MediaBank: Keyword Search and Tag Cloud Functionalities for a Multimedia Content Authoring Web Platform

Sonia Bergamaschi, Matteo Interlandi and Maurizio Vincini
University of Modena and Reggio Emilia

Abstract

The composition of multimedia presentations is a time- and resource-consuming task if not afforded in a well-defined manner. This is particularly true when people having different roles and following different high-level directives, collaborate in the authoring and assembling of a final product. For this reason we adopt the Select, Assemble, Transform and Present (SATP) approach to coordinate the presentation authoring and a tag cloud-based search engine in order to help users in efficiently retrieving useful assets. In the first of this paper we present MediaPresenter, the framework we developed to support companies in the creation of multimedia communication means, providing an instrument that users can exploit every time new communication channels have to be created. In the second part we describe how we adopt keyword search techniques coupled with Tag Cloud in order to summarize the results over the stored data.

1. Introduction

Nowadays companies use different software products for authoring and presenting their multimedia presentations. We define a multimedia presentation as a composition of textual information and digital assets such as images, photos, videos and audios. In an enterprise context, applications such as Microsoft Power Point or OpenOffice Impress have the advantage to be competitive from the point of view of costs, but limits arise when the presentation creation process becomes a collaborative and intensive activity. Then it requires agile dynamics and company-specific logics.

Usually medium/big companies rely on communication means developed **(1)** in **outsourcing** (web sites, multimedia presentations, paper communications), or **(2) internally** developed and managed, with poor efficiency compared with the potentiality provided by the used software tools. These modi operandi have emphasized a series of issues. In the first case **(1)** the company must be always followed by another external and specialized enterprise, which will cover every communication demand. This scenario comport an economic investment that, due to budget constraints, is often not feasible if new communication products have to be created constantly. The major issue comes when the workflow must be managed from inside to outside the company and vice-versa. The demand to communicate its own services, products and brand appeal, is very hard to apply to the outsourcing model, since, for every new activity, the company needs to invest time and resources in producing and approving shared materials and therefore efforts are subtracted to activities that create more added value for the company. In the second case **(2)** all the software packets do not contemplate the possibility to natively share among users the collective multimedia assets such as the enterprise logo, background images, internal photos, etc. Moreover, since these collective assets are local and personal, they have to be updated continuously. This update process can be very resource consuming if the number of devices and people is high.

Based on the extensive analysis of issues an enterprise can face during the creation of multimedia presentations, we proposed to represent the whole process as SATP compliant [1]. This approach divides the process of creating a multimedia presentation into four sequential phases: Select (the phase where users pick up the required digital assets), Assemble (the phase where users merge the predefined semi-processed), Transform (where final presentation can be rendered in different formats) and Present (where the product is available and ready to be displayed in local or over internet).

During the creation process, the discovery of the already stored digital assets useful for a new presentation can be considered as a very difficult and time-consuming activity. For this reason we related digital assets to a shared hierarchy by offering to the user a keyword search engine in order to improve the search process. Keyword search techniques have been extensively applied to extract relevant information from Internet [2] and the database research community has also recognized the benefit of keyword search capabilities into relational databases [3] [4] [5] and XML databases [6] [7].

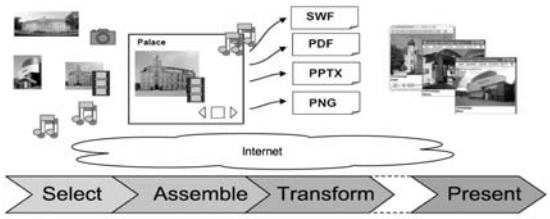


Figure 1. SATP representation

In the first part of the paper we introduce **MediaPresenter**, the framework we developed to support companies in the creation of multimedia communication means, providing an on-line and centralized instrument, called **MediaBank**, that users can exploit every time new communication channels have to be created. In this way the company can have a communication mean homogeneous and always on-line with the marketing directives. In the second part we propose to use keyword search techniques coupled with Tag Cloud [12] in order to summarize the results over the structured data stored in MediaBank.

This paper is structured as follow: Section II is an overview of MediaPresenter and the SATP approach we followed for its design. Section III outlines the architecture we developed, Section IV focuses on the design and implementation choice we made in the development of one of the pdf export modules, and Section V describes the Data Cloud and the algorithm we use for the ranking of the entities and related tags. Section VI contains some related work, and the paper is closed by a short conclusion in Section VII.

