World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:1, No:7, 2007

Issues and Architecture for Supporting Data Warehouse Queries in Web Portals

Minsoo Lee, Yoon-kyung Lee, Hyejung Yoon, Soo-kyung Song and Sujeong Cheong

***Abstract*—**Data Warehousing tools have become very popular and currently many of them have moved to Web-based user interfaces to make it easier to access and use the tools. The next step is to enable these tools to be used within a portal framework. The portal framework consists of pages having several small windows that contain individual data warehouse query results. There are several issues that need to be considered when designing the architecture for a portal enabled data warehouse query tool. Some issues need special techniques that can overcome the limitations that are imposed by the nature of data warehouse queries. Issues such as single sign-on, query result caching and sharing, customization, scheduling and authorization need to be considered. This paper discusses such issues and suggests an architecture to support data warehouse queries within Web portal frameworks.

***Keywords*—**Data Warehousing tools, data warehousing queries, web portal frameworks.

I. Introduction

MANY middle to large sized companies have been building data warehouses and performing analysis on their data in order to more effectively understand their businesses. The data warehouse tools that support this objective have continuously evolved and currently most of them have migrated to Web-based user interfaces. Users of data warehouses can easily form queries or view results through the Web. The next step of such tools is adapting to a portal framework. Portals are becoming increasingly popular as the mainstream to provide a user with a summary view of all interesting information on a single Web page. This Web page is mostly set up as the start page or home page of many employees of an organization and most often display different personalized information based on the employee's position within the organization. In many cases, business strategists of a company would want to see several summary tables or graphs of data warehouse query results on his/her start page along with other useful information that may be irrelevant of the data warehouse queries. As this paradigm of providing a portal page as a single place for all available resources, tools, and information of an organization is becoming popular, it is essential for data warehouse query tools to adapt to the portal framework.

We have identified several issues that need to be considered when a data warehouse query tool is used within a portal framework. Such issues arise due to the significantly different environment where portals are used and where data warehouse queries are executed. The issues that we discuss in this paper are about single sign-on, query result caching, customizing and sharing query results, scheduling of queries, and authorization issues. We look into several characteristics of portals and discuss how they affect the design and architecture of data warehouse query tools. We also suggest an architecture to solve the issues that are discussed.

The organization of the paper is as follows. Section 2 gives a survey on related research and a few commercial products. Section 3 explains the approaches that can be taken to support data warehouse queries in portals. Section 4 gives an outline of the architecture to solve the issues when developing a portal enabled data warehouse query tool. Section 5 gives an explanation of the current implementation status. Section 6 concludes with the summary and future work.

II. Related Research

With the growing need to analyze huge amounts of corporate data in a data warehouse, a variety of tools to analyze the data and create reports have been developed. These tools are particularly moving to Web-based interfaces in order to accommodate a wide range of users. Several such popular tools on the market are Brio 8 of Brio Software[1], Business Objects [2], PowerPlay of Cognos [3], and Discoverer of Oracle [4]. Brio 8 supports server based reporting in a Web environment and multiple data sources can be used in a single report. Business Objects supports multiple data source integration and uses a multi-pass SQL technology to accurately obtain query results. It also supports three tier architecture as well as client-server architecture. PowerPlay draws information from relational databases to model and build PowerCubes ("Cubes"). The cubes and reports can be deployed to Web clients, or to Windows and Excel clients, all using the same application

Manuscript received September 30, 2007. This work was supported in part by the second stage of the BK21 program of the Ministry of Education and Human Resources Development.

Minsoo Lee is with the Dept of Computer Science and Engineering, Ewha Womans University, 11-1 Daehyun-Dong, Seodaemoon-Ku, Seoul, Korea 120-750 (e-mail: mlee@ ewha.ac.kr).

Yoon-kyung Lee is with the Dept of Computer Science and Engineering, Ewha Womans University, 11-1 Daehyun-Dong, Seodaemoon-Ku, Seoul, Korea 120-750 (e-mail: polyandry@hanmail.net).

Hyejung Yoon is with the Dept of Computer Science and Engineering, Ewha Womans University, 11-1 Daehyun-Dong, Seodaemoon-Ku, Seoul, Korea 120-750 (e-mail: auroree@ewhain.net).

Soo-kyung Song is with the Dept of Computer Science and Engineering, Ewha Womans University, 11-1 Daehyun-Dong, Seodaemoon-Ku, Seoul, Korea 120-750 (e-mail: happymint@ewhain.net).

