), 2002.

[13] W.W. Cohen, "Data Integration Using Similarity Joins and a Word-Based Information Representation Language," ACM Trans. Information Systems, vol. 18, no. 3, pp. 288-321, 2000.

[14] N. Koudas, A. Marathe, and D. Srivastava, "Flexible String Matching against Large Databases in Practice," Proc. 30th Int'l Conf. Very Large Databases (VLDB '04), pp. 1078-1086, 2004.

[15] D. Dey, S. Sarkar, and P. De, "Entity Matching in Heterogeneous Databases: A Distance Based Decision Model," Proc. 31st Ann. Hawaii Int'l Conf. System Sciences (HICSS '98), pp. 305-313, 1998.

[16] S. Guha, N. Koudas, A. Marathe, and D. Srivastava, "Merging the Results of Approximate Match Operations," Proc. 30th Int'l Conf. Very Large Databases (VLDB '04), pp. 636-647, 2004.

[17] T. Dasu, T. Johnson, S. Muthukrishnan, and V. Shkapenyuk, "Mining Database Structure; or, How to Build a Data Quality Browser," SIGMOD, 2002.

[18] Dong, X., Halevy, A., and Madhavan, J. 2005. Reference reconciliation in complex information spaces. SIGMOD, 2005.

[19] H. Galhardas, D. Florescu, D. Shasha, E. Simon, and C.-A. Saita, "Declarative Data Cleaning: Language, Model, and Algorithms," VLDB, 2001.

[20] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In SIGMOD, 1996.

[21] Periklis Andritsos, Panayiotis Tsaparas, Renee J. Miller, and Kenneth C. Sevcik. LIMBO: Scalable Clustering of Categorical Data. In EDBT, 2004.

[22] Wang Chiew Tan: Provenance in Databases: Past, Current, and Future. IEEE Data Eng. Bull. 30(4): 3-12, 2007.

[23] P. Buneman, S. Khanna,On Propagation of Deletions and Annotations through Views. T. PODS 2002.

[24] Douglas S. Santry, Michael J. Feeley, Norman C. Hutchinson, Alistair C. Veitch, Ross W. Carton, Jacob Ofir: Deciding when to forget in the Elephant file system. SOSP 1999: 110-123.

[25] Mallik Mahalingam, Chunqiang Tang, Zhichen Xu: Towards a Semantic, Deep Archival File System. FTDCS 2003: 115-121.

[26] Shu-Yao Chien, Vassilis J. Tsotras, Carlo Zaniolo: Efficient schemes for managing multiversionXML documents. VLDB J. 11(4): 332-353 (2002).

[27] Shu-Yao Chien, Vassilis J. Tsotras, Carlo Zaniolo, Donghui Zhang: Supporting complex queries on multiversion XML documents. ACM Trans. Internet Techn. 6(1): 53-84 (2006).

[28] Ferry Irawan Tantono, Nishad Manerikar, Themis Palpanas: Efficiently Discovering Recent Frequent Items In Data Streams. SSDBM (2008).

[29] Adriane P. Chapman, H.V. Jagadish, Prakash Ramanan: Efficient Provenance Storage. SIGMOD 2008.