2. MediaPresenter

MediaPresenter is developed by join collaboration between the DBGroup at University of Modena and Reggio Emilia and Addiction Creation Media Lab, an Italian SME Media Agency. Part of the activity is founded by Italian Emilia Romagna region, within the LISEA lab.

MediaPresenter is an on-line cross-platform application, perfect incarnation of the Software-as-a-Service (SaaS) paradigm for business enterprise. It offers a large number of services for sharing and managing digital archives. Such services include the concurrent access to data by multiple users with different roles, import of multiple types of digital assets (3D objects, videos, images, etc), export in various formats (swf, pptx, png and pdf), and a keyword search engine for retrieving digital contents from MediaBank. MediaBank is the container of media digital assets, which contains a keyword search engine, based on a tag cloud approach in order to propose to the user search-related content.

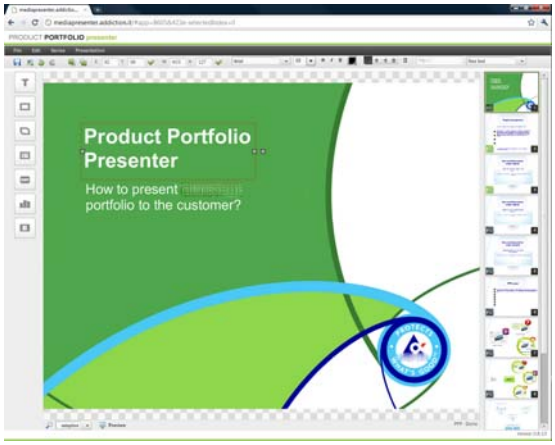


Figure 2. MediaPresenter series assembling

MediaPresenter has been designed as a collaborative framework, thus groups of users can be created and policies can be set both on groups and single users. We chose this approach because it allows modeling the internal hierarchy of an enterprise, so different people with different duties may contribute to a presentation in different manners. In MediaPresenter the authoring of new communication media follows the Select, Assemble, Transform and Present creation chain approach, whose schema is depicted in Figure 1. The original schema has been maintained, but in our approach we developed a different semantics for each phase, specifically tailored for our purpose. In the followings we briefly describe each phase.

2.1. Selection

In this phase, elements contained in the MediaBank are retrieved using the keyword search and the tag cloud. We define as elements, digital assets, slides, and series of slides or already composed presentations. Starting from the assumption that the simple file name is just an ID and it does not describe the element content(s), each element is inserted in the repository and has associated a series of tag which specify contents or properties that characterize such element. In this way, elements can be retrieved using a keyword-base search engine and the results visualized as tag cloud. The user is thus able to browse among the various tags and retrieve the best suitable element. Policies are associated to each user, based on the group the user belongs to. Search results, hence, may depend on the policy settings for the user.

2.2. Assemble

In this phase, the user assembles the final or semi-finished product (for example a single slide, a series of slides or a presentation) starting from the single elements already retrieved in the previous

phase. Users can also assemble different types of elements depending on their role. For instance one user can be a slide-maker and therefore he/she can assemble just slides, meanwhile a presentation-maker can assemble only complete presentations. Figure 2 shows a snapshot taken during the process of series assembling.

2.3. Transform

Once the user has repeated iteratively the previous phases and therefore the complete presentation has been created, the transform phase permits to save the final product in different formats following the user needs. Thanks to its ability to manage different formats without relying on a proprietary language, MediaPresenter is also able to integrate and make available to the user presentations not initially designed with this framework. If, for example, some legacy products, made with a third part software application, have to be integrated with other presentations designed in other formats, MediaPresenter seamlessly permits it.

2.4. Present

MediaPresenter client is a Flex application running on browser; hence a presentation can be potentially shown on each device having a browser and an Internet connection. But since a presentation can be transformed in different formats, a user may also employ third part software products.

3. Architecture

MediaPresenter is a 3-tier web system for contents managing and its architecture is shown in Figure 3. The information that the storage layer (MediaBank) memorizes can be classified in two sets: **presentations** and **digital assets**. For digital assets are considered all the digital elements such as images, photos, videos, audios animations, etc., which represent the company's multimedia knowledge. Since a medium/big enterprise can easily reach the number of 30.000 - 40.000 digital assets, users can associate significative keywords to each single asset. Digital assets can be easily accessed through a generic API and, since MediaPresenter has been designed to be completely independent to a specific repository, they could be stored directly in the MediaBank database or in specialized facilities called Digital Assets Management (DAM) such as Celum[1] and Xinet[2].