Sujeong Cheong is with the Dept of Computer Science and Engineering, Ewha Womans University, 11-1 Daehyun-Dong, Seodaemoon-Ku, Seoul, Korea 120-750 (e-mail: bloom01@ewhain.net).

server. Discoverer is a key component of Oracle9i Application Server (Oracle9iAS) integrated business intelligence solution and enables users to create, modify, and execute ad hoc queries and reports.

Several portal products have recently emerged and gained popularity due to the new paradigm of providing all information useful for an organization as a single portal page. The most significant products are Plumtree[5], Microsoft SharePoint Server[6], Oracle 9iAS Portal[7], IBM Websphere[8]. Plumtree supports personalized portal pages and communities which are a collection of portal pages, and document directories which are a repository of Web pages from the Internet. MS SharePoint Server provides a customized portal solution which enables enterprise search and integrated document management. Oracle 9iAS Portal provides an environment to support various open standards such as J2EE applications as well as integration with Oracle's own applications. It is an integrated management environment that also enables self service publishing. IBM Websphere consolidates business data, applications, external newsfeeds and includes a security layer for authentication, supports Web-based content publishing, and customization of portal contents.

III. Approaches to Integrate Data Warehouse Queries into Portals

A. Web Portal Characteristics

Portals have several characteristics that make them very appealing to support an organization composed of a variety of users with different levels of technical knowledge or experience. The most important characteristics are as follows.

- First, single sign-on is used. Using a single login id, users can view and organize all information within the portal framework.
- Second, portals require subsecond response time. Most of the portal pages are used as start pages that should come up immediately when a user starts the browser.
- Third, users are allowed to customize the view or information displayed on their portal page to suit their personal preferences.
- Fourth, users not only consume information but also provide information to other users through the portal framework.

B. Web-based Data Warehouse Query Tool

The Web-based data warehouse query tool is implemented as three-tier architecture. It is mainly a middle-tier component that is implemented as a servlet which accepts requests from browser clients and connects to the database to obtain query definitions and run the queries. Features of the data warehouse query tool are:

- A data warehouse query defines a sheet.
- A single sheet or groups of sheets along with layout information is defined as a report.
- Query formulation is driven by a Graphical User Interface.
- Ad-hoc query analysis is supported.
- Queries can include parameters that can be modified to customize the query.
- Query results are displayed as HTML tables to the user. XML output of the query result is available.

C. Integration Approaches

The two types of integration approaches for data warehouse queries into portals are called URL-based and Extension-based.

- *URL-based* : The portal keeps only a URL to the data warehouse query tool that can execute a query and return an html page which can be embedded into the portal page.
- *Extension-based* : An extension to the original data warehouse query tool is developed in order to support the required functionality to integrate data warehouse queries into portals as a separate module that can communicate with both the portal and the data warehouse query tool.

The advantage of the URL-based approach is that it is very simple to implement. Actually no additional coding is needed on the part of the data warehouse query tool. The disadvantage is that the response time may be quite long due to the fact that the query is being executed when the URL is invoked. This will violate the characteristics of portals to provide instant information to users.

The advantage of the Extension-based approach is that it can include complex capabilities to satisfy the portal characteristics. These capabilities will be discussed in the following subsections as we have decided to take this approach. The disadvantages of this approach are that it requires the development of an additional module (i.e., extension) and needs to be able to interface seamlessly with the portal as well as the data warehouse query tool.

IV. Architecture for Extension-Based Integration

We propose an architecture for a portal enabled data warehouse query tool using the Extension-based approach discussed in the previous section. Fig. 1 shows the proposed architecture.

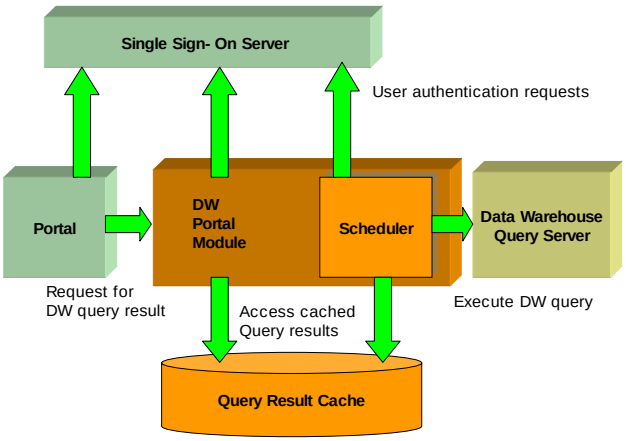

Fig. 1 Architecture of portal enabled data warehouse query tool

The portal server needs a single sign-on server to manage and authorize portal users. The data warehouse portal module is the extension to the original data warehouse server. It accepts requests from the portal server and returns the data warehouse

query results from the query result cache. It also stores any user specific customization information into the cache. There is a scheduler that periodically or immediately initiates the execution of the data warehouse query and stores the results into the query result cache. The data warehouse query server provides an interface that the scheduler can call and obtain the query results. The portal, data warehouse portal module, scheduler all need the authorization information stored in the single sign-on server.

