# Recommendation-Based Evolvement of Dynamic Schemata in Semistructured Information Systems

Wolfgang Gassler
Databases and Information Systems
Institute of Computer Science
University of Innsbruck, Austria
wolfgang.gassler@uibk.ac.at

Eva Zangerle
Databases and Information Systems
Institute of Computer Science
University of Innsbruck, Austria
eva.zangerle@uibk.ac.at

## ABSTRACT

Community-based collaborative information systems provide the flexibility of storing information without having to adhere to any predefined, rigid schema. However, the stored knowledge lacks common structure, which is crucial in terms of data access and search capability. We present a novel concept for semistructured information systems, which features a dynamic and self-learning schema system and provides the users with recommendations when entering information. The recommendations ensure the creation and maintenance of a common, homogeneous schema while at the same time not restricting the user's flexibility to enter any kind of information.

## 1. INTRODUCTION

When storing information, there are two common ways of doing so: either one uses a (relational) database to store information in a structured way or one stores information in a mostly unstructured manner.

The first approach is very well suited for applications which have to store strictly structured information. Considering the example of bank accounts, all information is structured and can easily be matched into a schema for the storage of e.g. bank accounts and their owners. A big disadvantage is that a change within the schema can be very time-consuming and complex, as it has to be done manually and the already stored information has to be adapted to the new structure. Therefore, the end-user is fixed to a given schema and cannot insert additional information not matching the given schema. For flexible information systems, such a restricting approach is not very well suited, as it can result in a big loss of information because of the fact that the user cannot insert all information he might want to.

On the other hand, the second, unstructured way of storage allows the user to store information in any arbitrary structure or format. This approach has the advantage that the user does not have to match the data into some predefined schema. A popular example of such flexible systems are wiki systems, where information can simply be added as text. However, such structureless storage prevents every possibility to provide structured access or search facilities on the stored information, as for example relational databases do. Consider the example of Wikipedia, which does not provide structured search capabilities at all. The only way to search through this unstructured information is to perform fulltext search, which is incapable of answering precise queries like "Which Austrian cities have between 10,000 and 20,000 inhabitants?". For this reason information repositories need to support both structured and unstructured information to fully exploit the advantages of both worlds, as also pointed out by Weikum *et al.* [13]. Our paper presents the Snoopy concept which combines the structured and unstructured approach and takes advantages of both concepts.

The rest of the paper is organized as follows. In Section 2 the basic concept of our approach is described. Section 3 outlines the measures taken for the creation of a common schema. Our proposed solution is described in Section 4 and related work is outlined in Section 5. Section 6 summarizes the paper and points out open research issues.

## 2. THE SNOOPY CONCEPT

The Snoopy concept offers the same flexibility as wiki systems but at the same time provides the possibility to structure information like (relational) databases. This is achieved by using a self-adapting and self-learning schema system and recommendations, which support the user during the insertion process.

In the Snoopy concept, information about a certain subject is stored as a *collection*, similar to a wiki page. A collection consists of an arbitrary number of key-value pairs, which can be specified by the user without any restrictions. The following example shows how information about the city "Innsbruck" can be stored in a simple and understandable way:

```
Collection Name:  Innsbruck

Country:Austria
State:Tyrol
numberOfInhabitants:117,916
PanoramaImage:pano_innsbruck_2010.jpg
```

The key-value pairs are similar to relational database columns (keys) and its rows (values). Despite the fact that information is still plain text, it still features structure. Such semistructured data, where structure is present but does not comply to a unified schema, can be used for structured

*GvD Workshop '10,* 25.-28.05.2010, Bad Helmstedt, Germany.

access (e.g. queries like "All entries containing the property `Country:Austria`"). One of the main challenges of such semistructured data is the proliferation of keys and values. Furnas *et. al.* [6] showed that two people would choose the same term for a certain object (within a restricted domain) with a probability of only 20%. Consider the key "numberOfInhabitants". It can be assumed that a multiuser system would produce many synonyms of this word, eg. "inhabitants", "citizens" or "numberOfInhabitants". This fact would imply that the stored information is no longer searchable in terms of unified access, as the keys are not aligned to a common schema. Wikipedia also has to cope with this problem of proliferation of structures and tries to solve it by a very big and committed community, which creates templates (schemata) and manually unifies already present knowledge. Boulain *et al.* [3] showed that only 35% of all edits in Wikipedia are changes of content. All other edits are related to structure and do not concern the content itself.

The Snoopy concept takes a different approach and pushes part of these structural adaptions to the user. The user knows more about the collection he enters than any process, machine or community, which try to enhance the information *after* the author inserted it. Therefore, the key idea of the Snoopy concept is to "snoop" as much information as possible from the user during the insertion process. Furthermore, the user is guided by an adapting, self-learning guidance engine which provides recommendations to the user based on key-value pairs entered earlier. These recommendations support the user in aligning the information he intends to enter to a commonly used schema.

It is important to note that all these recommendations are just suggestions and the user is not forced to use these recommendations in any way. Therefore, the user is able to enter information without any restrictions.