[1] Celum Digital Asset Management web page, http://www.celum.com (Access date: 24 May 2011).

[2] Xinet Digital Asset Management web page, http://www.xinet.com (Access date: 24 May 2011).

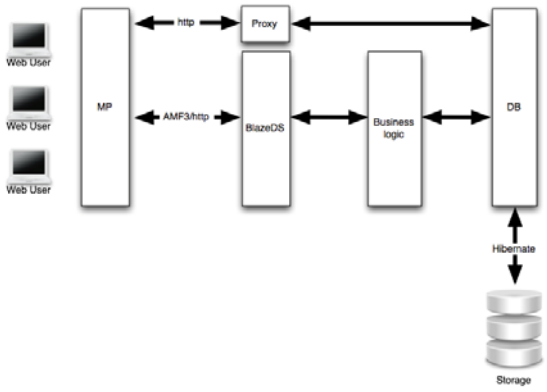


Figure 3. MediaPresenter architecture

Presentations represent the "final products" created by the user with MediaPresenter and they contain combinations of different digital assets together with textual contents. To facilitate the modularity and reuse, the user can create pieces of presentation, called series, which can be reused to compose multiple presentations. Similarly to digital assets, presentations can be tagged to increase the search accuracy. Each presentation is saved in an open XML-compliant format and can be exported in different formats (swf, pdf, png and pptx so far). In the next section we will describe an example on how we perform the export in Adobe pdf.

On the client side the user develops his/her presentation using the MediaPresenter web application completely developed in Adobe Flex. Since all the slides, series, presentations as well as digital assets are remotely stored, every time the user needs to perform some operation, a remote call to a java method is fired. The remote call is serialized using the AMF3 binary format protocol and sent over http to the application server where BlazeDS[3] is able to capture, deserialize and manage the request.

In order to speed up the process of content retrieving, we also implemented a RESTfull proxy, which allows users to get stored elements without passing through the MediaBank API, just specifying the object URL. The URL encoded all the information needed to retrieve from the database the correct element.

4. Pdf exporter

Pdf exporter is used to translate a presentation in a pdf document. Pdf exporter has been designed to be completely independent from a specific technology, and it can be divided in three parts: the **objects repository** which contains collections of java objects which mirror the elements a presentation can be

[3] Adobe BlazeDS web page, http://opensource.adobe.com/wiki/display/blazeds/BlazeDS (Access date: 24 May 2011).

composed by, the **parser** which maps each object in the input presentation with the related java object, and the **translator** which from each java object, creates the corresponding pdf object. In the Pdf exporter current implementation we used JAXB[4] and XML Schema to map each XML entry to a specific java class automatically generated, and then we implements the transformation using iText[5]. To perform the transformation, we designed each entity employed inside a presentation using XML Schema, and then through JAXB we performed the automatic java class generation. As output of this procedure we developed nine schemas, one for each of the following entity: image, text, shape, SWF entity, chart, data grid (table), slide action and two general-purpose schemas: one root schema and one schema for managing each data entry inside tables and charts. Pdf exporter is able to perform, at runtime, the validation of each presentation given as input and instantiate java objects containing all the needed information for the consequent translation in pdf. The translator module mainly contains mapping functions between the java objects enclosing the actual presentation information and iText functions.

5. Data cloud

The main goal of MediaPresenter is to produce multimedia communication channel for enterprise internal or external use. To reach this goal, the system provides to the user all the available information which can be briefly summarized in presentations, unpacked slides or series of slides and digital assets already used in the past or available as digital content of the enterprise. During the creation process, the user has to retrieve an already available asset stored into the MediaBank. The typical way to access these data are by searching, e.g. based on name, dimension, type, date of creation and so forth. The results are often unsatisfactory due to the lack of database structure knowledge and experience of the user. Furthermore, database search results are often considered as database tuples, whereas non-technical people think in terms of entities, not tuples.