The following subsections discuss the details of the architecture to support the basic characteristics of portals for a data warehouse query tool.

A. *Single Sign-On*

Portal users are usually assigned a single login id that enables them to use the portal framework in a personalized way. This mechanism of using a single login id is called single sign-on. The single sign-on id enables the portal framework to keep track of each user's privileges and personal preferences such as their portal page organization or page contents.

Although single sign-on may look very easy and promising in terms of eliminating the need for remembering multiple login ids for the portal user, it complicates situations when several data warehouse query results are shown on a portal page. Each query may need to be run with different accounts in several database systems. This means that the database account information needs to be integrated with the single sign-on mechanism of the portal framework.

The following Fig. 2 shows how the single sign-on mechanism should be extended with the database account information for each portal user. The single sign-on mechanism must not only provide the database account information available for each portal user, but also must provide management functions. When database user passwords change or accounts are dropped at the database system, these changes will affect the single sign-on repository. Assume that the single sign-on id of a portal user PU1 is mapped to several database user accounts DBU1, DBU2, DBU3 where the respective passwords are DBP1, DBP2, DBP3. If there is a change in the password DBP1 in the database, this changed password value should be updated in the single sign-on repository as well. Also, if the account DBU2 is dropped from the database, this should again be reflected in the single sign-on repository.

One way to automatically synchronize the database account information in the source database systems with the single sign-on repository is to require the source database systems to notify the portal framework of any changes in account information. However, most source database systems may not be capable or not willing to perform this task. Therefore, a passive approach that is dependent on the user is more favored. When users see error messages in the query results due to the fact that he/she has lost authorization or something has been modified at the database level, they will be able to determine and change any database account information in the single sign-on repository themselves if they are provided with a management mechanism to add, modify, and delete accounts in the single sign-on repository.

Also note that as the single sign-on repository holds very sensitive information, it should be encrypted and communication between the data warehouse portal module should be based on a trusted communication with a shared secure key.

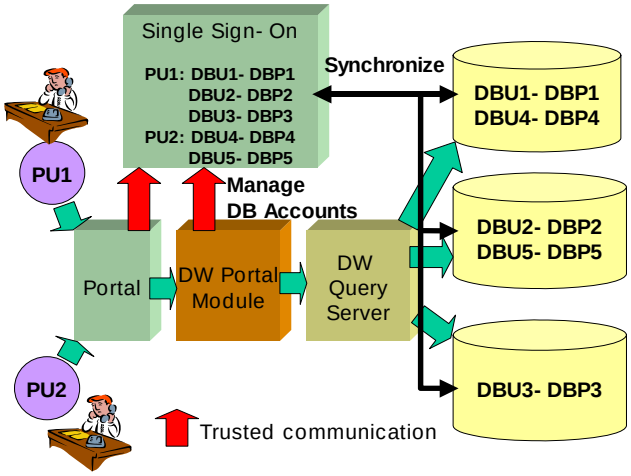

Fig. 2 Integration of Single Sign-On and DW Portal Module

B. *Query Result Caching*

Portal users anticipate subsecond response, but it is impossible for most data warehouse queries to be executed and return the results within this short timeframe. The data warehouse query result therefore needs to be obtained before a user requests to view a portal page containing data warehouse query results. In other words, the results should be cached. The cache can be in the form of a middle-tier database that is accessible by the data warehouse portal module.

The cache should be designed to hold the necessary data types for the query results. The query results can be obtained as XML and stored in the cache. The advantages of using XML become evident when supporting customization features for portals. Cached query results may also contain image data such as graphs. There also may be several images related to a query result.

The cache needs to be indexed so that the cache entry related to a query result can be quickly identified and retrieved. As an example, an index can be built on the following columns of a cache table schema.