# 3. SCHEMA ALIGNMENT

The process of integrating and transforming two or more (database) schemata into a common schema has proven to be a very complex task [10]. The Snoopy concept avoids any schema matching after the insertion of data as the guidance of the user significantly contributes to the alignment of the entered structured information to an already commonly used schema. This aligned "schema" is different from a fixed schema of a relational database to which information has to be adapted to. The user is free to extend or modify the recommended structure. Therefore, alignment can never be done for the totality of data.

The self-adapting schema is implicit and dynamically calculated based on already stored information. It consists of keys which are used together by the majority of similar collections. Hence, newly entered information can influence the schema and can result in a change of the schema. For example adding the key "numberOfStudents" to many collections about cities can lead to a recommendation of this key to further users.

As information systems contain information of various structures and types, the Snoopy concept automatically computes a suitable number of schemata according to the stored information - without any predefined settings. Schema alignment in the case of the Snoopy concept can be achieved by taking the following measures, which are all based on recommendations of values and keys.

## 3.1 Suggestion of Keys and Structure

The suggestion of keys and structure is the most important feature within the Snoopy concept. Considering a user entering information about cities (consisting of key-value pairs) into the system, there might be additional useful keys that the user might not think of at first hand. This is where the recommendation engine comes into play. If a user wants to add the key "inhabitants", the system computes that 90 percent of the users who added "inhabitants" as a key, also added "mayor". The recommendation feature is based on a data mining process within the already existing collections, which calculates keys that frequently occur together. Having done these computations, the system suggests this additional key to the user who can then accept this key and enter a respective value. Assume, the user wants to store information about the city "Innsbruck". Based on the already inserted keys "country", "state" and "numberOfInhabitants", the system recommends adding the keys "mayor" and "ZIP". Recommendations are simply further form fields displayed to the user that give him the opportunity to enter appropriate values for the suggested keys. The recommendations encourage the user to insert more information than originally intended and thereby increase the amount of information in the system.

## 3.2 Key Completion

When entering new key-value pairs, the user gets assistance by an auto-completion system. This feature suggests keys that are already stored in the information system, are similarly structured and therefore semantically related to the currently entered key. If the user types e.g. "number", the system automatically provides the user with the possibility to choose "number of inhabitants" or "number of districts", which are keys that are already existent in the database. The user still has the opportunity to provide a new key, e.g. "number of universities" by ignoring the recommendations and continue typing. This recommendation contributes to the creation of a common schema and data basis and is demonstrated by the following example. If the user wants to enter the key "inhabitants", after typing "inhab", the user is informed that a similar key "number of inhabitants" is already present. As accepting the recommendation is faster than typing the word "inhabitants", the user is encouraged to accept the recommendation.

# 4. PROPOSED SOLUTION

Our proposed solution is mostly concerned with the computation of recommendations and the underlying storage mechanisms, which are explained in the following section.

## 4.1 Recommendations

The computation of the recommendations can be achieved by using association rules [1], which can easily be adapted for the mining of relations between triples: formally, a frequent item set can be defined as a set of items $I = \{i_1, i_2, ...i_n\}$, where a transaction $T$ within the database consists of arbitrary many items of $I$. In the case of schema alignment, the item set $I$ consists of all properties $p_i$ occurring in the system and a transaction comprises all properties $p_{ij}$ occurring together within the subject $s_j$. The set of all transactions forms the transaction database $T = \{T_1, T_2, ...T_m\}$. Based

on this transaction database, the goal is to calculate association rules which are implications $X \rightarrow Y$, where $X$ is a property and $Y$ is another property which co-occurs with $X$ on the same subject.
This is the basis of the calculation of recommendation candidates, which are then further examined in order to provide the best possible recommendations to the user. Based on this (probably large) set of association rules, the final recommendations are computed taking the following impact factors into acount: (i) support of a rule, (ii) novelty of properties, (iii) recent popularity rise of properties and (iv) some random choice to give new properties a chance to be used and therefore rise in popularity.
Fundamentally, the computation process of the recommendations has to fulfill the following requirements: (i) ability to cope with huge amounts of data, (ii) compute recommendation candidates based on mining association rules, (iii) evaluation of the candidate sets according to the previously mentioned factors and (iv) real-time calculation of recommendations.

## 4.2 RDF Storage System

The underlying storage system is a crucial part of the application as it holds all Snoopy data. Basically, the main task is to store many key-value pairs belonging to a certain subject (collection). This pattern suggests storing this data as triples using the W3C Resource Description Framework [8], which has proven to be a modelling language very well suited for the description of any (real-world) resource [9]. Basically, every resource can be stored as a triple consisting of a subject (same as the collection name in the Snoopy concept), a predicate (the respective key) and an object (the respective value). The sentence "Innsbruck's number of inhabitants is 117,916" can therefore be stored as the following triple: `<Innsbruck>, <numberOfInhabitants>, <117,916>`. The main advantage of RDF is the possibility to easily model knowledge in a machine-understandable way. SPARQL, the query language for RDF, can be used to query all information.
Basically, the underlying RDF storage system has to fulfill the following requirements in order to ensure a scalable, efficient and highly performant system: (i) SPARQL support to query all stored data in a fast and efficient way, (ii) optimized RDF store: triples have to be stored in the most efficient and compact way as the representation of knowledge as triples leads to a very large number of triples, (iii) an interface optimized for data mining tasks, as SPAQRL is not best suited for data mining (iv) fast data mining: mining structure and content within the triples is the most time-critical part of the application as all recommendations are based on it.