To overcome this issue we implements a particular *search* method which permits users to performs a keyword search over the MediaBank database using terms that can be found in different fields (name, title, description, etc.) and in multiple relations, hiding the data structure to the user. In order to make this approach even more efficacious, the system permits to relate significative words to each digital assets and series: we define these words as **tags** that can be directly specified by the user

[4] JAXB project web site, http://jaxb.java.net (Access date: 24 May 2011).

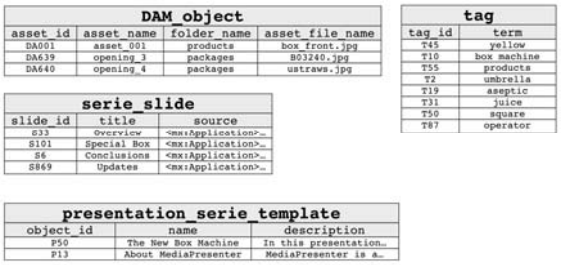
[5] iText library web page, http://www.itextpdf.com (Access date: 24 May 2011).

Figure 4. MediaPresenter asset search with tag cloud

either choosing among a predefined set, or created ex-novo at run-time. The action of organizing resources by adding metadata is called "tagging" and it is gaining popularity on the web in this years [9]. Using tags to label resources, allows the system, not only to specify the keyword search over the stored assets, but also to create a set of tags starting from the returned results. This set of tags, called **tags cloud**, considers each terms as an hyperlink that can be used to refine the search results, dynamically guiding users in the hidden relationship among contents and eventually leading to serendipitous discoveries of interesting results. The objective is then to perform a keyword search over the database and summarize results using **data cloud** and tag cloud that presents the most significant words (tags) associated with the search results. The advantage of this approach is to simplify the elements retrieving from MediaBank and to highlight, through the tag cloud, the most significant concepts and hidden complex relationship in the modeling context. For example, Figure 4 is showing the MediaPresenter interface used for keyword searching. In this case the user was looking for all the assets containing the term people. Meanwhile the user can select the preferred resource on the left, the tag cloud at the bottom right of the image can help the user in browsing all the related resources and therefore specifying the search.

5.1. Keyword search

Following the approach depicted in [12], we consider D as the MediaBank's database section involved in the *search* method and R_i as the i-th relation stored in D. Each relation contains a set C of columns. We consider $R_i.C_j$ as the j-th column of the i-th relation. With t we denote a generic tuple in D. Given R_i and R_j and a primary-foreign key relationship between R_i and R_j, we consider T_D as the tuple graph of the database D, where for nodes we



DAM_object

asset_id	asset_name	folder_name	asset_file_name
DA001	asset_001	products	box_front.jpg
DA639	opening_3	packages	B03240.jpg
DA640	opening_4	packages	ustraws.jpg

tag

tag_id	term
T45	yellow
T10	box machine
T55	products
T2	umbrella
T19	aseptic
T31	juice
T50	square
T87	operator

serie_slide

slide_id	title	source
S33	Overview	<mx:Application>…
S101	Special Box	<mx:Application>…
S6	Conclusions	<mx:Application>…
S869	Updates	<mx:Application>…

presentation_serie_template

object_id	name	description
P50	The New Box Machine	In this presentation…
P13	About MediaPresenter	MediaPresenter is a…

Figure 5. Primary and secondary relations involved in *D*

consider each tuple in D and given two tuples $t_i \in R_i$ and $t_j \in R_j$, an edge exists among them if $(t_i \bowtie t_j) \in (R_i \bowtie R_j)$.$D$ can be modeled as a collection V of *search entities*. In our case V correspond to the set of digital assets that can be returned by the *search* method, providing, thus, a sort of unit of representation for returned entities. For each entity $v \in V$ we consider C_i as the i-th attribute describing the entity. C_i can be seen both as a one-to-one mapping to a particular column in the database (e.g. the asset name), but also
as a many-to-one mapping, therefore grouping several information in one search entity attribute (e.g. the set of tags specified for a digital asset can be thought as an attribute of the asset entity). In particular, we consider C_0 as the identificator of each specific search entity. An entity id, hence, is a mapping to the primary key of the relation R_0, where R_0 is called *primary entity relation*. In our context, we recognize as primary entity relation the table *DAM_object*, which provides all the digital assets ids. On the other side, we call *secondary entity relation* all the other relations that join directly or indirectly with R_0 and provide additional information to v. We identify *tag*, *presentation_serie_template*, and *serie_slide* tables as secondary entity relations. Figure 5 depicts the database relations we took in consideration for the keyword search.