CREATE INDEX GETENTRY ON CID
CREATE INDEX CHECKEXIST ON DBCONN, QID, QPARAM

The first index is used by the portal to quickly obtain query results. CID is an identifier for the cache entry and is actually a primary key. The second index described is used to check the content of the cache entry in order to see if there already exists a relevant cache entry before creating a new one. DBCONN contains the database connection information, QID is an identifier for each predefined data warehouse query, and QPARAM is a combination of the query parameters that are to be used by the query identified by QID. This kind of index will differentiate the cache entry for each database connection, each query, and each query parameter. Because it doesn't contain

information about the portal user, a cache entry can be shared among several portal users.

Caches always have issues of expiration. Once information in the cache is no longer useful – the reference count to the cache entry becomes zero- the cache entry may be deleted. However, for data warehouse queries, the cached entry may be kept for some time in case the query result would be reused within a short time. This greatly reduces the overhead of re-running a data warehouse query whose result was already stored in the cache. A timestamp on each cache entry may be set when the entry is to be deleted. And after a specified period of time a background process may really delete this cache entry. The cache id (CID), reference count(REF), and timestamp(TS) are shown in Fig. 3.

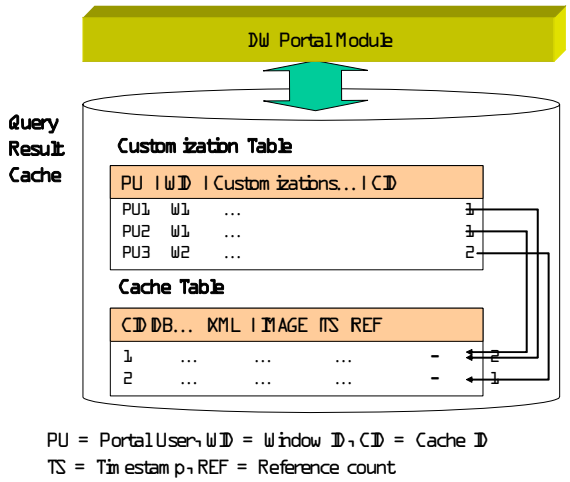

Fig. 3 Query Result Cache for DW Portal Module

C. Customization and Sharing

Portal users are provided with the ability to customize the view of their data or portal pages. This feature also applies to data warehouse query results. The users may want to see only tables or only graphs or probably both on their page. In some cases specific parameters that form the query may be customized. The looks of the tables or titles of the table or graph may be customized.

This kind of customization should be supported with minimum overhead to the system. If the query needs to be executed again after each customization operation by the user, then it would cause too much overhead and make it look like the customization operation itself is very slow. Most customization operations should be taken care of at the middle-tier level rather than going all the way to the database. This can be solved by the separation of presentation logic and data. The presentation information can be supported by stylesheets and the data provided in the form of XML documents. By retrieving the query result once as an XML document and applying different customizations with a stylesheet eliminates the need for re-running the time consuming data warehouse query.

It is also necessary to minimize the amount of storage that is used by these customization operations. If a user can reuse the query result from another user, the storage space for the cache can be dramatically reduced. Fig. 3 shows the cache table along with a separate customization table that contains customization options specified by a user for the query result windows on a portal page. By separating the customization information from cache entries, it is possible for several portal users to share the data stored in a cache entry. In this case, there should be reference counts being tracked on the cache entry in order to correctly determine when the cache entry could be safely deleted. In Fig. 3, the portal user PU1 created a window W1 and portal user PU2 created a window W2. These two windows actually can share the same cache entry CID=1 and thus the reference to this cache entry is set to REF=2. The reference count to the cache entries should be correctly maintained when sharing of cached data is permitted under the cover of different portal users.

D. Scheduling Data Warehouse Queries

As explained in subsection 4.2, data warehouse query results need to be cached in order to meet the subsecond response time requirement of portal users. For this purpose, a query scheduling mechanism is needed. A background scheduler should be running and periodically execute the data warehouse queries to populate the cache entry with the query results.

The most important design issue for the scheduler is providing scalability. The portal environment has a much larger user base than the traditional data warehouse whose user base is composed of a few business strategists. In the worst case if no query results can be shared among the portal users at all, the scheduler will have to run queries on behalf of every portal user. Additionally, a user may want to run several such data warehouse queries, which will dramatically increase the number of queries to be scheduled overall. The scheduler should be designed as a multithreaded module so that parallel threads can execute queries at the same time. Also, several scheduler instances should be able to run in parallel if necessary. This can be achieved by persisting data structures that contain information such as refresh periods used by the scheduler into a middle-tier repository and letting the multiple schedulers obtain the information in a consistent way. We can use the previously discussed Query Result Cache for this purpose.