## 4.3 Preliminary Results

SnoopyDB, a first prototype of the Snoopy concept has already been developed and evaluated in [7]. The evaluation showed that recommendations in SnoopyDB guide the user to common schemata and reduce the number of distinct properties. A system without any recommendations lead to 229 distinct properties on 50 test collections, whereas SnoopyDB including recommendation and guidance was able to store the same information by using only 154 distinct properties. Besides the 33% smaller property set, the recommendation-based version motivated the test users to insert 24% more key-value pairs.

# 5. RELATED WORK

The Snoopy concept can benefit from many research areas which cover parts of the Snoopy concept. Lots of research is done on how to introduce structure in Wikipedia or generally in information systems. All of the following approaches try to enhance information *after* the insertion process, do not consult the user and do not benefit from the user's extensive knowledge during the insertion process.
The DBPedia project [2] extracts structured information from Wikipedia infoboxes and stores it as RDF-triples. Another approach, YAGO [11] is also based on Wikipedia data and tries to semantically enhance this data. The Kylin/KOG System [14] automatically verifies semantically enhanced data by explicit community feedback. Semantic Wikipedia [12] extends MediaWiki by adding typed links between Wikipedia entries as well as attributes and types. However, the user is not guided in the process of specifying this additional semantics. Cimple/DBLife [5] presents an approach to build a structured community portal from already existing community sources. ExDB [4] extracts information from the web, adds structure and is then able to query this data in a structured manner.

# 6. CONCLUSION AND FUTURE WORK

We presented the Snoopy-concept, a novel approach for creating and maintaining a common schema in semistructured information systems by using recommendations. These recommendations guide the user during the insertion process in order to align the information to a commonly used schema and vocabulary. The implicit schema is based on all stored information in the system and adapts itself to the structure of this data. The recommendations are based on association rules and were implemented in a prototype. First results showed that recommendations ensure homogeneous schemata and vocabulary in a multi-user semistructured information system and therefore improve the structured data access capabilities. Further work will include ranking of rules enabling the system to compute top-n recommendations aiming at a higher precision of recommendations. Furthermore, capabilities to query the stored information efficiently have to be introduced and the storage mechanisms have to be improved.

# 7. REFERENCES


[1] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *SIGMOD '93: Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216, New York, NY, USA, 1993. ACM.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. *Lecture Notes in Computer Science*, 4825:722, 2007.

[3] P. Boulain, N. Shadbolt, and N. Gibbins. Hyperstructure maintenance costs in large-scale wikis. In Peter Dolog, Markus Kroetzsch, Sebastian Schaffert, and Denny Vrandecic, editors, *SWKM*, volume 356 of *CEUR Workshop Proceedings*.

CEUR-WS.org, 2008.

[4] M. J. Cafarella, C. Re, D. Suciu, and O. Etzioni. Structured querying of web text data: A technical challenge. In *CIDR*, pages 225–234, 2007.

[5] P. DeRose, W. Shen, F. Chen, A. Doan, and R. Ramakrishnan. Building structured web community portals: a top-down, compositional, and incremental approach. In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 399–410. VLDB Endowment, 2007.

[6] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):971, 1987.

[7] W. Gassler, E. Zangerle, M. Tschuggnall, and G. Specht. Snoopydb: Narrowing the gap between structured and unstructured information using recommendations. In *Proceedings of the 21th ACM Conference on Hypertext and Hypermedia*. ACM, 2010.

[8] G. Klyne and J. J. Carroll. Resource description framework (rdf): Concepts and abstract syntax. W3c recommendation, World Wide Web Consortium, February 2004. http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/.

[9] B. McBride. The resource description framework (rdf) and its vocabulary description language rdfs. *Handbook on Ontologies*, pages 51–66, 2004.

[10] P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. *Journal on Data Semantics*, 4:146–171, 2005.

[11] F. Suchanek, G. Kasneci, and G. Weikum. Yago: A large ontology from wikipedia and wordnet. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(3):203 – 217, 2008. World Wide Web Conference 2007, Semantic Web Track.

[12] M. Voelkel, M. Kroetzsch, D. Vrandecic, H. Haller, and R. Studer. Semantic wikipedia. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 585–594, New York, NY, USA, 2006. ACM.

[13] G. Weikum, G. Kasneci, M. Ramanath, and F. Suchanek. Database and information-retrieval methods for knowledge discovery. *Commun. ACM*, 52(4):56–64, 2009.

[14] D.S. Weld, F. Wu, E. Adar, S. Amershi, J. Fogarty, R. Hoffmann, K. Patel, and M. Skinner. Intelligence in wikipedia. In *Twenty-Third Conference on Artificial Intelligence (AAAI'08)*, 2008.