In summary, the keyword search engine we developed, returns digital asset entities identified by the asset id stored in the *DAM_object* primary relation. Each asset entity contains attributes directly related to the primary relation (asset name, file name, folder name) but also attributes grouping information belonging to secondary relations. Each entity gets information about the slide(s) it belongs to, thanks to the join with the serie_slide relation. In addition, information about the presentation (series, template) is added joining the relation serie_slide with presentation_serie_template. At the end the relation tag provide information about the set of assets' tags as well as information about the set of tags related to the presentation (series, template) containing the asset entity. Figure 6 contains a possible instance for the tuple graph T_D.

The input of our *search* function is a query q, and we assume that q is constituted with a certain number

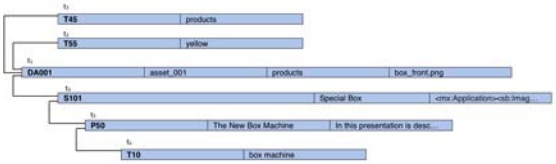


Figure 6. Example of a tuple graph in *D*

of keyword terms. We can assume that given a keyword term k and a search entity v identified by the id stored in the tuple t of the primary relation R_0, v contains k if one of the following statements holds:

a) one of the attributes values of t contains k.
b) T_D contains a tuple t_i stored in the relation R_i that contains an attribute values equal to k, and exist a path in the tuple graph connecting the tuple t_i to t.

Given a query q, the set of resulting entities is denoted as $V_q \subseteq V$, and it contains the set of search entities related at least to a keyword term k contained in q.

5.2. Ranking search entities

In order to rank the search entities returned by the *search* method given a query q, we create a score function based on IR-standard ranking methods [10], i.e., *tf*idf* for any term of the query. The term frequency of a keyword k in the query related to an object v can be computed as:

$$tf_{k,v} = \frac{\sum_C w_C * n_C}{n_v} \qquad (1)$$

where n_C count the number of occurrence of the keyword k in the attribute C of v, n_v is the total number of keyword found in v and w_B is the weight of the attribute C. Given that the tuple graph T_D is actually a three with a maximum height of 3, we define $w_B = (1 - n * 0.2)$, where n is the distance of C from the primary entity relation. While the inverse document frequency *idf* for k is:

$$idf_k = \ln\left(\frac{|V|}{|V_q(k)|}\right) \qquad (2)$$

where $V_q(k)$ is the set of search entity containing the keyword k. Then, we consider the *tf*idf* for each term in the query and we define the score for an object v w.r.t. a query q as [12]:

$$score(v,q) = \sum_{k \in q} tf_{k,c} * idf_k \qquad \square\square\square$$

This scoring function allows MediaBank to show the retrieved digital assets in an ordered way.

5.3. Tag cloud algorithm and scoring

We consider the set L of all tags. These tags are textual labels (words) assigned to a search entity, thus each search entity $v \in V$ is associated to a set of tags, denoted with L_v. We denote as L_q the set of tags related to the entities contained in V_q, similarly we consider $V_q(l) \subseteq V_q$ as the set of objects associated to the tag $l \in L_q$. Then we designed a tag selection algorithm able to maximize the number of entities in V_q that are covered by the set of tags $S \subseteq L_q$ resulting as output. Although the most used algorithm for the generation of tag cloud is the popularity-based [12], in our context we prefer to maximize the coverage of the set S since we would like to propose to users the largest number of asset referable to the query q.

The goal of the maximum coverage algorithm is to maximize the function:

$$\mathrm{cov}(S) = \frac{\left| \bigcup_{l \in S} V_q(l) \right|_{s,q}}{|V_q|_{s,q}} \qquad (5)$$

given the subsets $V_q(l_1)$, $V_q(l_2)$,…, $V_q(l_n)$ and a threshold MAX for which $|S| \leq MAX$. In literature this problem is known to be NP-Complete, but good approximation exists [12]. Our maximum coverage algorithm follows the greedy strategy and it starts with an empty set S. At each round, it is subtracted from the set L_q and added to S the tag that embrace the largest number of uncovered entities in that particular round. The algorithm stop when all the search entities (or tags) available at the beginning have been ordered, or when the number of tags contained in S is MAX.