An issue that has direct impact on the scalability requirements of the system is the scheduling frequency. If the scheduler is in synch with the data warehouse refresh schedule, the scheduling frequency would be fixed for all data warehouse queries except for those that are newly added to portal pages and require immediate execution. If portal users are provided with the capability to individually specify when their query results should be refreshed in the cache, the processing load of the scheduler may significantly increase. Even in such cases, there should be minimum refresh period restrictions such as hours being the minimum unit of refresh in order to control the load of the scheduler. Also, in order to avoid a situation where the load can grow infinitely, the interval of refreshing the query

results should not be allowed to be less than the estimated data warehouse query execution time. As an example, if the refresh period is 1 hour for a data warehouse query and the query takes 2 hours to execute, the execution of the query will be kicked off every 1 hour while some previously scheduled queries are still running.

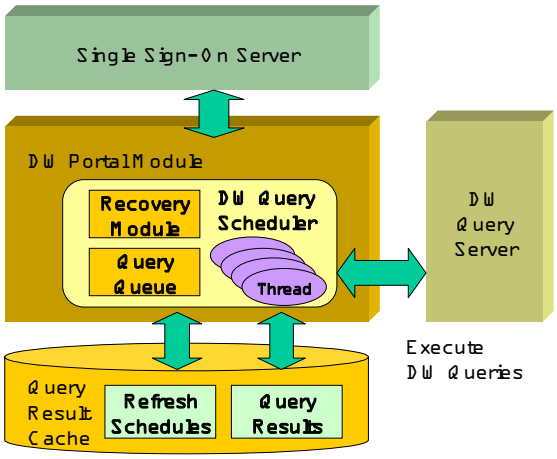

Fig. 4 Multithreaded DW Query Scheduler

Another problem when designing schedulers is to provide a recovery and management mechanism. This is especially important when dealing with data warehouse queries because these queries require large amounts of resources. Because the scheduler is run as a background process, it could experience errors while nobody is currently available to fix it or even notice it. Therefore, it is desired that in some situations, if possible, the scheduler can automatically diagnose its status and perform recovery operations. If the server is rebooted and the scheduler is reinitialized, it should be able to identify those jobs that were in progress and redo them again. Also, if data warehouse queries result in errors for some reason, the errors should be logged and shown to the portal users to understand the why the query result is not available. Functions to clean up queries that are infinitely looping or are erroneously using too much resources should be provided as well.

The scheduler also requires a super user privilege that can run any query on behalf of any user. This may complicate the process of authorization and potentially become a security hole if the scheduler is not properly secured.

E. Authorization

The authorization model in the portal becomes quite complicated when it comes to providing users to be able to publish their data to other users through the portal framework. This type of authorization for viewing data overrides the database authorization mechanism. It especially becomes complicated when a portal user publishes data that is prohibited at the database level for other portal users to view. A user may accidentally open up sensitive information to the public if he/she is not aware of the restrictions provided by the database security.

Therefore, in cases where a level of authorization is provided at the portal level aside from the database level, the authorization granting or publishing should be carefully designed so that the authorization of the data at the database level is not seriously violated. In our case, we decided to provide maximum flexibility to the portal user and do not restrict the user from publishing the information that he/she is allowed to view. However, if a user that does not have the database account required to run this published query in the Single Sign-On server, he/she is not able to view the query results. Therefore, even though we allow portal users to publish any query, the database security is still maintained.

V. Implementation

The issues regarding the design of the Data Warehouse Portal Module have been resolved and currently the module is implemented in Java as a servlet. The Data Warehouse Portal Module is currently being targeted first to support Oracle Portal as Oracle's products have a large user base. In order to support Oracle Portal, a module called a Provider is implemented with our servlet. Oracle Portal provides a PDK (Portal Development Kit) which is a library to make it easy to plug into the framework by just implementing several methods required by Oracle Portal. We are using an adapter approach to be able to plug into different portal products. Due to the fact that the Data Warehouse Portal Module is a servlet, it can coexist with the Data Warehouse Query Server or be separately installed on another machine with a servlet engine or even on the machine running Oracle Portal.

The Query Result Cache is a middle-tier database that needs to be available for use. This database can be installed and managed on the site that is running the Data Warehouse Portal Module. It is also possible to use an existing database instead. However, if using a remote database, the performance may be a little slower due to the communication required through the network. We are currently using an Oracle 9i database for the Query Result Cache.

VI. Conclusion and Future Work

We are currently extending our Web-based query reporting tool which is designed for data warehouse queries to support a portal environment. During the design phase we have identified several issues that need to be considered when developing a portal enabled data warehouse query tool. The issues are single sign-on, query result caching, customizing and sharing query results, scheduling of queries, and authorization issues. After carefully examining each issue, we have come up with architecture to support the portal environment.

Our future work will concentrate on evaluating the generic adapter to see if it can plug into several portal products, improving the cache performance and sharing mechanism, allowing more customization options, refining the authorization model to enable users to have a more sophisticated layer of control on their data at the portal level.

REFERENCES

[1] Brio, http://www.brio.com/products/overview.html
[2] Business Objects, http://www.businessobjects.com/products/bobj/
[3] Cognos,PowerPlay,http://www.cognos.com/products/powerplay/index.html
[4] Oracle, Discoverer, http://otn.oracle.com/products/discoverer/content.html
[5] Plumtree, http://www.plumtree.com
[6] Microsoft, Sharepoint Server,http://www.microsoft.com/sharepoint/server/ default.asp
[7] Oracle,Oracle9iASPortal,http://portalcenter.oracle.com/servlet/page?_pageid=356&_dad=ops&_schema=OPSTUDIO
[8] IBM,Websphere,http://www-3.ibm.com/software/info1/websphere/index.jsp?tab=products/portaltoolkit&S_TACT=103BGW01&S_CMP=campaign
[9] On Personalizing the Catalogs of Web Portals, N. Spyratos, Y. Tzitzikas, V. Christophides, Special Track on Semantic Web at the 15th International FLAIRS'02 Conference, Florida, May 14-16, 2002.
[10] Times-Ten Team, Mid-tier caching: the TimesTen approach. Proceedings of ACM SIGMOD conference, p. 588-593, Madison, Wisconsin, June 3-6, 2002.
[11] Haifeng Liu, Wee Keong Ng, Ee-Peng Lim, Query Integration for Refreshing Web Views. Proceedings of DEXA conference, p. 557-566, Munich, Germany, 2001.
[12] Kurt Stockinger, Kesheng Wu, Arie Shoshani, Strategies for processing ad hoc queries on large data warehouses. Proceedings of ACM Fifth International Workshop on Data Warehousing and OLAP, p.72-79, November 8, 2002, McLean, VA, 2002.
[13] Alexander Kuckelberg, Ralph Krieger, Efficient Structure Oriented Storage of XML Documents Using ORDBMS. Proceedings of Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web, VLDB 2002 Workshop EEXTT and CAiSE 2002 Workshop DTWeb, p. 131-143, Hong Kong, China, 2002.
[14] Dieter Gawlick, Infrasturucture for Web-based Application Integration. Proceedings of the 17th International Conference on Data Engineering, p. 473-476, Heidelberg, Germany, April 2-6, 2001.
[15] Christian Zirpins, Harald Weinreich, Andreas Bartelt, Winfried Lamersdorf: Advanced Concepts for Next Generation Portals. WBC - First International Workshop on Web Based Collaboration (Workshop with DEXA), p. 501-506, Munich, Germany, 2001.
[16] Arne Koschel: Base Technologies for iPortal development: IONA's iPortal Suite. Engineering Federated Information Systems, Proceedings of the 3rd Workshop EFIS 2000, p. 102-105, Dublin, Ireland, June 19-20, 2000.
[17] Charu C. Aggarwal, Philip S. Yu, An Automated System for Web Portal Personalization, Proceedings of VLDB conference, p. 1031-1040, Hong Kong, China, 2002.
[18] Christian Wege, Portal Server Technology. IEEE Internet Computing, vol. 6 no. 1, p. 73-77, 2002.
[19] Wen-Syan Li, K. Selçuk Candan, Wang-Pin Hsiung, Oliver Po, Divyakant Agrawal, Qiong Luo, Wei-Kuang Waine Huang, Yusuf Akca, Cemal Yilmaz, Cache Portal: Technology for Accelerating Database-driven e-commerce Web Sites. VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, p. 699-700, September 11-14, 2001, Roma, Italy.
[20] Peter H. Aiken, Kathi Hogshead Davis: Metadata Engineering for Corporate Portals Using XML. Conceptual Modeling - ER 2000, 19th International Conference on Conceptual Modeling, p. 572-573, Salt Lake City, Utah, USA, October 9-12, 2000.