Max Coverage Algorithm

Input: V_q, L_q
Outuput: $S \subseteq L_q$

$S \leftarrow 0$;
while $(|S| < MAX) \wedge (L_q \neq 0) \wedge (V_q \neq 0)$ **do**
{
  $l^* = \underset{l \in L_q}{arg\,max} (|V_q(l) \cap V_q|)$
  $S \leftarrow S \cup \{l^*\}$;
  $V_q(l) = V_q(l) - V_q(l^*)$;
  $L_q = L_q - \{l^*\}$;
}
return S;

Once the set S has been filled, we defined the following score function:

$$score(l,q) = \frac{|V_q(l)|}{|V_q|} * \ln\left(\frac{|V|}{|V_q(l)|}\right) \qquad (6)$$

which permits to produce a suitable visual summary given the collection of tags S by visually depicting the tag score by font size.

In addition, since the user creates tags in a context of a group, and each group has a label identifying its generic topic, we can consider that given a certain search result, the sum of all the groups' labels is itself a tag cloud summarizing the whole topics of the results. In order to sum up the tag clouds resulting from the tags and the one resulting from the group labels, we use colors to identify for each tag the group it belongs to, and an index showing all the groups related to the search. In this way the user is able to perform a refinement over the results using two different levels of granularity: by generic topic, selecting the group of interest on the index, and one more detailed using the tags from the cloud.

6. Related work

In this section we discuss related work on tag word cloud used to present web and database structured search results.

Many papers and some research prototypes have been developed in this area: for example PubCloud [13] is a project that uses tag clouds for summarizing query result of PubMed biomedical literature database. The tag clouds are generated from words extracted from the paper abstracts of the query results. The font size of the words in the cloud is calculated just using terms frequency, and the set of visualized tags are obtained by using the tags having frequency higher than 10% of the best frequency: both the score and the visualized set computation is very simple w.r.t. other approach, such as our technique.

In [14] three different approaches to determine word cloud generation from web search results are used: full-text, query biased and anchor text based clouds. They define a specific model to score the terms and a greedy algorithm to select best tags: preliminary results are obtained by using 2009 TREC Web Track as documents set evaluation.

Also in the area of structured database some results has to be mentioned: for example [15] describes a social tagging extension with tag clouds. Each employee profile got enriched with two tag clouds, an incoming and outgoing tag cloud, where the incoming tag cloud visualised the tags assigned to that person from their co-workers and the outgoing tag cloud summarised the terms used by that person to tag their co-workers. Their studies showed that people tag other people as a form of contact management, to see all the people associated with a project, or to locate experts in a particular area. They also found that tags were considered by users as giving accurate descriptions of their interests and expertise.

In [12] tag clouds are proposed to summarize keyword search result over structured data. In this case, the term data cloud is used to refer to their particular adaptation of tag clouds for summarising keyword search results. Data clouds were implemented as part of CourseRank: a social tool to access official university information and statistics, such as course descriptions, grade distributions and course evaluations, as well as user-generated information, such as course ratings, comments, questions and answers.

7. Conclusion and future work

In medium/big companies the creation of multimedia presentations can be a high resource-consuming task, especially if it needs collaboration among people with different roles. In this paper we described the industrial requirements we tried to fulfill and the approach we followed in the development of a framework that enables users to collaborate in the creation of multimedia presentations. We put particular focus on the recognition and export of presentations in multiple formats, and in the design of the keyword search engine with related tag cloud.

The future work will be related to automatic detection of tags for documents and enterprise products. A previous work related to product classification integration [16] should be useful to automatically assign tags to presentation and DAM objects related to enterprise products: exploiting AI functionalities defined in [17, 18] and included in [19, 20] will be possible to infer new tags. Another enhancement of the techniques will be to use automatic annotation [11] of titles and descriptions to generate significant related tags.

8. References

[1] A. Scherp, “Canonical processes for creating personalized semantically rich multimedia presentations”, *Multimedia Systems* vol. 14, 2008, pp. 415–425.

[2] G. Li, J. Feng, B. Chin Ooi, J. Wang, and L. Zhou, “An effective 3-in-1 keyword search method over heterogeneous data sources”, *Inf. Syst*. vol. 36, 2011, pp. 248–266.

[3] S. Agrawal, S. Chaudhuri, and G. Das, “Dbxplorer: A system for keyword-based search over relational databases”, in *Proceedings of the 18th International Conference on Data Engineering ICDE*, 2002, p. 0005.

[4] A. Hulgeri and C. Nakhe, “Keyword searching and browsing in databases using banks”, in *Proceedings of the 18th International Conference on Data Engineering ICDE*, 2002, p. 431.

[5] V. Hristidis and Y. Papakonstantinou, “Discover: Keyword search in relational databases,” in *VLDB*, 2002, pp. 670– 681.

[6] L. J. Chen and Y. Papakonstantinou, “Supporting top-k keyword search in xml databases”, in *ICDE*, 2010, pp. 689–700.

[7] Z. Bao, J. Lu, T. W. Ling, and B. Chen, “Towards an effective xml keyword search”, *IEEE Trans. Knowl. Data Eng*., vol., no. 8, 2010, pp. 1077–1092.

[8] V.V Vazirani, “Approximation Algorithm”, Springer, 2004.

[9] S. A. Golder and B. A. Huberman, “The structure of collaborative tagging systems”, *CoRR*, vol. abs/cs/0508082, 2005.

[10] V. Hristidis, L. Gravano, and Y. Papakonstantinou, “Efficient ir-style keyword search over relational databases”, in *Proceedings of the 29th international conference on Very large data bases* VLDB, Volume 29, 2003, pp. 850–861.

[11] S. Bergamaschi, P. Bouquet, D. Giacomuzzi, F. Guerra, L. Po, and M. Vincini,”MELIS: an incremental method for the lexical annotation of domain ontologies”, in *International Journal on Semantic Web and Information Systems* IJSWIS 3(3), 2007, p.p.57-80.

[12] P. Venetis, G. Koutrika, and H. Garcia-Molina, “On the selection of tags for tag clouds”, in *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM, New York, NY, USA, 2011, pp. 835–844.

[13] B. Yu-Lin Kuo, T. Hentrich, B. M. Good, M. D. Wilkinson, "Tag clouds for summarizing web search results", in *Proceedings of the 16th International Conference on World Wide Web* WWW 2007, 2007, pp 1203-1204.

[14] R. Kaptein, J. Kamps, "Word Clouds of Multiple Search Results", *Proceedings of Multidisciplinary Information Retrieval - Second Information Retrieval Facility Conference*, IRFC 2011, 2011, pp. 78-93.

[15] Farrell, S., Lau, T., Nusser, S., Wilcox, E., Muller, M., “Socially augmenting employee profiles with people-tagging”, in *Proc. ACM Symposium on User Interface Software and Technology* UIST 2007, 2007, pp. 91–100.

[16] D. Beneventano, F. Guerra, S. Magnani, M. Vincini, "A Web Service Based Framework for the Semantic Mapping Amongst Product Classification Schemas", *Journal of Electronic Commerce Research* vol. 5 number 2, 2004.

[17] D. Beneventano, S. Bergamaschi, Claudio Sartori, Maurizio Vincini, "ODB-Tools: a description logics based tool for schema validation and semantic query optimization in Object Oriented Databases", *Proceeding of the Fifth Conference of the Italian Association for Artificial Intelligence* AI*IA97, *Advance in Artificial*

Intelligence, LNAI col. 1321, Springer. Berlin, September 17-19 1997.

[18] S. Bergamaschi, D. Beneventano, C. Sartori, M. Vincini, "ODB-QOptimizer: A Tool For Semantic Query Optimization in OODB", in *13th International Conference on Data Engineering* ICDE, 1997, p. 578.

[19] S. Bergamaschi, S. Castano, S. De Capitani di Vimercati, S. Montanari, M. Vincini, "A Semantic Approach to Information Integration: the MOMIS project", *Proceeding of the Sixth Conference of the Italian Association for Artificial Intelligence* AI*IA98, Padova IT, 1998.

[20] D. Beneventano, S. Bergamaschi, J. Gelati, F. Guerra, M. Vincini: "MIKS: an agent framework supporting information access and integration", *Intelligent Information Agents - The AgentLink Perspective*, (editor S. Bergamaschi, M. Klusch, P. Edwards, P. Petta) - March 2003, *Lecture Notes in Computer Science* N. 2586.

